МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МАТЕРИАЛЫ

XI-й Международной молодёжной научной конференции

«МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННЫХ, ТЕХНИЧЕСКИХ И ЭКОНОМИЧЕСКИХ СИСТЕМ»

Томск, 24–27 мая 2024 г.

Под общей редакцией
кандидата технических наук И.С. Шмырина

Томск
Издательство Томского государственного университета
2024

ББК 22.17– 22.19
УДК 519.2, 519.7, 519.8
Т78

Т78 Труды Томского государственного университета. – Т. 309. Серия физико-математическая: Математическое и программное обеспечение информационных, технических и экономических систем : материалы X-й Международной молодёжной научной конференции. Томск, 24–27 мая 2024 г. / под общ. ред. И.С. Шмырина. – Томск : Издательство Томского государственного университета, 2024. – 324 с.

ISBN 978-5-907890-15-2

Сборник содержит материалы XI-й Международной молодёжной научной конференции «Математическое и программное обеспечение информационных, технических и экономических систем», проводившейся 24–27 мая 2024 г. на базе Института прикладной математики и компьютерных наук Томского государственного университета. Материалы сгруппированы в соответствии с работавшими на конференции секциями.

Для научных работников, преподавателей, аспирантов, магистрантов и студентов.

УДК 539.3.004
ББК 22.17-22.19

ISBN 978-5-907890-15-2

I. СТАТИСТИЧЕСКИЕ МЕТОДЫ ОБРАБОТКИ ДАННЫХ, УПРАВЛЕНИЕ И ИССЛЕДОВАНИЕ СТОХАСТИЧЕСКИХ СИСТЕМ

ИССЛЕДОВАНИЕ ЭЛЕКТРОМАГНИТНОГО РАССЕЯНИЯ НА ТОНКОМ ИДЕАЛЬНО ПРОВОДЯЩЕМ ЦИЛИНДРЕ, РАСПОЛОЖЕННОМ ВНУТРИ ДИЭЛЕКТРИЧЕСКОГО ШАРА

Балашова О.М.
Томский государственный университет
balashovajkz@mail.ru

Введение

В [1] был представлен численный метод решения задачи электромагнитного рассеяния на тонком идеально проводящем цилиндре, размещенном внутри диэлектрического шара. Суть предложенного метода заключалась в том, что рассеянное поле вне структуры представляется в виде суммы полей электрических диполей, расположенных на вспомогательной поверхности внутри диэлектрического шара, и поля нити электрического тока, расположенного внутри цилиндра. Поле внутри диэлектрического шара представляется в виде суммы полей электрических диполей, расположенных на вспомогательной поверхности, охватывающей диэлектрический шар. Эти представления удовлетворяют уравнениям Максвелла во внешней среде и внутри шара и условиям излучения на бесконечности. Входящие в эти представления неизвестные диполи и элементы токов определяются из граничных условий.

В данной работе представлена программа, реализующая предложенный ранее метод, а также полученные с помощью данной программы результаты расчетов сечений обратного рассеяния для цилиндра, расположенного внутри шара, характеризующегося различными значениями мнимых частей диэлектрической проницаемости, определяющих электромагнитные потери внутри шара.

1. Постановка задачи и ее решение

Геометрия задачи показана на рис. 1. В безграничной однородной изотропной среде D_e с диэлектрической и магнитной проницаемостями ε_e,μ_e в декартовой системе координат $Oxyz$ расположен однородный диэлектрический шар D_i радиуса R с проницаемостями ε_i,μ_i, ограниченный поверхностью S. Внутри шара размещён прямолинейный идеально проводящий цилиндр длиной l и радиусом r, ограниченный поверхностью S_p. Предполагается, что поверхности S и S_p не пересекаются, а параметры цилиндра удовлетворяют условиям тонкого цилиндра: $2r << \lambda$, $2r << l$, где λ – длина падающей на рассматриваемую структуру волны. Структура возбуждается стационарным электромагнитным полем $\vec{E}_0, \vec{H}_0$, зависимость которого от времени t выбрана в виде $e^{-i\omega t}$. Требуется найти рассеянное поле $\vec{E}_e, \vec{H}_e$ в области D_e.

Решение сформулированной выше задачи представлено в [1]. Оно имеет следующий вид:

$$E_{e,\theta}(M) = \sqrt{\frac{\mu_e}{\varepsilon_e}} H_{e,\phi}(M) = \frac{e^{ik_e R}}{R} D_\theta(\theta) + O\left(R^{-2}\right),$$

$$E_{e,\phi}(M) = -\sqrt{\frac{\mu_e}{\varepsilon_e}} H_{e,\theta}(M) = \frac{e^{ik_e R}}{R} D_\phi(\theta) + O\left(R^{-2}\right).$$

Компоненты диаграммы рассеяния $D_\theta(\theta)$ и $D_\phi(\theta)$ определяются выражениями

$$D_\theta(\theta,\varphi)=\frac{i\omega k_e\mu_e}{4\pi}\sum_{n=1}^{N_e}G_{n,e}(\theta,\varphi)\Big\{\Big(\cos\theta\cos\varphi\cos\alpha_1^{n,e}+\cos\theta\sin\varphi\cos\beta_1^{n,e}-\sin\theta\cos\gamma_1^{n,e}\Big)p_{\tau_1}^{n,e}+$$
$$+\Big(\cos\theta\cos\varphi\cos\alpha_2^{n,e}+\cos\theta\sin\varphi\cos\beta_2^{n,e}-\sin\theta\cos\gamma_2^{n,e}\Big)p_{\tau_2}^{n,e}\Big\}+$$
$$+\frac{i\omega\mu_e}{4\pi}\Big(\cos\theta\cos\varphi\cos\alpha_p+\cos\theta\sin\varphi\cos\beta_p-\sin\theta\cos\gamma_p\Big)\times$$
$$\times\sum_{i=1}^{N_p}J_{p,i}^e\int_{l_{p,i-1}}^{l_{p,i}}\exp\Big[-ik_e\Big(x_p'\sin\theta\cos\varphi+y_p'\sin\theta\sin\varphi+z_p'\cos\theta\Big)\Big]dl_p\,,$$
$$D_\varphi(\theta,\varphi)=\frac{i\omega k_e\mu_e}{4\pi}\sum_{n=1}^{N_e}G_{n,e}(\theta,\varphi)\Big\{\Big(-\sin\varphi\cos\alpha_1^{n,e}+\cos\varphi\cos\beta_1^{n,e}\Big)p_{\tau_1}^{n,e}+$$
$$+\Big(-\sin\varphi\cos\alpha_2^{n,e}+\cos\varphi\cos\beta_2^{n,e}\Big)p_{\tau_2}^{n,e}\Big\}+\frac{i\omega\mu_e}{4\pi}\Big(-\sin\varphi\cos\alpha_p+\cos\varphi\cos\beta_p\Big)\times$$
$$\times\sum_{i=1}^{N_p}J_{p,i}^e\int_{l_{p,i-1}}^{l_{p,i}}\exp\Big[-ik_e\Big(x_p'\sin\theta\cos\varphi+y_p'\sin\theta\sin\varphi+z_p'\cos\theta\Big)\Big]dl_p\,,$$
$$G_{n.e}(\theta,\varphi)=\exp\Big[-ik_e\Big(x_{n,e}\sin\theta\cos\varphi+y_{n,e}\sin\theta\sin\varphi+z_{n,e}\cos\theta\Big)\Big]. \qquad (1)$$

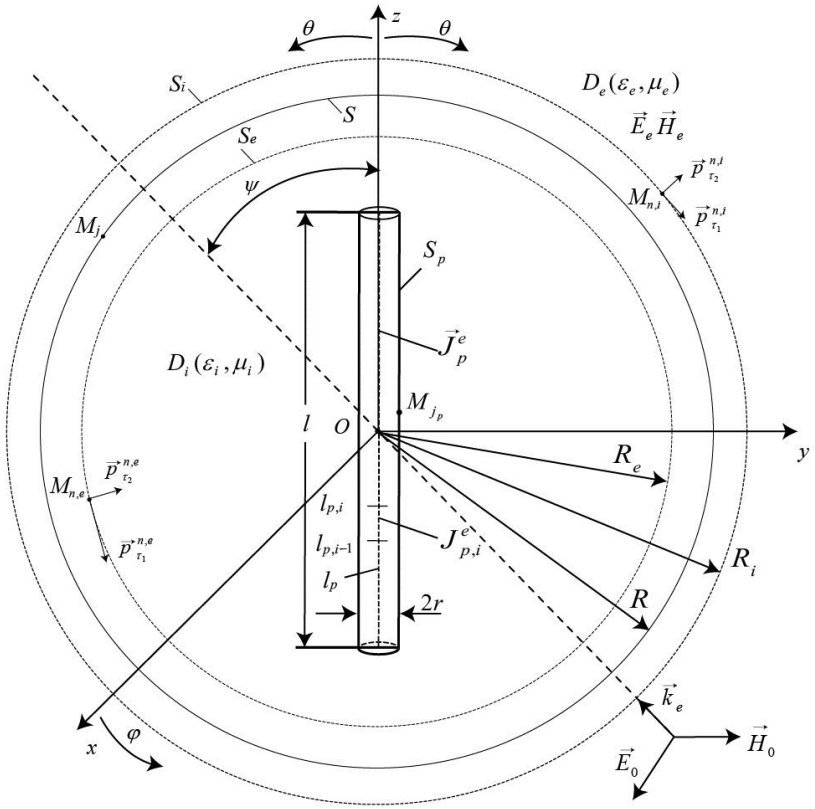


Рис. 1. Геометрия задачи

В выражениях (1) угол θ отсчитывается от положительного направления оси *z*; элементы токов $J_{p,i}^e$ ($i=\overline{1,N_p}$), являющиеся кусочно-постоянной аппроксимацией осевого тока идеально проводящего цилиндра (рис. 1) и дипольные моменты, расположенные на внутренней вспомогательной поверхности диэлектрического шара, находятся из системы линейных алгебраических уравнений

$$\left[\vec{n}^{\,j},\left(\vec{E}_i^{\,j}-\vec{E}_e^{\,j}\right)\right]=\left[\vec{n}^{\,j},\vec{E}_0^{\,j}\right],\ \left[\vec{n}^{\,j},\left(\vec{H}_i^{\,j}-\vec{H}_e^{\,j}\right)\right]=\left[\vec{n}^{\,j},\vec{H}_0^{\,j}\right],\ j=\overline{1,L},$$
$$\left[\vec{n}_p^{\,j},E_i^{p,j}\right]=0,\ j=\overline{1,L_p}, \qquad (2)$$

где $\vec{n}^{\,j}$ – единичный вектор нормали в точках коллокации M_j на поверхности S диэлектрического тела; $\vec{n}_p^{\,j}$ – единичный вектор нормали в точках коллокации M_j^p на поверхности идеально проводящего цилиндра; $\vec{E}_i^{\,j}$, $\vec{H}_i^{\,j}$, $\vec{E}_e^{\,j}$, $\vec{H}_e^{\,j}$ – компоненты внутреннего и внешнего полей в точке коллокации M_j; $E_i^{p,j}$ – вектор напряжённости внутреннего электрического поля в точке коллокации M_j^p на поверхности цилиндра; $\vec{E}_0^{\,j}$, $\vec{H}_0^{\,j}$ – компоненты возбуждающего поля в точке M_j. Для вычисления компонент векторов $\vec{E}_e^{\,j}$, $\vec{H}_e^{\,j}$, $\vec{E}_i^{\,j}$, $\vec{H}_i^{\,j}$, а также компоненты $E_e^{p,j}$ использованы выражения (6) и (7) работы [1], которые являются объединением основных идей вариантов метода вспомогательных источников, предложенных в работах [2,3].

2. Описание компьютерной программы

Изложенные выше соотношения реализованы в виде программы для расчета компонент рассеянного поля и контроля точности получаемых результатов по критерию невязки граничных условий. Программа написана в среде Jupyter Notebook, подробная схема основных блоков программы на рис. 2.

Рис. 2. Блок-схема компьютерной программы

Входными величинами программы являются геометрические параметры структуры: длина цилиндра l_p, радиус цилиндра r_p, относительные диэлектрическая и магнитная проницаемости диэлектрического шара $\varepsilon_i'=\varepsilon_i/\varepsilon_e$ и $\mu_i'=\mu_i/\mu_e$, радиус диэлектрического шара R, а также параметры метода: число разбиений осевой линии идеально про-

водящего цилиндра N_p, соотношение радиусов вспомогательных поверхностей к радиусу диэлектрического шара $K_i = R_i/R$, $K_e = R_e/R$ и др.

В блоке 2 выполняется расчет координат точек коллокации и точек размещения диполей. В блоке 3 осуществляется вычисление элементов матрицы системы линейных алгебраических уравнений (2), а также вектора правой части системы.

В блоке 4 выполняется поиск решения системы линейных алгебраических уравнений (2) методом сопряженных градиентов. Алгоритм метода сопряженных градиентов следующий. Задается начальное приближение $\vec{p}^{(0)}$ и находится начальное значение вектора невязки $\vec{R}^{(0)} = A\vec{p}^{(0)} - \vec{f}$ и вектора направления $\vec{S}^{(0)} = -A^*\vec{R}^{(0)}$, где A – матрица системы (2), $\vec{p}$ – вектор, состоящий из неизвестных дипольных моментов и элементов токов, A^* – матрица, транспонированная комплексно-сопряженная по отношению к матрице A. Далее выполняется следующая последовательность итераций:

$$t^{(k)} = \frac{\left\|A^*\vec{R}^{(k)}\right\|^2}{\left\|A\vec{S}^{(k)}\right\|^2},\ \vec{p}^{(k+1)} = \vec{p}^{(k)} + t^{(k)}\vec{S}^{(k)},\ \vec{R}^{(k+1)} = \vec{R}^{(k)} + t^{(k)}A\vec{S}^{(k)},$$

$$q^{(k)} = \frac{\left\|A^*\vec{R}^{(k+1)}\right\|^2}{\left\|A^*\vec{R}^{(k)}\right\|^2},\ \vec{S}^{(k+1)} = -A^*\vec{R}^{(k+1)} + q^{(k)}\vec{S}^{(k)},\ k = 0,1,2,...$$

Итерационный процесс останавливается при выполнении условия $\frac{\Phi^{(k)} - \Phi^{(k+1)}}{\Phi^{(k)}} < \varepsilon$, где $\Phi^{(k)} = \left\|A\vec{p}^{(k)} - f\right\|^2$, $\Phi^{(k+1)} = \left\|A\vec{p}^{(k+1)} - f\right\|^2$, ε – заданная точность; в данной программе $\varepsilon = 0.001$.

Блок 5 осуществляет вычисление компонент диаграммы рассеяния по формулам (1), а также бистатических сечений рассеяния по формуле

$$\frac{\sigma}{\lambda^2} = 10\lg\left(\frac{\left|D_\theta(\theta,\varphi)\right|^2 + \left|D_\varphi(\theta,\varphi)\right|^2}{\pi}\right).$$

Результаты расчетов представляются на экране в виде таблиц и графиков. Задавая конкретный угол φ, при котором пользователя интересует рассеянное поле, вычисляют компоненты $D_\theta(\theta,\varphi)$ и $D_\varphi(\theta)$ при всех $\theta = \overline{0,\pi}$ с заданным шагом и строят график зависимости бистатического сечения рассеяния от θ.

Блок 6 осуществляет контроль точности решения задачи по величине относительной нормы невязки граничных условий $\Delta = \sqrt{\frac{\Phi'}{\Phi_0}}$, где $\Phi' = \left\|A\vec{p} - f\right\|^2$, а $\Phi_0 = \left\|f\right\|^2$, на совокупности точек, промежуточных по отношению к точкам коллокации.

3. Численные результаты

Описанная выше программа была использована для исследования зависимости сечений обратного рассеяния структуры от угла падения плоской волны при различных значениях величины поглощения внутри диэлектрического шара.

На рис. 3, 4 показаны полученные в результате выполненных расчетов зависимости сечений обратного рассеяния исследуемой структуры от угла ψ падения плоской волны при различных значениях относительной диэлектрической проницаемости диэлектрического шара. Мнимая часть относительной диэлектрической проницаемости диэлектрического шара характеризует величину потерь (поглощения) внутри шара. Предполагается, что плоская волна падает на структуру таким образом, что векторы $\vec{k}_e$ и $\vec{E}_0$ ле-

жат в плоскости *xz*, угол ψ – угол между направлением падения волны, характеризуемым волновым вектором $\vec{k}_e$, и осью *z*. Угол ψ отложен по оси абсцисс, а по оси ординат отложено сечение обратного рассеяния, нормированное на квадрат длин волны.

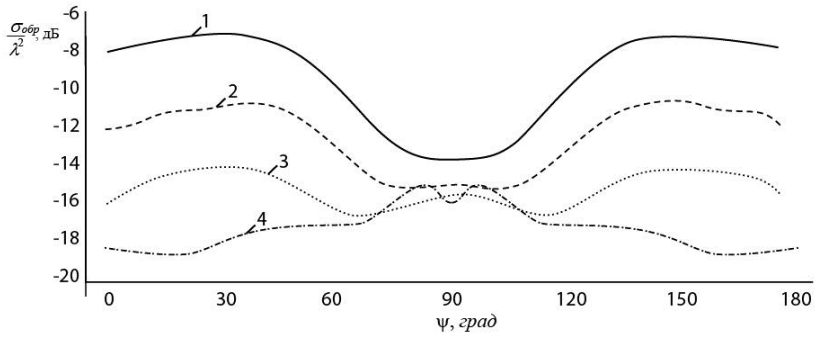


Рис. 3. Зависимости сечений обратного рассеяния от угла ψ падения плоской волны

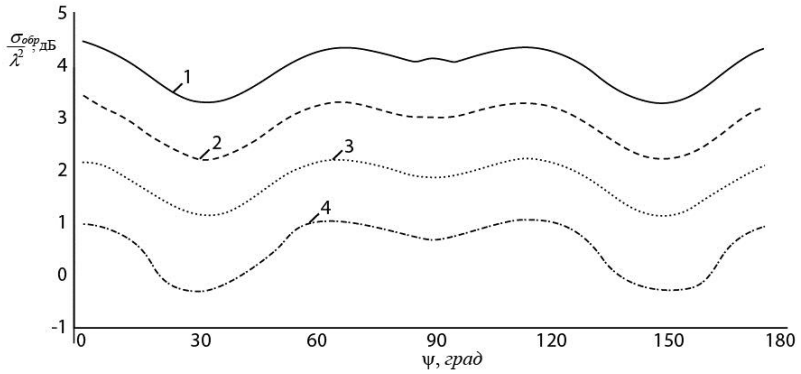


Рис. 4. Зависимости сечений обратного рассеяния от угла ψ падения плоской волны

На рис. 3 кривая 1 относится к случаю, когда относительная диэлектрическая проницаемость диэлектрического цилиндра $\varepsilon_i' = 2$, кривая 2 – к случаю, когда $\varepsilon_i' = 2 + 0.1i$, кривая 3 – к случаю, когда $\varepsilon_i' = 2 + 0.2i$, и кривая 4 – к случаю, когда $\varepsilon_i' = 2 + 0.3i$. На рис. 4 кривая 1 относится к случаю, когда относительная диэлектрическая проницаемость диэлектрического цилиндра $\varepsilon_i' = 10$, кривая 2 – к случаю, когда $\varepsilon_i' = 10 + 0.1i$, кривая 3 – к случаю, когда $\varepsilon_i' = 10 + 0.2i$, и кривая 4 – к случаю, когда $\varepsilon_i' = 10 + 0.3i$. Во всех случаях предполагалось, что безразмерная длина идеально проводящего цилиндра $k_e l_p = 3.1416$, радиус $k_e r_p = 0.1$, а радиус диэлектрического шара $k_e R = 3$. Число разбиений осевой линии идеально проводящего цилиндра выбрано равным 30, а соотношение радиусов вспомогательных поверхностей к радиусу диэлектрического шара выбраны $K_i = 4$, $K_e = 0.5$.

Как показывает рис. 3, при малых значениях относительной диэлектрической проницаемости наличие поглощения приводит к существенному уменьшению сечений обратного рассеяния, а также к качественному изменению характера зависимости сечений обратного рассеяния от угла падения плоской волны.

На рис. 4 видно, что увеличение величины поглощения также приводит к уменьшению сечений обратного рассеяния, но характер зависимости практически не меняется.

Заключение

Таким образом, в данной работе представлена программа расчета характеристик рассеянного поля на тонком идеально проводящем цилиндре, расположенном внутри диэлектрического шара. Подробно исследованы зависимости сечений рассеяния от угла падения плоской волны для случая, когда идеально проводящий цилиндр с параметрами $k_e l_p = 3.1416$ и $k_e r_p = 0.1$ расположен внутри шара с безразмерным радиусом $k_e R = 3$, при различных мнимой части диэлектрической проницаемости диэлектрического шара, характеризующей потери электромагнитной энергии внутри диэлектрического шара. Показано, что при увеличении мнимой части сечения обратного рассеяния значительно уменьшаются, а при близких значениях мнимой и реальной части относительной диэлектрической проницаемости диэлектрического шара изменяется также характер зависимости сечений обратного рассеяния от угла падения плоской волны.

ЛИТЕРАТУРА

1. *Балашова О.М., Дмитренко А.Г.* Численный метод решения задачи электромагнитного рассеяния на тонком идеально проводящем цилиндре, размещенном внутри диэлектрического шара // Материалы X-й Международной молодёжной научной конференции «Математическое и программное обеспечение информационных, технических и экономических систем». – 2023. – С. 36–41.

2. *Дмитренко А.Г., Мукомолов А.И.* Численный метод решения задач электромагнитного рассеяния на трёхмерном магнитодиэлектрическом теле произвольной формы // Радиотехника и электроника. – 1995. – Т. 40. – № 6. – С. 875–880.

3. *Дмитренко А.Г., Колчин В.А.* Рассеяние электромагнитных волн на структурах, содержащих тонкие проводники // Изв. вузов. Радиофизика. – 2003. – Т. 46. – № 1. – С. 31–40.

ПРИМЕНЕНИЕ МЕТОДА ПРОВЕРКИ СТАТИСТИЧЕСКИХ ГИПОТЕЗ ДЛЯ ИЗУЧЕНИЯ ИЗЛУЧЕНИЯ БЕЗОБЛАЧНОЙ АТМОСФЕРЫ

Дорошенко А.А.
Томский государственный университет
magw1901@rambler.ru

Введение

Изучение излучения безоблачной атмосферы является важной задачей для понимания воздействия окружающей среды на наблюдаемые объекты. Безоблачная атмосфера играет ключевую роль в процессе распространения солнечного излучения на поверхность Земли, а также влияет на климатические изменения и состояние окружающей среды [1]. Для более глубокого понимания этого процесса, в данной статье рассматривается метод проверки статистических гипотез для анализа характеристик излучения безоблачной атмосферы.

Цель исследования заключается в построении вероятностной модели и выявлении взаимосвязи между интенсивностью солнечной дымки и другими атмосферными параметрами. С помощью статистического анализа была построена вероятностная модель, которая позволила выявить взаимосвязь между интенсивностью солнечной дымки и другими атмосферными параметрами.

В работе были использованы тесты Колмогорова и Пирсона для проверки статистических гипотез о соответствии эмпирического распределения теоретическому. Эти тесты являются одними из наиболее распространенных методов в области статистики и позволяют оценить, насколько хорошо данные соответствуют предполагаемой теоретической модели.

1. Методы проверки статистических гипотез: тест Колмогорова и тест Пирсона

Тест Колмогорова является одним из наиболее распространенных статистических тестов, используемых для проверки соответствия данных предполагаемому распределению. Он основан на сравнении эмпирической функции распределения с теоретической, что позволяет оценить, насколько хорошо данные соответствуют предполагаемому распределению и выявить наличие значимых отклонений. Этот тест может быть применен для различных типов данных, включая непрерывные, дискретные и категориальные. Схема вычисления критерия Колмогорова для нашего вида работы приведен на рис. 1 [2].

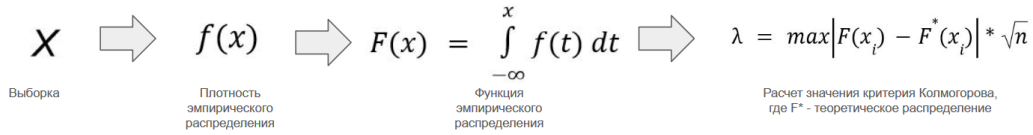


Рис. 1. Критерий Колмогорова

Тест Пирсона, или хи-квадрат тест, также является одним из наиболее распространенных статистических методов, используемых для анализа категориальных данных. Он основан на сравнении наблюдаемых и ожидаемых частот в различных категориях данных. Этот тест позволяет оценить, насколько хорошо данные соответствуют предполагаемой теоретической модели и выявить наличие значимых отклонений. Значение статистики хи-квадрат сравнивается с критическим значением, которое зависит от уровня значимости и степеней свободы. Если значение статистики хи-квадрат превышает критическое значение, то мы можем сделать вывод о наличии значимых отклонений в данных. В противном случае, мы не можем отвергнуть предположение о соответствии данных теоретической модели. Схема вычисления критерия Пирсона для нашего вида работы приведен на рис. 2 [2].

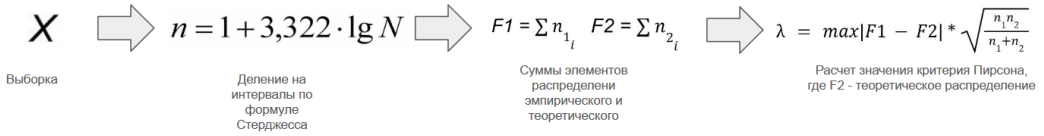


Рис. 2. Критерий Пирсона

Одним из важных этапов метода проверки статистических гипотез является расчет характеристик распределений. Это позволяет определить тип распределения данных и оценить его параметры, такие как среднее значение и стандартное отклонение. Эмпирическое распределение представляет собой фактические данные, полученные в результате наблюдений или экспериментов. Теоретическое распределение, в свою очередь, является математической моделью, описывающей вероятность различных значений случайной величины. Сравнение этих двух типов распределений позволяет оценить, насколько хорошо теоретическая модель соответствует реальным данным. Для этого используются различные статистические показатели, такие как среднее значение, медиана, дисперсия и другие. Также важно обратить внимание на форму распределения и наличие выбросов.

Кроме того, для оценки точности полученных результатов необходимо рассчитать среднеквадратичную ошибку. Она показывает, насколько сильно полученные значения отличаются от истинных. Чем меньше значение среднеквадратичной ошибки, тем более точными будут результаты исследования. Также важным показателем при работе с методом проверки статистических гипотез является максимальная разность между максимальными элементами распределения. Это позволяет оценить разброс данных и определить наличие выбросов, которые могут исказить результаты исследования [3].

2. Метод обработки данных

В ходе предобработки данные преобразуются в одно распределение. Далее находится плотность распределения, а затем на основе нее вычисляется сама функция распределения. Такой подход позволяет более точно исследовать данные и выявить закономерности, которые могут быть полезны для дальнейшего анализа. Алгоритм предобработки для наших исходных данных представлен на рис. 3.

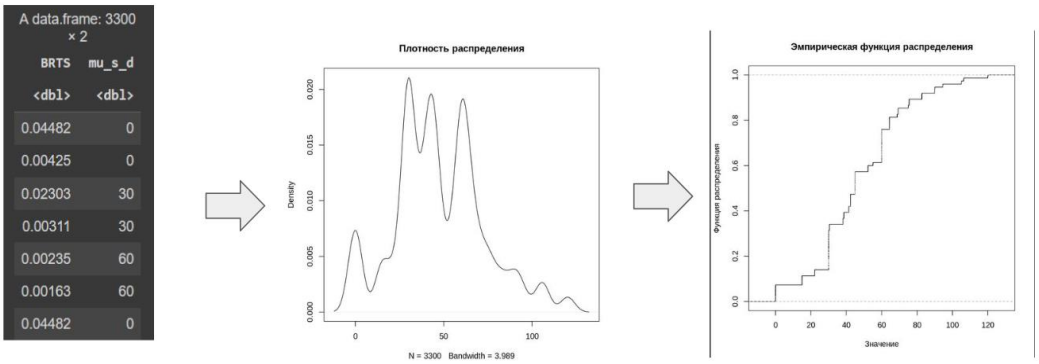


Рис. 3. Обработка исходных данных
на примере распределения яркости от косинуса угла между направлением солнечных лучей и наблюдением

Далее на двух наглядных примерах рассмотрим эффективность применения метода проверки статистических гипотез.

3. Пример 1: яркость от косинуса угла между направлением солнечных лучей и наблюдением

В первом примере приводится распределение яркости от косинуса угла между направлением солнечных лучей и наблюдением. На рис. 4 представлено данное распределение двух переменных. На рис. 5 представлено то же распределение, но яркость представлена в логарифмическом масштабе.

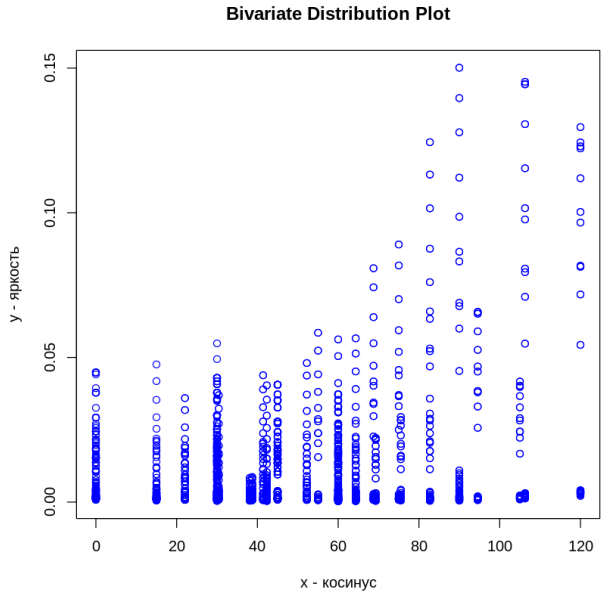


Рис. 4. Распределение яркости от косинуса

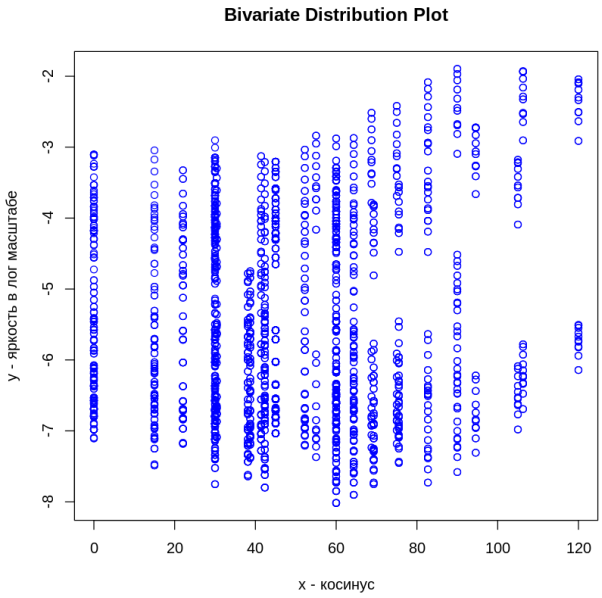


Рис. 5. Распределение яркости в логарифмическом масштабе от косинуса

Распределение отображает хорошую согласованность на тестах с теоретическими данными. На рис. 6 представлены вычисленная эмпирическая и теоретические функции.

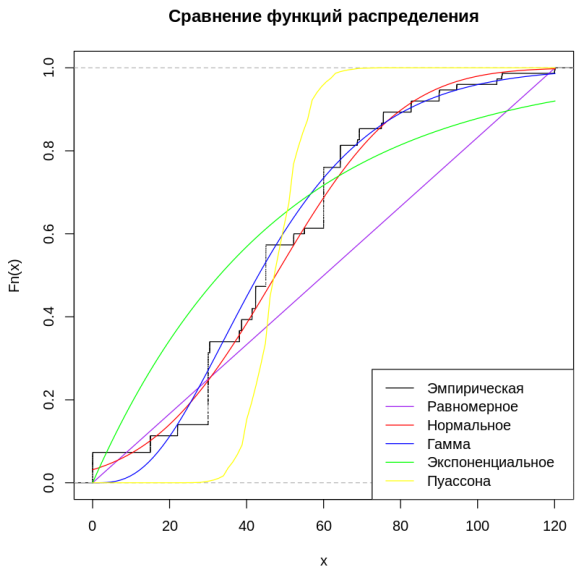


Рис. 6. Сравнение эмпирического и теоретических распределений

По тесту Колмогорова принимаются гипотезы о соответствии данного эмпирического распределения двум предложенным теоретическим – нормальному и гамма-распределению с результатами в 0.3667 и 0.3667 соответственно. По тесту Пирсона нормальное и гамма-распределения также считаются вполне схожими с данным эмпирическим распределением с результатами в 0.2762 и 0.2762 соответственно. Ниже

представлена таблица с результатами вычислений (MSE – Mean Squared Error, среднеквадратическая ошибка).

Таблица 1

Результаты вычислений в Примере 1

	Равномерное	Нормальное	Гамма	Экспоненциальное	Пуассона
MSE	0.021	0.001	0.002	0.016	0.023
Макс. разность	0.268	0.096	0.116	0.318	0.341
Тест Колмогорова (значение p-value)	0.001365	0.3667	0.3667	0.0002468	5.099e-10
Тест Пирсона (значение p-value)	0.01999	0.2762	0.2762	0.002762	0.005412

Результаты показывают, что данное распределение более точно соответствует нормальному и гамма-распределению. Это означает, что модель и прогнозирование могут быть построены на основе этих распределений.

4. Пример 2: яркость от зенитного угла с учетом длины волны

Второй пример иллюстрирует то, что не все вычисленные эмпирические функции распределения хорошо подходят под используемые нами методы проверки статистических гипотез. К таким распределениям нужен другой подход. Тем не менее, изучение таких распределений в рамках такого исследования может дать свои интересные результаты.

На рис. 7 представлено распределение двух переменных – яркости от зенита. На рис. 8 представлено то же распределение, но яркость представлена в логарифмическом масштабе.

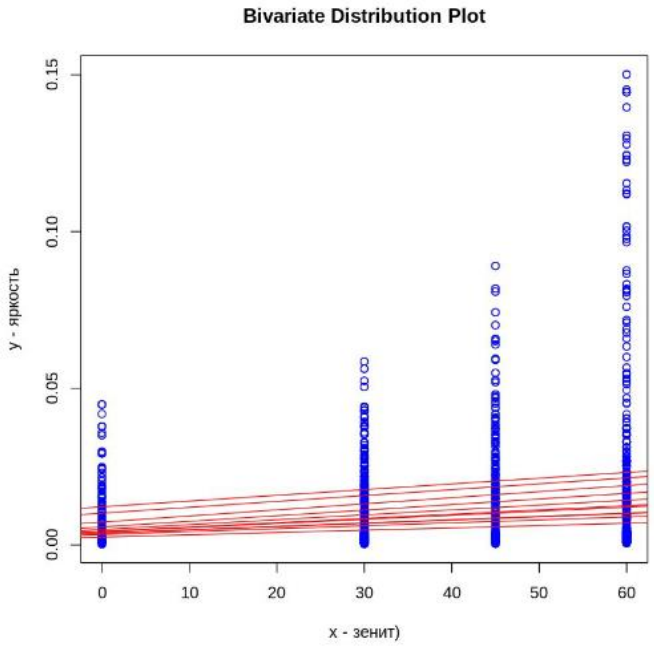


Рис. 7. Распределение яркости от зенита

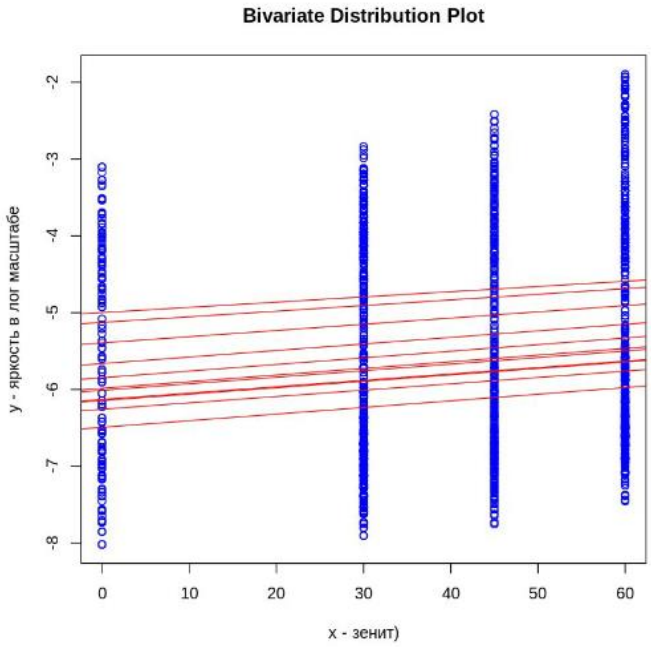


Рис. 8. Распределение яркости в логарифмическом масштабе от зенита

На обоих графиках распределений параллельные красные линии представляют собой элементы корреляционного поля. Они расположены в зависимости от значения третьей переменной выборки данного примера – длины волны.

Также по алгоритму из Примера 1 находим функцию распределения и отображаем ее на графике с другими пятью теоретическими распределениями. На рис. 9 представлены эмпирическая и теоретические функции.

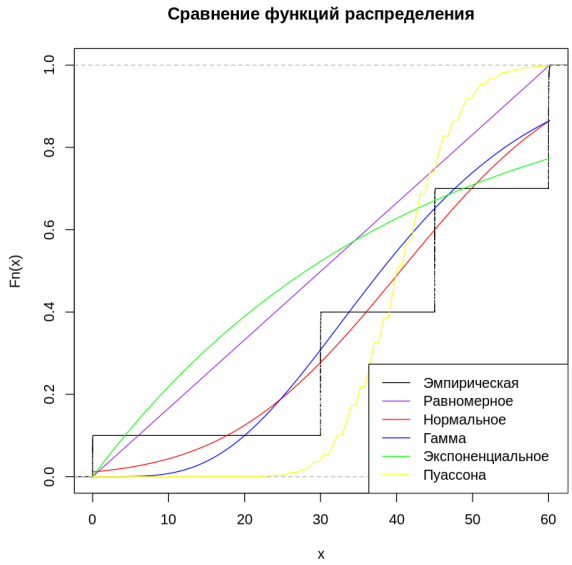


Рис. 9. Сравнение эмпирического и теоретических распределений

Ниже представлены результаты вычислений (MSE – Mean Squared Error, среднеквадратическая ошибка).

Таблица 2

Результаты вычислений в Примере 2

	Равномерное	Нормальное	Гамма	Экспоненциальное	Пуассона
MSE	0.044	0.01	0.011	0.041	0.033
Макс. разность	0.394	0.2	0.116	0.42	0.35
Тест Колмогорова (значение p-value)	2.251e-07	0.0004453	7.143e-05	4.366e-08	4.366e-08
Тест Пирсона (значение p-value)	0.0004343	0.0004343	0.0004343	0.0004343	5.334e-06

Результаты тестов показывают, что для представленного распределения отклоняется гипотеза о соответствии пяти выбранным теоретическим распределениям.

Заключение

В ходе исследования были использованы тесты Колмогорова и Пирсона, которые позволили получить интересные для дальнейшей работы результаты. Оба теста показали свои уникальные результаты в рамках данного исследования. Это подтверждает эффективность и применимость метода проверки статистических гипотез в данной области исследований.

Данная работа является важным шагом в изучении излучения безоблачной атмосферы и может служить основой для дальнейших исследований в этой области. Результаты данного исследования могут быть использованы для улучшения понимания процессов, происходящих в атмосфере, а также для разработки новых методов и технологий в области оптики атмосферы.

ЛИТЕРАТУРА

1. *Матвеев Л.Т.* Курс общей метеорологии. Физика атмосферы. – Ленинград: Гидрометеоиздат, 1976 – 640 с.

2. *Митропольский А.К.* Техника статистических вычислений. – М.: Наука, 1971 – 576 с.

3. *Кремер Н.Ш.* Теория вероятностей и математическая статистика: Учебник для вузов. – 2-е изд., перераб. и доп. – М.: ЮНИТИ, 2004 – 573 с.

ИСПОЛЬЗОВАНИЕ ПРЯМОГО МОДЕЛИРОВАНИЯ ДЛЯ РАСЧЕТА ХАРАКТЕРИСТИК УХОДЯЩЕГО ИЗЛУЧЕНИЯ ПОДСТИЛАЮЩЕЙ ПОВЕРХНОСТИ

Клименко А.И., Гендрина И.Ю.
Томский государственный университет
anyaklimenko@inbox.ru

Введение

В современном мире актуален вопрос об изучении процесса переноса излучения в атмосфере, в связи с тем, что антропогенные факторы приводят к изменениям в радиационном поле Земли. Радиационные процессы играют ключевую роль в тепловом и энергетическом обмене. Их понимание необходимо для прогнозирования и оценки климатических изменений и их воздействия на окружающую среду. Кроме того, моделирование переноса излучения в атмосфере важно для различных областей науки и техники, таких, как атмосферная оптика и аэрокосмическая инженерия.

Целью данной работы является исследование влияния атмосферы на перенос уходящего излучения подстилающей поверхности.

В приближении лучевой оптики задачи теории переноса излучения в атмосфере и океане с учетом многократного рассеяния и детализированной радиационной модели среды, как правило, описываются интегро-дифференциальными уравнениями с соответствующими граничными условиями [1]. С учетом различных реальных условий эти уравнения очень сложно или даже невозможно решать классическими методами. По-

этому целесообразно использование методов Монте-Карло, или методов прямого моделирования. К плюсам прямого моделирования переноса излучения в атмосфере относится возможность учитывать неоднородность атмосферы и различные характеристики подстилающих поверхностей.

Методы Монте-Карло заключаются в воспроизведении с помощью электронной вычислительной машины функционирования вероятностной модели некоторого объекта. Метод Монте-Карло для моделирования процесса переноса излучения в атмосфере представляет собой моделирование траекторий полёта отдельных фотонов в рассеивающей и поглощающей среде и вычисление различных статистических оценок для искомых функционалов.

Прямое моделирование позволяет вычислять интегральные характеристики переноса излучения в атмосфере, а также его статистические характеристики, например, математические ожидания характеризующих его величин, их дисперсии и ковариации. В данной работе рассматривается поток через верхнюю границу атмосферы и подстилающую поверхность.

1. Постановка задачи

В работе рассматривается излучение точечного изотропного источника, находящегося на подстилающей поверхности. Его местоположение принимаем за начало координат (0,0,0). Дана плотность распределения направлений излучения источника:

$$f(\mu,\varphi) = 1/2\pi\,, \qquad (1)$$

где μ – косинус зенитного угла θ, $\mu \in [0;1]$, φ – азимутальный угол, $\varphi \in [0;2\pi]$.

Рассматриваются 5 длин волн излучения из интервала от 0.55 до 1.06 микрометров, которые находятся в окнах прозрачности, т.е. для них можно пренебречь поглощением газами в атмосфере.

Для решения задач теории переноса излучения в атмосфере методом прямого моделирования обычно среду разбивают на достаточно малые области с постоянным коэффициентом ослабления σ. В данной работе в качестве модели атмосферы была использована плоскопараллельная модель с горизонтально однородными слоями. Нижняя граница среды представляет собой поверхность Земли на высоте в 0 км, верхняя граница – 100 км над поверхностью. В качестве приемников излучения рассматриваются верхняя и нижняя границы среды.

В качестве исходных данных были предоставлены коэффициенты ослабления потока σ (табл. 1) и альбедо однократного рассеяния, т.е. вероятность рассеяния при столкновении фотонов со средой (табл. 2).

Таблица 1

Коэффициенты ослабления потока

высота (км)	(0;3)	[3;13)	[13;25)	[25;35)	[35;100)
σ	0.1	0.0025	0.000218	0.0003	0

Таблица 2

Альбедо однократного рассеяния

длина волны/ высота (км)	0.55	0.633	0.694	0.86	1.06
(0;13]	0.891	0.883	0.879	0.841	0.804
[13;100]	1	1	1	1	1

Схематично модель атмосферы представлена на рис. 1. Пунктирные линии – границы однородных слоев атмосферы.

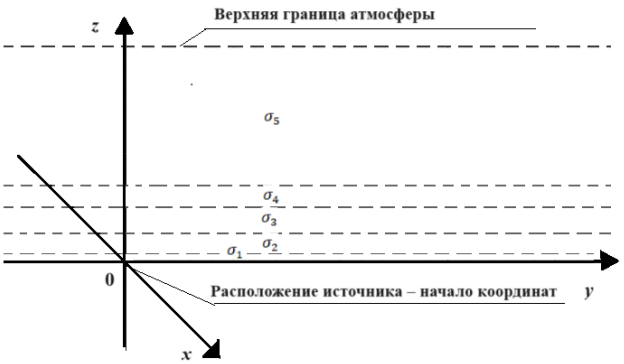


Рис. 1. Модель атмосферы

В работе было учтено переотражение излучения от подстилающей поверхности и рассмотрены различные виды отражения. Способность поверхности отражать излучение и распределение направлений его отражения зависят от её типа, температуры, влажности и других факторов. Альбедо подстилающей поверхности – величина, характеризующая отношение рассеянного поверхностью во всех направлениях светового потока к общему потоку, падающему на поверхность.

Чем светлее поверхность и ниже ее температура, тем больше ее альбедо. Так, свежевыпавший снег имеет альбедо 0.80–0.95, тающий снег – 0.30–0.65, хвойный лес – 0.1–0.15. Примеры значений были взяты из [2].

Для различных поверхностей характерны различные виды отражения. Вероятностное распределение направлений, в котором поверхность отражает падающее на неё световое излучение, зависит от угла падающего излучения, от коэффициента отражения поверхности, от того, гладкая ли поверхность или имеет неровности и шероховатости. На рис. 2 схематично показаны виды отражения излучения.

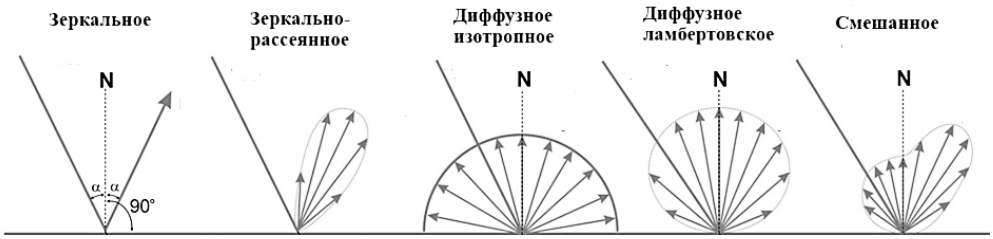


Рис. 2. Виды отражения излучения

В данной работе были рассмотрены 3 вида отражения: зеркальное, изотропное и ламбертовское.

Зеркальное отражение происходит по следующему закону: угол падения света равен углу его отражения. Другими словами, косинус зенитного угла падения излучения равен косинусу зенитного угла отражения, умноженному на –1. Зеркальное отражение имеет место при отражении от гладких поверхностей, таких, как, например, зеркало или вода. Существует так же зеркально-рассеянное отражение, когда траектория отраженного излучения немного отклоняется от траектории зеркального отражения.

Диффузное отражение имеет место при отражении излучения от шероховатых и матовых поверхностей, если неровности поверхности имеют порядок длины волны или превышают ее, или от материалов с низкой степенью отражения. Возможно диффузное отражение и от гладких поверхностей, если в прозрачной среде материала поверхности имеются центры рассеивания света – неоднородности, например, полированный белый мрамор или поверхность молока. Одна и та же поверхность может быть матовой, диф-

фузно-отражающей для видимого или ультрафиолетового излучения, но гладкой и зеркально-отражающей для инфракрасного излучения.

Для изотропного отражения интенсивность отраженного излучения одинаковая по всем направлениям. Плотность распределения направлений отраженного излучения представлена в формуле (1).

Для ламбертовского отражения сила отраженного в направлении зенитного угла θ света изменяется по закону косинусов: $I = I_0 \cos\theta$, где I_0 – сила отраженного света в направлении нормали к поверхности, θ – угол между рассматриваемым направлением и нормалью. Плотность распределения направлений отраженного излучения: $f(\mu,\varphi) = \mu/\pi$, где μ – косинус зенитного угла θ, $\mu \in [0;1]$, φ – азимутальный угол, $\varphi \in [0;2\pi]$.

2. Интегральная оценка характеристик переноса излучения

Прямое моделирование позволяет вычислять различные интегральные характеристики переноса излучения. Обозначим через $f(\vec{r},\vec{\omega})$ плотность столкновений, через $\Phi(\vec{r},\vec{\omega})$ – поток частиц (интенсивность излучения). Поток частиц Ф определяется следующим образом: величина $\Phi(\vec{r},\vec{\omega})dsd\vec{\omega}$ равна среднему числу частиц, пересекающих площадку *ds*, ориентированную перпендикулярно $\vec{\omega}$ в точке $\vec{r}$, имея направление движения $[\vec{\omega},\vec{\omega}+d\vec{\omega}]$. Известно, что $f(\vec{r},\vec{\omega}) = \sigma(\vec{r})\Phi(\vec{r},\vec{\omega})$.

Поток частиц Φ_s через произвольную элементарную площадку *ds* определяется формулой $\Phi_s(\vec{r},\vec{\omega}) = |\vec{n}_s,\vec{\omega}|\Phi(\vec{r},\vec{\omega})$, где $\vec{n}_s$ – единичный вектор нормали к *ds*.

Среднее число столкновений частицы в некоторой области D_i является оценкой интеграла $\int_{D_i} f(x)dx$, $x = (\vec{r},\vec{\omega})$.

Среднее число пересечений некоторой области *S* в интервале направлений Ω_i дает оценку интеграла $\int_S ds \int_{\Omega_i} \Phi(\vec{r}(s),\vec{\omega})|\vec{\omega},\vec{n}_s|d\vec{\omega}$.

Если эти пересечения учитывать с весом $1/|\vec{n}_s,\vec{\omega}|$, т.е. с учетом направления пересечения, то получаем оценку такого интеграла:

$$\int_S ds \int_{\Omega_i} \Phi(\vec{r}(s),\vec{\omega})d\vec{\omega}. \quad (2)$$

Таким образом, можно оценивать средние интегральные значения функций *f*(*x*) и Ф(*x*).

3. Результаты

Моделирование было произведено с использованием общей схемы моделирования процесса переноса [3] в программе в среде Visual Studio 2019 на языке C++. Число моделируемых пробегов частиц для каждой длины волны равно одному миллиону. В программе была произведена оценка интегральных характеристик уходящего излучения подстилающей поверхности, а именно, найдены потоки частиц через верхнюю границу атмосферы и подстилающую поверхность. Поток излучения через поверхность выражается через интеграл (2). Оценки были получены для различных характеристик подстилающей поверхности: альбедо подстилающей поверхности и вида отражения.

Таблица 3

Оценка потоков частиц

длина волны (мкм)	0.55	0.633	0.694	0.86	1.06
поток частиц через верхнюю границу атмосферы	1.93826	1.92847	1.9143	1.8699	1.83071
поток частиц через подстилающую поверхность	0.420182	0.422182	0.41705	0.39375	0.373043

Оба потока уменьшаются с увеличением длины волны излучения.

Была исследована зависимость потоков частиц от характеристик подстилающей поверхности, таких, как альбедо подстилающей поверхности и виды отражения. Результаты для потока частиц через верхнюю границу атмосферы для рассматриваемых длин волн представлены на графиках на рис. 3–7.

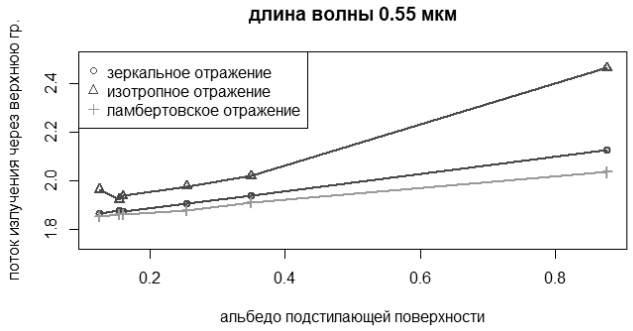


Рис. 3. Зависимость потока частиц через верхнюю границу атмосферы от характеристик подстилающей поверхности для длины волны 0.55 мкм

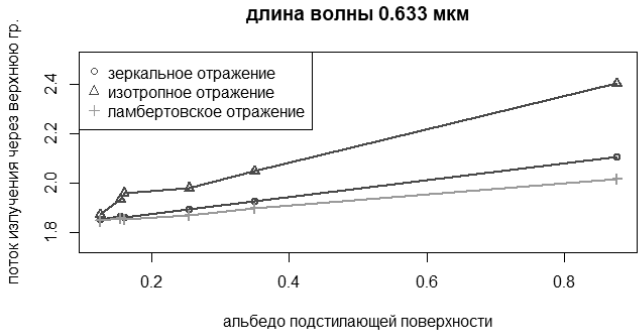


Рис. 4. Зависимость потока частиц через верхнюю границу атмосферы от характеристик подстилающей поверхности для длины волны 0.633 мкм

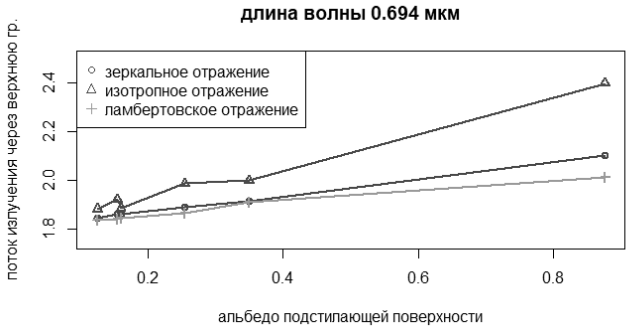


Рис. 5. Зависимость потока частиц через верхнюю границу атмосферы
от характеристик подстилающей поверхности для длины волны 0.694 мкм

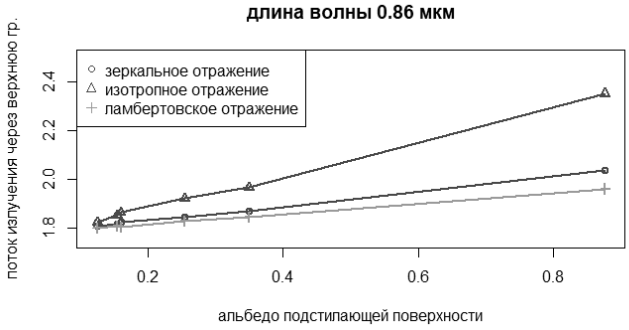


Рис. 6. Зависимость потока частиц через верхнюю границу атмосферы
от характеристик подстилающей поверхности для длины волны 0.86 мкм

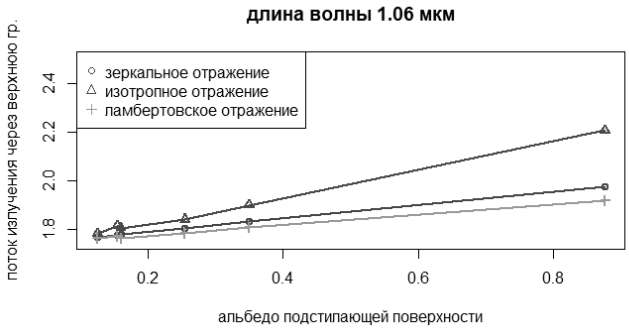


Рис. 7. Зависимость потока частиц через верхнюю границу атмосферы
от характеристик подстилающей поверхности для длины волны 1.06 мкм

Для рассматриваемых длин волн поток частиц через верхнюю границу атмосферы уменьшается с увеличением альбедо подстилающей поверхности. Самый маленький поток характерен для ламбертовского отражения. Немного больше поток для зеркального отражения. Для изотропного отражения поток заметно больше, чем для двух других. С увеличением альбедо подстилающей поверхности разница результатов для различных видов отражения увеличивается.

На рис. 8–12 представлены графики результатов для потока частиц через подстилающую поверхность для рассматриваемых длин волн.

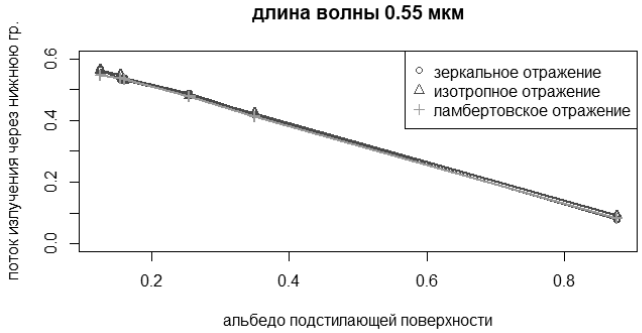


Рис. 8. Зависимость потока частиц через подстилающую поверхность от характеристик подстилающей поверхности для длины волны 0.55 мкм

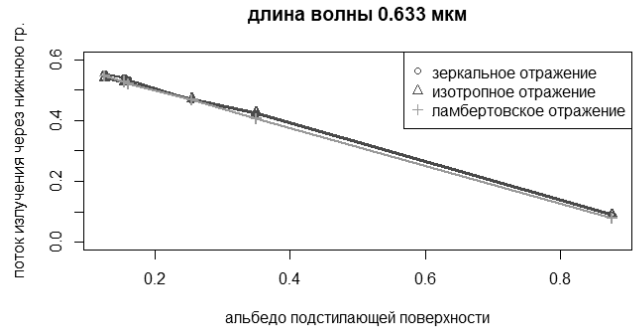


Рис. 9. Зависимость потока частиц через подстилающую поверхность от характеристик подстилающей поверхности для длины волны 0.633 мкм

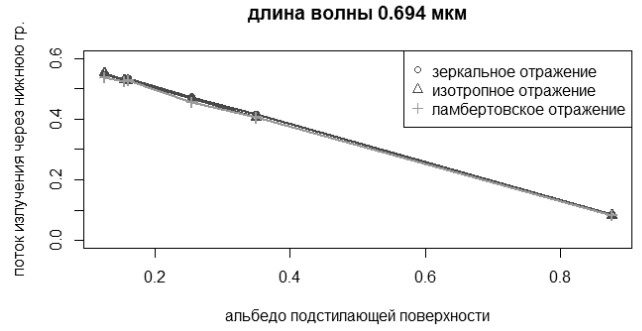


Рис. 10. Зависимость потока частиц через подстилающую поверхность от характеристик подстилающей поверхности для длины волны 0.694 мкм

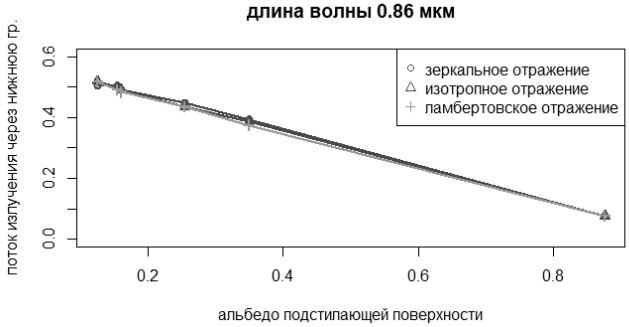


Рис. 11. Зависимость потока частиц через подстилающую поверхность от характеристик подстилающей поверхности для длины волны 0.86 мкм

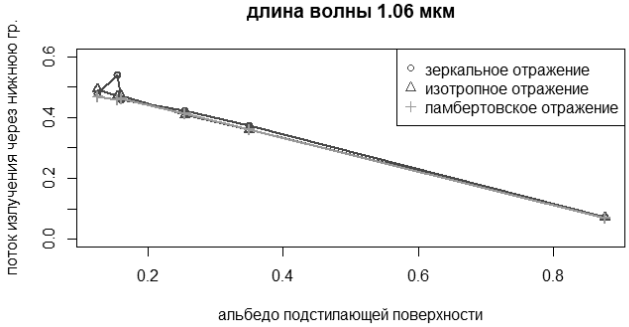


Рис. 12. Зависимость потока частиц через подстилающую поверхность от характеристик подстилающей поверхности для длины волны 1.06 мкм

Поток частиц через подстилающую поверхность для рассматриваемых длин волн уменьшается с увеличением альбедо подстилающей поверхности. Вид отражения от поверхности не оказывает существенного влияния на поток. Возможно, это значит, что частицы почти всегда переотражаются от подстилающей поверхности не более 1-го раза.

Заключение

В данной работе был исследован перенос уходящего излучения подстилающей поверхности и произведена оценка его интегральных характеристик. Моделирование переноса уходящего излучения подстилающей поверхности было реализовано с использованием общей схемы моделирования переноса излучения [3]. Было учтено переотражение излучения.

Была исследована зависимость потоков частиц на границах атмосферы от оптических характеристик атмосферы и от характеристик подстилающей поверхности, таких, как альбедо подстилающей поверхности и виды отражения.

ЛИТЕРАТУРА

1. *Марчук Г.И., Михайлов Г.А., Назаралиев М.А.* Метод Монте-Карло в атмосферной оптике. – Новосибирск: Изд-во «Наука», Сиб. Отделение, 1976. – 280 с.

2. *Павлов А.В.* Мониторинг Криолитозоны. – Новосибирск: Акад. изд-во «Гео», 2008. – 224 с.

3. *Ермаков С.М., Михайлов Г.А.* Курс статистического моделирования. – М.: Изд-во «Наука», 1976. – 320 с.

СТАТИСТИЧЕСКИЕ ЭКСПЕРИМЕНТЫ НА ИМИТАЦИОННОЙ МОДЕЛИ ПОЛУСИНХРОННОГО ПОТОКА СОБЫТИЙ ВТОРОГО ПОРЯДКА В СХЕМЕ С ПРОДЛЕВАЮЩИМСЯ МЁРТВЫМ ВРЕМЕНЕМ

Нежельская Л.А., Останин Р.О.
Томский государственный университет
ludne@mail.ru, ostanin.roma15@gmail.com

Введение

Первые задачи теории массового обслуживания (ТМО) были рассмотрены сотрудником Копенгагенской телефонной компании, датским учёным А.К. Эрлангом. В своей статье, опубликованной в 1909 г. [1], А.К. Эрланг использовал модель пуассоновского потока как математическую модель поступающих на телефонную станцию вызовов. Далее теорию потока однородных событий, которая легла в основу ТМО, разработал советский математик А.Я. Хинчин [2].

В связи со значительным ростом информационных потоков сообщений, запросов, заявок в сетях связи, телекоммуникационных сетях модель пуассоновского потока событий стала устаревать и терять адекватность отображения реальных потоков данных. На смену простейшему (стационарному пуассоновскому) потоку пришли разработанные дважды стохастические потоки событий, которые способны адекватно отображать реальные информационные потоки.

Впервые модель дважды стохастического потока была опубликована в 1955 г. Д. Коксом. В своей работе Д. Кокс рассматривал дважды стохастические потоки событий, интенсивность которых является непрерывным случайным процессом. В 1979 г. в работах Г.П. Башарина, В.А. Кокотушкина, В.А. Наумова и М. Ньютса, а в 1991 г. – в работе Д. Лукантони, были рассмотрены дважды стохастические потоки с интенсивностью, являющейся кусочно-постоянным случайным процессом с конечным числом состояний.

Систематизированный материал по дважды стохастическим потокам событий, интенсивность которых является кусочно-постоянным случайным процессом с конечным числом состояний, представлен в [3].

В данной работе исследована одна из таких моделей дважды стохастического потока событий, интенсивность которого является кусочно-постоянным случайным процессом с двумя состояниями – полусинхронный поток событий второго порядка, функционирующий в условиях продлевающегося мёртвого времени фиксированной длительности.

Впервые математическая модель полусинхронного потока событий второго порядка как в условиях полной наблюдаемости, так и при наличии непродлевающегося мёртвого времени, была рассмотрена в [4].

В данной работе представлена модель потока, функционирующего в условиях продлевающегося мёртвого времени, что представляет новизну проведённого исследования.

Построена имитационная модель потока и проведен ряд статистических экспериментов с целью получения результатов, численно доказывающих адекватность построенной модели.

1. Постановка задачи

Рассматривается полусинхронный поток событий второго порядка с двумя состояниями, сопровождающий случайный процесс которого $\lambda(t)$ является кусочно-постоянным принципиально ненаблюдаемым процессом с двумя состояниями S_1, S_2.

Если $\lambda(t)=\lambda_i$, то говорят, что имеет место i-е состояние (S_i), $i=1,2$, процесса $\lambda(t)$ (потока). Обязательное условие, налагаемое на процесс $\lambda(t)$: $\lambda_1>\lambda_2\geq 0$.

Длительность интервала между событиями потока $\lambda(t)$ в состоянии S_1 определяется случайной величиной $\eta=\min\left(\xi^{(1)},\xi^{(2)}\right)$, где случайная величина $\xi^{(1)}$ имеет функцию распределения $F^{(1)}(t)=1-e^{-\lambda_1 t}$, $t\geq 0$, случайная величина $\xi^{(2)}$ имеет функцию распределения $F^{(2)}(t)=1-e^{-\alpha_1 t}$, $t\geq 0$; $\xi^{(1)}$, $\xi^{(2)}$ – независимые случайные величины. В момент наступления события потока процесс $\lambda(t)$ переходит из состояния S_1 в состояние S_2 либо с вероятностью $P_1^{(1)}\left(\lambda_2\,|\,\lambda_1\right)$, либо с вероятностью $P_1^{(2)}\left(\lambda_2\,|\,\lambda_1\right)$ в зависимости от того, какая из случайных величин $\xi^{(1)}$ или $\xi^{(2)}$ приняла минимальное значение. Также в момент наступления события потока процесс $\lambda(t)$ остается в состоянии S_1 либо с вероятностью $P_1^{(1)}\left(\lambda_1\,|\,\lambda_1\right)$, либо с вероятностью $P_1^{(2)}\left(\lambda_1\,|\,\lambda_1\right)$ в зависимости от того, какая из случайных величин $\xi^{(1)}$ или $\xi^{(2)}$ приняла минимальное значение. Вероятности по каждой из случайных величин удовлетворяют условиям: $P_1^{(1)}\left(\lambda_1\,|\,\lambda_1\right)+P_1^{(1)}\left(\lambda_2\,|\,\lambda_1\right)=1$, $P_1^{(2)}\left(\lambda_1\,|\,\lambda_1\right)+P_1^{(2)}\left(\lambda_2\,|\,\lambda_1\right)=1$. Длительность пребывания процесса $\lambda(t)$ в состоянии S_1 является случайной величиной с функцией распределения $F(t)=1-e^{-(\lambda_1+\alpha_1)t}$, $t\geq 0$ [4].

Длительность пребывания процесса $\lambda(t)$ во 2-м состоянии является случайной величиной с функцией распределения $F(t)=1-e^{-\alpha_2 t}$, $t\geq 0$. В течение времени пребывания процесса $\lambda(t)$ во 2-м состоянии имеет место пуассоновский поток событий с параметром λ_2. Переход процесса $\lambda(t)$ из 2-го состояния в 1-е происходит в произвольный момент времени, не связанный с моментом времени наступления событий пуассоновского потока с параметром λ_2.

Рассмотрим полусинронный поток событий второго порядка при его неполной наблюдаемости. К условиям математической модели, изложенной выше, добавляются условия функционирования потока в схеме с продлевающимся мёртвым временем. Схема формирования наблюдаемого потока событий определяется следующим образом.

После каждого зарегистрированного в момент времени t_k события исходного потока, $\kappa\in N$, наступает период мёртвого времени фиксированной длительности T, в течение которого каждое последующее пришедшее событие не наблюдается, однако продлевает период ненаблюдаемости исходного потока на фиксированную длительность T (продлевающееся мёртвое время). После окончания периода ненаблюдаемости первое наступившее событие снова создает период мёртвого времени длительности T и т.д. Таким образом, поскольку события наступают в случайные моменты времени, то общий период ненаблюдаемости является случайной величиной.

Рассмотрев всевозможные вероятности переходов процесса $\lambda(t)$ из состояния S_i в состояние S_j, $i,j=1,2$, можно выписать матрицы инфинитезимальных характеристик $\mathbf{D_0}$ и $\mathbf{D_1}$ процесса $\lambda(t)$:

$$\mathbf{D_0}=\begin{Vmatrix}-(\lambda_1+\alpha_1) & 0\\ \alpha_2 & -(\lambda_2+\alpha_2)\end{Vmatrix},$$

$$\mathbf{D_1}=\begin{Vmatrix}\lambda_1 P_1^{(1)}\left(\lambda_1\,|\,\lambda_1\right)+\alpha_1 P_1^{(2)}\left(\lambda_1\,|\,\lambda_1\right) & \lambda_1 P_1^{(1)}\left(\lambda_2\,|\,\lambda_1\right)+\alpha_1 P_1^{(2)}\left(\lambda_2\,|\,\lambda_1\right)\\ 0 & \lambda_2\end{Vmatrix}.$$

Элементами матрицы $\mathbf{D_1}$ являются интенсивности переходов процесса $\lambda(t)$ из состояния в состояние с наступлением события потока. Недиагональные элементы матрицы $\mathbf{D_0}$ – интенсивности переходов из состояния в состояние без наступления событий

потока. Диагональные элементы матрицы $\mathbf{D_0}$ – интенсивности выхода процесса $\lambda(t)$ из своих состояний, взятые с противоположным знаком.

Для наглядности на рис. 1 приведён пример возникающей ситуации, где $t_1, t_2, ..., t_n, ...$ – моменты наступления событий в наблюдаемом потоке; формирование периодов мёртвого времени длительности T обозначено заштрихованными серыми прямоугольниками; общий период ненаблюдаемости обозначен чёрным закрашенным прямоугольником; чёрными закрашенными кружками обозначены события полусинхронного потока второго порядка, недоступные для наблюдения.

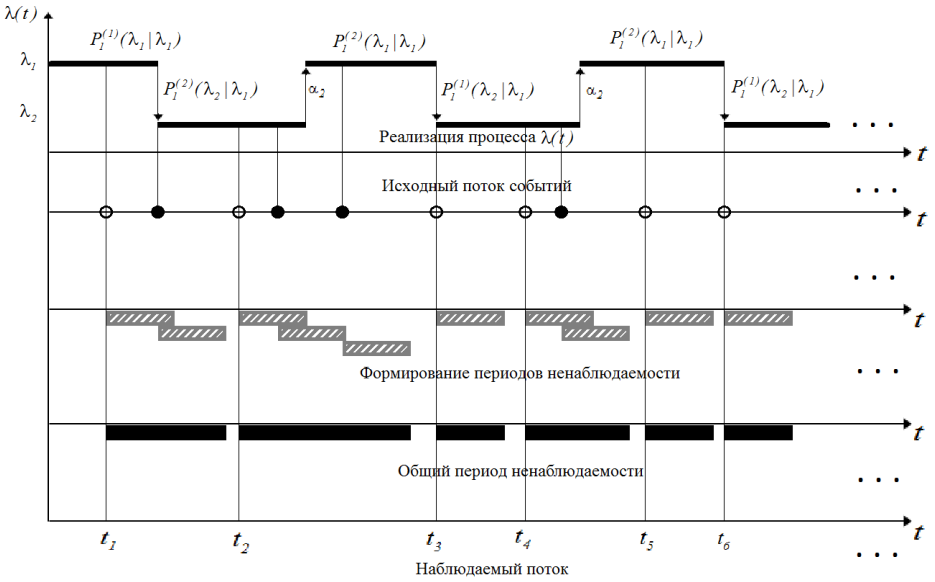


Рис. 1. Формирование наблюдаемого потока событий

2. Имитационное моделирование

Имитационная модель – это компьютерная программа, которая описывает структуру и воспроизводит поведение реальной системы во времени. Имитационная модель позволяет получать подробную статистику о различных аспектах функционирования системы в зависимости от входных данных.

Имитационная модель исследуемого потока событий построена с помощью метода обратных функций [5]. Пусть $t \geq 0$ – значение длительности пребывания процесса $\lambda(t)$ в состоянии. Случайная величина τ – длительность пребывания процесса $\lambda(t)$ в состоянии – имеет функцию распределения $F(t) = P(\tau < t) = 1 - e^{-\theta t}$, $t \geq 0$. В соответствии с методом обозначим $F(t) = \gamma$. Имеем $\gamma = 1 - e^{-\theta t}$, $t \geq 0$, тогда $-\theta t = \ln(1-\gamma)$, откуда находим формулу моделирования значений длительности интервалов пребывания процесса $\lambda(t)$ в состоянии: $t = -\frac{1}{\theta}\ln(1-\gamma)$, $\theta \in \{\lambda_1, \lambda_2\, \alpha_1\}$; γ – равномерно распределённая на интервале (0,1) случайная величина; λ_1, λ_2, α_1 – параметры потока.

3. Блок-схема имитационного моделирования потока

На рис. 2 представлена блок-схема алгоритма имитационного моделирования полусинхронного потока событий второго порядка в схеме с продлевающимся мёртвым временем.

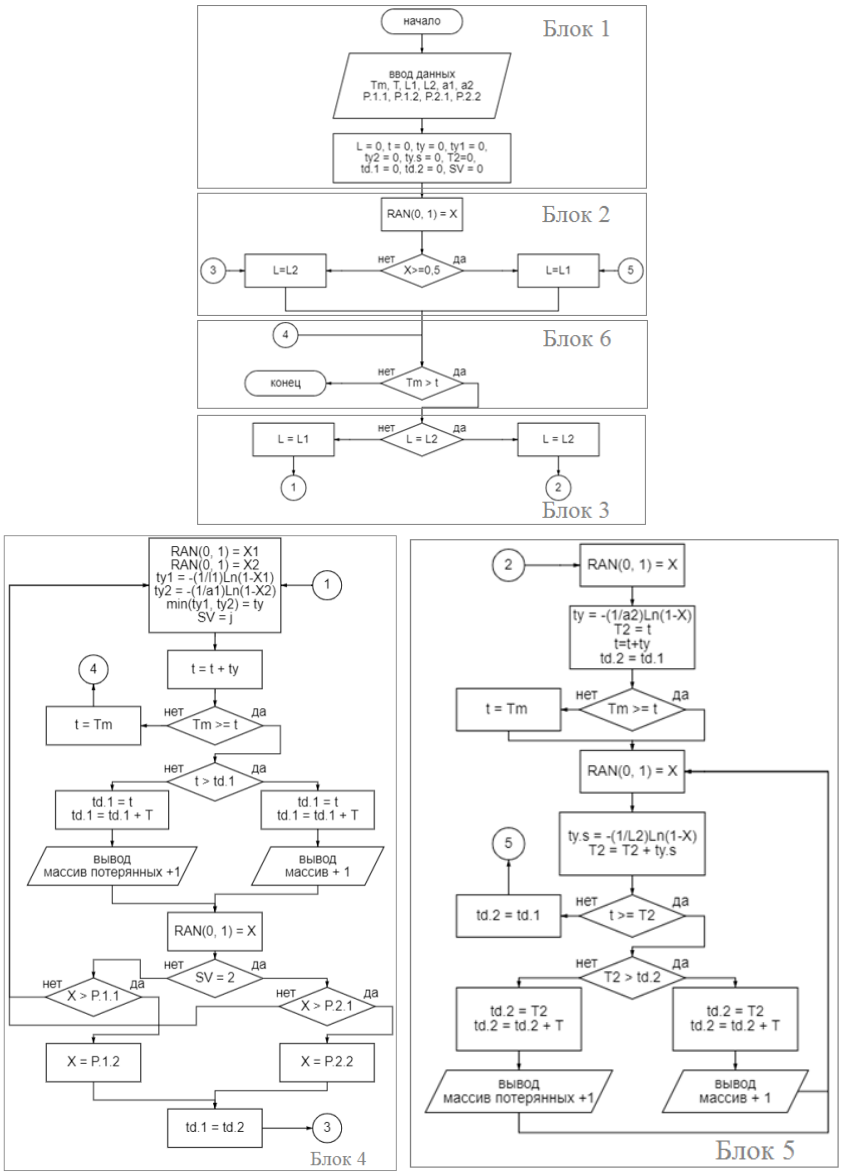


Рис. 2. Блок-схема имитационного моделирования полусинхронного потока событий второго порядка в схеме с продлевающимся мёртвым временем

При построении блок-схемы введены обозначения:

1) Tm – время моделирования (время наблюдения за потоком);
2) параметры потока: L1 – λ_1, L2 – λ_1, a1 – α_1, a2 – α_2;
3) вероятности P.1.1 – $P_1^{(1)}(\lambda_1|\lambda_1)$, P.1.2 – $P_1^{(2)}(\lambda_1|\lambda_1)$, P.2.1 – $P_1^{(1)}(\lambda_2|\lambda_1)$, P.2.2 – $P_1^{(2)}(\lambda_2|\lambda_1)$;
4) RAN (0,1) – датчик случайных чисел, равномерно распредёленных на интервале [0,1);
5) L – текущее состояние потока;
6) t – текущее время моделирования;
7) ty – длительность пребывания процесса $\lambda(t)$ в 1-м и 2-м состояниях; ty1, ty2 – периоды между событиями в 1-ом состоянии процесса $\lambda(t)$ по первой и второй слу-

чайной величине соответственно, ty.s – период между событиями пуассоновского потока с параметром λ_2;

8) SV – случайная величина, по которой реализовался минимальный период между событиями в 1-м состоянии;

9) td.1, td.2 – правая граница периода ненаблюдаемости в 1-м и 2-м состояниях соответственно;

10) T2 – дополнительная переменная значения времени моделирования во 2-м состоянии.

Результатом работы алгоритма является вектор $\mathbf{t} = (t_1 \quad t_2 \quad ... \quad t_n)$ моментов времени наступления событий полусинхронного потока второго порядка в схеме с продлевающимся мёртвым временем, где $0 < t_1 < t_2 < ... < t_n < T_m$.

Описание блоков, представленных на рис. 2

Блок 1. Начало алгоритма, инициализация параметров.

Блок 2. Розыгрыш начального состояния процесса $\lambda(t)$.

Блок 3. Переключение имитационной модели между состояниями процесса $\lambda(t)$.

Блок 4. Генерация длительности интервала между состояниями потока в 1-м состоянии процесса $\lambda(t)$.

Блок 5. Генерация длительности интервала между состояниями потока во 2-м состоянии процесса $\lambda(t)$.

Блок 6. Завершение алгоритма по достижении значения T_m.

На рис. 3 представлен результат работы программы в виде скриншота, где показана одна из реализаций полусихронного потока событий второго порядка в схеме с продлевающимся мёртвым временем. В реализации кусочно-постоянного случайного процесса $\lambda(t)$ синим и зелёным цветом представлены периоды между событиями, сгенерированные первой и второй случайной величинами соответственно в состоянии S_1 процесса $\lambda(t)$.

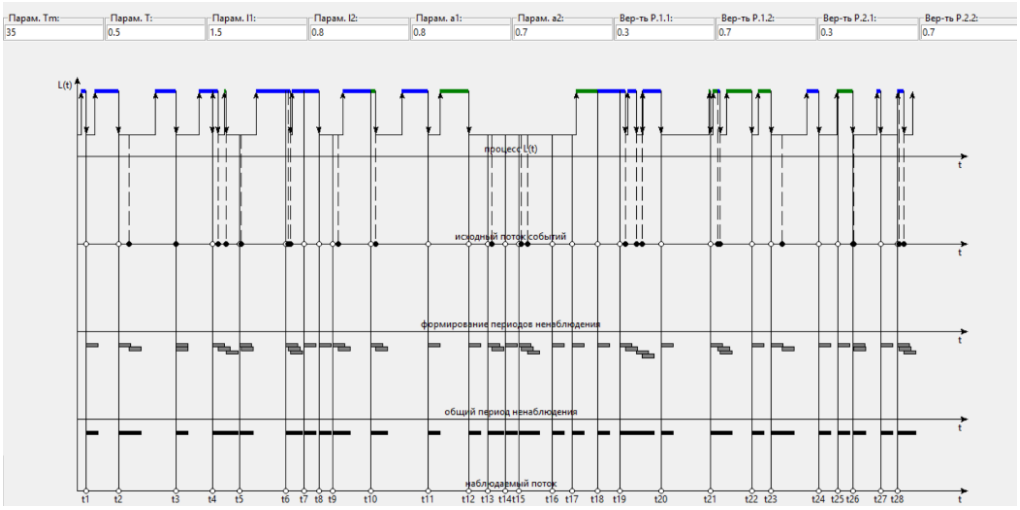


Рис. 3. Скриншот программы

4. Статистические эксперименты

ХАРАКТЕРИСТИКИ ПОТОКА

Введем случайные величины: $\boldsymbol{\tau}$ – длительность интервала между наблюдаемыми событиями; $\boldsymbol{\xi}$ – длительность периода ненаблюдаемости потока; $\boldsymbol{\eta_1}$ – длительность пребывания процесса $\lambda(t)$ в 1-м состоянии; $\boldsymbol{\eta_2}$ – длительность пребывания процесса $\lambda(t)$ во 2-м состоянии.

Проводится $N=100$ реализаций процесса $\lambda(t)$. В j-й реализации, $j=\overline{1,N}$, имеем выборку l интервалов между событиями в наблюдаемом потоке: $\tau_1^{(j)},\tau_2^{(j)},...,\tau_l^{(j)}$. Вычисляется значение оценки среднего случайной величины $\boldsymbol{\tau}$ для j-й реализации: $\hat{\tau}^{(j)}=\frac{1}{l}\sum_{i=1}^{l}\tau_i^{(j)}$. Затем рассчитывается оценка среднего случайной величины $\boldsymbol{\tau}$ по всем N реализациям: $\hat{M}(\hat{\tau})=\frac{1}{N}\sum_{j=1}^{N}\hat{\tau}^{(j)}$.

Аналогично находится значение оценки среднего для $\boldsymbol{\xi}$, $\boldsymbol{\eta_1}$, $\boldsymbol{\eta_2}$ в j-й реализации:

$$\hat{\xi}^{(j)}=\frac{1}{l}\sum_{i=1}^{l}\xi_i^{(j)},\ \hat{\eta}_1^{(j)}=\frac{1}{l}\sum_{i=1}^{l}\eta_{1i}^{(j)},\ \hat{\eta}_2^{(j)}=\frac{1}{l}\sum_{i=1}^{l}\eta_{2i}^{(j)}.$$

Затем вычисляется оценка среднего $\boldsymbol{\xi}$, $\boldsymbol{\eta_1}$, $\boldsymbol{\eta_2}$ по всем N реализациям:

$$\hat{M}(\hat{\xi})=\frac{1}{N}\sum_{j=1}^{N}\hat{\xi}^{(j)},\ \hat{M}(\hat{\eta}_1)=\frac{1}{N}\sum_{j=1}^{N}\hat{\eta}_1^{(j)},\ \hat{M}(\hat{\eta}_2)=\frac{1}{N}\sum_{j=1}^{N}\hat{\eta}_2^{(j)}.$$

Характеристиками потока будем считать:

$\hat{M}(\hat{\tau})$ – оценка средней длительности интервала между событиями в наблюдаемом потоке; $\hat{M}(\hat{\xi})$ – оценка среднего периода ненаблюдаемости потока; $\hat{M}(\hat{\eta}_1)$ – оценка средней длительности пребывания процесса $\lambda(t)$ в 1-м состоянии; $\hat{M}(\hat{\eta}_2)$ – оценка средней длительности пребывания процесса $\lambda(t)$ во 2-м состоянии.

ЭКСПЕРИМЕНТ 1. Установление стационарного режима функционирования потока.

Изменяется время моделирования T_m от 100 ед. времени до 4000 ед. времени с шагом 100. Фиксируются параметры потока. Рассматривается изменение характеристик потока. Исходные данные для эксперимента 1 представлены в табл. 1; численные результаты − в табл. 2.

Таблица 1

Исходные данные для эксперимента 1

T	λ_1	λ_2	α_1	α_2	$P_1^{(1)}(\lambda_1\|\lambda_1)$	$P_1^{(1)}(\lambda_2\|\lambda_1)$	$P_1^{(2)}(\lambda_1\|\lambda_1)$	$P_1^{(2)}(\lambda_2\|\lambda_1)$
0.1	1.5	0.9	1.1	0.2	0.3	0.7	0.4	0.6

Таблица 2

Результаты эксперимента 1

T_m	100	200	300	…	2900	3000	…	3900	4000
$\hat{M}(\hat{\tau})$	1.0403	1.0203	1.0322	…	1.0306	1.0305	…	1.0268	1.0291
$\hat{M}(\hat{\xi})$	0.1055	0.1052	0.1053	…	0.1055	0.1054	…	0.1054	0.1054
$\hat{M}(\hat{\eta}_1)$	0.5752	0.5872	0.5770	…	0.5854	0.5872	…	0.5858	0.5802
$\hat{M}(\hat{\eta}_2)$	5.1510	4.9356	5.0463	…	5.0029	5.0004	…	4.9732	4.9801

На рис. 4−7 представлены графики зависимостей характеристик $\hat{M}(\hat{\tau})$, $\hat{M}(\hat{\xi})$, $\hat{M}(\hat{\eta}_1)$ и $\hat{M}(\hat{\eta}_2)$ от времени моделирования T_m.

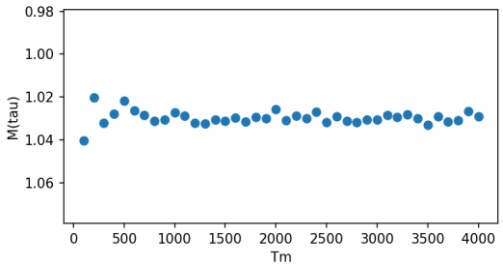


Рис. 4. График зависимости $\hat{M}(\hat{\tau})$ от T_m

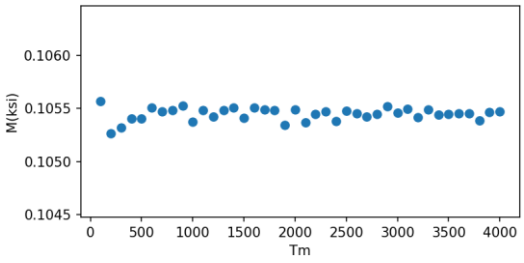


Рис. 5. График зависимости $\hat{M}(\hat{\xi})$ от T_m

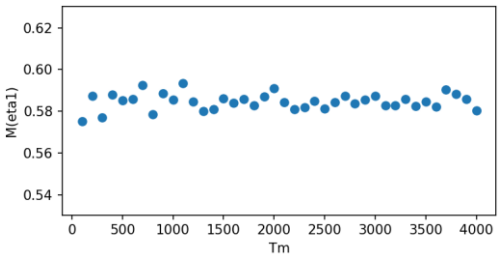


Рис. 6. График зависимости $\hat{M}(\hat{\eta}_1)$ от T_m

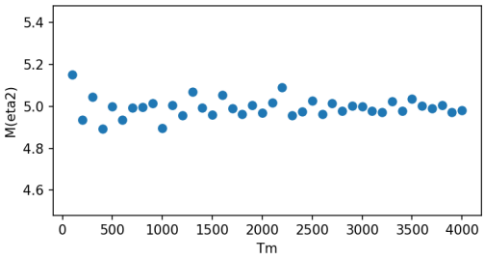


Рис. 7. График зависимости $\hat{M}(\hat{\eta}_2)$ от T_m

ВЫВОД. По численным и графическим результатам видно установление стационарного режима при времени моделирования $T_m \geq 3000$ ед. времени, т.к. оценки средних значений стремятся к постоянным величинам. В дальнейших экспериментах будет использоваться значение времени моделирования $T_m = 3000$ ед. времени.

ЭКСПЕРИМЕНТ 2. Зависимость характеристик потока от длительности мёртвого времени T.

Изменяется длительность мёртвого времени T от 0.1 ед. времени до 1.5 ед. времени с шагом 0.1. В каждой реализации задаётся $T_m = 3000$ ед. времени. Фиксируются параметры потока. Исследуется изменение характеристик потока в зависимости от значения T. Исходные данные для эксперимента 2 представлены в табл. 3; численные результаты – в табл. 4.

Таблица 3

Исходные данные для эксперимента 2

T_m	λ_1	λ_2	α_1	α_2	$P_1^{(1)}(\lambda_1 \mid \lambda_1)$	$P_1^{(1)}(\lambda_2 \mid \lambda_1)$	$P_1^{(2)}(\lambda_1 \mid \lambda_1)$	$P_1^{(2)}(\lambda_2 \mid \lambda_1)$
3000	1.5	0.9	1.1	0.2	0.3	0.7	0.4	0.6

Таблица 4

Результаты эксперимента 2

T	0.1	0.2	0.3	…	0.8	0.9	…	1.3	1.4	1.5
$\hat{M}(\hat{\tau})$	1.0283	1.1425	1.4191	…	2.1771	2.4380	…	3.7581	4.1974	4.6929
$\hat{M}(\hat{\xi})$	0.1054	0.2225	0.3532	…	1.2675	1.5206	…	2.4590	3.2843	3.7799
$\hat{M}(\hat{\eta}_1)$	0.5830	0.5864	0.5843	…	0.5820	0.5829	…	0.5839	0.5892	0.5866
$\hat{M}(\hat{\eta}_2)$	5.0235	4.9966	4.9906	…	5.0083	4.9805	…	5.0080	5.0047	5.0358

На рис. 8−11 представлены графики зависимостей вычисляемых характеристик потока $\hat{M}(\hat{\tau})$, $\hat{M}(\hat{\xi})$, $\hat{M}(\hat{\eta}_1)$, $\hat{M}(\hat{\eta}_2)$ от длительности мёртвого времени T.

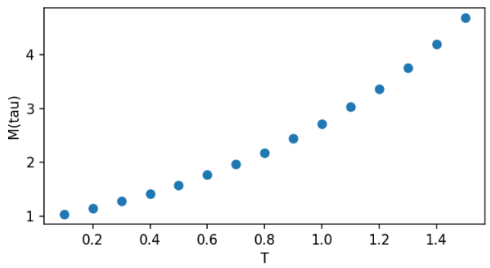


Рис. 8. График зависимости $\hat{M}(\hat{\tau})$ от T

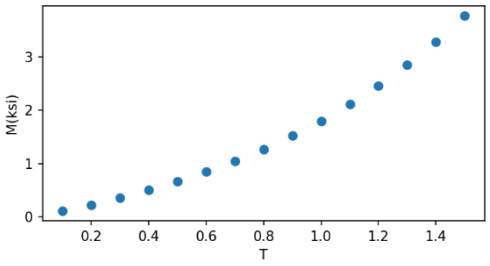


Рис. 9. График зависимости $\hat{M}(\hat{\xi})$ от T

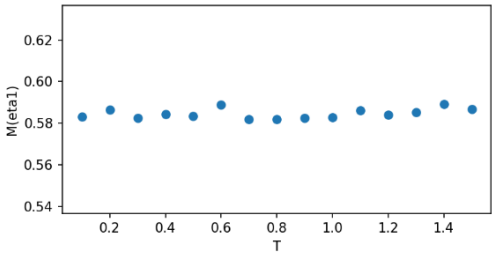


Рис. 10. График зависимости $\hat{M}(\hat{\eta}_1)$ от T

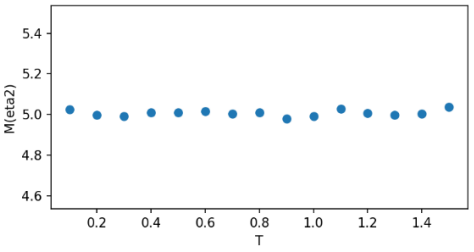


Рис. 11. График зависимости $\hat{M}(\hat{\eta}_2)$ от T

ВЫВОД. Из численных и графических результатов видно, что с ростом длительности мёртвого времени T растут значения $\hat{M}(\hat{\tau})$ и $\hat{M}(\hat{\xi})$. Рост длительности мёртвого времени влечёт за собой рост периода ненаблюдаемости, поэтому наступающие события исходного потока чаще не наблюдаются и, соответственно, значение длительности интервала между наблюдаемыми событиями увеличивается. Можно заметить, что значения $\hat{M}(\hat{\eta}_1)$ и $\hat{M}(\hat{\eta}_2)$ почти не изменяются. Такое поведение этих оценок объясняется тем, что значения случайных величин $\boldsymbol{\eta_1}$ и $\boldsymbol{\eta_2}$ не зависят непосредственно от значения длительности мёртвого времени T.

ЭКСПЕРИМЕНТ 3. Исследование зависимости $\hat{M}(\hat{\eta}_1)$ от интенсивности λ_1 процесса $\lambda(t)$ в состоянии S_1.

Приведём два случая: *в первом случае* увеличивается вероятность перехода процесса $\lambda(t)$ из 1-го во 2-е состояние по обеим случайным величинам, а *во втором случае* увеличивается вероятность остаться в 1-м состоянии по обеим случайным величинам. В обоих случаях задаётся $T_m = 3000$ ед. времени; исходные данные для эксперимента 3 зафиксированы в табл. 5 (случай 1) и в табл. 7 (случай 2). Параметр λ_1 изменяется от 0.5 до 5 с шагом 0.5. Численные результаты эксперимента 3 для *первого и второго случаев* представлены в табл. 6 и 8 соответственно.

На рис. 12 (случай 1) и рис. 13 (случай 2) представлены графики зависимостей характеристики потока $\hat{M}(\hat{\eta}_1)$ от значения параметра λ_1.

Таблица 5

Исходные данные для эксперимента 3 (случай 1)

T_m	T	λ_2	α_1	α_2	$P_1^{(1)}(\lambda_1 \mid \lambda_1)$	$P_1^{(1)}(\lambda_2 \mid \lambda_1)$	$P_1^{(2)}(\lambda_1 \mid \lambda_1)$	$P_1^{(2)}(\lambda_2 \mid \lambda_1)$
3000	0.1	0.2	1.1	0.2	0.3	0.7	0.3	0.7

Таблица 6

Результаты эксперимента 3 (случай 1)

λ_1	0.5	1.0	1.5	…	3.5	4.0	4.5	5.0
$\hat{M}(\hat{\eta}_1)$	0.8950	0.6789	0.5494	…	0.3131	0.2793	0.2553	0.2346

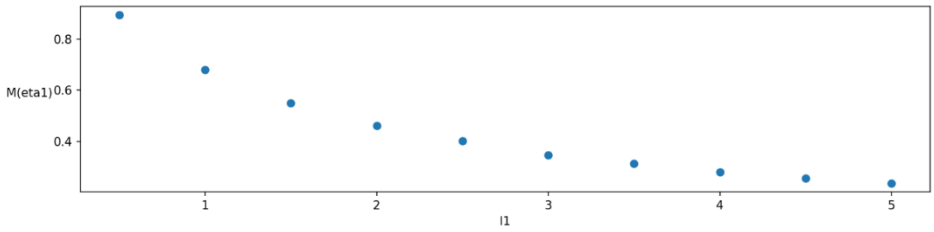


Рис. 12. График зависимости $\hat{M}(\hat{\eta}_1)$ от λ_1 (случай 1)

Таблица 7

Исходные данные для эксперимента 3 (случай 2)

T_m	T	λ_2	α_1	α_2	$P_1^{(1)}(\lambda_1 \mid \lambda_1)$	$P_1^{(1)}(\lambda_2 \mid \lambda_1)$	$P_1^{(2)}(\lambda_1 \mid \lambda_1)$	$P_1^{(2)}(\lambda_2 \mid \lambda_1)$
3000	0.1	0.2	1.1	0.2	0.7	0.3	0.7	0.3

Таблица 8

Результаты эксперимента 3 (случай 2)

λ_1	0.5	1.0	1.5	…	3.5	4.0	4.5	5.0
$\hat{M}(\hat{\eta}_1)$	2.0658	1.5897	1.2765	…	0.7210	0.6545	0.5961	0.5469

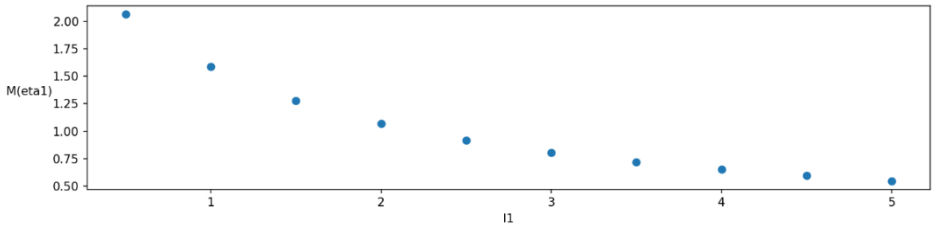


Рис. 13. График зависимости $\hat{M}(\hat{\eta}_1)$ от λ_1 (случай 2)

ВЫВОД. Из численных и графических результатов видно, что при одинаковых значениях параметра λ_1 значения $\hat{M}(\hat{\eta}_1)$ различны. Это связано с тем, что в *первом случае* вероятность перейти во 2-е состояние по обеим случайным величинам выше и, соответственно, с приходом события потока в первом состоянии процесс $\lambda(t)$ чаще уходит во 2-е состояние. Во *втором случае* вероятность остаться в 1-м состоянии по обеим случайным величинам выше и, соответственно, с приходом события процесс $\lambda(t)$ чаще остаётся в 1-м состоянии, поэтому значения $\hat{M}(\hat{\eta}_1)$ больше. Однако в обоих случаях с ростом параметра λ_1 наблюдается уменьшение значения $\hat{M}(\hat{\eta}_1)$. Это связано с тем, что при увеличении значения λ_1 учащается поток событий в 1-м состоянии и розыгрыш уйти или остаться процессу $\lambda(t)$ происходит чаще. А так как в *первом случае* вероятность

уйти выше, а во *втором случае* она меньше, но все же есть, то процесс $\lambda(t)$ уходит из 1-го состояния.

ЭКСПЕРИМЕНТ 4. Исследование зависимости $\hat{M}(\hat{\eta}_2)$ от интенсивности α_2 процесса $\lambda(t)$ в состоянии S_2.

Приведем два случая. Пусть в *первом случае* диапазон изменения α_2 от 0.1 до 1 с шагом 0.1, а во *втором* – от 0.5 до 5 с шагом 0.5. В обоих случаях задаём $T_m = 3000$ ед. времени. Зафиксируем параметры потока в табл. 9 и табл. 11. Численные результаты *первого и второго случаев* представлены, соответственно, в табл. 10 и 12.

Таблица 9

Исходные данные для эксперимента 4 (случай 1)

T_m	T	λ_1	λ_2	α_1	$P_1^{(1)}(\lambda_1 \mid \lambda_1)$	$P_1^{(1)}(\lambda_2 \mid \lambda_1)$	$P_1^{(2)}(\lambda_1 \mid \lambda_1)$	$P_1^{(2)}(\lambda_2 \mid \lambda_1)$
3000	0.1	1.5	0.2	1.1	0.4	0.6	0.6	0.4

Таблица 10

Результаты эксперимента 4 (случай 1)

α_2	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
$\hat{M}(\hat{\eta}_2)$	100.43	49.45	33.10	24.95	20.04	16.76	14.29	12.73	11.23	9.96

На рис. 14 (случай 1) и рис. 15 (случай 2) представлены графики зависимостей характеристики потока $\hat{M}(\hat{\eta}_2)$ от значения параметра α_2.

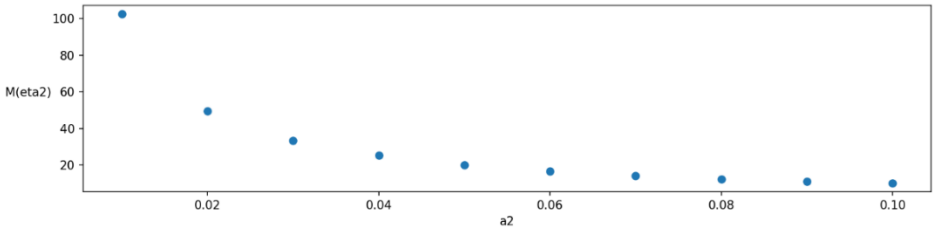


Рис. 14. График зависимости $\hat{M}(\hat{\eta}_2)$ от α_2 (случай 1)

Таблица 11

Исходные данные для эксперимента 4 (случай 2)

T_m	T	λ_1	λ_2	α_1	$P_1^{(1)}(\lambda_1 \mid \lambda_1)$	$P_1^{(1)}(\lambda_2 \mid \lambda_1)$	$P_1^{(2)}(\lambda_1 \mid \lambda_1)$	$P_1^{(2)}(\lambda_2 \mid \lambda_1)$
3000	0.1	1.5	0.2	1.1	0.4	0.6	0.6	0.4

Таблица 12

Результаты эксперимента 4 (случай 2)

α_2	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
$\hat{M}(\hat{\eta}_2)$	2.006	1.002	0.665	0.501	0.400	0.333	0.286	0.249	0.223	0.199

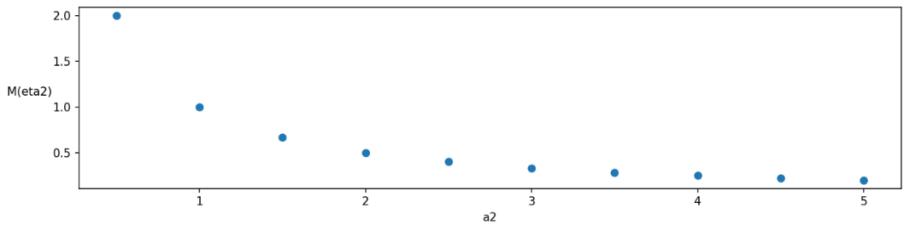


Рис. 15. График зависимости $\hat{M}(\hat{\eta}_2)$ от α_2 (случай 2)

ВЫВОД. Из численных и графических результатов видно, что с увеличением значения параметра α_2 значение $\hat{M}(\hat{\eta}_2)$ в обоих случаях уменьшается. Это связано с тем, что параметр α_2 определяет длительность интервала пребывания процесса $\lambda(t)$ во 2-м состоянии.

Заключение

В данной работе исследован полусинхронный поток событий второго порядка с двумя состояниями, функционирующий в условиях продлевающегося мёртвого времени фиксированной длительности. Построена имитационная модель данного потока в виде программного кода на языке программирования Python с использованием различных библиотек (numpy, tkinter, matplotlib и т.д.).

Анализ корректности результатов проведённых статистических экспериментов 1−4 даёт возможность утверждать, что получена работоспособная модель, не противоречащая математической модели потока.

ЛИТЕРАТУРА

1. *Erlang A.K.* The theory of probabilities and telephone conversations // Nyt Tidsskrift for Matematik. Seria B. – 1909. – Vol 20, is. B. – P. 33–39.

2. *Хинчин А.Я.* Работы по математической теории массового обслуживания. – М.: Физматгиз, 1963. – 236 с.

3. *Нежельская Л.А.* Оценка состояний дважды стохастических потоков событий. – Томск: Издательство Томского государственного университета, 2020. – 210 с.

4. *Тумашкина Д.* Оценка состояний, длительности мертвого времени и параметров распределения в полусинхронном потоке событий второго порядка // дис. … канд. физ.-мат. наук: 05.13.01. – Томск, 2021. – 154 с.

5. *Соболь И.М.* Численные методы Монте-Карло. − М.: Наука, 1973. – 312 с.

ВЫВОД УРАВНЕНИЯ МОМЕНТОВ В ЗАДАЧЕ ОЦЕНИВАНИЯ ДЛИТЕЛЬНОСТИ ПРОДЛЕВАЮЩЕГОСЯ МЁРТВОГО ВРЕМЕНИ В РЕКУРРЕНТНОМ ОБОБЩЁННОМ АСИНХРОННОМ ПОТОКЕ СОБЫТИЙ

Нежельская Л.А., Пономаренко В.Д.
Томский государственный университет
ludne@mail.ru, valya.ponomarenko.00@mail.ru

Введение

В данной работе рассматривается обобщённый асинхронный поток событий с продлевающимся мёртвым временем и выводится уравнение моментов для оценки длительности мёртвого времени. Математическая модель рассматриваемого потока является наиболее приближенной к реальности моделью потоков заявок, запросов, сообщений, протекающих в компьютерных сетях, телекоммуникационных сетях и сетях связи. Для дважды стохастического потока характерно следующее: сопровождающий случайный процесс является принципиально ненаблюдаемым и моменты наступления событий случайны. Дважды стохастические потоки делятся на два класса: к первому классу относятся потоки, интенсивность которых есть непрерывный случайный процесс [1], ко второму – потоки с интенсивностью в виде кусочно-постоянного случайного процесса с конечным числом состояний [2−4].

В большом количестве работ по теории массового обслуживания решаются задачи, когда все события входящего потока доступны для наблюдения, однако такая ситуация далеко не всегда соответствует реальности, т.к. в реальных системах входящее событие способно породить период мёртвого времени для регистрирующего прибора. В течение этого периода другие события входящего потока становятся ненаблюдаемыми, говоря иначе − теряются [5]. Мёртвое время, порождаемое наступившим событием, является в некотором роде искажающим фактором при решении различного рода задач оценива-

ния состояний потока или же его параметров по моментам наступления событий в наблюдаемом потоке.

В работах [6,7] решены задачи оценивания длительности мёртвого времени, когда наступившие в течение периода мёртвого времени события не вызывают его продления (непродлевающееся мёртвое время). Мёртвое время может иметь фиксированную длительность, либо быть случайным.

Поток, рассматриваемый в данной работе, относится ко второму классу дважды стохастических потоков, и предполагается, что событие, наступившее в течение периода мёртвого времени, хотя и не наблюдается, всё же способно продлить общий период ненаблюдаемости исходного потока (продлевающееся мёртвое время). При этом принимается, что мёртвое время имеет фиксированную длительность.

В настоящей работе, являющейся продолжением исследования, начатого в [8], выводится уравнение моментов для решения задачи оценивания длительности мёртвого времени в обобщённом асинхронном потоке событий, функционирующем в условиях продлевающегося мёртвого времени.

1. Математическая модель потока

Рассматривается обобщённый асинхронный поток событий, обладающий следующими свойствами: 1) стационарность – вероятностные свойства потока не зависят от выбранного интервала времени на временной оси; 2) ординарность – невозможность наступления двух или более событий в один и тот же момент времени; 3) наличие последействия – выражает собой взаимную зависимость поведения потока в непересекающихся между собой промежутках времени.

Сопровождающий процесс данного потока $\lambda(t)$ является принципиально ненаблюдаемым кусочно-постоянным случайным процессом с двумя состояниями S_1 и S_2; будем говорить, что имеет место состояние S_i процесса $\lambda(t)$, если $\lambda(t)=\lambda_i$, $i=1,2$, $\lambda_1>\lambda_2\geq 0$. Длительность пребывания процесса $\lambda(t)$ в состоянии S_i является случайной величиной с функцией распределения $F_i(t)=1-e^{-\alpha_i t}$, $t\geq 0$, $i=1,2$ В течение времени пребывания процесса $\lambda(t)$ в состоянии S_1 имеет место пуассоновский поток событий с параметром λ_1; в течение времени пребывания процесса $\lambda(t)$ в состоянии S_2 − пуассоновский поток событий с параметром λ_2.

В момент перехода сопровождающего случайного процесса $\lambda(t)$ из состояния S_1 в состояние S_2 с вероятностью p ($0\leq p\leq 1$) инициируется дополнительное событие потока в состоянии S_2 (сначала переход из S_1 в S_2, затем наступление дополнительного события в S_2). Аналогично при переходе сопровождающего случайного процесса $\lambda(t)$ из состояния S_2 в состояние S_1 с вероятностью q ($0\leq q\leq 1$) инициируется дополнительное событие потока в состоянии S_1 (сначала переход из S_2 в S_1, затем наступление дополнительного события в S_1).

В момент наступления каждого события потока наступает период ненаблюдаемости событий фиксированной длительности T (мёртвое время), так что другие события, наступившие в течение времени T, недоступны наблюдению. Каждое событие, ненаблюдаемое в течение мёртвого времени, вызывает продление периода ненаблюдаемости на величину T; наблюдаться будет лишь то событие, которое наступило после окончания последнего периода ненаблюдаемости.

На рис. 1 приведена схема формирования наблюдаемого потока событий и одна из реализаций сопровождающего процесса $\lambda(t)$, где λ_i – значения процесса $\lambda(t)$ в состоянии S_i, $i=1,2$; $t_1,t_2,\ldots$ – моменты времени наступления событий в наблюдаемом потоке. Наблюдаемые события обозначены незакрашенными кружками, а ненаблюдаемые события, т.е. недоступные наблюдению из-за наличия мёртвого времени, обозначены за-

крашенными кружками. Штриховкой обозначен период ненаблюдаемости. Общий период ненаблюдаемости ξ – случайная величина.

Матрицы инфинитезимальных характеристик сопровождающего процесса $\lambda(t)$ имеют вид [9]:

$$\mathbf{D_0} = \begin{Vmatrix} -(\lambda_1 + \alpha_1) & (1-p)\alpha_1 \\ (1-q)\alpha_2 & -(\lambda_2 + \alpha_2) \end{Vmatrix}, \ \mathbf{D_1} = \begin{Vmatrix} \lambda_1 & p\alpha_1 \\ q\alpha_2 & \lambda_2 \end{Vmatrix}.$$

Элементами матрицы $\mathbf{D}_1$ являются интенсивности переходов процесса $\lambda(t)$ из состояния в состояние с наступлением события потока. Недиагональные элементы матрицы $\mathbf{D}_0$ – интенсивности переходов процесса $\lambda(t)$ из состояния в состояние без наступления события. Диагональные элементы матрицы $\mathbf{D}_0$ – интенсивности выхода процесса $\lambda(t)$ из своих состояний, взятые с противоположным знаком.

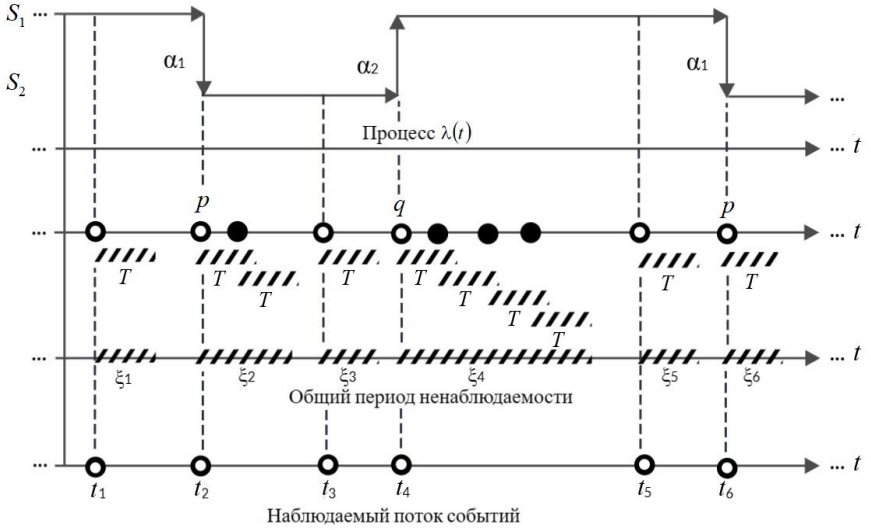


Рис. 1. Формирование наблюдаемого потока событий

2. Двумерная плотность вероятности длительностей двух смежных интервалов. Условия рекуррентности

Пусть $t_1, t_2, ..., t_k, ...$ – моменты времени наступления событий в потоке.

Лемма 1. Последовательность $\{\lambda(t_k)\}$, образуемая совокупностью моментов наступления событий $t_1, t_2, ..., t_k, ...$, является вложенной цепью Маркова [9].

Обозначим $\tau_k = t_{k+1} - t_k$, $\tau_k \geq 0$, – значение длительности k-го интервала между соседними событиями t_k и t_{k+1}, $k = 1, 2, ...$ наблюдаемого потока. Рассматривается стационарный режим функционирования потока, поэтому для плотности вероятности значений τ_k справедливо $p(\tau_k) = p(\tau)$, $\tau_k \geq 0$, $\forall k$. Это позволяет без ограничения общности считать момент наступления события t_k равным нулю или, что то же самое, момент наступления события есть $\tau = 0$.

Теорема 1. Плотность вероятности значений длительности интервала между соседними событиями в обобщённом асинхронном потоке имеет вид [10]:

$$p(\tau) = \gamma z_1 e^{-z_1\tau} + (1-\gamma) z_2 e^{-z_2\tau},\ \tau \ge 0,$$

$$\gamma = \frac{1}{z_2 - z_1}\left[-z_1 + \lambda_1 + \lambda_2 - \frac{(\alpha_1+\alpha_2)(\lambda_1\lambda_2 - pq\alpha_1\alpha_2)}{\alpha_1\lambda_2 + \alpha_2\lambda_1 + (p+q)\alpha_1\alpha_2}\right],$$

$$z_1 = \frac{1}{2}\left[\lambda_1 + \alpha_1 + \lambda_2 + \alpha_2 - \sqrt{(\lambda_1 + \alpha_1 - \lambda_2 - \alpha_2)^2 + 4\alpha_1\alpha_2(1-p)(1-q)}\right], \quad (1)$$

$$z_2 = \frac{1}{2}\left[\lambda_1 + \alpha_1 + \lambda_2 + \alpha_2 + \sqrt{(\lambda_1 + \alpha_1 - \lambda_2 - \alpha_2)^2 + 4\alpha_1\alpha_2(1-p)(1-q)}\right],$$

$$z_2 > z_1 > 0,\ z_2 - z_1 \ne 0.$$

Теорема 2. Совместная плотность вероятности $p(\tau_1, \tau_2)$ длительностей двух соседних интервалов для обобщённого асинхронного потока событий имеет вид [10]:

$$p(\tau_1, \tau_2) = p(\tau_1)\, p(\tau_2) + \gamma(1-\gamma)\frac{\lambda_1\lambda_2 - pq\alpha_1\alpha_2}{z_1 z_2}\left(z_1 e^{-z_1\tau_1} - z_2 e^{-z_2\tau_1}\right)\left(z_1 e^{-z_1\tau_2} - z_2 e^{-z_2\tau_2}\right),$$

$$\tau_1 \ge 0,\ \tau_2 \ge 0,$$

$$\gamma(1-\gamma) = \frac{\alpha_1\alpha_2 z_1 z_2 (\lambda_1 - \lambda_2 + p\alpha_1 - q\alpha_2)(\lambda_1 - \lambda_2 + q\alpha_1 - p\alpha_2)}{(z_2 - z_1)^2 (\lambda_1\alpha_2 + \lambda_2\alpha_1 + (p+q)\alpha_1\alpha_2)^2}, \quad (2)$$

$$z_2 > z_1 > 0,\ z_2 - z_1 \ne 0,$$

где z_1, z_2 определены в (1); $p(\tau_k)$ определены в (1) для $\tau = \tau_k$, $k = 1, 2$.

Из анализа (2) вытекают условия рекуррентности и соответствующие плотности $p(\tau)$:

1. Если $\lambda_1 - \lambda_2 + p\alpha_1 - q\alpha_2 = 0$, то $\gamma(1-\gamma) = 0$ и $p(\tau_1, \tau_2)$ факторизуется. Подставляя условие $\lambda_1 - \lambda_2 + p\alpha_1 - q\alpha_2 = 0$ в (1), находим плотность вероятности значений длительности интервала между соседними событиями $p(\tau)$ в виде $p(\tau) = (p\alpha_1 + \lambda_1) e^{-(p\alpha_1 + \lambda_1)\tau}$, $\tau \ge 0$. В этом случае обобщённый асинхронный поток вырождается в простейший.

2. Если $\lambda_1 - \lambda_2 + q\alpha_1 - p\alpha_2 = 0$, то $\gamma(1-\gamma) = 0$ и $p(\tau_1, \tau_2)$ факторизуется. Подставляя условие $\lambda_1 - \lambda_2 + q\alpha_1 - p\alpha_2 = 0$ в (1), находим плотность вероятности значений длительности интервала между соседними событиями $p(\tau)$ в виде $p(\tau) = (q\alpha_1 + \lambda_1) e^{-(q\alpha_1 + \lambda_1)\tau}$, $\tau \ge 0$. Как и в предыдущем случае, обобщённый асинхронный поток вырождается в простейший.

3. Если $\lambda_1\lambda_2 - pq\alpha_1\alpha_2 = 0$, то совместная плотность вероятности $p(\tau_1, \tau_2)$ факторизуется. Подставляя условие $\lambda_1\lambda_2 - pq\alpha_1\alpha_2 = 0$ в (1), находим плотность вероятности значений длительности интервала между соседними событиями $p(\tau)$:

$$p(\tau) = \gamma z_1 e^{-z_1\tau} + (1-\gamma) z_2 e^{-z_2\tau},\ \tau \ge 0,\ \gamma = \frac{z_2 - \lambda_1 - \lambda_2}{z_2 - z_1},\ z_2 - z_1 \ne 0. \quad (3)$$

Далее рассматривается рекуррентный поток в случае 3.

3. Преобразование Лапласа плотности вероятности общего периода ненаблюдаемости

Пусть ξ – длительность общего периода ненаблюдаемости в рекуррентном потоке. Последовательность $t_1, t_2, \ldots$ моментов наступления событий в наблюдаемом потоке образует вложенную цепь Маркова, и рекуррентность наблюдаемого потока сохраняется.

Переобозначим в (3) плотность $p(\tau)$ на $\tilde{p}(x)$; получим

$$\tilde{p}(x) = \gamma z_1 e^{-z_1 x} + (1-\gamma) z_2 e^{-z_2 x},\ x \geq 0. \quad (4)$$

Теорема 3. Преобразование Лапласа плотности вероятности значений длительности общего периода ненаблюдаемости в рекуррентном обобщённом асинхронном потоке с продлевающимся мёртвым временем в случае $z_2 - z_1 \neq 0$ имеет вид

$$g_\xi(s) = \frac{\varphi_0(T)}{e^{sT}}\left[1 - \left[(1-e^{-(s+z_1)T})z_1 \frac{z_2 - \lambda_1 - \lambda_2}{(s+z_1)(z_2-z_1)} + \left(1-e^{-(s+z_2)T}\right) z_2 \frac{-z_1 + \lambda_1 + \lambda_2}{(s+z_2)(z_2-z_1)}\right]\right]^{-1}. \quad (5)$$

Доказательство. Рассмотрим функцию Пальма $\varphi_0(T) = \int_T^\infty \tilde{p}(x)dx$ – вероятность того, что на интервале $(0,T)$ событий рекуррентного потока не наступит при условии, что в начальный момент времени интервала $(0,T)$ событие наступило. Вычислив интеграл, находим

$$\varphi_0(T) = \gamma e^{-z_1 T} + (1-\gamma) e^{-z_2 T}, \quad (6)$$

где величина γ определена в (3), z_1, z_2 определены в (1).

Преобразование Лапласа плотности вероятности значений длительности общего периода ненаблюдаемости в рекуррентном дважды стохастическом потоке событий, функционирующем в условиях продлевающегося мёртвого времени, имеет вид [10]:

$$g_\xi(s) = \frac{\varphi_0(T)}{e^{sT}}\left[1 - \int_0^T e^{-sx}\tilde{p}(x)dx\right]^{-1}. \quad (7)$$

Подставляя (4), (6) в (7) и вычисляя интеграл, получаем (5). *Теорема доказана.*

Лемма 2. Математическое ожидание длительности общего периода ненаблюдаемости ξ в рекуррентном обобщённом асинхронном потоке событий с продлевающимся мёртвым временем имеет вид

$$M\xi = \frac{1}{z_1 z_2 \varphi_0(T)}\left[\gamma z_2 + (1-\gamma) z_1 - \gamma z_2 e^{-z_1 T} - (1-\gamma) z_1 e^{-z_2 T}\right]. \quad (8)$$

Доказательство. Нетрудно показать, что с использованием вида $g_\xi(s)$, определённого в (7), математическое ожидание ξ запишется в виде

$$M\xi = -g_\xi'(s)\big|_{s=0} = T + \frac{1}{\varphi_0(T)}\int_0^T x\tilde{p}(x)dx. \quad (9)$$

Подставляя (4) и (6) в (9), находим (8). *Лемма доказана.*

4. Преобразование Лапласа плотности вероятности длительности интервала между соседними событиями в наблюдаемом потоке

Рассмотрим интервал времени (t_k, t_{k+1}), значение длительности которого есть $\tau_k = t_{k+1} - t_k$. С другой стороны, длительность этого интервала равна $\boldsymbol{\tau} = \boldsymbol{\xi} + \boldsymbol{\eta}$, где $\boldsymbol{\eta}$ – длительность интервала между моментом окончания общего периода ненаблюдаемости и моментом t_{k+1}. Величины $\boldsymbol{\eta}$ и $\boldsymbol{\xi}$ являются зависимыми. Тогда плотность вероятности $p(\tau)$ запишется в виде

$$p(\tau) = \int_0^\tau p(\xi) p(\eta|\xi) d\xi = \int_0^\tau p(\xi) p(\tau - \xi|\xi) d\xi. \quad (10)$$

Теорема 4. Преобразование Лапласа плотности вероятности значений длительности интервала между соседними событиями в рекуррентном обобщённом асинхронном потоке с продлевающимся мёртвым временем в случае $z_2 - z_1 \neq 0$ имеет вид

$$g_\tau(s)=\left(z_1+s\right)^{-1}\left(z_2+s\right)^{-1}\left(\alpha_1+\alpha_2\right)^{-1}\Big[z_1z_2\left(\alpha_1+\alpha_2\right)g_\xi(s)+$$
$$+s\lambda_1\lambda_2\frac{\lambda_1-\lambda_2+q\alpha_1-p\alpha_2}{\lambda_1\alpha_2+\lambda_2\alpha_1+(p+q)\alpha_1\alpha_2}\left(\lambda_1+p\alpha_1-\lambda_2-q\alpha_2\right)g_\xi\left(\lambda_1+\lambda_2+s\right)\Big], \qquad (11)$$

где функция $g_\xi(s)$ определена формулой (5), $g_\xi\left(\lambda_1+\lambda_2+s\right)$ определена формулой (5), в которой нужно заменить s на $\lambda_1+\lambda_2+s$.

Доказательство. Пусть момент наступления события в наблюдаемом потоке есть $\tau=0$. Рассмотрим интервал $(0,\tau)=(0,\xi+\eta)$. Зафиксируем ξ. В формуле (10) условная плотность $p\left(\tau-\xi\,|\,\xi\right)$, $\tau\ge\xi$, по смыслу совпадает с плотностью $p_T(\tau)=p\left(\tau\,|\,T\right)$, $\tau\ge T$ для обобщённого асинхронного потока событий, функционирующего в условиях непродлевающегося мёртвого времени T [10]. Выполнив достаточно трудоёмкие преобразования и учитывая условие рекуррентности $\lambda_1\lambda_2-pq\alpha_1\alpha_2=0$, находим $p\left(\tau-\xi\,|\,\xi\right)$ в виде

$$p(\tau-\xi|\xi)=\begin{cases}0,\ 0\le\tau<\xi,\\ \Gamma(\xi)z_1e^{-z_1(\tau-\xi)}+(1-\Gamma(\xi))z_2e^{-z_2(\tau-\xi)},\ \tau\ge\xi,\end{cases}$$
$$\Gamma(\xi)=\frac{1}{z_2-z_1}\Big[z_2-\left(\lambda_1+p\alpha_1\right)\pi_1(\xi)-\left(\lambda_2+q\alpha_2\right)\pi_2(\xi)\Big],\ z_2-z_1\ne0, \qquad (12)$$
$$\pi_1(\xi)=\pi_1\left[1+\alpha_1\frac{\lambda_1-\lambda_2+q\alpha_1-p\alpha_2}{\alpha_2\lambda_1+\alpha_1\lambda_2+(p+q)\alpha_1\alpha_2}e^{-(\alpha_1+\alpha_2)\xi}\right],$$
$$\pi_2(\xi)=\pi_2\left[1-\alpha_1\frac{\lambda_1-\lambda_2+q\alpha_1-p\alpha_2}{\alpha_2\lambda_1+\alpha_1\lambda_2+(p+q)\alpha_1\alpha_2}e^{-(\alpha_1+\alpha_2)\xi}\right].$$

Найдём преобразование Лапласа плотности $p(\tau)$, используя (10) и (12). Имеем

$$g_\tau(s)=\int_0^\infty e^{-s\tau}p(\tau)d\tau=$$
$$=\int_0^\infty e^{-s\tau}\left[\int_0^\tau p(\xi)p(\tau-\xi|\xi)d\xi\right]d\tau=\int_0^\infty p(\xi)[\int_\xi^\infty e^{-s\tau}p(\tau-\xi|\xi)d\tau]d\xi=$$
$$=\int_0^\infty p(\xi)[\int_\xi^\infty e^{-s\tau}\Gamma(\xi)z_1e^{-z_1(\tau-\xi)}d\tau+\int_\xi^\infty e^{-s\tau}(1-\Gamma(\xi))z_2e^{-z_2(\tau-\xi)}d\tau]d\xi.$$

В выражении для $g_\tau(s)$ выполним замену переменных: $\tau-\xi=\eta$, $\tau=\xi+\eta$; при этом $\tau:(\xi;\infty)$, $\eta:(0;\infty)$. Тогда

$$g_\tau(s)=\int_0^\infty p(\xi)\left[\int_0^\infty e^{-s(\eta+\xi)}z_1\Gamma(\xi)e^{-z_1\eta}d\eta+\int_0^\infty e^{-s(\eta+\xi)}z_2(1-\Gamma(\xi))e^{-z_2\eta}d\eta\right]d\xi.$$

Подставляя явный вид $\Gamma(\xi)$, определённый в (12), и выполняя интегрирование, приходим к (11). *Теорема доказана.*

5. Вывод уравнения моментов

Преобразование (11) позволяет получить начальные моменты $M\left(\boldsymbol{\tau}^l\right)=(-1)^l g_\tau^{(l)}(s)\big|_{s=0}$, $l=1,2...$. Вычисляя производную от (11) по s в точке $s=0$, взятую со знаком минус, получаем

$$M(\boldsymbol{\tau}) = \frac{1}{z_1 z_2}\left(z_1 + z_2 - \frac{z_1 z_2}{\alpha_1 + \alpha_2}\right) - \frac{\alpha_1\alpha_2\left(\lambda_1 - \lambda_2 + q\alpha_1 - p\alpha_2\right)\left(\lambda_1 - \lambda_2 + p\alpha_1 - q\alpha_2\right)}{\left(\alpha_1 + \alpha_2\right)\left(z_1 z_2\right)^2} \times$$
$$\times \frac{\gamma e^{-(z_1+\alpha_1+\alpha_2)T} + (1-\gamma)e^{-(z_2+\alpha_1+\alpha_2)T}}{\left[1 - \frac{\gamma z_1}{z_1 + \alpha_1 + \alpha_2}\left(1 - e^{-(z_1+\alpha_1+\alpha_2)T}\right) - \frac{(1-\gamma)z_2}{z_2 + \alpha_1 + \alpha_2}\left(1 - e^{-(z_2+\alpha_1+\alpha_2)T}\right)\right]} + M\xi. \quad (13)$$

Будем решать задачу оценивания длительности мёртвого времени T методом моментов. Введём статистику $C_l = \frac{1}{n}\sum_{k=1}^{n}\tau_k^l,\ ,\ k=\overline{1,n}$, где $\tau_k = t_{k+1} - t_k$ – значение длительности интервала между моментами t_k и t_{k+1} наступления событий в рекуррентном обобщённом асинхронном потоке с продлевающимся мёртвым временем.

Предположим, что параметры α_1, α_2, λ_1, λ_2, p, q являются известными. При количестве наблюдений $n \to \infty$ выборочный момент C_1 стремится к теоретическому моменту $M(\boldsymbol{\tau})$ [11]. Тогда для оценки длительности мёртвого времени T имеем уравнение моментов

$$M(\boldsymbol{\tau}) = C_1, \quad (14)$$

где функция $M(\boldsymbol{\tau})$ определена в (13).

Обозначим левую часть (14) через $f(T)$. Численно можно показать, что функция $f(T)$, $T \ge 0$, является возрастающей. Уравнение $f(T) = C_1$ решается численно на интервале $0 \le T \le \tau_{\min}$, $\tau_{\min} = \min\{\tau_k\}$, $k = \overline{1,n}$.

Возможные ситуации поведения функции $f(T)$ приведены на рис. 2–5.

1. $f(T=0) > C_1$; в качестве значения оценки $\hat{\mathbf{T}}$ выбирается $\hat{T} = 0$ (рис. 2).

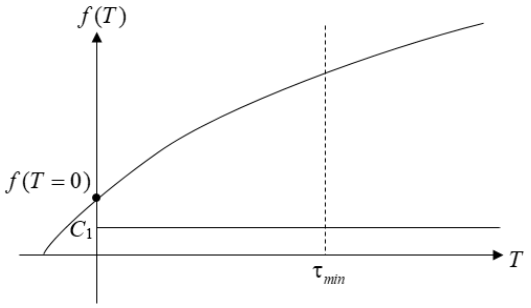


Рис. 2. Поведение функции $f(T)$, $f(0) > C_1$

2. $f(T=0) = C_1$; в качестве значения оценки $\hat{\mathbf{T}}$ выбирается $\hat{T} = 0$ (рис. 3).

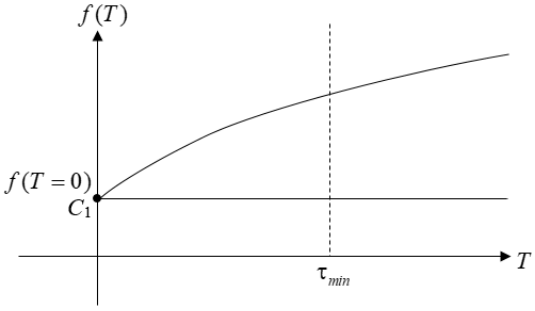


Рис. 3. Поведение функции $f(T)$, $f(0) = C_1$

3. $f(T=0) < C_1$; возможны случаи:

a) корень уравнения моментов (14) попадает в полуинтервал $\left(0, \tau_{\min}\right]$, тогда в качестве значения оценки $\hat{\mathbf{T}}$ выбирается $\hat{T} = \hat{T}^*$ (рис. 4).

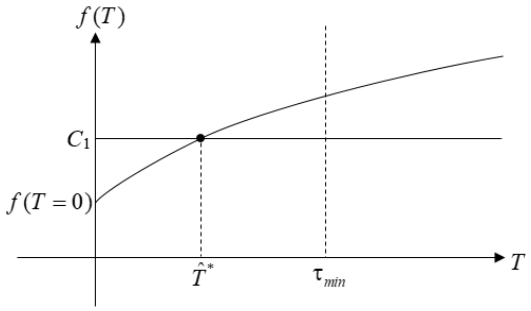


Рис. 4. Поведение функции $f(T)$, $f(0) < C_1$, $\tau_{\min} > \hat{T}^*$

b) корень уравнения (14) больше, чем $\tau_{\min}$, тогда в качестве значения оценки $\hat{\mathbf{T}}$ выбирается $\hat{T} = \tau_{\min}$ (рис. 5).

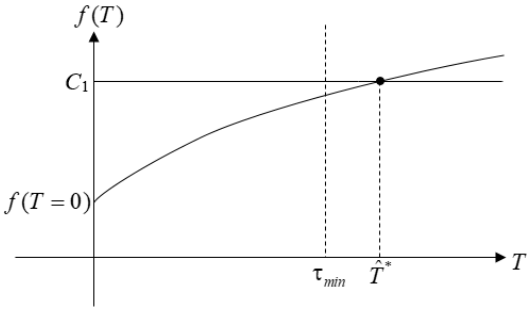


Рис. 5. Поведение функции $f(T)$, $f(0) < C_1$, $\tau_{\min} \le \hat{T}^*$

Заключение

В данной работе рассмотрен дважды стохастический обобщённый асинхронный поток случайных событий с двумя состояниями, функционирующий в стационарном режиме в условиях продлевающегося мёртвого времени фиксированной длительности. Построена математическая модель данного потока. Получено преобразование Лапласа плотности вероятности значений длительности интервалов между соседними событиями в рекуррентном обобщённом асинхронном потоке (11), с использованием которого выведено уравнение моментов (14) для решения задачи оценивания длительности мёртвого времени T.

ЛИТЕРАТУРА

1. *Cox D.R.* The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables // Proc. Cambr. Phil. Soc. – 1955. – V. 51. – No. 3. – P. 433–441.

2. *Башарин Г.П., Кокотушкин В.А., Наумов В.А.* О методе эквивалентных замен расчета фрагментов сетей связи. Ч. 1 // Изв. АН СССР. Техн. кибернетика. – 1979. – № 6. – С. 92–99.

3. *Neuts M.F.* A versatile Markov point process // Journal of Applied Probability. – 1979. –V. 16. – P. 764–779.

4. *Lucantoni D.M.* New results on single server with a bath Markovian arrival process // Stochastic Models. – 1991. – V. 7. – P. 1–46.

5. *Апанасович В.В., Коляда А.А., Чернявский А.Ф.* Статистический анализ случайных потоков в физическом эксперименте. – Минск: Университетское, 1988. – 256 с.

6. *Горцев А.М., Леонова М.А., Нежельская Л.А.* Сравнение МП- и ММ-оценок длительности мертвого времени в обобщенном асинхронном потоке событий // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. – 2013. – № 4 (25). – С. 32–42.

7. *Горцев А.М., Калягин А.А., Нежельская Л.А.* Оценка максимального правдоподобия длительности мёртвого времени в обобщённом полусинхронном потоке // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. – 2015. – № 1 (25). – С. 27–37.

8. *Нежельская Л.А., Пономаренко В.Д.* Имитационное моделирование обобщённого асинхронного потока событий с продлевающимся мёртвым временем // Труды Томского государственного университета. Сер. физико-математическая. – 2023. – Т. 308. – С. 27 − 35.

9. *Нежельская Л.А.* Оценка состояний дважды стохастических потоков событий: учебное пособие. – Томск: Издательство Томского государственного университета, 2020. – 210 с.

10. *Нежельская Л.А.* Оценка состояний и параметров дважды стохастических потоков событий: дис. … д-ра физ.-мат. наук. Томск, 2016. – 341 с.

11. *Малинковский Ю.В.* Теория вероятностей и математическая статистика (Часть 2. Математическая статистика). – Гомель: УО «ГГУ им. Ф. Скорины», 2004. – 146 с.

ЧИСЛЕННОЕ РЕШЕНИЕ УРАВНЕНИЯ МОМЕНТОВ ДЛЯ НАХОЖДЕНИЯ ОЦЕНКИ ДЛИТЕЛЬНОСТИ ПРОДЛЕВАЮЩЕГОСЯ МЕРТВОГО ВРЕМЕНИ В РЕКУРРЕНТНОМ ОБОБЩЕННОМ ПОЛУСИНХРОННОМ ПОТОКЕ СОБЫТИЙ В ОСОБОМ СЛУЧАЕ

Нежельская Л.А., Степаненко И.Д.
Томский государственный университет
ludne@mail.ru, 97step@mail.ru

Введение

Системы массового обслуживания (СМО) являются широко распространенными математическими моделями реальных физических, технических, экономических, информационных систем и сетей. СМО включают: 1) множество входящих потоков событий; 2) множество обслуживающих приборов; 3) множество бункеров (накопителей, очередей); 4) дисциплины обслуживания.

Первый элемент, определяющий СМО, включает в себя множество различных типов потоков. Первым из таких потоков для исследования и применения в теории массового обслуживания (ТМО) был простейший (стационарный пуассоновский) поток событий. Однако в настоящее время модель простейшего потока является непригодной для описания реальных информационных потоков сообщений (запросов, заявок, событий) в телекоммуникационных сетях. Это связано с тем, что математическая модель простейшего потока не учитывает разнородность передаваемых данных и их возможную взаимную корреляцию [1]. Таким образом, требования практики послужили стимулом к рассмотрению дважды стохастических потоков (коррелированных потоков) [2−4], которые являются широко используемой математической моделью реальных потоков сообщений в телекоммуникационных системах, глобальных компьютерных сетях, спутниковых сетях связи.

Авторы работ по ТМО исследуют, как правило, математические модели потоков событий, когда все события потока доступны наблюдению. В реальности же зарегистрированное событие может создать период мертвого времени для регистрирующего прибора (период ненаблюдаемости) [5], в течение которого другие события потока становятся недоступными для регистрирующего прибора (теряются). При этом наступившее в течение периода мертвого времени событие может как продлить общий период ненаблюдаемости исходного потока (продлевающееся мертвое время), так и не продлевать его (непродлевающееся мертвое время).

Таким образом, мертвое время выступает искажающим фактором при решении задач оценивания как состояний потока [6], так и его параметров [7−9].

В данной работе находится оценка длительности мертвого времени в рекуррентном обобщенном полусинхронном потоке событий с продлевающимся мертвым временем в особом случае соотношения параметров потока как решение уравнения моментов. На имитационной модели потока проводятся статистические эксперименты с целью уста-

новления качества получаемых оценок и приводится анализ полученных численных результатов оценивания.

1. Постановка задачи

Рассматривается дважды стохастический обобщенный полусинхронный поток событий, сопровождающий процесс которого есть кусочно-постоянный принципиально ненаблюдаемый случайный процесс $\lambda(t)$ с двумя состояниями: если $\lambda(t)=\lambda_1$, будем говорить, что имеет место первое состояние (S_1) процесса $\lambda(t)$ (потока), если $\lambda(t)=\lambda_2$, то второе состояние (S_2) процесса $\lambda(t)$ (потока), где $\lambda_1>\lambda_2\geq 0$. В течение временного интервала, когда $\lambda(t)=\lambda_i$, имеет место пуассоновский поток событий с параметром λ_i, $i=1,2$. Переход из первого состояния процесса $\lambda(t)$ во второе ($S_1\to S_2$) возможен только в момент наступления события, при этом переход осуществляется с вероятностью p ($0<p\leq 1$); с вероятностью $1-p$ процесс $\lambda(t)$ остается в первом состоянии. Тогда длительность временного интервала, на котором значение процесса $\lambda(t)=\lambda_1$, – участка стационарности процесса $\lambda(t)$ в первом состоянии – есть случайная величина с экспоненциальной функцией распределения $F_1(t)=1-e^{-p\lambda_1 t}$, $t\geq 0$ [4]. При этом длительность пребывания процесса $\lambda(t)$ во втором состоянии распределена по экспоненциальному закону $F_2(t)=1-e^{-\alpha_2 t}$, $t\geq 0$. Переход из второго состояния процесса $\lambda(t)$ в первое состояние ($S_2\to S_1$) может осуществляться в произвольный момент времени, не связанный с моментом наступления события потока. При переходе процесса $\lambda(t)$ из второго состояния в первое инициируется с вероятностью δ ($0<\delta\leq 1$) дополнительное событие в первом состоянии (т.е. сначала осуществляется переход, а затем инициируется дополнительное событие). При этом блочная матрица инфинитезимальных характеристик процесса $\lambda(t)$ принимает вид [4]:

$$\mathbf{D}=\left\|\begin{matrix}-\lambda_1 & 0\\ (1-\delta)\alpha_2 & -(\lambda_2+\alpha_2)\end{matrix}\right|\left.\begin{matrix}(1-p)\lambda_1 & p\lambda_1\\ \delta\alpha_2 & \lambda_2\end{matrix}\right\|=\|\mathbf{D_0}|\mathbf{D_1}\|.$$

Элементами матрицы $\mathbf{D_1}$ являются интенсивности переходов процесса $\lambda(t)$ из состояния в состояние с наступлением события. Недиагональные элементы матрицы $\mathbf{D_0}$ – интенсивности переходов из состояния в состояние без наступления событий. Диагональные элементы матрицы $\mathbf{D_0}$ – интенсивности выхода процесса $\lambda(t)$ из своих состояний, взятые с противоположным знаком.

Утверждение 1.1. Для обобщенного полусинхронного потока событий сопровождающий случайный процесс $\lambda(t)$ является марковским.

Утверждение 1.2. Моменты наступления событий $t_1,t_2,...,t_k,...$ в обобщенном полусинхронном потоке порождают вложенную цепь Маркова $\{\lambda(t_k)\}$.

Доказательства утверждений представлены в [4].

Будем рассматривать обобщенный полусинхронный поток событий при его неполной наблюдаемости, т.е. когда не все события потока доступны наблюдению. Каждое зарегистрированное в потоке событие порождает период ненаблюдаемости фиксированной длительности T (мертвое время), другие события, произошедшие в этот период, недоступны наблюдению (теряются). Хотя события и не наблюдаются в течение периода мертвого времени, каждое из них вызывает продление периода ненаблюдаемости на ту же величину T. Следующее наблюдаемое событие регистрируется после окончания последнего периода ненаблюдаемости и снова порождает период мертвого времени длительности T. Таким образом, общий период ненаблюдаемости является случайной величиной.

На рис. 1 представлен пример одной из возможных реализаций процесса $\lambda(t)$ и наблюдаемого потока событий; 1,2 − состояния случайного процесса $\lambda(t)=\lambda_i$, $i=1,2$; черные точки − моменты наступления ненаблюдаемых событий; ξ − значения периодов ненаблюдаемости; $t_1, t_2, \ldots$ − моменты наступления событий в наблюдаемом потоке.

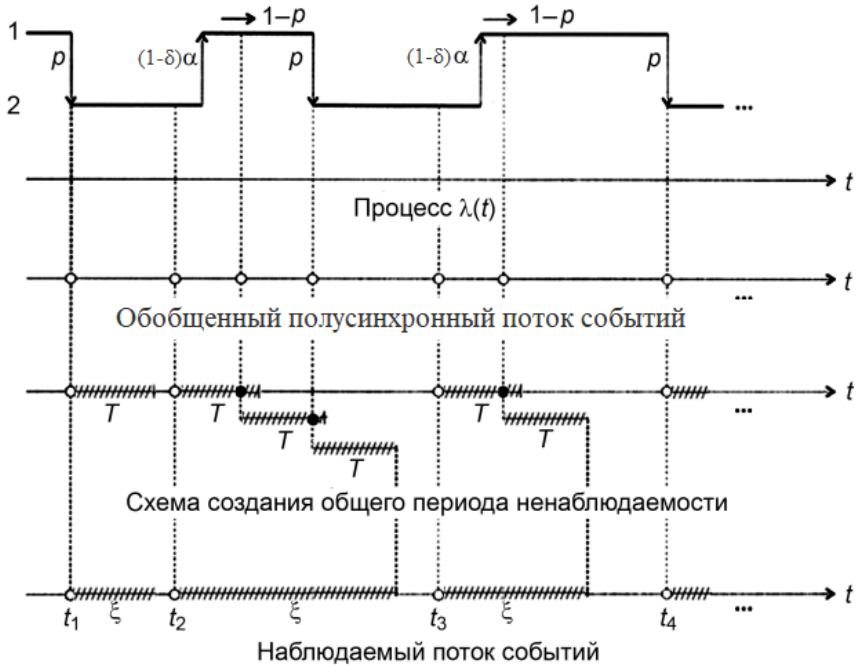


Рис. 1. Реализация обобщенного полусинхронного потока
с продлевающимся мертвым временем фиксированной длительности T

2. Уравнение моментов для оценки длительности мертвого времени

В данном разделе работы выписано уравнение моментов для нахождения оценки длительности мертвого времени T для особого случая соотношения параметров $\lambda_1-(\lambda_2+\alpha_2)=0$.

Введем статистику $C_m=\frac{1}{n}\sum_{k=1}^{n}\tau_k^{\,m}$, где $\tau_k=t_{k+1}-t_k$ − значение длительности интервала (t_k,t_{k+1}) между соседними событиями t_k и t_{k+1}, $k=\overline{1,n}$, в наблюдаемом потоке. Зафиксируем параметры потока λ_1, λ_2, α_2, δ, p. Уравнение моментов, позволяющее оценить длительность мертвого времени T, имеет вид $M\boldsymbol{\tau}=C_1$, где статистика C_1 при большом количестве наблюдений n стремится к начальному теоретическому моменту первого порядка $M\boldsymbol{\tau}$ [10]. Используя явный вид $M\boldsymbol{\tau}$, полученный в [11], выпишем уравнение моментов для оценивания T в рекуррентном обобщенном полусинхронном потоке событий с продлевающимся мертвым временем:

$$\Big(\lambda_1\big(1-(1+p\alpha_2(1-\delta)T)e^{-\lambda_1 T}\big)+p\alpha_2(1-\delta)\big(1-e^{-\lambda_1 T}\big)\Big)\frac{e^{\lambda_1 T}}{\lambda_1^{\,2}}\big(1+p\alpha_2(1-\delta)T\big)^{-1}+$$

$$+\frac{2}{\lambda_1}+\frac{(1+p\alpha_2(1-\delta)T)e^{-(\lambda_1+p\lambda_1+\alpha_2)T}}{\lambda_1^{\,2}(p\lambda_1+\alpha_2)(p\alpha_2(1-\delta))^{-2}}\Bigg[1-\frac{\lambda_1^{\,2}+(\lambda_1-p\alpha_2(1-\delta))(p\lambda_1+\alpha_2)}{(\lambda_1+p\lambda_1+\alpha_2)^2}\times \qquad (1)$$

$$\times\big(1-e^{-(\lambda_1+p\lambda_1+\alpha_2)T}\big)+\frac{p\alpha_2(1-\delta)\lambda_1}{\lambda_1+p\lambda_1+\alpha_2}Te^{-(\lambda_1+p\lambda_1+\alpha_2)T}\Bigg]^{-1}-\frac{1}{p\lambda_1+\alpha_2}=C_1,$$

где $0 \le T \le \tau_{\min}$, $\tau_{\min} = \min \tau_k$, $k = \overline{1,n}$.

Решение уравнения (1) возможно только численно. Пусть $f(T)$ − левая часть уравнения (1): $M\boldsymbol{\tau} = f(T)$. Покажем, что $f(T)$ является возрастающей функцией переменной T, $T > 0$. Имеем

1) $f(T=0) = \dfrac{p\lambda_1 + \alpha_2}{\lambda_1^{\,2}} > 0$;

2) $\lim\limits_{T\to\infty} f(T) = \infty$;

3) $f'(T) = \dfrac{\lambda_1 + p\alpha_2(1-\delta)}{\lambda_1^{\,2}\varphi_0^{\,2}(T)}\, p(T) + \dfrac{\left(p\alpha_2(1-\delta)\right)^2}{\lambda_1^{\,2}\varphi_0^{\,2}(T)e^{2\lambda_1 T}} + A(T)$,

$$A(T) = \frac{\left(p\alpha_2(1-\delta)\left(\lambda_1 + p\lambda_1 + \alpha_2\right)\right)^2}{\lambda_1^{\,2}\left(p\lambda_1 + \alpha_2\right)e^{(p\lambda_1+\alpha_2)T}B^2(T)} \times$$

$$\times\left\{\left(\lambda_1 + p\lambda_1 + \alpha_2\right)^2 \varphi_0(T)\, p(T) e^{-(p\lambda_1+\alpha_2)T} - B(T)\left(\varphi_0(T)\left(p\lambda_1 + \alpha_2\right) + p(T)\right)\right\},$$

$$B(T) = \left(p\lambda_1 + \alpha_2\right)\left(\lambda_1 + p\lambda_1 + \alpha_2\right) + \lambda_1\left(\lambda_1 + p\lambda_1 + \alpha_2\right)\varphi_0(T)e^{-(p\lambda_1+\alpha_2)T} +$$

$$+ p\alpha_2(1-\delta)\left(p\lambda_1 + \alpha_2\right)\left(1 - e^{-(\lambda_1 + p\lambda_1 + \alpha_2)T}\right),$$

$$p(T) = \left[\lambda_1 + p\alpha_2(1-\delta)\left(\lambda_1 T - 1\right)\right]e^{-\lambda_1 T},\ \varphi_0(T) = \left(1 + p\alpha_2(1-\delta)T\right)e^{-\lambda_1 T}.$$

Исследуя функцию $f'(T)$, обнаруживаем, что $f'(T) > 0$, $\forall T > 0$.

Таким образом, (1) имеет либо единственное решение, либо решения не имеет. Если $f(0) \ge C_1$, тогда в качестве значения оценки естественно выбрать $\hat{T} = 0$. Если $f(0) < C_1$, то:

а) либо корень уравнения (1) принадлежит полуинтервалу $\left(0, \tau_{\min}\right]$, тогда он и есть значение искомой оценки $\hat{T}$ (оценка $\hat{\mathbf{T}}$ является состоятельной, поскольку выполнены условия теоремы о состоятельности оценок [10]);

б) либо корень уравнения (1) больше, чем $\tau_{\min}$, тогда полагаем $\hat{T} = \tau_{\min}$.

3. Численные результаты оценивания

Построена имитационная модель обобщенного полусинхронного потока событий с продлевающимся мертвым временем фиксированной длительности T на основе классических подходов к имитации входящих потоков событий в СМО [12]. Результатом работы модели является выборка $t_1, t_2, ..., t_n$ моментов времени наступления событий в наблюдаемом потоке.

Алгоритм оценивания состоит из следующих этапов:

1) рассчитывается статистика C_1 как результат работы имитационной модели обобщенного полусинхронного потока событий, функционирующего в условиях неполной наблюдаемости;
2) методом Ньютона находится корень уравнения (1);
3) в результате повторения N раз шагов 1) и 2) находятся значения оценки $\hat{T}_1, \hat{T}_2, ..., \hat{T}_N$ в каждой i-й реализации, $i = \overline{1,N}$;
4) рассчитываются статистические характеристики − выборочное среднее и выборочная вариация оценки $\hat{\mathbf{T}}$, а также процент наблюдаемых событий по отношению ко всем событиям потока по формулам

$$\hat{M}(\hat{\mathbf{T}}) = \frac{1}{N}\sum_{i=1}^{N}\hat{T}_i,\ \hat{V}\left(\hat{\mathbf{T}}\right) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\hat{T}_i - T\right)^2,\ P = \frac{100}{N}\sum_{k=1}^{N}\frac{j^{(k)}}{j^{(k)} + i^{(k)}},$$

где $i^{(k)}$ – количество ненаблюдаемых событий в реализации, $j^{(k)}$ – количество наблюдаемых событий в реализации, T – значение длительности мертвого времени, известное из имитационной модели исследуемого потока.

Первый статистический эксперимент – установление стационарного режима функционирования потока.

Зафиксируем параметры потока с учетом выполнения условия особого соотношения параметров $\lambda_1 - (\lambda_2 + \alpha_2) = 0$ и условия рекуррентности $\lambda_1(1-p) = \alpha_2 - p\alpha_2(1-\delta)$ следующим образом: $N = 100$, $\lambda_1 = 0.8$, $\lambda_2 = 0.3$, $\alpha_2 = 0.5$, $\delta = 0.4$, $p = 0.6$, $T = 2.5$.

В табл. 1 представлены численные результаты первого эксперимента. В первой строке таблицы указана длительность времени моделирования $T_m = 200, 300, ..., 1900, 2000$ ед. времени; во второй и третьей строках – выборочное среднее $\hat{M}\left(\hat{\mathbf{T}}\right)$ и выборочная вариация $\hat{V}\left(\hat{\mathbf{T}}\right)$ оценки длительности мертвого времени T; в четвертой строке – процент наблюдаемых событий ко всем событиям исходного потока P.

Таблица 1

Численные результаты первого эксперимента

T_m	200	300	400	500	…	1600	1700	1800	1900	2000
$\hat{M}\left(\hat{\mathbf{T}}\right)$	2.528	2.552	2.523	2.517	…	2.506	2.497	2.492	2.489	2.501
$\hat{V}\left(\hat{\mathbf{T}}\right)$	0.038	0.029	0.016	0.014	…	0.005	0.004	0.003	0.004	0.004
P	19.36	19.15	19.26	19.3	…	19.49	19.58	19.72	19.70	19.53

По данным табл. 1 построены графики зависимостей $\hat{M}\left(\hat{\mathbf{T}}\right)$, $\hat{V}\left(\hat{\mathbf{T}}\right)$, P от времени моделирования T_m, приведенные на рис. 2–4.

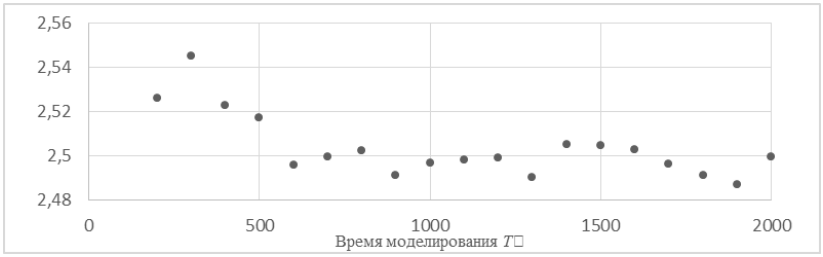


Рис. 2. Выборочное среднее $\hat{M}\left(\hat{\mathbf{T}}\right)$

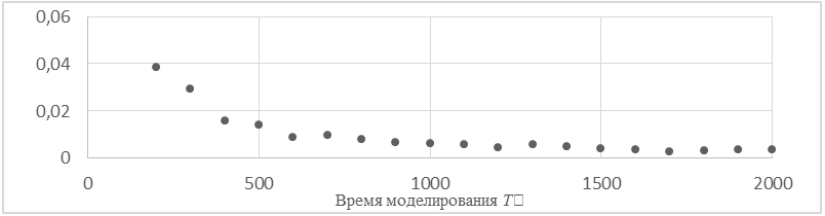


Рис. 3. Выборочная вариация $\hat{V}\left(\hat{\mathbf{T}}\right)$

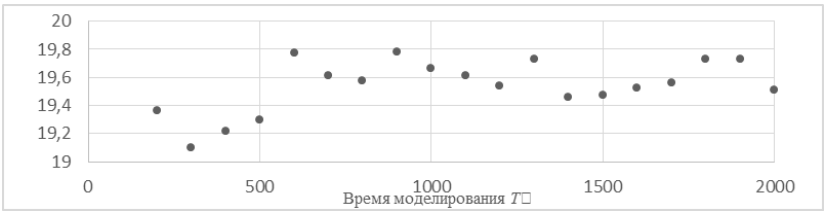


Рис. 4. Процент наблюдаемых событий ко всем событиям исходного потока *P*

Вывод: из анализа численных результатов первого статистического эксперимента следует, что стационарный режим функционирования исследуемого потока достигается при значении времени моделирования $T_m \geq 1000$ ед. времени, т.к. выборочное среднее $\hat{M}\left(\hat{\mathbf{T}}\right)$ стремится к постоянному значению.

Второй статистический эксперимент – исследование получаемых оценок при изменяющейся длительности мертвого времени *T*.

Зафиксируем два набора параметров потока с учетом условия рекуррентности и особого случая задания параметров:

1) $N = 100$, $\lambda_1 = 0.8$, $\lambda_2 = 0.3$, $\alpha_2 = 0.5$, $\delta = 0.4$, $p = 0.6$;

2) $N = 100$, $\lambda_1 = 1.5$, $\lambda_2 = 1$, $\alpha_2 = 0.5$, $\delta = 0.5$, $p = 0.8$.

Время моделирования как для первого набора параметров потока, так и для второго набора принимается $T_m = 1000$ ед. времени, что соответствует установлению стационарного режима.

В табл. 2 приведены численные результаты второго эксперимента для первого набора параметров потока. В первой строке таблицы указана длительность мертвого времени $T = 0.5, 1.0, ..., 2.5, 3.0$ ед. времени; во второй и третьей строках − выборочное среднее $\hat{M}\left(\hat{\mathbf{T}}\right)$ и выборочная вариация $\hat{V}\left(\hat{\mathbf{T}}\right)$ оценки длительности мертвого времени *T*; в четвертом столбце – процент наблюдаемых событий по отношению ко всем событиям исходного потока *P*.

Таблица 2

Численные результаты второго эксперимента для первого варианта набора параметров

T	0.5	1.0	1.5	2.0	2.5	3.0
$\hat{M}\left(\hat{\mathbf{T}}\right)$	0.493	0.998	1.496	2.000	2.500	3.007
$\hat{V}\left(\hat{\mathbf{T}}\right)$	0.001	0.001	0.001	0.002	0.003	0.005
P	76.42	53.17	38.44	24.52	19.57	18.27

Для наглядного представления поведения исследуемых характеристик построены графики зависимостей $\hat{M}\left(\hat{\mathbf{T}}\right)$, $\hat{V}\left(\hat{\mathbf{T}}\right)$, *P* от длительности мертвого времени *T*, приведенные на рис. 5−7.

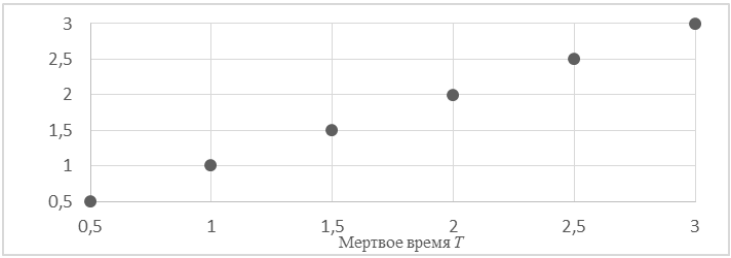


Рис. 5. Выборочное среднее $\hat{M}\left(\hat{\mathbf{T}}\right)$

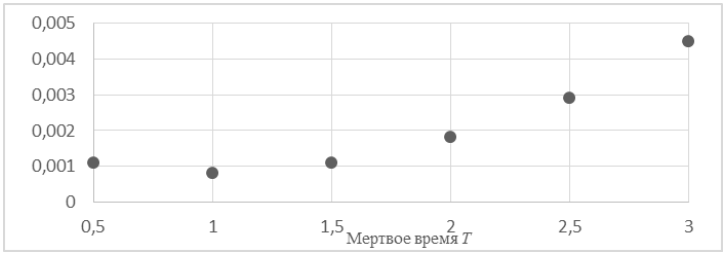


Рис. 6. Выборочная вариация $\hat{V}(\hat{\mathbf{T}})$

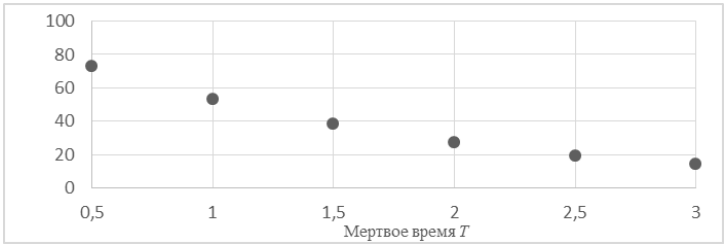


Рис. 7. Процент наблюдаемых событий ко всем событиям исходного потока P

В табл. 3 представлены численные результаты второго статистического эксперимента для второго набора параметров.

Таблица 3

Численные результаты второго эксперимента для второго варианта набора параметров

T	0.5	1.0	1.5	2.0	2.5	3.0
$\hat{M}(\hat{\mathbf{T}})$	0.491	0.989	1.497	1.999	2.493	3.004
$\hat{V}(\hat{\mathbf{T}})$	0.001	0.001	0.001	0.002	0.004	0.011
P	52.43	22.75	17.51	7.98	2.52	1.14

По данным табл. 3 построены графики зависимостей $\hat{M}(\hat{\mathbf{T}})$, $\hat{V}(\hat{\mathbf{T}})$, P от длительности мертвого времени T, представленные на рис. 8–10.

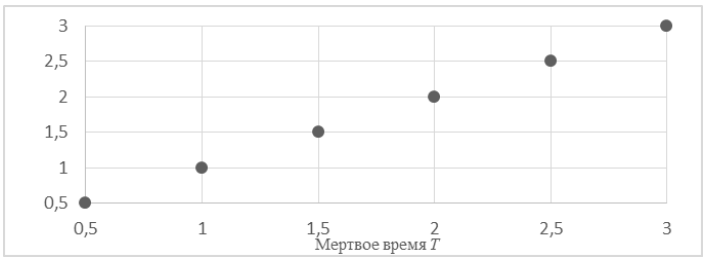


Рис. 8. Выборочное среднее $\hat{M}(\hat{\mathbf{T}})$

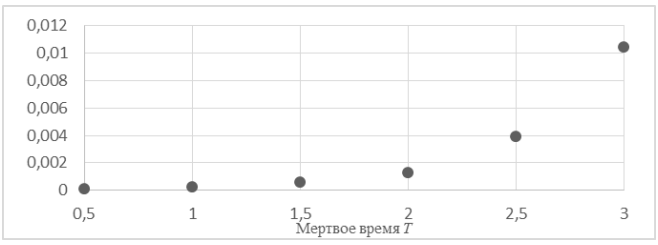


Рис. 9. Выборочная вариация $\hat{V}(\hat{\mathbf{T}})$

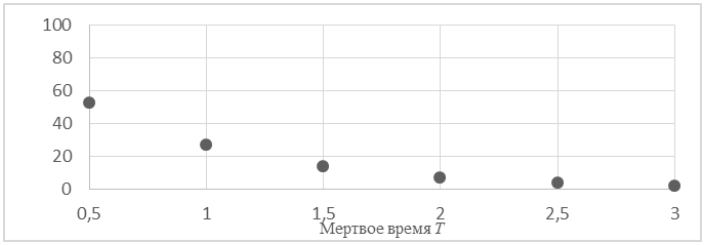


Рис. 10. Процент наблюдаемых событий ко всем событиям исходного потока *P*

Вывод: из анализа графиков, представленных на рис. 5, 8, видно, что выборочные средние значения оценок $\hat{M}\left(\hat{\mathbf{T}}\right)$ достаточно близки к истинным значениям *T*. На рис. 6, 7 и 9, 10 соответственно видна взаимная попарная обратная корреляция между выборочной вариацией $\hat{V}\left(\hat{\mathbf{T}}\right)$ и процентом наблюдаемых событий *P*. Увеличение значения длительности мертвого времени *T* приводит к уменьшению процента наблюдаемых событий *P* и, соответственно, к увеличению выборочной вариации оценки $\hat{V}\left(\hat{\mathbf{T}}\right)$

Заключение

В настоящей работе рассмотрен особый случай соотношения параметров $\lambda_1 - \left(\lambda_2 + \alpha_2\right) = 0$ в задаче оценивания длительности мертвого времени *T* в рекуррентном обобщенном полусинхронном потоке событий, функционирующем в условиях продлевающегося мертвого времени.

Численно получено единственное решение $\hat{T}$ уравнения моментов. Задача оценивания решена с использованием имитационной модели, реализованной на языке программирования C# в среде Visual Studio 2019.

Проведено достаточное количество статистических экспериментов, устанавливающих адекватность построенной имитационной модели, а также приемлемое качество получаемых оценок.

ЛИТЕРАТУРА

1. *Вишневский В.М., Дудин А.Н., Клименок В.И.* Стохастические системы с коррелированными потоками. Теория и применение в телекоммуникационных сетях. – М.: Техносфера, 2018. – 564 с.

2. *Башарин Г.П., Кокотушкин В.А.*, Наумов В.А. О методе эквивалентных замен расчёта фрагментов сетей связи. Ч. 1 // Изв. АН СССР. Техн. кибернетика. – 1979. – № 6. – С. 92–99.

3. *Башарин Г.П., Кокотушкин В.А., Наумов В.А.* О методе эквивалентных замен расчёта фрагментов сетей связи. Ч. 2 // Изв. АН СССР. Техн. кибернетика. – 1980. – № 1. – С. 55–61.

4. *Нежельская Л.А.* Оценка состояний дважды стохастических потоков событий. – Томск: Издательство Томского государственного университета, 2020. – 210 с.

5. *Апанасович В.В., Коляда А.А., Чернявский А.Ф.* Статистический анализ случайных потоков в физическом эксперименте. – Минск: Университетское, 1988. – 256 с.

6. *Nezhelskaya L., Tumashkina D.* Optimal state estimation of semi-synchronous event flow of the second order under its complete observability // Communications in Computer and Information Science. – 2018. – Vol. 912. – P. 93 – 105.

7. *Калягин А.А., Нежельская Л.А.* Сравнение МП- и ММ-оценок длительности мертвого времени в обобщенном полусинхронном потоке событий // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. – 2015. – № 3 (32). – С. 23−32.

8. *Нежельская Л.А.* Совместная плотность вероятностей длительности интервалов модулированного MAP-потока событий и условия рекуррентности потока // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. – 2015. – № 1 (30). – С. 57−67.

9. *Василевская Т.П., Горцев А.М., Нежельская Л.А.* Оценивание длительности мертвого времени и параметров синхронного альтернирующего потока с проявлением либо непроявлением событий // Вестник Томского государственного университета. – 2004. – № S9-2. – С. 129−138.

10. *Шуленин В.П.* Математическая статистика. Ч.1. Параметрическая статистика: учебник. – Томск: Изд-во НТЛ, 2012. – 540 с.

11. *Нежельская Л.А., Степаненко И.Д.* Оценка длительности мертвого времени в рекуррентном обобщенном полусинхронном потоке событий с продлевающимся мертвым временем в особом случае // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2024. – № 66. – С. 55–68.
12. *Лифшиц А.Л., Мальц Э.А.* Статистическое моделирование систем массового обслуживания. – М.: Сов. радио, 1978. – 248 с.

МЕТОД ЛОКАЛЬНОЙ ОЦЕНКИ ДЛЯ УГЛОВОГО РАСПРЕДЕЛЕНИЯ ЯРКОСТИ ТОЧЕЧНОГО ИСТОЧНИКА

Халиуллина А.Р.
Томский государственный университет
alinalailai12347@mail.ru

Введение

Существует множество задач теории переноса излучения в атмосфере. Самыми распространенными и актуальными являются задачи моделирования климата. При их решении важно учитывать множественные изменения светового потока на пути к нашей планете, которые происходят ввиду рассеяния и поглощения фотонов, составляющих этот поток. Важно учитывать неоднородность атмосферы, локальные и временные характеристики поля излучения в ней. Для получения данных характеристик используется один из методов Монте-Карло – метод локальной оценки углового распределения яркости.

1. Реализация схемы "прямого моделирования" методом Монте-Карло

Взаимодействие рассматриваемой среды с излучением точечного источника моделируется как процесс случайных столкновений фотонов с аэрозольными частицами, которые приводят либо к рассеянию, либо к поглощению [1].

Модель среды предполагает задание следующих параметров: σ_s – коэффициент рассеяния, σ_c – коэффициент поглощения, σ – коэффициент ослабления ($\sigma = \sigma_s + \sigma_c$).

Перемещения фотонов в среде задаются точкой, характеризующейся тремя декартовыми координатами для определения положения. Схема "прямого моделирования" методом Монте-Карло представлена на рис. 1 [2].

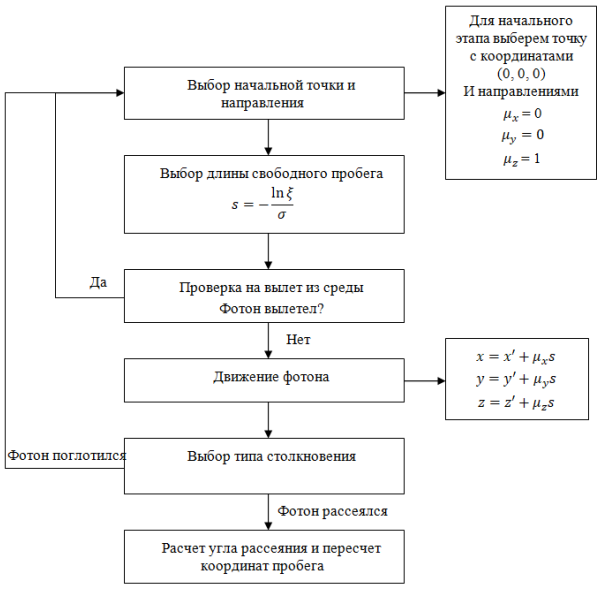


Рис. 1. Схема прямого моделирования

Пусть среда ограничена плоскостью $z = 0$, которая соответствует подстилающей поверхности Земли, и плоскостью $z = h$, при этом h является толщиной среды и выше данной границы процесс рассеяния отсутствует. Точечный приемник определяется координатами (0,0,h). Точечный источник излучения фотонов располагается на подстилающей поверхности в точке (0,0,0). Определим начальное направление фотонов с помощью направляющих косинусов, установив их следующим образом: $\mu_x = 0$, $\mu_y = 0$, $\mu_z = 1$.

Моделируем длину свободного пробега с помощью

$$s = -\frac{\ln \xi}{\sigma}, \qquad (1)$$

где ξ – случайное число от 0 до 1.

По формуле (1) генерируются случайные длины свободного пробега s, распределенные по экспоненциальному закону, соответствующему физической природе взаимодействия фотонов с веществом в методе Монте-Карло.

Для определения дальнейшего направления распространения аналогично будем использовать три направляющих косинуса путем обновления координат по следующим формулам:

$$\begin{cases} x = x' + \mu_x s, \\ y = y' + \mu_y s, \\ z = z' + \mu_z s. \end{cases}$$

В точке с обновленными координатами происходит столкновение, это столкновение приводит либо к поглощению, либо к рассеянию. Данная альтернатива моделируется с помощью вероятности рассеяния. Если в точке столкновения происходит поглощение, то происходит возврат к выбору начальной точки и направления (рис. 1), если же происходит рассеяние, то рассчитывается новое направление фотона.

Чтобы уменьшить дисперсию и улучшить качество модели, применяем модификацию "без поглощения", в данном случае вес фотона, который первоначально равен 1, при каждом столкновении умножается на альбедо однократного рассеяния [3].

Среднее значение косинуса угла рассеяния фотонов известно как анизотропия рассеяния (g), которая имеет значение от –1 до 1. Чтобы определить новое направление фотонов (и, следовательно, косинусы направления фотонов), нужно знать фазовую функцию рассеяния (индикатрису). Индикатриса рассеяния или фазовая функция в теории рассеяния света – это угловое распределение интенсивности рассеянной компоненты оптического излучения. Зададим индикатрису аналитически, с помощью формулы Хеньи – Гринштейна:

$$\cos\theta = \begin{cases} \dfrac{1}{2g}\left[1 + g^2 - \left(\dfrac{1 - g^2}{1 - g + 2g\xi}\right)\right], \ g \neq 0, \\ 1 - 2\xi, \ g = 0. \end{cases}$$

На основе исходных косинусов направления можно найти новый набор направляющих косинусов. Новое направление распространения может быть представлено в декартовой системе координат с помощью формул

$$\mu'_x = \frac{\sin\theta\left(\mu_x\mu_z\cos\varphi - \mu_y\sin\varphi\right)}{\sqrt{1-\mu_z^2}} + \mu_x\cos\theta, \ \mu'_y = \frac{\sin\theta\left(\mu_y\mu_z\cos\varphi - \mu_x\sin\varphi\right)}{\sqrt{1-\mu_z^2}} + \mu_y\cos\theta,$$

$$\mu'_z = -\sqrt{1-\mu_z^2}\sin\theta\cos\varphi + \mu_z\cos\theta.$$

Предполагается, что полярный угол φ равномерно распределен между 0 и 2π, т.е. $\varphi = 2\pi\xi$.

Рассмотрим особые случаи. Если фотон движется вверх по оси OZ, то новое направление пересчитывается с помощью формул $\mu'_x = \sin\theta\cos\varphi$, $\mu'_y = \sin\theta\sin\varphi$, $\mu'_z = \cos\theta$. Если же фотоны двигаются по оси OZ по направлению вниз, то новые направления определяются по формулам $\mu'_x = \sin\theta\cos\varphi$, $\mu'_y = -\sin\theta\sin\varphi$, $\mu'_z = -\cos\theta$.

Схема прямого моделирования была реализована в виде программного кода на языке Python для количества фотонов, равного 10000, и геометрической толщины среды, равной 100. Рис. 2–5 демонстрируют распространение фотонов в среде.

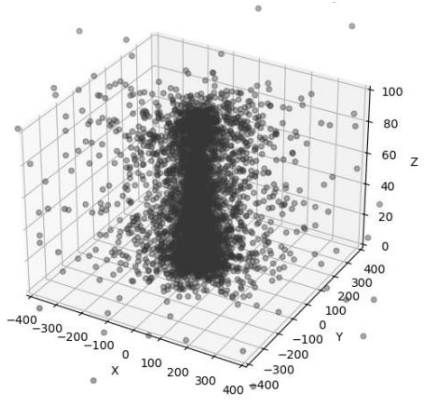


Рис. 2. Распределение фотонов в пространстве при анизотропии рассеяния равной 0.85

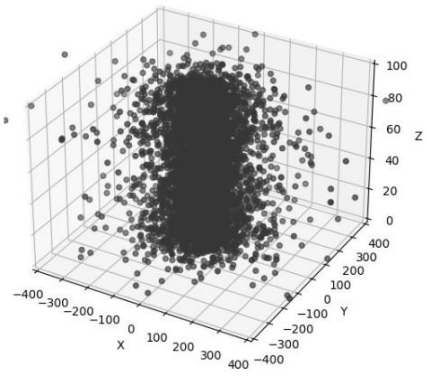


Рис. 3. Распределение фотонов в пространстве при анизотропии рассеяния равной 0.4

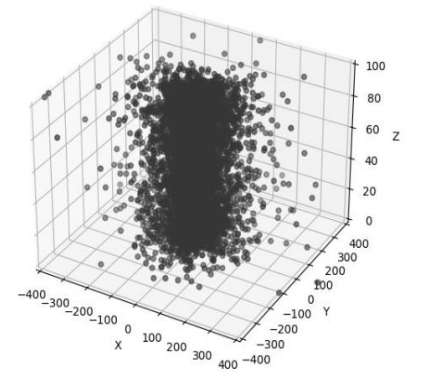


Рис. 4. Распределение фотонов в пространстве при анизотропии рассеяния равной 0.1

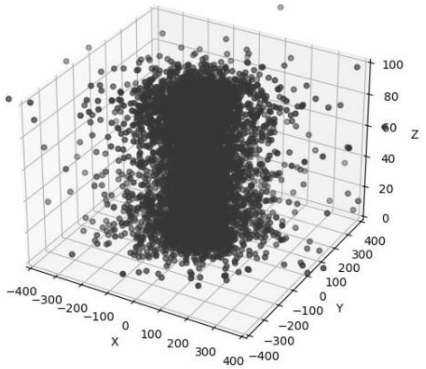


Рис. 5. Распределение фотонов в пространстве при анизотропии рассеяния, приближенной к значению 0

Можно заметить, что при увеличении оптической анизотропии и приближению ее значения к 1 рассеяние происходит преимущественно в прямом направлении, основной вес фотонов концентрируется вдоль оси OZ. Если же значение оптической анизотропии приближается к значению 0, то рассеяние происходит практически изотропно, т.е. равновероятно по всем направлениям.

2. Реализация метода локальной оценки углового распределения яркости

При реализации схемы прямого моделирования случайные направления рассеяния фотонов выбираются напрямую из заданной фазовой функции (индикатрисы) рассеяния. Это приводит к значительным статистическим флуктуациям в результатах, особенно для углов рассеяния, где фазовая функция мала. Локальная оценка позволит проанализировать, как яркость распределяется в различных углах относительно точечного приемника и различных областей пространства.

Получение углового распределение яркости предполагает вычисление следующего функционала:

$$\int_{\Omega_i} \Phi\left(\vec{r^*}, \vec{\omega^*}\right) d\omega = \int_X l_i\left(\vec{x'}, \vec{x^*}\right) f\left(\vec{x'}\right) d\vec{x'} = M \sum_{n=0}^{N} Q_n l_i\left(\vec{x_n}, \vec{x^*}\right), \tag{2}$$

$$l_i\left(\vec{x}, \vec{x^*}\right) = \frac{\exp\left[-\tau\left(\vec{r}, \vec{r^*}\right)\right] p\left(\mu^*\right)}{2\pi\left|\vec{r^*} - \vec{r}\right|^2} \Delta_i\left(\vec{s^*}\right), \tag{3}$$

$$\vec{s^*} = \frac{\vec{r^*} - \vec{r}}{\left|\vec{r^*} - \vec{r}\right|}, \ \mu^* = \left(\vec{\omega^*}, \vec{s^*}\right),$$

$$p\left(\mu^*\right) = \frac{1 - g^2}{\left(1 + g^2 - 2g\mu^*\right)^{\frac{3}{2}}}. \tag{4}$$

Здесь $\Delta_i\left(\vec{s}\right)$ – индикатор исследуемой области Ω_i, $f\left(\vec{x}\right)$ – плотность столкновений, Φ – поток частиц в заданной точке $\vec{x^*}$, Q_n – вспомогательный вес.

Формула (2) определяет плотность вероятности того, что фотон, находящийся в определенной точке и имеющий определенное направление рассеется по направлению к точечному приемнику и попадет в точку с координатами расположения точечного приемника [1].

Формула (3) имеет смысл плотности вероятности того, что фотон, который находится в точке *x*, которая характеризуется положением и направлением, после столкновения рассеется по направлению к точке *x**. Точка *x**— точечный приемник.

По формуле (4) вычисляется индикатриса Хеньи – Гринштейна, т.е. вероятность того, что фотон рассеется под определенным углом.

Для определения областей пространства используем телесные углы. Телесный угол – это пространственный угол, который охватывает часть сферы. Разделим пространство на 89 телесных углов, представленных в виде круговых конусов с высотой, равной h и постоянно увеличивающимся радиусом на 1 (рис. 6).

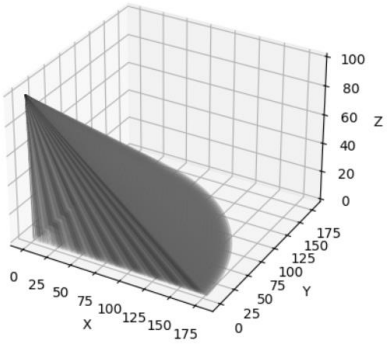


Рис. 6. Деление пространства на телесные углы в I-м октанте

Представленный алгоритм применяется для построения углового распределения яркости точечного источника, которое находится через отношение плотности вероятности (2) в области Ω_i к величине телесного угла.

Мы исследовали угловое распределение яркости для коэффициентов ослабления, численно равных 0.1 (рис. 7) и 0.9 (рис. 8).

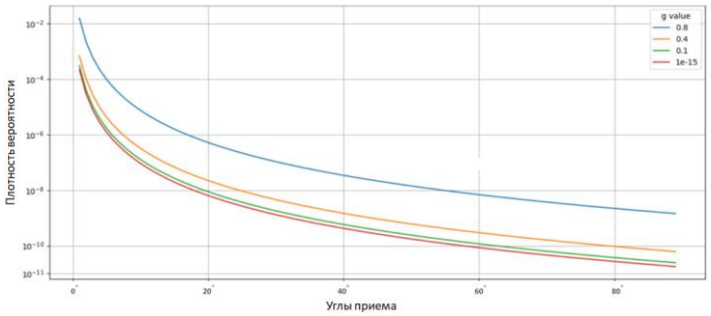


Рис. 7. Угловое распределение яркости точечного источника при $\sigma = 0.1$

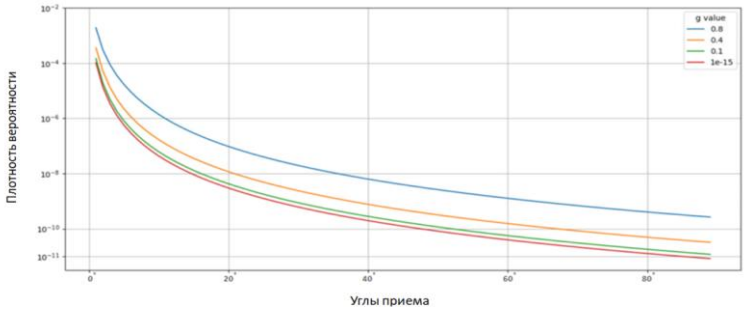


Рис. 8. Угловое распределение яркости точечного источника при $\sigma = 0.9$

Анализ результатов позволил выявить следующие закономерности.

1. Значения плотности вероятности уменьшаются при отдалении от начальных углов приема. При малых углах отклонения от начального направления распространения

фотоны испытывают меньше рассеяний в среде и, следовательно, теряют меньше энергии. Это приводит к более высоким значениям плотности вероятности вблизи начальных углов приема. По мере увеличения угла отклонения от начального направления, фотоны проходят большее расстояние в среде, что увеличивает вероятность их рассеяния. С каждым рассеянием часть света теряется, что приводит к уменьшению интенсивности света, достигающего точечного приемника под отдаленными от центра областями.

2. С увеличением коэффициента ослабления плотности вероятности уменьшаются. В процессе увеличения σ и, соответственно, взаимодействия фотонов со средой увеличивается и вклад многократного рассеяния. Многократное рассеяние приводит к тому, что свет теряет больше энергии, т.к. с каждым столкновением часть света рассеивается. Это ведет к увеличению расстояния, которое проходят фотоны в среде и увеличению оптической толщины. Увеличение оптической толщины приводит к уменьшению интенсивности фотонов.

Заключение

Таким образом, результаты исследования позволяют сделать следующие выводы:

1. Если оптическая анизотропия увеличивается и приближается к 1, то рассеяние происходит преимущественно в прямом направлении, основной вес фотонов концентрируется вдоль оси OZ.

2. В процессе увеличения σ и, соответственно, взаимодействия фотонов со средой, увеличивается и вклад многократного рассеяния. Это ведет к увеличению расстояния, которое проходят фотоны в среде и увеличению оптической толщины. Увеличение оптической толщины приводит к уменьшению интенсивности фотонов.

ЛИТЕРАТУРА

1. *Марчук Г.И.* Метод Монте-Карло в атмосферной оптике. – Новосибирск: Наука, 1976. – 284 с.

2. Метод Монте-Карло [электронный ресурс] // Википедия: свободная энциклопедия. – Электрон.дан. – [Б.м.], 2013. – URL: http://ru.wikipedia.org/wiki/Метод_Монте-Карло (дата обращения: 25.04.2024).

3. *Зуев В.Е., Креков Г.М.* Оптические модели атмосферы. – Ленинград: Гидрометеоиздат , 1986. – 256 с.

II. ПРОБЛЕМЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ И РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ

СОЗДАНИЕ ВЕБ-РЕСУРСА ДЛЯ ВЗАИМОДЕЙСТВИЯ ВРАЧА И ПАЦИЕНТА

Абросимова М.А., Пахомова Е.Г.
Томский государственный университет
pocyriteko@gmail.com, PakhomovaEG@yandex.ru

Введение

Сегодня о стрессе говорят практически все, и многие люди склонны считать его основной причиной своих проблем, в том числе и проблем со здоровьем. Вследствие этого значительно возрос спрос на услуги профессионалов в области психологического здоровья. Если в 2009 г. за помощью к психологам обращалось всего 6% россиян, то к 2022 году эта цифра увеличилась до 12%. Однако, несмотря на этот рост, по оценкам специалистов, около 25% россиян действительно нуждаются в психологической помощи.

В нынешних условиях обращение к профессионалам в области психологического здоровья становится почти необходимостью. Однако отсутствие налаженной и эффективной связи между психотерапевтом и пациентом часто затрудняет процесс лечения. Это создает сложности как для специалистов, так и для тех, кто обращается к ним за помощью.

Разрабатываемый веб-ресурс призван решить эту проблему, облегчая взаимодействие между врачом и пациентом. С его помощью можно будет существенно сократить время на сбор анамнеза, т.к. врач сможет заранее изучить психологическое состояние пациента до момента приема. Кроме того, практика ведения журнала (или дневника) психического здоровья, которая становится все более популярной, будет проще и удобнее с использованием разрабатываемого веб-ресурса. Это позволит пациентам более тщательно отслеживать свои эмоциональные состояния и динамику лечения, а врачам – более эффективно анализировать и корректировать терапевтические подходы.

1. Архитектура

Для разработки веб-ресурса была выбрана архитектура MVC (Model-View-Controller), которая помогла разделить приложение на три составляющие: модель, представление и контроллер. Модель отвечает за обработку данных и бизнес-логику приложения, представление отображает данные пользователю, а контроллер управляет взаимодействием между моделью и представлением, обрабатывая входящие запросы и инициируя необходимые действия.

Архитектура включает в себя три основных компонента: клиентскую часть (Frontend), серверную часть (Backend) и уровень данных (Database).

Клиентская часть. Для клиентской части веб-сайта была выбрана архитектура Multi-page Application (MPA). Multi-page Application (MPA) – это веб-приложение, для которого сервер при каждом запросе новой страницы отправляет полный HTML-документ. Для создания клиентской части использовались языки HTML, CSS и JavaScript [1].

В процессе разработки сайта, благодаря возможностям шаблонизатора Jinja, была использована функциональность создания шаблонов. В результате их было создано три: base.html, base-posts.html и base-files.html. Также шаблонизатор Jinja в проекте используется для генерации HTML-страниц с динамическим содержимым. Это позволяет

интегрировать данные из серверной части на стороне клиента, обеспечивая более гибкий и интерактивный пользовательский интерфейс.

Серверная часть. Архитектура backend является монолитной. В монолитной архитектуре весь код серверной части находится в одном приложении. Все его компоненты тесно связаны и работают в одном процессе (например, обработка маршрутов, взаимодействие с базой данных, логика авторизации и аутентификации).

Взаимодействие клиента и сервера. Front-end взаимодействует с сервером через HTTP-запросы, которые являются средством обмена информацией между клиентом и сервером. Эти запросы отправляются на определенный URL и включают метод запроса, такой как, например, GET и POST.

GET-запросы нужны для получения данных с сервера. Например, при загрузке веб-страницы браузер отправляет запрос, чтобы получить HTML-документ и другие связанные с ним ресурсы (CSS, JavaScript, изображения).

POST-запросы нужны для отправки данных на сервер. В основном они используются для того, чтобы передать информацию для обработки при отправке форм.

Уровень данных. Для инициализации базы данных используется Flask-SQLAlchemy [2]. В проекте реализованы три ключевые модели базы данных: Users, Records и Files (рис. 1).

Модель Users управляет информацией о пользователях, обеспечивая безопасное хранение данных и поддержку функциональности аутентификации и авторизации. Также внутри класса определены функции для шифрования, дешифрования, получения идентификатора пользователя и проверок активности, аутентификации.

Модель Records отвечает за хранение записей, которые могут содержать различную информацию, относящуюся к пользователям или их действиям в системе.

Модель Files предназначена для управления файлами, загружаемыми и хранящимися в системе, обеспечивая их безопасность и доступность.

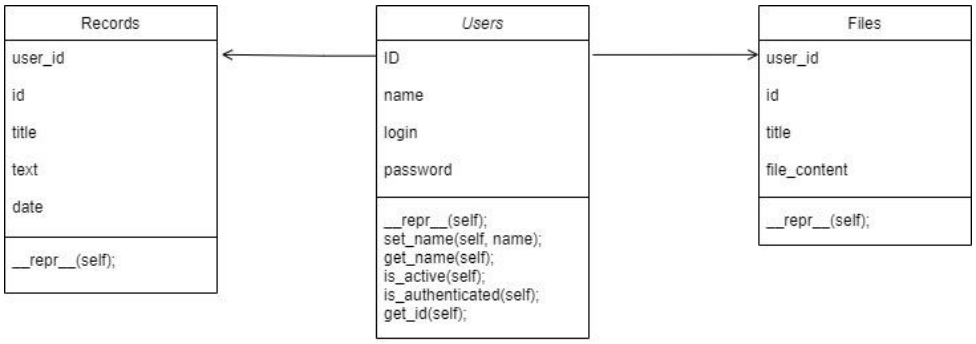


Рис. 1. Модель базы данных

Основное приложение (app.py). Файл app.py отвечает за настройку маршрутов и их обработку. Маршруты (route) – главная часть любого приложения Flask [3]. Они задают логику, которая должна быть выполнена, когда пользователь обращается к определенному URL. Создание маршрутов в Flask происходит с помощью специального декоратора @app.route. Он сопоставляет URL с функцией Python [4], позволяя приложению отвечать на различные HTTP-запросы.

Основное приложение, которое содержится в файле app.py, отвечает за инициализацию приложения Flask и его конфигурацию. В этом файле настраивается Flask-Login для управления процессами аутентификации и авторизации, а также создаются маршруты для регистрации, входа и выхода пользователей. Веб-ресурс предоставляет маршруты для взаимодействия с пользователем. Рассмотрим их подробнее.

Начнем с главной страницы и страницы с информацией. Маршрут для главной страницы (/) отображает шаблон home.html, предоставляя пользователям основной ин-

терфейс для начала работы с приложением. Страница с информацией (/information) аналогично отображает шаблон information.html, который содержит важную информацию для пользователей.

Регистрация пациента происходит через маршрут /patient-registration, который обрабатывает как GET-, так и POST-запросы. При GET-запросе отображается форма регистрации, а при POST-запросе обрабатываются данные, введенные пользователем в форму. В случае успешной регистрации создается новый пользователь, шифруется его имя и сохраняется в базе данных [5]. Если регистрация прошла успешно, пользователь перенаправляется на страницу идентификации (/patient-id).

Маршрут /patient-id предназначен для отображения информации о пациенте. Используя идентификатор пользователя из сессии, он извлекает данные из базы и дешифрует имя пациента для отображения на странице patient-id.html.

Для входа пациентов используется маршрут /patient-enter. При GET-запросе он отображает форму входа, а при POST-запросе проверяет введенные логин и пароль. Если данные корректны, пользователь аутентифицируется и перенаправляется на страницу с записями пациента (/patient-posts).

Маршрут /logout отвечает за выход пользователя из системы. Он удаляет идентификатор пользователя из сессии и перенаправляет его на страницу входа пациента.

Доктора могут войти в систему через маршрут /doctor-enter, который обрабатывает GET- и POST-запросы. При POST-запросе вводимые доктором уникальные код и имя проверяются на соответствие в базе данных. Если данные корректны, доктор перенаправляется на страницу с записями пациентов (/doctor-posts).

Маршрут /patient-posts отображает записи текущего пользователя. Все записи извлекаются из базы данных, дешифруются и передаются в шаблон patient-posts.html для отображения.

Для докторов предусмотрен маршрут /doctor-posts, который также обрабатывает GET-запросы. При GET-запросе он отображает записи пациента, если доктор аутентифицирован; если нет – доктору предлагается войти в систему.

Создание новых записей пациента осуществляется через маршрут /patient-create-record, который обрабатывает GET- и POST-запросы. При GET-запросе отображается форма для создания записи, а при POST-запросе данные формы шифруются и сохраняются в базе данных.

Маршрут /posts/<string:id>/delete (в поле string:id вставляется уникальный идентификатор каждой записи) позволяет удалять записи пациента. В случае, когда у пользователя есть права на удаление записи, она удаляется из базы данных.

Маршрут /posts/<string:id>/update отвечает за обновление записей пациента. При GET-запросе этот маршрут отображает форму для редактирования выбранной заметки. Запись из базы данных извлекается по ее идентификатору, далее содержимое дешифруется и передается в шаблон patient-post-update.html. Это позволяет, собственно, редактировать имеющуюся запись, а не вводить ее заново. При POST-запросе обрабатываются данные, введенные пользователем в форму. Обновляются заголовок и текст записи, после чего текст снова шифруется и сохраняется обратно в базу данных. В случае успешного обновления пользователь перенаправляется на страницу со списком записей пациента.

Для взаимодействия с файлами используются маршруты /patient-files для загрузки и отображения файлов пациента и /patient-files/<int:id> (int:id – уникальный идентификатор каждого файла) для скачивания на компьютер конкретного файла. Первый маршрут обрабатывает как GET-, так и POST-запросы, позволяя загружать файлы и отображать список загруженных файлов. Второй маршрут позволяет пользователю скачать файл, если у него есть на это права.

2. Интерфейс

Рассмотрение интерфейса начнем с главной страницы сайта (рис. 2).

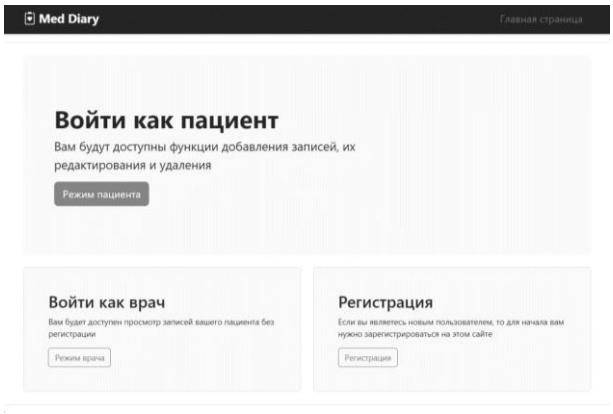


Рис. 2. Главная страница

В центре страницы находятся три информационных блока с кнопками, направляющими пользователей на страницы с регистрацией или входа в одном из двух режимов: врача или пациента.

В верхней части (header) располагается надпись «Главная страница», которая является ссылкой, позволяющей вернуться на данный раздел из любой точки сайта. Надпись «Med Diary» помимо того, что представляет собой название данного ресурса, также служит ссылкой на страницу с информацией о сервисе.

Для того, чтобы пользоваться сайтом в режиме пациента необходима предварительная регистрация [5], интерфейс которой представлен на рис. 3.

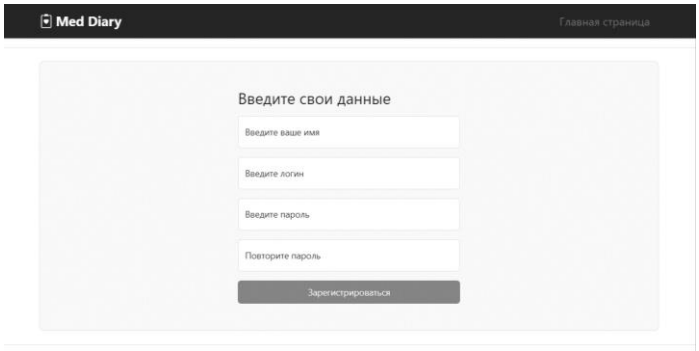


Рис. 3. Регистрация

При корректном вводе имени, логина и пароля, пользователю отображается сообщение об успешной регистрации вместе с уникальным кодом (ID). Этот код рекомендуется передать своему лечащему врачу. Далее пользователь перенаправляется на страницу ввода логина и пароля. При корректно введенных данных пациент попадает на страницу со своими записями (рис. 4). В шапке страницы также появляется кнопка для управления аккаунтом, имеющая 3 функции: выход из аккаунта, повторная генерация уникального кода (ID) и удаление аккаунта со всеми связанными файлами, записями и личной информацией.

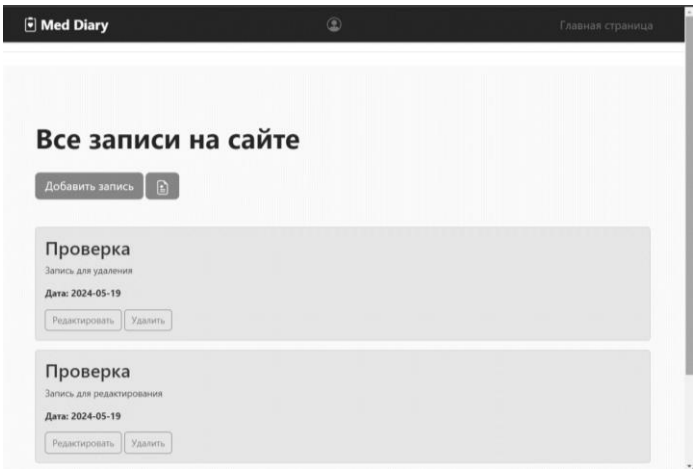


Рис. 4. Все записи пациента

Внизу каждой записи расположены кнопки для редактирования и удаления. Пациент может воспользоваться этими кнопками, если какая-либо из записей окажется неактуальной или будет содержать ошибки.

После взаимодействия с кнопкой «удалить», пользователю будет предложено подтвердить свое намерение удалить запись. В случае положительного ответа заметка будет безвозвратно удалена.

При нажатии кнопки «редактировать», пользователь переадресовывается на страницу редактирования. При взаимодействии с кнопкой «Добавить запись» пользователю будет предложено создать новую запись.

Кнопка просмотра загруженных файлов перенаправляет пациента на страницу, где хранятся его файлы. В верхней части этой страницы расположена кнопка для возврата ко всем записям пациента. В центральной части размещены кнопки «Выбрать файл» и «Загрузить файл», благодаря которым пользователь может добавлять необходимые ему документы (рис. 5).

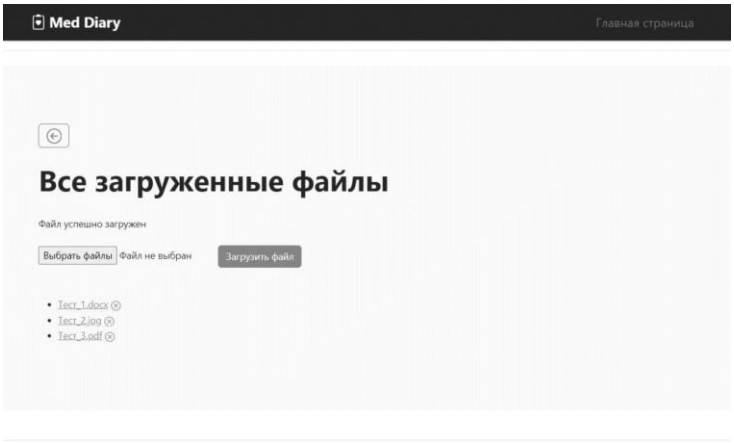


Рис. 5. Все файлы пациента

При работе с сайтом в режиме врача пользователю предлагается ввести имя пациента, указанное при регистрации, и уникальный код, присвоенный пациенту автоматически (рис. 6).

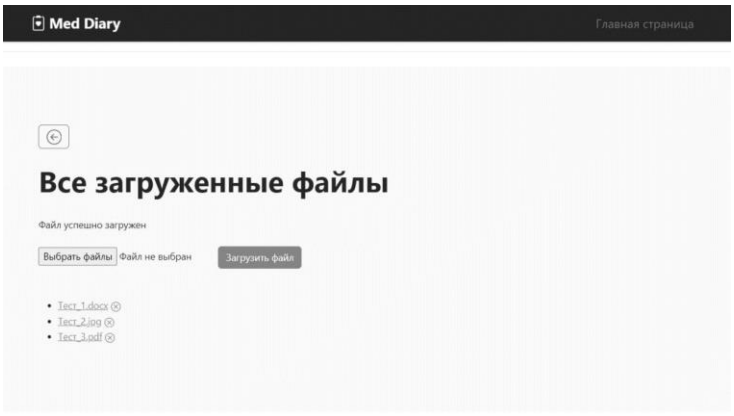


Рис. 6. Вход для врача

После успешного ввода данных врач перенаправляется на страницу, где отображаются все записи его пациента (рис. 7). В шапке сайта дополнительно появляется кнопка «Сменить пациента», которая возвращает пользователя на страницу ввода данных пациента. В центральной части страницы видны все заметки пациента, а также кнопка, которая при нажатии открывает все ранее добавленные файлы конкретного пациента (рис. 8).

Врачу недоступны функции редактирования, удаления и добавления записей и файлов.

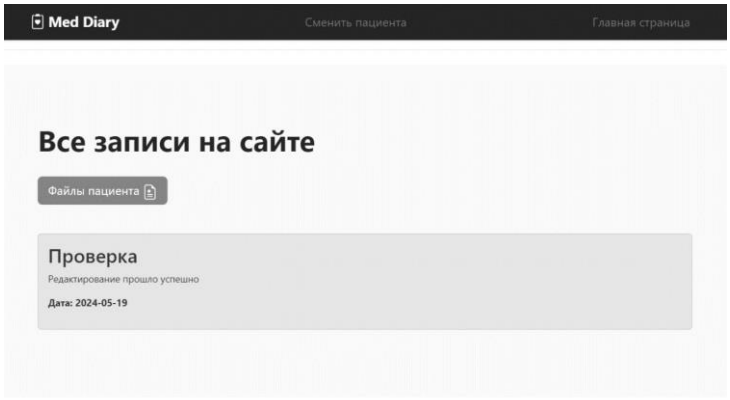


Рис. 7. Все записи пациента (режим врача)

Рис. 8. Все файлы пациента (режим врача)

Заключение

В заключении следует отметить, что цель создания функционального веб-сайта для взаимодействия врача и пациента была успешно достигнута. Созданный веб-ресурс позволяет пациентам вести записи о своем самочувствии, загружать соответствующие файлы и делиться ими с врачами, что существенно упрощает и улучшает процесс взаимодействия между врачом и пациентом.

Этот ресурс имеет потенциал для дальнейшего развития и масштабирования, что позволяет адаптировать его под различные нужды и запросы пользователей в сфере медицинского обслуживания. Планируется добавить возможность выбора цвета эмоций к каждой записи, чтобы легче было наблюдать за изменениями эмоционального фона пациента, реализовать систему заметок для врачей, загрузить ресурс на удаленный сервер.

Можно заключить, что данный проект может быть использован в реальных условиях для улучшения качества медицинского обслуживания и взаимодействия между пациентами и медицинскими работниками.

Полностью ознакомиться с кодом программы можно по ссылке https://github.com/aabrosimovam/Final-Qualification-Work

ЛИТЕРАТУРА

1. Руководство по JavaScript [Электронный ресурс] // MDN Web Docs URL: https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide (дата обращения 10.05.2024)
2. Документация SQLAlchemy [Электронный ресурс] // SQLAlchemy 2.0 Documentation URL: https://docs.sqlalchemy.org/en/20/ (дата обращения 20.05.2024)
3. Документация Flask [Электронный ресурс] // flask-russian-docs URL: https://flask-russian-docs.readthedocs.io/ru/0.10.1/ (дата обращения 18.05.2024)
4. Руководство по Python [Электронный ресурс] // python.org URL: https://www.python.org/doc/ (дата обращения 15.05.2024).
5. Документация cryptography [Электронный ресурс] // cryptography.io URL: https://cryptography.io/en/latest/ (дата обращения 15.05.2024)

СОЗДАНИЕ ПРИЛОЖЕНИЯ УПРАВЛЕНИЯ НЕМАТЕРИАЛЬНЫМИ АКТИВАМИ

Алексеенко И.С.
Томский государственный университет
ilya.alekseenko.s@gmail.com

Введение

С развитием информационных технологий становится все больше и больше IT-компаний, которые продают свои услуги по разработке, интеграции и обслуживанию программного обеспечения. При этом каждая из таких организаций обладает собственной структурой, имеет собственный подход к разработке программного обеспечения, собственный подход к организации коммуникаций внутри команд разработки и между ними. В процессе разработки каждая команда и каждый сотрудник ведут отчетность того, что было сделано за конкретный период. Хороший сценарий – документирование кода происходит параллельно его написанию. Но, как показывает практика, так дела обстоят далеко не всегда. Команды разработчиков сосредоточены на выполнении бизнес-задач, которые были перед ними поставлены владельцами продуктов; у каждой такой задачи есть конкретные сроки по реализации. Времени на документирование хода разработки часто не остается.

Аудиторские проверки – это один из способов оценить текущее состояние активов компании. Всего существует два вида активов: материальные, например, стол, машина, загородный дом, и нематериальные, например, патент, код, который пишет программист, логотип. Компании отлично справляются с учетом материальных активов, есть

множество систем и механизмов, позволяющих контролировать такие активы и вести их учет. Нематериальные активы – это объекты интеллектуальной собственности. Вариативность таких активов очень высокая и даже отдельные области права оказываются неполными в части управлениями такими активами.

Продуктом IT-компании по большей части является нематериальный актив, от него зависит капитализация компании, стоимость ее отдельных услуг, возможность сотрудничества с другими компаниями, например, использование в своем продукте отдельного модуля или целого продукта сторонней компании.

1. Понятие нематериального актива

Нематериальные активы – это объекты интеллектуальной собственности, которые не имеют материально-вещественной формы, которые могут быть отделены от другого имущества [1].

Управление нематериальными активами – это процесс идентификации, учета, оценки стоимости объектов интеллектуальной собственности, целью которого является извлечение прибыли, поддержание прозрачности в вопросе стоимости актива и его конкурентоспособности.

С развитием информационных технологий расширилась и область, в которой можно создавать интеллектуальную собственность. Нематериальными активами являются компьютерные программы, которые пишут разработчики, изобретения и промышленные модели, товарные знаки, ноу-хау, логотипы и т.д. Поскольку большой спрос на рынке представляют компьютерные программы, IT-компании непременно развиваются в сторону повышения собственной конкурентоспособность и создают всё больше продуктов, которые удовлетворяют потребности пользователей.

На сегодняшний день законодательная база не описывает полностью процесс управления нематериальными активами. Постоянно изменяющиеся средства и способы передачи информации и прав на то или иное произведение человеческой мысли только усугубляют ситуацию.

Наличие проблемы управления нематериальными активами среди IT-компаний не кажется серьезным препятствием перед выпуском своих продуктов на рынок, но является серьезным обстоятельством, которое замедляет процесс разработки и уменьшает итоговую стоимость трудозатрат компаний.

2. Проблема управления нематериальными активами

Рассмотрим проблемы, с которым сталкиваются IT-компании в процессе управления нематериальными активами (НМА).

Проблема классификации НМА в России. Широкое распространение информационные технологии получили относительно недавно. Однако первые экономические исследования, связанные с изучением объектов, которые можно отнести к нематериальным активам, появились в конце 19-го – начале 20-го веков. Поскольку нематериальный актив – понятие достаточно многогранное, ситуацию с его изучением осложняет область, к которой относится актив: нематериальный актив в бухгалтерской сфере и экономической уже имеет отличия и нюансы.

Проблема классификации нематериальных активов исследована учеными только на теоретическом уровне, однако на законодательном уровне еще не собрана достаточного уровня практика в области управления нематериальными активами и достаточно полная классификация нематериальных активов.

Нематериальные активы неоднородны по своему составу, по своему жизненному циклу, по характеру использования в процессе производства, по степени влияния на финансовое состояние и результаты хозяйственной деятельности предприятия [2,3]. Пример классификации нематериальных активов приведен в табл. 1.

Таблица 1

Пример классификации нематериальных активов

Классификационный признак	Группы (виды) активов
По отраслям народного хозяйства	1. Используемые в промышленности. 2. Применяемые в сельском хозяйстве. 3. Необходимые в лесном хозяйстве. 4. Используемые в строительстве. 5. Используемые в других отраслях.
По функциональному назначению	1. Производственные. 2. Непроизводственные.
По натурально-вещественному признаку	1. Объекты интеллектуальной собственности. 2. Влияющие на деловую репутацию.
По отношению к процессу эксплуатации	1. Функционирующие (работающие). 2. Нефункционирующие (неработающие).
По срокам использования	1. От 1 до 5 лет. 2. От 5 до 10 лет. 3. От 10 до 20 лет. 4. Свыше 20 лет.
По возможности использования механизма амортизации	1. Амортизируемые. 2. Не амортизируемые.
По нормативному регулированию	1. Регулируемые патентным правом. 2. Регулируемые авторским правом. 3. Регулируемые другими отраслями права.
Степень отчуждения	1. Отчуждаемые. 2. Неотчуждаемые.
По степени влияния на финансовые результаты предприятия	1. Объекты, оказывающие прямое влияние. 2. Объекты, оказывающие косвенное влияние.
По этапам жизненного цикла изделия	1. На этапе разработки товара. 2. На этапе выведения на рынок. 3. На этапе роста. 4. На этапе зрелости. 5. На этапе упадка.
По стадиям кругооборота движения средств	1. На стадии процесса снабжения. 2. На стадии производства продукции. 3. На стадии продажи продукции.
По отношению к хозяйствующему субъекту	1. Внешние. 2. Внутренние.

Со временем практика по управлению нематериальными активами будет накоплена, однако с развитием технологий появляются все новые и новые инструменты и технологии, которые требуют описания. И в условиях быстро меняющихся технологий необходимо успевать писать документацию и соответствующие нормативные правовые акты, регламентирующие их использование, управление и учет.

Проблема управления нематериальными активами в IT-компаниях. В условиях отсутствия документов, которые бы регламентировали механизм управления нематериальных активов, находящихся на балансе компании, приходится описывать процессы, труд людей и отделов имеющимися средствами.

Например, по окончании отчетного периода работы над проектом, приходится проходить внешние и внутренние аудиторские и налоговые проверки, которые требуют больших трудозатрат команды.

Такие проверки включают в себя подготовку соответствующих сопровождающих на протяжении разработки продукта документов, описание руководителями продуктовых команд труда, рассредоточенного по различным системам управления проектами, и ведения документации, а также сравнение результатов, заявленных перед выполнением работ, с фактическим состоянием продукта.

Продуктовые команды сосредоточены непосредственно на разработке программного продукта, поэтому не уделяют должного внимания к описанию процессов труда, полного стека технологий, используемых библиотек с указанием конкретных лицензий на их использование, актов передачи прав, отчуждения труда и т.д.

Если разработчики будут уделять достаточно времени для описания всего вышеперечисленного, то порог входа на должность разработчика возрастет, а время на разработку продукта вырастет в разы. Для таких целей нужно нанимать отдельную команду, которая бы непредвзято описывала ход и результаты разработки, что сильно повысит затраты на разработку.

Пренебрегая своевременным управлением и фиксацией нематериального актива, компании вынуждены закладывать серьезную часть бюджета на затраты, связанные с нарушением законодательства, и рискуют потерять свою реальную стоимость или возможность повысить собственную капитализацию, ведь недолжное описание работ ведет к недооценке своих услуг на рынке со всеми вытекающими последствиями в виде потери прибыли и оттока сотрудников.

На сегодняшний день компании открыты к решениям по управлению нематериальными активами, которые способны обеспечить прозрачность процессов в ходе разработки программного продукта, снизить затраты, связанные с недобросовестным оформлением результатов работ, повысить капитализацию компании, не привлекая серьезных технических и человеческих трудовых ресурсов.

3. Продукт Iptimizer

Рассмотрим разрабатываемое решение по управлению нематериальными активами в IT-компаниях. Стоит отметить, что продукт находится в активной стадии разработки, поэтому часть сведений составляет коммерческую тайну; сведения такого рода рассмотрены не будут.

3.1. Общее назначение Iptimizer

Iptimizer – это Web-приложение, которое отражает реальное состояние нематериальных активов компании. Общее назначение приложения Iptimizer – комплексное управление нематериальными активами в IT-компаниях. Основные цели и задачи приложения следующие.

Отражение реального состояния нематериальных активов компании. Iptimizer позволяет вести учет и мониторинг всех нематериальных активов, которые будут занесены в систему, например, код, который пишет разработчик, патенты, торговые марки, авторские права и лицензии.

Учет структуры IT-компании. Приложение учитывает организационную структуру компании и принятые процессы управления продуктами, позволяя отслеживать и управлять нематериальными активами на различных уровнях, таких как отделы, команды и отдельные сотрудники.

Отслеживание связей между продуктами. Iptimizer позволяет устанавливать и отслеживать связи между различными продуктами компании, а также связи с продуктами, которые используется самой компанией извне; это могут быть как внутренние сервисы компании, так и необходимые интеграции в продуктах. Это помогает фиксировать зависимости и потенциальные риски, связанные с использованием сторонних модулей и компонент.

Управление лицензиями и авторскими правами. Приложение позволяет отслеживать лицензии на используемые библиотеки и компоненты, а также управлять передачей авторских прав от разработчиков к компании. Это обеспечивает соблюдение лицензионных соглашений и защиту интеллектуальной собственности с точки зрения акционеров компании, а также аудиторских и налоговых проверок.

Поддержка отчетности. Iptimizer предоставляет возможность генерировать отчеты по заданным шаблонам на языке LaTeX. Это позволяет создавать профессионально оформленные документы, отражающие состояние нематериальных активов компании.

В целом, Iptimizer призван помочь IT-компаниям эффективно управлять своими нематериальными активами, минимизировать риски, связанные с интеллектуальной

собственностью, и обеспечить соблюдение правовых и лицензионных требований. Приложение предоставляет централизованную платформу для учета, мониторинга и управления нематериальными активами, что способствует принятию обоснованных решений и повышению эффективности управления интеллектуальной собственностью в IT-компаниях.

3.2. Песочница Iptimizer

Как было заявлено ранее, Iptimizer дает возможность задать структуру компании, в которой осуществляется управление нематериальными активами. Это стало возможным благодаря созданию песочницы.

Проведенные исследования в области управления нематериальными активами позволили выделить более 8-и типов данных, которые являются компонентами системы внутри Iptimizer. Система представляет из себя некоторую атомарную сущность, над которой осуществляется управление.

Сама структура компании представляет собой объект, который поддерживает любую степень вложенности и все настройки которого вынесены в Web-приложение, которое и работает с этим объектом. Необходимо было разработать визуальное представление такого объекта и интерфейса для его управления. Поскольку мы работаем со сколь угодно вложенной и настраиваемой структурой, данная задача представляет большую сложность.

С помощью песочницы, используя свойства выделенных типов данных и при помощи вложения одних типов данных в другие (для типов данных, которые поддерживают вложенность), Iptimizer позволяет описывать структуру компании любой сложности и вариативности (рис. 1).

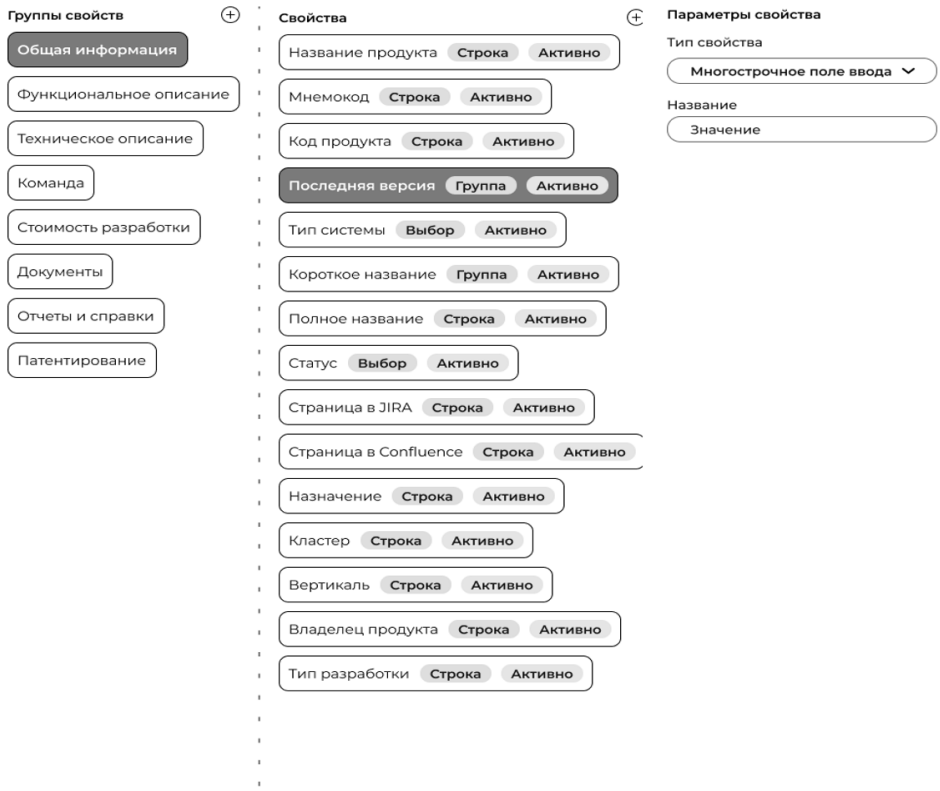


Рис. 1. Настройка структуры с помощью песочницы Iptimizer

После описания системы как самой низкоуровневой управляемой единицы следует описание более высокоуровневых категорий, например, отделов и подразделений. Это означает, что пользователи могут детально описать иерархию своей организации, включая отделы, подразделения, команды и отдельных сотрудников. Такая гибкость позволяет учитывать даже самые сложные и нестандартные структуры компаний.

Пользователи могут легко добавлять и редактировать элементы структуры, настраивать их свойства и взаимосвязи. Это позволяет быстро адаптировать Iptimizer к изменениям в организационной структуре компании, обеспечивая актуальность и точность данных.

3.3. Системы Iptimizer

После задания структурного базиса Систем в песочнице дальнейшее управление нематериальными активами осуществляется с помощью вкладки Системы.

Все наши системы будут представлены в виде карточек, содержащих определенную краткую информацию, которая предлагается в качестве базовой для первоначального получения информации, не вдаваясь в подробности.

Полную информацию о системе можно увидеть в более развернутом представлении после нажатия на соответствующую кнопку на карточке.

Система может быть как проектом, так и отделом, смотря, что внутри компании принимается за единицу управления. Как правило, под системой будем понимать конкретный продукт, над которым ведется разработка.

Поскольку продукты постоянно добавляются и выводятся из использования компаниями, Iptimizer позволяет легко создать систему, указав минимальный набор полей или отправить отдельную систему в Архив, чтобы не загромождать рабочее пространство.

У каждого продукта есть статус, в котором он находится, например, в разработке, в архиве или в эксплуатации. Это далеко не весь перечень статусов, но Iptimizer предполагает настройку и этого функционала. Для упрощения и дальнейших выкладок и сохранения коммерческой тайны Iptimizer не будем затрагивать рассмотрение статусов.

При просмотре подробной информации о системе (рис. 2), пользователь может заполнить обязательные поля, прикрепить ссылки на документации, заполнить сведения о команде, скачать загруженные файлы, в общем, заполнить все то, что он настроил в песочнице.

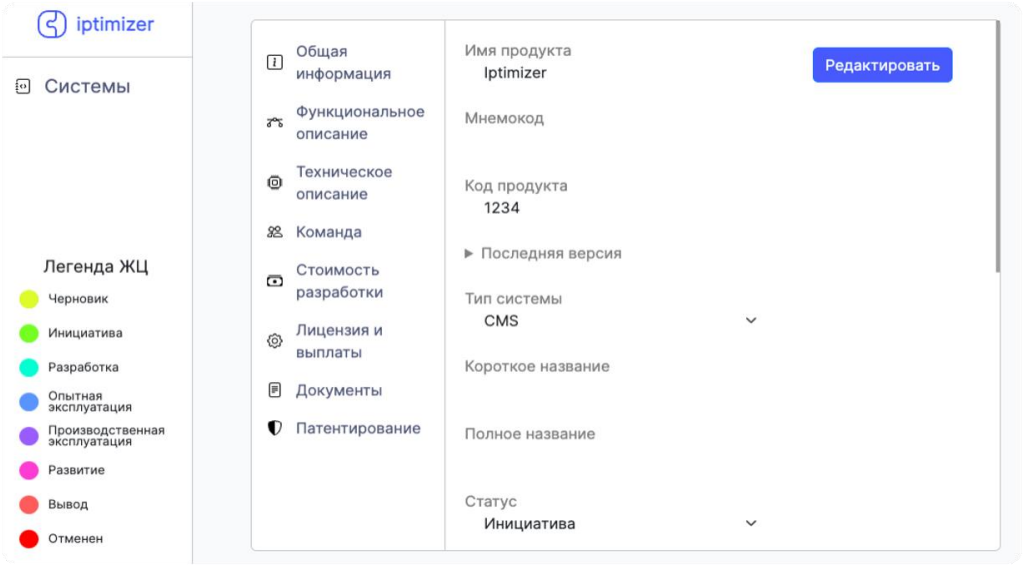


Рис. 2. Просмотр системы

После заполнения всех обязательных полей, пользователь вправе перевести статус проекта в следующий согласно настроенным статусам и отправить на подтверждение заполненности и проверку администратору.

Функциональность систем, в том числе, укомплектована удобным поиском по соответствующим позициям, которые можно настроить в песочнице; также есть фильтры и сортировки по различным параметрам, которые можно задать.

Функции, связанные с подготовкой различной отчетности, не будут рассматриваться в данной работе из соображений коммерческой тайны.

3.4. Сведения об архитектуре

Разработка продукта Iptimizer разделена на несколько стадий, называемых MVP. Iptimizer по своей концепции и идеологии является приложением, которое постоянно требует расширения и подготовки нового функционала для использования, поэтому архитектуру Iptimizer составляют отдельные микросервисы, выполняющие конкретные задачи (рис. 3).

Backend часть приложения разработана на языке C# с использованием фреймворков ASP.NET и .NET. Frontend часть разработана с помощью фреймворка Vue.js.

Микросервисная архитектура позволяет Iptimizer оставаться гибким и открытым ко внедрению новых функциональностей, которые открываются при изучении рынка продуктов по управлению нематериальными активами.

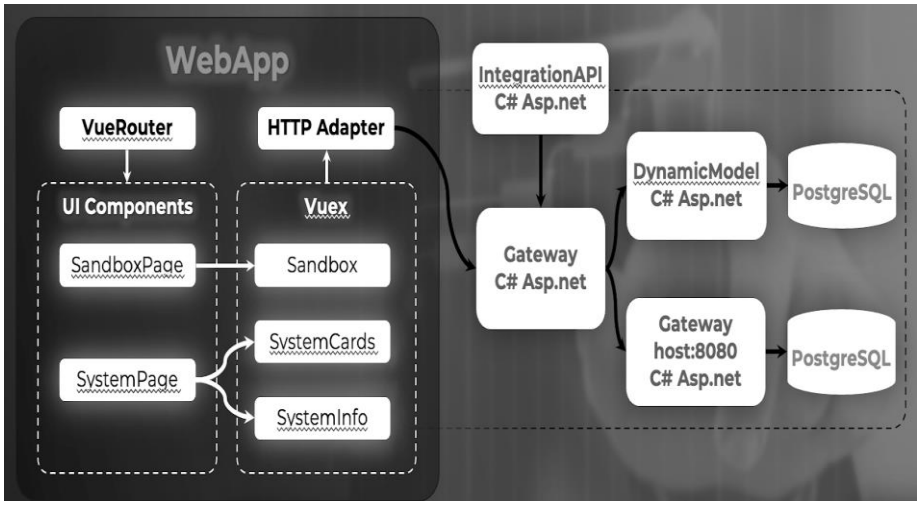


Рис. 3. Общая схема Iptimizer

Заключение

Управление нематериальными активами – актуальная задача для нашего времени, постоянное расширение областей, в которых появляются нематериальные активы предоставляют Iptimizer огромный потенциал для развития.

IT-компании, как никто, заинтересованы в добросовестном учете и качественном управлении нематериальными активами.

Сейчас ниша в сфере управления не занята, каждая компания закладывает часть бюджета, чтобы обезопасить себя в случае, если у налоговых проверок будут претензии. Компании организуют внутренние аудиты, которые повышают прозрачность и реальную стоимость продуктов и услуг компании, все это – периодические затраты, которые никогда не заканчиваются.

IT-компании используют продукты других компаний, чтобы сэкономить на разработке и внедрить то, что уже хорошо протестировано и используется, Iptimizer поможет юридически обезопасить процесс передачи прав и доступов между компаниями, подго-

товить договоры об отчуждении прав и интеллектуальной собственности, защитить акционеров и инвесторов, позволит повысить капитализацию за счет правильного описания трудозатрат компаний и трудозатрат отдельных команд.

Iptimizer – отличный инструмент для управления нематериальными активами.

ЛИТЕРАТУРА

1. *Рейли Р.Ф., Швайс Р.П.* Оценка нематериальных активов. – М.: Изд-во «Квинто-консалтинг», 2005. – 761 с.

2. *Лев Б.* Нематериальные активы. Управление, измерение, отчетность. – М.: Изд-во «Квинто-консалтинг», 2003. – 240 с.

3. *Селиванова Л.А., Городничев А.А.* Исследование предприятия на основе монитора нематериальных активов К.Э Свейби // Балтийский экономический журнал. 2010 – № 4 – С. 165–176.

WEB-ПРИЛОЖЕНИЕ ДЛЯ ТУРИСТОВ

Алхулаев А.М., Пахомова Е.Г.
Томский государственный университет
tuhum_05@mail.ru, PakhomovaEG@yandex.ru

Введение

В современном мире туризм играет значительную роль в жизни многих людей. Развитие технологий и интернета сделало путешествия более доступными, а многочисленные сайты и приложения существенно упростили планирование поездок. Однако в связи с изменением геополитической ситуации многие известные иностранные туристические сервисы не могут работать в России. В связи с этим появляется потребность создания отечественных альтернатив иностранным сервисам и предоставления туристам необходимой информации о местах отдыха в России.

Работа посвящена разработке веб-приложения для туристов, ориентированного исключительно на российские туристические места. Одной из ключевых проблем, с которой сталкиваются пользователи существующих сайтов, является ограниченная база данных по достопримечательностям и туристическим объектам России. Цель данной работы – создать платформу, которая будет не только содержать обширную и актуальную информацию о туристических местах в России, но и предоставлять удобные инструменты для планирования поездок.

1. Задачи

Разработка любого веб-приложения требует решения ряда ключевых задач. В нашем случае это будут следующие задачи.

Выбор технологий для разработки. Требуется выбрать стек технологий для фронтенда и бэкенда приложения; выбрать подходящую базу данных для хранения информации. Все выбранные технологии должны быть современными и достаточно эффективными.

Разработка пользовательского интерфейса. Интерфейс приложения должен быть удобен и интуитивно понятен. У пользователя не должно возникать проблем с поиском и просмотром информации.

Создание и поддержка базы данных. Веб-приложение будет собирать данные о туристических местах, достопримечательностях, природных объектах и т.п., поэтому необходимо создать базу данных, которая будет хранить информацию о географическом расположении, истории, часах работы, стоимости билетов и других важных аспектах каждого объекта.

Обеспечение безопасности и надежности. Пользователи будут регистрироваться на сайте, поэтому необходимо предусмотреть механизмы защиты персональных данных. Кроме того, необходимо обеспечить устойчивость приложения к нагрузкам. Выполнение этих задач позволит создать надежное, функциональное и удобное веб-

приложение, которое станет ценным ресурсом для туристов, путешествующих по России.

2. Стек технологий

Фронтенд-технологии. Для разработки фронтенд-части веб-приложения, предназначенного для туристов, были выбраны следующие современные технологии.

HTML (HyperText Markup Language) –язык гипертекстовой разметки текста, который применяют для создания и структурирования содержимого веб-страниц [1]. Также HTML необходим для создания элементов интерфейса, таких как таблицы, кнопки, ссылки и изображения [2].

CSS (Cascading Style Sheets) – язык, который описывает внешний вид документа. Другими словами, он определяет внешний вид веб-страницы (цвета фона, декоративные элементы, размеры и стили шрифтов). Для удобства был использован препроцессор CSS, а именно – SASS. Его использование упрощает и ускоряет написание CSS-кода, т.к. он расширяет возможности CSS. Благодаря SASS появляется возможность использовать CSS-константы, наследование, смешанные стили и т.д. Также он позволяет избегать многократного повторения одинаковых фрагментов кода [3].

JavaScript – является основным языком программирования, с помощью которого пишется функционал веб-страницы. Он необходим для обработки пользовательских событий, таких как клики, ввод данных и т.п. [4].

Для облегчения написания кода была выбрана библиотека jQuery. Данная библиотека является инструментом, упрощающим взаимодействие с DOM (представление HTML-документа в виде дерева тегов) и выполнение асинхронных запросов к серверу (т.е. AJAX-запросов) [5].

Бэкенд-технологии. Разработка бэкенда находится еще на начальной стадии. Но инструменты, необходимые для реализации серверной логики, управления данными и обеспечения безопасной аутентификации пользователей, уже определены. Для разработки серверной части приложения были выбраны следующие инструменты:

– Node.js – платформа для использования JavaScript на стороне сервера;
– Express.js – мощный и гибкий веб-фреймворк для Node.js, который облегчает создание веб-приложений;
– Passport.js – решение для реализации авторизации и аутентификации в web-приложении, построенном на Node.js;
– PostgreSQL – реляционная база данных, выбранная для хранения и обработки данных.

Выбор был обусловлен высокими показателями производительности и надежностью перечисленных систем.

3. Функционал приложения

Для того, чтобы веб-приложение выполняло свои функции и было удобно в использовании, необходимо обеспечить соответствующий функционал. Для создаваемого сайта он следующий:

Регистрация пользователя. Пользователь может зарегистрироваться, создав профиль с именем и электронной почтой. После регистрации он получит доступ к таким функциям, как добавление мест, написание отзывов и создание маршрутов. Доступ к профилю пользователя защищен паролем.

Добавление места. Данная функция предполагает, что пользователи могут добавлять новые туристические места с подробным описанием, фотографиями или видео. Кроме того, зарегистрированные пользователи могут оставлять отзывы о местах, добавленных другими пользователями.

Просмотр карт. Приложение будет предоставлять интерактивную карту с отметками туристических мест. При наведении мышки (или нажатии мышкой) на метку, бу-

дет отображаться краткая информация о месте (его название, фотографии и ключевые детали).

Создание маршрутов. Зарегистрированные пользователи смогут создавать собственные маршруты. При этом можно указывать удобные места остановок, мест отдыха и т.п. Маршруты будут сохраняться в профиле пользователя и ими можно будет делиться с другими пользователями.

Фильтрация и поиск. Пользователи смогут сортировать результаты по популярности, рейтингу и расстоянию. Будут доступны фильтры для поиска по категориям, видам деятельности, услугам и другим параметрам.

Модерация. Для обеспечения качества и правдивости информации в приложении будет внедрена система модерации. Все вновь добавленные места и обзоры будут проходить ручную модерацию, чтобы предотвратить возникновение неверных или несоответствующих данных. Модераторы будут проверять загруженные файлы и тексты отзывов на соответствие стандартам и правилам приложения.

4. Интерфейс сайта

Рассмотрим интерфейс нашего сайта Travelers Around. Начнем с главной страницы. На ней можно увидеть краткую информацию о возможностях сайта, навигацию для перехода в разделы "Каталог", "Сервис", "О нас". Также на главной странице имеется возможность создать профиль (кнопка "Регистрация") (рис. 1).

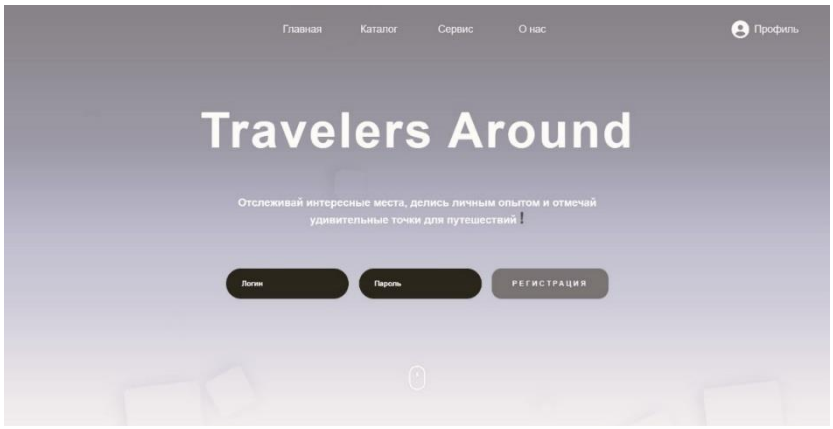


Рис. 1. Главная страница сайта

Чтобы пользователь мог войти в свой профиль, ему необходимо заполнить поля своим логином и паролем, и тогда кнопка "Регистрация" изменится на кнопку "Войти" (рис. 2).

Рис. 2. Вход в профиль

После регистрации и авторизации пользователю станет доступен функционал сайта (рис. 3).

Рис. 3. Главная страница, секция "Функционал"

После секции "Функционал" идет секция с кнопкой "ссылка-якорь" (рис. 4), которая перенаправляет пользователя на секцию "Рекомендуемые места" (рис. 6).

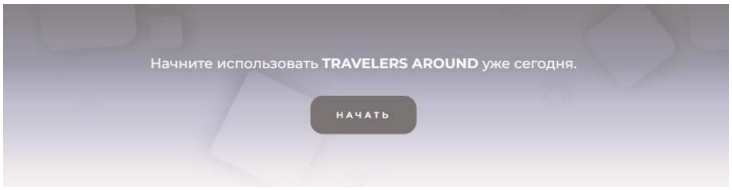


Рис. 4. Главная страница, секция с кнопкой "ссылка-якорь"

Рис. 5. Главная страница, секция с контентом

Рис. 6. Главная страница, секция "Рекомендуемые места"

Секция с ручным слайдером для просмотра отзывов о сайте (рис. 7).

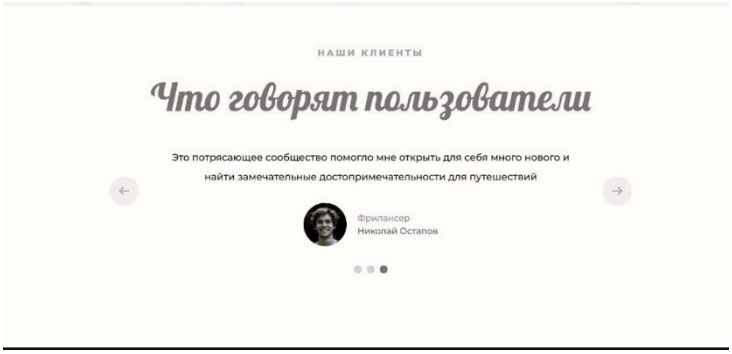


Рис. 7. Главная страница, секция с отзывами

Современные сайты уделяют большое внимание "подвалу" страницы. Он обычно предоставляет доступ к важной информации технического характера. На нашем сайте в "подвале" каждой страницы будет находиться навигация сайта, социальные сети и копирайт (рис. 8).

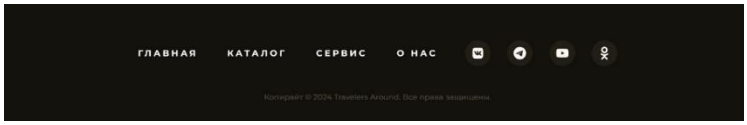


Рис. 8. "Подвал" сайта

Теперь перейдем к странице "Каталог". В первую очередь мы увидим то же, что и на главной странице, а именно – навигацию сайта и возможность зарегистрироваться или войти (рис. 1). Далее идет секция с примером списка мест. Несложно заметить, что места можно отсортировывать по разным критериям (рис. 9). После этого идет "подвал", как и на главной странице сайта (рис. 8).

Рис. 9. Страница "Каталог", список доступных мест

Далее перейдем к разделу "Сервис". На этой странице нас также встречает навигация сайта, возможность войти или зарегистрироваться (рис. 1), функционал добавления мест (рис. 10) и "подвал" сайта (рис. 8).

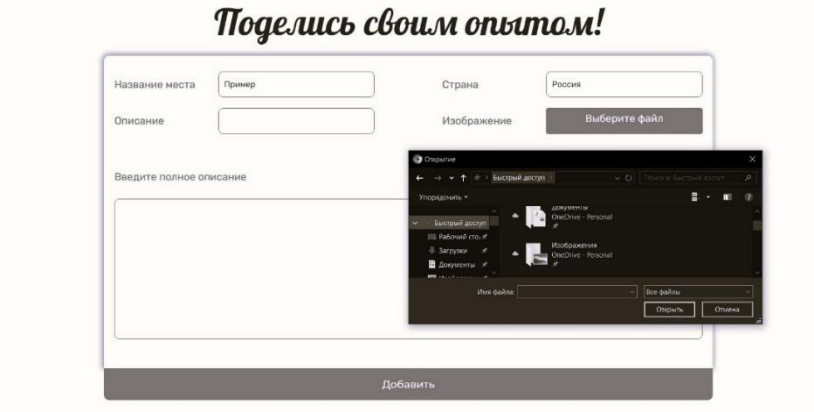


Рис. 10. Страница "Сервис", функционал добавления мест

Страница "О нас" – страница с информацией о сайте и его разработчиках (рис. 11).

Рис. 11. Страница "О нас"

5. Анализ программного кода

Рассмотрим и проанализируем программный код фронтенд-части нашего веб-приложения. Для начала рассмотрим архитектуру фронтенда (рис. 12).

Рис. 12. Архитектура фронтенд части

Как можно заметить, у нас 4 html-файла (каждый отвечает за свою страницу на сайте) и 1 css-файл который хранит в себе стилизацию и внешний вид веб-страниц, определяя шрифт, цвет, границы, анимации.

Теперь перейдем к папке "sass". Она хранит в себе папки, такие как "base" (рис. 13), которая в свою очередь хранит в себе файлы: "_base.scss" – хранит в себе новые базовые свойства для страниц; "_reset.scss" – обнуляет стандартные свойства; "_mixins.scss" – содержит в себе шаблоны, называемые миксинами, которые предоставляют способ повторного использования блоков CSS-кода; "_vars.scss" – хранит в себе переменные.

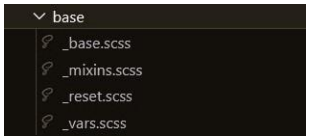


Рис. 13. Папка "base"

Папка "blocks" (рис. 14) содержит в себе файлы, которые отвечают отдельно за свой блок. Блоки, в свою очередь, позволяют определить части кода, которые можно использовать и переопределять в разных контекстах, аналогично миксинам, но с фокусом на более гибкое переопределение содержания.

Рис. 14. Папка "blocks"

Перейдем к папке "owl-carousel" (рис. 15). В ней хранятся обязательные файлы, которые хранят в себе описание стилей слайдера (рис. 8). Эти файлы импортируются вместе с установкой слайдера.

Рис. 15. Папка "owl-carousel"

Папка "sections" содержит в себе файлы-стили, относящиеся к различным секциям или разделам веб-страницы. Например, "_header.scss" – стили для секции заголовка, "_footer.scss" – стили для секции подвала, "_about-us.scss" – стили для страницы "О нас" и т.д. (рис. 16).

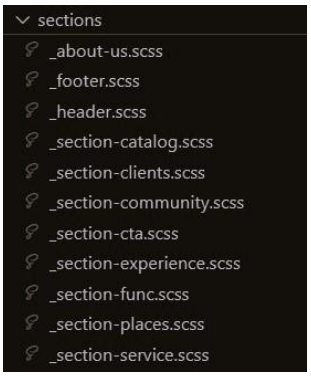


Рис. 16. Папка "sections"

Папка "utilities" (рис. 17) содержит в себе различные утилиты и вспомогательные функции.

Рис. 17. Папка "utilities"

Файл "main.scss" (рис. 18) в папке "scss" служит главной точкой входа для всех SCSS-файлов проекта. В этом файле импортируются все другие SCSS-файлы, чтобы собрать их в один итоговый CSS-файл.

```
@import './base/vars';
@import './base/mixins';

@import './base/reset';

// sections
@import './sections/header';
@import './sections/section-func';
@import './sections/section-cta';
@import './sections/section-community';
@import './sections/section-experience';
@import './sections/section-places';
@import './sections/section-clients';
@import './sections/footer';
@import './sections/section-catalog';
@import './sections/section-service';
@import './sections/about-us';
// blocks
@import './blocks/btn-round';
@import './blocks/form';
@import './blocks/title-subtitle';
@import './blocks/function-card';
@import './blocks/button';
@import './blocks/button-outline';
@import './blocks/button-solid';
@import './blocks/slider';
@import './blocks/navigation';
@import './blocks/glass-squares';
@import './blocks/place-card';
@import './blocks/nav-icon';
@import './blocks/nav';
@import './blocks/dropdown-select';
@import './blocks/create-button';
// utilities
@import './utilities/util';

// owl-carousel theme
@import './owl-carousel/vars';
@import './owl-carousel/theme';
@import './base/base';
```

Рис. 18. Файл "main.scss"

Перейдем к файлам JavaScript. Рассмотрим файл "model.js" –файл, отвечающий за управление данными и бизнес-логикой приложения [6]. Т.к. на данном этапе разработки у нас не проводилась работа с данными, файл пока пуст.

Далее перейдем к файлу "view.js". Он отвечает за представление данных пользователю и управление отображением. Прокомментируем код из нашего файла: у нас имеется объект elements, содержащий ссылки на различные элементы DOM, которые будут использоваться в методах класса [6] (рис. 19). Метод headerChangeFormButton в файле "view" отвечает за изменения текста кнопки. При вводе данных в поля формы, текст на кнопке автоматически изменяется в зависимости от заполненности полей логина и пароля.

Файл "controller.js" играет роль посредника между моделью "model.js" и представлением "view.js". Контроллер обрабатывает ввод пользователя, взаимодействует с моделью для обработки данных и обновляет представление, чтобы отразить изменения [6] (рис. 20).

```
export default class View {
    constructor(){

    };
    elements = {
        navBurger: document.querySelector('.nav'),
        navIcon: document.querySelector('.nav-icon'),
        navLinks: document.querySelectorAll('.nav a'),
        headerFormButton: document.querySelector('#headerFormButton'),
        headerFormInput: document.querySelectorAll('[data-input]'),
        headerFormLogin: document.querySelector('[data-input="login"]'),
        headerFormPassword: document.querySelector('[data-input="password"]'),
        dropDownBtn: document.querySelector('.dropdown-button'),
        dropDownList: document.querySelector('.dropdown-list'),
        dropDownListItem: document.querySelectorAll('.dropdown-list__item'),

    };

    tabnine: test | explain | document | ask
    headerChangeFormButton(){
        const elements = this.elements;
        elements.headerFormInput.forEach((input)=>{
            input.addEventListener('input', ()=>{
                if(elements.headerFormLogin.value !== '' && elements.headerFormPassword.value !== ''){
                    elements.headerFormButton.innerText = 'Войти'
                }else{
                    elements.headerFormButton.innerText = 'Регистрация'
                }

            });
        });
    }
```

Рис. 19. Файл "view.js"

```
window.onbeforeunload = () => sessionStorage.setItem('scrollPos', window.scrollY);
tabnine: test | explain | document | ask
window.onload = () => window.scrollTo(0, sessionStorage.getItem('scrollPos') || 0);

import Model from './model.js';
import View from './view.js';

const model = new Model();
const view = new View();

view.navBurgerView();
view.headerChangeFormButton();
view.dropDownList();
```

Рис. 20. Файл "controller.js"

Файл "clients-carousel.js" служит для инициализации и настройки карусели с использованием библиотеки Owl Carousel. Этот файл позволяет настроить параметры слайдера, такие как количество отображаемых элементов, скорость прокрутки, автоматическое воспроизведение и т.д. (рис. 21).

```
$(document).ready(function(){

    const owl = $('#clients-slider');
    owl.owlCarousel({
        items: 1,
        loop: true,
    });

    const prev = $('#owlSliderPrev');
    const next = $('#owlSliderNext');

    prev.click(function(){
        owl.trigger('prev.owl.carousel');

    });

    next.click(function(){
        owl.trigger('next.owl.carousel');

    })
});
```

Рис. 21. Файл "clients-carousel.js"

Заключение

Таким образом, поставленная задача по разработке фронтенд-части веб-приложения для туристов была успешно выполнена. Приложение корректно отображает данные о туристических местах в России, предоставляет пользователям возможность регистрироваться и авторизовываться, добавлять новые места с фотографиями и видео, фильтровать и искать информацию в базе данных. Также оно обладает удобным и привлекательным пользовательским интерфейсом.

В дальнейшем планируется разработка и интеграция бэкенд-части приложения. Будет реализована обработка данных, хранение информации в базе данных, работа с картой и маршрутами и расширение функционала.

ЛИТЕРАТУРА

1. *Фрэйн Б.* Отзывчивый дизайн на HTML5 и CSS3 для любых устройств. 3-е изд. – СПб.: Питер, 2022. – 336 с.

2. Бесплатный курс по верстке сайтов (Front End) [Электронный ресурс] // YouTube канал про разработку сайтов. URL: https://www.youtube.com/watch?v=yJcCKuxfb2o&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&ab_channel=Фрилансерпожизни-ITифриланс (дата обращения: 01.04.2024)

3. Документация SCSS [Электронный ресурс] // Sass: Документация на русском языке. URL: https://sass-scss.ru/documentation/ (дата обращения 01.04.2024)

4. *Haverbeke M.* Eloquent JavaScript (4th ed.). No Starch Press. – URL: https://eloquentjavascript.net/ (дата обращения 01.04.2024)

5. Документация jQuery [Электронный ресурс] // jQuery API Documentation. URL: https://api.jquery.com/ (Дата обращения 01.04.2024)

6. Документация MVC [Электронный ресурс] // MDN Web Docs: MVC Design Pattern. URL: https://developer.mozilla.org/en-US/docs/Glossary/MVC (дата обращения 01.04.2024)

СОЗДАНИЕ ПРИЛОЖЕНИЯ, ОТСЛЕЖИВАЮЩЕГО ИЗМЕНЕНИЯ РАСПИСАНИЯ

Андреева А.А., Пахомова Е.Г.

Томский государственный университет

aandreeva1313@mail.ru, pakhomovaeg@yandex.ru

Введение

Реалии учебного процесса в вузе таковы, что расписание корректируется практически каждую неделю. Кроме того, возможны непредвиденные ситуации, когда его приходится изменять в середине недели. Это приводит к необходимости постоянно следить

за расписанием. Поэтому создание ресурса, который бы отслеживал изменения и своевременно сообщал об этом студентам, было бы очень полезно для учебного процесса [1]. Один из вариантов реализации этой задачи – создать мобильное приложение.

Мобильные приложения играют ключевую роль в современном мире, обеспечивая удобный доступ к информации и услугам. Они быстрее загружают контент и легче в использовании, по сравнению с аналогичными веб-сайтами. Их дизайн позволяет подстраиваться под разные размеры экрана. Они позволяют пользователям создавать личные учетные записи и хранить важную информацию под рукой и, в случае необходимости, отправляют push-уведомления, информирующие пользователя об изменениях в реальном времени. Т.о., использование мобильных приложений, которые можно установить на смартфоны, имеющиеся почти у каждого, упрощает повседневную жизнь людей.

Именно поэтому было решено создать мобильное приложение, которое проверяет расписание через определенные интервалы времени и уведомляет пользователей о любых изменениях, включая отмены занятий, переносы, изменения преподавателей, аудиторий или корпусов. Оно предназначено для тех, кто хочет вовремя получать актуальную информацию об изменениях в расписании, но не имеет возможности постоянно отслеживать информацию на сайте. Такое приложение позволяет сэкономить время и делает образовательный процесс более комфортным для всех его участников.

Для достижения поставленной цели потребовалось проанализировать несколько сайтов и приложений с расписанием томских ВУЗов, изучить язык программирования Kotlin для написания приложения, способы автоматизированного сбора информации с интернет сайтов и функциональность платформы разработки мобильных приложений firebase, а также разработать и реализовать приложение.

На текущий момент, приложение создано и способно отслеживать расписание Томского государственного и Томского политехнического университетов, а также расписание Томского государственного университета систем управления и радиоэлектроники. Для каждого из этих ВУЗов существует сайт с расписанием, а у ТГУ также есть и соответствующее мобильное приложение, которое называется TSU.InTime. Однако ни один из сайтов не уведомляет пользователя о каких-либо изменениях. В TSU.InTime такая функция предусмотрена, но уведомления в нем отправляются некорректно, т.е. пользователи получают информацию об изменениях не только своего расписания, но и всех других. Помимо вышеупомянутых сайтов, существует еще и приложение "Дневач", в котором можно смотреть расписание различных ВУЗов, в том числе и некоторых томских, однако в этом приложении также отсутствует функция отслеживания изменений. Поэтому создание такого ресурса действительно полезно для студентов перечисленных учебных заведений.

1. Стек технологий

1. Клиентская часть

Мобильная разработка бывает нативная и кроссплатформенная. Выбор подхода зависит от функций будущего приложения, целевой аудитории, бюджета и других факторов. Для данного приложения был выбран именно нативный подход.

Данное приложение разрабатывалось для устройств с операционной системой Android, т.к. она обеспечивает высокий уровень безопасности, что важно для защиты пользователей данных, а также предоставляет мощные инструменты, упрощающие разработку приложений, например, Android Studio. Кроме того, согласно данным с сайта Statcounter, на апрель 2024-го года Android занимает 71.31% рынка [2] и работает на более чем 24 000 устройствах 1300 разных брендов, что дает приложению доступ к огромному количеству пользователей [3].

Для разработки использовался язык программирования Kotlin. Это типизированный объектно-ориентированный язык, работающий "поверх" JVM (Java Virtual

Machine), которая является основной частью исполнительной системы Java. Этот язык появился в 2011-м году, а в 2017-м получил официальный статус от компании Google в качестве инструмента для работы на Android Studio [4].

2. Серверная часть

В серверной части данного приложения основной задачей является парсинг сайтов с расписаниями ТГУ, ТПУ и ТУСУРа.

Парсингом сайтов называется автоматический сбор информации из интернета и структурирование полученных данных. Программа ищет информацию по заданным параметрам на определённых сайтах. Данные собираются, систематизируются и приводятся к подходящему формату. После этого их можно обрабатывать и использовать.

Для данного приложения был выбран язык программирования Python, отличающийся своей гибкостью и универсальностью, а также простотой и лаконичностью синтаксиса. Весомым его достоинством является также то, что он предоставляет широкий ассортимент библиотек для парсинга. Вдобавок, на выбор языка повлияло наличие опыта программирования на нем и знание некоторых его основ.

3. Базы данных

Базы данных – это набор структурированных данных. Для этого приложения была выбрана база данных Firebase Realtime – облачная база данных NoSQL, которая позволяет хранить и синхронизировать данные в формате JSON между пользователями в режиме реального времени. Приложения Firebase остаются отзывчивыми даже в автономном режиме, т.к. SDK Firebase Realtime Database используют локальный кеш на устройстве для обслуживания и хранения изменений. Firebase Realtime также интегрируется с аутентификацией Firebase, обеспечивая простую аутентификацию для разработчиков. Кроме того, на платформе Firebase предусмотрен сервис Cloud Messaging, который позволяет отправлять пуш-уведомления [5].

2. Проектирование приложения

1. Разработка клиентской части

Разберем архитектуру клиентской части разрабатываемого приложения с помощью диаграммы классов. Рассмотрим, какие классы были определены в приложении, их функциональность, а также то, как они взаимодействуют друг с другом (рис. 1).

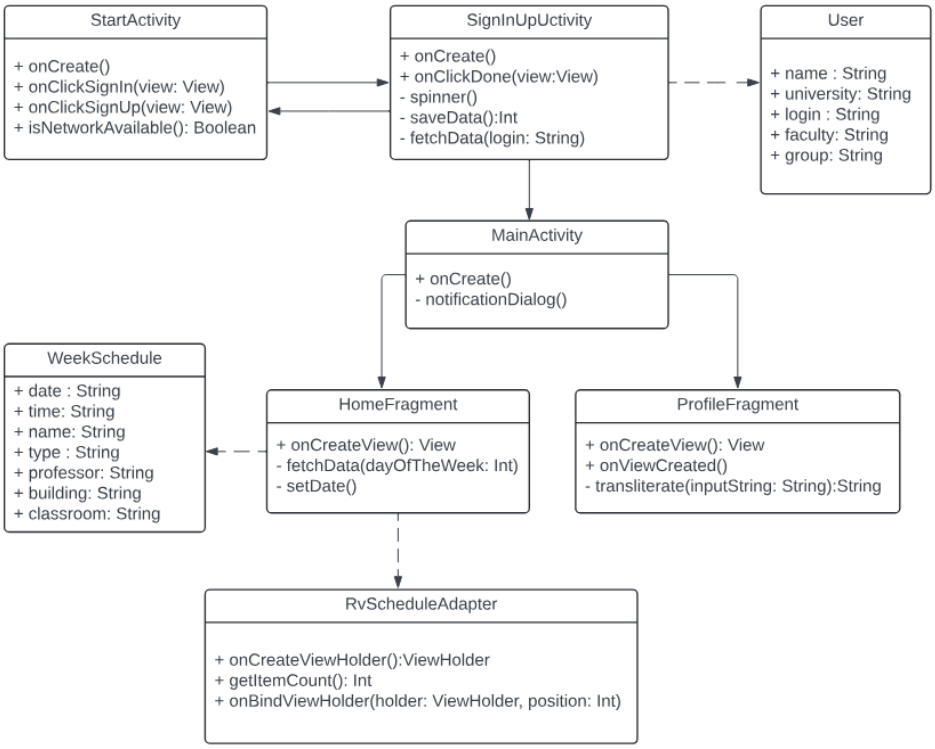


Рис. 1. Диаграмма классов

Класс User содержит в себе информацию о клиенте – его имя, ВУЗ, факультет, группу и логин.

Класс WeekSchedule содержит в себе информацию о занятии: дату и время, название, тип занятия (практика, лекция, семинар, и т.д.), имя преподавателя, аудиторию и корпус, в котором это занятие проходит.

Класс StartActivity – класс, который формируется при открытии приложения. В нем выполняется проверка на подключение к интернету и при его наличии еще одна проверка на то, выполнен ли вход, после чего дается возможность перейти к классу SignInUpActivity.

Класс SignInUpActivity –класс, формирующийся для регистрации или входа пользователя. В зависимости от того, на какую кнопку нажмет пользователь на начальном экране, на этом появляются разные поля для ввода данных и выпадающие списки. В этом классе также выполняется подключение к облачной базе данных firebase realtime.

При регистрации введенные пользователем данные считываются и выполняется его авторизация с помощью сервиса firebase. При этом вся остальная информация сохраняется в таблицу users в базе данных реального времени. На почту пользователя отправляется письмо, с просьбой подтвердить email. Производится проверка, успешно ли прошла авторизация, затем производится возврат на начальный экран.

При входе по введенному пользователем логину, если он был заранее подтвержден, в базе данных ищется информация о человеке. После нахождения она локально сохраняется на устройстве с помощью встроенного класса SharedPreference. Затем, при успешной аутентификации, происходит переход к классу MainActivity.

В классе MainActivity организована возможность переключаться между фрагментами соответствующего экрана HomeFragment и ProfileFragment; при первом вхожде-

нии вызывается диалоговое окно с помощью встроенного класса AlertDialog с просьбой разрешить получение пуш-уведомлений.

Класс ProfileFragment представляет информацию о текущем пользователе, которая ранее была сохранена на устройстве. Сдесь же производится подписка на рассылку уведомлений об изменениях расписания группы студента, использующего приложение.

В классе HomeFragment происходит получение данных из БД. В зависимости от ВУЗа и группы пользователя формируется название таблицы, к которой происходит обращение. Из нее извлекается расписание текущей недели; также происходит вычисление дат начала и конца текущей недели.

Класс RvScheduleAdapter используется для представления каждой карточки, на которой располагается название и время занятия, а также остальная информация о нем.

Помимо этих классов в приложении также представлен объект Constance, который хранит константы, в том числе списки университетов г. Томска (ТГУ, ТПУ, ТУСУР), факультетов для каждого из них и соответствующих групп.

Почти для каждого из перечисленных классов существует соответствующий файл с разметкой. В этих файлах для всех экранов описываются элементы, такие как текст, кнопки, списки и т.д., а также их параметры, например, размер, цвет, расположение и т.д.

2. Разработка серверной части

Серверная часть приложения написана на языке программирования Python в среде программирования pyCharm. Общая схема работы программы представлена на рис. 2.

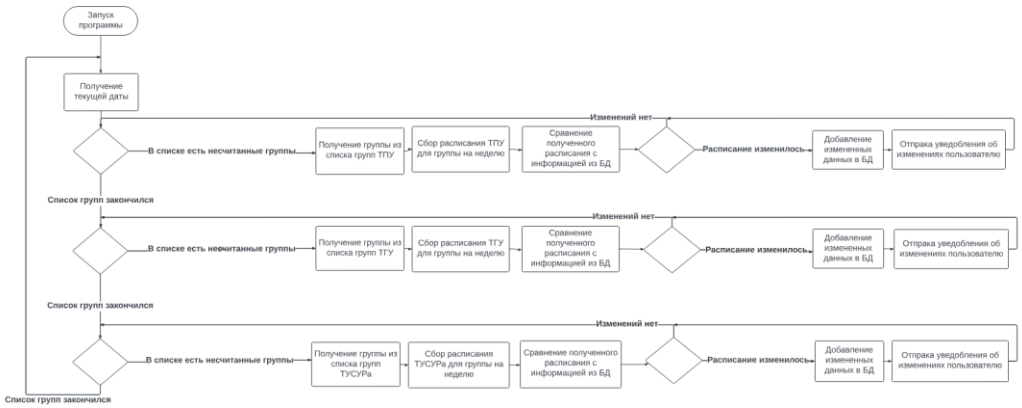


Рис. 2. Схема работы серверной части приложения

В основном файле main.py располагаются все необходимые функции, а также сам алгоритм сравнивания расписания. Рассмотрим подробнее его содержание.

Первым делом производится подключение к базе данных с помощью библиотеки firebase_admin и функции initialize_app(). Затем определяются несколько словарей, необходимых для приведения расписаний трех разных вузов к одному формату, после чего задаются функции parsingTSU, parsingTPU и parsingTUSUR, производящие сбор информации с сайтов ТГУ, ТПУ и ТУСУРа соответственно.

Каждой из этих функций на вход поступают словарь, состоящий из номеров групп и их идентификаторов, а также либо даты начала и конца недели, либо номер недели. Все значения подставляются в url-строку, однозначно определяющую нахождение ресурса в интернете, после чего производится попытка получить информацию о расписании с сайта по этой ссылке. Для расписания ТГУ эти данные хранятся в файле в формате json. Для их получения применяются библиотеки json и request. В функции parsingTPU сначала инициализируется драйвер браузера и производится попытка от-

крытия сайта с расписанием по переданной ссылке. В случае успеха с помощью метода page_source() получается код страницы в формате html, и сеанс браузера завершается командой quit(). Это происходит с помощью библиотек selenium и webdriver. Для ТУСУРа с помощью библиотеки request создается сессия, являющаяся связью с определённым браузером. Затем происходит считывание данных с сайта по ссылке в html формате. Из полученного html-файла извлекаются данные с помощью библиотеки BeautifulSoup. Из этих данных выбирается нужная информация, которая записывается в словарь. Из словарей составляется список со всеми парами текущей недели, который затем упорядочивается по дате и времени.

Помимо этих трех функций в программе есть функции для сравнения расписаний.

Функция weekdaySchedule возвращает расписание на конкретный день недели и некоторый индекс. На вход ей поступает список с расписанием на всю текущую неделю, индекс, являющийся счетчиком элементов в этом списке, а также номер дня недели. Если счетчик меньше числа занятий в расписании, то до тех пор, пока дата пары соответствует заданному дню недели, она добавляется в расписание на этот день. Эта функция является вспомогательной и используется в следующей.

Функция scheduleChangesWeekday в качестве параметров получает расписание текущей недели и расписание, хранящееся в базе данных. Сначала счетчики для обоих списков задаются равными нулю. Запускается цикл по дням недели, начиная с понедельника и заканчивая субботой. Для каждого из расписаний вызывается предыдущая функция и извлекается список с занятиями на этот день. Запоминается номер счетчика, который используется для следующего дня недели.

Например, нужно получить расписание на понедельник. Согласно текущему расписанию в этот день – три пары, а в базе данных написано, что – две, тогда weekdaySchedule вернет индексы 3 и 2 соответственно. Затем, при получении расписания на вторник, поиск занятий начнется сразу с этих индексов.

После создания этих списков списки сравниваются. Если их длина различна – соответствующий день недели сохраняется, если одинакова – они сопоставляются друг другу поэлементно, и наличие изменений также учитывается. После завершения цикла функция возвращает список номеров дней недели, для которых расписание было перестроено.

Функция checkingChanges получает на вход расписание текущей недели и название таблицы. Происходит подключение к базе данных и считывание информация из указанной таблицы. С помощью функции scheduleChangesWeekday происходит проверка на наличие изменений. Если они есть, то содержимое таблицы в базе данных перезаписывается, список номеров дней недели преобразуется в строку и отправляется в качестве пуш-уведомления на устройство пользователя, расписание группы которого было изменено.

В основном алгоритме запускается бесконечный цикл. Определяется текущая дата и номер недели. Из файла, в котором хранятся словари и списки с группами, извлекаются группы ТПУ. Запускается цикл по ним, в котором задается название таблицы по шаблону week_schedule_tpu_group_id и запускается функция parsingTPU. При успешном получении расписания вызывается функция сравнения. Затем те же манипуляции производятся с группами ТГУ и ТУСУРа.

3. Интерфейс приложения

Когда пользователь открывает приложение, перед ним появляется экран (рис. 3) с двумя кнопками: вход и регистрация. При нажатии на кнопку регистрации открывается соответствующий экран. На нем расположены несколько полей для ввода данных о пользователе. Пользователю предлагается ввести имя, почту и пароль, а также выбрать свой университет, факультет и группу с помощью выпадающих списков. Помимо этих полей, на экране расположена кнопка, при нажатии на которую выполняется авториза-

ция пользователя. Если пользователь не заполнил все поля и нажал на кнопку, появится сообщение об этом, и регистрация не произойдет до тех пор, пока всё не будет заполнено. После завершения этой процедуры происходит возврат на стартовый экран. В случае, если при регистрации произойдет ошибка, пользователь будет об этом оповещен.

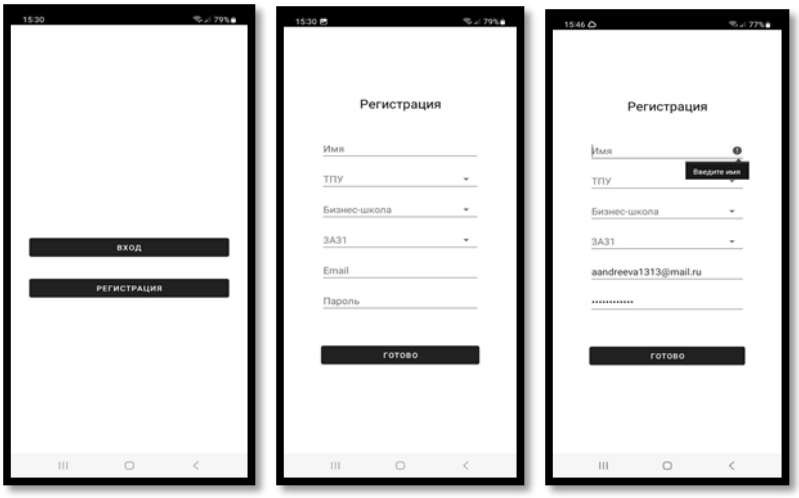


Рис. 3. Стартовый экран и экран регистрации

При выборе на начальном экране второй кнопки, открывается экран (рис. 4), позволяющий пользователю войти в приложение. Для этого нужно ввести почту и пароль. В случае, если человек ошибется при вводе, появится всплывающее сообщение о том, что логин или пароль написан неверно.

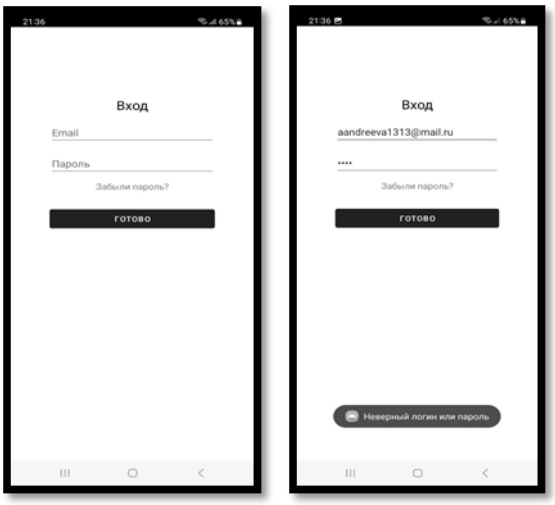


Рис. 4. Экран входа

Если пользователь забыл пароль, он может нажать на соответствующую надпись под полями ввода данных. Тогда откроется диалоговое окно, в нем надо будет ввести свой email, на который придет письмо со ссылкой, позволяющей поменять пароль на новый.

При успешном попадании пользователя в систему, перед ним появляется диалоговое окно с просьбой разрешить отправку уведомлений об изменениях в расписании (рис. 5). Если он нажмет "разрешить" – откроются настройки уведомлений приложения,

где он сможет включить оповещения; если же он пропустит это сообщение, то оно появится снова при следующем входе. Если оповещение о смене расписания включено и в какой-то момент расписание группы студента, использующего это приложение, будет изменено, ему на устройство придет уведомление (рис. 5). В нем будет сказано, в какие конкретно дни недели произошли перемены, после чего он сможет зайти в приложение и посмотреть, в чем именно состояла перестройка расписания.

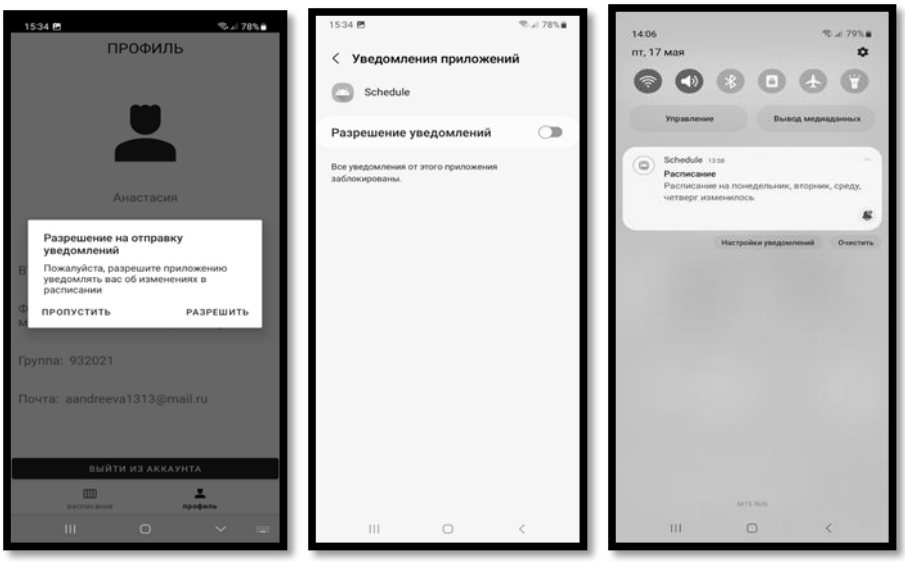


Рис. 5. Отправка уведомлений

После закрытия диалогового окна становится виден экран с профилем пользователя (рис. 6), на котором отображена информация о нем, а также кнопка выхода из аккаунта, при нажатии на которую происходит возврат на стартовый экран. При повторном входе в приложение, если пользователь не выходил из аккаунта, откроется сразу экран с профилем. Снизу экрана располагается меню, на котором расположены две кнопки. При нажатии на левую кнопку можно открыть расписание текущей недели (рис. 6). Сверху показаны даты начала и конца недели и панель с выбором дня недели. Кликая на элементы этой панели, можно просматривать информацию о занятиях в этот день. Если пар нет, на экране появляется соответствующая надпись.

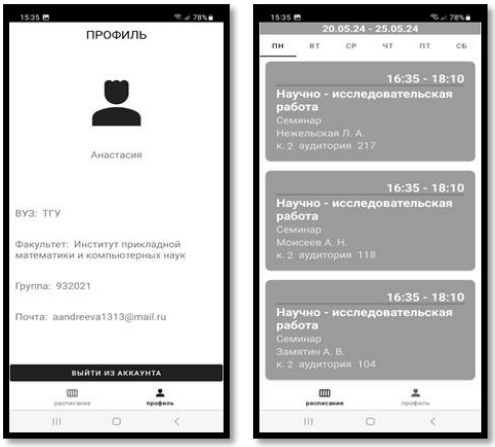


Рис. 6. Экраны с профилем и расписанием

Заключение

В данной работе была представлена разработка и реализация приложения отслеживающего изменения в расписании. Основным требованием при реализации этого приложения являлось требование отправки уведомлений студентам о любых изменениях расписания, таких как переносы и отмены занятий, смены аудиторий или преподавателей.

В процессе работы было разработано и реализовано мобильное приложение, способное показывать расписание на текущую неделю для студентов трех томских ВУЗов: ТГУ, ТПУ и ТУСУРа. Была добавлена возможность получать уведомления прямо на устройство пользователя о любых изменениях в расписании группы, указанной им при регистрации.

Несмотря на достигнутые результаты, данная работа оставляет место для дальнейшего совершенствования. Для этого планируется загрузить программу на сервер для беспрерывной работы, а также добавить возможность отслеживания расписания преподавателей и просмотра графика не только текущей недели, но и других. Помимо этого, хорошим дополнением к функциям приложения стала бы опция добавления в расписание личных дел и планов пользователя.

В заключение можно сказать, что разработанное приложение, отслеживающее изменения расписания и своевременно сообщающее об этом студентам, очень полезно для учебного процесса. Оно помогает тем, кто хочет вовремя получать актуальную информацию о смене графика, но не имеет возможности или желания постоянно отслеживать информацию на сайте. Такое приложение позволяет сэкономить время и делает образовательный процесс более комфортным.

С полным кодом приложения можно ознакомиться по ссылке: https://github.com/Nastyand/Schedule.git

ЛИТЕРАТУРА

1. *Андреева А.А., Пахомова Е.Г.* Создание Telegram-бота, отслеживающего изменения в расписании // Материалы X-й Международной молодежной научной конференции "Математическое и программное обеспечение информационных, технических и экономических систем", Томск, 26-29 мая 2023 г. – Труды Томского государственного университета. Т. 308: Серия физико-математическая. – Томск, 2023. – С. 95 – 101.

2. Доля рынка мобильных операционных систем по всему миру [Электронный ресурс]. – URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата обращения: 21.05.2024)

3. Роль Android [Электронный ресурс]. – URL: https://www.android.com/intl/ru_ru/everyone/enabling-opportunity/ (дата обращения: 21.05.2024)

4. Java vs Kotlin – большой обзор [Электронный ресурс]. – URL: https://www.sravni.ru/kursy/info/java-vs-kotlin/ (дата обращения: 21.05.2024)

5. Firebase Realtime Database [Электронный ресурс]. – URL: https://firebase.google.com/docs/database (дата обращения: 21.05.2024)

ОЦЕНКА СКОРОСТИ ВСТРАИВАНИЯ ВОДЯНЫХ ЗНАКОВ ПРИ ПОКАДРОВОМ И ВНУТРИКАДРОВОМ РАСПАРАЛЛЕЛИВАНИИ

Анжин В.А.
Томский государственный университет
viktor.anjin@gmail.com

Введение

Развитие технологий производства и доставки мультимедиа материалов существенно увеличило объёмы их производства и потребления. При этом всё более актуальной становится задача защиты авторских прав на цифровой контент, несанкционированное копирование и распространение которого приводит к потере доходов производителей или легальных дистрибьюторов.

Одним из подходов к защите авторских прав на мультимедиа материалы является использование водяных знаков. В случае утечки материала водяные знаки могут быть использованы для ограничения распространения пиратской копии. При внедрении уникальной для каждого из клиентов метки цифровые водяные знаки могут использоваться для поиска клиента, допустившего несанкционированное копирование и распространение материала [1].

Встраивание водяных знаков в изображение может осуществляться в разных вариантах представления данных, называемых областями внедрения [2]. Алгоритмы, использующие пространственную область, непосредственно манипулируют значениями яркости пикселей картинки. Алгоритмы, использующие частотную область, осуществляют внедрение водяного знака за счёт корректировки частотных характеристик, и требуют предварительного перевода изображения в частотную область. В общем случае алгоритмы и пространственной, и частотной области внедрения требуют побайтовой обработки данных кадра, что делает встраивание водяных знаков ресурсоёмкой задачей. Скорость встраивания водяного знака при работе на многопроцессорной системе может быть увеличена за счет организации параллельной обработки [3].

В рамках данной работы рассматриваются подходы к распараллеливанию реализаций алгоритмов встраивания водяных знаков. В первой части описывается алгоритм E_BLIND/D_LC [4], выбранный для оценивания подходов к распараллеливанию. Во второй части рассматриваются и оцениваются подходы к внутрикадровому распараллеливанию. В третьей части сравнивается эффективность покадрового и внутрикадрового распараллеливания.

Исходный код на языке C++, реализованный в рамках исследования алгоритма встраивания водяного знака E_BLIND/D_LC, реализации различных подходов к его распараллеливанию и используемые при проведении тестов изображения доступны в [5].

1. Описание алгоритма E_BLIND/D_LC

Алгоритм E_BLIND/D_LC использует пространственную область для внедрения однобитного сообщения m в исходный кадр Co. Результатом работы является кадр Cw. Для встраивания сообщения используется шаблонный кадр Wr, имеющий размеры сторон, соответствующие исходному кадру Co. Значение Wm выбирается исходя из встраиваемого бита. Настройка свойств устойчивости и видимости [6] алгоритма осуществляется через выбор значения α, используемого для масштабирования значений шаблонного кадра Wr:

$$\mathrm{Wm} = \begin{cases} \mathrm{Wr}, & \mathrm{m} = 1, \\ -\mathrm{Wr}, & \mathrm{m} = 0, \end{cases} \quad \mathrm{Wa} = \alpha \cdot \mathrm{Wm}, \ \mathrm{Cw} = \mathrm{Co} + \mathrm{Wa}.$$

В рамках описанного процесса можно выделить подготовительный этап, в ходе которого выбирается шаблонный кадр, осуществляется его масштабирование и корректировка знака, и следующий за ним этап внедрения, заключающийся в попиксельном суммировании полученного на подготовительном этапе кадра Wm и исходного кадра Co. Набор действий, осуществляемых на этих этапах, отличается от одного водяного знака к другому, но этап внедрения в общем случае требует попиксельного обхода кадра и выполнения некоторых арифметических действий. Исходя из этого, можно сделать предположение, что скорость внедрения водяных знаков, использующих пространственную или частотную область при различных подходах к распараллеливанию, будет изменяться в соответствии с тенденциями изменения скорости внедрения, полученными при оценке реализации алгоритма E_BLIND/D_LC.

2. Оценка подходов внутрикадрового распараллеливания

Для осуществления эффективной параллельной реализации нужно разбить задачу на независимые подзадачи. В рамках алгоритма внедрения E_BLIND/D_LC воздействие, осуществляемое на некоторый пиксель, зависит только от него самого и соответствующего ему пикселя масштабированного шаблонного кадра Wa, и не оказывает влияния на изменения значений других пикселей. Это даёт возможность произвольным образом разбивать кадр для осуществления его параллельной обработки. Рассмотрим разбиение кадра по группам горизонтальных строк и оценим его в сравнении с разбиением по группам столбцов (рис. 1).

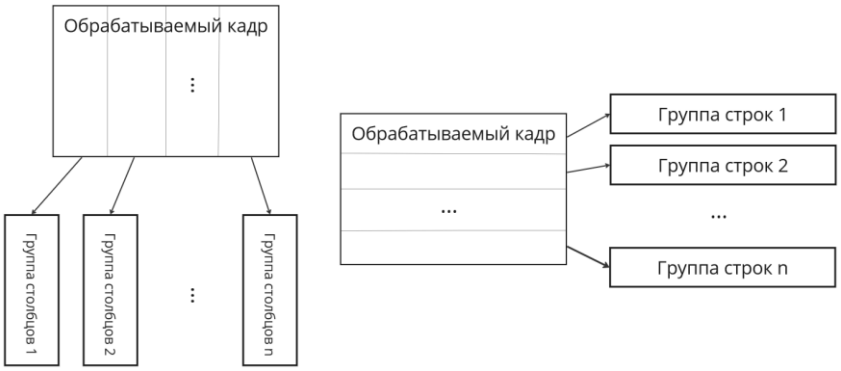


Рис 1. Разбиение кадра по столбцам(слева) и строкам(справа)

Для оценивания вариантов распараллеливания осуществлена программная реализация на языке C++, доступная по ссылке [5]. Измерения производительности проводились на компьютере с процессором Intel Core i7-12700, имеющем 12 независимых ядер и 20 логических процессоров. Оценка производилась на группах изображений различных размеров: группа SD включает изображения малого размера (размер варьируется около 640x480 пикселей), группа HD – изображения с размером, варьирующимся около 1920x1080 пикселей, и группа 4K – большие изображения размером более чем 4032x3000 пикселей. В столбцах табл. 1 указано количество параллельно работающих потоков, в ячейках – средние времена встраивания водяного знака в кадр в миллисекундах.

Таблица 1

Результаты замера скорости внедрения

	Количество потоков распараллеливания										
	1	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
SD столбцы	1.74	0.93	0.51	0.39	0.36	0.39	0.37	0.358	0.32	0.31	0.32
SD строки	1.75	0.93	0.49	0.37	0.35	0.39	0.35	0.35	0.34	0.38	0.33
HD столбцы	15.59	8.17	4.35	3.28	2.84	3.04	2.83	2.83	2.66	2.51	2.49
HD строки	15.62	8.18	4.32	3.21	2.75	2.96	2.75	2.77	2.60	2.46	2.41
4K столбцы	42.99	22.08	11.55	8.30	7.08	8.00	7.45	7.19	6.84	6.53	6.42
4K строки	43.08	22.32	11.48	8.29	6.98	7.79	7.36	7.10	6.75	6.44	6.31

Визуально результаты замера скорости, полученные в рамках эксперимента, представлены на рис. 2, отображающем зависимость скорости внедрения водяного знака исходя из типа распараллеливания (по группам строк и столбцов), размера кадра и количества потоков. Горизонтальная ось указывает число использующихся в параллельном режиме потоков. По вертикальной оси отложено время (в миллисекундах) внедрения водяного знака в кадр.

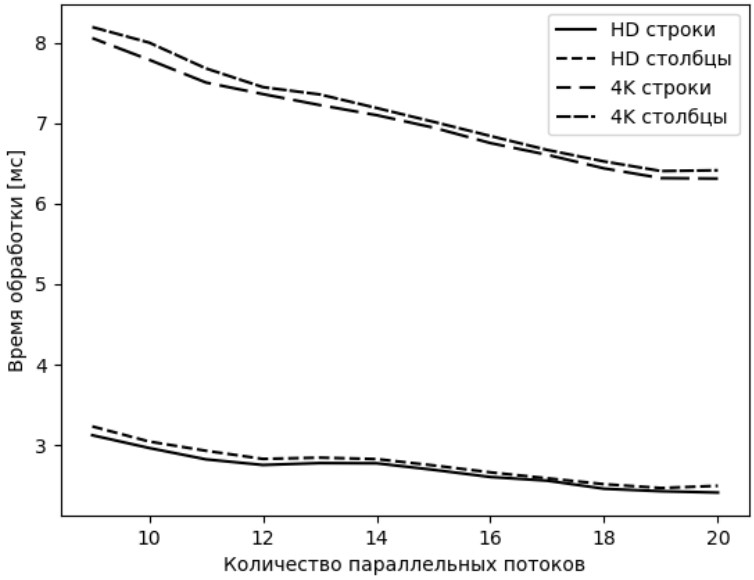


Рис 2. Зависимость скорости внедрения водяных знаков (в миллисекундах) от числа параллельных потоков

Из полученных результатов видно, что для изображений групп HD и 4K скорость внедрения при параллельной обработке по строкам выше скорости внедрения при параллельной обработке по столбцам. Одна из возможных причин полученных результатов заключается в том, что при параллельной обработке по строкам каждый из потоков имеет большую, по сравнению с параллельной обработкой по столбцам, область непрерывной памяти. За счёт этого повышается вероятность попадания обрабатываемых данных в кэш-память, что приводит к ускорению работы [7]. В рамках проведённого эксперимента для маленьких картинок размером SD, тип внутрикадрового распараллеливания не оказывает влияния на скорость внедрения. Причина этого может заключаться в том, что накладные расходы распараллеливания и влияние сторонних приложений многопользовательской системы оказываются более существенными, чем различия в скорости между разными типами распараллеливания на маленьком размере картинок и используемом в эксперименте компьютере.

3. Оценка скорости при покадровом и внутрикадровом распараллеливании

Параллельная обработка может быть осуществлена как внутри кадра, так и на уровне отдельных кадров. Для организации внутрикадрового распараллеливания, как видно из проведённого ранее эксперимента, наиболее эффективно использовать разбиение на области последовательных строк. Параллельное внедрение на уровне отдельных кадров применимо для систем, имеющих в каждый момент времени несколько различных кадров для обработки.

В рамках оценки скорости внедрения осуществлена программная реализация алгоритма E_BLIND/D_LC с использованием покадрового и внутрикадрового распараллеливания на языке C++, доступная по ссылке [5]. Замеры производительности осуществлены на нескольких компьютерах и разных размерах кадров. Результаты в табл. 2 получены для процессора AMD Ryzen 7 5700U, имеющего 8 ядер и 16 независимых потоков и набора картинок размером 4K. В ячейках таблицы содержится среднее время (в миллисекундах) встраивания водяного знака в кадр. Строки таблицы указывают количество потоков покадрового распараллеливания, столбцы – внутрикадрового. Значение на пе-

ресечении строки n и столбца m указывает время, затраченное на внедрение водяного знака в кадр с использованием n потоков покадрового распараллеливания и m потоков внутрикадрового распараллеливания для каждого из этих кадров.

Таблица 2

Скорость внедрения при покадровом и внутрикадровом распараллеливании

		Количество потоков внутрикадрового распараллеливания								
		1	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**
Количество потоков покадрового распараллеливания	**1**	13.432	7.161	7.157	5.364	4.580	3.923	3.492	3.238	3.081
	2	8.202	6.990	4.702	3.376	2.811	3.071	2.970	2.925	2.797
	4	5.617	4.261	2.780	2.793	2.767	2.757	2.725	2.717	2.731
	6	4.237	3.204	2.657	2.690	2.676	2.628	2.606	2.676	2.660
	8	3.776	2.757	2.753	2.738	2.695	2.693	2.665	2.656	2.672
	10	3.210	2.730	2.727	2.709	2.777	2.675	2.675	2.687	2.647
	12	3.011	2.700	2.646	2.573	2.570	2.522	2.570	2.530	2.496
	14	2.743	2.564	2.599	2.623	2.484	2.553	2.546	2.547	2.573
	16	2.569	2.453	2.439	2.404	2.390	2.512	2.439	2.391	2.481

Визуально результаты, полученные в рамках эксперимента, представлены ниже на рисунках, отображающих зависимость скорости внедрения от количества используемых потоков. По осям графиков отложено число потоков внутрикадрового и покадрового распараллеливания и время внедрения в миллисекундах.

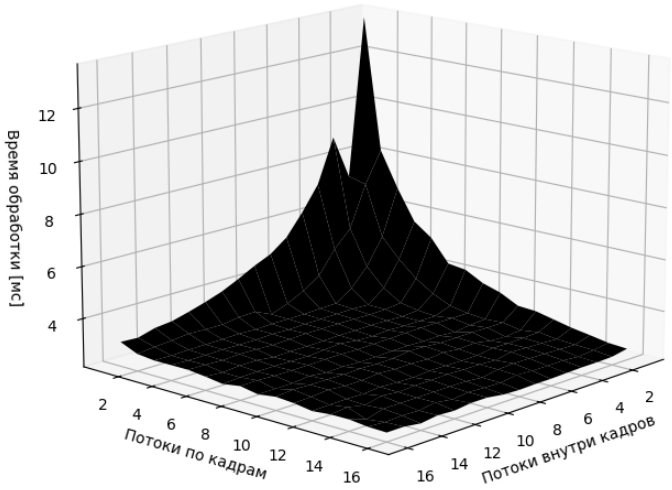


Рис 3. Зависимость скорости внедрения водяных знаков (в миллисекундах) от числа параллельных потоков (картинки группы 4K, AMD Ryzen 7 5700U)

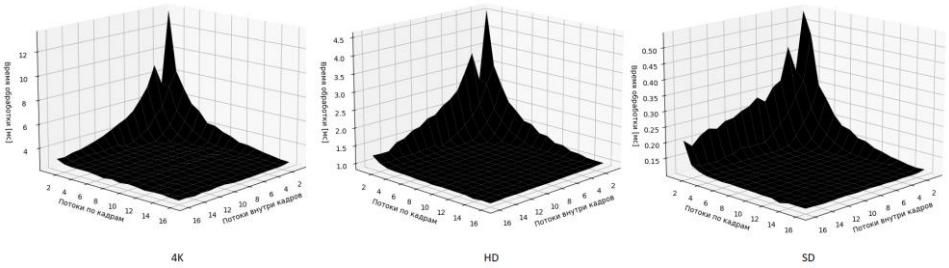


Рис 4. Зависимость скорости внедрения водяных знаков (в миллисекундах) от числа параллельных потоков (AMD Ryzen 7 5700U)

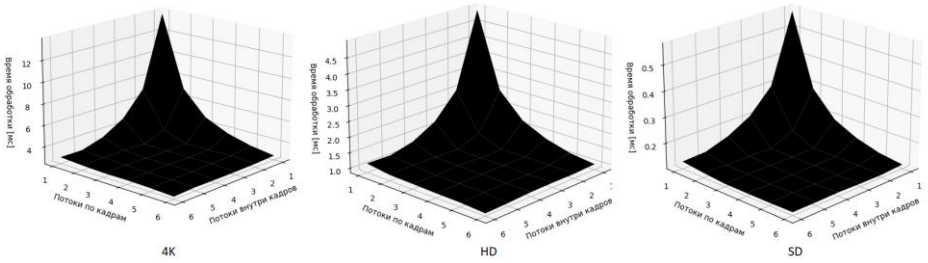


Рис 5. Зависимость скорости внедрения водяных знаков (в миллисекундах) от числа параллельных потоков (Intel Core i7-12700)

На рис. 6 отражена зависимость скорости внедрения водяного знака исходя из типа распараллеливания, размера кадра и количества потоков, полученная в результате эксперимента на компьютере с процессором AMD Ryzen 7 5700U. Горизонтальная ось указывает число использующихся в параллельном режиме потоков. По вертикальной оси отложено время (в миллисекундах) внедрения водяного знака в кадр. На рисунке изображена ситуация, в которой скорость внутрикадрового распараллеливания оказалась выше покадрового при параллельной обработке с использованием 2-х потоков. Подобная ситуация не была получена ни на каких других участвующих в эксперименте компьютерах.

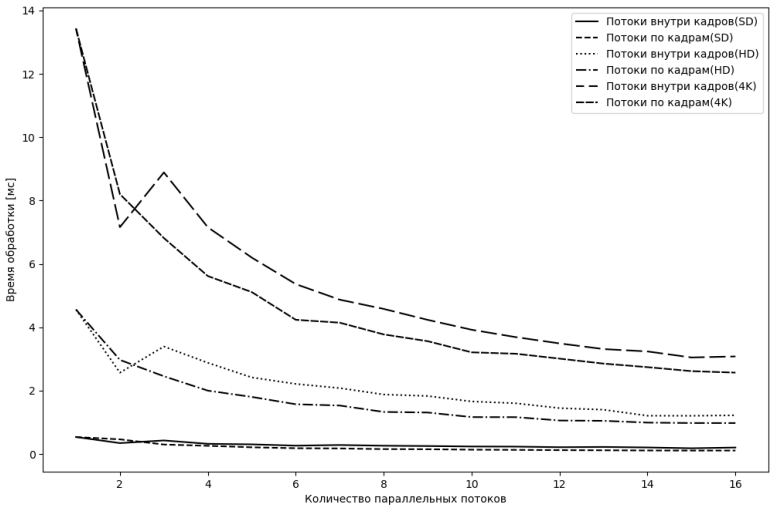


Рис 6. Зависимость скорости внедрения водяных знаков (в миллисекундах) от числа параллельных потоков

Из полученных в ходе эксперимента результатов видно, что скорость внедрения при параллельной обработке по кадрам выше скорости внедрения при параллельной внутрикадровой обработке. Но при малом количестве потоков, для ряда процессоров, скорость внутрикадрового распараллеливания может быть выше покадрового. Таким образом, если проектируемая система имеет возможность покадровой параллельной обработки, то целесообразно использовать её. Если количество возможных потоков покадрового распараллеливания меньше числа независимых потоков процессора, то имеет смысл осуществлять внутрикадровое распараллеливание с целью максимально задействовать вычислительные ресурсы процессора.

Заключение

В рамках эксперимента получены оценки скорости внедрения водяного знака E_BLIND/D_LC с использованием различных подходов к распараллеливанию. Полученные результаты позволяют сформулировать следующие рекомендации по параллельному встраиванию водяных знаков.

При параллельной обработке данных внутри фрейма наиболее эффективным оказывается его разбиение по группам горизонтальных строк пикселей. При параллельной обработке, организованной по отдельным фреймам, скорость внедрения оказывается быстрее по сравнению с параллельной обработкой, организованной по данным внутри каждого из фреймов. Таким образом, если проектируемая система имеет возможность покадровой параллельной обработки, то целесообразно использовать её. Если количество возможных потоков покадрового распараллеливания меньше числа независимых потоков процессора, то имеет смысл осуществлять внутрикадровое распараллеливание с целью максимально задействовать вычислительные ресурсы процессора.

ЛИТЕРАТУРА

1. *Wagner N.R.* Fingerprinting // 1983 IEEE Symposium on Security and Privacy. – IEEE, 1983. – P. 18.

2. *Орешкина Е.И., Фаворская М.Н.* Классификация методов нанесения цифровых водяных знаков // Актуальные проблемы авиации и космонавтики. – 2015. – Т. 1. – №. 11. – С. 414–415.

3. *Гергель В.П.* Теория и практика параллельных вычислений. – 2007.

4. *Cox I. et al.* Digital watermarking and steganography. – Morgan kaufmann, 2007.

5. Watermarks embedding performance evaluation and optimization // GitHub URL: https://github.com/anjin-viktor/watermarking_performance (дата обращения: 05.05.2024).

6. *Asikuzzaman M., Pickering M.R.* An overview of digital video watermarking // IEEE Transactions on Circuits and Systems for Video Technology. – 2017. – V. 28. – №. 9. – P. 2131–2153.

7. *Kowarschik M., Weiß C.* An overview of cache optimization techniques and cache-aware numerical algorithms // Algorithms for memory hierarchies: advanced lectures. – 2003. – P. 213–232.

МИНИМИЗАЦИЯ КОРНЯ ЛОГИЧЕСКОГО УРАВНЕНИЯ КНФ = 1

Бирюков М.О., Андреева В.В.
Томский государственный университет
iilll@duck.com, avv.21@mail.ru

Введение

Задача выполнимости булевых формул (SAT-задача) – фундаментальная задача в области информатики и теории вычислительной сложности. Она является первой, для которой удалось доказать ее принадлежность к классу NP-полных задач в начале 1970-х гг. (при формулировке относительно КНФ). Она имеет широкое применение в ряде областей, включая искусственный интеллект, верификацию программного обеспечения, криптографию, проектирование интегральных схем [1]. Активные исследования в её направлении ведутся по сей день, ежегодно проводится международное соревнование SAT-решателей (программных реализаций алгоритмов решения SAT-задачи) [2].

Большой класс существующих полных алгоритмов решения SAT-задачи основывается на методе ветвей и границ (поиске с возвратом), при котором поиск корня логиче-

ского уравнения КНФ = 1 осуществляется через последовательное "разветвление" пространства поиска путем фиксирования значения переменных, входящих в КНФ, и рекурсивного рассмотрения каждой из этих ветвей [1].

Одним из первых и наиболее известных алгоритмов решения SAT-задачи является DPLL-алгоритм, разработанный в 1961 г., в основе которого лежит метод ветвей и границ вместе с двумя правилами логического вывода: правила единичного дизъюнкта и правила чистой литеры.

Независимо от DPLL, советским и белорусским кибернетиком А.Д. Закревским в 1964 г. был разработан во многом схожий алгоритм решения SAT-задачи [3], который будем называть алгоритмом Закревского.

Одним из наиболее эффективных алгоритмов решения SAT-задачи на сегодняшний день считается алгоритм CDCL, представляющий собой улучшение DPLL.

В некоторых практических задачах возникает необходимость в решении смежной задачи минимизации корня логического уравнения КНФ = 1. Так, например, в [4] предложен метод построения тестовых наборов для неисправности схемы, который сводится к поиску допустимых интервалов некоторой КНФ и формированию проверяющего теста на их основе. При этом, чем меньше ранг найденных интервалов, тем больше возможностей для сокращения длины проверяющего теста, что является предпочтительным.

В данной работе разработан стохастический метод минимизации корня логического уравнения вида КНФ = 1 на основе алгоритма CDCL.

1. Постановка задачи и существующие методы ее решения

Формально задача минимизации корня логического уравнения вида КНФ = 1 представляет собой задачу комбинаторной оптимизации, заключающуюся в поиске для заданной КНФ допустимого интервала (корня) как можно меньшего ранга. Данная задача во многом схожа с задачей выполнимости булевых формул, ввиду чего для нее по большей части применимы результаты из этой области.

Из различий рассматриваемой задачи с задачей выполнимости булевых формул важно отметить, что пространство поиска представляет собой не n-мерное булево пространство B^n, где n – количество переменных рассматриваемой КНФ, а множество всевозможных интервалов в этом булевом пространстве, которое обозначим V_n.

Из существующих методов минимизации корня логического уравнения КНФ = 1 обратим внимание на метод, предложенный в [5] и представляющий собой модификацию алгоритма Закревского. Основная модификация состоит в специальной эвристике ветвления, направленной на поиск корня как можно меньшего ранга, и основная идея которой заключается в том, чтобы в троичной матрице, соответствующей рассматриваемой КНФ, в первую очередь покрывать строки минимального ранга, т.к. они в наибольшей степени влияют на вид корня (допустимого интервала), если он существует.

Эффективность этого метода на случайно сгенерированных КНФ в сравнении с исходным алгоритмом Закревского была показана экспериментально [5], однако данный метод все же обладает несколькими недостатками.

В первую очередь, ввиду природы механизма поиска DPLL-подобных алгоритмов (включая алгоритм Закревского), часть пространства поиска V_n рассматриваемой оптимизационной задачи неизбежно упускается из рассмотрения. Это происходит ввиду того, что формально поиск корня через метод ветвей и границ в DPLL-подобных алгоритмах сводится к обходу "дерева поиска", которое представляет собой направленный ациклический граф, состоящий из $n+1$ уровней (n – количество переменных в КНФ), где k-й уровень содержит 2^k вершин, соответствующих различным интервалам ранга k в B^n. Так, суммарное количество вершин этого дерева (точек пространства поиска), кото-

рые возможно рассмотреть в ходе поиска, меньше, чем мощность пространства поиска V_n рассматриваемой оптимизационной задачи (1):

$$\sum_{k=0}^{n} 2^k < \sum_{k=0}^{n} C_n^k \cdot 2^k \,,\ n \in N \,. \qquad (1)$$

На рис. 1 представлен пример одного из возможных полных деревьев поиска для двумерного булева пространства; на рис. 2 изображен направленный ациклический граф ("граф поиска" для B^2), объединяющий в себя всевозможные деревья поиска для двумерного булева пространства B^2. Серым помечены вершины (точки пространства поиска), которые упускаются из рассмотрения при обходе дерева поиска, как на рис. 1.

Также данный алгоритм, как и все DPLL-подобные алгоритмы, не применяющие периодический перезапуск поиска, подвержен "застреванию" в неудачно выбранных ветвях (областях), являющихся значительной долей огромного пространства поиска, но при этом не содержащих корней [1,6].

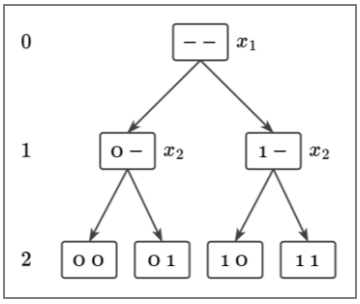


Рис. 1. Одно из возможных деревьев поиска для B^2

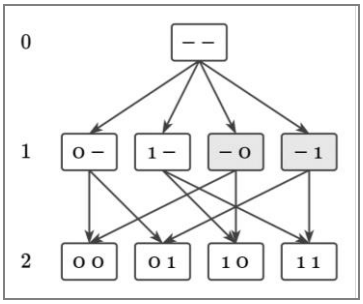


Рис. 2. Граф поиска для B^2

2. Стохастический метод минимизации корня логического уравнения КНФ = 1

2.1. Описание метода

С учетом недостатков предыдущего метода в данной работе поиск корня логического уравнения вида КНФ = 1 как можно меньшего ранга (минимизацию корня) предлагается осуществлять через модифицированный алгоритм CDCL с периодическим перезапуском поиска и со стохастической эвристикой ветвления. Модификация CDCL состоит в нескольких небольших изменениях, адаптирующих этот алгоритм к рассматриваемой оптимизационной задаче:

1. Исходная КНФ проверяется на выполнимость в каждом узле дерева поиска, т.е. для каждого рассматриваемого промежуточного решения (набора присвоений значений переменным, интервала).

2. Поиск продолжается до тех пор, пока не выполнится заданное условие останова. При нахождении корня он добавляется в список найденных корней, и после хронологического отката на предыдущий не до конца рассмотренный узел принятия решения поиск продолжается дальше. При этом узлы дерева поиска, лежащие на уровне больше или равным минимальному рангу из найденных корней, отмечаются как не содержащие решений.

3. Для найденных интервалов (корней) предпринимается попытка их расширения за счет удаления некоторых внешних компонент.

4. Подобно алгоритму Закревского, рассматриваемая в ходе поиска КНФ представляется в виде троичной матрицы. Выбор переменной ветвления и её значения в описанной ниже стохастической эвристике ветвления опирается только на анализ этой троичной матрицы.

5. Используется стохастическая эвристика ветвления, направленная на поиск корня как можно меньшего ранга.

2.2. Стохастическая эвристика ветвления

Основой предлагаемого метода является стохастическая эвристика ветвления, при которой выбор переменной ветвления и её значения определяется реализацией дискретной случайной величины H со множеством значений $X \times B$ (X– множество переменных КНФ, B – булево множество), распределение вероятностей которой формируется на основе анализа троичной матрицы, соответствующей текущей упрощенной КНФ. Такая эвристика позволяет отразить в себе весь спектр "весов" или "качеств" возможных выборов, а не какой-то один выбор, соответствующий максимальному "весу", как при детерминированной эвристике ветвления рассмотренного выше метода, основанного на алгоритме Закревского.

В совокупности же с периодическим перезапуском поиска с начала, появляется возможность рассмотрения нескольких близких по "предпочтительности" деревьев поиска, за счет чего более тщательно исследуется пространство поиска V_n.

Пусть на данном этапе выбора текущая упрощенная КНФ содержит m дизъюнктов, а соответствующая ей троичная матрица обозначена как U. Пусть также исходная КНФ зависит от множества переменных X мощности n.

Итоговое распределение вероятностей случайной величины H строится в несколько этапов. В первую очередь, для каждой пары переменной и её значения $x_k = a$ (выбора) из множества $X \times B$ вычисляется соответствующий ей неотрицательный "вес" с помощью функции оценки $\sigma : X \times B \to [0, +\infty)$, представленной в (2):

$$\sigma(x_k, a) = \sum_{s=1}^{m} I\{u_{s,k} = a\}\,\omega(u_s), \tag{2}$$

где I – индикаторная функция, u_s – s-я строка матрицы U, $u_{s,k}$ – k-я компонента s-й строки матрицы U, $\omega(u_s)$ – "вес" или "значимость" s-ой строки матрицы U. Так, для выбора переменной ветвления и ее значения $x_k = a$ величина $\sigma(x_k, a)$ представляет собой сумму весов обращающихся в истину дизъюнктов (строк) при данном назначении переменной, где вес дизъюнкта определяется рангом соответствующей ему строки: чем меньше ранг, тем больше вес.

В качестве весовой функции ω предлагается взять (3):

$$\omega(c) = 2^{-\mathrm{rang}(c)}. \tag{3}$$

В (3) вес ω(c) дизъюнкта (строки) с равен отношению мощности соответствующего ему интервала из множества нулевых значений исходной КНФ к мощности всего булева пространства B^n.

Для всех $x \in X$ из пар значений $\sigma(x,0)$ и $\sigma(x,1)$ предлагается рассматривать только максимальное из них – большее из этих значений берется неизменным, а оставшееся принимается равным нулю. В результате такого "прореживания" в ноль обращаются n из исходных $2n$ значений весов. Наконец, из результирующего набора весов $\{\sigma(x,a) : (x,a) \in X \times B\}$ формируется итоговое распределение вероятностей через "нормирующее" преобразование f с параметром $\gamma > 0$ (4):

$$f_\gamma(\sigma(x,a)) = \frac{\sigma(x,a)^\gamma}{\sum\limits_{(x,a) \in X \times B} \sigma(x,a)^\gamma}. \tag{4}$$

Важным свойством преобразования (4) является то, что $f_\gamma(0) = 0$ при $\gamma > 0$, т.е., выбор переменной ветвления и ее значения, которому при функции оценки вида (2) соответствует нулевой вес, будет иметь нулевую вероятность быть "выбранным" данной стохастической эвристикой ветвления. Это необходимо для того, чтобы эвристика выбирала только те переменные для ветвления, которые не входят в текущее промежуточное решение, и которым в упрощенной КНФ (троичной матрице) соответствуют

столбцы, содержащие хотя бы одну определенную компоненту. Так, вектор (набор) итогового распределения вероятностей длины $2n$ содержит как минимум $n+r$ нулевых компонент и выбор сводится к $n-r$ вариантам, где r – число назначений переменным в текущем промежуточном решении.

Параметр γ преобразования (4) позволяет варьировать "контраст" результирующего распределения вероятностей:

– при $\gamma \in (0,1)$ результирующее распределение вероятностей "приближается" к равновероятному, ненулевые компоненты которого равны $(n-r)^{-1}$;

– при $\gamma = 1$ преобразование (4) вырождается в нормирование вектора весов по норме l_1, при котором компоненты результирующего вектора распределения вероятностей пропорциональны соответствующим им весам;

– при $\gamma > 1$, подобно многопеременной логистической функции (softmax), в большей степени выражается максимальная компонента распределения.

На рис. 3, 4 приведены примеры распределений вероятностей, полученных через описанный выше алгоритм при разных значениях параметра γ для КНФ (троичной матрицы) из табл. 6 работы [5] (нулевые компоненты не изображены).

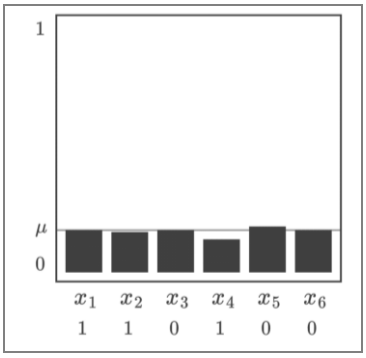


Рис. 3. Распределение вероятностей при γ = 1/4

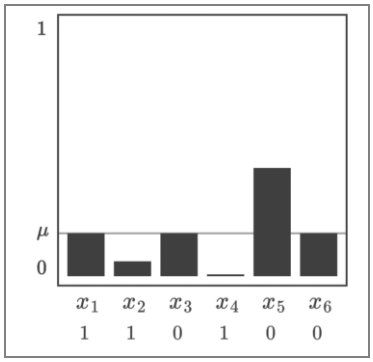


Рис. 4. Распределение вероятностей при γ = 4

3. Экспериментальные результаты

Разработанный алгоритм был реализован на языке C++ и испытан на различных бенчмарках (КНФ) [7], включающих в себя равномерно случайные КНФ (uf*, uuf*), состоящие из трех равновероятно взятых литер, а также "индустриальные" бенчмарки, соответствующие реальным задачам: задаче индуктивного вывода (ii*) и задаче проверки наличия неисправности в логической схеме (ssa*) (табл. 1–3).

Таблица 1

Результаты работы разработанного метода на индустриальных бенчмарках

b	*n*	*m*	*r*	*c*	*s*	*rs*
ii8a2	180	800	143	9842	514 445	157
ii8a3	264	1552	191	12 352	165 018	62
ii8a4	396	2798	283	14 794	30 450	22
ii8b1	336	2068	191	815	101 844	44
ii8b4	1068	8214	658	2289	1155	2
ssa.7552.038	1501	3575	1448	1514	80 927	31
ssa.7552.158	1363	3034	1327	1718	117 013	48
ssa.7552.159	1363	3032	1327	1988	124 188	62
ssa.7552.160	1391	3126	1359	3776	86 833	45

Примечания:
n – количество переменных КНФ;
m – количество дизъюнктов КНФ;
r – минимальный ранг из найденных в течение 3-х минут корней;
c – количество встреченных конфликтов;

s – количество "пропусков" областей ниже минимального ранга;
rs – количество раз был произведен перезапуск поиска с начала;
b – название бенчмарка (КНФ).

Таблица 2

Результаты работы предыдущего метода на индустриальных бенчмарках

b	***n***	***m***	***r***	***c***	***i***	***t***
ii8a2	180	800	159	1 443 750	12 588 158	723 854
ii8a3	264	1552	–	568 375	9 516 214	900 002
ii8a4	396	2798	349	20	681	328
ii8b1	336	2068	191	7448	205 704	26 772
ii8b4	1068	8214	–	30 071	1 729 557	900 010
ssa.7552.038	1501	3575	–	44 733	1 471 992	900 008
ssa.7552.158	1363	3034	–	57 515	1 853 963	900 002
ssa.7552.159	1363	3032	1327	38	2753	1781
ssa.7552.160	1391	3126	–	64 681	2 339 482	900 002

Примечания:
n – количество переменных КНФ;
m – количество дизъюнктов КНФ;
r – ранг найденного корня (прочерк, если не найден за 15 минут);
c – количество встреченных конфликтов;
i – количество имплицированных присвоений;
t – время работы алгоритма в миллисекундах;
b – название бенчмарка (КНФ).

Таблица 3

Результаты работы двух методов на равномерно случайных и невыполнимых КНФ

b	***n***	***m***	***r***	***r****	***t***	***t****
uf.125.538 (100)	125	538	116,74	116,09	1 123,93	3 494,07
uf.150.645 (64)	150	645	140,53	139,61	4 079,96	11 121,98
uuf.150.645.1	150	645	–	–	900 008 †	547
uuf.200.860.1	200	860	–	–	900 002 †	1 136
ssa.0432.003	435	1027	–	–	5 996	189
ssa.2670.141	986	2315	–	–	900 002 †	3498

Примечания:
n – количество переменных КНФ;
m – количество дизъюнктов КНФ;
r – средний ранг первого найденного корня предыдущим методом;
*r** – средний ранг первого найденного корня новым методом;
t – среднее время работы предыдущего метода в миллисекундах;
*t** – среднее время работы нового метода в миллисекундах;
b – название бенчмарка (КНФ).
Символ "†" означает, что алгоритм не пришел к выводу о невыполнимости КНФ по истечении 15-ти минут.

В приведенных в табл. 2 результатах работы предыдущего метода наблюдается эффект "застревания" DPLL-подобных алгоритмов, не применяющих периодический перезапуск поиска, в неудачно выбранных "плохих" областях пространства поиска: для некоторых КНФ удается найти корень за относительно короткое время, а на других алгоритм уходит в длительный и безрезультатный поиск.

На выполнимых равномерно случайных 3-КНФ оба алгоритма показывают близкие к равным результаты.

В табл. 3 также представлено время работы двух алгоритмов на невыполнимых КНФ, из которого можно видеть, что разработанный метод, во-первых, остается полным алгоритмом решения SAT-задачи, и, во-вторых, благодаря анализу конфликтов позволяет прийти к выводу о невыполнимости формулы быстрее, чем предыдущий метод.

Наконец, обращая внимание на результаты работы двух методов на некоторых индустриальных бенчмарках в табл. 1 и 2, можно видеть, что, по крайней мере, на рассмотренных КНФ разработанный в данной работе стохастический метод минимизации корня логического уравнения вида КНФ = 1 позволяет находить корни меньшего ранга

и за меньшее время, чем рассмотренный существующий метод, основанный на алгоритме Закревского.

Заключение

В данной работе была рассмотрена задача минимизации корня логического уравнения вида КНФ = 1, а также смежная с ней задача выполнимости булевых формул в формулировке КНФ. Также был рассмотрен один из существующих методов минимизации корня уравнения КНФ = 1, основанный на алгоритме Закревского. Были выявлены несколько его недостатков.

На основе этого анализа был разработан стохастический метод минимизации корня логического уравнения вида КНФ = 1, представляющий собой модифицированный алгоритм CDCL со стохастической эвристикой ветвления.

Разработанный метод был реализован на языке C++, его эффективность в сравнении с предыдущим методом показана экспериментально.

ЛИТЕРАТУРА

1. *Biere A., Heule M., Van Maaren H., Walsh T.* Handbook of Satisfiability // Frontiers in Artificial Intelligence and Applications. – IOS Press, 2021. – P. 336. – ISBN 978-1-64368-160-3.
2. The International SAT Competition Web Page // URL: https://satcompetition.github.io; (дата обращения 25.05.2024).
3. *Закревский А.Д.* Логические уравнения. – М.: Едиториал УРСС, 2003. – 96 с.
4. *Тычинский В.З., Андреева В.В.* Получение тестовых пар для робастно тестируемых неисправностей задержек путей с использованием SAT- решателей // Математическое и программное обеспечение информационных, технических и экономических систем : материалы Международной научной конференции, Томск, 28–30 мая 2020 г. – Томск: Изд-во Том. ун-та, 2020. – С. 194–200.
5. *Андреева В.В., Тарновская Т.П.* Сокращение ранга конъюнкции, представляющей корень логического уравнения // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. – 2015. – №4 (33). – С. 62–68.
6. *Gomes C.P., Selman B., Kautz H.* Boosting combinatorial search through randomization // AAAI/IAAI 98.1998 (1998): P. 431–437.
7. SATLIB – Benchmark problems // URL: https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html; (дата обращения 25.05.2024).

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ КАТАЛОГИЗАЦИИ АУДИОКНИГ

Гайсин Р.Е., Пахомова Е.Г.
Томский государственный университет
gaisin.roman@mail.ru, PakhomovaEG@yandex.ru

Введение

В последние годы аудиокниги стали весьма популярными благодаря своей практичности и доступности [1]. Они позволяют наслаждаться литературой в ситуациях, когда чтение печатных или электронных книг невозможно, например, при вождении автомобиля, занятиях спортом или выполнении домашних дел. С ростом количества аудиокниг появляется потребность в разработке эффективных средств для их каталогизации и управления.

Целью данной работы является разработка эффективного и удобного инструмента, который позволит пользователям легко управлять своей коллекцией аудиокниг. Приложение будет обеспечивать удобную организацию, хранение, поиск, фильтрацию и легкий доступ к метаданным аудиокниг. Помимо этого, оно должно обладать удобным и интуитивно понятным интерфейсом.

1. Проектирование приложения

Архитектура приложения. Для приложения была выбрана архитектура MVC. Причиной стала возможность четкого разделения ответственности между данными,

бизнес-логикой и пользовательским интерфейсом [2,3]. Это значительно упрощает разработку, тестирование и дальнейшее сопровождение приложения.

Логическая модель данных. Структура информации на уровне абстракции, независимом от конкретной СУБД, описывается в логической модели данных. В ней в процессе разработки каталогизатора были выделены две сущности: "Файл аудиокниги" и "Аудиокнига" (табл. 1, 2).

Таблица 1

Сущность "Файл аудиокниги"

Атрибут	Тип	Описание
file_id	INTEGER	Уникальный идентификатор файла
is_listened	BOOLEAN	Статус прослушивания
file_path	TEXT	Путь к файлу
book_id	INTEGER	Идентификатор связанной аудиокниги

Таблица 2

Сущность "Аудиокнига"

Атрибут	Тип	Описание
book_id	INTEGER	Уникальный идентификатор книги
title	TEXT	Название аудиокниги
author	TEXT	Автор
genre	TEXT	Жанр
year	INTEGER	Год выпуска
narrator	TEXT	Чтец
date_added	TEXT	Дата добавления
description	TEXT	Описание
is_completed	BOOLEAN	Статус завершения
is_favorite	BOOLEAN	Статус избранного
bitrate	INTEGER	Битрейт аудио
duration	INTEGER	Длительность (секунды)
size	INTEGER	Размер
path	TEXT	Путь к аудиокниге

Между "Аудиокнига" и "Файл аудиокниги" будет существовать связь "один ко многим" (одна аудиокнига может иметь несколько файлов). Эта связь устанавливается через атрибут book_id в таблице "Файл аудиокниги", который, в свою очередь, ссылается на book_id в таблице "Аудиокнига".

Диаграмма компонентов приложения. Проводя анализ всевозможных пользовательских сценариев с учетом специфики разрабатываемого приложения и предполагаемого пользовательского интерфейса, можно выделить следующую диаграмму компонентов приложения (рис. 1).

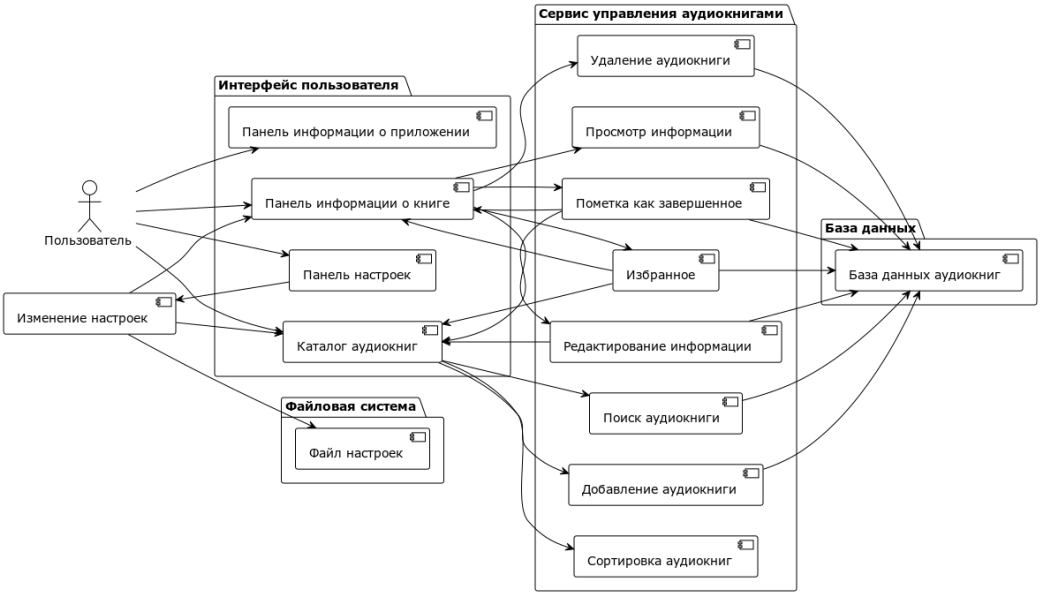


Рис. 1. Диаграмма компонентов приложения

2. Реализация приложения

Выбор технологий и инструментов. Для разработки приложения был выбран язык программирования Python версии 3.12.3 [4]. Python отлично подходит для написания кроссплатформенных приложений. Каталогизатор сможет работать на различных операционных системах (Windows, macOS и Linux). В качестве базы данных была выбрана SQLite. Python имеет библиотеки (к примеру, sqlite3), которые обеспечивают интерфейс для работы с SQLite [5]. Для работы с аудиофайлами было решено использовать готовую библиотеку Mutagen [6]. Для поиска информации в интернете, а также парсинга страниц используются библиотеки requests и BeautifulSoup (bs4). В качестве фреймворка для создания GUI было решено использовать PyQt [7].

Реализация представлений и стилей. Главное окно приложения является центральной частью, где размещаются все основные элементы интерфейса. В главном окне происходит инициализация остальных виджетов и их интеграция в общую структуру приложения.

В конструкторе класса "AudioBookCataloguer" задается минимальный и начальный размер окна, а затем вызываются методы для настройки интерфейса, моделей данных, контроллеров и их взаимодействий. Рассмотрим их в порядке вызовов. В методе setup_ui происходит настройка основного интерфейса, устанавливается заголовок окна и создается основной горизонтальный макет QHBoxLayout, в который добавляются все основные компоненты интерфейса.

Для создания и добавления панели каталога вызывается метод setup_catalog_panel, где создается экземпляр CatalogPanel и добавляется в основной макет.

В методе setup_tabs создается виджет QTabWidget, в который добавляются три вкладки: BookInfoTab, SettingsTab и AboutTab – информация о книге, панель настроек и информация о приложении соответственно. Они настроены так, чтобы работать в режиме документов и расширяться вместе с расширением окна. Далее в методе setup_models инициализируются модели данных, такие как AudioBookInfoModel, SettingsModel и CatalogPanelModel.

Метод setup_controllers отвечает за создание контроллеров, которые связывают представления с моделями данных, обрабатывают события и обновляют интерфейс в ответ на изменения данных. Например, AudiobookInfoController связывает вкладку с

информацией о книге с моделью данных аудиокниг, а SettingsController управляет взаимодействием между вкладкой настроек и моделью настроек.

Метод setup_connections настраивает базовые сигналы и слоты для взаимодействия между различными компонентами; так возможна передача информации из одного контроллера в другой.

В конце для загрузки настроек приложения и обновления интерфейса вызывается метод load_app_settings, который обращается к контроллеру настроек и запрашивает информацию о том, с какими настройками запустить приложение. Дополнительно имеется функция для применения стиля из CSS-файла – apply_stylesheet, которая считывает содержимое файла стилей и применяет его ко всему интерфейсу приложения, в том числе и к дочерним интерфейсам.

Основной класс CatalogPanel наследуется от QWidget и представляет собой отдельную панель для управления списком аудиокниг.

В конструкторе класса вызывается метод setup_ui, который настраивает интерфейс панели. Создается вертикальный макет QVBoxLayout, в который добавляются все основные компоненты панели: кнопка добавления аудиокниги, элементы управления сортировкой, панель поиска и таблица аудиокниг.

Для настройки кнопки добавления аудиокниги используется метод setup_searching_button. В нем создается кнопка QPushButton с надписью "Добавить аудиокнигу", ей присваивается объектное имя для стилизации.

Метод setup_sorting_controls отвечает за настройку элементов управления сортировкой. Создаются метка QLabel с текстом "Сортировать по:", выпадающий список QComboBox для выбора параметра сортировки и кнопка QPushButton для изменения направления сортировки. Все эти элементы объединены макетом QHBoxLayout.

Для создания панели поиска используется метод setup_search_panel. В нем создается текстовое поле QLineEdit с подсказкой "Введите текст для поиска...". Также создается таймер QTimer, который будет использоваться для реализации динамического поиска.

Метод setup_audiobook_table создает и настраивает таблицу для отображения аудиокниг. Вертикальные заголовки созданной таблицы скрываются, а горизонтальные заголовки, в свою очередь, настраиваются так, чтобы их размеры подстраивались под содержимое, показывая как можно больше информации. Редактирование ячеек в таблице отключается, чтобы не было возможности изменять информацию через таблицу, задается политика размера, чтобы она могла расширяться и занимать все доступное пространство в макете (рис. 2).

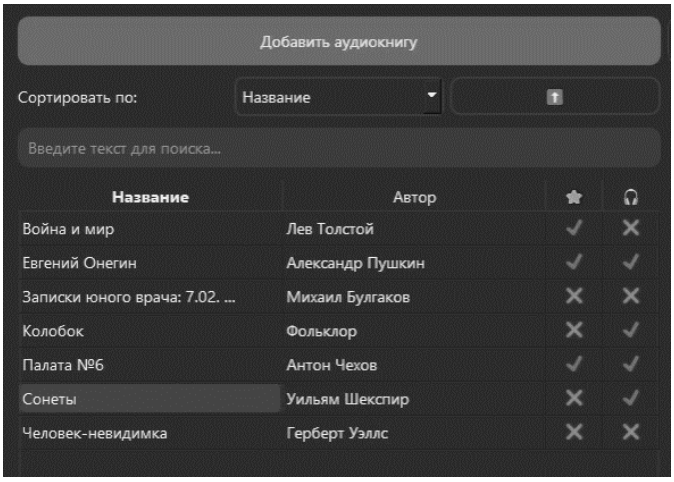


Рис. 2. Панель каталога аудиокниг

Информация об аудиокниге (audiobook_info_view) является одним из ключевых элементов приложения. Класс BookInfoTab является наследником QWidget и представляет собой отдельную вкладку с информацией о книге и элементами управления. Конструктор класса вызывает метод setup_ui, который настраивает интерфейс вкладки.

Создается вертикальный макет QVBoxLayout, который будет основным макетом вкладки. Затем создается вертикальный разделитель QSplitter, который будет использоваться для распределения пространства между двумя основными областями: областью прокрутки для информации о книге и областью прокрутки для кнопок и таблицы файлов.

Область прокрутки для информации о книге создается с использованием виджета QScrollArea. Внутри этого виджета создается еще один виджет QWidget, который будет содержать все элементы информации о книге, и устанавливается как содержимое области прокрутки. Для компоновки элементов информации о книге используется вертикальный макет QVBoxLayout. Только после этого вызывается метод setup_book_info_tab, который настраивает отображение обложки книги и меток информации. Обложка книги отображается с использованием QGraphicsScene и QGraphicsView, которые размещаются в горизонтальном макете QHBoxLayout, центрируя обложку с помощью растягивающих элементов. Метки информации о книге размещаются в сеточном макете QGridLayout.

Для области прокрутки, содержащей кнопки и таблицу файлов, создается второй виджет QScrollArea с такой же структурой. Внутри этого виджета создается макет QVBoxLayout, и вызываются методы setup_action_buttons и setup_file_table. Первый из них создает кнопки для различных действий (например, "Найти информацию в интернете", "Удалить", "Добавить в избранное" и т.д.), добавляет их в макет. Кнопки отключены до тех пор, пока они не понадобятся. Метод setup_file_table создает таблицу с двумя столбцами для отображения файлов и информации о их проигрывании. Таблица скрыта до тех пор, пока не потребуется ее отображение.

Обе созданные ранее области прокрутки добавляются в разделитель QSplitter, который затем добавляется в основной макет вкладки. Для разделителя устанавливается пропорциональное распределение пространства: 70% для информации о книге и 30% для кнопок и таблицы файлов.

При таком подходе пользователю всегда будет видна вся необходимая информация, кнопки для управления книгой не будут теряться за большим текстом, а таблица с файлами аудиокниги будет легко доступна к просмотру (рис. 3).

Рис. 3. Окно с информацией об аудиокниге

Класс настроек SettingsTab наследуется от QWidget и предоставляет пользователю интерфейс для настройки различных параметров приложения. В конструкторе класса аналогично вызывается метод setup_ui, который настраивает интерфейс вкладки. Основной макет вкладки – вертикальный QVBoxLayout, в который добавляются все элементы интерфейса. Первым создается виджет области прокрутки QScrollArea, который содержит все группы настроек. Внутри него создается виджет QWidget, который задается как содержимое области прокрутки, и вертикальный макет QVBoxLayout для размещения всех групп настроек.

Для настроек сортировки создается группа QGroupBox с вертикальным макетом. Внутри этой группы вызывается метод add_sort_checkboxes, который добавляет флажки для каждого параметра сортировки из SORT_OPTIONS (константа, содержащая всевозможные опции сортировки). Эти флажки позволяют пользователю выбрать, по каким критериям сортировать данные (рис. 4).

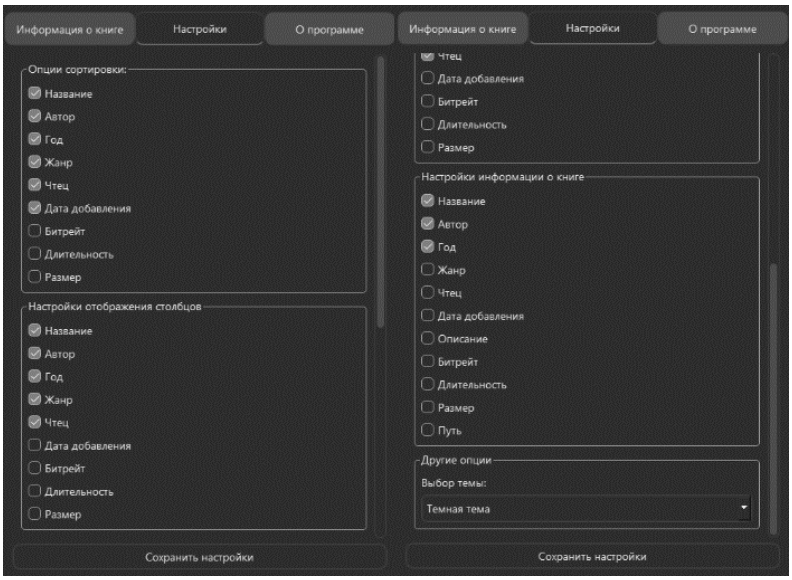


Рис. 4. Опции сортировки

Далее создается группа для настроек отображения столбцов. Она также реализована с помощью QGroupBox и содержит вертикальный макет. Метод add_table_display_checkboxes добавляет флажки для каждого параметра отображения столбцов из COLUMN_OPTIONS (константа, содержащая всевозможные настройки столбцов), позволяя пользователю выбрать, какие столбцы будут видны в таблице.

Следующая группа настроек относится к отображаемой информации о книге. Она организована аналогично предыдущим группам с помощью метода add_book_info_checkboxes, который добавляет флажки для каждого параметра отображаемой информации из BOOK_INFO_OPTIONS (константа, содержащая всевозможные настройки отображения информации).

Также добавляется группа для дополнительных опций, таких как выбор темы оформления. В этой группе создается метка и выпадающий список QComboBox, который позволяет пользователю выбрать между темной и светлой темами.

Все эти группы добавляются в макет области прокрутки. Затем область прокрутки добавляется в основной макет вкладки.

В конце создается кнопка "Сохранить настройки", которая добавляется в основной макет вкладки. Эта кнопка предназначена для сохранения всех изменений настроек, сделанных пользователем.

Класс EditBookDialog наследуется от QDialog и позволяет пользователю изменять информацию о книге и сохранять изменения. Это окно в дальнейшей работе будет появляться в двух случаях: при непосредственном изменении информации об аудиокниге, либо для удобной замены текущей информации на ту, которая нашлась в Интернете.

Конструктор этого класса принимает два аргумента: parent и book_info, представляющий родительский виджет и содержащий предоставляемую виджету информацию для редактирования. Внутри конструктора вызывается метод init_ui, который настраивает интерфейс диалогового окна.

Этот метод устанавливает заголовок и размер окна, создает основной вертикальный макет и добавляет в него все необходимые элементы. В начале вызывается метод create_input_fields, который создает поля ввода для редактирования информации о книге. Он перебирает все элементы из book_info, полученного ранее, и для каждого элемента добавляет метку и текстовое поле в горизонтальный макет. Затем этот макет до-

бавляется в основной вертикальный макет, а текстовое поле сохраняется в словаре inputs для дальнейшей работы.

Далее вызывается метод create_action_buttons, который создает кнопки "Сохранить" и "Отмена". Кнопка "Сохранить" привязывается к методу save, который собирает данные из всех полей ввода и обновляет book_info новыми значениями. Если обновление проходит успешно, диалоговое окно закрывается, в противном случае показывается окно сообщения об ошибке.

В модуле выбора аудиокниги класс SelectBookDialog наследуется от QDialog и позволяет пользователю выбрать книгу из списка. Данное окно появляется при успешном поиске аудиокниг в Интернете.

Аналогично предыдущему примеру, конструктор данного класса принимает два параметра: parent и books. Первый параметр имеет такое же предназначение, второй же содержит список всех найденных в Интернете книг.

Метод init_ui устанавливает заголовок и размер окна, создает основной вертикальный макет и добавляет в него список книг и кнопки действий. Сначала создается виджет списка QListWidget, который наполняется элементами на основе списка книг. Каждый элемент списка содержит информацию об авторе, названии и чтеце книги. Затем виджет списка добавляется в основной макет.

Далее создается горизонтальный макет для кнопок "Выбрать" и "Отмена". Также происходит привязка кнопок. Кнопка "Выбрать" привязывается к методу select, который проверяет, был ли выбран элемент из списка. Если элемент выбран, информация о книге сохраняется, и диалоговое окно закрывается. Если элемент не выбран, появляется предупреждающее сообщение. Кнопка "Отмена" привязывается к встроенному методу reject, который закрывает диалоговое окно без сохранения изменений. Оба виджета кнопок добавляются в горизонтальный макет, который затем добавляется в основной вертикальный макет.

Вкладка "О программе" содержит в себе несколько элементов с информацией о программе, версии, разработчике и контактах для связи.

Стилизация интерфейса в приложении происходит через использование файлов CSS, в которых, подобно веб-разработке, прописаны разнообразные свойства отдельных элементов, задающих внешний вид виджетов. Реализовано два файла стилей: dark_theme.css и light_theme.css, что соответствует темной и светлой теме приложения.

Реализация моделей. Модель AudioBookInfoModel отвечает за управление данными, связанными с аудиокнигами, в приложении. Она устанавливает связь с базой данных и предоставляет методы для получения, обновления и удаления информации о книгах. При создании экземпляра AudioBookInfoModel инициализируется соединение с базой данных, используется класс Database. Модель взаимодействует с базой данных, производит чтение, запись и удаление данных.

Метод get_book_info_by_id позволяет получить информацию о книге по ее идентификатору. Метод принимает идентификатор книги и список полей, информацию по которым необходимо получить. Сначала формируется список столбцов на основе переданных опций, затем выполняется запрос к базе данных. Если книга найдена, возвращается словарь, где ключи являются опциями, а значения – соответствующими данными из БД.

Для получения списка файлов аудиокниги используется метод get_audiobook_files, который выполняет соответствующий запрос к базе данных. Метод find_audiobook_cover предназначен для извлечения обложки аудиокниги из файла. Он получает путь к первому файлу аудиокниги и использует функцию extract_cover_from_file для получения изображения обложки.

Модель предоставляет методы для управления статусами книги, такими как добавление в избранное (add_audiobook_to_favorite), удаление из избранного (remove_audiobook_from_favorite) и проверка, является ли книга избранной (is_favorite).

Аналогично реализованы методы для отметки книги как завершенной (mark_as_completed) или же незавершенной (mark_as_incompleted), а также для проверки статуса завершения (is_completed).

Метод update_book_info позволяет обновлять информацию о книге. Он принимает идентификатор книги и словарь с новыми данными, преобразует ключи словаря в соответствующие поля базы данных и отправляет запрос. Метод update_listened_status используется для обновления статуса прослушивания файла. Он принимает состояние и путь к файлу. Метод get_all_book_info_options, в свою очередь, возвращает перечень всех доступных опций для получения информации о книге. Модель CatalogPanelModel осуществляет управление данными, которые будут отображаться во всей левой части приложения, а именно – в панели каталога. При инициализации, как и в предыдущем примере, устанавливается соединение с базой данных. Рассмотрим наиболее важные методы класса "CatalogPanelModel".

Специально для импорта аудиокниг был реализован метод import_audiobook. Этот метод принимает путь как к файлу, так и к директории. Отметим, что если путь к директории не указан, то по умолчанию он считается отсутствующим. Если директория была указана, то реализованный метод обходит все файлы в ней и извлекает определенные метаданные из каждого файла (битрейт, продолжительность и размер). Затем эти значения форматируются и сохраняются в базу данных. В случае, когда указывается путь к отдельному файлу, метаданные извлекаются непосредственно из этого файла и также добавляются в базу данных. Метод import_file позволяет отправлять запросы к базе данных для добавления отдельного файла аудиокниги. Этот метод вызывается контроллером при добавлении новых аудиокниг. Получение списка аудиокниг реализовано в методе get_audiobooks_list. Происходит формирование запроса к базе данных на основе параметров сортировки, текста из поисковой строки и опций отображения столбцов. При этом статусы избранного и завершенного установлены по умолчанию.

Для считывания файлов из директории используется метод get_audio_files. Он получает список аудиофайлов и отправляет их контроллеру. Метод get_book_id возвращает идентификатор книги в базе данных по пути к файлу или директории. Проверка существования аудиокниги в базе данных осуществляется с помощью метода audiobook_exists. Его реализация необходима для избегания дублирования записей и поддержки целостности информации в базе данных.

Модель SettingsModel отвечает за управление настройками приложения, обеспечивая их сохранение и загрузку из файла.

При создании экземпляра класса SettingsModel происходит инициализация с указанием имени файла настроек. По умолчанию используется файл settings.json. В конструкторе вызывается метод load_settings, происходит открытие и считывание файла настроек. Если он не найден, выводится сообщение об ошибке и возвращаются значения по умолчанию.

Метод save_settings отвечает за сохранение настроек в файл. Он принимает параметры сортировки, отображения, информации о книге и тему оформления, формирует словарь с этими параметрами и записывает его в файл. Для записи используется текстовый формат JSON.

Дополнительно модель предоставляет методы для получения включенных опций сортировки, отображения и информации о книге.

Для получения всех возможных опций сортировки, отображения и информации о книге в панели настроек, независимо от того, включены они или нет, используются методы get_sorting_options, get_display_options и get_book_info_options.

Метод get_theme возвращает текущую тему оформления, указанную в настройках. Если она не была задана, метод устанавливает тему по умолчанию.

Реализация контроллеров. Контроллер AudiobookInfoController обеспечивает взаимодействие между представлением и моделью, управляя потоком данных и обра-

боткой событий, связанных с изменениями данных книги. Инициализация контроллера осуществляется с передачей представления и модели, а также вызовом метода setup_connections для установки сигналов и слотов. Контроллер хранит текущее состояние, включая выбранную строку в таблице, идентификатор книги, статус избранного и завершенного, а также текущие опции информации о книге для отображения. Изначально создаются всевозможные метки для отображения информации об аудиокниге в методе init_info_labels, при этом они скрыты и не отображаются до определенного момента.

Когда происходит обновление опций отображаемой информации, метод update_book_info_options обновляет текущие опции в контроллере и вызывает update_info_labels, чтобы отображать только те метки, которые были выбраны пользователем. При выборе файла в таблице эмитируется сигнал, срабатывает метод update_selected_book_id, происходит сохранение идентификатора книги и обновление отображаемой информации.

Метод display_audiobook_info отвечает за отображение подробной информации о книге. Происходит проверка существования книги: при успехе информация отображается с помощью вызова метода update_current_book_info, в противном случае вся информация очищается, а кнопки управления книгой деактивируются. Метод update_current_book_info, в свою очередь, занимается отображением и скрытием лейблов с учетом текущих настроек приложения и управляет видимостью таблицы файлов.

Таблицей файлов занимается метод update_file_table. Он загружает список файлов из модели и отображает их в таблице. Каждый файл сопровождается флажком для отметки прослушанного статуса. Метод toggle_listened обновляет статус файла при изменении состояния флажка.

Для того, чтобы отобразить на представлении обложку, используется метод update_cover, загружающий изображение из модели. Если возникают ошибки, обложка не отображается и выводится сообщение об ошибке.

Удаление аудиокниги осуществляется через метод delete_audiobook, Записи из базы данных удаляются, интерфейс обновляется, кнопки деактивируются до выбора новой книги.

Контроллер также управляет статусом завершенности книги через метод update_completed_button, который обновляет текст, и обработчик событий для кнопки. Методы mark_as_complete и mark_as_incomplete изменяют статус книги в модели и обновляют интерфейс.

Для управления избранными книгами аналогично используются методы update_favorite_button, add_to_favorites, remove_from_favorites. Редактирование информации о книге осуществляется через метод edit_book, который открывает диалоговое окно для редактирования. После сохранения изменений информация обновляется в модели и интерфейсе. Открытие папки с аудиокнигой реализовано в методе open_audiobook_folder, который открывает проводник файлов на соответствующем пути.

Поиск информации об аудиокниге осуществляется методом find_audiobook_info. Он обращается к модели для поиска информации в интернете и открытии диалогового окна для выбора книги. Метод setup_connections связывает кнопки в интерфейсе с соответствующими методами контроллера.

Файл catalog_panel_controller.py включает в себя класс MultiDirectoryDialog. Класс расширяет функциональность стандартного диалогового окна (QFileDialog); другими словами – переопределяет обычное диалоговое окно и позволяет выбирать сразу несколько директорий.

Основной класс CatalogPanelContoller при инициализации связывает представление и модель панели каталога. Он хранит в себе текущую опцию сортировки и опции отображения столбцов в таблице.

Когда пользователь выбирает аудиокнигу в таблице, метод new_item_clicked сначала обрабатывает это событие, а затем обновляет текущий выбранный элемент и отправляет сигнал с идентификатором книги. С учетом настроек метод update_sort_panel обновляет параметры сортировки.

Метод do_search выполняет поиск по тексту, введенному в панель поиска. В результате в соответствии с результатами поиска обновляет таблицу аудиокниг. Метод toggle_sort_direction переключает направление сортировки и изменяет соответствующую метку на кнопке. Метод update_audiobook_table обновляет таблицу аудиокниг, загружает данные о них из модели и отображает в таблице. Он настраивает количество столбцов, их заголовки и размеры, а также заполняет таблицу данными из БД. Для каждого аудиофайла в таблице добавляются эмодзи, которые указывают на статус избранного и завершенного.

Процесс добавления новых аудиокниг реализован через метод add_audiobook. Он отображает диалоговое окно для выбора типа добавляемого файла или директории. В зависимости от выбора, метод add_files или add_directories обрабатывает добавление аудиофайлов (или директорий) в базу данных. Метод add_files вызывается для добавления файлов аудиокниг в таблицу audiobooks_files.

Контроллер SettingsController управляет настройками приложения. Он обеспечивает их загрузку, сохранение и применение. При инициализации контроллера метод setup_ui_state загружает настройки из файла, а затем устанавливает их в пользовательском интерфейсе.

Метод load_initial_settings загружает текущие настройки из модели и устанавливает состояния флажков в интерфейсе.

Метод emit_settings_to_view отправляет сигналы с текущими включенными опциями; это необходимо, чтобы обновить виджеты и другие компоненты интерфейса.

Темы оформления загружаются и применяются методом load_theme. Он устанавливает текущую тему из списка, а следом вызывает метод change_theme для применения соответствующего файла стилей.

Сохранение настроек осуществляется через метод save_settings. Метод собирает текущие значения флажков и выпадающего списка темы, и сохраняет их в модели. Перед сохранением выполняется проверка: хотя бы одна опция в каждом разделе должна быть включена. Если ни одна опция не выбрана, выводится предупреждающее сообщение и принудительно включается первая опция.

Метод change_theme изменяет тему оформления. В зависимости от выбранной темы отправляется сигнал с именем соответствующего CSS-файла, который затем применяется к приложению.

Заключение

В данной работе была представлена разработка и реализация приложения "Каталогизатор аудиокниг" для управления коллекциями аудиофайлов.

Разработанное приложение имеет приятный, удобный в использовании интерфейс и предоставляет своим пользователям широкий спектр возможностей. Пользователь может добавлять аудиокниги в различных форматах, создавая собственную структурированную библиотеку, которую может полностью настроить под свои потребности. Удобство использования этой библиотеки заключается во множестве функций, предлагаемых приложением, например, добавление недостающей информации о книге вручную или через Интернет.

В дальнейшем планируется расширять данный проект, добавить поддержку многопользовательского режима, выгрузку библиотеки в облачное хранилище, а также реализовать собственный проигрыватель аудиофайлов, который дополнял бы существующее решение. Эти нововведения обеспечивали бы еще более высокий уровень управления библиотекой.

Подробно с кодом программы и документацией можно ознакомиться по ссылке https://github.com/Romkagai/books

ЛИТЕРАТУРА

1. *Костенко И.В., Грицкова А.А.* Феномен современной аудиокниги // МедиаВектор. – 2023. – № 8. – С. 52–56.

2. *Грузин Н.А.* Сравнение шаблонов проектирования архитектуры приложения: MVC, MVP и MVVM // Modern Science. – 2021. – № 1-1. – С. 434–440.

3. *Галигузова Е.В., Илларионова Ю.Е.* Сравнительный анализ архитектурных паттернов: MVC, MVP, MVVM // Матрица научного познания. – 2023. – № 5-2. – С. 106–111.

4. Руководство по Python [Электронный ресурс] // python.org URL: https://www.python.org/doc/ (дата обращения 20.05.2024).

5. Документация SQLite [Электронный ресурс] // sqlite.org URL: https://www.sqlite.org/docs.html (дата обращения 20.05.2024)

6. Документация библиотеки Mutagen [Электронный ресурс] // mutagen.readthedocs.io URL: https://mutagen.readthedocs.io/ (дата обращения 20.05.2024)

7. Документация QT [Электронный ресурс] // doc.qt.io URL: https://doc.qt.io/ (дата обращения 20.05.2024)

МЕТОД СТАТИЧЕСКОГО АНАЛИЗА УТЕЧКИ ПАМЯТИ В ДИНАМИЧЕСКИХ СТРУКТУРАХ ПРОИЗВОЛЬНОЙ СЛОЖНОСТИ

Голомазова А.Е., Андреева В.В.
Томский государственный университет
n-minus@mail.ru, avv.21@mail.ru

Введение

В процессе разработки программного обеспечения осуществляется работа с обширным объёмом данных и написание программного кода, который направлен на решение задач, связанных с обработкой потенциальных уязвимостей [1,2]. При увеличении сложности проекта происходит увеличение объёма обрабатываемой информации, что увеличивает вероятность возникновения ошибок или проблем в проекте.

Для сокращения количества ошибок в программном продукте программисту необходимо провести тщательный анализ. Однако у такого подхода есть недостатки, такие как человеческий фактор и затраты времени на поиск уязвимостей.

На сегодняшний день лучшим методом оценки качества программного обеспечения остаётся анализ, проводимый программистом в совокупности с вспомогательными инструментами разработчика.

Вспомогательные инструменты разработчика бывают нескольких видов:

1. Статические анализаторы кода. Эти инструменты проводят анализ исходного кода без его фактического выполнения и могут выявлять потенциальные уязвимости;
2. Динамические анализаторы памяти. Эти инструменты отслеживают использование памяти во время выполнения программы и могут обнаруживать некорректные операции, проводимые с ней.

Но можно выделить пару преимуществ статического анализа по сравнению с динамическим, а именно:

1. Способен обнаружить потенциальные утечки памяти ещё на этапе компиляции или анализа исходного кода до запуска программы.
2. Проводится на всем объёме исходного кода программы, что позволяет выявить утечки памяти во всех его частях, включая те, которые могут быть упущены при динамическом анализе.
3. Способен выявить потенциальные проблемы и ошибки в управлении памятью.
4. Поскольку статический анализ выполняется без фактического выполнения программы, он обычно требует меньше вычислительных ресурсов и времени.

5. И самое главное преимущество – есть возможность точно указать местоположение, где произошла неверная операция.

Цель данной работы заключается в разработке метода статического анализа исходного кода на языке программирования C/C++ с использованием динамического распределения памяти с целью обнаружения утечек памяти для динамических структур произвольной сложности. Это поможет предотвратить проблемы с управлением памятью на более ранних этапах разработки.

1. Утечки памяти

Любая линейная или нелинейная динамическая структура данных характеризуется тем, что для неё выделяется определённый участок памяти, который содержит указатель на первый элемент(далее по тексту будем обозначать HEAD), будь то список или дерево. Поэтому в случае утери или переопределения адреса (HEAD) неудалённые объекты, хранящиеся в памяти, будут продолжать занимать место до полной перезагрузки вычислительной машины.

Можно заключить, что утечка памяти представляет собой ситуацию, когда ресурсы выделяются, но не освобождаются, что приводит к невозможности доступа к объектам, хранящимся в куче [3].

Выделим несколько классов ошибок, которые могут встретиться при работе с произвольным связанным графом:

2.1. Потеря связи между узлами

Операции, связанные с перестановкой узлов в динамической структуре или их добавлением/удалением, могут повлечь за собой нарушение связей в связанном графе, вследствие чего нельзя будет восстановить доступ к соседним вершинам.

Рассмотрим пример. Пусть существует некоторый ориентированный граф G и связанные вершины: A, B, C, D, E, F, где рёбра представлены в качестве списка дуг: $(A \to B)$, $(B \to A)$, $(B \to D)$, $(B \to C)$, $(C \to B)$, $(D \to B)$, $(D \to C)$, $(C \to D)$, $(D \to E)$,$(E \to D)$, $(E \to F)$, $(F \to E)$. Граф G представлен на рис. 1.

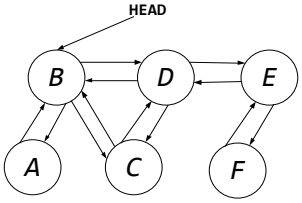


Рис. 1. Ориентированный связанный граф G

На рис. 2 показана операция удаления вершины D, в результате чего произошло нарушение связей между вершинами.

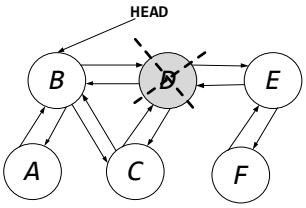


Рис. 2. Удаление вершины D в графе G

Удаление вершины D привело к потере доступа к вершинам E и F, оставив доступ только до вершины C. В результате этих изменений граф перестал быть связанным.

2.2. Потеря именованных указателей, имеющих доступ к динамической структуре

Вследствие некорректного переопределения указателей, указывающих на узлы динамических структур, есть возможность потерять доступ до динамической структуры в целом. Примером может послужить возможность переопределения указателя на новую, только что созданную вершину (рис. 3).

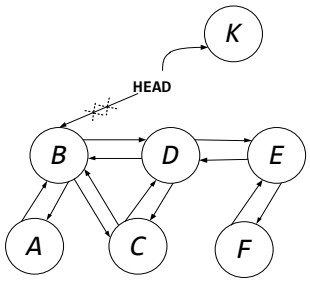


Рис. 3. Переопределение указателя HEAD

2. Предложенный метод статического анализа

В [4] был предложен метод, обнаруживающий утечки памяти в динамических списочных структурах, в основе которых лежит графовая структура [5]. Такая структура позволяет отследить состояние связей в процессе моделирования работы программы и метод посимвольного чтения. В данной работе предлагается масштабировать используемый метод посимвольного чтения и дополнить алгоритм подходами из нескольких рассмотренных методологий: использование графа потока управления и составление резюме межпроцедурных функций [6–10].

Рассмотрим множества Ψ (*память*), *Y* (*рабочая среда*). Множество Ψ представляет собой линейное множество – набор векторов, где первое поле вектора соответствует значению, эквивалентному вершинам вспомогательного графа, а второе поле вектора является подмножеством рёбер соответствующей вершины:

$$\Psi = \left\{ \left\langle K^1, \left(A_1^1, A_2^1, ..., A_m^1 \right) \right\rangle, ..., \left\langle K^n, \left(A_1^n, A_2^n, ..., A_p^n \right) \right\rangle \right\}.$$

Множество *Y* является линейным множеством, которое представляет собой набор векторов, состоящих из указателей анализируемого метода и вершин вспомогательного графа, на которые указывают указатели: $Y = \left\{ \left\langle V_1, K_1 \right\rangle, ..., \left\langle V_m, K_n \right\rangle \right\}$.

Введём множество *S* (*область действия*). Множество *S* является линейным множеством, которое представляет собой набор векторов, состоящих из подмножеств Ψ и *Y*: $S = \left\{ \left\langle \Psi_1, Y_1 \right\rangle, ..., \left\langle \Psi_m, Y_n \right\rangle \right\}$.

Вектор $\left\langle \Psi_1, Y_1 \right\rangle$ множества *S* будем именовать как *контекст графа* потока управления. Под контекстом будем понимать путь обхода анализируемого кода, который влияет на множества "память" (Ψ) и "рабочая среда" (*Y*), формируя историю их изменения, а множество контекстов – как множество путей исполнения программы и её окружения.

3. Алгоритм формирования множества S и его контекстов во время статического анализа кода

Пусть на первом шаге анализируемого кода существует некоторое множество *S* – набор элементов окружения *Y*, и некоторый набор элементов вспомогательного графа Ψ в нём: $L_1 = \left\langle \Psi_1, Y_1 \right\rangle$, $S = \left\{ L_1 \right\}$. (В данном случае можно сказать, что существует 1 контекст на первом шаге анализа.)

Если на втором шаге анализируемого кода следует условный оператор if, то:

а) внутри блока if существует такой контекст L_2: $L_2 = \{\Psi_2, Y_2 \mid \Psi_1 \in L_1 \cup \Psi_2 \text{ и } Y_1 \in L_1 \cup Y_2\}$ где Ψ_2 – сформированный вспомогательный граф динамической структуры блока if, Y_2 – переменные области видимости блока if;

б) внутри блока else существует такой контекст L_3: $L_3 = \{\Psi_3, Y_3 \mid \Psi_3 \in L_1 \cup \Psi_3 \text{ и } Y_1 \in L_1 \cup Y_3\}$, где Ψ_3 – сформированный вспомогательный граф динамической структуры блока else, Y_3 – переменные области видимости блока else;

в) после анализа условных операторов сформированные множества также подлежат изменению:

$$L_4 = \{\Psi_4, Y_4 \mid Y_4 \cup (Y_2 \in L_2 \cap Y_1 \in L_1) \text{ и } \Psi_4 \cup \Psi_2 \in L_2 \text{ и } \Psi_4 \cup \Psi_2 \in L_2\}$$

и

$$L_5 = \{\Psi_4, Y_4 \mid Y_4 \cup (Y_3 \in L_3 \cap Y_1 \in L_1) \text{ и } \Psi_4 \cup \Psi_3 \in L_3 \text{ и } \Psi_4 \cup \Psi_3 \in L_3\}.$$

Результат, не попавший в пересечения данных множеств, подлежит дополнительной проверке на наличие потерь ссылок $(Y_2 \in L_2 \cap Y_1 \in L_1)$ и $(Y_3 \in L_3 \cap Y_1 \in L_1)$. Теперь множество S содержит только контексты L_4 и $L5$: $S = \{L_4, L_5\}$.

Количество контекстов для количества m условных операторов равно $2^{m(n+1)}$, где n – количество вложенных условных операторов, m – количество невложенных условных операторов в коде.

Если на втором шаге анализируемого кода следует оператор while, то:

а) внутри блока while существует такой контекст L_2:

$$L_2 = \{\Psi_2, Y_2 \mid \Psi_1 \in L_1 \cup \Psi_2 \text{ и } Y_1 \in L_1 \cup Y_2\};$$

б) После анализа блока while или for

$$L_3 = \{\Psi_3, Y_3 \mid Y_3 \cup (Y_2 \in L_2 \cap Y_1 \in L_1) \text{ и } \Psi_3 \cup \Psi_2 \in L_2 \text{ и } \Psi_3 \cup \Psi_2 \in L_2\}.$$

Теперь множество S содержит только контекст L_3: $S = \{L_3\}$.

4. Составление резюме межпроцедурных функций

Если в анализируемом коде следует вызов метода, принадлежащего этому же проекту, и ранее данный метод был проанализирован, то будет применена методология резюме межпроцедурных функций, которая вернёт все возможные контексты программы. Если метод вернул нам n контекстов $S_1 = \{M_1, ..., M_n\}$, то множество S будет содержать $S = \{L_1 \cup M_1 \in S_1 \text{ и } ..., \text{ и } L_1 \cup M_n \in S_n\}$.

Объединение контекстов происходит в соотношении "многие ко многим". Для исключения повторного анализа функции мы сохраняем при первом её прохождении множество контекстов как результат её анализа.

Рассмотрим правила объединения двух контекстов $L_1 \cup M_1$:

$$L_1 \cup M_1 = \{(Y \in L_1) \cup (Y \in L_1 \cap Y_1 \in M_1) \text{ и } (\Psi \in L_1) \cup (\Psi \in M_1)\},$$

где $(Y \in L_1) \cup (Y \in L_1 \cap Y_1 \in M_1)$ – множество области видимости, сформировавшееся при объединении множеств именованных указателей.

Под *множеством областей видимости* будем понимать набор областей, в которых именованные указатели могут быть доступны и использованы. Опишем правила объединения множества памяти из разных контекстов для вызываемого объекта и реализуемого им метода: $(\Psi \in L_1) \cup (\Psi \in M_1)$.

Если вершина $\left\langle K^i, \left(A_1^i, A_2^i, ..., A_m^i\right)\right\rangle$ множества Ψ из контекста M_1 (вызываемой повторно функции) появилась в анализируемом коде через оператор new, то добавляем данную вершину в новый контекст как новую, сохраняя за ней все связи из контекста M_1.

Если указатель, ссылающийся на вершину K^i из M_1 ссылался на иную вершину из контекста L_1, то в новом контексте приоритет связи сохраняется за M_1.

Если вершина $\left\langle K^1, \left(A_1^1, A_2^1, ..., A_m^1\right)\right\rangle$ множества Ψ из контекста M_1 была добавлена через обращение именованного указателя, то добавляем данную вершину в новый контекст как уже имеющуюся, сохраняя за ней все связи из контекста M_1 и L_1.

Если указатель, ссылающийся на вершину K^i из M_1 ссылался на иную вершину из контекста L_1, то в новом контексте приоритет связи сохраняется за M_1.

Если к вершине $\left\langle K^1, \left(A_1^1, A_2^1, ..., A_m^1\right)\right\rangle$ множества Ψ из контекста M_1 было зафиксировано какое-либо обращение и после было произведено удаление, то вершина K^1 не сохраняется в новый контекст, а ссылки других вершин, указывающих на неё, удаляются.

Объединение множеств происходит от сопоставления стартовой вершины контекста M_1 и вершины, на которую указывает указатель реализующий вызываемый метод (текущая вершина).

Стартовая и текущая вершины считаются эквивалентными друг другу, если стартовая вершина не была отмечена в контексте через оператор new.

Оставшиеся вершины считаются эквивалентными друг другу, если входящие рёбра и вершины, относительно которых эти дуги являются исходящими, также имеют эквивалент в другом графе (будем называть их вершины-предки); иначе они различны. Пример сопоставлений с учётом указателя new указан на рис. 4.

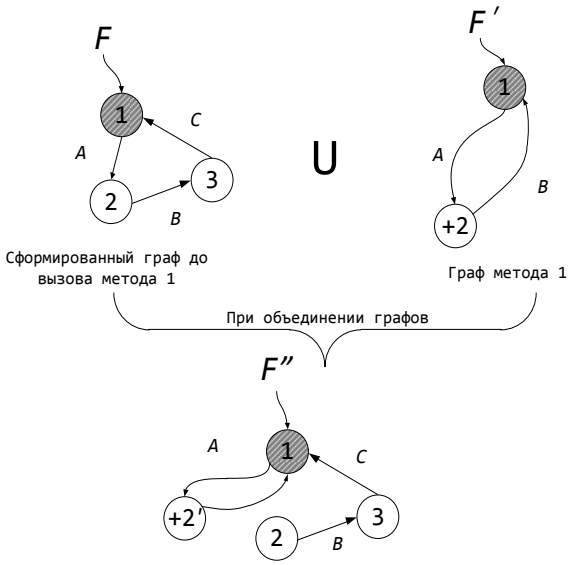


Рис. 4. Наложение графов анализируемого метода и метода 1

Указатель *F* указывает на текущую вершину текущего анализируемого метода в некотором контексте. Граф с указателем *F* описывает текущее состояние множества "Память" до вызова метода 1.

Вершина 1 из графа метода 1 является стартовой вершиной во множестве "Память" в некотором контексте. При формировании резюме метода 1 указатель в данном примере *F'* также по окончании анализа ссылался на стартовую вершину.

Будем считать, что вершины 1 из текущего анализируемого метода и из метода 1 – эквивалентны.

Обращение к вершине +2 было осуществлено через оператор new (строка 2, рис. 5), поэтому нельзя считать, что она эквивалентна вершине 2, хотя входящее ребро совпадает.

Вершина 3 из графа метода 2 не имеет эквивалентной вершины в графе метода 1, т.к. вершина 2, от которой направлена входящая дуга, не имеет аналога в графе метода 1.

```
1    void Method_1(int data){
2    this->A = new NodeA(data);
3    NodeA*temp = this->A;
4    temp->B = this;
5    }
```

Листинг 1– Метод 1

На рис. 5 приведён пример сопоставлений с учётом прямого обращения к указателю.

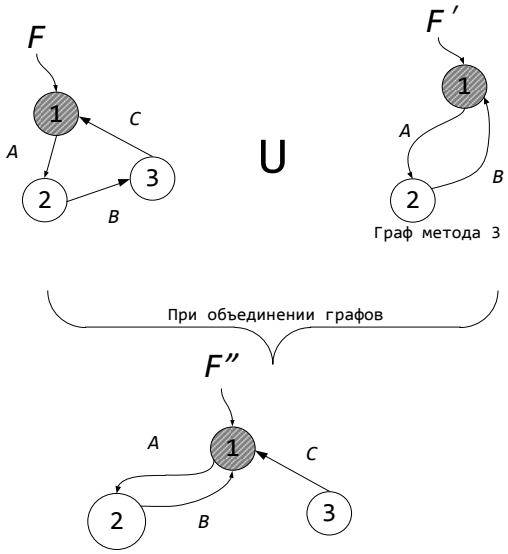


Рис. 5. Наложение графов анализируемого метода и метода 3

Указатель *F* указывает на текущую вершину текущего анализируемого метода в некотором контексте. Граф с указателем *F* описывает текущее состояние множества "Память" до вызова метода 1.

Вершина 1 из графа метода 3 является стартовой вершиной во множестве "Память" в некотором контексте. *F*' также ссылался на стартовую вершину метода.

Будем считать, что вершины 1 из текущего анализируемого метода и из метода 3 – эквивалентны.

Обращение к вершине 2 было осуществлено напрямую, как к уже имеющейся в куче узлу, без оператора new (строка 8 листинга 2).

Вершины-предки у вершин 2 их обоих графов совпадают, поэтому будем считать, что они эквивалентны. Вершина 3 из графа метода 3? по-прежнему? не имеет эквивалентной вершины.

Метод 3 продемонстрирован на листинге 2.

```
7    void Method_3(int data){
8    NodeA*temp = this->A;
9    temp->B = this;
10   }
```

Листинг 2 – Метод 3

Пример сопоставлений с учётом оператора delete представлен на рис. 6.

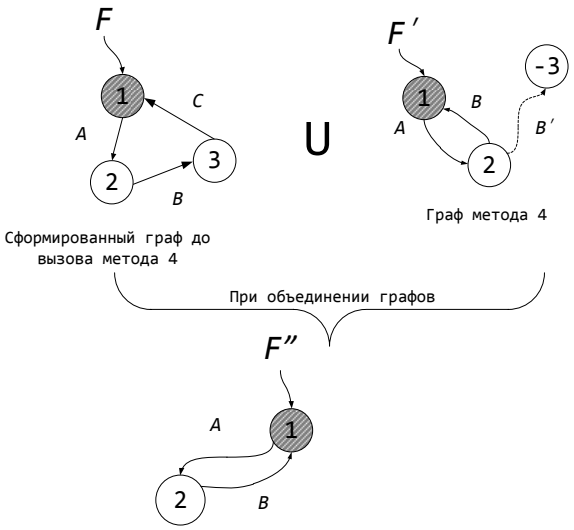


Рис. 6. Объединение графов анализируемого метода и метода 4

Разбор осуществляется, как представлено на вышеуказанных примерах, за исключением вершины –3. Т.к. в методе 4 она была удалена, то в резюме мы её отмечаем как удалённую вершину и в новый контекст со всем входящими и исходящими дугами она не попадает. Если в анализируемом текущем методе была найдена эквивалентная ей вершина, то она также удаляется и в уже новом резюме нового метода она также будет отмечена удалённой.

Метод 4 продемонстрирован на листинге 3.

```
        void Method_4(int data){
2       NodeA*temp = this->A;
        delete temp->B;
3       temp->B = this;

4       }

5

6
```

Листинг 3- Метод 4

Пример объединения графа с учётом сдвига головного указателя на рис. 7.

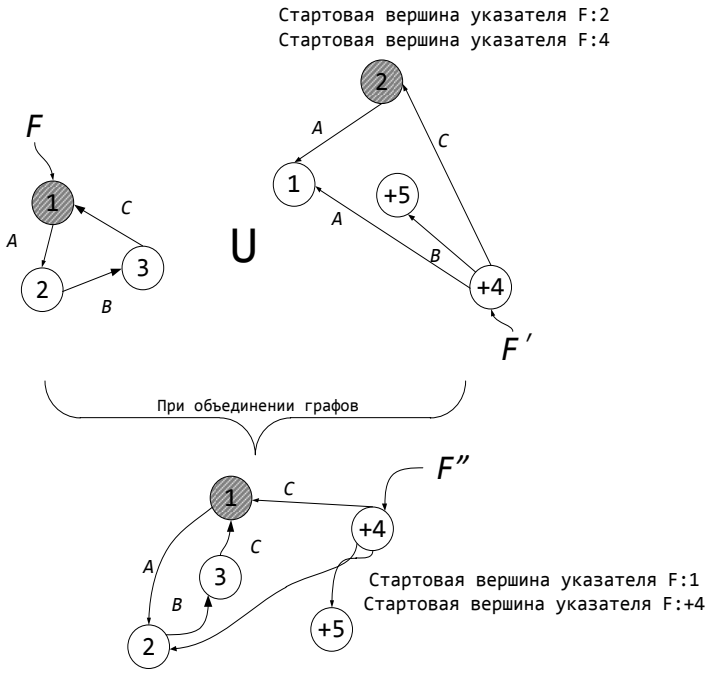


Рис. 7. Наложение графа, сформировавшегося до вызова некоторого метода, и графа вызываемого метода (соответственно), с учётом сдвига головного указателя

Рассмотрим пример наложения графов, когда именованные указатели, которые ссылаются на данные структуры, были сдвинуты. Наложение графов ведётся от стартовой вершины вызываемого метода и от текущей вершины анализируемого метода. Но в формировании нового контекста, именованный указатель одной и той же структуры будет эквивалентен указателю вызываемого метода, т.е. также ссылается на вершину +4.

Заключение

В работе предлагается метод статического анализа для обнаружения утечек ресурсов в динамических структурах произвольной сложности. Метод основывается на модификации модели, предложенной в работе [4] где моделирование кучи выполняется с помощью трёх множеств: "память", "рабочая среда" и "результат". В данной работе были введены дополнительные понятия множества контекста и контекст резюме межпроцедурных функций. Множество "контекст" состоит из подмножеств памяти и рабочей среды, что в процессе анализа позволяет учесть все сценарии развития анализируемого кода. Резюме межпроцедурных функций, в свою очередь, помогает на уже ранее построенном контексте получить результаты анализа вызываемого метода без дополнительной на то проверки. Также были введены правила для определения утечек памяти при работе с динамическими структурами произвольной сложности. Благодаря данному анализу результатом будут строки, где были совершены потери.

ЛИТЕРАТУРА

1. *Кулямин В.В.* Методы верификации программного обеспечения. – М.: Институт Системного Программирования РАН, 2008. – 111 с.

2. Static Code Analysis. In: van Tilborg H.C.A., Jajodia S. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-5906-5_1371 (дата обращения 10.06.2024).

3. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы. Построение и анализ. – Вильямс, 2013. – С. 13–24.

4. Материалы IX-й Международной научной конференции "Математическое и программное обеспечение информационных, технических и экономических систем", Томск, 26-28 мая 2022 г. /под общ. ред.: И.С. Шмырина. Томск: Изд-во Том. ун-та, 2022. – 309 с.

5. *Sai-ngern S.* An address mapping approach for test data generation of dynamic linked structures. Information and Software Technology. – 2005. – Vol. 47. – № 3. – P. 199–214.

6. A study on the weakness analysis for binary code in embedded environments – 2017. – International Journal of Software Engineering and Its Applications 11(10). – P. 53–62.

7. *Кошелев В.К.* Межпроцедурный статический анализ для поиска ошибок в исходном коде программ на языке C#: автореферат дис. кандидата физико-математических наук : 05.13.11. – Москва, 2017. – 24 с.

8. Improving Memory Management Security for C and C++. – International Journal of Secure Software Engineering – April 2010. – 1(2) – P. 57–82.

9. Static Verification Tools for C Programs and Linux Device Drivers: A Survey. – January 2012. – Proceedings of the Institute for System Programming of RAS, 22. – P. 293–326.

10. Modeling Memory with Uninterpreted Functions for Predicate Abstractions. – January 2015 . – Proceedings of the Institute for System Programming of RAS, 27 (5). – P. 117–142.

РЕАЛИЗАЦИЯ БЕЗОПАСНОГО УПРАВЛЕНИЯ УМНЫМ ДОМОМ С ПОМОЩЬЮ МЕССЕНДЖЕРА TELEGRAM

Горький К.Д., Самохина С.И.
Томский государственный университет
gorkykirill@mail.ru

Введение

Умный дом – это концепция, которая становится все более популярной. Данная тенденция предполагает использование современных технологий для управления устройствами в доме, такими как освещение, отопление, кондиционирование воздуха, безопасность и многое другое [1]. Системы умного дома позволяют контролировать всё это с помощью смартфона, планшета или голосового помощника. На рынке существует огромное количество платформ, которые облегчают управление умным домом, среди них самые популярные – Apple HomeKit, Google Home, Mi Home, Яндекс.Умный дом и др. [2]. Но, проанализировав популярные решения от именитых брендов, можно выделить общие недостатки, которые в той или иной мере присутствуют у всех продуктов, а именно – ограниченная совместимость, высокая стоимость, недостаточная приватность данных, ограниченная безопасность и ограниченная интеграция.

В последние годы рынок устройств для умного дома переживает стремительный рост, что сопровождается появлением новых угроз безопасности. Эти угрозы могут иметь серьёзные последствия как для конфиденциальности пользователей, так и для физической безопасности их жилищ [1].

В настоящее время существует множество угроз для умного дома, среди которых особенно выделяют кражу личных данных, атаки «человек посередине», распределенные атаки типа «отказ в обслуживании», несанкционированный доступ и отслеживание местоположения

В свете множества угроз крайне важно, чтобы умный дом был безопасен, чтобы злоумышленники с помощью известных уязвимостей не могли заполучить важные данные пользователей либо сам доступ к системе умного дома. Пользователям же, в свою очередь, также следует быть осведомлёнными о потенциальных рисках и активно использовать все доступные средства защиты, такие как двухфакторная аутентификация и сложные пароли.

1. Выбор архитектуры системы и оборудования

На рынке существует огромное множество различных реализаций, но все они имеют свои недостатки: от завышенной стоимости компонент умного дома до некачественного программного обеспечения, поэтому было решено создать свой проект, который отличается от других open-source-разработкой и возможностью заменой модулей

устройства. Роль пользовательского интерфейса выполняет популярный мессенджер Telegram.

Использование мессенджера для управления умным домом становится все более популярным решением благодаря его универсальности и возможностям. Telegram позволяет создавать ботов, которые могут выполнять множество функций, среди которых может выступать и управление – от удаленного контроля устройств до получения уведомлений о событиях в доме. Эти боты могут также интегрироваться с различными системами через API, предоставляя широкие возможности для автоматизации и упрощения домашнего управления.

Кроме того, приложение уже широко используется многими пользователями как средство ежедневного общения, что делает его привлекательным вариантом для интеграции в систему умного дома, ведь пользователи могут с легкостью управлять своими устройствами через уже знакомый интерфейс без освоения новых инструментов или приложений [3]. Таким образом, вся основная работа будет проходить в Telegram-боте под управлением компьютером.

Основное внимание уделяется архитектуре системы умного дома на базе Raspberry Pi, которая играет роль главного звена в моей конфигурации. Выбор данного решения обеспечивает гибкость и масштабируемость системы. Одним из плюсов является то, что система строится на модульной структуре, где каждый компонент может быть заменен без необходимости перестраивать всю систему [4]. Это достигается благодаря централизованной роли RP, которая обеспечивает управление и координацию всех подключенных устройств и модулей.

Архитектура программного обеспечения системы умного дома строится на принципах многослойности. Данные слои можно представить следующим образом:

1. Слой представления – пользовательский интерфейс, через который осуществляется взаимодействие пользователя с системой.
2. Слой передачи данных – обеспечивает связь между физическими устройствами и программным обеспечением, управляющим этими устройствами.

Структура, где Telegram выполняет роль пользовательского интерфейса, а RP – сервера, позволяет обеспечить не только удобство использования и настройки системы со стороны пользователя, но и высокую степень адаптации к изменяющимся требованиям безопасности и функциональности (рис. 1). В дополнение к этому, использование Raspberry Pi как центрального узла значительно упрощает интеграцию различных устройств и сервисов, повышая общую эффективность и расширяемость системы умного дома [4].

Рис. 1. Архитектура безопасного управления умным домом

2. Разработка Telegram-бота и программное обеспечение для Raspberry Pi

Прежде чем приступить к разработке telegram-бота, необходимо построить структуру программы. Самым удобным решением для автоматизации процессов, будет ис-

пользование конечных автоматов. Основными компонентами конечного автомата являются состояния (начальное состояние, допускающие состояния) и переходы.

Далее необходимо определиться с функционалом программы. Сформируем базовые сценарии использования устройств умного дома: видеонаблюдение и музыкальное сопровождение.

Теперь можно приступить к самой реализации (рис. 2).

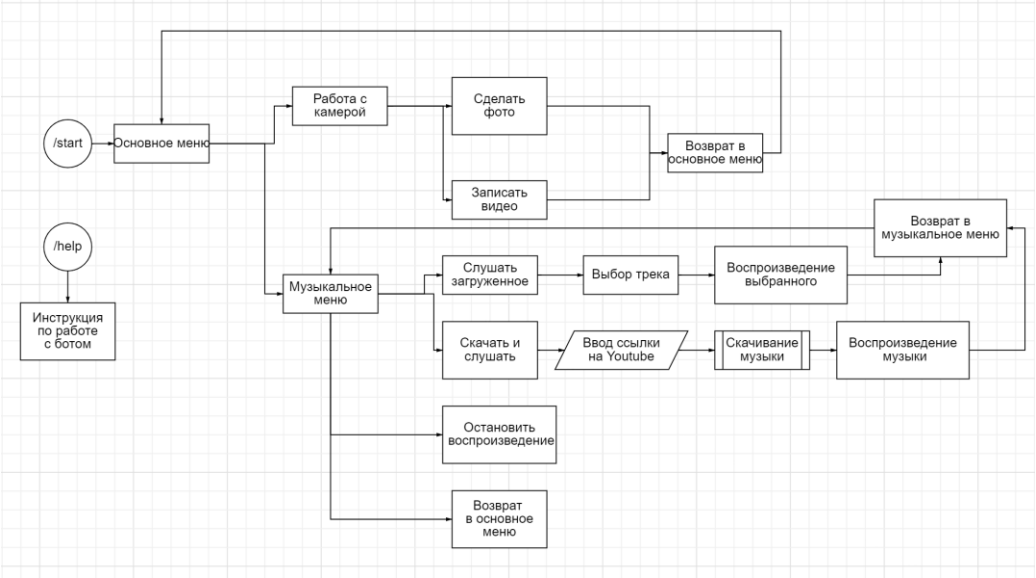


Рис. 2. Блок-схема Telegram-бота

Основная логика переключения между состояниями реализована через обработчики входящих сообщений, что позволяет боту адекватно реагировать на действия пользователя и поддерживать контекст текущей сессии (рис. 3).

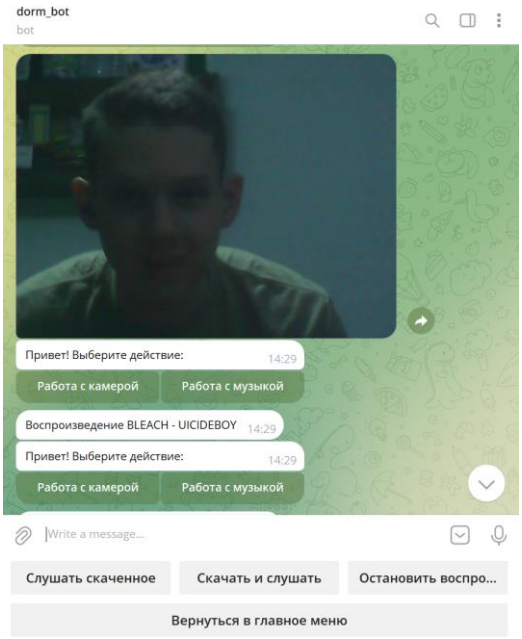


Рис. 3. Функционирующий Telegram-bot

Полученный код бота был структурирован таким образом, чтобы обеспечить лёгкость добавления новых функций и обработки сообщений от пользователей. Логика работы бота основана на принципах асинхронного программирования, что позволяет эффективно обрабатывать входящие запросы, даже при высокой нагрузке.

Теперь можно приступить и к настройке Raspberry Pi: необходимо обновить все патчи безопасности, закрыть все ненужные порты, оставив два системных порта [5] на крайние случаи, поставить дополнительные драйверы, которые определяют подключенное устройство, будь то камера или колонка (устройство для воспроизведения музыки).

3. Методы обеспечения безопасности: аутентификация, шифрование, защита от доступа

Методы обеспечения безопасности играют критически важную роль в защите системы умного дома. Основными ошибками, допущенными разработчиками, могут выступать отсутствие аутентификации, шифрования важных данных и недостаточная защита от несанкционированного доступа [6]. Именно эти угрозы безопасности мы пытались предотвратить в данной работе.

В реализации выбрана аутентификация с использованием пароля – наиболее удобное средство защиты, которое запрашивает пароль у пользователя перед предоставлением доступа к функциям бота (рис. 4).

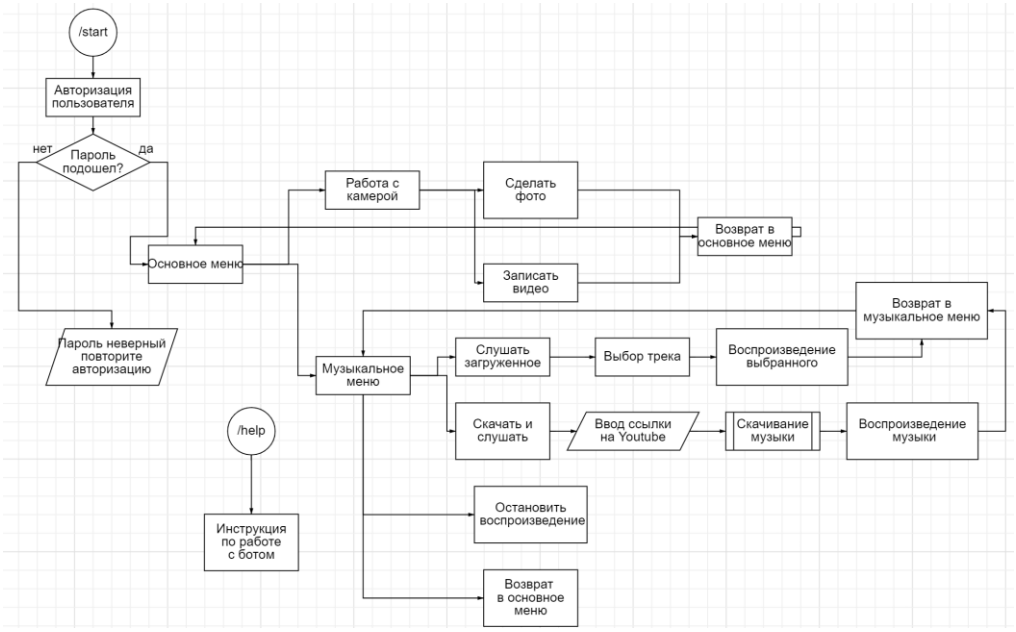


Рис. 4. Финальное представление Telegram-бота

Такое добавление аутентификации подразумевает изменение структуры автомата: после начала сессии пользователь должен авторизоваться, далее он может приступить к использованию бота. Кроме того, сессия не может быть бесконечной, ограничивается по времени 12-тью часами.

Теперь на Raspberry Pi хранятся не только уникальные данные пользователя (фото, видео, скаченные треки), но и пароль сессии. Чтобы эти данные были защищены, они подвергаются шифрованию. Шифрование обеспечивает защиту данных, передаваемых между устройствами умного дома и сервером, а также пользователя и бота Telegram.

Помимо шифрования в самом приложении на уровне «устройство с мессенджером – RP», должно происходить и шифрование на устройстве. Это реализуемо с ис-

пользованием соответствующих библиотек, таких как Cryptography. Данная библиотека представляет огромный набор инструментов и алгоритмов для шифрования, дешифрования, создание хэшей. Мы выбрали симметричное шифрование AES (рис. 5), которое позволяет зашифровать и расшифровать данные с использованием одного ключа, известного только Raspberry.

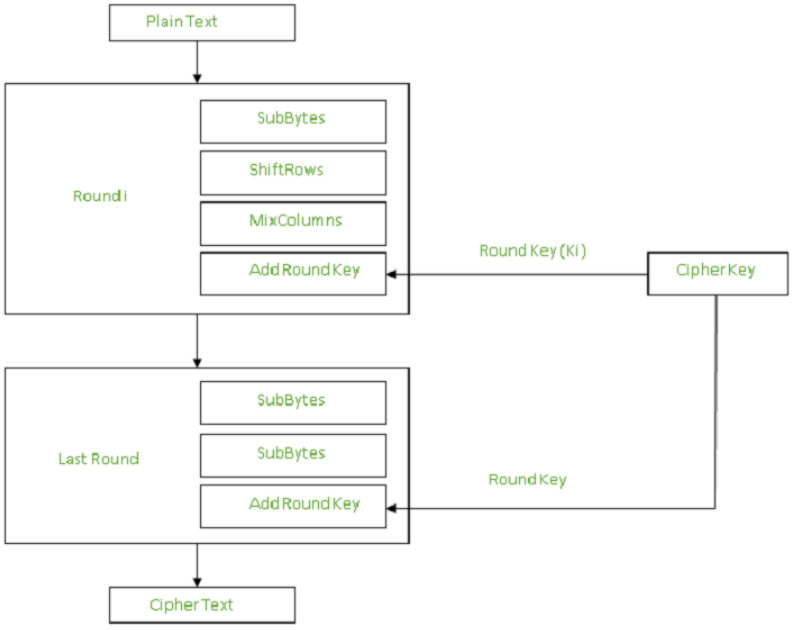


Рис. 5. Шифрование данных с помощью алгоритма AES

Но самой критичной зоной является несанкционированный доступ, поскольку даже в самых сложных проектах, имея физический доступ, злоумышленник может получить приватные данные. Поэтому ключевой задачей было снизить количество возможных рисков. Решением рисков могут послужить сетевой фаервол, обновление безопасности и ограничение прав доступа.

В роли сетевого фаервола выступает пользовательская утилита iptable, позволяющую системному администратору настраивать правила фильтрации IP-пакетов брандмауэра ядра Linux, реализованные в виде различных модулей Netfilter [5]. Данная утилита настроена на белом списке доступного трафика, что подразумевает запрет всего, что не разрешено. В нашем случае разрешены только SSH и HTTPS. Стоит пояснить, что по умолчанию это порты 22 и 80 соответственно, но в целях безопасности они были изменены на нестандартные, а остальные были закрыты, кроме 2 – системных, чтобы в случае неполадок, атак была возможность по ним подключиться к системе удаленно.

Обновление безопасности необходимо постоянно, чтобы не возникло атак на систему, ведь когда находят какую-нибудь уязвимость, она тут же предается огласке, попадая в «Банк данных угроз». Разработчики моментально пытаются ее “закрыть”, злоумышленники пытаются ей воспользоваться.

Ограничение прав доступа. Это важный аспект безопасности. Код запускается с учетной записи Linux, которая не имеет системных прав на изменение и модификацию файлов, что делает данную систему функциональной только в действиях вида "получи запрос, обработай, скачай или сделай фото, воспроизведи или верни фото".

Заключение

В данной работе рассмотрены основные аспекты создания системы управления умным домом с использованием мессенджера Telegram на платформе Raspberry Pi. Разработан Telegram-бот, обеспечивающий удобный и функциональный интерфейс для управления умными устройствами дома. Применены методы аутентификации, шифрования данных и защиты от несанкционированного доступа, что делает систему безопасной и функциональной.

ЛИТЕРАТУРА

1. *Greengard S*. The Internet of Things. – 2015. (дата обращения: 15.10.2024)
2. *Rozetked* Умный дом в 2024! Что такое Matter, Thread и Zigbee? [Электронный ресурс] – URL: https://www.youtube.com/watch?v=7zkt1vaNELI/ (дата обращения: 12.11.2023)
3. Overbafer1. Как Павел Дуров взломал Матрицу. [Электронный ресурс] – URL: https://www.youtube.com/watch?v=UujDkEi_BGY&list=LL&index=33&t=2539s (дата обращения: 15.03.2024)
4. Безопасный умный дом: сложная технология, полезная каждому. URL: http://news.ifmo.ru/ru/startups_and_business/startup/news/5832/ (дата обращения: 04.12.2024)
5. *Кирх О., Доусон Т.* Linux для профессионалов. Руководство администратора сети, второе издание. – СПб.:Питер, 2001. (дата обращения: 28.04.2024)
6. *Борисов М.В., Иванов А.П.* Актуальные угрозы информационной безопасности интернете вещей // Современные тенденции развития науки и технологий. – 2017. – №1-1. – С. 23–25

СОЗДАНИЕ ПРИЛОЖЕНИЯ, АНАЛИЗИРУЮЩЕГО ТЕКСТ

Литвак А.И., Пахомова Е.Г.
Томский государственный университет
sasha.litvak02@mail.ru, pakhomovaeg@yandex.ru

Введение

Объем текстовой информации, доступной для анализа и дальнейших выводов, в современном мире растет очень быстро, поэтому становится все более актуальной разработка инструментов и сервисов, которые смогут автоматически, быстро и бесплатно подготавливать к обработке и непосредственно обрабатывать и анализировать тексты. Одним из ключевых инструментов в данной теме являются методы обработки естественного языка (Natural Language Processing – NLP).

Русский язык в связи со своей долгой историей, обширностью, огромным количеством правил и исключений, множеством падежных форм, окончаний, различных видов спряжений и склонений, а также отсутствием четкой структуры предложений является очень сложным для изучения. Именно поэтому обработка текстов на русском – такая нетривиальная и интересная задача.

Цель проекта – создать сайт с требуемой для проведения анализа текста на русском функциональностью, включая морфологический анализ, проверку орфографии, извлечение именованных сущностей, анализ тональностей, построение поверхностно-семантического графа, выделение ключевых слов, перевод, суммаризация и анализ схожести текстов.

Выбор именно веб-приложения обусловлен удобством и универсальностью взаимодействий. Таким образом, не нужно скачивать отдельные приложения под каждую задачу.

Прежде чем приступить к созданию проекта, был проведен анализ существующих решений. Основными аналогами являются веб-сайты AOT.ru и retext.AI. Первый сервис может помочь с анализом морфологии слов и построением семантического графа, а второй с суммаризацией и извлечением ключевых слов.

1. Стек технологий

1.1. Трудности обработки русского языка

NLP на данный момент является трендом, поэтому существует огромное количество библиотек, ML-моделей и датасетов. Однако большинство этих ресурсов нацелены на английский и немецкий языки, что создает некоторые сложности для разработчиков, работающих с текстами на других языках, включая русский.

Русский язык отличается высокой сложностью и многообразием грамматических конструкций, что затрудняет его обработку и анализ. В отличие от английского и немецкого, для русского языка гораздо меньше готовых решений и специализированных инструментов.

Сложность русского языка для анализа связано с рядом факторов:

– большое количество правил и исключений к ним;
– множество окончаний, различных видов спряжений и склонений, создающих дополнительные сложности при лемматизации и стемминге;
– омонимы усложняют задачи по семантическому анализу;
– отсутствие четкой структуры предложений усложняет синтаксический анализ.

1.2. Стек технологий клиентской и серверной частей

Для клиентской части были выбраны технологии CSS и HTML, выбор был остановлен на них из-за того, что они являются стандартами в современной разработке веб-сайтов, а также ввиду легкости дальнейшего масштабирования проекта.

Для серверной части веб-сайта были выбраны технологии Flask и Jinja2. Jinja2 – быстрый и расширяемый шаблонизатор. Он по умолчанию экранирует переменные, выводимые в HTML, что предотвращает атаки XSS (межсайтовый скриптинг). Flask позволяет легко интегрировать различные расширения и библиотеки, обеспечивая модульность приложения.

1.3. Фреймворки, библиотеки, модели и алгоритмы NLP

Pymorphy2 [1] – морфологический анализатор. Его основными функциями являются приведение слова к нормальной форме (для глаголов – инфинитив, для существительных – именительный падеж, единственное число), установка слова в нужную морфологическую форму и возвращение грамматической информации о слове (например, число, род, падеж, часть речи, время, наклонение и т.д.).

В данном проекте эта библиотека используется для страницы с функциональностью морфологического анализа. Ниже приведены функции, использующие pymorphy2:

– анализ первичной морфологии слова:
  – получение нормальной формы слова;
  – получение грамматической информации о слове;
– подсчет количества разных частей речи в тексте;
– поиск и сохранение в список существительных и глаголов;
– извлечение именованных сущностей;
– получение всех форм отправленного слова (извлечение всех словоформ).

Pymorphy2 использует словарь OpenCorpora, который для русского языка содержит около 400 тысяч лексем и 5 миллионов отдельных слов. Для неизвестных слов строятся предположения. Чтобы сэкономить оперативную память и обеспечить быстрый анализ как словарных, так и несловарных слов, pymorphy2 извлекает парадигмы из лексем, преобразует информацию в цифры и кодирует слова в Directed acyclic word graph – направленный ациклический граф слов.

Spacy – популярная библиотека для обработки естественного языка. Она имеет поддержку 75+ языков, в том числе и русского, 84 предобученных модели для 25 языков, встроенный визуализатор и другие функции, необходимые для анализа.

На рис. 1 представлена архитектура модели. Конвейеры обучения разработаны с учетом эффективности и возможности настройки. Например, несколько компонентов

могут использовать общую модель "токен-вектор", а лемматизатор можно заменять или исключать. В последней версии некоторые компоненты зависят от результата предыдущих моделей.

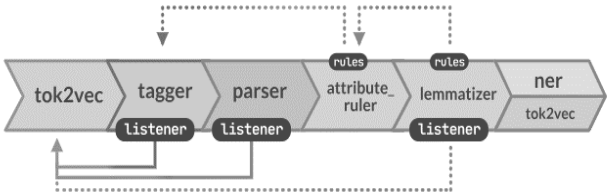


Рис. 1. Архитектура модели

В проекте используется модуль displacy, который позволяет построить по предложению семантический граф, наглядно показывающий, как именно компьютер понимает естественный язык.

TextBlob – библиотека Python для обработки текстовых данных. Она предоставляет простой API для решения распространенных задач обработки естественного языка (NLP), таких как тегирование частей речи, извлечение фраз существительных, анализ настроения, классификация и многое другое.

Эта библиотека используется в проекте для анализа тональностей текстов из файлов. После обращения к библиотеке функция возвращает именованный кортеж вида Sentiment(polarity,subjectivity). Оценка полярности (polarity) – вещественное число в диапазоне [–1.0;1.0], где –1.0 означает резко отрицательную, 0 – нейтральную, а 1.0 – резко положительную тональность. Субъективность (subjectivity) – также вещественное число, только в диапазоне [0.0;1.0], где 0.0 – очень объективная оценка (подходящее всем), а 1.0 – резко субъективная оценка (что-то очень личное).

Например, предложение "Этот ресторан отличается прекрасным обслуживанием, отличным качеством и свежестью еды, мне очень понравилось проводить время здесь, советую всем!" с помощью библиотеки был оценен, как положительная тональность и высокий уровень объективности:

Sentiment(polarity=0.8966666666666666,subjectivity=0.1333333333333334),

а предложение: "Лично мне фильм показался скучным и растянутым, потому что я не люблю боевики, я едва удерживал интерес до конца.":

Sentiment(polarity= –0.69791666666666666, subjectivity=0.7854166666666667).

К сожалению, библиотека TextBlob работает исключительно с текстом, написанным на английском языке, поэтому для других функций были использованы другие библиотеки, а для анализа тональностей предварительно выполняется машинный перевод с помощью API googletrans.

GoogleTrans –бесплатная и неограниченная библиотека, реализующая API Google Translate. Она не является официальной и использует Google Translate Ajax API для вызова таких методов, как detect и translate, использует те же серверы, что и translate.google.com.

Расстояние Левенштейна – метрика, измеряющая по модулю разность между двумя последовательностями символов. Она определяется как минимальное количество односимвольных операций (вставка, удаление, замена), необходимых для превращения одной строки в другую.

В данном проекте алгоритм используется для сравнения двух строк. Т.к. чаще всего при компьютерном наборе текста ошибками являются нажатие клавиши клавиатуры, рядом с необходимой (замена), пропуск буквы (вставка) и случайное повторное нажатие клавиш, например при залипании (удаление), то метрика Левенштейна достаточно хорошо проверяет данные тексты.

Если к списку разрешённых операций добавить транспозицию (замена местами двух соседних символов), получается расстояние Дамерау – Левенштейна. Для неё также существует алгоритм, требующий $O(M \cdot N)$ операций. Дамерау показал, что 80% ошибок при наборе текста человеком являются транспозициями.

В [2] на основе эксперимента было доказано, что, несмотря на то, что алгоритм Дамерау – Левенштейна показывает меньшую эффективность на больших строках, часто при решении задачи автоматического исправления ошибок, например, в поисковых запросах, он будет работать эффективнее, чем алгоритм Левенштейна. Это как раз связано с ошибками-транспозициями.

Для вычисления расстояния используется матричный алгоритм Вагнера – Фишера

$$D(i,j)=\begin{cases}0, & i=0,\ j=0,\\ i, & j=0,\ i>0,\\ j, & i=0,\ j>0,\\ \min\left\{D(i,j-1)+1, D(i-1,j)+1, D(i-1,j-1)+m\left(S_1[i],S_2[i]\right)\right\}, & j>0,\ i>0.\end{cases}$$

Т.к. нам не нужно хранить все элементы матрицы, можно сохранять только текущую и предыдущую строку для повышения производительности.

Алгоритм RAKE (Rapid Automatic Keyword Extraction) – один из гибридных (лингвостатистических) алгоритмов для выделения ключевых слов в текстах. Он основан на предположении, что ключевые элементы могут быть неоднословными, не содержат знаков пунктуации, служебных и десемантизированных слов (слова, частично или полностью семантически опустошенные, и, следовательно, малоинформативные или вовсе неинформативные). Ключевые выражения, выделенные в тексте с учетом словаря разделителей, ранжируются по весу, определяемому как сумма трех метрик (частота, степень (мера совместной встречаемости), отношение частоты к степени).

В [3] был проведен анализ методов извлечения ключевых слов и их сравнение. Алгоритм RAKE дал достаточно хорошие результаты, поэтому было решено использовать именно его.

В проекте использовалась библиотека rake-nltk, которая позволяет автоматически производить извлечение ключевых слов и выражений из текстов естественного языка. Библиотека предоставляет удобный интерфейс, одним из ключевых преимуществ использования rake-nltk является её способность работать с большими объемами текста, обеспечивая быстрое и точное извлечение ключевых слов.

HuggingFace – одна из популярных платформ для обработки естественного языка. На этом сайте собрано некоторое количество предобученных моделей, в том числе для суммаризации текстов на кириллице. Все модели должны быть обучены на каком-либо наборе исходных данных. Самыми популярными являются датасеты, использующие заголовки статей газет в качестве целевой переменной и сами тексты в качестве признаков или входных переменных. Также часто используются собранные посты из мессенджеров и социальных сетей, где целевым являются первые строки или заголовки, а остальной текст поста является признаками.

Выбранная модель обучена на заголовках российских новостных изданий. Именно эта модель показалась нам самой логичной для тестовых текстов.

Яндекс.Спеллер – сервис проверки правописания, предлагающий разработчикам использовать возможность интерактивной проверки орфографии на страницах своих сайтов. Спеллер анализирует слова, основываясь на правилах орфографии и лексике современного языка, а также использует технологии машинного обучения (ML – Machine Learning). Поддерживает 3 языка: русский, украинский и английский. В проекте API используется для выявления орфографических ошибок на странице морфологического анализа.

2. Интерфейс

2.1. Архитектура и классификация сайта

Назначение разработанного сайта – предоставление пользователям возможности анализа текстовой информации с использованием современных методов обработки естественного языка (NLP). Сайт предлагает функции, такие как морфологический анализ, построение семантического графа, сравнение текстов, выделение ключевых слов, суммаризация, анализ тональности и переводчик.

Сайт является динамическим, поскольку контент и результаты анализа генерируются в процессе работы с пользователем и добавляются по мере готовности. Динамичность сайта достигается за счет использования серверных технологий и взаимодействия с внешними библиотеками для обработки данных. Все результаты показываются на той же странице, без перехода на другие.

Клиентская часть является многостраничной (MPA, Multi-Page Application) для удобства пользователя.

Серверная часть сайта использует монолитную архитектуру (рис. 2), что означает, что все компоненты взаимодействуют напрямую через вызов функций и методов. Запуск файлов обработки из routes.py добавляет модульность.

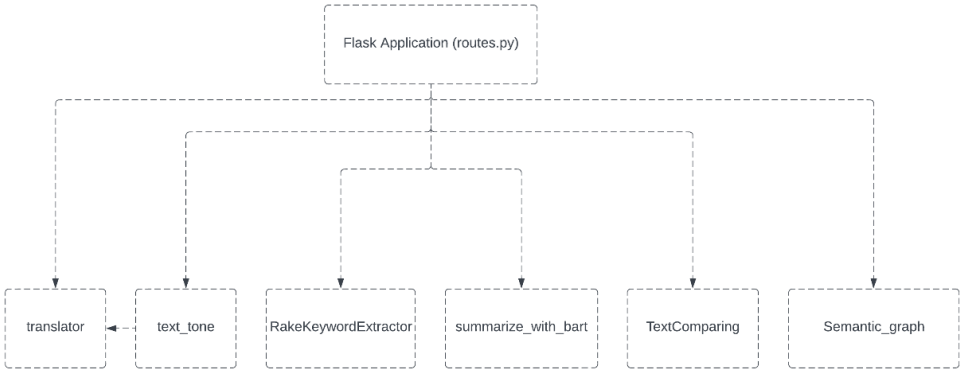


Рис. 2. Архитектура серверной части

2.2. Информация о каждой странице сайта

Все страницы содержат сайдбар, находящийся в левой части экрана (рис. 3), для легкого перехода к необходимым функциям, без необходимости возвращаться на начальную страницу. Также в хедере всех страниц, кроме главной, есть кнопка для перехода к начальной странице.

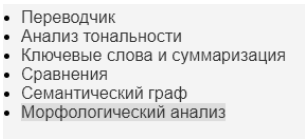


Рис. 3. Сайдбар

Переводчик. Как показано на рис. 4, сначала пользователю в специальное окно необходимо ввести текст. После этого при нажатии на кнопку "Перевести", отравляется POST-запрос, в котором в переменную сохраняется значения поля, и она передается в файл translator.py, который, с помощью API googletrans, делает автоматический машинный перевод и после этого возвращает уже переведенный текст и динамически, с помощью Jinja, встраивает в шаблон. Произведена обработка исключений, связанных с тем, что пользователь нажал кнопку, но никакой текст не был введен.

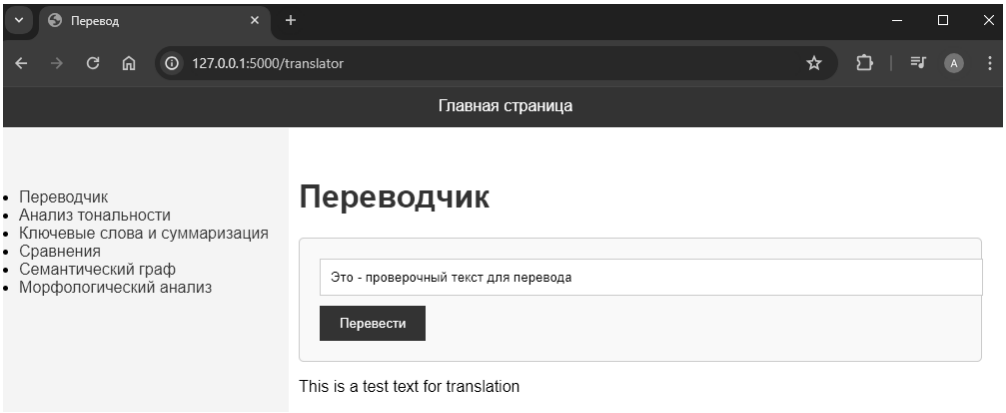


Рис. 4. Демонстрация работы страницы "Переводчик"

Анализ тональности. На рис. 5 представлен интерфейс страницы "Анализ тональности". Пользователю предлагается загрузить файл в формате .txt и нажать на кнопку "Upload and Analyze". Необходимо, чтобы каждое отдельное высказывание или отзыв были на отдельной строке. После проверяется, что файл был загружен, проводится предобработка текста (разделение по строкам, очистка и автоматический машинный перевод). После этого список очищенных и переведенных строк отправляется в файл text_tone.py, в котором с помощью библиотеки TextBlob узнается тональность текста и его субъективность, а также средний показатель тональности.

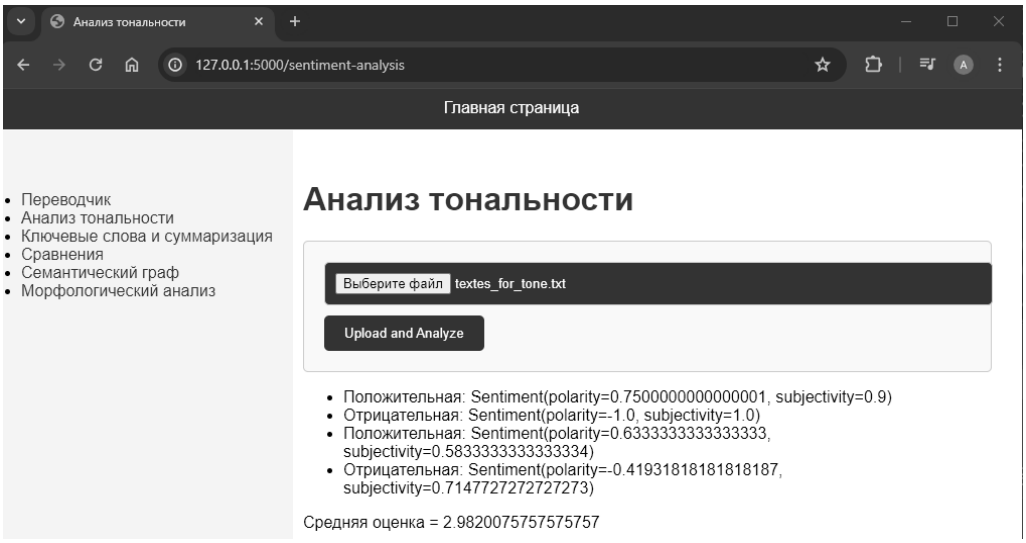


Рис. 5. Демонстрация работы страницы "Анализ тональности"

Т.к. тональность представляет собой величину в промежутке [–1.0;1.0], а отзывы чаще всего представлены в промежутке [1.0;5.0], было принято решение воспользоваться линейной трансформацией. Итоговая формула выглядит как $Y = 2X + 3$, где X – исходная величина, полученная из TextBlob, Y – конечная.

Средний показатель тональности может быть использован для оценки правдивости оценок мест, заведений, фильмов или другим объектов, на которые пользователи ставят оставляют оценку и отзыв.

Ключевые слова и суммаризация. Ключевые слова помогают алгоритмам поисковых систем находить веб-страницы, которые следует показать пользователям, а также часто используются в научных статьях. Правильный подбор ключевых слов помога-

ет бизнесу лучше продать свои услуги посредством более релевантных названий для продвижения товаров и услуг.

На рис. 6 продемонстрирован пользовательский интерфейс. Для начала пользователю необходимо прикрепить непустой файл в формате .txt в форму, далее нажать на кнопку "Upload and Analyze". В это время файл открывается, проверяется, что он был добавлен и является непустым, и его содержимое отправляется в файл keywords_and_summarize.py, а именно – в класс RakeKeywordExtractor, в котором с помощью алгоритма RAKE выполняется извлечение ключевых слов. Результат возвращается в виде списка кортежей, содержащих само слово и его "вес" – важность в данном тексте. Далее, уже на странице, распаковывается и записывается динамически в виде ненумерованного списка.

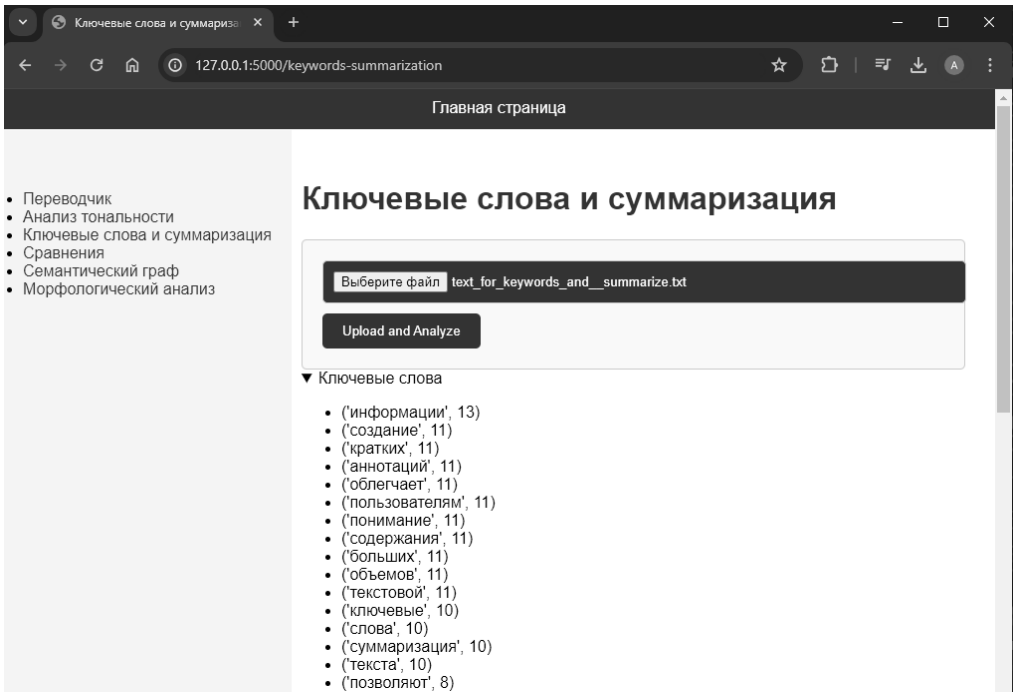


Рис. 6. Демонстрация работы извлечения ключевых слов

Суммаризация – удобный инструмент для экономии времени. Он помогает быстро понять основное содержание большого текста без необходимости читать его полностью.

На рис. 7 представлен пример работы модуля суммаризации. Т.к. было принято решение проводить извлечение ключевых слов и суммаризацию на одной странице, обработка происходит с тем же файлом. Полученный из файла текст напрямую отправляется в файл keywords_and_summarize.py, а именно – в функцию summarize_with_bart, которая на вход получает текст и личный API_TOKEN, полученный после регистрации на сайте HuggingFace. Далее, после отправки POST запроса напрямую к выбранной модели, возвращается уже суммаризованный текст. Готовое предложение возвращается в routes.py, в котором возвращается html-шаблон с результатом.

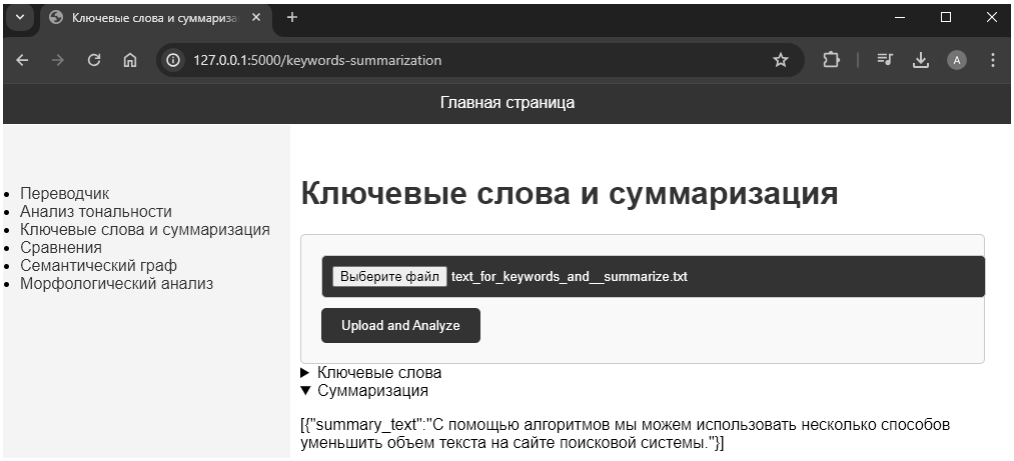


Рис. 7. Демонстрация работы суммаризации

Сравнения. На странице "Сравнения" используется метрика Левенштейна. Такой инструмент просто необходим при анализе текстовых данных. На основе такого алгоритма можно создать модуль, который будет помогать исправлять ошибки и опечатки на этапе очистки или предобработки датасета. На рис. 8 продемонстрирован результат работы алгоритма Вагнера – Фишера для метрики Левенштейна.

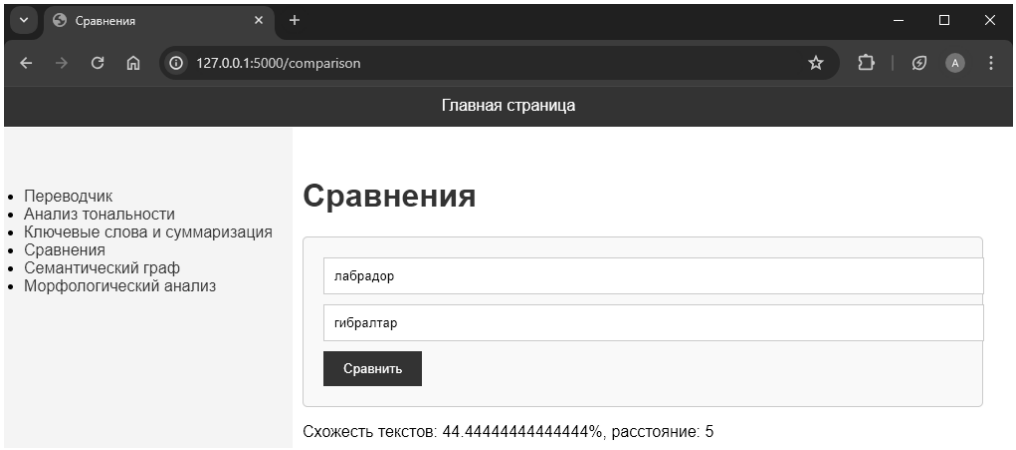


Рис. 8. Демонстрация работы страницы "Сравнения"

Семантический граф. В общем виде семантическая сеть – множество вершин, каждая из которых соответствует определенному объекту (понятию, факту, явлению или процессу), а между вершинами заданы различные отношения, представляемые дугами.

Достоинства:

– большие выразительные возможности, естественность и наглядность системы знаний, представленной графически;
– близость структуры сети, представляющей систему знаний, семантической структуре фраз естественного языка;
– данная модель более других соответствует современным представлениям об организации долговременной памяти человека.

Недостатки:

– громоздкость и неэффективность представления знаний только аппаратом семантической сети;
– сложность организации процедуры поиска нужного знания (как фрагмента сети).

Страница "Семантический граф" представлена на рис. 9. Она ожидает от пользователя ввода предложения и нажатия кнопки "Построить граф". После этого введенное предложение отправляется в файл semantic_graph.py. В этом файле с помощью библиотеки spaCy и модели "ru_core_news_sm" создается svg-разметка, в которой заменяются английские названия частей речи и зависимостей на русские для более понятного интерфейса. Перевод зависимостей был выполнен вручную, он включает в себя все возможные слова и аббревиатуры, которые могут использоваться в разметке. Далее уже переведенная и отредактированная разметка возвращается в semantic_graph.html и вставляется напрямую.

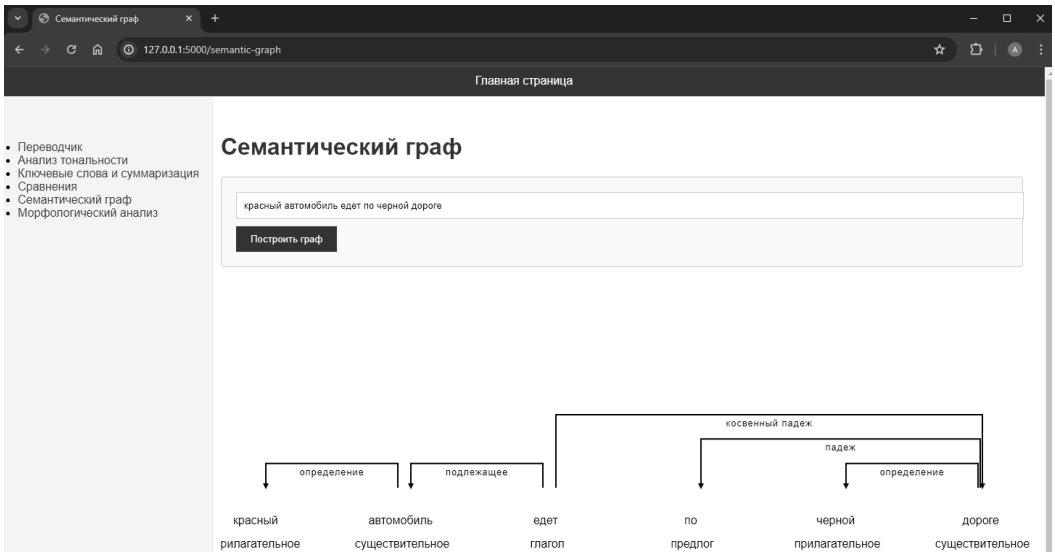


Рис. 9. Демонстрация работы страницы "Семантический граф"

Как видно на примере, под словами указывается их часть речи, а под дугами – отношения между ними.

Морфологический анализ. Страница морфологического анализа предлагает обширные статистические критерии оценки текста, анализ морфологии слов, извлечение именованных сущностей и исправление опечаток. К статистическим функциям относятся количество слов, количество именных частей речи, частота встречаемости по словам и частям речи.

Именованные сущности (Named Entity Recognition, NER) – слова или словосочетания, которые выделяют определенный предмет или явление из ряда однотипных предметов и явлений. К ним относятся ФИО, город, название компаний и организаций, адреса, названия книг, фильмов и др.

Извлечение именованных сущностей нужно для большого числа задач, например, для

– анонимизации текста;

– автоматического создания метаданных;

– создания геоинформационной системы (GIS) для дальнейшего анализа;

– обнаружения упоминаний и мониторинг новостей.

В данном проекте извлечение сущностей работает с помощью библиотеки pymorphy2. Проверяются теги (наборы граммем, характеризующих данное слово) при морфологическом анализе, и, если слово имеет тег Name (имя), Surn (фамилия), Patr (отчество), Geox(топоним), Orgn (организация), Trad (торговая марка), то данное слово добавляется в список и возвращается пользователю. Демонстрация работы на рис. 10.

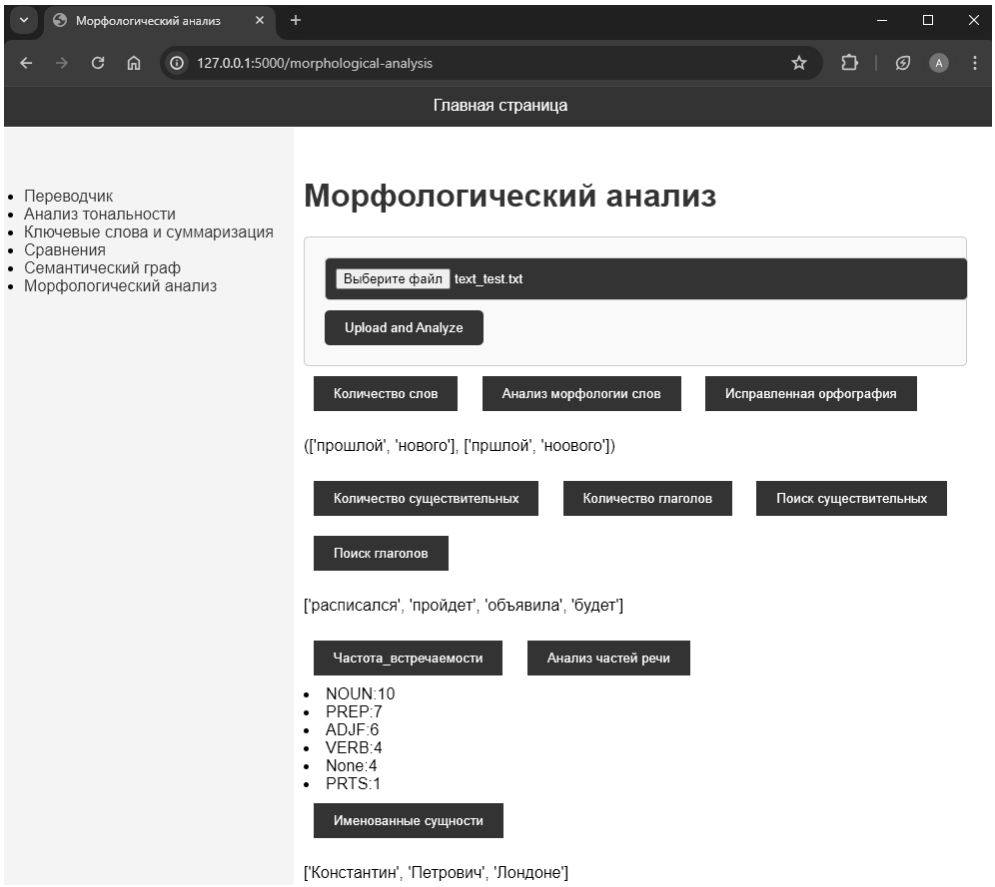


Рис. 10. Демонстрация работы исправления орфографии и NER

Исправление опечаток работает с помощью обращения к API Яндекс.Спеллер, возвращается наиболее вероятное слово. По результатам тестирования можно сделать вывод, что исправление работает хорошо, т.к. наиболее частые ошибки при компьютерном вводе – односимвольные (пропуск буквы, замена на букву, стоящую на клавиатуре рядом или повторное нажатие при залипании). Пример работы на рис. 10.

Заключение

В рамках проекта была поставлена цель создать веб-приложение, которое осуществляет анализ текстовой информации с применением современных методов обработки естественного языка и современных технологий.

Поставленная цель достигнута. В результате проделанной работы был создан функциональный веб-сайт с дружественным интерфейсом, предоставляющий широкий спектр возможностей для анализа текста по различным параметрам.

В ходе работы были изучены методы анализа текста, необходимые библиотеки, алгоритмы, модели и языки программирования. Также были проанализированы конкуренты.

Разработанный сервис может быть полезен лингвистам, людям, которые хотят профессионально изучать русский язык, иностранцам, которые хотят научиться говорить на нем правильно, а также аналитикам данных.

Можно заключить, что данный проект внес вклад в развитие инструментов для анализа текстовых данных, и он может быть использован в научной и профессиональной деятельности.

В дальнейшем планируется добавить дополнительную функциональность, чтение разных форматов документов, обеспечить безопасность сайта от внешних атак и загрузить программу на удаленный сервер для непрерывной работы.

ЛИТЕРАТУРА

1. Морфологический анализатор pymorphy2 – URL: https://pymorphy2.readthedocs.io/en/stable/ (дата обращения: 20.05.2024)

2. *Погорелов Д.А., Таразанов А.М., Волкова Л.Л.* Сравнительный анализ алгоритмов редакционного расстояния Левенштейна и Дамерау – Левенштейна // Синергия Наук. – 2019. – № 31. – С. 803–811.

3. *Митрофанова О.А., Гаврилик Д.А.* Эксперименты по автоматическому выделению ключевых выражений в стилистически разнородных корпусах русскоязычных текстов // Terra Linguistica. – 2022. – Т. 13. – №4. – С. 22–40.

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ "CINEMA SCHEDULE"

Плиско В.В., Пахомова Е.Г.

Томский государственный университет

vova.plisko@mail.ru, pakhomovaeg@yandex.ru

Введение

В настоящий момент современные технологии позволяют нам улучшать повседневную жизнь, в том числе и в области развлечений. По-прежнему важную роль в области развлечений играет посещение кинотеатров. Конечно, большая часть фильмов доступна на различных онлайн-ресурсах, но премьеры большинства фильмов все же проходят в кинотеатрах. Кроме того, домашний телевизор, как правило, воспроизводит картинку и звук более низкого качества, чем оборудование кинотеатра. В общем, можно найти много аргументов в пользу посещения кинотеатра. Фактом же является то, что кинотеатры востребованы сейчас так же, как и в те времена, когда домашний телевизор только входил в жизнь людей.

Целью данной работы является разработка веб-приложения "Cinema Schedule", с помощью которого пользователь смог бы получать актуальную информацию о фильмах в кинотеатрах г. Томска и г. Северска, сеансах и ценах на интересующие его сеансы. Также пользователь может подписаться на уведомления о новинках, которые выйдут в кинотеатрах в будущем.

Существующим аналогом созданного ресурса является онлайн-сервис "Киноафиша" – сервис, который предоставляет фильмы и сериалы в зависимости от выбранного пользователем города. Он показывает фильмы и возможные на них сеансы в кинотеатрах того города, который указал пользователь. Минусом данной площадки является неполнота предоставляемой информации. О некоторых фильмах информации нет, т.к. эти фильмы проходят в рамках киноклуба в связи со сложившейся в стране ситуацией.

1. Стек технологий

Для реализации проекта был выбран язык программирования Python, т.к. он наиболее прост и интуитивен. Для этого языка существует множество библиотек, которые облегчают разработку.

Рассмотрим фреймворки для создания клиент-серверного приложения.

Django. Это открытый фреймворк для разработки веб-приложений на языке Python. Он использует шаблон проектирования MVC. Django предоставляет готовые инструменты и правила для работы с веб-сайтами [1].

Flask. Flask представляет собой легковесный фреймворк для создания веб-приложения на Python. Он предоставляет самые базовые возможности, чтобы не нагружать программиста изучением фреймворка.

FastAPI. Это простой в понимании, легковесный веб-фреймворк, изначально созданный для разработки API [2]. В его основе лежат две библиотеки – Starlette и Pydentic.

Т.к. веб-приложение не будет большим, то и фреймворк должен быть легок в понимании и разработке; именно поэтому был выбран FastAPI.

2. Архитектура базы данных приложения

Существуют два типа баз данных: реляционные и нереляционные. Реляционные базы данных содержат в себе таблицы, состоящие из записей. Эти таблицы имеют между собой какие-либо отношения. В нереляционных базах данных, в свою очередь, не используется табличная схема строк и столбцов. Обычно они содержат более сложные структуры данных. Такие данные только частично структурированы. Такие базы данных используют для хранения больших данных.

Была выбрана реляционная база данных SQLite [3], т.к. она достаточно компактная и подходит под размеры веб-приложения. Для веб-приложения используются три реляционные базы данных: films, send_message и user. Рассмотрим каждую из них подробнее.

База данных films состоит из 3-х таблиц: cinema_table, film_table и sessions_table. Таблица cinema_table состоит из полей id – уникального идентификатора кинотеатра, и name – названия кинотеатра. В таблице film_table содержатся следующие столбцы: id – уникальный идентификатор фильма, name – название фильма, url – ссылка на фильм, img_url – ссылка на постер, cinema_id – уникальный идентификатор кинотеатра. Таблицы sinema_table и film_table связаны между собой полями id в таблице cinema_table и cinema_id в таблице film_table. Связь является типом "один ко многим". В таблице sessions_table находятся поля film_id – id фильма, datetime – дата сеанса, price – цена. Таблицы film_table и sessions_table имеют связь один ко многим с помощью полей film_id в таблице sessions_table и id в таблице film_table.

Связи и все поля в базе данных films показаны на рис. 1.

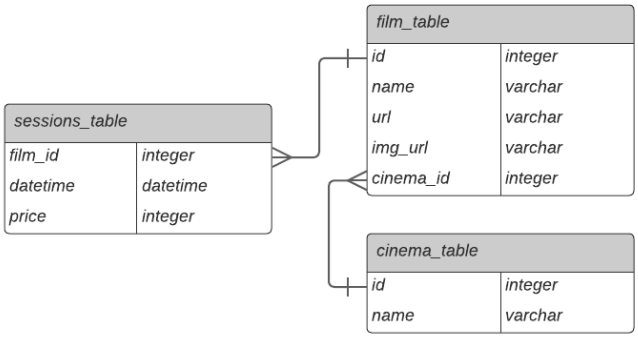


Рис. 1. Архитектура базы данных с фильмами

3. Общая архитектура веб-приложения

Рассмотрим общую архитектуру веб-приложения и его работу. Она изображена на рис. 2. Клиент отправляет http-запросы на сервер. С помощью ASGI-сервера uvicorn происходят взаимодействия между бизнес-логикой на стороне сервера и самим веб-сервером. Это происходит потому, что python-приложение не может напрямую принимать и обрабатывать http-запросы. Обращаясь к базе данных SQLite в бизнес-логике, получаем нужную информацию и возвращаем ответ клиенту также через uvicorn. В

файле main_parser происходит плановое обновление базы данных с фильмами и рассылка email сообщений.

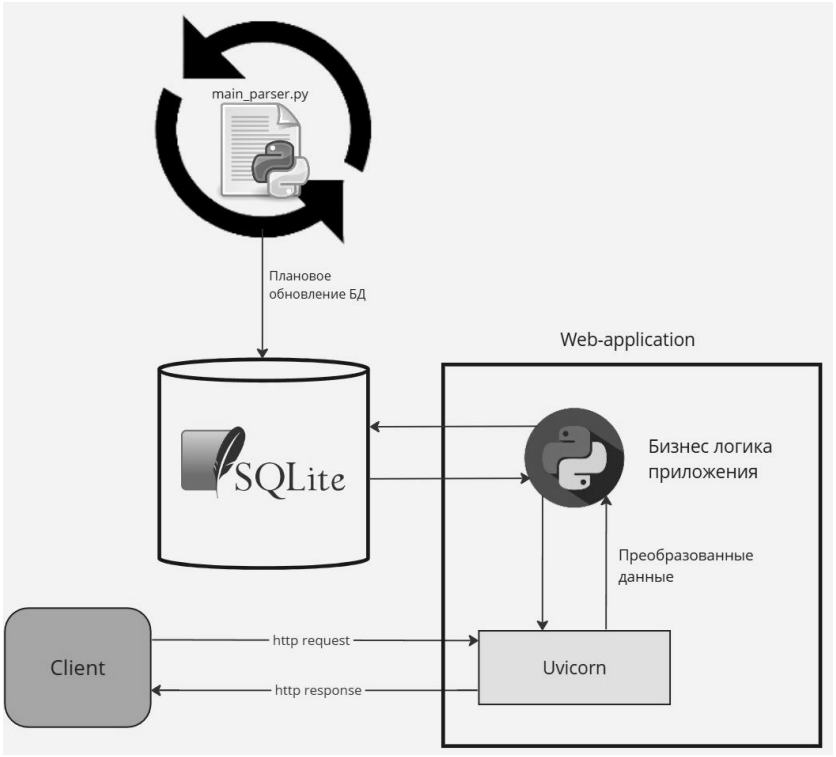


Рис. 2. Общая архитектура веб-приложения

4. Диаграмма вариантов использования

Диаграмма вариантов использования отражает функционал разрабатываемого ПО, который доступен для определенной группы пользователей. На рис. 3 изображена диаграмма вариантов для авторизованного и неавторизованного пользователей.

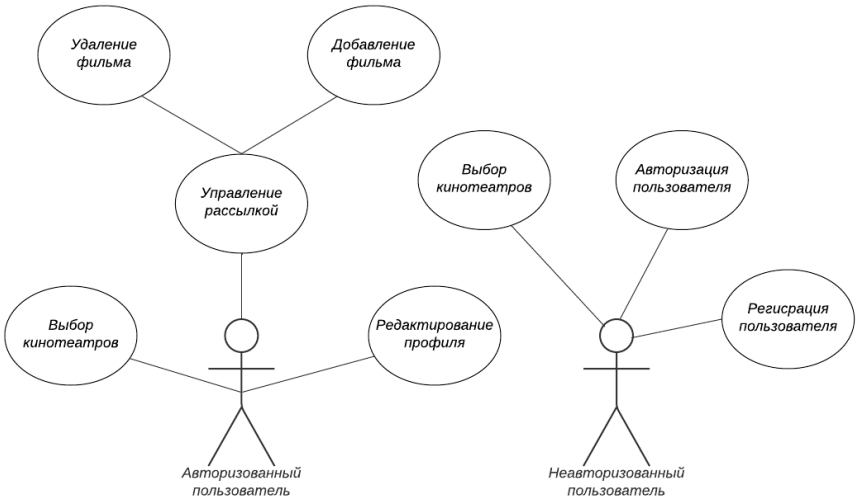


Рис. 3. Диаграмма вариантов использования

5. Реализация клиентской части приложения

Рассмотрим реализацию клиентской части приложения для авторизованного и неавторизованного пользователей.

Для неавторизованного пользователя структура клиентской части показана на рис. 4, для авторизованного пользователя – на рис. 5.

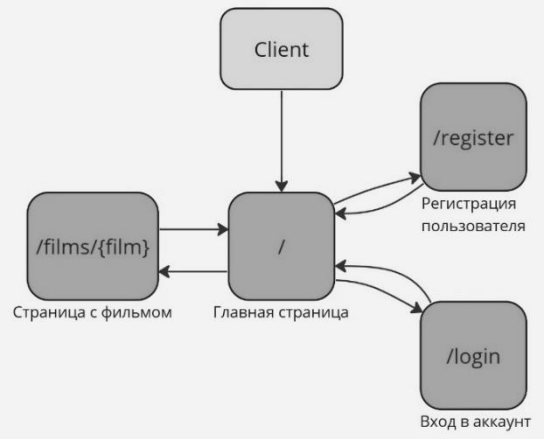


Рис. 4. Структура клиентской части для неавторизованного пользователя

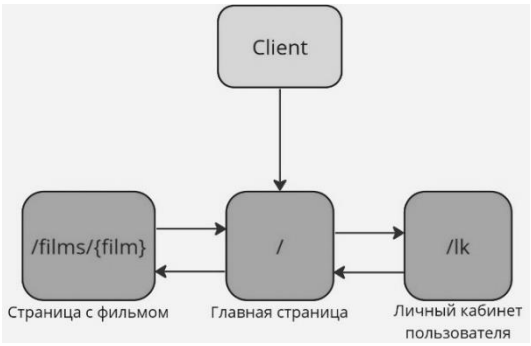


Рис. 5. Структура клиентской части для авторизованного пользователя

6. Интерфейс пользователя

На рис. 6 показано как выглядит главная страница сайта.

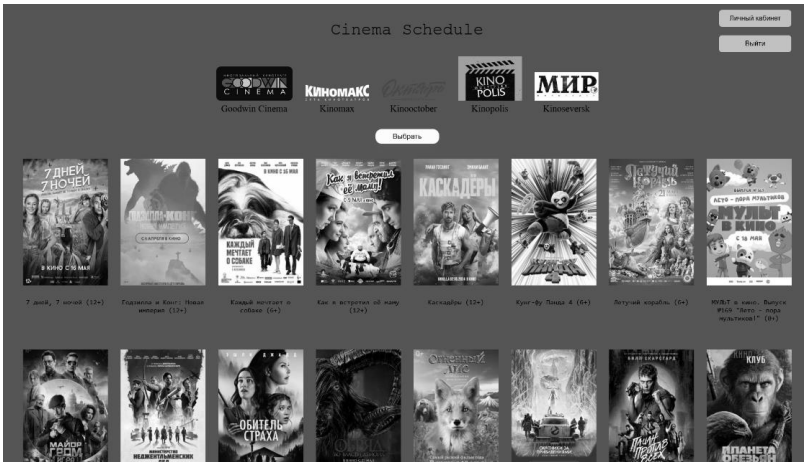


Рис. 6. Главная страница

На рис. 7 изображена страница регистрации пользователя.

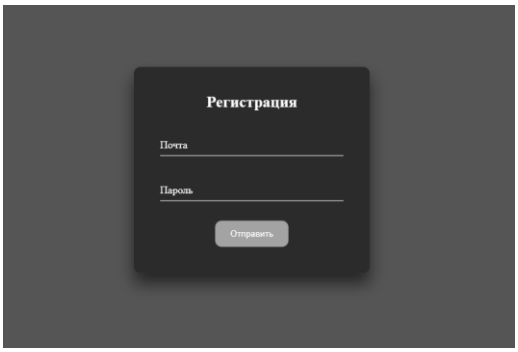


Рис. 7. Страница регистрации пользователя

7. Реализация серверной части приложения

Вся реализация серверной части находится в файле main. Создаются экземпляр класса FastAPI, экземпляр класса Jinja2Templates, экземпляр класса manager. На рис. 8 показано создание экземпляров класса.

```
app = FastAPI()
templates = Jinja2Templates(directory='templates')
database = manager()
```

Рис. 8. Создание экземпляров классов

Далее создаются обработчики для конечных точек. Всего реализовано 11 обработчиков – 5 обработчиков типа GET и 6 типа POST. GET-обработчики возвращают пользователю html-шаблон. POST-обработчики нужны для того, чтобы сервер принял введенные пользователем данные и обработал их. Рассмотрим подробнее GET-обработчики.

Обработчик главной страницы. Происходит проверка наличия cookie под названием cinemas. Это – те кинотеатры, которые выбрал пользователь. С помощью библиотеки SQLAlchemy производится обращение в базу данных для получения информации о фильмах. Если cookies есть, тогда из базы данных получаем только фильмы выбранных кинотеатров. Если их нет – получаем фильмы всех кинотеатров. Результат запроса в базу данных записываем в словарь под названием context, где ключом является строка films. Далее проверяется наличие JWT токена в cookie. Если он имеется, тогда в словарь context с ключом token устанавливается значение True. Пользователю возвращается Jinja-шаблон, в который передано название html-шаблона и словарь context, который использует шаблонизатор для заполнения шаблона данными. На рис. 9 показана реализация обработчика.

```python
@app.get('/')
def get_main(request: Request):
    engine = database.film_engine
    meta = MetaData()
    meta.reflect(engine)
    film_table = meta.tables['film_table']

    if (request.cookies.get('cinemas')):
        try:
            cinemas = [int(i) for i in request.cookies.get('cinemas').split(',')]
        except:
            cinemas = [1,2,3,4,5]
        with engine.connect() as connection:
            films = connection.execute(film_table.select().group_by(film_table.c.name)
                                       .where(film_table.c.img_url !='')
                                       .where(film_table.c.cinema_id.in_(cinemas))
                                       .with_only_columns(film_table.c.name,film_table.c.img_url)).fetchall()
    else:
        with engine.connect() as connection:
            films = connection.execute(film_table.select().group_by(film_table.c.name)
                                       .where(film_table.c.img_url !='')
                                       .with_only_columns(film_table.c.name,film_table.c.img_url)).fetchall()

    context = {"films":films, 'token':False}
    if request.cookies.get('token'):
        context['token'] = True
    return templates.TemplateResponse(request=request, name='main.html',context=context)
```

Рис. 9. Реализация обработчика главной страницы

Обработчик входа пользователя в аккаунт. Здесь проверяется только наличие у пользователя JWT-токена. Если он присутствует, значит, пользователь уже авторизован, поэтому возвращается главная страница. Если же пользователь не авторизован, ему возвращается шаблон login.html. На рис. 10 представлена реализация обработчика.

```python
@app.get('/login')
def login(request: Request):
    if (request.cookies.get('token')):
        return RedirectResponse('/')
    return templates.TemplateResponse(request=request, name='login.html')
```

Рис 10. Обработчик входа пользователя в аккаунт

Обработчик личного кабинета. Проверяется наличие JWT-токена; если его нет, то происходит переадресация пользователя на главную страницу. Если же он имеется, тогда декодируем токен и получаем информацию о почте пользователя и его уникальном идентификаторе. Обращаясь в базу данных, получаем список фильмов которые пользователь выбрал для отслеживания их появления в кинотеатрах. Пользователю возвращается шаблон Jinja, в который передано название шаблона lk.html и словарь context, содержащий данные для заполнения ими шаблона. Реализация представлена на рис. 11.

```python
@app.get('/lk')
def lk(request:Request):
    if (request.cookies.get('token')):
        lst = jwt.decode(request.cookies.get('token'), JWT_KEY, algorithms=["HS256"])

        email = lst['email']
        engine = database.message_engine
        meta = MetaData()
        meta.reflect(engine)
        user = meta.tables['users']
        with engine.connect() as connection:
            films = connection.execute(user.select().where(user.c.id == lst['id']).with_only_columns(user.c.film)).fetchall()
        #get info about planned sending emails
        return templates.TemplateResponse(request=request, name='lk.html', context={'email':email,'films':films, 'token':True})
    return RedirectResponse('/')
```

Рис. 11. Обработчик личного кабинета

Далее рассмотрим POST-обработчики.

Обработчик регистрации. Когда пользователь заполнил поля почты и пароля, он отправляет на сервер запрос. В обработчике проверяется уникальность пользователя, т.е. его отсутствие в базе данных пользователей. Если пользователь существует, обработчик возвращает сообщение о том, что такой пользователь уже зарегистрирован. Если такого пользователя не было найдено, он добавляется в базу данных, создается JWT-токен и добавляется в cookie. Пользователю возвращается главная страница. На рис. 12 показана реализация обработчика.

```
@app.post('/register')
async def register_user( email: Annotated[str, Form()], password: Annotated[str, Form()], request: Request):
    engine = database.user_engine
    meta = MetaData()
    meta.reflect(engine)
    users = meta.tables['user']
    with engine.connect() as connection:
        if(connection.execute(users.select().where(users.c.email == email)).fetchone()):
            print('пользователь уже существует')
            return templates.TemplateResponse(request=request,name='register.html', context={"bad":True})
        connection.execute(insert(users).values(email = email, hashed_password = bcrypt.hashpw(str.encode(password),bcrypt.gensalt())))
        connection.commit()
        user_id = connection.execute(users.select().where(users.c.email == email).with_only_columns(users.c.id)).fetchone()[0]

    #get JWT to user
    encoded_jwt = jwt.encode({"id": user_id, "email":email}, JWT_KEY, algorithm="HS256")
    #set JWT into cookie
    res = RedirectResponse('/', status_code= status.HTTP_302_FOUND)
    res.set_cookie(key="token",value=encoded_jwt, httponly=True)
    return res
```

Рис. 12. POST-обработчик регистрации

Обработчик входа в аккаунт. Когда пользователь ввел почту и пароль на странице с входом в аккаунт, на сервер отправляется запрос. Проверяется почта и хэш переданного пароля, и, если такой пользователь есть, создается JWT-токен, сохраняется в cookie и пользователю возвращается главная страница. Если же пользователь не был найден или были введены неправильные данные, то пользователю возвращается сообщение о некорректности переданных данных. На рис. 13 представлена реализация обработчика входа в аккаунт.

```
@app.post('/login')
async def login_user(email: Annotated[str, Form()], password: Annotated[str, Form()],request: Request):
    #validate user
    engine = database.user_engine
    meta = MetaData()
    meta.reflect(engine)
    users = meta.tables['user']
    with engine.connect() as connection:
        user_info = connection.execute(users.select().where(users.c.email == email).with_only_columns(users.c.hashed_password, users.c.id))
        if(user_info and bcrypt.checkpw(str.encode(password), user_info[0])):
            encoded_jwt = jwt.encode({"id": user_info[1], "email": email}, JWT_KEY, algorithm="HS256")
            res = RedirectResponse('/', status_code= status.HTTP_302_FOUND)
            res.set_cookie(key="token",value=encoded_jwt, httponly=True)
            return res
        return templates.TemplateResponse(request=request,name='login.html', context={"bad":True})
```

Рис. 13. POST-обработчик входа в аккаунт

Обработчик выхода из аккаунта выглядит довольно просто. Происходит удаление JWT-токена и пользователю возвращается главная страница. На рис. 14 показан данный обработчик.

```
@app.post('/logout')
async def logout_user():
    resp = RedirectResponse('/', status_code= status.HTTP_302_FOUND)
    resp.delete_cookie('token')
    return resp
```

Рис. 14. POST-обработчик выхода из аккаунта

Обработчик добавления фильма для отслеживания его появления в кинотеатрах. Пользователь вводит название фильма в форму и отправляет на сервер запрос. В обра-

ботчике происходит проверка: если пользователь ничего не ввел, возвращается страница личного кабинета, иначе декодируем JWT-токен и получаем почту и id пользователя, добавляем в базу данных для рассылки почту пользователя и название фильма, возвращаем страницу личного кабинета. На рис. 15 представлена реализация обработчика.

```
@app.post('/film/add_film')
async def add_film(request:Request, film: Annotated[str , Form()]= None):
    if film==None:
        return RedirectResponse('/lk',status_code= status.HTTP_302_FOUND)
    info = jwt.decode(request.cookies.get('token'), JWT_KEY, algorithms=["HS256"])
    id,email = info['id'],info['email']
    print(id,email)
    engine = database.message_engine
    meta = MetaData()
    meta.reflect(engine)
    user = meta.tables['users']
    with engine.connect() as connection:
        connection.execute(insert(user).values(id = id, email = email, film = film))
        connection.commit()
    return RedirectResponse('/lk',status_code= status.HTTP_302_FOUND)
```

Рис. 15. POST-обработчик добавления фильма для отслеживания его появления в кинотеатрах

8. Реализация рассылки уведомлений

Для рассылки уведомлений на почту в файле my_schedule реализован класс scheduler. Он имеет методы check_films и send_message. При создании экземпляра класса scheduler создается smtp-сервер и производится вход в почту, с которой будет осуществляться рассылка. На рис. 16 показан метод, отвечающий за инициализацию экземпляра.

```
def __init__(self):
    self.my_mail = MAIL
    self.mail_password = MAIL_PWD
    self.message_engine = create_engine('sqlite:///databases/send_message.db')
    self.film_engine = create_engine('sqlite:///databases/flims.db')
    self.smtp_server = smtplib.SMTP_SSL('smtp.mail.ru', 465)
    self.smtp_server.login(self.my_mail, self.mail_password)
```

Рис 16. Инициализация экземпляра класса scheduler

Метод check_films сравнивает фильмы, которые ожидают пользователи с фильмами в базе данных. Если такой фильм есть в базе данных, то вызывается метод send_message, который принимает как аргументы почту пользователя, фильм, который вышел в прокат и картинку фильма. Реализация метода check_films проиллюстрирована на рис. 17.

```
def check_films(self):
    meta_film = MetaData()
    meta_users = MetaData()
    meta_film.reflect(self.film_engine)
    meta_users.reflect(self.message_engine)
    film_table = meta_film.tables['film_table']
    users = meta_users.tables['users']
    with self.film_engine.connect() as connection:
        films = connection.execute(film_table.select().group_by(film_table.c.name).with_only_columns(film_table.c.name, film_table.c.img_url)
    with self.message_engine.connect() as connection:
        users = connection.execute(users.select().with_only_columns(users.c.email, users.c.film)).fetchall()
    for film in films:
        for user_info in users:
            cur_film = film[0][:-5].rstrip()
            if (SequenceMatcher(None,cur_film.lower(),user_info[1].lower()).ratio()>0.6):
                self.send_message(user_info[0],cur_film, film[1])
```

Рис. 17. Работа метода check_films

С помощью метода send_films происходит отправка сообщения на почту пользователя. На рис. 18 можно увидеть реализацию данного метода.

```
def send_message(self,email,film, img_url):
    msg = MIMEMultipart()
    msg["From"] = self.my_mail
    msg["To"] = email
    msg["Subject"] = "Рассылка Cinema Schedule"
    text = f"<h1>В прокате появился фильм {film}</h1> <img src='{img_url}' alt='{film}' style='width:400px'>"
    msg.attach(MIMEText(text,'html'))
    self.smtp_server.sendmail(self.my_mail, email, msg.as_string())
```

Рис. 18. Работа метода send_message

Заключение

В данной работе была представлена разработка веб-приложения "Cinema Schedule", которое позволяет пользователю получать актуальную информацию о фильмах в кинотеатрах Томска и Северск, расписание сеансов и информацию о ценах на билеты. Также приложение отслеживает интересующие пользователя фильмы и сообщает об их появлении в афише кинотеатров, посылая уведомление по почте.

Реализованное веб-приложение успешно протестировано. Полный код веб-приложения доступен по ссылке: https://github.com/CrazyF1re/CinemaSchedule-2.0.

ЛИТЕРАТУРА

1. Документация шаблонизатора Jinja2 [Электронный ресурс]: // URL: https://jinja.palletsprojects.com/en/3.1.x/ (дата обращения 05.05.2024).

2. Документация библиотеки FastAPI [Электронный ресурс]: // URL: https://fastapi.tiangolo.com/ (дата обращения 28.04.24)

3. Документация библиотеки SQLAlchemy [Электронный ресурс]: // URL: https://docs.sqlalchemy.org/en/20/ (дата обращения 26.04.24).

WEB-ПРИЛОЖЕНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ РАБОТЫ С РЕДКИМИ КОЛЛЕКЦИЯМИ КНИГ В НБ ТГУ

Пурыскин Р.Д., Щербина Д.А., Самохина С.И., Ивановская Е.В.

Томский государственный университет

rpuryskin@mail.ru, sv.sam.tsk@gmail.com

Введение

Научная библиотека Томского государственного университета – одна из старейших вузовских библиотек Сибири и Дальнего Востока. Её фонд формировался вместе с возведением зданий Томского государственного университета. Многие ученые, профессоры российских университетов, общественные и политические деятели, различные организации приняли решение передать свои библиотеки или книжные коллекции в дар Томскому университету. За долгие годы существования сюда поступило более 200 книжных собраний. В данный момент фонд НБ ТГУ имеет научную, историческую и культурную значимость, обеспечивает учебную и научную деятельность факультетов и научных подразделений университета. Документы широко используются исследователями. Разнообразие и уникальность материалов представляет собой неисчерпаемый ресурс для просветительской и образовательной деятельности.

В последние несколько десятилетий в библиотеке университета ведется большая работа по выявлению, сохранению и учету книжных памятников [1–4]. С каждым годом массив материалов, накопленный сотрудниками НБ ТГУ, увеличивается, и появилась потребность в создании программы, которая позволила бы объединять в себе данные, собранные разными сотрудниками, по разным личным библиотекам и книжным коллекциям.

Цель данной работы – создание информационной системы для работы с тематическими коллекциями и личными библиотеками НБ ТГУ. В задачи входит создание Web-приложения, которое объединяет в себе клиентскую (пользовательский интерфейс) и серверную части информационной системы.

Для создания полноценного Web-приложения необходимо реализовать серверную часть, где будут храниться данные (backend), и клиентскую часть с интерфейсом (frontend) [5]. Для создания серверной части необходимо

1) проанализировать существующие решения,
2) спроектировать информационную систему,
3) реализовать серверную логику.

Для создания клиентской части необходимо:

1) провести обзор существующих инструментов и информационных технологий для создания Web-приложений,
2) определить структуру разрабатываемого Web-приложения,
3) реализовать основной функционал.

1. Выбранный инструментарий разработки

Для реализации проекта были использованы:

– PostgreSQL – СУБД для хранения данных о книгах [6],
– Java 17 (фреймворк Spring) – создание информационной системы,
– Visual Studio Code – IDE для разработки Web-приложения,
– JavaScript / TypeScript – создание пользовательского интерфейса [7],
– React JS – фреймворк для разработки пользовательского интерфейса [8].

2. Архитектура и технические особенности системы

Для реализации информационной системы выбрана классическая архитектура клиент-сервер (рис. 1).

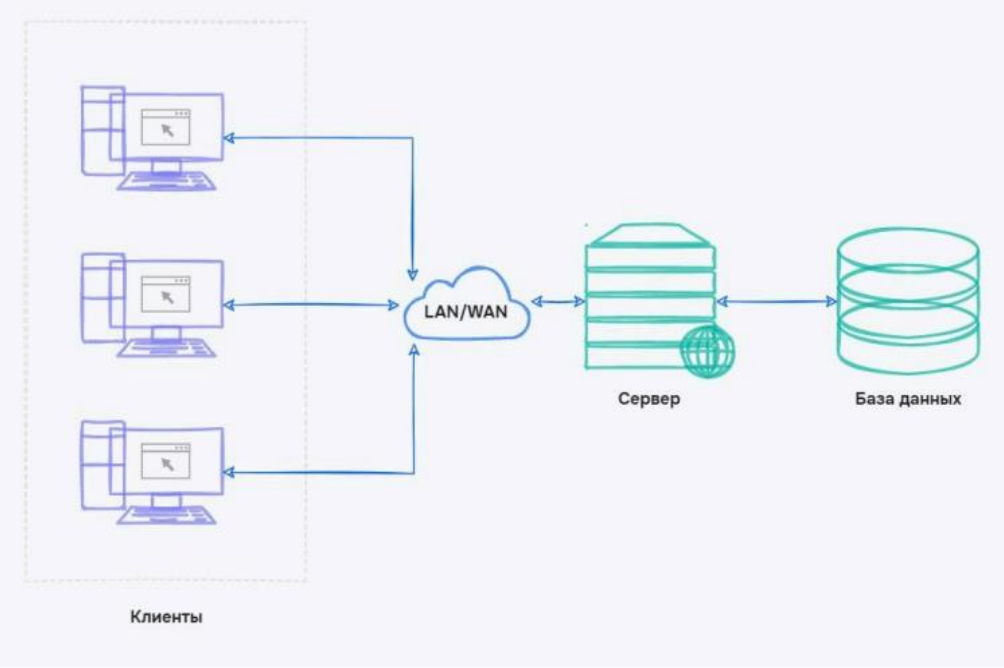


Рис. 1. Схема клиент-серверной архитектуры

Клиент-серверная архитектура веб-приложений характеризуется разделением ответственности: клиентская сторона отвечает за визуализацию данных и взаимодействие

с пользователем, а серверная сторона занимается обработкой логики, данных и их хранением. Это упрощает разработку и поддержку, а также облегчает масштабируемость, поскольку сервер можно расширять независимо от клиента. Централизованное управление данными на сервере упрощает их хранение, обеспечивает безопасность и резервное копирование. Благодаря универсальности доступа, пользователи могут подключить его к веб-приложению с любого устройства, где есть выход в интернет. Облегченная клиентская часть позволяет устройствам быстрее загружать приложение, т.к. большая часть обработки данных происходит именно на сервере, а не на самом устройстве. Обновления и поддержка происходят также через сервер, поэтому новые функции становятся доступными для пользователей без нужды обновлять клиентскую часть.

При проектировании системы использовалась архитектурная модель REST [9]. REST (Representational State Transfer) – модель взаимодействия компонентов распределенного приложения в сети. Она предоставляет ряд преимуществ, которые делают ее отличным выбором для современных веб-приложений.

3. Создание базы данных

Для информационной системы хранения данных о книжных коллекциях спроектирована основная часть базы данных (схема представлена на рис. 2).

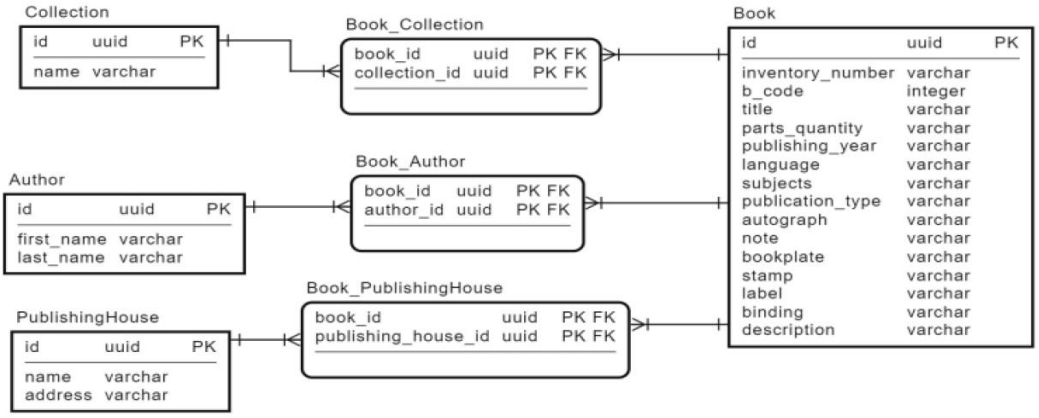


Рис. 2. Схема таблиц в базе данных

В рамках системы ключевым аспектом является реализация функционала по отслеживанию истории модификаций объектов. Для реализации спроектирована часть базы данных для хранения истории (рис. 3).

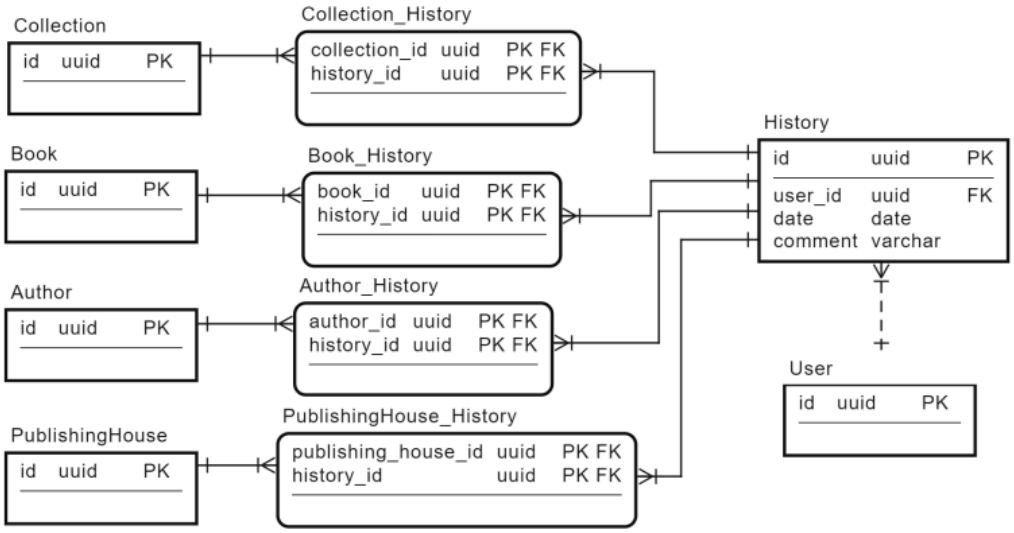


Рис. 3. Схема таблиц в базе данных

Для предотвращения проблем с расширением базы данных или ее изменениями использована система Liquidbase.

4. Описание функционала Web-приложения

Создаваемое web-приложение должно предоставлять возможность работы с тремя сущностями: **коллекции**, **книги**, **авторы**.

Администратор может добавлять в базу данных новые коллекции. В эти коллекции можно будет помещать новые книги, к которым можно будет привязывать существующих авторов или добавлять новых. Функционал приложения для администратора проиллюстрирован на рис. 4.

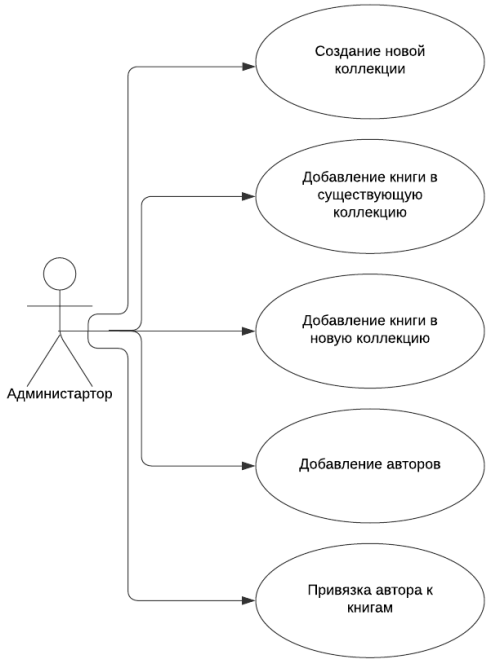


Рис. 4. Функционал Web-приложения для администратора

5. Клиентская часть Web-приложения

На главной странице веб-приложения располагается список коллекций, которые уже имеются на сервере, а в шапке страницы расположен основной функционал по добавлению новых экземпляров сущностей (коллекции, книги, авторы).

Рис. 5 отображает вид главной страницы.

Рис. 5. Главная страница приложения

Форма для добавления новой коллекции будет появляться с помощью плавной анимации при нажатии на кнопку с соответствующим функционалом (рис. 6).

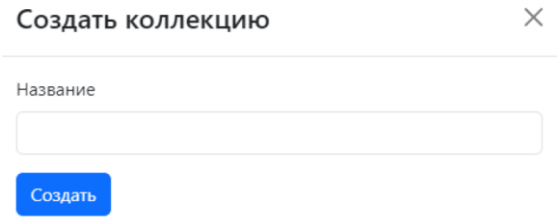


Рис. 6. Форма для создания новой коллекции

Форма для добавления нового автора будет появляться с помощью плавной анимации при нажатии на кнопку с соответствующим функционалом (рис. 7).

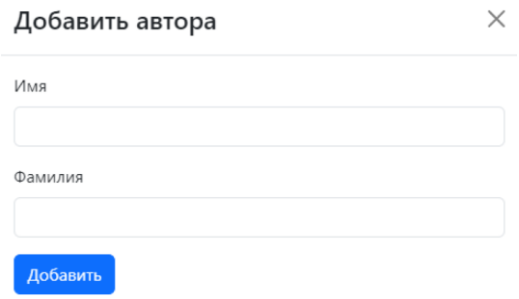


Рис. 7. Форма для создания новой коллекции

На рис. 8 показана форма для добавления новой книги, которая появляется после нажатия кнопки в шапке странице с соответствующим функционалом.

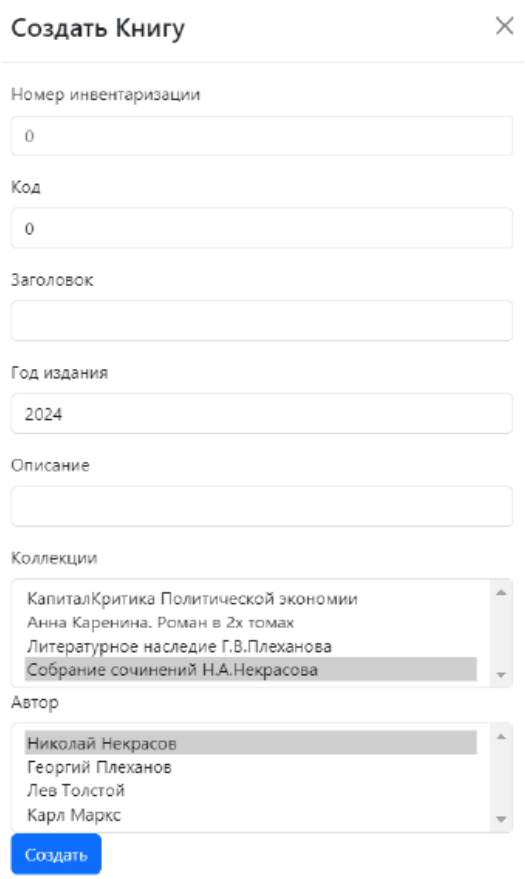


Рис. 8. Форма для добавления новой книги

Также обеспечена возможность выбрать, в какую из существующих коллекций добавить эту книгу, а также выбрать автора для нее из предложенного списка.

При добавлении новых коллекций они будут располагаться списком на главной странице, а новые книги появляются в выпадающем списке соответствующих коллекций, что показано на рис. 9.

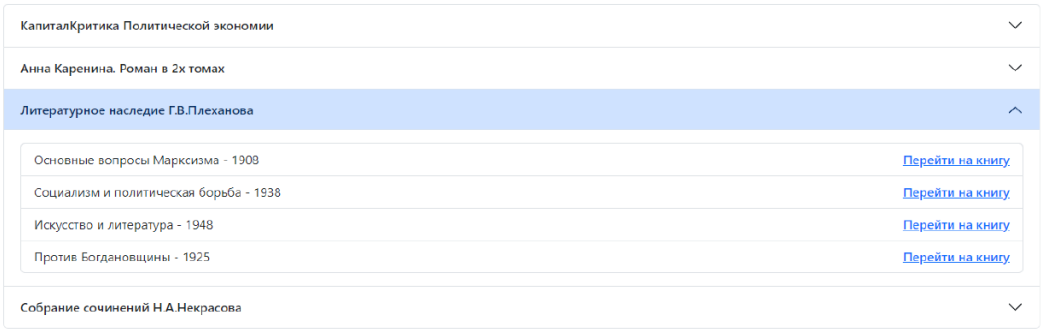


Рис. 9. Добавление новых экземпляров

Заключение

В ходе работы была выполнена задача создания информационной системы для работы с коллекциями редких книг в НБ ТГУ.

Для серверной части разработана система на базе СУБД PostgreSQL с использованием современных технологий безопасности. Результаты данной работы могут быть применены сотрудниками научных библиотек, обеспечивая им более качественный и безопасный доступ к коллекциям редких книг.

Для клиентской части разработки проведен обзор существующих инструментов и информационных технологий для создания Web-приложений, определена структура и дизайн веб-приложения, реализован основной функционал клиентской части веб-приложения, позволяющий пользователю удобно и наглядно добавлять и обрабатывать данные, которые будут храниться на сервере.

В перспективе разработки – реализация более детального функционала, возможности добавления новых атрибутов при добавлении книг в форме, добавления возможности входа в приложение по уникальным логину и паролю для администраторов для обеспечения большей безопасности хранения данных.

ЛИТЕРАТУРА

1. *Ивановская Е.В.* Личные книжные собрания в Отделе основного фонда Научной библиотеки Томского государственного университета (проблема выявления, сохранности и учета) // Междисциплинарная научно-практическая конференция "Книжные памятники в аспекте сохранности". – М., 2020. – С. 47–54.
2. *Ивановская Е.В.* Методы раскрытия основного фонда Научной библиотеки Томского государственного университета // Одиннадцатые Макушинские чтения : материалы научной конференции (Томск, 29– мая 2018 г.). – Новосибирск, 2018. – С. 107–114.
3. *Ивановская Е.В.* Книги из личной библиотеки первого директора Томского политехнического университета в фонде Научной библиотеки Томского государственного университета // Вестник Томского государственного университета. Культурология и искусствоведение. – 2021. – № 44. – С. 314–322.
4. *Ивановская Е.В., Гончарова Н.В.* К проблеме формирования книжных памятников-коллекций Томской области в фонде Научной библиотеки Томского университета // Двенадцатые Макушинские чтения : материалы международной научной конференции (Тюмень, 25–27 мая 2021 г.). – Новосибирск, 2021. – С. 5–14.
5. *Дронов В.А.* Разработка современных Web-сайтов. – СПб.: БХВ-Петербург, 2013.
6. Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом. URL: https://habr.com/ru/post/282764/
7. Документация по TypeScript. URL: https://metanit.com/web/typescript/
8. Использование фреймворка React в своих проектах. URL:https://ru.legacy.reactjs.org/docs/getting-started.html
9. Архитектура REST: URL: https://habr.com/ru/articles/38730/

АВТОМАТНЫЕ МОДЕЛИ СИММЕТРИЧНЫХ КРИПТОСИСТЕМ

Рудай Д.В., Тренькаев В.Н.

Томский государственный университет

zzzing5@yandex, trenkaev.vn@mail.ru

Введение

Теоретико-автоматный подход является одним из направлений при исследовании свойств криптографических систем. Автоматная модель представлена в трудах таких ученых как Агибалов Г.П. [1], Бабаш А.В. [2,3], Кудияров Д.С. [4], Пудовкина М.А. [5], Тренькаев В.Н. [6] и др. При построении криптосистем конечные автоматы могут выступать в качестве отдельного компонента, например, комбайнера [7] или элемента генератора ключевого потока [8], либо в виде основной функциональной подсистемы, как в шифре Закревского А.Д. [1]. Поведение криптосистем может описываться автоматными сети, как в [4], где генератор ключевого потока (ГКП) представлен в виде последовательно соединенных автономного и неавтономного автоматов.

В теории автоматных криптосистем рассматриваются особые классы автоматов (т.н. обратимые автоматы), для которых всегда разрешима задача определения входно-

го слова автомата по известному начальному состоянию и выходному слову. Также представлены линейные векторные автоматы, перестановочные автоматы, полноцикловые автоматы и др. [2,3]. Решаются задачи, связанные с запретами автоматов, оценкой периодов выходных слов автоматов, определению входного слова автомата по известному выходному слову и пр. Получены результаты разного рода, в частности, установлена функциональная эквивалентность классов поточных и автоматных шифрсистем.

Однако исследователи не придерживаются единого стиля изложения своих результатов. Отсутствие единой терминологии затрудняет изучение материала. В качестве базового класса, который моделирует симметричные криптосистемы, разные авторы рассматривают разного вида автоматы: шифрующие автоматы, криптоавтоматы, перестраиваемые автоматы.

В данной работе проводится анализ базовых понятий теории автоматных криптосистем, дается краткий обзор некоторых известных результатов по использованию автоматной модели для проектирования и анализа симметричных криптосистем, а также представлена теоретико-автоматная модель шифра RC4.

1. Необходимые определения

Определение 1.1. *Конечным автоматом* A называется пятёрка (X,S,Y,ψ,ϕ), где S – конечное непустое множество состояний, X и Y – конечные входной и выходной алфавиты соответственно, $\psi: X\times S\to S$ и $\phi: X\times S\to Y$ – функции переходов и выходов соответственно.

Четвёрку $s-x/y\to s`$, где $s`=\psi(x,s)$ и $y=\phi(x,s)$, называют *переходом* автомата A. Говорят, что входное слово $x_1x_2\ldots x_n\in X^*$ *переводит* автомат A из состояния s в состояние $s`$ с выдачей выходного слова $y_1y_2\ldots y_n\in Y^*$, если существует последовательность переходов $s=s_1-x_1/y_1\to s_2$, $s_2-x_2/y_2\to s_3$, …, $s_n-x_n/y_n\to s_{n+1}=s`$.

Автомат A при фиксированном состоянии s реализует алфавитное отображение $f_s: X^*\to Y^*$, для которого $f_s(x_1x_2\ldots x_n)=y_1y_2\ldots y_n$.

Определение 1.2. Автомат A *обратим*, если при любом состоянии s для отображения f_s существует обратное отображение f_s^{-1}.

Определение 1.3. Автомат $A^{-1}=(Y,S,X,\psi_0,\phi_0)$ называется *обратным* к автомату $A=(X,S,Y,\psi,\phi)$, реализующему $\{f_s: s\in S\}$, если A^{-1} реализует $\{f_s^{-1}: s\in S\}$.

Определим т.н. частичные функции переходов $(\psi_x)_{x\in X}$ и выходов $(\phi_x)_{x\in X}$ через ψ и ϕ следующим образом: $\psi_x: S\to S$, $\psi_x(s)=\psi(x,s)$; $\phi_x: S\to Y$, $\phi_x(s)=\phi(x,s)$.

Определение 1.4. Автомат A называют *перестановочным*, если его частичные функции переходов $(\psi_x)_{x\in X}$ осуществляют взаимно однозначные отображения S в S.

Определение 1.5. Автомат A называется *векторным*, если $S=V_n$ – векторное пространство размерности n над полем F_q из q элементов, а $Y=F_q$. Векторный автомат называется *линейным*, если $(\psi_x)_{x\in X}$, $(\phi_x)_{x\in X}$ – линейные отображения.

2. Шифрующий автомат

Бабаш А.В. в своих трудах [2,3] представляет теорию синтеза и анализа шифрующих автоматов. Изучаются различные типы (перестановочные, линейные векторные, полноцикловые и др.) и характеристики (периодичность слов автомата и др.) шифрующих автоматов. Решаются задачи, связанные с неотличимостью состояний, запретами автоматов, оценкой периодов выходных слов автоматов, определению входного слова автомата по известному выходному слову и др.

Бабаш А.В. приводит несколько определений понятия шифрующего автомата. Например, автомат называется *шифрующим*, если для любого состояния s он реализует *инъективное* алфавитное отображение f_s.

Далее приведено наиболее общее определение шифрующего автомата по Бабашу.

Определение 2.1. Пусть K – конечное множество ключей. Автомат $(X,S\times K,Y,h,f)$, где $h: S\times K\times X\to S\times K$ и $f: S\times K\times X\to Y$, называется *шифрующим*. При этом $h=(h_1,h_2)$, т.е.

$h(s,k,x)=(h_1(s,k,x),h_2(s,k,x))$, где $h_1: S\times K\times X\to S$ – функция переходов и $h_2: S\times K\times X\to K$ – функция обновления ключа. Также есть функция инициализации $z: K\to S$.

Пудовкина М.А. в своей диссертации [5] приводит примеры описания различных поточных шифров с помощью автоматной модели, в частности, RC4, GI, Solitaire, Веста. В диссертации приведены автономные автоматы, которые моделируют поточный шифр RC4 и алгоритмы поточного шифрования семейства Веста. Введено семейство алгоритмов поточного шифрования GI, в результате чего известные алгоритмы IA, IBAA, ISAAC представлены в общей теоретико-автоматной модели.

Рассмотрим определение шифрующего автомата по Пудовкиной.

Определение 2.2. Пусть $g: X\times K\times S\to S$, $f: X\times K\times S\to Y$ – функции, где $g((x,k),s)=g_{(x,k)}(s)$, $f(x,(k,s))=f_{(k,s)}(x)$. Автомат $(X\times K,S,Y,g,f)$ называется *шифрующим*, если частичная функция $g_{(x,k)}: S\to S$ – биекция для любых пар $(x,k)\in X\times K$, а также при фиксированном ключе k и состоянии s отображение $f_{(k,s)}: X\to Y$ является инъективным.

3. Криптоавтомат

Агибалов Г.П. в своей работе [1] вводит понятие криптоавтомата, приводит многочисленные примеры использования конечных автоматов и криптоавтоматов в криптографии, описывает конечно-автоматные симметричные шифры и криптосистемы с открытым ключом, демонстрирует эквивалентность поточных и автоматных шифрсистем.

Определение 3.1. *Криптоавтомат* – это набор из шести объектов $C=(X,S,Y,K,\psi,\phi)$, в котором X, S, Y, K – конечные множества, $K=S_0\times K_0$, $S_0\subseteq S$, ψ, ϕ – функции, $\psi: K_0\times X\times S\to S$ и $\phi: K_0\times X\times S\to Y$. Элементы в множестве K называются ключами.

В произвольном ключе $k=(s_0,k_0)\in K$ компонента k_0 выступает в роли параметра, выделяющего в криптоавтомате C автомат $C_k=(X,S,Y,\psi_k,\phi_k)$, где функции $\psi_k: X\times S\to S$ и $\phi_k: X\times S\to Y$ для любых $x\in X$ и $s\in S$ определяются как $\psi_k(x,s)=\psi(k_0,x,s)$ и $\phi_k(x,s)=\phi(k_0,x,s)$ и называются соответственно функциями переходов и выходов криптоавтомата C на ключе k. Автомат C_k называется *проекцией криптоавтомата C на ключе k*. Компонента s_0 в ключе k используется в качестве начального состояния автомата C_k.

Автономный криптоавтомат определяется аналогично автономному автомату, т.е. как $C=(S,Y,S_0\times K_0,\psi,\phi)$, где $\psi: K_0\times S\to S$ и $\phi: K_0\times S\to Y$, $\psi_k: S\to S$ и $\phi_k: S\to Y$ и $\psi_k(s)=\psi(k_0,s)$ и $\phi_k(s)=\phi(k_0,s)$ для любых $k=(s_0,k_0)\in K$ и $s\in S$.

Стоит отметить, что ГКП в поточных шифрах может быть описан некоторым автономным криптоавтоматом. Примером синхронного конечно-автоматного ГКП может служить генератор MUGI [9].

Рассмотрим определение автоматной шифрсистемы, данное Агибаловым, на основе понятия "криптоавтомат". Обозначим через $C\ f_s^k$ алфавитное отображение $f_s: X^*\to Y^*$ при состоянии s для проекции криптоавтомата C на ключе k.

Определение 3.2. *Автоматная шифрсистема* – это пятерка объектов $<X,Y,K,E,D>$, где $E=(X,S,Y,K,\psi,\phi)$ и $D=(Y,S^*,X,K,\psi^*,\phi^*)$ – свободно инициализируемые криптоавтоматы, называемые автоматами шифрования и расшифрования соответственно, если любого состояния $s\in S$ найдется такое состояние $s^*\in S^*$, что выполняется

$$\forall s\in S\ \exists s^*\in S^*\ \forall k\in K\ \forall\alpha\in X^*\ \forall\beta\in Y^*\left[E\ f_s^k(\alpha)=\beta\Rightarrow D\ f_{s^*}^k(\beta)=\alpha\right].$$

4. Перестраиваемый автомат

Для задания автоматной шифрсистемы можно использовать перестраиваемый автомат, предложенный Тренькаевым В.Н. [6]. Неформально под перестраиваемым понимается автомат с возможностью "настройки" функции переходов. Конкретный вариант настройки (вместе с начальным состоянием) является ключом шифрсистемы. Показано [6], что мощность ключевого множества может быть достаточно большим, чтобы противостоять атаке грубой силы на автоматную шифрсистему, построенную на базе перестраиваемого автомата. Также показано, что такая автоматная шифрсистема не-

стойка к атаке на основе выбранного открытого текста, когда криптоаналитик знает начальное состояние и имеет несколько экземпляров шифратора.

Определение 4.1. Пусть даны две функции переходов ψ_0: $X\times S\to S$ и ψ_1: $X\times S\to S$ и функция настройки π: $K\times X\times S\to\{0,1\}$. *Перестраиваемый автомат* – это восьмерка объектов $(X,S,Y,K,\psi_0,\psi_1,\pi,\phi)$, где $\phi(x,s)=\phi_s(x)$ – биекция для любого состояния s. Фиксируя $k\in K$, получаем автомат $A_k=(X,S,Y,\psi_k,\phi)$, у которого функция переходов ψ_k определяется по следующему правилу: если $\pi_k(x,s)=0$, то $\psi_k(x,s)=\psi_0(x,s)$, иначе $\psi_k(x,s)=\psi_1(x,s)$.

Таким образом, выбор ключа k однозначно определяет функцию настройки $\pi_k(x,s)$, а, следовательно, и функцию переходов ψ_k. Отметим, что секретная функция $\pi_k(x,s)$ может быть задана с помощью булевого вектора длины $|X|x|S|$ и при $|X|=|S|=10$ мы имеем 2^{100} ключей, что достаточно, чтобы противостоять атаке грубой силы.

Стоит отметить, что перестраиваемый автомат задает множество обратимых автоматов $\{A_k=(X,S,Y,\psi_k,\phi)\colon k\in K\}$ такое, что у каждого автомата A_k из этого множества существует обратный к нему автомат A_k^{-1}, причем выполняется следующее свойство: если для некоторого k автомат A_k при фиксированном состоянии s преобразует открытый текст $\alpha\in X^*$ в шифрованный текст $\beta\in Y^*$, т.е. реализует $f_s(\alpha)=\beta$, то автомат A_k^{-1} реализует $f_s^{-1}(\beta)=\alpha$.

5. Теоретико-автоматная модель RC4

Кудиярова Д.С. в своей диссертации [4] рассматривает задачу построения теоретико-автоматной модели ГКП RC4 и оценки с ее помощью периода выходной последовательности и состояний RC4. Показано, что ГКП RC4 может быть представлен в виде последовательного соединения автономного и неавтономного автоматов. Далее приведено описание алгоритма RC4 и его теоретико-автоматная модель.

Алгоритм RC4 [10], предложенный Ривестом в 1994 г., представляет собой быстрый способ генерации псевдослучайных чисел. Алгоритм работает с n-битовыми словами (обычно n=8). Вычисления проводятся по модулю $2n$. На базе L-словного ключа $K=K_0K_1...K_{L-1}$ генерируется последовательность слов $z_1z_2z_3...$. Генерация очередного псевдослучайного слова z_i осуществляется на базе значений элементов массива S из $2n$ слов и двух переменных i и j. Массив S содержит все возможные n-битовые числа в перемешанном виде. Значение элемента массива S можно трактовать либо как число, либо как номер другого элемента массива.

Ключ K задает начальное перемешивание чисел в массиве S, которое формируется с помощью следующего алгоритма:

$j \leftarrow 0$, $S \leftarrow (0,1,...,2^n-1)$
FOR i=0,1,...,2^n-1 DO
  $j \leftarrow (j+S_i+K_i \bmod L) \bmod 2^n$
  $S_j \leftrightarrow S_i$
$i \leftarrow 0$, $j \leftarrow 0$

Генерация псевдослучайного слова z_i производиться с помощью следующего алгоритма:

$i \leftarrow (i+1) \bmod 2^n$
$j \leftarrow (j+S_i) \bmod 2^n$
$S_j \leftrightarrow S^i$
$t \leftarrow (S_i+S_j) \bmod 2^n$
$z_i \leftarrow S_t$

Пример. Пусть n=3, K=25 (L=2). Сформируем начальную перестановку чисел в S:
j=0, S =(0,1,2,3,4,5,6,7),
i=0, j=0+0+2=2, S=(2,1,0,3,4,5,6,7),
i=1, j=2+1+5=0, S=(1,2,0,3,4,5,6,7),
i=2, j=0+0+2=2, S=(1,2,0,3,4,5,6,7),

i=3, j=2+3+5=2, S=(1,2,3,0,4,5,6,7),
i=4, j=2+4+2=0, S=(4,2,3,0,1,5,6,7),
i=5, j=0+5+5=2, S=(4,2,5,0,1,3,6,7),
i=6, j=2+6+2=2, S=(4,2,6,0,1,3,5,7),
i=7, j=2+7+5=6, S=(4,2,6,0,1,3,7,5).

Теперь вычислим несколько элементов псевдослучайной последовательности:
i=1, j=0+2=2, S=(4,6,2,0,1,3,7,5), t=2+6=0, z_1=4,
i=2, j=2+2=4, S=(4,6,1,0,2,3,7,5), t=1+2=3, z_2=0,
i=3, j=4+0=4, S=(4,6,1,2,0,3,7,5), t=2+0=2, z_3=1,
i=4, j=4+0=4, S=(4,6,1,2,0,3,7,5), t=0+0=0, z_4=4,
i=5, j=4+3=7, S=(4,6,1,2,0,5,7,3), t=5+3=0, z_5=4,
i=6, j=7+7=6, S=(4,6,1,2,0,5,7,3), t=7+7=6, z_6=7.

В итоге, записав числа в двоичном виде, получаем последовательность:

1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 1 1 . . .

Теоретико-автоматная модель RC4

Пусть $\mathbb{Z}_N$ – кольцо вычетов по модулю N, S_N – симметрическая группа степени N, $s(x)$ – переход в подстановке s под номером x, (x,y) – транспозиция двух переходов подстановки под номерами x и y.

Определим два автомата. Пусть $A=(S,Y,\psi,\phi)$ – автономный автомат, у которого $S=Y=\mathbb{Z}_N$, ψ: $\mathbb{Z}_N\to\mathbb{Z}_N$ – функция переходов, причем $\psi(i)=i\boxplus 1$, где $\boxplus$ – операция сложения по модулю N в кольце вычетов $\mathbb{Z}_N$, ϕ: $\mathbb{Z}_N\to\mathbb{Z}_N$– функция выхода, причем $\phi(i)=i\boxplus 1$, $i\in\mathbb{Z}_N$. Пусть автомат $B=(\Sigma,X,Y,(h_x)_{x\in X},(f_x)_{x\in X})$ – неавтономный автомат, где $\Sigma=\mathbb{Z}_N\times S_N$ – множество состояний, $X=\mathbb{Z}_N$ – входной алфавит, $Y=\mathbb{Z}_N$ – выходной алфавит, $(h_x\colon \Sigma\to\Sigma)_{x\in X}$ – частичные функции переходов и $(f_x\colon \Sigma\to Y)_{x\in X}$ – частичные функции выходов, для которых выполняется

$$h_x(j,s)=(j\boxplus s(x)),s\circ(s(x),s(j\boxplus s(x))),\ f_x(j,s)=s(s(x)\boxplus s(j)),\ \text{где } s\in S_N,\ j\in\mathbb{Z}_N.$$

В [4] показано, что выходная последовательность ГКП RC4 представляет собой выходную последовательность автомата B с состояния $(0,s_0)$, где s_0 – произвольная начальная подстановка при входной последовательности Q=1,2,…,N−1,0,1,2… .

Также в [4] доказано, что выходная последовательность ГКП RC4 и последовательность подстановок в его внутренних состояниях являются чисто периодическими.

Определение 5.1. Последовательность $b_1,b_2,\ldots,b_t,\ldots$ элементов некоторого алфавита называется *чисто периодической*, если найдется натуральное число d, при котором $b_t=b_{t+kd}$ для любых натуральных чисел t, k. Минимальное число d с указанным свойством называется *периодом* последовательности.

6. Сравнительный анализ

В табл. 1 представлен сравнительный анализ рассмотренных выше автоматных моделей симметричных криптосистем. Как видно из таблицы, наиболее общей моделью является криптоавтомат или шифрующий автомат по Бабашу.

Стоит отметить, что ограничения, которые накладываются на функцию переходов и функцию выходов в случае шифрующего автомата по Пудовкиной или в случае перестраиваемого автомата, позволяют адаптировать используемые теоретико-автоматные модели к практике.

Таблица 1

Сравнительный анализ автоматных моделей симметричных криптосистем

	Шифрующий автомат (б – по Бабашу, п – по Пудовкиной)	**Криптоавтомат**	**Перестраиваемый автомат**
Функция выходов зависит от ключа	да	да	нет
Функция переходов	да	да	да

зависит от ключа			
Частичная функция переходов: $S \to S$	нет ограничений (б) биекция (п)	нет ограничений	нет ограничений
Частичная функция выходов: $X \to Y$	нет ограничений (б) инъекция (п)	нет ограничений	биекция
Особенности	есть функция обновления ключа (б)	на базе криптоавтомата дано определение автоматной шифрсистемы	есть функция настройки, функция переходов строится на базе двух заданных функций

Заключение

В данной работе в едином стиле рассмотрены базовые понятия теории автоматных криптосистем от разных исследователей. Представлены три варианта использования теоретико-автоматной модели для описания симметричных криптосистем: шифрующий автомат, криптоавтомат, перестраиваемый автомат. Данные варианты имеют явные общие характеристики, например, все рассмотренные автоматы строятся на добавлении в традиционную модель автомата ключевого множества, а также и некоторые уникальные различия, например, в шифрующем автомате по Бабашу имеется функция обновления ключа, а в перестраиваемом автомате имеется функция настройки, которая позволяет "собрать" функцию переходов из двух заданных функций. Показана эффективность использования автоматной модели для анализа и синтеза симметричных криптосистем, в частности для анализа шифра RC4.

ЛИТЕРАТУРА

1. *Агибалов Г.П.* Конечные автоматы в криптографии // Прикладная дискретная математика. Приложение. – 2009. – № 2. – С. 43–73

2. *Бабаш А.В.* Теория автоматов. Синтез шифрующих автоматов: учебное пособие. – Москва : ФГБОУ ВО "РЭУ им. Г.В. Плеханова", 2020. – 348 с.

3. *Бабаш А.В.* Теория автоматов. Анализ шифрующих автоматов: учебное пособие. – Москва : ФГБОУ ВО "РЭУ им. Г.В. Плеханова", 2021. – 296 с.

4. *Кудияров Д.С.* Защита от угроз конфиденциальности аутентификационным данным в системах хранения, основанных на ГПСЧ RC4: диссертация на соискание ученой степени канд. тех. наук // Институт проблем управления им. В.А. Трапезникова. – Москва, 2019. – 157 с.

5. *Пудовкина М.А.* Свойства программно реализуемых поточных шифров: на примере RC4, GI, Веста: диссертация на соискание ученой степени канд. физ.-мат. наук // Московский государственный инженерно-физический институт. – Москва, 2004. – 151 с.

6. *Тренькаев В.Н.* Реализация шифра Закревского на основе перестраиваемого автомата // Прикладная дискретная математика. – 2010. – № 3(9). – С. 69–76.

7. *O'Neil S., Gittins B., Landman H.* VEST Hardware-Dedicated Stream Cipher // IACR Cryptology ePrint Archive. – 2005. – 413 p.

8. *Ekdahl P., Johansson T., Maximov A., Yang J.* A new SNOW stream cipher called SNOW-V // IACR Transactions on Symmetric Cryptology. – 2019 (3). – P. 1–42.

9. *Watanabe D., Furuya S., Yoshida H., Takaragi K., Preneel B.* A New Keystream Generator MUGI // LNCS. – 2002. – No. 2365. – P. 179–194.

10. *Рябко Б.Я., Фионов А.Н.* Криптографические методы защиты информации: учебное пособие. – М.: Горячая линия. Телеком, 2005. – 229 с.

ПРИЛОЖЕНИЕ ДЛЯ АНАЛИЗА ТОЧНОСТИ ПРОГНОЗА ПОГОДЫ

Савоськин Н.А., Пахомова Е.Г.

Томский государственный университет

sav-n-an@mail.ru, pakhomovaeg@yandex.ru

Введение

Представим себе ситуацию: во время отпуска или на выходных Вы решили съездить на природу. Выбираете время, место, смотрите прогноз погоды на это время. Вроде-бы, всё замечательно: солнышко, ясно. Однако, когда вы приехали на место, оказывается, что там ливень, и идти он будет ещё трое суток.

Для того, чтобы избежать подобной ситуации, нужно понять, какому сайту можно доверять, чьи прогнозы оказываются наиболее точными.

Как правило, мы ориентируемся на тот сайт прогноза погоды, который при прошлом нашем обращении к нему дал более точный результат, т.е. мы ориентируемся на наш жизненный опыт. Но тот же жизненный опыт подсказывает, что сайт, который некоторое время назад давал очень точный прогноз погоды, может вдруг снизить качество прогноза.

Очевидно, что выходом из такой ситуации является непрерывный мониторинг точности прогноза погоды сайта.

Целью данной работы было создание приложения, способного анализировать точность погоды с некоторых сайтов, а также выводить прогноз на оставшийся день.

1. Стек технологий

Для разработки приложения был использован язык Python [1], т.к. парсинг на этом языке значительно проще и удобнее. Для обработки полученных данных была использована система управления реляционными базами данных PostgreSQL, потому что её функционал шире и работать с ней проще относительно её аналогов. Также были использованы библиотеки:

- **Psycopg2** – с помощью данной библиотеки мы можем связать наше приложение и базу данных, настроив между ними обмен данными [2];
- **requests** – наиболее популярная библиотека для получения данных с вебстаницы [3];
- **BeautifulSoup** – используется для поиска данных в HTML-странице [4];
- **datetime** – библиотека используются для работы с таким типом данных, как "время" [5];
- **sys** и **os** – библиотеки, которые обычно используются в паре; с помощью первой – получаем информацию об операционной системе, с помощью второй – взаимодействуем с ней [6,7];
- **re**– библиотека, работающая с регулярными выражениями [8];
- **shedule** – библиотека, которая позволяет запускать выбранные функции в определённое время или с определённым промежутком [9];
- **PyQt6** – библиотека, которая адаптирует язык Qt к Python; используется для реализации интерфейса [10].

2. Архитектура приложения

Суть программного продукта заключается в следующем. Было выбрано три сайта прогноза погоды с дружественным API: rp5, GISMETEO и ПогодаMail.ru. Прогнозы этих сайтов и будут анализироваться в приложении. С помощью библиотеки requests будем делать GET-запросы на сайт и получать HTML-страницу, после чего с помощью библиотеки BeautifulSoup и регулярных выражений, обрабатываемых в библиотеке re, будем доставать из полученной страницы информацию о прогнозах погоды и записывать в базу данных, используя Psycopg2. В базе данных информация обрабатываться, формируется точность. Далее она извлекается из базы данных и показывается пользователю.

Архитектуру программы можно разбить на 3 блока: 1) блок подсчёта точности, 2) блок парсинга погоды, 3) блок интерфейса.

Блок подсчёта точности. В данном блоке формируется коэффициент точности для выбранных сайтов. Для этого используя библиотеки BeautifulSoup, requests, re, shedule. Получаем в 00:00 прогнозы с сайтов на время 19:00, временно заносим их в папку с помощью библиотек sys и os, затем в 19:00 получаем фактическое значение погоды, достаём прогнозы из папки и одним кортежем отправляем в таблицу Predicts (рис. 1).

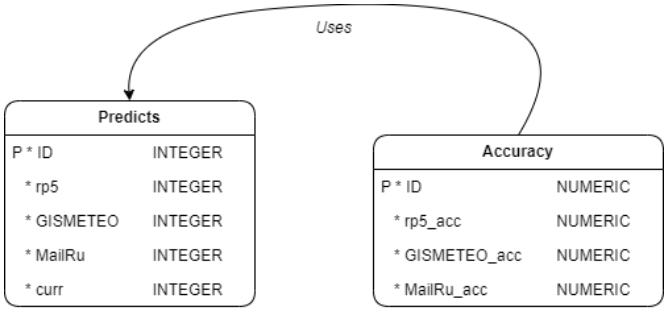


Рис. 1. Схема базы данных

Таблица Accuracy содержит в себе коэффициенты точности каждого сайта, которые с помощью trigger-функции пересчитываются при каждом внесении данных в таблицу Predicts. Алгоритм подсчёта точности следующий:

1. Подсчитываем число строк в таблице Predicts. Пусть их n штук (n – число предсказаний).
2. Считаем для каждого сайта количество совпадений предсказанной погоды с фактической (пусть m – число успешных предсказаний).
3. Точность для выбранного сайта вычисляется по формуле: acc = m/n.

Блок парсинга погоды. В данном блоке используются библиотеки BeautifulSoup, requests, re, datetime. Здесь формируется предсказание погоды на оставшийся день. Сначала получаем HTML-страницу, обрабатываем её через BeautifulSoup и re, получаем на выходе массив предсказаний погоды. С помощью datetime сортируем массив и выделяем только прогнозы на оставшийся день. По нажатию пользователем соответствующей кнопки выводим ему сформированный прогноз выбранного сайта.

Блок интерфейса. В данном блоке, в основном, фигурирует библиотека PyQt6. Было решено делать интерфейс на языке Qt, т.к. он хорошо адаптирован библиотекой для использования на Python. Интерфейс содержит в себе название каждого сайта, кнопку «Показать погоду» справа от названия, коэффициент точности, а также фрагмент со слайдером, в котором показывается пользователю прогноз выбранного сайта (рис. 2).

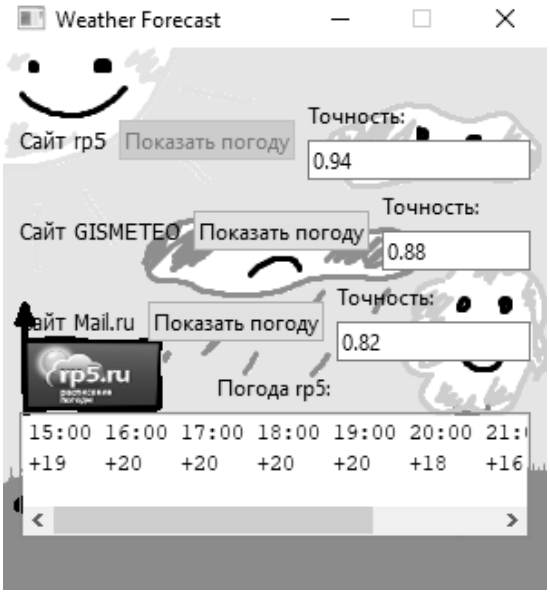


Рис. 2. Главное меню

3. Работа приложения

При запуске программы видим окно, представленное на рис. 3.

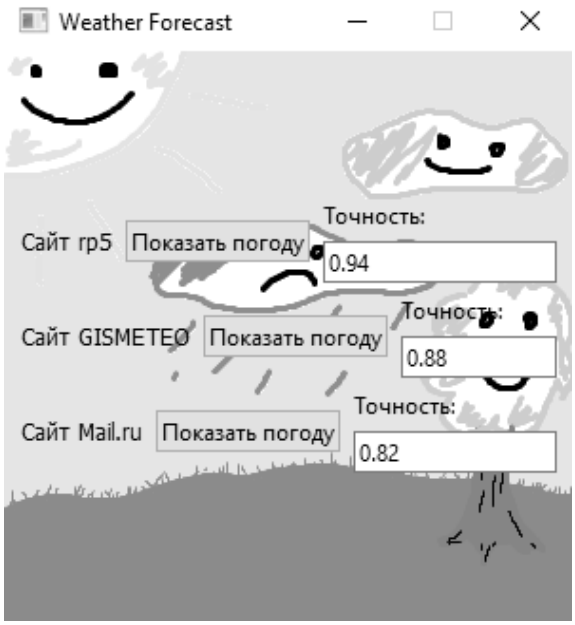


Рис. 3. Стартовый вид приложения

Все кнопки «Показать погоду» активны, а поле с прогнозом на оставшийся день скрыто от пользователя. Коэффициент точности запрашивается при запуске приложения и помещается в поле, из которого можно его скопировать, но нельзя отредактировать.

При выборе сайта, погоду которого мы хотим посмотреть, кнопка «Показать погоду» этого сайта становится неактивной и появляется поле с прогнозом на оставшийся день. Также меняется фон приложения, появляется флаг с логотипом выбранного сайта (снимок экрана был сделан в 16:31) (рис. 4).

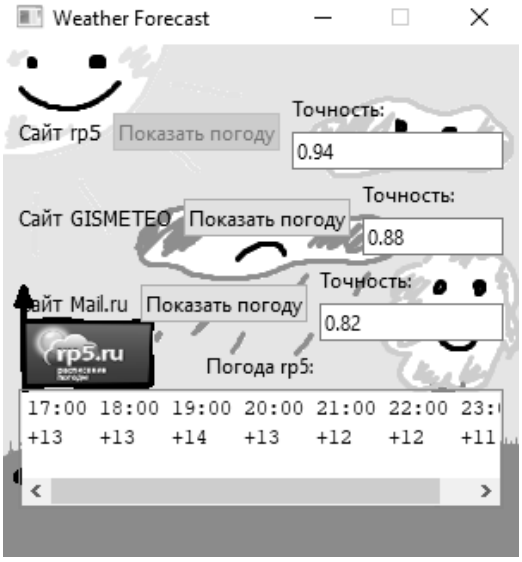


Рис. 4. Приложение с выбранным сайтом rp5

При выборе другого сайта кнопка «Показать погоду» выбранного сайта становится неактивной, в то время как та же кнопка предыдущего сайта активируется. Изменяется поле прогноза, а также фон приложения на соответствующий выбранному сайту (рис. 5)

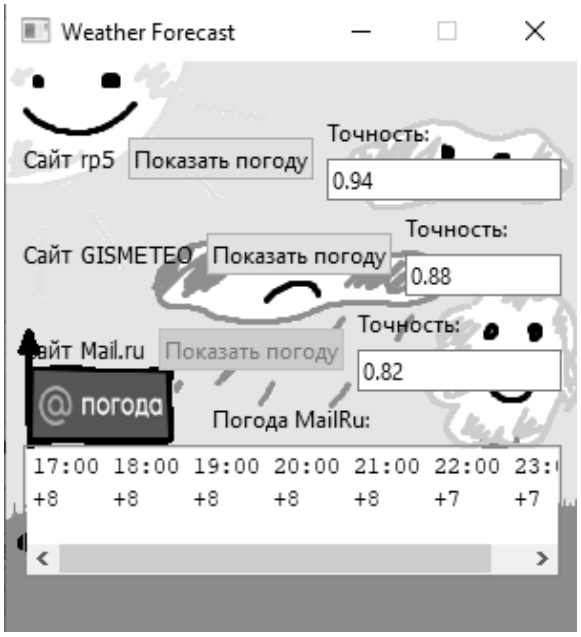


Рис. 5. Приложение с выбранным сайтом Mail.ru

Таким образом, взаимодействие с приложением достаточно простое и интуитивно понятное пользователю.

Со своей основной задачей – определением точности прогноза погоды, приложение справляется тем лучше, чем дольше оно работает, т.к. точность обновляется ежедневно.

Приложение также показывает и прогноз погоды на текущий день для любого из трех сайтов, причем (при желании) имеется возможность скопировать эти данные для дальнейшего использования, но заблокирована возможность их редактирования напрямую, что делает функционал программы устойчивым.

Заключение

В работе ставилась задача по созданию приложения, анализирующего точность предсказания прогноза погоды сайтов гр5, GISMETEO и ПогодаMail.ru. Данные сайты были выбраны в силу того, что они обладают дружественным API. В качестве показателя точности прогнозов выбрано отношение числа подтвердившихся прогнозов к общему числу прогнозов.

Поставленная задача была выполнена. Приложение ежедневно сравнивает прогноз погоды с реальной погодой и обновляет показатель точности. Кроме того, приложение показывает прогноз погоды выбранного пользователем сайта на текущий день. Созданное приложение обладает дружественным и интуитивно понятным интерфейсом. Приложение было успешно протестировано. Данное приложение позволит определить наиболее точный на текущий момент сайт, что, в свою очередь, помогает в организации любых мероприятий, связанных, например, с природой.

В дальнейшем планируется модифицировать приложение, увеличив частоту сравнения реальной погоды с прогнозом, что, несомненно, повысит точность оценки сайта. Кроме того, предполагается предоставить пользователю возможность выбора длитель-

ности выводимого прогноза погоды (до конца дня, на сутки, на три дня). Будет также немного изменен и сам интерфейс приложения.

ЛИТЕРАТУРА

1. Руководство по Python [Электронный ресурс] // URL: https://docs-python.ru (дата обращения: 25.05.2024 г.)
2. Документация библиотеки psycopg2 [Электронный ресурс] // URL: https://www.psycopg.org/docs/ (дата обращения 17.03.2024 г.)
3. Документация библиотеки requests [Электронный ресурс] // URL: https://requests.readthedocs.io/en/latest/ (дата обращения 29.10.2023)
4. Документация библиотеки BeautifulSoup [Электронный ресурс] // URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата обращения 17.11.2023)
5. Документация библиотеки datetime [Электронный ресурс] // URL: https://docs.python.org/3/library/datetime.html (дата обращения 04.10.2023)
6. Документация библиотеки sys [Электронный ресурс] // URL: https://docs.python.org/3/library/sys.html (дата обращения 21.11.2023)
7. Документация библиотеки os [Электронный ресурс] // URL: https://docs.python.org/3/library/os.html (дата обращения 21.11.2023)
8. Документация библиотеки re [Электронный ресурс] // URL: https://docs.python.org/3/library/re.html (дата обращения 15.11.2023)
9. Документация библиотеки shedule [Электронный ресурс] // URL: https://schedule.readthedocs.io/en/stable/index.html (дата обращения 23.01.2024)
10. Документация библиотеки PyQt6 [Электронный ресурс] // URL: https://www.riverbankcomputing.com/static/Docs/PyQt6/ (дата обращения 11.04.2024)

ПОИСК УТЕЧЕК ПАМЯТИ В ДИНАМИЧЕСКИХ СТРУКТУРАХ СПИСОЧНОГО ТИПА ПУТЁМ ПРИМЕНЕНИЯ 2SAT-ЗАДАЧИ

Сафонов Д.Д., Андреева В.В.
Томский государственный университет
safonov.dd.00@gmail.com, avv.21@mail.ru

Введение

Тестирование и верификация программного обеспечения (ПО) играют критическую роль в обеспечении его качества и надёжности. Верификация фокусируется на соответствии ПО заданным требованиям и спецификациям, предотвращая дорогостоящие ошибки на последующих этапах жизненного цикла.

Существует множество методов верификации и тестирования, которые можно разделить на несколько групп. Экспертиза, основанная на опыте и знаниях специалистов, статический анализ – автоматизированная проверка кода на соответствие формальным правилам и шаблонам [1,2], динамический анализ – выявление ошибок во время выполнения программы, формальные методы – использование математических моделей для доказательства корректности ПО, синтетические методы – комбинирование различных подходов.

В данной работе рассматривается статический анализ для выявления утечек памяти в динамических структурах данных, таких как списки и деревья.

Языки программирования C и C++ предоставляют гибкие возможности управления памятью, но эта свобода может привести к утечкам, когда выделенная память не освобождается должным образом. Накопление таких утечек может привести к сбоям и некорректной работе программы [3].

Статический анализ является эффективным инструментом для раннего выявления утечек памяти, не требуя запуска программы. Основными проблемами при статическом анализе утечек памяти на вышеупомянутых языках могут являться точность моделирования действий с памятью. Если говорить о списочных структурах, их можно смоделировать в виде графа, а утечка памяти в данном случае будет представлять собой отсутствие определённых рёбер между узлами. Проверить связность графа можно с помощью 2SAT-задачи. Данная работа является продолжением развития подхода к обнару-

жению утечек ресурсов, которые были предложены в [2,4]. Прилагаемый подход статического анализа кода анализирует именно тот участок, в котором непосредственно выполняются операции над списочными структурами. Сами операции моделируются с помощью рассматриваемой модели 2SAT, которая представляет логическую формулу вида КНФ, состоящей из дизъюнктов только от двух переменных. С помощью дизъюнктов описываются отношения между элементами динамических структур. Результатом анализа является проверка такой формулы на выполнимость с помощью SAT-решателя. Невыполнимая формула демонстрирует факт отсутствия утечек памяти, т.к. построена модель, в которой нет разрывов связей между элементов.

В контексте данной работы поиск утечек памяти будет также сводиться к построению соответствующей 2SAT-формулы и поиску решения с определенным ограничением решения. По результату этих действий можно делать вывод о наличии утечек памяти, указать утерянные элементы динамической структуры списочного типа, а также найти место ошибки.

Важно отметить, что сведение задачи поиска утечек к 2SAT не всегда возможно. Однако в некоторых случаях этот подход может быть эффективным и позволит быстро выявить потенциальные утечки.

1. Списочные структуры данных

Связанные списки позволяют эффективно хранить и обрабатывать произвольное количество данных благодаря гибкой структуре и динамическому изменению размера.

1.1. Линейный односвязный список (ЛОС)

Линейный односвязный список представляет собой последовательность элементов, связанных указателями. Объектом, определяющим объект списка, является переменная-указатель Head, которая указывает на первый элемент списка. Соответственно, если Head – пустой указатель, значит, список пуст.

Каждый элемент списка содержит данные и указатель на следующий элемент. В случае последнего элемента списка этот указатель является пустым.

Использование списков обеспечивает эффективное добавление и удаление элементов за константное время, динамический размер, а также гибкость в физическом размещении элементов. Из недостатков стоит отметить сложность доступа к произвольному элементу, необходимость выделения дополнительной памяти на указатели, менее эффективный последовательный доступ, чем у массивов и векторов, а также уязвимость к потере указателей на объекты памяти, что и является утечкой памяти.

Пример линейного односвязного списка предоставлен на рис. 1.

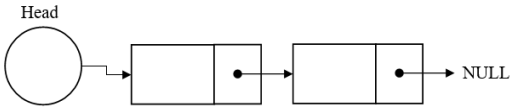


Рис. 1. Линейный односвязный список

1.2. Списочная структура "дерево"

Списочная структура дерево – это структура данных, представляющая собой иерархическую организацию объектов (узлов), где каждый объект может иметь одного или нескольких потомков, связанных между собой отношением "родитель – потомок". Каждый узел содержит информацию и ссылки на своих потомков. Корень дерева не имеет родителя, а листья – потомков. Двоичное дерево изображено на рис. 2.

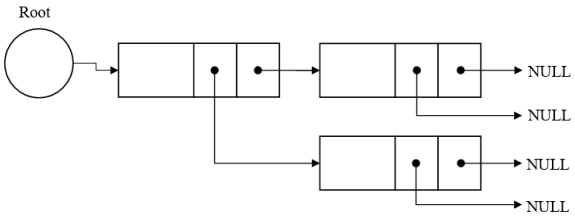


Рис. 2. Двоичное дерево

2. Операции над списочными структурами

Программист при работе со списочными структурами может допустить ошибку при добавлении/удалении элементов или при изменении значений указателей. Чтобы избежать ошибки при добавлении, нужно определить место и правильно провести связи между уже существующими элементами и новым.

2.1. Добавление элемента в указанное место

На рис. 3 изображена ситуация, когда нужно добавить новый элемент.

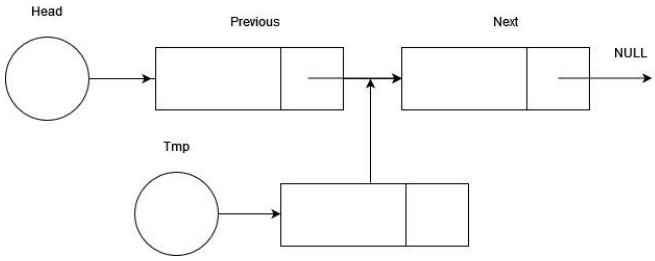


Рис. 3. Добавления нового элемента

Для добавления нужно сначала установить связь между объектом Tmp и Next (рис. 4), после чего установить связи между Previous и новым элементом (рис. 5), затем допускается удаление переменной-указателя Tmp (рис. 6).

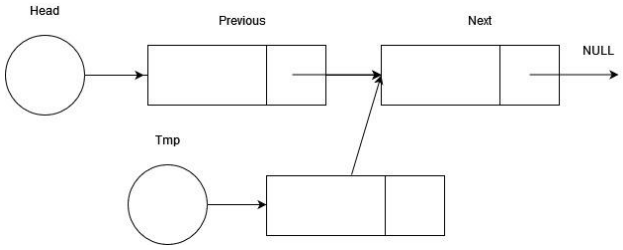


Рис. 4. Установление связи с следующим узлом

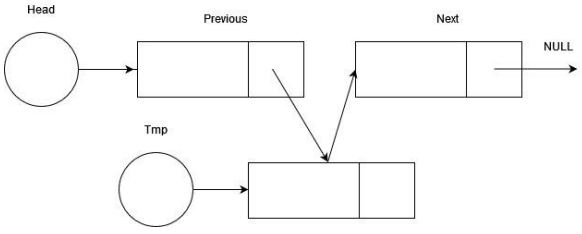


Рис. 5. Установление связи с предыдущим узлом

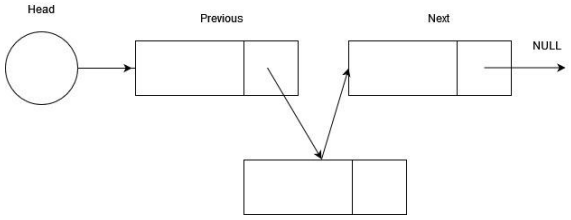


Рис. 6. Удаление указателя Tmp

Если связь между Previous и Tmp будет установлена раньше, чем между Tmp и Next, это приведет к утечке памяти (рис. 7).

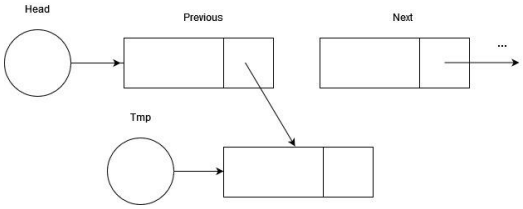


Рис. 7. Утечка памяти при неправильном добавлении элемента

2.2. Добавление в начало списка

На рис. 8 изображена ситуация, когда нужно добавить элемент в начало списка.

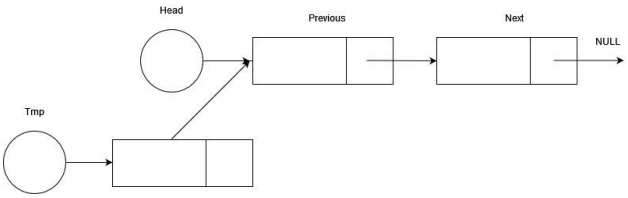


Рис. 8. Добавление в начало списка

Сначала нужно провести связи между новым элементом и элементом Next (рис. 9), после – установить связи между Head и новым элементом (рис. 10) и удалить указатель Tmp (рис. 11).

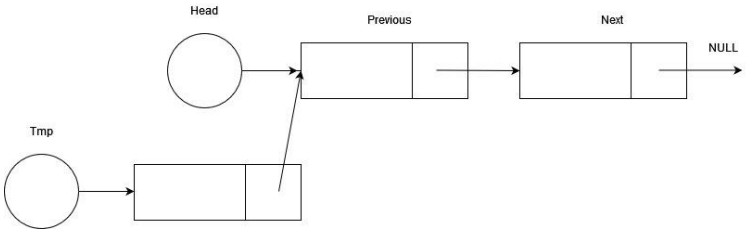


Рис. 9. Проведение связи между новым элементом и элементом Next

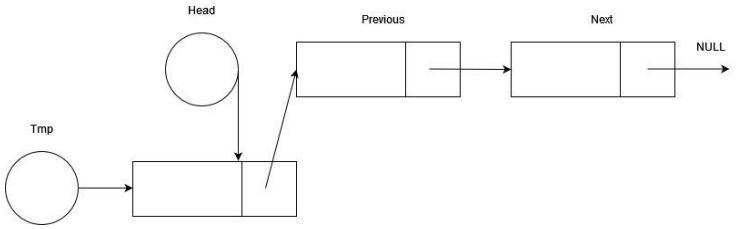


Рис. 10. Проведение связи между переменной-указателем Head и новым элементом

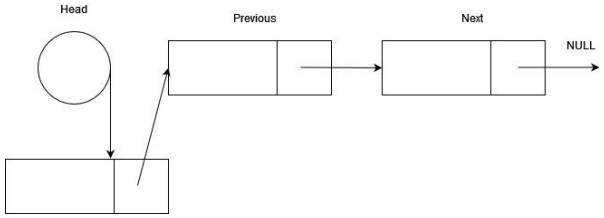


Рис. 11. Освобождение ресурсов из-под переменной-указателя Tmp

Т.к. это частный случай операции "добавление элемента в указанное место в списке", первый шаг с определением места отсутствует по причине того, адрес первого элемента списка уже известен (элемент Next на рис. 8), переменная-указатель Head хранит адрес на этот элемент. В данной операции может быть допущена аналогичная ошибка, отличие заключается в том, что теперь безвозвратно утерян доступ ко всему списку. Либо ошибка может быть связана с тем, что не будет перемещен указатель Head, что впоследствии может привести к тому, что память, захваченная под указатель Tmp, не будет сохранена.

2.3. Добавление в конец списка

Данная операция является ещё одним частными случаем операции "добавление элемента в указанное место в списке".

Для выполнения операции нужно сделать обход списка, начиная с первого элемента списка, на который указывает переменная-указатель Head. Это делается для получения адреса последнего элемента. Этот элемент в данной операции будет именоваться Previous (рис. 12).

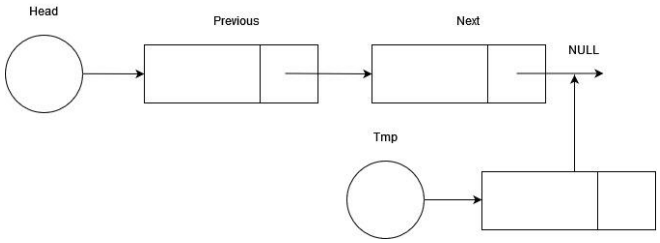


Рис. 12. Получение адреса последнего элемента списка,
для определения места, в которое будет помещён новый элемент

Затем нужно установить связи между элементом Previous и новым элементом списка (рис. 13).

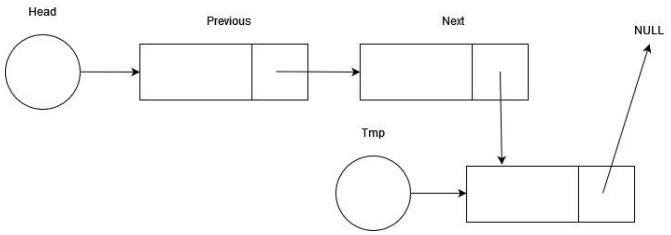


Рис. 13. Проведение связи между элементом Previous и новым элементом списка

После этого допускается освобождение ресурсов из-под переменной-указателя Tmp (рис. 14).

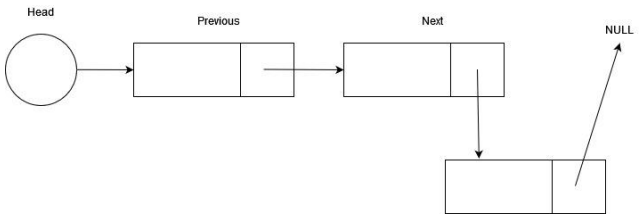


Рис. 14. Освобождение ресурсов из-под переменной-указателя Tmp

Следует отметить тот факт, что в данной операции невозможно допустить ошибку неправильной последовательности проведения связей.

2.4. Удаление указанного элемента

Операция "удаление указанного элемента из списка" является следующей основной операцией над линейными односвязными списками. Схематичное представление операции изображено на рис. 15. Она также выполняется в несколько этапов и имеет особенности проведения связей между элементами списка.

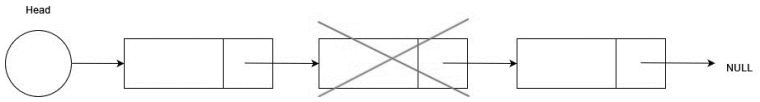


Рис. 15. Операция "удаление указанного элемента из списка"

Для начала нужно определить место элемента, который нужно удалить из списка. Для этого необходимо пройти список с самого начала в поиске интересующего нас элемента (рис. 16).

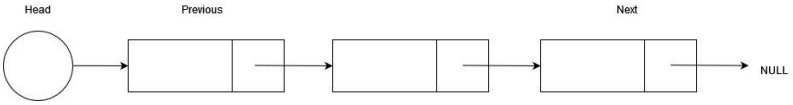


Рис. 16. Определение места элемента, который нужно удалить

Затем нужно создать вспомогательную переменную-указатель Tmp и установить связи между ней и элементом, который необходимо удалить (рис. 17).

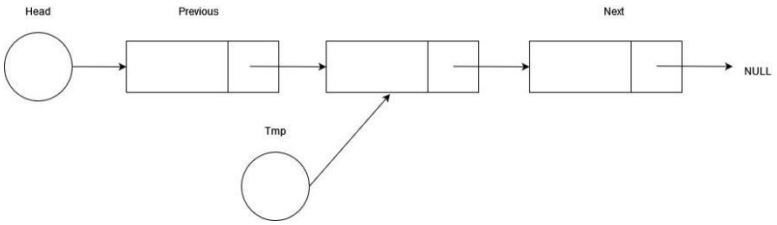


Рис. 17. Создание вспомогательной переменной-указателя Tmp
и проведение связи между ней и элементом, который нужно удалить

Установить связи между элементами Previous и Next (рис. 18).

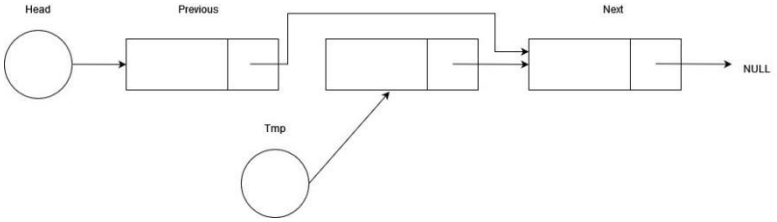


Рис. 18. Проведение связи между элементом Previous и Next

После этого можно удалить элемент из списка (рис. 19) и освободить ресурсы из-под переменной-указателя Tmp (рис. 20).

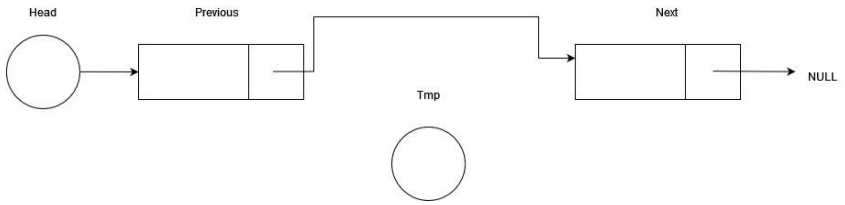


Рис. 19. Удаление элемента списка

Рис. 20. Освобождение ресурсов из-под переменной-указателя Tmp

При проведении данной операции могут возникнуть две ситуации, которые приведут к утечке памяти. В первой из них этап 2 пропущен, что приводит к тому, что вспомогательная переменная-указатель Tmp, которая должна содержать адрес элемента, который необходимо удалить, не создаётся. Это приводит к утечке памяти, вызванной тем, что потерян доступ к элементу списка, который необходимо было удалить из списка. Этот дефект опасен тем, что, хотя желаемый результат достигается – элемент удаляется из списка и факт утечки остается незамеченным, поскольку память из кучи не освобождается. Пример данной ошибки представлен на рис. 21.

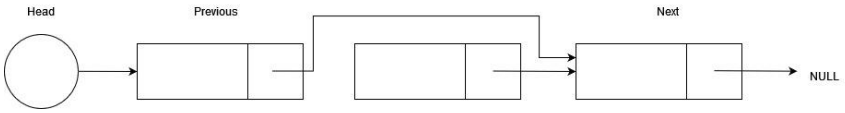


Рис. 21. Ошибка проведения связи, без сохранения адреса на удаляемый элемент

Вторая ситуация – это пропуск этапов 2 и 3. Удаление элемента происходит без проведения связей между частями списка. Такая ошибка приводит к потере части списка, начиная с элемента Next, а элемент Previous указывает на мусор в куче. Это может привести к аварийному завершению работы программы. Пример ошибки представлен на рис. 22.

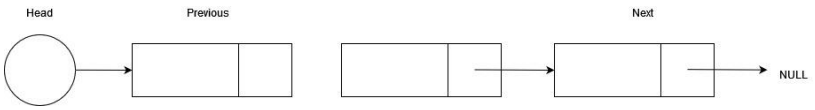


Рис. 22. Ошибка удаления элемента, до проведения связей

Далее рассмотрим частные случаи данной операции, такие как "удаление первого элемента списка", "удаление последнего элемента списка".

2.5. Удаление первого элемента

Операция "удаление первого элемента списка" по сути своей мало чем отличается от предыдущей. Этапы идентичны, за исключением первого, потому что адрес начального элемента списка уже известен, он хранится в переменной-указателе Head.

Ошибки, приводящие к утечкам ресурсов, аналогичны ошибкам при удалении указанного элемента списка. Схематично операция "удаление первого элемента списка" представлена на рис. 23.

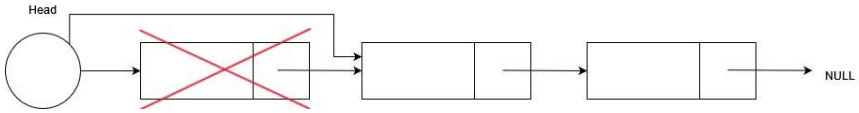


Рис. 23. Операция "удаление первого элемента списка"

2.6. Удаление последнего элемента

Операция "удаление последнего элемента списка" имеет отличия от базовой операции и имеет свои особенности проведения связей. Схематично операция представлена на рис. 24.

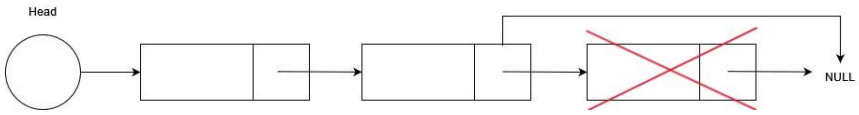


Рис. 24. Операция "удаление последнего элемента списка"

Для этого нужно сначала сделать обход всего списка, начиная с первого элемента списка, на который указывает переменная-указатель Head. Это нужно для получения адреса последнего элемента, который необходимо удалить. Предпоследний элемент в данной операции будет именоваться Previous (рис. 25).

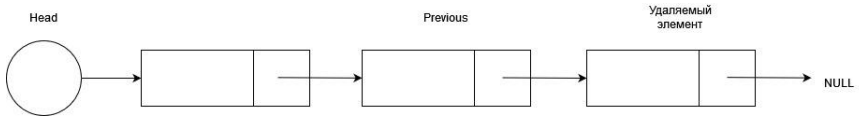


Рис. 25. Определение адреса элемента, который нужно удалить

Затем происходит удаление элемента списка (рис. 26) и присвоение полю-указателю элемента списка Previous значения NULL (рис. 27).

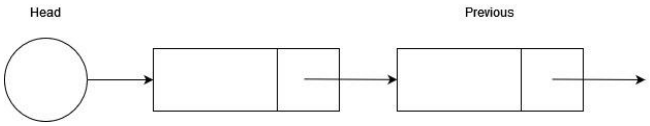


Рис. 26. Удаление элемента списка

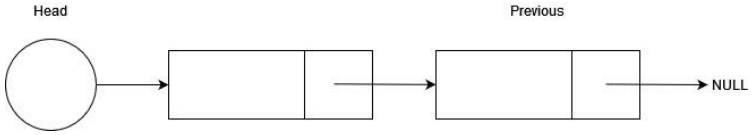


Рис. 27. Присвоение полю-указателю элемента списка Previous значения NULL

Распространенной ошибкой в данной операции является то, что пользователь не присваивает значение NULL полю-указателю элемента списка Previous, из-за чего элемент указывает на мусор в куче. Пример данной ошибки представлен на рис. 26.

2.7. Операции над списочной структурой "дерево"

С точки зрения представления действий над древовидными структурами в вышеописанной форме, существенных отличий от линейного односвязного списка нет. Узлы также содержат указатели, но вместо одного указателя на следующий элемент списка каждый может содержать несколько указателей на узлы-потомки, в зависимости от вида дерева.

Существуют виды деревьев, где есть определенные правила добавления элементов, такие как сбалансированное дерево, где ранг левого и правого поддерева могут отличаться не более, чем на 1, или дерево поиска, где для каждого узла каждый элемент в

левом поддереве меньше, а в правом – больше. Эти ограничения могут лишь уменьшать наглядность операций, но не изменяют суть работы с указателями в структурах.

Например, в общем случае структуры "дерево" для операции добавления нового узла необходимо также найти место добавления. Для этого необходимо определить правила поиска места. В случае, если необходима навигация по значениям в узлах, требуется обход дерева, который, в свою очередь, также может выполняться по-разному, например, вглубь или вширь. Когда место добавления будет определено, выбирается, каким из потомков станет новый узел, или, например, если новый узел добавляется в корень дерева, каким потомком станет узел, который был корнем до этого.

3. Логическая формула 2КНФ как модель ориентированного графа

Списочные структуры можно представить в виде ориентированного графа, где вершины соответствуют элементам, а дуги – связям между ними. Обнаружение утечек памяти сводится к анализу графа на нарушение связей.

В [5] предлагается метод интерпретации ориентированного графа с использованием 2КНФ (2-конъюнктивной нормальной формы) булевой формулы. Этот метод заключается в следующем: для каждой пары вершин, между которыми существует направленная связь, создается дизъюнкция из двух переменных. Например, направленному ребру от вершины V_i к V_j будет соответствовать логическое выражение вида $\overline{x_i} \vee x_j$, где x_i и x_j – логические переменные. Таким образом, логическая формула для ориентированного графа представляет собой конъюнкцию дизъюнкций, где каждая дизъюнкция соответствует ребру графа и состоит из двух переменных. Формула 2КНФ представляет собой булеву формулу конъюнктивной нормальной формы т.е. конъюнкция дизъюнкций от двух переменных.

Как показано в [5], такая формула обладает интересными свойствами. Первое свойство заключается в том, что каждая дизъюнкция содержит ровно одну положительную и одну отрицательную переменную, что, в свою очередь, гарантирует выполнимость данной формулы. Такая формула имеет два решения: белое решение, где каждая переменная принимает значение "истина", и черное решение, где каждая переменная принимает значение "ложь". Если формула имеет только эти два решения и ни одного другого, то граф считается сильно связным [5,6]. Под сильно связанным графом понимается граф, в котором каждая вершина может быть достигнута из другой вершины [7].

Для обеспечения этого условия предлагается внедрить в формулу два дизъюнкта, которые будут представлять чёрно-белое ограничение: один состоит только из положительных переменных, а другой - только из отрицательных. Например, для графа с дугами $V_1 \to V_2$, $V_2 \to V_3$, $V_3 \to V_1$ соответствующая 2КНФ будет иметь вид $(\overline{x_1} \vee x_2)(\overline{x_2} \vee x_3)(\overline{x_3} \vee x_1)$, а чёрно-белое ограничение будет выглядеть как $(x_1 \vee x_2 \vee x_3)(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$.

Таким образом, формула $(\overline{x_1} \vee x_2)(\overline{x_2} \vee x_3)(\overline{x_3} \vee x_1)(x_1 \vee x_2 \vee x_3)(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$ с чёрно-белым ограничением является невыполнимой в случае, если граф сильно связан, иначе формула будет выполнимой.

В этой работе рассматривается подход, который сводит задачу обнаружения утечек памяти к анализу графа, представляющего списочную структуру, на предмет сильной связности. Происходит это с использованием решения 2SAT-задачи. Основная идея заключается в том, что невыполнимость формулы указывает на сильную связность графа, что, в свою очередь, говорит об отсутствии утечки памяти. Это гарантирует корректность связей между элементами в списочных структурах. Таким образом, решение задачи определения факта наличия утечки памяти сводится к формированию формулы 2КНФ, наложению черно-белого ограничения для анализируемой списочной структуры

и последующему решению 2SAT-задачи. При этом предлагается в процессе формирования формулы вводить специальные правила, которые обеспечивают сильную связность, не затрагивая операции, проводимые над списочными структурами, но при этом позволяющие выявить нарушение связей, а также сохранение промежуточного результата образования формулы для возможности поиска места ошибки.

Рассмотрим граф G – граф, соответствующий списочной структуре. Граф состоит из вершин и рёбер, первые описывают элементы списочной структуры, вторые – связи между этими элементами. Можно выделить два типа вершин графа G: вершины, которые моделируют переменные-указатели, хранящие адреса элементов списка, и вершины, моделирующие непосредственно элементы списка.

Пусть $V=\{V_i\}$ – множество переменных-указателей, хранящих адреса элементов списочной структуры, $N=\{N_i\}$ – множество адресов элементов, хранящихся в динамической памяти.

Генерация формулы происходит параллельно с анализом пользовательского программного кода, промежуточные результаты сохраняются. Для проверки сгенерированной формулы на выполнимость используется SAT-решатель [6].

Перед проверкой формулы SAT-решателем добавляется к ней два дизъюнкта, состоящие из всех элементов множеств V и N. Они представляют собой чёрное и белое решения задачи 2SAT.

Дизъюнкт, представляющий ограничение в виде белого решения, выглядят следующим образом: $(V_1 \vee V_2 \vee ... \vee V_i \vee N_1 \vee N_2 \vee ... \vee N_j)$ для белого решения, и $(\overline{V}_1 \vee \overline{V}_2 \vee ... \vee \overline{V}_i \vee \overline{N}_1 \vee \overline{N}_2 \vee ... \vee \overline{N}_j)$, для черного решения, где i соответствует количеству элементов множества V, а j соответствует количеству элементов множества N.

Таким образом, это ограничение позволяет сфокусироваться только на поиске решения, которое дает ответ на вопрос о том, является ли граф сильно связным.

3.1. Принцип и порядок формирования булевой формулы для линейного односвязного списка

Для обнаружения потенциальных дефектов, которые могут привести к утечкам памяти, проводится верификация кода. Для этого производится анализ необходимых фрагментов кода, которые содержат операции над динамическими списочными структурами [8], такими как списки, строится булева формула для участка кода, производится проверка с помощью решения SAT-задачи [9].

Для операций над односвязными списками разработаны правила преобразования в КНФ. Для деревьев структура отличается только количеством указателей в узлах.

3.2. Создание списка

Для создания списка программисту нужно объявить указатель, который принято называть Head, добавить первый элемент списка, установить связь с Head, присвоить полю-указателю элемента списка значение NULL.

Для каждого действия происходит изменение формулы. При создании элемента и установлении связи между указателем нужно добавить в множества V и N по одному элементу. V и N – это множества указателей и объектов. В формулу добавляется первый дизъюнкт $(\overline{V}_1 \vee N_1)$. Для обеспечения сильной связности графа проводится дополнительная связь между концом списка и указателем на структуру. При создании списка это будет $(\overline{N}_1 \vee V_1)$, формула принимает вид $(\overline{V}_1 \vee N_1)(\overline{N}_1 \vee V_1)$. Результат создания списка изображён на рис. 28.

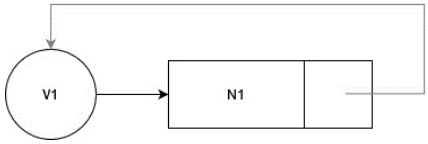


Рис. 28. Граф, описывающий список, соответствующий сгенерированной формуле

3.3. Добавление элемента в начало списка

Программисту нужно создать переменную-указатель для нового элемента и элемент списка, провести связи между ними, затем установить связи между связи между новым элементом и элементом Next, между Previous и новым элементом, после этого удалить указатель Tmp.

При установлении связи между новым элементом и Next создаётся элемент N_i, где i – количество уже созданных элементов множества N. В формулу добавляется дизъюнкт, который соответствует созданной связи. Изменение значения Head вызывает изменение дизъюнктов, как это показано на рис. 29, формула принимает вид $(\overline{V}_1 \vee N_2)(\overline{N}_2 \vee N_1)(\overline{N}_1 \vee V_1)$.

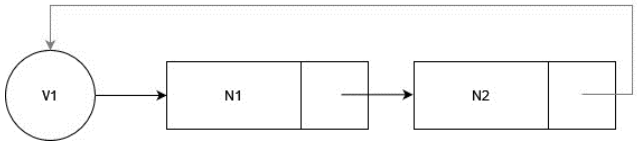


Рис. 29. Граф после выполнения операции "добавление элемента в начало списка"

3.4. Добавление элемента в конец списка

При добавлении в конец списка программисту нужно создать указатель Tmp и новый элемента, установить связи между ними, затем установить связи между Previous и новым элементом.

Как уже было упомянуто, при рассмотрении первой операции создание переменных-указателей и элементов списка не вызывает изменение формулы. При установлении связи между новым элементом и Next создаётся элемент N_1, который моделирует новый элемент, где i – количество уже созданных элементов множества N. Таким образом, в формулу будет добавлен дизъюнкт, который соответствует созданной связи. Также нужно изменить связи последнего элемента и переменной-указателя Head, которая является дополнительной, которая обеспечивает сильную связность графа).

На рис. 30 изображены граф, моделирующий список, а также соответствующая ему формула $(\overline{V}_1 \vee N_1)(\overline{N}_1 \vee N_2)(\overline{N}_2 \vee N_3)(\overline{N}_3 \vee V_1)$.

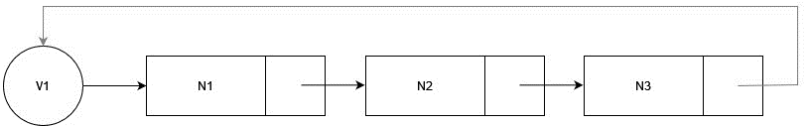


Рис. 30. Граф после выполнения операции "добавление элемента в конец"

3.5. Добавление элемента в указанное место в списке

Для выполнения данной операции нужно создать переменную-указатель Tmp и новый элемент, установить связи между ними, найти место добавления нового элемента, установить связи между новым элементом и Next, Previous и новым элементом. После этого допускается удаление переменной-указателя Tmp.

Создание элемента или переменной не вызывает изменений в формуле. При обходе списка также никакие связи не изменяются, следовательно, формула остается прежней. При установлении связи между новым элементом и Next создаётся элемент N_i, где i со-

ответствует количеству существующих элементов множества *N*. Также в формулу добавляется дизъюнкт, который соответствует новой связи. Необходимо изменить связь Previous и Next на связь между Previous и новым элементом.

На рис. 31 изображены граф, моделирующий список, а также соответствующая ему формула $(\overline{V}_1 \vee N_1)(\overline{N}_1 \vee N_2)(\overline{N}_2 \vee N_4)(\overline{N}_4 \vee N_3)(\overline{N}_3 \vee V_1)$.

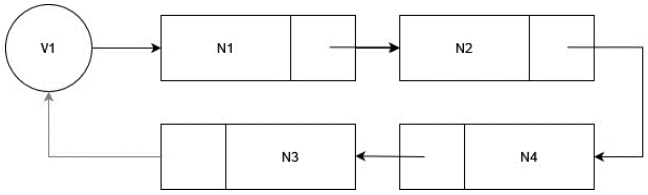


Рис. 31. Граф после выполнения операции "добавление элемента в указанное место в списке"

3.6. Удаление указанного элемента из списка

Для удаления произвольного элемента из списка нужно определить в списке место элемента, который необходимо удалить, затем создать вспомогательную переменную-указатель Tmp и установить связи между ними, изменить связь между Previous и удаляемым на Next. После этого допускается удаление элемента списка и освобождение ресурсов из-под переменной-указателя Tmp.

При установлении связи между переменной-указателем Tmp и удаляемым элементом необходимо создать элемент *V*, который представляет собой переменную-указатель, где *i* соответствует количеству уже созданных элементов множества *V*. После этого происходит добавление в формулу дизъюнкта, соответствующего новой связи.

Следует обратить внимание, что при добавлении такой связи, граф перестаёт быть сильно связанным, а также это приводит к ошибке в формуле. Для решения данной проблемы при проведении связи между новой переменной-указателем и уже созданным элементом списка необходимо провести дополнительную связь из последнего элемента списка к переменной-указателю. Такие связи для наглядности будут выделяться красным цветом. Пример представлен на рис/ 32. Формула принимает вид $(\overline{V}_1 \vee N_1)(\overline{N}_1 \vee N_2)(\overline{N}_2 \vee N_4)(\overline{N}_4 \vee N_3)(\overline{N}_3 \vee V_1)(\overline{V}_2 \vee N_2)(\overline{N}_2 \vee V_2)$.

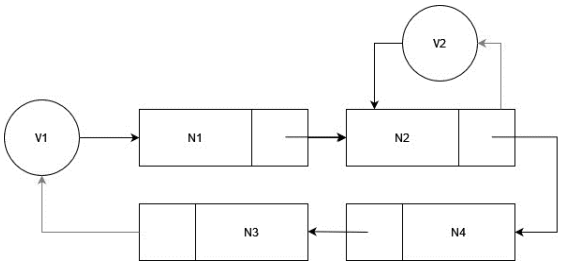


Рис. 32. Проведение связи между новой переменной-указателем и существующим элементом списка

Изменение связи между элементами Previous и Next отражается в формуле добавлением дизъюнкта этих двух элементов. Удаление элемента списка вызывает удаление из формулы всех связанных с этим элементом дизъюнктов и рёбер соответственно. Последний шаг операции не вызывает изменений в формуле. Это связано с тем, что все необходимые дизъюнкты и рёбра уже были удалены на предыдущем шаге.

На рис. 33 изображены граф, моделирующий список, а также соответствующая ему формула $(\overline{V}_1 \vee N_1)(\overline{N}_1 \vee N_4)(\overline{N}_4 \vee N_3)(\overline{N}_3 \vee V_1)$.

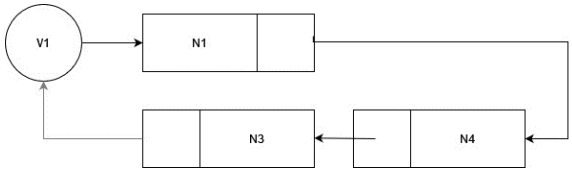


Рис. 33. Граф после выполнения операции "удаление указанного элемента из списка"

3.7. Удаление последнего элемента списка

Для удаления последнего элемента нужно сделать обход списка для получения адреса последнего элемента, удалить его и присвоить значения NULL полю-указателю, новому последнему элементу в списке.

При формировании графа и формулы обход списка не приводит к изменениям. Удаление последнего элемента списка требует удаления из формулы и графа всех связанных с ним дизъюнктов и рёбер соответственно. Присвоение значения NULL полю-указателю нового последнего элемента списка требует обеспечения сильной связности графа. Следовательно, требуется проведение дополнительных связей между последним элементом и указателями на структуру.

На рис. 34 изображены граф, моделирующий список, а также соответствующая ему формула $(\overline{V}_1 \vee N_1)(\overline{N}_1 \vee N_4)(\overline{N}_4 \vee V_1)$.

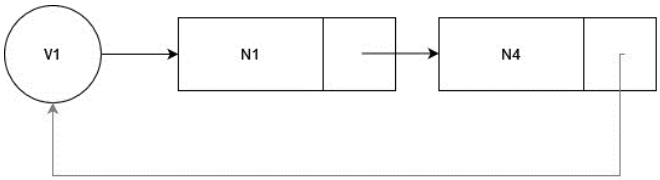


Рис. 34. Граф после выполнения операции "удаление последнего элемента списка"

3.8. Удаление первого элемента списка

При удалении первого элемента нужно создать вспомогательную переменную-указатель Tmp и установить связи с первым элементом в списке, затем установить связи между переменной-указателем Head и элементом Next. После этого можно удалить элемент из списка и освободить ресурсы из-под переменной-указателя Tmp.

Для установления связи между переменной-указателем Tmp и первым элементом списка элемент V_i, который моделирует переменную-указатель, где i соответствует количеству уже существующих элементов множества V. Далее в формулу добавляется дизъюнкт, моделирующий новую связь. Установление связи между переменной-указателем Head и элементом Next происходит с помощью добавления дизъюнкта от этих двух элементов в формулу. Удаление элемента списка вызывает удаление из формулы и графа всех связанных с ним дизъюнктов и рёбер соответственно. Освобождение ресурсов из-под переменной-указателя Tmp изменяет формулу. Это происходит потому, что все необходимые дизъюнкты и рёбра уже были удалены на предыдущем этапе.

На рис. 35 изображены граф, моделирующий список, а также соответствующая ему формула $(\overline{V}_1 \vee N_1)(\overline{N}_1 \vee N_4)(\overline{N}_4 \vee V_1)$.

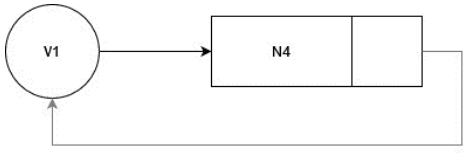


Рис. 35. Граф после выполнения операции "удаление первого элемента списка"

3.9. Присвоение нового значения переменной-указателю Head. Разбор и устранение утечки памяти

Рассмотрим ситуацию, когда пользователю необходимо присвоить переменной-указателю Head новое значения, чтобы указатель на структуру содержал в себе адрес следующего элемента списка. Необходимо установить новую связь между переменной-указателем Head и первым элементом списка на связь между переменной-указателем Head и вторым элементом списка. Граф после выполнения перезаписи, представлены на рис. 36, формула имеет вид $(\overline{V}_1 \vee N_2)(\overline{N}_2 \vee N_4)(\overline{N}_4 \vee N_3)(\overline{N}_3 \vee V_1)(\overline{N}_1 \vee N_2)$.

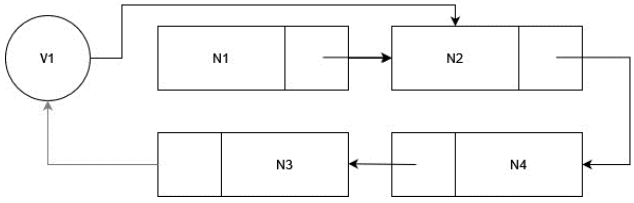


Рис. 36. Граф после выполнения перезаписи связи

Стоит отметить, что до элемента списка N1 больше нельзя добраться с помощью элементов V_i или N_j. Это сигнализирует о том, что теперь граф не является сильно связанным. Следовательно, формула будет иметь решение, что, в свою очередь, говорит о том, что существует подтвержденная утечка памяти.

Далее будет рассмотрен процесс определения объекта ошибки и поиск места утечки в коде.

Для исправления ошибки нужно создать вспомогательную переменную-указатель Tmp, присвоение ей значение адреса первого элемента списка. Сделать это необходимо до присвоения нового значения переменной-указателя Head.

Для установления связи между переменной-указателя Tmp и первым элементом списка создаётся элемент V_i, который описывает переменную-указатель, где i соответсвует количеству уже созданных элементов множества V. После этого в формулу добавляется дизъюнкт, моделирующий новую связь, Также добавляется вспомогательная связь между элементом списка, значение адреса которого было присвоено переменной-указателю Head и вспомогательной переменной-указателем Tmp.

Пример устранения утечки памяти представлен на рис. 37, формула имеет вид $(\overline{V}_1 \vee N_2)(\overline{N}_2 \vee N_4)(\overline{N}_4 \vee N_3)(\overline{N}_3 \vee V_1)(\overline{N}_1 \vee N_2)(\overline{N}_2 \vee V_2)(\overline{V}_2 \vee N_1)$.

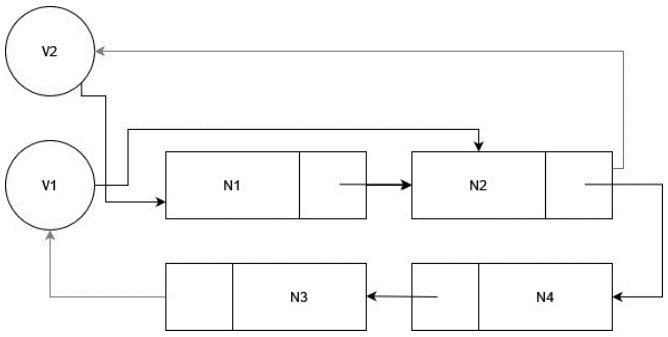


Рис. 37. Граф после устранения утечки памяти, вызванной неверным проведением связей

4. Поиск места ошибки

Задача поиска места возникновения ошибок в статическом анализе является нетривиальной, поскольку иногда констатировать факт наличия ошибки получается сильно позже, чем она появляется.

Самым простым примером утечки памяти может служить ситуация, когда была выделена память под объект памяти. Адрес этого объекта был утерян. Данный элемент не был частью списка, с этим объектом не существует связей. Определение объекта ошибки в этом случае сводится к анализу наличия объектов из множества *N* в формуле. Но что делать, если с элементом существуют связи, он является частью списка или дерева, но до него больше нельзя добраться с помощью указателей?

При исследовании проблемы поиска места ошибки было сделано предложение по решению данной задачи в случае, когда были нарушены связи между узлами в списке или дереве и адрес одного из узлов был утерян. Этот узел может быть частью списка или поддеревом, до которых больше нельзя добраться из головы списка или корня дерева. Идея заключается в том, что если на объект больше ничто не указывает, и этот объект был частью списка или дерева, то он будет входить в 2SAT-формулу только с отрицанием, следовательно, найденное решение будет содержать соответствующую ему переменную со значением 0, в то время как другие переменные – 1. Поскольку утечка памяти – ошибка, фактически, необратимая в рамках выполнения кода, остаётся найти место, где впервые произошла потеря этого объекта. Т.к., согласно алгоритму, предполагается, что формула строится вместе с анализом кода, на каждом шаге анализа формулу можно сохранять для возможности возврата, что сделает возможным повторную проверку кода в произвольном моменте. Данный подход поможет обнаружить популярный, но далеко не единственный тип ошибок в рассматриваемых структурах. При этом подход основывается на анализе построенной формулы и полученного решения, что является эффективным в рамках предложенного алгоритма по обнаружению утечек памяти.

Но даже в таком случае это может быть затруднительным, поскольку код может иметь множество вариантов исполнения в зависимости от различных условий, а также сложные синтаксические конструкции, что требует использования дополнительных инструментов для статического анализа кода.

На рис. 38 приведён пример кода, содержащего утечку памяти при работе с линейным односвязным списком.

```
struct ListNode{
    int nValue;
    ListNode*pNext;
}

void Initialize(ListNode*pNode, ListNode*pNextInit, int nValue){
    pNode->pNext = pNextInit;
    pNode->nValue = nValueInt;
}

int main()
{
    struct ListNode*elemH = nullptr;
    struct ListNode*tmp = nullptr;
    for(int i=0; i < 2; i++){
    tmp = new ListNode;
    Initialize(tmp, elemH,i);
    elemH = tmp;
    }
    tmp = new ListNode;
    Initialize(tmp, elemH,3);
    tmp = nullptr;
    /*Освобождение памяти ...*/
}
```

Рис. 38. Пример кода, содержащий утечку памяти при работе с линейным односвязным списком

В этом примере происходит корректное добавление в голову списка двух элементом, а затем добавление с ошибкой. Ошибка связана с тем, что указатель на список не начинает отслеживать память, захваченную под новый объект, как это происходило ранее. Затем указатель tmp, который отслеживал созданный элемент, обнуляется. Та-

ким образом, мы имеем утечку памяти, при которой элемент является частью списка, имеет связи с его элементами, но до него больше нельзя добраться с помощью указателей.

Графы, моделирующие списки при корректном добавлении двух элементов в голову, представлены на рис. 39. Формулы для каждой строки последовательно строятся согласно тому, как это описано в 3.

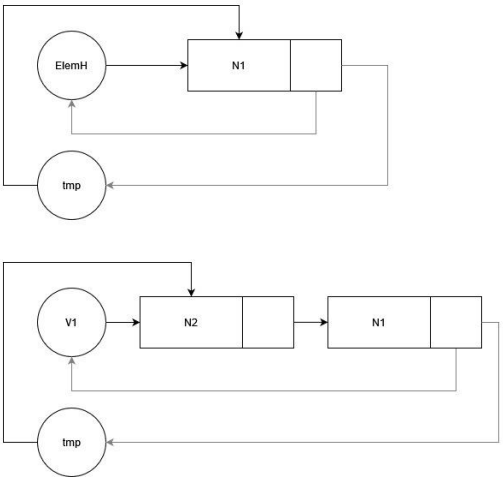


Рис. 39. Графы, соответствующие первому и второму добавлению элементов

После первого добавления формула выглядит следующим образом: $(\overline{\text{ElemH}} \vee N_1)(\overline{N}_1 \vee \text{ElemH})(\overline{\text{tmp}} \vee N_1)(\overline{N}_1 \vee \text{tmp})$, после второго добавления – $(\overline{\text{ElemH}} \vee N_2)(\overline{N}_2 \vee N_1)(\overline{N}_1 \vee \text{ElemH})(\overline{\text{tmp}} \vee N_2)(\overline{N}_2 \vee \text{tmp})$.

Рассмотрим подробнее добавление, приводящее к утечке памяти. После создания нового элемента под указателем tmp проводится связь между новым элементом и первым элементом списка, как это показано на рис. 40. Этой ситуации соответствует формула $(\overline{\text{ElemH}} \vee N_2)(\overline{N}_2 \vee N_1)(\overline{N}_1 \vee \text{ElemH})(\overline{\text{tmp}} \vee N3)(\overline{N}_3 \vee N_2)(\overline{N}_3 \vee \text{tmp})$.

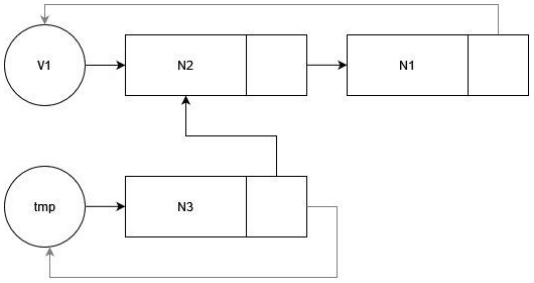


Рис. 40. Создание связи между новым элементом и первым элементом списка

Затем вместо обновления указателя ElemH на новый первый элемент в списке происходит обнуление указателя tmp. Таким образом, до нового элемента больше нельзя добраться с помощью указателей. Это действие показано на рис. 41. Формула приобретает вид $(\overline{\text{ElemH}} \vee N_2)(\overline{N}_2 \vee N_1)(\overline{N}_1 \vee \text{ElemH})(\overline{N}_3 \vee N_2)$.

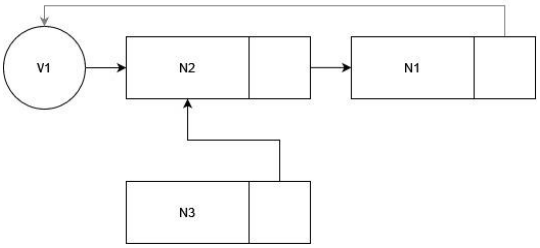


Рис. 41. Действие, приводящее к утечке памяти

Для определения наличия ошибки построенную формулу необходимо проверить SAT-решателем. К ней добавляется ограничение в виде черного и белого решения. Если решение будет найдено, это сигнализирует о том, что в программе существует утечка памяти и необходимо ее найти. Для итоговой формулы

$$(\overline{\mathrm{ElemH}} \vee N_2)(\overline{N}_2 \vee N_1)(\overline{N}_1 \vee \mathrm{ElemH})(\overline{N}_3 \vee N_2)$$

$$(\mathrm{ElemH} \vee N_1 \vee N_2 \vee N_3)(\overline{\mathrm{ElemH}} \vee \overline{N}_1 \vee \overline{N}_2 \vee \overline{N}_3)$$

будет найдено решение $\mathrm{ElemH} = 1$, $N_1 = 1$, $N_2 = 1$, $N_3 = 0$.

Согласно предложению по поиску места утечки, был утерян объект N_3, и необходимо найти предыдущее состояние формулы, при котором до этого объекта можно добраться с помощью указателей. Вернувшись к состоянию, изображённому на рис. 40, по формуле легко определить, что до этого объекта можно добраться с помощью указателя tmp. Проверяются дизъюнкты, начиная с тех, в которые утерянный объект входит без отрицания, производится поиск такой последовательной цепочки, где элемент с отрицанием является указателем. В приведённом примере таким единственным дизъюнктом будет $(\overline{\mathrm{tmp}} \vee N_3)$. Таким образом, происходит определение места утечки памяти.

Заключение

В данной работе разработан алгоритм статического анализа программного кода, позволяющий обнаруживать утечки памяти в динамических структурах списочного типа. Предлагаемое решение основано на формировании логической формулы 2КНФ для анализируемого участка кода и решении 2SAT-задачи.

Разработаны специальные правила, позволяющие обеспечить важное свойство для 2SAT-задачи, такое как сильносвязность графа, являющегося интерпретацией списочной структуры, при этом не нарушающие операции, проводимые над списочными структурами, но позволяющие обнаружить нарушение связей.

Предложенный подход программно реализован в виде анализатора на языке Python, обеспечивающего выполнение этапов алгоритма анализа. В реализации использованы библиотеки Clang [10] для синтаксического анализа C++ кода с помощью абстрактного синтаксического дерева [11], а также SAT-решатель из picosat [12]. В результате анализатор способен указать утерянные элементы динамической структуры списочного типа, а также найти место ошибки.

ЛИТЕРАТУРА

1. Static Analysis Tools [Электронный ресурс] URL: https://testingfaqs.org/t-static.html (дата обращения 25.04.2024).

2. *Серебренникова К.А., Андреева В.В.* Применение 2-SAT задачи для обнаружения утечки ресурсов в списочных структурах // Математическое и программное обеспечение информационных, технических и экономических систем : Материалы VIII Международной молодежной научной конференции, Томск, 26–30 мая 2021 года / Под общей редакцией И.С. Шмырина. – Труды Томского государственного университета. Серия физико-математическая. – Т. 306.– С. 214–218.

3. *Evans D.* Static detection of dynamic memory errors. – MIT Laboratory for Computer Science.

4. *Носов В.С.* Анализ динамических структур списочного типа на наличие утечек памяти путём применения 2SAT-задачи: выпускная бакалаврская работа по направлению подготовки: 02.03.02 – Фундаментальная ин-

форматика и информационные технологии – Томск: [б.и.], 2023. URL: https://vital.lib.tsu.ru/vital/access/manager/Repository/vital:18046
5. *Biro C., Kusper G.* Equivalence of strongly connected graphs and black-and-white 2-SAT problems // Miskolc Mathematical Notes. – 2018.
6. Связный граф [Электронный ресурс] // URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84; (дата обращения 25.04.2024).
7. Компонента сильной связности // Википедия: свободная энциклопедия: [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Компонента_сильной_связности#:~:text=Ориентированный%20граф%20(орграф)%20называется%20сильно,множество%20вершин%20компонентов%20сильной%20связности (дата обращения 25.04.2024).
8. Динамические структуры данных (лекция 29) // [Электронный ресурс] URL: https://www.intuit.ru/studies/courses/648/504/lecture/11455; (дата обращения 25.05.2024).
9. *Gopinath D., Zubair M.M., Khurshid J.D., Vaziri M.* Specification-Based Program Repair Using SAT. Finding bugs with a constraint solver // ISSTA (2000)
10. Clang Static Analyzer // [Электронный ресурс] URL: https://clang-analyzer.llvm.org/; (дата обращения 25.04.2024).
11. Абстрактное синтаксическое дерево // [Электронный ресурс] URL: Абстрактное синтаксическое дерево — Википедия (wikipedia.org); (дата обращения 25.04.2024).
12. Pycosat [Электронный ресурс] // URL: https://pypi.org/project/pycosat/; (дата обращения 25.04.2024).

TELEGRAM-БОТ ДЛЯ ЗНАКОМСТВ ПО ИНТЕРЕСАМ

Серен А.А., Пахомова Е.Г.
Томский государственный университет
dehaekx@gmail.com, pakhomovaeg@yandex.ru

Введение

В современном мире социальные сети и мессенджеры играют значительную роль в повседневной жизни, облегчая коммуникацию и обмен информацией. Так, в России очень большой популярностью пользуется мессенджер Telegram. Обусловлено это тем, что он имеет высокий уровень безопасности, широкие функциональные возможности и прост в использовании. В частности, Telegram позволяет достаточно легко создавать ботов. Telegram-боты – это автоматизированные аккаунты, которые могут предоставлять пользователям различный функционал. Их создание и использование становится все более популярным способом автоматизации процессов предоставления и поиска информации.

Целью данной работы являлась разработка и реализация Telegram-бота для знакомств по интересам. Бот должен предоставлять пользователям возможность находить единомышленников и общаться с ними на основе общих интересов, делая процесс знакомства более удобным и эффективным.

В работе будут рассмотрены основные принципы построения и функционирования Telegram-ботов, методы обработки сообщений и команд, а также взаимодействие с базой данных для хранения информации о пользователях и их предпочтениях.

Использование бота для знакомств по интересам позволит пользователям расширить круг общения, найти новых друзей и партнеров для общих увлечений. Кроме того, разработка такого бота способствует освоению современных технологий и практическому применению знаний в области программирования и разработки социальных приложений.

1. Стек технологий

Для разработки чат-бота был выбран язык программирования Python [1], который отличается своей простотой, хорошей читаемостью кода и широкими возможностями в области разработки приложений. Также были использованы библиотеки:

– **aiogram –** библиотека была выбрана в связи с тем, что ее реализация является асинхронной по умолчанию, а, значит, программа будет обрабатывать информа-

цию гораздо быстрее. Также библиотека позволяет использовать машину состояний [2–4].

– **asincio** – библиотека, необходимая для асинхронной работы самой программы. Асинхронность программы, очевидно, улучшит ее производительность [5].
– **dotenv** – библиотека, позволяющая загружать переменные окружения из файла .env. Это удобно и безопасно, т.к. позволяет избежать хранения конфиденциальных данных в открытом коде.
– **asyncpg** – высокопроизводительная библиотека для асинхронного доступа к PostgreSQL. Это важно для эффективного взаимодействия бота с базой данных при выполнении запросов и хранении информации.
– **pillow** – библиотека для обработки изображений, что позволяет боту оперативно обрабатывать графические данные и визуально взаимодействовать с пользователями.

2. Функционал бота

С помощью команды /start мы начинаем общение с ботом (рис. 1). Пользователь нажимает кнопку "Начать", далее происходит процесс регистрации.

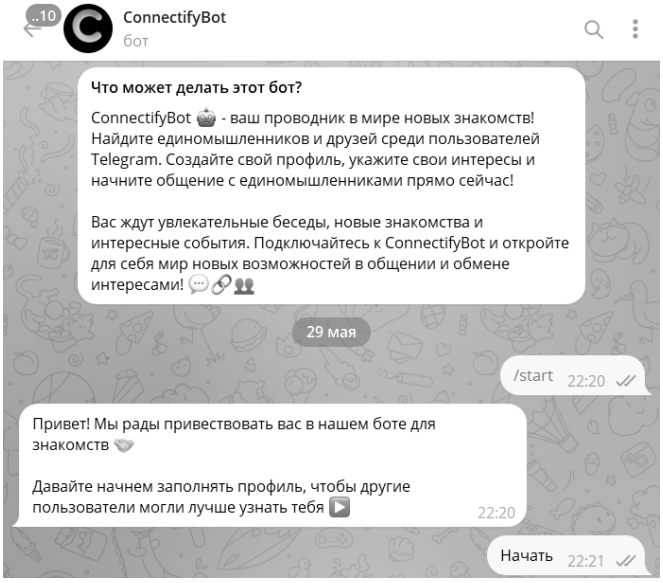


Рис. 1. Стартовое окно бота

Этап регистрации. Для начала использования бота пользователь должен пройти процесс регистрации. Необходимые шаги:

1. Пользователь запускает бота и нажимает кнопку "Начать".
2. Бот запрашивает у пользователя информацию о его имени, возрасте, городе проживания, информацию о себе, выбор своего пола, выбор, кто интересен, а также блок с интересами, который может выбрать пользователь (рис. 2, 3).
3. Пользователь нажимает кнопку "Подтвердить" после того, как прошел блок с интересами.
4. Пользователь отправляет свою фотографию или видео.

Далее вам приходит сообщение о том, что ваша анкета заполнена, с просьбой ожидать подтверждения от администраторов.

Полученные сведения сохраняются в базе данных и используются в алгоритме поиска подходящих собеседников. Сформированный профиль пользователь может в дальнейшем посмотреть и, в случае необходимости, скорректировать.

Рис. 2. Процесс регистрации

Рис. 3. Процесс регистрации, выбор интересов

Регистрация – важный этап, который позволяет боту предлагать пользователю наиболее подходящие варианты для знакомства.

Этап верификации анкеты. После завершения процесса регистрации администратору приходит письмо с базовой информацией о пользователе: имя, возраст, город, фото (рис. 4). Администратор верифицирует аккаунт нового пользователя или отправляет пользователю сообщение о необходимости повторного заполнения анкеты. Это позволяет убедиться в подлинности пользователей и предотвратить создание фейковых аккаунтов.

Рис. 4. Верификация профиля

После верификации пользователю приходит письмо, что его профиль успешно подтвержден, и он может полноценно пользоваться ботом. У пользователя появляются две кнопки: "Анкеты" и "Посмотреть профиль" (рис. 5).

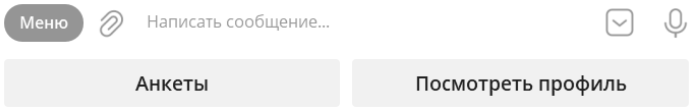


Рис. 5. Просмотр анкет или профиля

Рассмотрим подробнее информацию и возможные действия при выборе каждой из этих кнопок.

Мой профиль. В профиле можно видеть имя, возраст, информацию о себе, населенный пункт и 7 кнопок (рис. 6): 1) Мои друзья; 2) Изменить имя; 3) Изменить возраст; 4) Изменить фото/видео; 5) Изменить текст обо мне; 6) Изменить интересы; 7) Удалить профиль. С помощью такого функционала пользователь имеет возможность редактировать свою анкету.

Рис. 6. Мой профиль

Просмотр анкет. После процесса регистрации пользователь может начать просматривать анкеты других пользователей, отвечающих его интересам. Бот использует алгоритм, чтобы найти наиболее подходящих собеседников для данного пользователя. Процесс поиска подходящей кандидатуры для общения происходит следующим образом.

1. Алгоритм поиска определяет пользователей, у которого с вами есть хотя бы один общий интерес и совпадает выбор предпочтения по полу. Анкеты именно этих людей вы увидите при просмотре анкет (рис. 7).
2. При появлении анкеты у пользователя есть четыре варианта действий, которые реализуются нажатием соответствующих кнопок: а) лайк; б) отправить сообщение; в) дизлайк (просто пролистывается анкета); г) больше не показывать этого человека.
3. После отправки лайка или сообщения выбранному вами пользователю приходит письмо с уведомлением, что кто-то его лайкнул, и он может ответить взаимностью или пролистать человека (рис. 8).
4. Если интерес пользователей оказался взаимным, то можно добавить пользователя в друзья и начать общение (рис. 9).

Рис. 7. Просмотр анкет

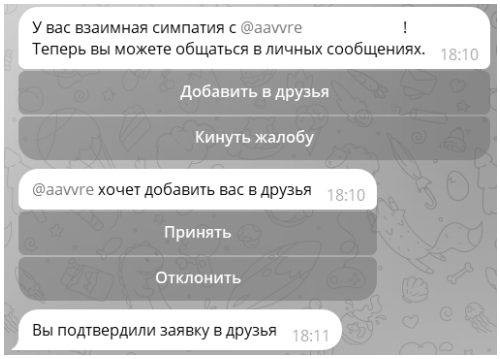


Рис. 8. Уведомление лайка

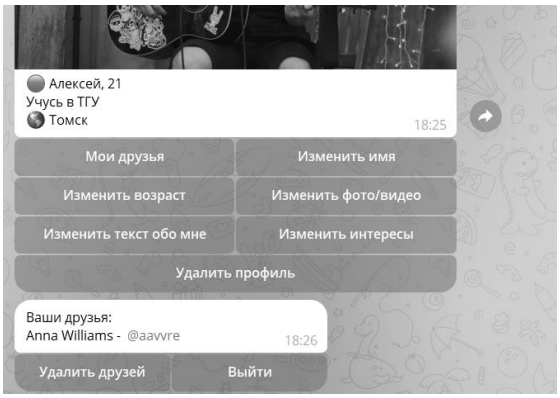


Рис. 9. Друзья

Поиск собеседников основан на анализе предпочтений и интересов пользователей, что позволяет боту предлагать качественные и интересные варианты для общения.

Панель администратора. У администратора имеются две кнопки – "Посмотреть заявки" и "Посмотреть жалобы" (рис. 10). Первая – для верификации новых пользователей, вторая – для просмотра жалоб.

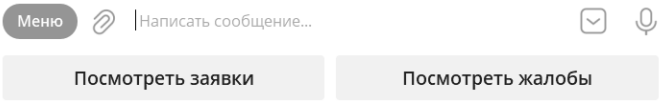


Рис. 10. Панель администратора

О процессе верификации аккаунтов уже говорилось ранее. Рассмотрим, что происходит при нажатии кнопки для просмотра жалоб. Появляется сообщение с кнопкой о том, какой пользователь пожаловался и на кого, а также причина жалобы. Администратор имеет возможность подтвердить или отклонить жалобу (рис. 11).

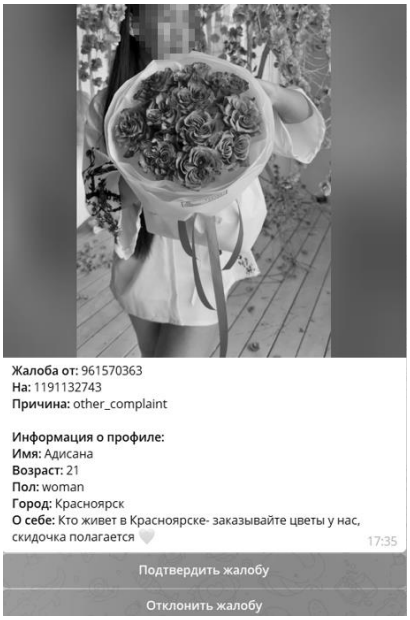


Рис. 11. Жалоба

Безопасность. Безопасность пользователей является одним из основных приоритетов бота для знакомств. При создании Telegram-бота для обеспечения безопасности были предприняты следующие меры:

1. **Верификация профилей.** Администраторы проверяют и верифицируют каждый профиль. Это позволяет убедиться в подлинности пользователей и предотвратить создание фейковых аккаунтов.

2. **Возможность подать жалобу.** Любой пользователь может пожаловаться на другого пользователя, если он считает, что его собеседник нарушает правила общения или если считает, что его активность начала носить подозрительный характер.

3. **Удаление профиля.** Пользователь имеет возможность удалить свой профиль и при необходимости создать новый. Это дает контроль над своими данными и учетной записью.

4. **Удаление друзей.** Друзья появляются только после взаимного лайка и добавления в друзья, что делает этот процесс взаимным. Однако после начала общения может оказаться, что несмотря на общность ваших интересов, собеседник вам не интересен. Тогда вы можете удалить такого пользователя из списка своих друзей и прекратить общение. Таким образом, вы всегда имеете контроль над списком своих собеседников.

Внедрение перечисленных мер безопасности помогает усилить доверие пользователей к платформе для знакомств и создать комфортное и безопасное пространство для общения. Пользователи Telegram-бота будут чувствовать себя защищенно и уверенно в процессе знакомств.

3. Архитектура проекта

🗀 **ConnectifyBot** – корневая директория всего проекта. Содержащиеся в ней файлы и папки:

1. **main.py** – основной исполняемый файл (точка входа в бот);
2. **.env** – файл с переменными окружения для конфигурации бота;
3. **requirements.txt** – список, установленных пакетов;
4. 🗀 **config_data** – директория с модулем конфигурации бота:
   4.1. **config.py** – модуль для конфигурации бота;
5. 🗀 **handlers** – пакет с обработчиками событий:
   5.1. **admin_handlers.py** – модуль с хэндлерами, срабатывающими на действия пользователя, если он является администратором бота;
   5.2. **user_handlers.py** – модуль с хэндлерами пользователей. Все основные обработчики апдейтов находятся здесь.
6. 🗀 **keyboards** – пакет с модулями, где хранятся клавиатуры бота:
   6.1. **keyboards.py** – модуль с клавиатурами для работы с ботом.
7. 🗀 **utils** – пакет для работы с базой данных и скриптами:
   7.1. **database.py** – база данных и функции;
   7.2. **Image_collage.py** – пакет, где создается коллаж;
   7.3. **maps.py** – список городов и интересов;
   7.4. **states.py** – группа состояний для регистрации пользователя;
   7.5. **scripts.py** – пакет для скачки фотографий и видео.

Рассмотрим основные файлы проекта.

Файл main.py. В файле main.py находятся основная функция main() и две асинхронные функции bot_task() и update_data(), которые запускают работу бота и сбор данных с сайта каждые 10 минут соответственно.

Файл .env. В этом файле находятся переменные окружения для конфигурации бота, а именно – токен, хост, название базы данных, пароль, порт, ссылка на базу данных.

Файл config.py. В файле config.py загружаются все переменные окружения с помощью библиотеки dotenv, который делает это удобно и безопасно.

Файл admin_handlers.py. В этом файле находятся все хэндлеры, которые нужны администраторам бота, такие как панель администратора, все обработчики событий, связанные с дальнейшими действиями.

Файл user_handlers.py. В этом файле находятся все основные обработчики событий, связанные с пользователями: весь процесс регистрации и записи всех данных в базу данных, процесс редактирования своего профиля, показа новых анкет и т.д.

Файл keyboards.py. В этом файле написаны все клавиатуры, которые используются в боте: выбор пола при регистрации, выбор интересов, просмотр своего профиля и т.д.

Пакет database.py. В этом пакете создается база данных, и далее в ней – все необходимые таблицы: users, messages, likes, dislikes, photos, notifications, complains и др. Также там находятся все необходимые функции с запросами к базе данных, чтобы извлекать необходимую информацию.

Заключение

Telegram-бот для знакомств по интересам открывает новые возможности для эффективного поиска партнеров, друзей и единомышленников. Используя современные технологии, мы стремимся сделать процесс знакомства приятным и плодотворным для каждого пользователя.

ЛИТЕРАТУРА

1. Руководство по Python [Электронный ресурс] // URL: https://docs-python.ru (дата обращения: 20.05.2024).
2. Документация библиотеки aiogram [Электронный ресурс] // URL: https://docs.aiogram.dev/en/latest/ (дата обращения: 20.05.2024).
3. *Крыжановский М.* Телеграм-Боты на Python и Aiogram [Электронный ресурс] // URL: https://stepik.org/lesson/744814 (дата обращения: 20.05.2024).
4. Документация API Telegram [Электронный ресурс] // URL: https://core.telegram.org/bots/api (дата обращения: 20.05.2024).
5. Документация библиотеки asyncio [Электронный ресурс] // URL: https://docs.python.org/3/library/asyncio.html (дата обращения: 20.05.2024).

РАЗРАБОТКА ANDROID-ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ ФИНАНСАМИ

Хомяков Д.А., Самохина С.И.
Томский государственный университет
homyakov.denis2021@yandex.ru, sv.sam.tsk@gmail.com

Введение

Информационные системы играют ключевую роль в современном мире, предоставляя пользователям возможность обрабатывать и анализировать большие объемы данных. Одной из наиболее востребованных областей применения информационных систем является сфера финансового управления. В связи с этим, разработка приложений для управления финансами становится особенно актуальной задачей. Пользователи хотят иметь возможность отслеживать свои расходы и доходы в любое время и в любом месте.

В последнее время идет активный рост рынка мобильных приложений, появляется все больше и больше программных продуктов, используемых в различных областях. Люди переходят с компьютера на мобильные устройства, и, как следствие, мобильные приложения становятся неотъемлемой частью нашей жизни. Все больше людей обращаются к мобильным приложениям не только для решения рабочих и учебных задач, но также и для упрощения повседневной жизни. В сфере мобильных финансовых приложений также появилось огромное количество программных продуктов, которые по-

могают людям проводить финансовые операции и сделки, вести бюджет, планировать покупки и т.д.

В данной работе представлена разработка Android-приложения для управления финансами, которое предоставляет пользователям широкий функционал для удобного отслеживания как личных, так и семейных денежных средств, а также планирование бюджета. Разработанное приложение отличается от существующих аналогов простотой и интуитивно понятным интерфейсом, а также широкими функциональными возможностями.

1. Обзор существующих приложений для учета финансов

Для того, чтобы дать более полное представление о существующих приложениях для учета финансов и выявить наиболее востребованные функциональные возможности, было проведено исследование рынка мобильных приложений. В ходе исследования были рассмотрены отзывы и оценки пользователей, а также проанализированы статистические данные о загрузках приложений.

На основании проведенного исследования выбраны 5 наиболее популярных приложений для учета финансов, которые имеют высокие оценки пользователей и большое количество загрузок в магазине приложений Play Market. Эти приложения были подвергнуты детальному анализу с целью выявления их достоинств и недостатков, а также определения наиболее востребованных функциональных возможностей.

Ни одно из рассмотренных приложений не обладает обширными функциональными возможностями, которые бы полностью удовлетворяли потребностям всех пользователей.

Таблица 1

Результат сравнения существующих приложений для учета финансов

	Money Lover	Денежный менеджер	Money Manager Expense & Budget	Быстрый Бюджет	Дзен-мани
Учет доходов и расходов	да	да	да	да	да
Статистика	да	да	да	да	да
Необходимые расходы	нет	нет	нет	да	да
Цели	нет	нет	нет	нет	нет
Накопления	нет	нет	да	нет	нет
Совместный бюджет	нет	нет	да	нет	да

Поэтому при разработке нового приложения для учета финансов необходимо учитывать наиболее востребованные функциональные возможности, а также предоставлять пользователям уникальные и полезные особенности, которые отличают его от других приложений на рынке.

2. Практическая реализация мобильного приложения для учета финансов

В этом разделе представлена практическая реализация мобильного приложения для учета финансов, которая включает в себя выбор языка программирования и среды разработки, создание базы данных, а также разработку и тестирование приложения.

2.1. Выбор языка программирования и среды разработки

Существует множество языков программирования, подходящих для данного вида разработки, однако для текущей реализации был выбран язык программирования Java. Java является одним из официальных языков программирования для разработки Android-приложений. В дополнение к этому, Java – кроссплатформенный язык программирования, и это означает, что приложения, написанные на этом языке, могут работать на любых устройствах, поддерживающих платформу Android. Это особенно

важно для мобильных приложений, которые должны работать на различных устройствах, таких как смартфоны, планшеты и другие устройства.

Для реализации мобильного приложения на языке программирования Java выбрана среда разработки Android Studio, которая является официальной средой разработки Android-приложений. Данная среда обладает рядом преимуществ, таких как предоставление всех необходимых инструментов для написания кода на Java, включая редактор кода, отладчик и профайлер. Кроме того, среда включает в себя визуальный редактор интерфейсов, что значительно облегчает процесс создания интерфейса и ускоряет разработку приложения. Более того, среда предоставляет инструменты для тестирования и отладки приложения, включая эмулятор Android, который позволяет запускать приложение на виртуальном устройстве, что упрощает тестирование и отладку разрабатываемого приложения.

2.2. Создание базы данных мобильного приложения

Для создания мобильного приложения для учета финансов необходимо разработать базу данных, которая хранит информацию о доходах и расходах, финансовых целях, накоплениях и регулярных платежах пользователя.

Для разработки мобильного приложения для учета финансов выбрана библиотека RoomDatabase, которая представляет собой абстракцию над SQLite – встроенной реляционной базой данных в Android [1]. SQLite является надежной и эффективной базой данных, однако работа с ней может быть достаточно трудоемкой, поскольку требует написания соответствующих SQL-запросов. RoomDatabase представляет более простой способ работы с базой данных, поскольку использует аннотации для определения схемы базы данных и автоматически генерирует код для доступа к базе данных [2].

В разрабатываемом приложении использована Room версии 2.6.1. Для создания базы данных определены сущности, которые представляют таблицы в базе данных: Accumulation, Expense, Goal, Income и RegularPayment. Каждая сущность имеет свои столбцы, которые определяют тип и формат данных, которые будут храниться в таблице (рис. 1).

Рис. 1. Диаграмма сущностей приложения для учета финансов

Далее определены DAO (Data Access Object) интерфейсы для каждой сущности, которые предоставляют методы доступа к базе данных. Эти методы позволяют выполнять операции с данными, такие как добавление, изменение, удаление и извлечение данных из базы данных (рис. 2).

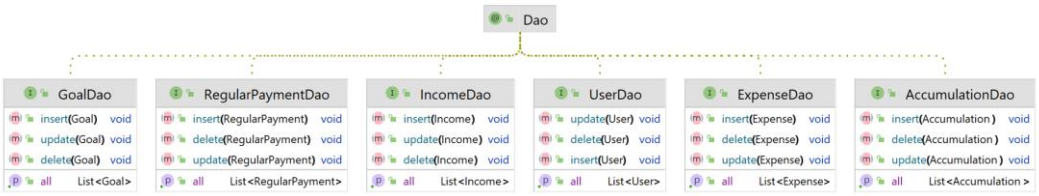


Рис. 2. Диаграмма DAO интерфейсов приложения для учета финансов

Созданная база данных включает в себя сущности и DAO-интерфейсы, она представляет собой класс, который расширяет RoomDatabase. Для лучшего понимания структуры и взаимосвязи таблиц представлена диаграмма базы данных (рис. 3).

Рис. 3. Диаграмма базы данных приложения для учета финансов

2.3. Разработка мобильного приложения

Для обеспечения обратной совместимости с более старыми версиями Android использована библиотека androidx.appcompat:appcompat:1.6.1; это позволяет приложению работать на широком спектре устройств.

Для создания макетов пользовательского интерфейса использована библиотека androidx.constraintlayout:constraintlayout:2.1.4, которая позволяет создавать сложные макеты, что упрощает и ускоряет процесс разработки. Кроме того, задействована библиотека com.google.android.material:material:1.11.0 для использования материального дизайна в приложении.

Для тестирования приложения использованы такие библиотеки, как junit:junit:4.12.2 для модульного тестирования, androidx.test.ext:junit:1.1.5 для тестирования на устройствах Android и androidx.test.espresso-core:3.5.1 для тестирования пользовательского интерфейса.

Приложение имеет следующие параметры: compileSdk 34 (Android 14) – версия, использованная при сборке, targetSdk 34 (Android 14) – целевая версия, minSdk 24 (Android 7) – минимальная поддерживаемая версия. Это означает, что приложение будет работать на устройствах с API уровня 24 и выше, и оптимизировано для работы на устройствах с API уровня 34 [3].

Для отображения информации разработаны классы, которые представляют экраны приложения. Каждый класс отвечает за определенный экран, например, класс RegularPaymentActivity отображает регулярные платежи, а AccumulationActivity – финансовые накопления (рис. 4).

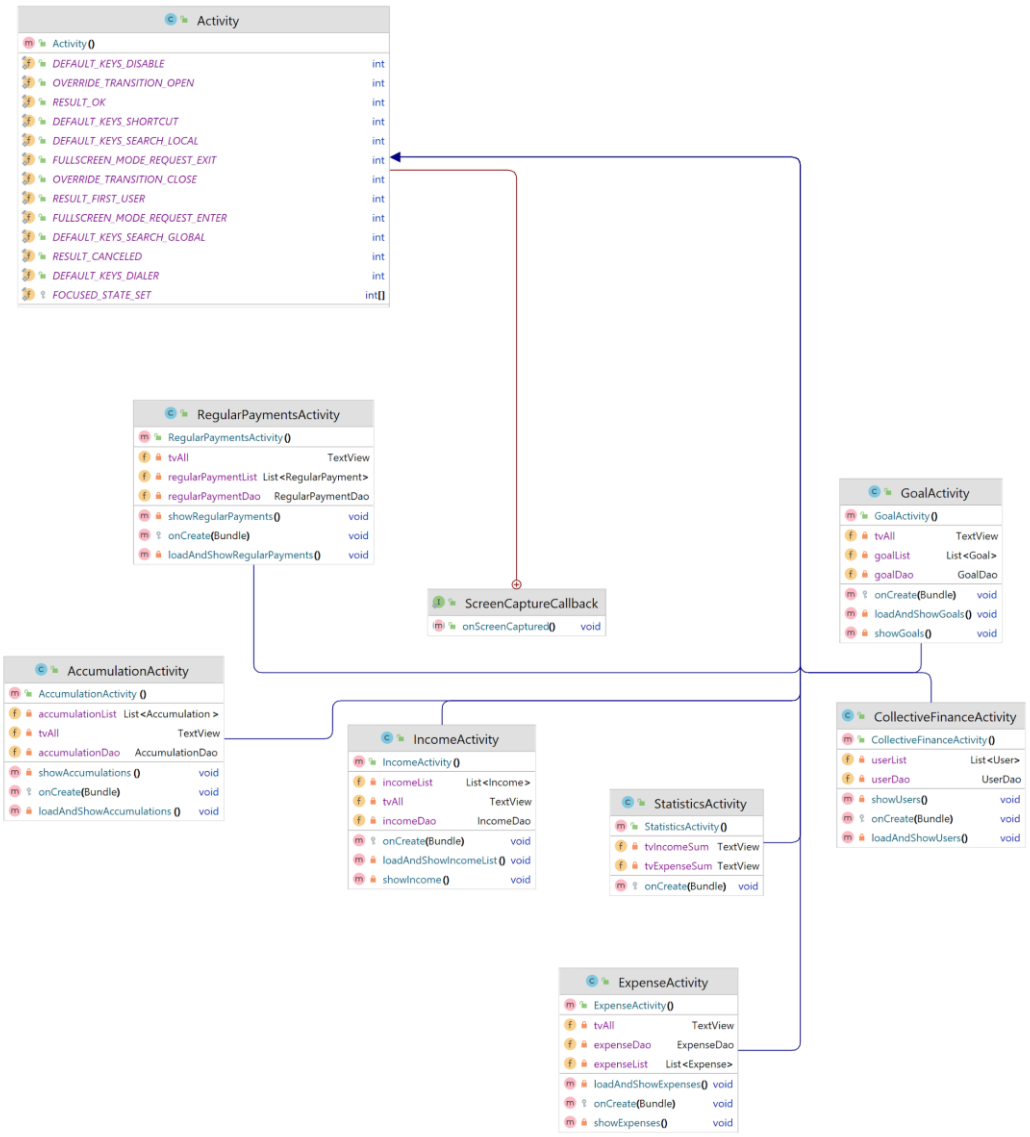


Рис. 4. Диаграмма экранов приложения для учета финансов

Для редактирования экранов приложения разработаны соответствующие классы, каждый из которых отвечает за определенный экран и содержит логику для обработки взаимодействия пользователя с экраном, например, класс AddEditIncomeActivity отвечает за добавление и редактирование доходов (рис. 5).

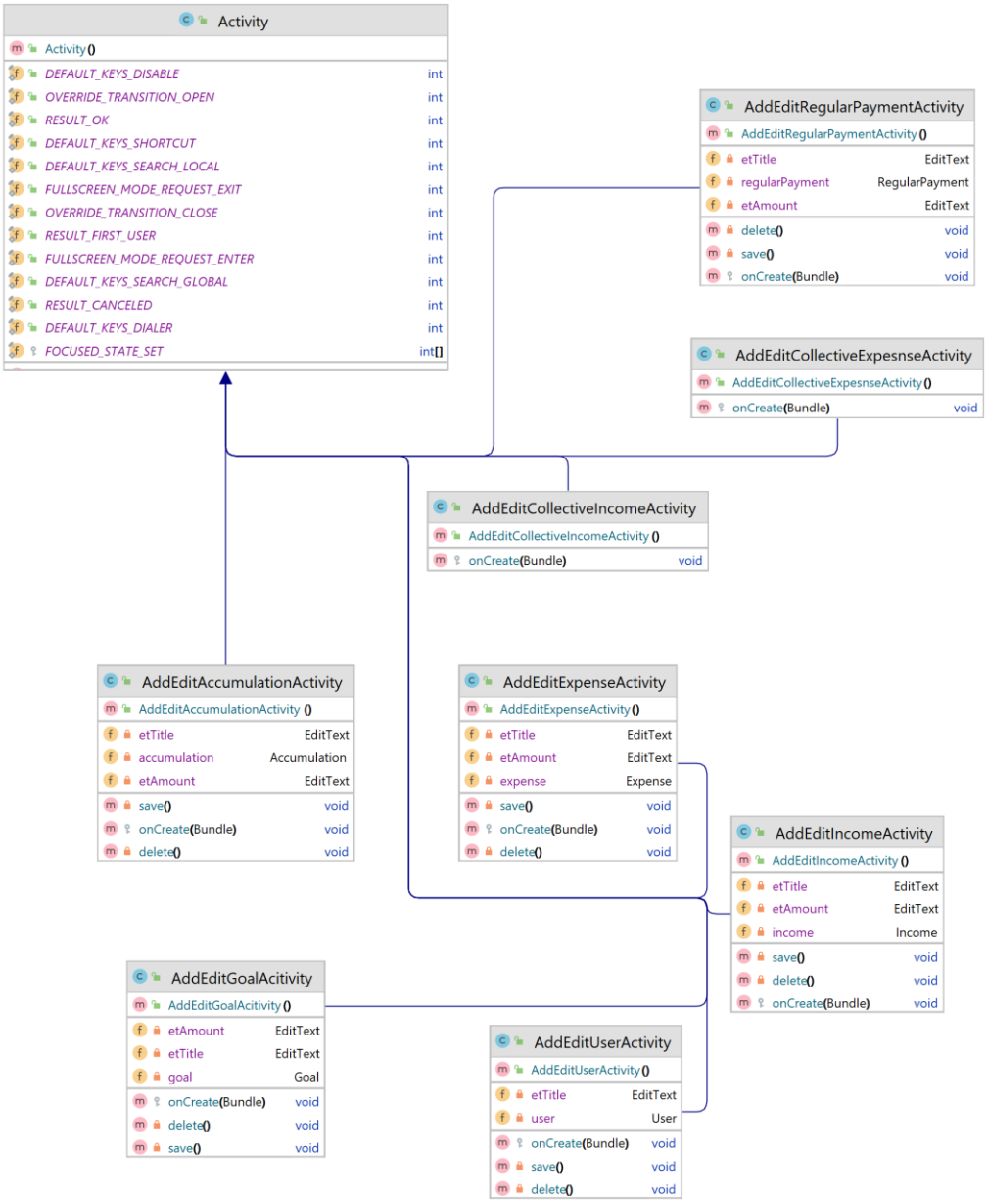


Рис. 5. Диаграмма добавления и редактирования экранов приложения для учета финансов

Имеются адаптеры, отвечающие за отображение списков данных в приложении. Например, класс AccumulationAdapter отвечает за отображение списка финансовых накоплений, а GoalAdapter – списка финансовых целей. Кроме того, имеются интерфейсы OnItemClickListener для обработки события нажатия и классы ViewHolder, отвечающие за предоставление списка элементов (рис. 6).

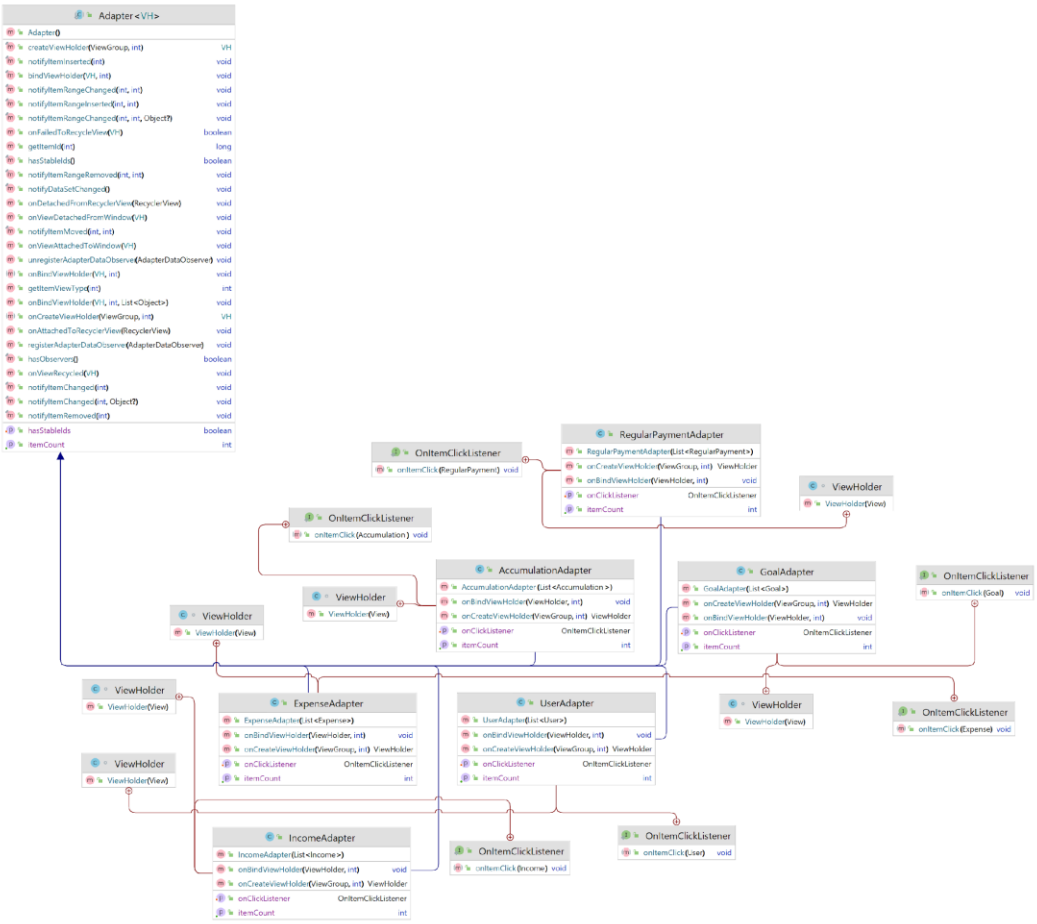


Рис. 6. Диаграмма адаптеров приложения для учета финансов

2.4. Тестирование мобильного приложения

Тестирование мобильного приложения является неотъемлемой частью разработки, которая позволяет проверить его функциональность, производительность и надежность, а также выявить и исправить любые ошибки и недостатки.

В среде разработки Android Studio разработчик мобильных приложений имеет возможность выбрать любую из версий операционных систем Android, которые были созданы. После выбора желаемых версий производится загрузка и установка соответствующих SDK.

Далее необходимо выбрать целевое тестовое устройство, на котором будет запускаться приложение. Это может быть как физическое устройство, так и эмулятор. После выбора устройства запускается сборка и отладка проекта. Переходы между экранами и срабатывание нажатия по кнопкам свидетельствуют о корректности исходного кода и работоспособности программы.

Кроме того, приложение тестировалось вручную с помощью .apk файла на различных устройствах, таких как Xiaomi 13 Lite и Xiaomi Pad 6, и всё сработало корректно. Это позволило проверить, как приложение будет работать на разных устройствах с различными размерами экранов и характеристиками.

2.5. Демонстрация мобильного приложения

Главным экраном приложения является экран со статистическими данными расходов и доходов пользователя (рис. 7). При нажатии на расходы или доходы появляется

возможность добавить соответствующую финансовую операцию, а также редактировать добавленные ранее финансовые операции.

Рис. 7. Главный экран приложения для учета финансов

На экране регулярных платежей пользователь имеет возможность просматривать общую сумму регулярных платежей, редактировать существующие и добавлять новые (рис. 8).

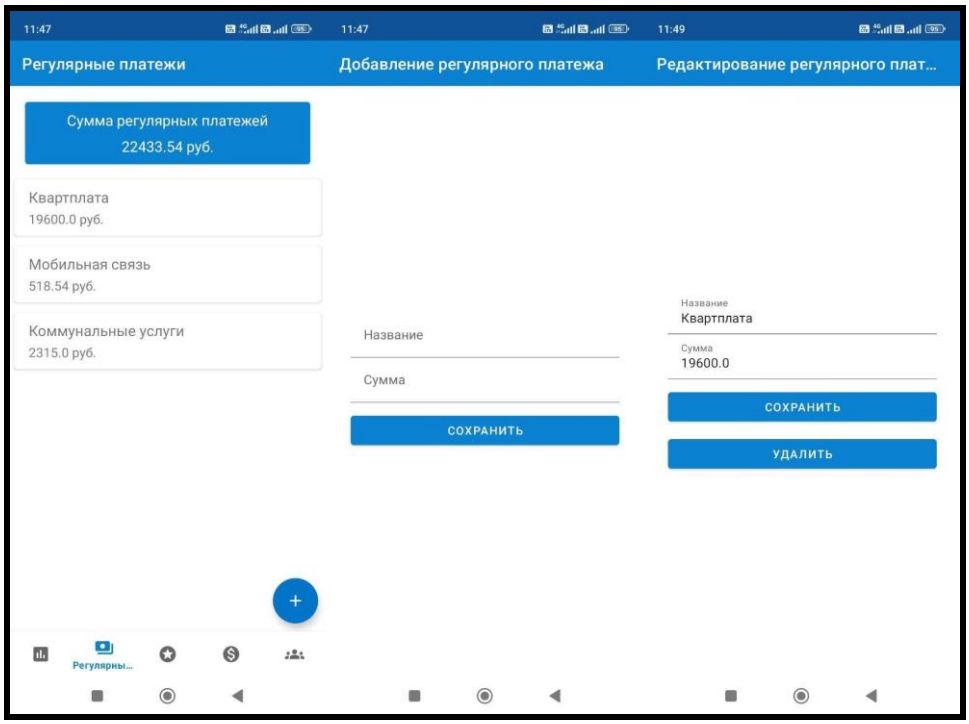


Рис. 8. Экран регулярных платежей приложения для учета финансов

Экран финансовых целей также предоставляет пользователю возможность просматривать общую сумму финансовых целей, редактировать существующие и добавлять новые (рис. 9).

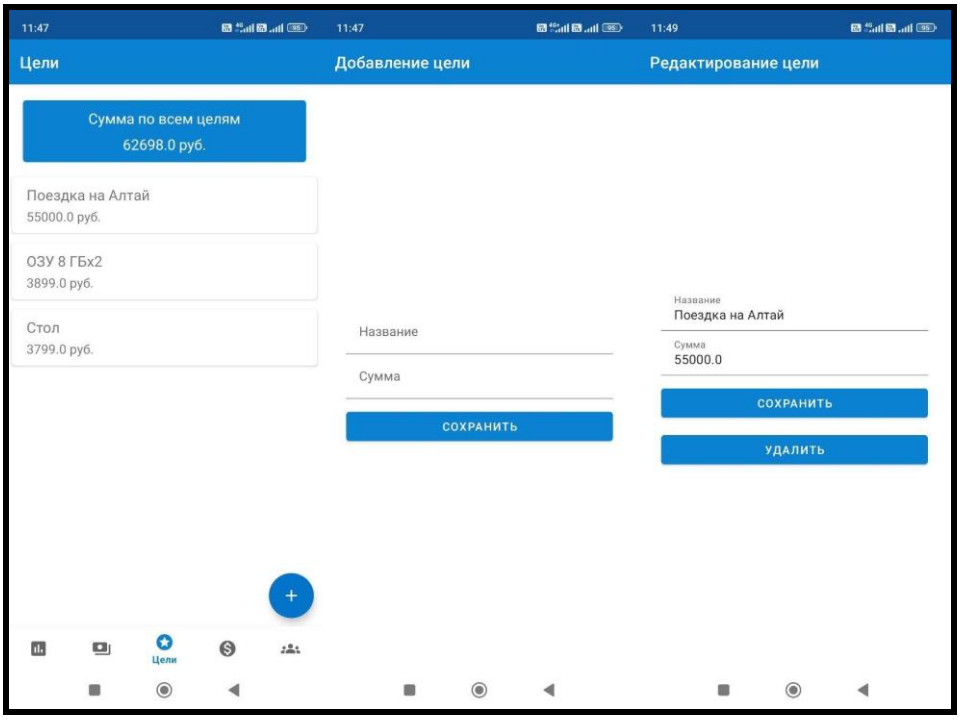


Рис. 9. Экран финансовых целей приложения для учета финансов

Заключение

В настоящей статье ставилась задача разработать приложение для мобильной платформы Android, которое позволяет пользователям эффективно контролировать и планировать свой бюджет. Приложение представляет надежный и удобный инструмент для отслеживания доходов и расходов, контролирования регулярных платежей, ведения финансовых целей и накоплений, а также возможность вести совместный бюджет.

В процессе разработки был проведен тщательный анализ аналогов, определены функциональные требования, выбраны инструменты и средства для программной разработки, создана общая модель приложения и его отдельных частей, произведена реализация приложения на основании спроектированной модели, а также выполнены тестирование и отладка.

В перспективе планируется развитие приложения, добавление возможности отслеживать статистику финансовых операций за определенные промежутки времени, улучшение дизайна, доработка таких функциональных возможностей, как финансовые цели и совместное ведению бюджета.

ЛИТЕРАТУРА

1. Android Room with a View – Java. URL: https://developer.android.com/codelabs/android-room-with-a-view#0
2. SQlite Java – How To Use JDBC To Interact with SQLite. URL: https://www.sqlitetutorial.net/sqlite-java/
3. API Levels | Android versions, SDK/API levels, version codes, codenames, and cumulative usage. URL: https://apilevels.com/

БЕЗОПАСНАЯ ИНФОРМАЦИОННАЯ СИСТЕМА "ДНЕВНИК СТУДЕНТА"

Шокаримов Ш.Ш., Самохина С.И.
Томский государственный университет
sv.sam.tsk@gmail.com

Введение

В сегодняшнем мире информационные технологии играют ключевую роль в жизни людей. Информационные системы создаются в разных направлениях человеческой деятельности [1,2]. Одной из основных задач студента является эффективное планирование и организация учебного процесса. Особенно перед контрольной неделей, сессией, когда нужно сделать много учебных заданий и других важных дел. Решением этой проблемы является разработка веб-систем "Дневник студента", которая позволяет делать заметки, планировать задачи и помогать установить приоритет между задачами, а также хранить информацию в безопасном режиме.

Проведенный анализ существующих решений показал, что на данный момент имеются приложения для планирования задач и ведения заметок, например: PlanNote, Todoist, Things и Evernote. Однако наша система будет иметь уникальные функции, например, реализация матрицы Эйзенхауэра для управления задачами и улучшения информационный безопасности.

Основные функции системы включают в себя регистрацию и авторизацию, создание списков задач, создание и редактирование заметок, разделение по секциям, добавление тегов и приоритетов, защиту от SQL-инъекций, поиск и фильтрацию, а также интеграцию с другими приложениями.

1. Архитектура информационной системы

Информационная система "Дневник студента" использует интерфейс прикладного программирования передачи репрезентативного состояния (API), шаблон проектирования, который использует протокол передачи гипертекста (HTTP) и формат данных но-

тации объектов JavaScript (JSON) для облегчения связи между клиентами и серверами (рис. 1).

Рис. 1. Архитектура информационной системы

В рамках этой архитектуры клиенты и серверы взаимодействуют друг с другом через набор API-интерфейсов RESTful, обеспечивая доступ к системным ресурсам. Каждый ресурс имеет отдельный идентификатор (унифицированный указатель ресурса/URL), позволяющий клиентам извлекать, изменять или удалять указанные ресурсы.

Архитектура состоит из трех уровней: уровня клиента, API и базы данных. Клиентский уровень служит интерфейсом, в котором пользователи взаимодействуют с системой, реализованным с использованием технологий HTML, каскадных таблиц стилей (CSS) и JavaScript. Этот уровень взаимодействует с уровнем API через HTTP-запросы (GET, POST, PUT, DELETE) для получения, создания, обновления или удаления ресурсов.

Уровень базы данных отвечает за хранение и обработку системных данных, реализованных с помощью MySQL. Этот уровень обеспечивает безопасное хранение, извлечение, обработку транзакций и целостность данных.

Node.js и Express.js Framework являются двумя ключевыми технологиями, используемыми для разработки серверной части информационной системы "Дневник студента". Node.js и Express.js предоставляют широкий набор инструментов для упрощения разработки веб-приложений. При создании приложения импортируются необходимые модули, включая Express для обработки HTTP-запросов, MySQL2 для взаимодействия с базой данных, а также модули для работы с файловой системой, защиты от внедрения кода и управления сеансами пользователей.

После импорта модулей создается экземпляр приложения, а также вносятся изменения в промежуточное ПО, такие как express.json() и express.urlencoded() для обработки входящих HTTP-запросов. Использование Node.js и Express.js позволяет создать масштабируемое приложение, способное обрабатывать большое количество одновременных соединений и запросов, а также обеспечивает высокий уровень безопасности.

2. Особенности информационной системы "Дневник Студента"

1. Управление задачами с помощью матрицы Эйзенхауэра: в отличие от других аналогичных приложений, данная веб-система предоставляет функцию матрицы Эйзенхауэра для управления задачами, которая позволяет студентам эффективно планировать свои задачи и определять их приоритеты.

2. Интеграция с другими приложениями: система предлагает возможность интеграции с другими приложениями по аккаунту пользователя, что облегчает синхронизацию данных и работу с несколькими приложениями.

3. Защита от SQL-инъекций: в отличие от некоторых других аналогичных приложений, предлагаемая система использует защиту от SQL-инъекций, что обеспечивает более высокий уровень безопасности данных и предотвращает несанкционированный доступ.

4. Хеширование паролей с использованием bcrypt: система использует bcrypt для хеширования паролей, что обеспечивает более безопасную аутентификацию и защиту данных пользователей.

5. Разделение заметок на секции: система предоставляет возможность разделять заметки на секции, что облегчает их поиск и организацию, в отличие от других систем, где могут возникнуть трудности с нахождением нужной информации.

3. Безопасность информационной системы "Дневник Студента"

Безопасность является одним из ключевых аспектов любого веб-приложения, и разрабатываемая информационная система "Дневник студента" не является исключением. Для защиты данных от несанкционированного доступа и манипулирования были использованы два основных подхода: защита от SQL-инъекций и шифрование паролей [3].

SQL-инъекция – это один из самых распространенных типов атак на веб-приложения, при котором злоумышленник вводит специально сформированный код в поля ввода приложения, чтобы получить несанкционированный доступ к базе данных [4]. Для защиты от этой угрозы использованы параметризованные запросы, экранирование символов и проверка входных данных. В нашем коде используется модуль mysql2, который поддерживает параметризованные запросы, что позволяет передавать значения переменных отдельно от текста запроса, предотвращая внедрение злонамеренного кода.

Для обеспечения безопасности использования паролей [5] пользователей применён алгоритм Bcrypt (рис. 2). Этот алгоритм использует соль и многократное шифрование для создания уникального хеш-пароля для каждого пользователя. В нашем случае использованы 10 итераций хеширования и 22-хсимвольная соль для защиты от предварительно вычисленных таблиц хешей. Кроме того, использован асинхронный режим Bcrypt для хеширования паролей в фоновом режиме, не блокируя цикл событий сервера.

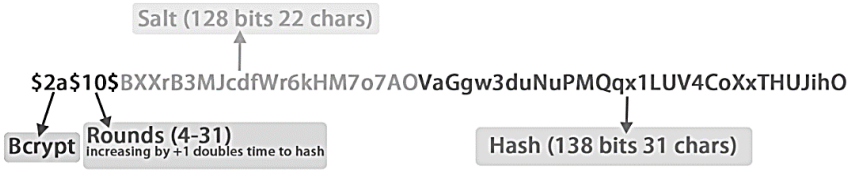


Рис. 2. Зашифрованный пароль

Для аутентификации пользователей использован модуль Passport и стратегия LocalStrategy. При попытке входа в систему вызывается функция обратного вызова, которая выполняет запрос к базе данных для проверки существования пользователя с предоставленным именем пользователя. Если пользователь найден, код проверяет, прошло ли более одной минуты с момента последнего входа пользователя. Если прошло более одной минуты, код сравнивает предоставленный пароль с сохраненным паролем пользователя с помощью функции bcrypt.compare. Если пароли совпадают, временная метка последнего входа пользователя обновляется в базе данных, и пользователь успешно авторизован.

При создании информационной системы "Дневник студента" для хранения данных была выбрана система управления базами данных MySQL. Преимуществом данной

СУБД является открытый исходный код, простота установки и использования, широкий функционал, высокая безопасность, масштабируемость и скорость работы.

Для подключения к базе данных MySQL из приложения Node.js используется библиотека mysql2. Она предоставляет надежный и эффективный способ взаимодействия с базой данных, гарантируя безопасность и целостность хранящихся данных. При подключении к базе данных используются параметры конфигурации: diary, users, section, sections. Отношения diary и section связаны с отношением users через внешний ключ owner, который ссылается на поле username в таблице users. Из этого следует, что только пользователь может имеет доступ к своим данным.

В базе данных хранятся отношения, содержащие информацию о пользователях, задачах, заметках и других данных, необходимых для работы приложения (рис. 3). Каждое отношение имеет определенную структуру, которая определяет типы и формат хранимых данных. Например, отношение пользователей содержит атрибуты с именем пользователя, электронной почтой, паролем и датой регистрации.

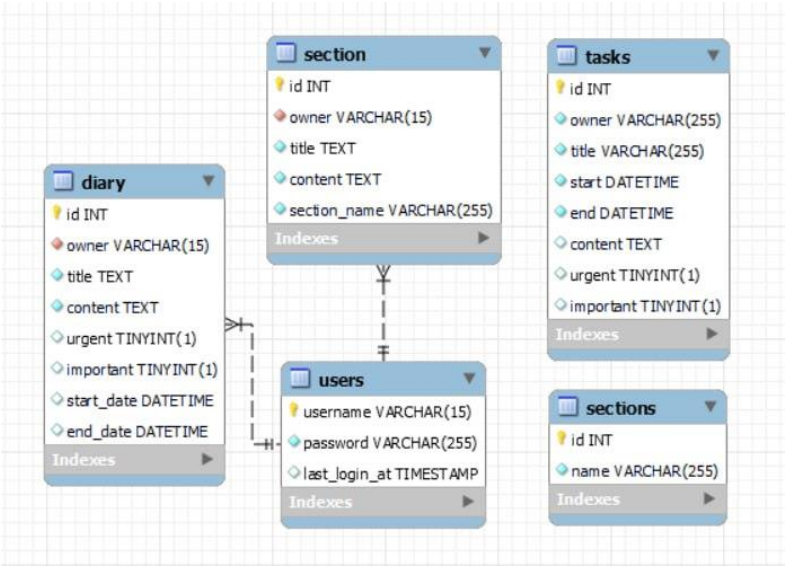


Рис. 3. Схема базы данных

4. Матричный метод Эйзенхауэра

Управление временем и приоритетами является ключевым аспектом для студентов, которые стремятся добиться успеха в учебе. Матричный метод Эйзенхауэра является одним из эффективных методов управления временем, разработанным 34-м президентом США Дуайтом Дэвидом Эйзенхауэром. Этот метод помогает определить наиболее важные и срочные задачи и распределить их по четырем категориям: "важно и срочно", "важно, но не срочно", "неважно, но срочно" и "неважно и не срочно".

В данной работе планируется реализовать этот метод в веб-системе "Дневник студента" для управления задачами и планирования времени. Студенты смогут размещать свои задачи в соответствующих категориях матрицы Эйзенхауэра, что поможет им лучше понимать, какие задачи требуют немедленного внимания, а какие можно отложить на более поздний срок. Это позволит студентам концентрироваться на важных задачах и избежать перегрузки.

Для удобства пользователей планируется добавить возможность настраивать матрицу в соответствии с их потребностями. Например, пользователь сможет выбрать количество категорий, переименовать их и настроить цвета для каждой категории. Это

позволит пользователям адаптировать матрицу Эйзенхауэра под свои нужды и предпочтения.

Кроме того, планируется реализовать функцию перетаскивания задач между категориями, что упростит процесс перепланирования и изменения приоритетов. Также будет добавлена возможность добавлять теги и комментарии к задачам, назначать даты и напоминания.

Внедрение матричного метода Эйзенхауэра в сочетании с другими функциями разрабатываемой веб-системы поможет студентам более эффективно управлять своим временем и выполнять свои обязанности. Это будет полезным инструментом для студентов, которые стремятся улучшить свои навыки управления временем и повысить свою эффективность.

Основные страницы информационной системы "Дневник студента" показаны на рис. 4–7.

Рис. 4. Главная страница "Дневника студента"

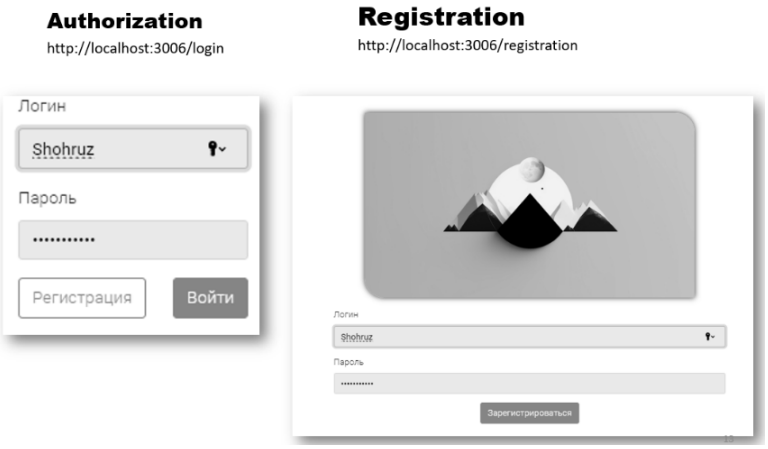


Рис. 5. Страница авторизации

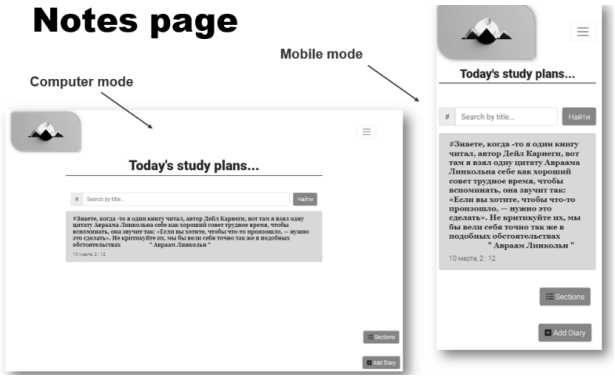


Рис. 6. Заметки

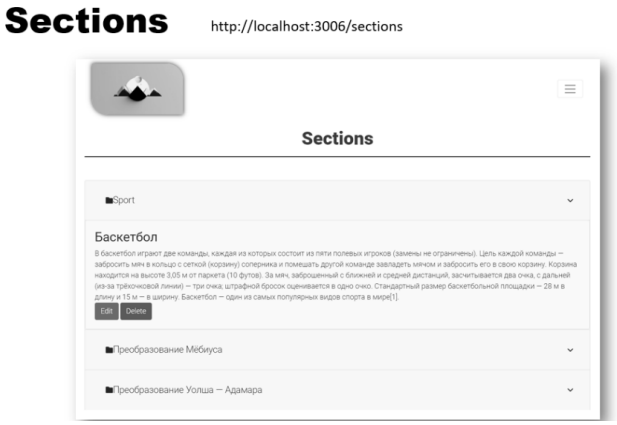


Рис. 7. Страница Sections

Заключение

Разработана безопасная информационная система "Дневник студента", которая позволяет делать заметки, планировать задачи и устанавливать приоритет между задачами, хранить информацию в безопасном режиме.

Проведен анализ существующих проектов и определены уникальные особенности нашей системы, такие как функция матрицы Эйзенхауэра для управления задачами и улучшенная безопасность. Основные функции системы включают в себя регистрацию и авторизацию, создание списков задач, создание и редактирование заметок, разделение по секциям, добавление тегов и приоритетов, защиту от SQL-инъекций, поиск и фильтрацию, а также интеграцию с другими приложениями по аккаунту. Реализована архитектура информационной системы на основе API с использованием Node.js и Express.js Framework, что позволило создать высокоэффективное и масштабируемое веб-приложение. Для хранения данных выбрана СУБД MySQL, обеспечивающая надежное и эффективное хранение и извлечение данных.

Реализованы механизмы безопасности, такие как защита от SQL-инъекций и хеширование паролей с использованием bcrypt, что обеспечивает высокий уровень безопасности данных и предотвращает несанкционированный доступ. Также планируется реализовать функции матрицы Эйзенхауэра для управления задачами, которая позволяет студентам планировать свою работу и определять приоритеты.

В целом, разработанная веб-система "Дневник студента" является удобным и надежным инструментом для студентов, который поможет им планировать учебные задачи и хранить важную информацию в безопасном режиме.

ЛИТЕРАТУРА

1. *Устинова В.А., Самохина С.И.* Информационная система анализа состояния здоровья человека "H.TOGETHER" // Математическое и программное обеспечение информационных, технических и экономических систем: материалы VI Междунар. молодежной науч. конф. Томск, 24–26 мая 2018 г. – Томск: Издательский Дом ТГУ, 2018. – С. 88–91.

2. *Нарушева К.С., Самохина С.И., Ивановская Е.В., Стуканова О.Б.* Информационная система учета движения документов в Научной Библиотеке Томского государственного университета // Инноватика–2020: сборник материалов XVI Международной школы-конференции студентов, аспирантов и молодых ученых, 23–25 апреля 2020 г. – Томск: STT, 2020. – С. 372–375.

3. Building Secure Web Applications: A Sourcebook for Developers. – Mike Shema, O'Reilly Media, 2019.

4. *Прохоренок Н., Дронов В.* HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера.

5. Тактика защиты и нападения на Web-приложения. – СПб.:БХВ-Петербург, 2005. – 432 с.

ТЕСТИРОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ПРОТОТИПА ПЛАТФОРМЫ SECUREDBMS РАЗРАБОТКИ ЗАЩИЩЕННОЙ ОБЛАЧНОЙ СУБД

Яковлев Г.А., Тренькаев В.Н.

Томский государственный университет

yakovlev-grisha@mail.ru, trenkaev.vn@mail.ru

Введение

Применение облачных технологий приводит к возникновению рисков, связанных с возможностью утечки конфиденциальных данных пользователей облачных услуг. Провайдер может корректно выполнять действия, необходимые для предоставления облачной услуги, но при этом вести наблюдения за информационными потоками. В случае облачных СУБД возможен несанкционированный доступ к информации на стороне сервера базы данных, расположенного в облачной инфраструктуре. При этом традиционное "прозрачное" шифрование базы данных не решает проблему, т.к. ключи шифрования доступны провайдеру.

Обеспечить безопасность облачных данных возможно за счет применения специальных схем шифрования, которые позволяют производить защищенные вычисления над конфиденциальными данными. Данный подход используется в ряде исследовательских проектов [1], например, CryptDB, Monomi, Arx, ZeroDB, причем для СУБД Arx показана безопасность протоколов защиты. Специализированные алгоритмы шифрования достаточно ресурсоемки, что приводит к ухудшению производительности защищенных баз данных. Требуется гибкое решение, которое при высоком уровне безопасности предоставляет необходимый для предметной области набор операций над данными и производительность, сравнимые с обычными незащищенными СУБД.

В [2] представлена архитектура открытой платформы SecureDBMS для разработки защищенных облачных СУБД. Платформа SecureDBMS предоставляет инструмент, позволяющий построить на базе целевой СУБД ее защищенный аналог. При этом имеется возможность адаптировать выбор криптографических алгоритмов под требуемый уровень безопасности и для заданной предметной области.

В данной работе представлены результаты реализации и тестирования задержек базовых модулей SecureDBMS (парсер, прокси, сетевой протокол), предложены варианты улучшения производительности платформы.

1. Архитектура платформы SecureDBMS

В [2] подробно описана архитектура SecureDBMS, объяснена логика взаимодействия компонент платформы, расписаны требования к модулям шифрования, представлено "движение" данных внутри платформы (запросов от клиента на сервер и результатов исполнения обратно). Главная особенность платформы – для обеспечения конфиденциальности SQL-запросы выполняются над зашифрованными данными. Информа-

ция об обрабатываемых данных раскрывается в той мере, которая требуется для выполнения запроса.

На рис. 1 представлена архитектура платформы SecureDBMS, которая является расширением типовой архитектуры клиент-серверной СУБД. Добавляются прокси-компоненты как на стороне клиента, так и на стороне сервера. Так, на стороне клиента происходит выявление SQL-запроса и передача его на "Распознаватель запросов", который строит дерево разбора. Главной задачей распознавателя является вычленение важных сущностей, например, название таблиц и столбцов, псевдонимы, текстовые и числовые константы. "Менеджер идентификаторов" отслеживает данные, которые требуется шифровать. "Модуль шифрования" и "Менеджер шифрования" отвечают за шифрование данных и логику совместного использования разных шифров. "Менеджер ключей шифрования" и "Менеджер хранилища" ведут работу с хранением и управлением секретами и служебными данными.

Рис. 1. Архитектура платформы SecureDBMS

Платформа SecureDBMS – это единая настраиваемая среда для создания защищенных облачных СУБД, которая обладает следующими базовыми характеристиками. Во-первых, платформа позволяет реализовывать различные архитектурные варианты построения СУБД: без поддержки / с выборочной поддержкой специальных схем шифрования; с одиночным клиентским прокси; с двойным прокси (клиентским и серверным). Во-вторых, платформа обеспечивает открытый доступ к коду и инструментам, что делает ее гибкой и масштабируемой. В-третьих, платформа позволяет разработчикам внедрять собственные модули шифрования и логику их взаимодействия, что обеспечивает высокую степень конфиденциальности и безопасности данных.

На данный момент прототип платформы предполагает в качестве целевой СУБД использование MySQL. Работа по протоколу MySQL происходит за счет клиентских библиотек на языках Go и C++. "Распознаватель запросов" построен на базе инструмента для генерации парсеров ANTLR. Применение ANTLR, использующего актуальные грамматики языка SQL, позволяет быстро адаптировать "Менеджер шифрования" к особенностям работы MySQL.

2. Методика тестирования и особенности реализации

В работе представлены результаты тестирования производительности платформы SecureDBMS в разных конфигурациях (с парсером, без парсера) без задействования клиентских и серверных модулей шифрования. Выявляется эффект "замедления" вы-

полнения SQL-запросов при "включении/выключении" таких базовых модулей, как парсер и сетевой протокол. Для проведения тестов реализован прототип платформы с интеграцией сбора метрик во все компоненты.

При тестировании используется запуск 500 SQL-запросов четырех типов: INSERT, UPDATE, SELECT, DELETE. Все запросы выполняются над одной таблицей для тестов со случайно сгенерированными данными. Запросы типа INSERT, UPDATE, DELETE работают с одной записью в таблице (простые операции записи), а запросы типа SELECT обрабатывают большую часть данных таблицы, представляя собой усиленную нагрузку на чтение данных. В дальнейшем планируется использовать заготовленные запросы (prepared statements), которые часто применяются в реальных приложениях.

Особенности реализации. Используется кеширование сессий пользователя (в случае повторных запросов не открывается новое соединение), что снижает задержку выполнения запросов в рамках работы одного пользователя. Реализовано неблокирующее асинхронное выполнение запросов, что позволяет не ждать выполнения "долгих" запросов от множества клиентов. В парсере на базе ANTLR4 может использоваться режим SLL-грамматики, что дает возможность увеличить производительности парсера в 2 раза, хотя некоторые запросы не могут быть распознаны в рамках упрощенной грамматики.

3. Результаты тестирования

Рис. 2 демонстрирует распределение времени выполнения SQL-запросов при их обработке СУБД MySQL. На графиках представлены гистограммы распределения числа запросов по времени выполнения в миллисекундах.

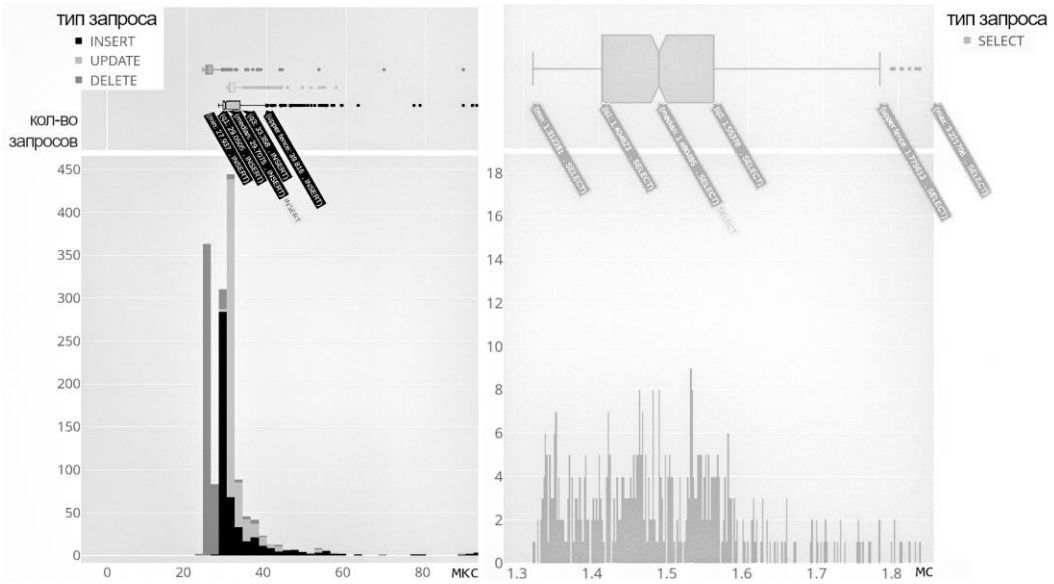


Рис. 2. Тестирование целевой СУБД MySQL

Как видно из графиков, наиболее "выделяются" запросы типа SELECT. Б*о*льшая часть запросов выполняется менее чем за 1.8 мс (SELECT), для запросов типа INSERT, UPDATE, DELETE время намного меньше – 60 мкс. Эти данные используются в качестве эталонной производительности.

На рис. 3 представлены результаты тестирования подсистемы передачи данных платформы, которая состоит из MySQL-прокси, клиента и сервера RPC, клиента MySQL. Как видно из рис. 3, время прохождения запросов по сравнению с целевой

СУБД увеличилось: для запросов INSERT, UPDATE, DELETE – в 15 раз; для запросов SELECT – в 12 раз.

Исследование показало, что подсистема передачи данных дает задержку для большинства запросов до 23 мс. В частности, MySQL-прокси вносит задержку для запросов типа SELECT в диапазоне 12–20 мс, а для остальных типов запросов – менее 2 мс. Упаковка и пересылка данных на базе gRPC вносит задержки до 10 мс, причем большие задержки обусловлены запросами типа SELECT, когда происходит упаковка множества данных, в других случаях задержки – менее 2 мс.

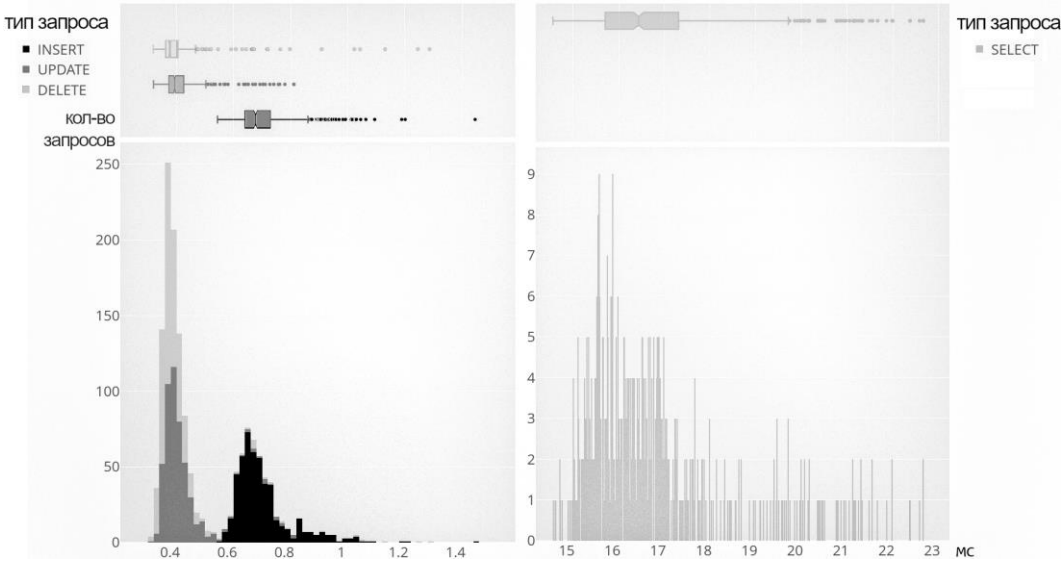


Рис. 3. Тестирование подсистемы передачи данных платформы

Также проведено тестирование платформы, когда вместе с подсистемой передачи данных функционирует "Распознаватель запросов" (парсер) (см. рис. 4). Выявлено, что клиентский прокси (парсер, клиент RPC) в 10 раз замедляет отклик СУБД MySQL. Причем основное время тратится на построение дерева разбора для SQL-запросов. В случае "отключения" парсера наблюдается значительное (в 10 раз) ускорение выполнения запросов типа INSERT, UPDATE, DELETE, а также небольшое (в 2.5 раза) ускорение выполнения запросов типа SELECT.

Таким образом, простые запросы выполняются менее чем за 10 мс, в то время как сложные – менее чем за 55 мс. Зависимость задержки от объема данных для запросов, обрабатывающих небольшое количество строк, "зашумляется" из-за использования парсера, который тратит время на распознавание при любом виде запроса, вне зависимости от размера ответа.

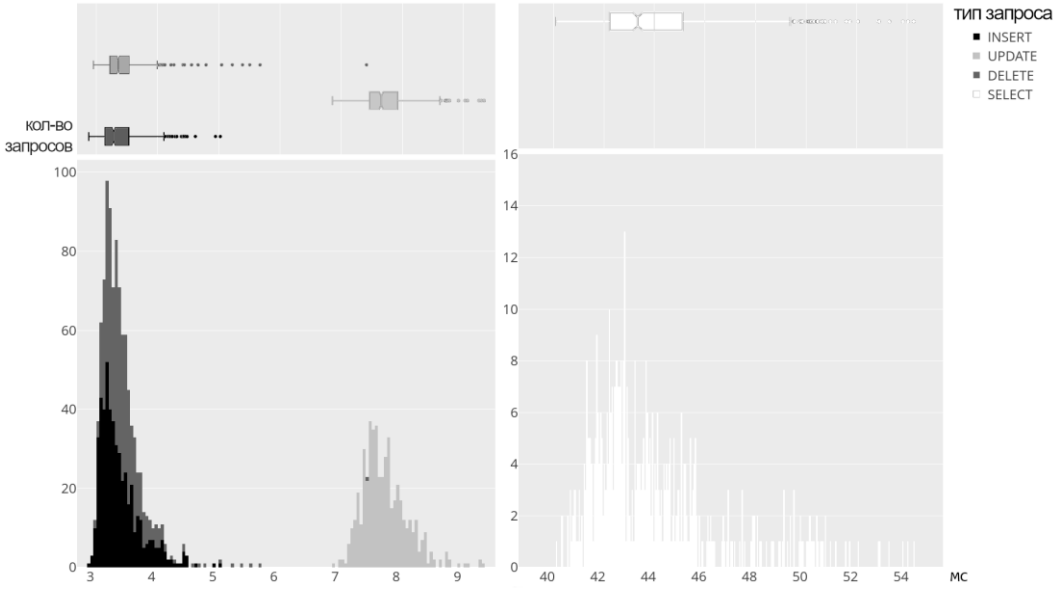


Рис. 4. Тестирование платформы с парсером

На рис. 5 можно увидеть диаграмму времени выполнения SQL-запросов, "разнесенную" по компонентам платформы, которые представлены на правой стороне рисунка. В частности, mysqlproxy – это прокси, обрабатывающий запросы по протоколу MySQL, clientproxy – это прокси, обрабатывающий запросы по собственному протоколу на стороне клиента (в дальнейшем будет выполнять операции шифрования данных), serverproxy – прокси, обрабатывающий запросы по собственному протоколу на стороне облачной СУБД (в дальнейшем будет производить операции над зашифрованными данными).

Рис. 5. Диаграмма времени выполнения SQL-запросов

Из рис. 5 видно, что клиентский прокси (clientproxy) успевает обработать простые запросы менее чем за 10 мс и сложные – менее чем за 40 мс, в то время как MySQL-прокси (mysqlproxy) – простые – так же, как clientproxy, а сложные – менее чем за 50 мс. Представленные на рис. 5 метрики показывают, что бо́льшая часть времени выполнения SQL-запросов тратится на распаковку/упаковку и изменение структур данных, а также последующее ожидание запроса в очереди асинхронной обработки.

В результате нагрузочного тестирования платформы выявлено, что процедура построения дерева разбора для SQL-запросов (парсер на базе ANTLR) существенно влияет на производительность, добавляя задержки до 7 мс на запросы INSERT, UPDATE, DELETE, и 16–20 мс – на запросы SELECT, что сравнимо с задержками, вносимыми преобразованием данных (протокол gRPC и MySQL).

Выводы. Использование платформы замедляет отклик целевой СУБД MySQL 1) для простых запросов, включающих мало данных, в 120 раз (при этом запросы выполняются меньше чем за 10 мс), 2) для "тяжелых" запросов, обрабатывающих много данных, – в 20 раз (при этом запросы выполняются менее, чем за 40 мс). Медленная работа парсера в большей степени сказывается на выполнении простых запросов. Протокол gRPC и упаковка данных в Protobuf "съедает" время выполнения тяжелых запросов. Имеется необходимость дальнейшего изучения взаимодействия компонент платформы с целью снижения издержек – сокращения операций преобразования данных, сокращения объема передаваемых данных.

4. Предложения по повышению производительности

Возможно повышение производительности платформы за счет оптимизации или замены парсера, а также замены протокола gRPC на Cap'n Proto [3]. Существует возможность повышения производительности сетевого протокола за счет: 1) улучшения взаимодействия языков Go и C++ (в случае работы на одной машине или в рамках одного процесса); 2) настройки соответствующих параметров gRPC в Go (таких, как размер сообщения при большом объеме данных или потоковом методе передачи данных); 3) оптимизации работы MySQL-прокси (переупаковка данных между протоколами, обработка очереди запросов клиентов). Также возможна передача данных напрямую через вызов функций (из Go в C++)) с использованием SWIG [4] вместо реализованной в текущей версии платформы сетевой/сокет–абстракции (gRPC over TCP/Unix Domain Sockets).

Заключение

В работе представлены результаты тестирования производительности прототипа платформы SecureDBMS без модулей шифрования. Исследуется влияние на время выполнения SQL-запросов базовых операций анализа и преобразования передаваемых данных. Выявлено существенное замедление отклика целевой СУБД MySQL, хотя при этом SQL-запросы выполняются с приемлемой для практики задержкой. Изложены текущие особенности и предложения по улучшению производительности платформы. В дальнейшем планируется реализация и тестирование платформы с модулями шифрования.

ЛИТЕРАТУРА

1. *Мартишин С.А., Храпченко М.В., Шокуров А.В.* Исследование задачи обеспечения безопасности при хранении и обработке конфиденциальных данных // Труды ИСП РАН. – Т. 33 (Вып. 2). – 2021. – С. 173–190.

2. *Яковлев Г.А., Тренькаев В.Н.* Платформа SECUREDBMS для разработки защищенной облачной СУБД // Материалы X Международной молодежной научной конференции "Математическое и программное обеспечение информационных, технических и экономических систем". Томск, 2023. – Т. 308. – С. 74–81.

3. Сайт проекта "Cap'n Proto" – URL: https://capnproto.org/ (дата обращения: 30.06.2024)

4. Репозиторий Github проекта SWIG – URL: https://github.com/swig/swig (дата обращения: 14.05.2024)

III. МАТЕМАТИЧЕСКАЯ ТЕОРИЯ ТЕЛЕТРАФИКА И ТЕОРИЯ МАССОВОГО ОБСЛУЖИВАНИЯ

ПРОТОТИП ПРИЛОЖЕНИЯ ДЛЯ ПОИСКА РАСПРЕДЕЛЕНИЯ, НАИБОЛЕЕ БЛИЗКОГО К ЗАДАННОМУ ЭМПИРИЧЕСКОМУ

Прокудина Ю.А., Моисеев А.Н.
Томский государственный университет
prokudina_ua@mail.ru, moiseev.tsu@gmail.com

Введение

В настоящее время в ИПМКН ТГУ ведется разработка программного комплекса SimQ [1]. Одной из задач этого программного комплекса является нахождение распределения числа заявок в системах массового обслуживания (СМО) [2] в аналитическом виде. Для того, чтобы это выполнить, нужно по результатам имитационного моделирования найти наиболее подходящее распределение вероятностей и оценить его параметры.

Предполагается, что имеется некоторый банк распределений, в котором производится поиск. Главной целью данной работы является построение прототипа приложения для оценки параметров отобранных распределений и нахождения наиболее подходящего распределения.

1. Поведение и структура прототипа

Предлагается следующее поведение прототипа (рис. 1). На вход приложения подается эмпирическое распределение вероятностей дискретной случайной величины, полученное в результате имитационного моделирования.

Рис. 1. Поведение прототипа

Для оценки параметров распределений используются метод моментов и метод, основанный на нескольких первых значениях эмпирического распределения. С помощью данных методов выполняется оценка параметров распределений. Полученные оценки подставляются в качестве параметров для каждого распределения из банка распределе-

ний и производится сравнение полученных аналитических распределений с заданным эмпирическим. Для оценки погрешности используется расстояние Колмогорова. В качестве наиболее подходящего распределения из банка выбирается то, для которого расстояние Колмогорова получилось наименьшим. При оценке параметров распределений могут получиться значения, которые не будут удовлетворять ограничениям, накладываемым на параметры этих распределений. Поэтому, если нельзя оценить параметры распределения, оно не берется для выполнения дальнейших вычислений (вычисления расстояния Колмогорова и выбора наилучшего распределения).

Предлагается следующая структура прототипа (рис. 2). Структура использует паттерн проектирования – Шаблонный метод [3]. Абстрактный класс AbstractDistribution является шаблонным классом для своих классов потомков (ConcreteDistribution).

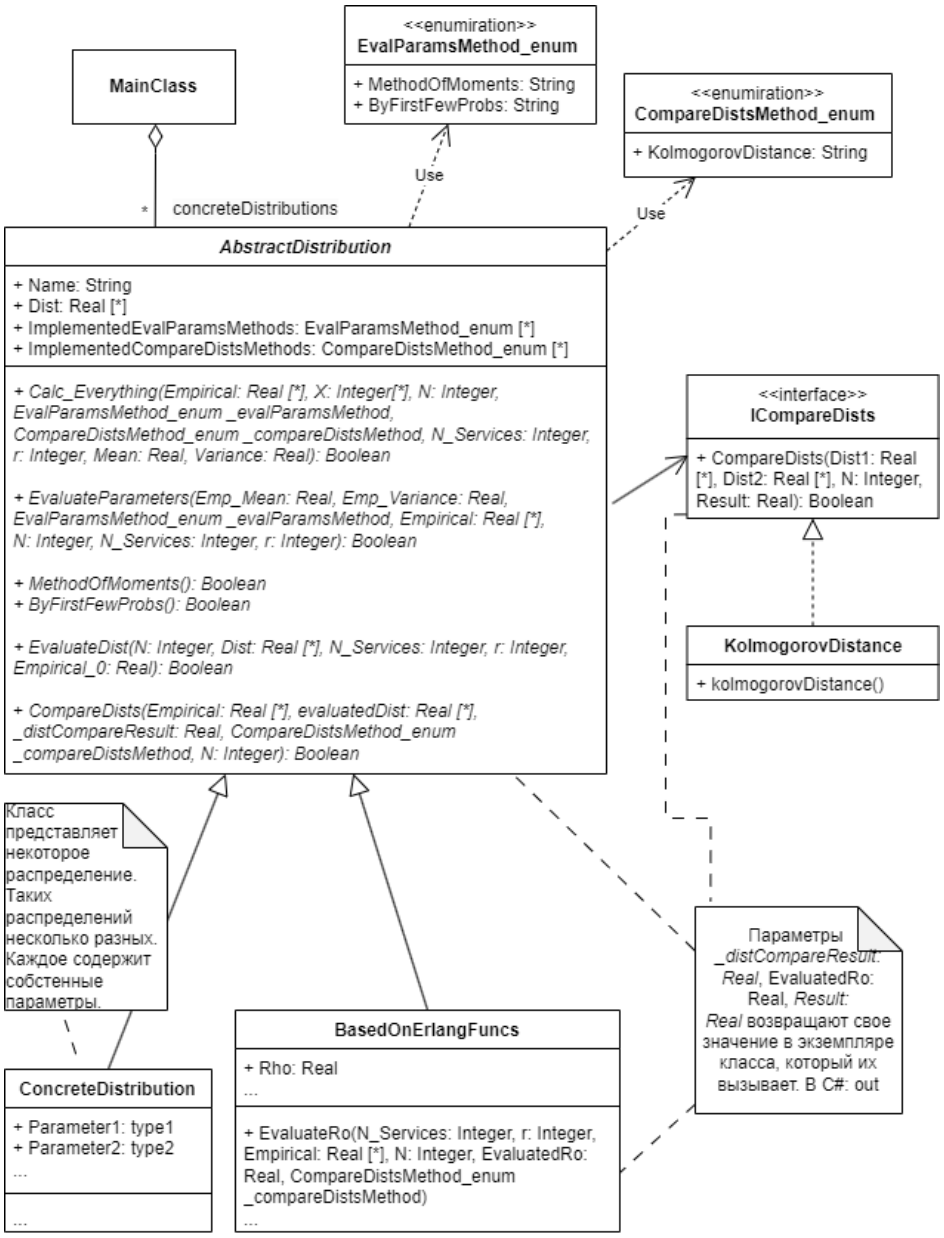


Рис. 2. Структура прототипа

Класс AbstractDistribution – абстрактный класс, представляющий оцениваемое распределение. Класс ConcreteDistribution – класс, представляющий конкретное распределение, наследуемое от класса AbstractDistribution. Атрибут ImplementedEvalParamsMethods содержит массив реализованных потомком методов оценки параметров. Атрибут ImplementedEvalDistsMethods содержит массив реализованных потомком методов сравнения распределений. Атрибут Name – название распределения, атрибут Dist содержит вероятности для распределения с вычисленными оценками параметров (далее – оценочный ряд распределения). Метод Calc_Everything вычисляет все характеристики конкретного распределения (оценка его параметров, построение оценочного ряда распределения, сравнение эмпирического и оценочного рядов с помощью расстояния Колмогорова). Метод EvaluateParameters оценивает параметры распределения, выполняя заданный метод оценки. Метод MethodOfMoments реализует метод моментов для оценки параметров распределения. Метод ByFirstFewProbs выполняет метод оценки параметров на основе нескольких первых значений выборки. Метод EvaluateDist вычисляет и сохраняет оценочный ряд распределения. Метод CompareDists сравнивает его с эмпирическим распределением.

Класс MainClass – управляющий объект приложения, содержит коллекцию конкретных распределений, которые используются в программе. Класс KolmogorovDistance отвечает за реализацию расстояния Колмогорова. Перечисления EvalParamsMethod_enum и CompareDistsMethod_enum содержат названия методов для оценки параметров и сравнения распределений соответственно.

Также структура использует паттерн Стратегия [3]: ICompareDists предоставляет интерфейс для методов сравнения распределений. Операция CompareDists выполняет сравнение распределений. В частности, в реализованном прототипе класс KolmogorovDistance для вычисления расстояния Колмогорова реализует этот интерфейс.

В качестве примера, при оценке параметров для распределения, основанного на формулах Эрланга, параметр Rho может принимать несколько значений и, чтобы выбрать наилучшее значение, используется расстояние Колмогорова.

2. Распределения числа заявок в СМО с неограниченным числом приборов и оценка их параметров

В [4] было установлено, что для бесконечнолинейных СМО [5] встречаются следующие стационарные распределения вероятностей числа заявок в системе: пуассоновское, обобщенное отрицательное биномиальное, гауссовская аппроксимация (дискретизированное и нормированное на положительную полуось нормальное распределение). Все эти распределения помещены в банк распределений.

Для того, чтобы выполнить оценку параметров этих распределений, использован метод моментов [6,7]. Для исходной выборки вычислены значения математического ожидания и дисперсии, эти значения подставлены в выражения для аналитического вычисления математического ожидания и дисперсии каждого из распределений. В результате решения систем уравнений для каждого распределения получены значения его параметров: M_{emp} – эмпирическое математическое ожидание, D_{emp} – эмпирическая дисперсия.

Распределение Пуассона. Закон распределения вероятностей: $P(\lambda) = \frac{e^{-\lambda}\lambda^{k}}{k!}, k = 0,1,2,\ldots$. Здесь $\lambda \in (0;\infty)$ – параметр распределения. Для оценки параметра этого распределения следует использовать формулу $\lambda = M_{\text{emp}}$.

Гауссовская аппроксимация. Закон распределения вероятностей: $P(i)=\frac{G(i+0.5)-G(i-0.5)}{1-G(-0.5)}$, $i=0,1,2,...$ Здесь $G(x)$ – функция распределения нормальной случайной величины с математическим ожиданием μ и дисперсией σ^2. Ограничение, накладываемое на параметры данного распределения: $\sigma^2>0$. Формулы для оценки параметров распределения получаются прямым приравниванием моментов первого и второго порядков: $\mu=M_{\text{emp}}$, $\sigma^2=D_{\text{emp}}$.

Обобщенное отрицательное биномиальное распределение (ООБР). Закон распределения вероятностей: $NB(r,p)=\frac{\Gamma(r+k)}{k!\Gamma(r)}p^r q^k, k=0,1,2,....$ Здесь Γ(z) – гамма-функция, *r*, *p*, *q* – параметры распределения. Ограничения, накладываемые на параметры: $r>0$, $p\in(0,1)$, $q=1-p$. Формулы оценки параметров распределения на основе эмпирических моментов: $p=\frac{M_{\text{emp}}}{D_{\text{emp}}}$, $r=\left|\frac{M_{\text{emp}}^2}{D_{\text{emp}}-M_{\text{emp}}}\right|$, $q=1-p$.

3. Распределения для аппроксимации числа заявок в СМО с ожиданием и оценка их параметров

Для простых СМО с ожиданием были выделены следующие распределения: геометрическое; квази-геометрическое; распределение, основанное на формулах Эрланга, формула Поллачека – Хинчина. Эти распределения также помещены в банк распределений.

Для того, чтобы выполнить оценку параметров квази-геометрического распределения, основанного на формулах Эрланга, в формулу распределения вероятностей подставлены значения вероятностей для первого элемента выборки, составлены уравнения и в результате вычислений для каждого распределения получены значения его параметров. Для геометрического распределения использован метод моментов.

Геометрическое распределение. Закон распределения вероятностей: $\text{Geo}(p)=q^k p$, $k=0,1,2,....$ Здесь p и q – параметры распределения. Ограничения, накладываемые на параметры: $0<p<1$, $q=1-p$. Формулы для оценки параметров этого распределения имеют вид: $p=\frac{M_{\text{emp}}}{D_{\text{emp}}}$, $q=1-p$.

Квази-геометрическое распределение [8,9]. Закон распределения вероятностей:

$$\begin{cases} p_i=\rho(z_0-1)(z_0)^{-i}, & i\geq 1, \\ p_0=1-\rho, & i=0. \end{cases}$$

Здесь ρ и z_0 – параметры распределения. Ограничения, накладываемые на параметры: $\rho>0$, $z_0>1$. Формулы для оценки параметров квази-геометрического распределения: $\rho=1-p_{\text{emp0}}$, $z_0=\frac{\rho}{\rho-p_{\text{emp1}}}$. Здесь p_{emp0} и p_{emp1} – значения нулевого и первого элементов эмпирического распределения.

Распределение, основанное на формулах Эрланга. Закон данного распределения вероятностей имеет вид

$$p_i=\frac{\rho^i}{i!}p_0,\ i=\overline{0,n},\ p_i=\frac{\rho^i p_0}{n^{i-n}n!},\ i=\overline{n+1,n+r},\ p_0=\left(\sum_{i=0}^{n}\frac{\rho^i}{i!}+\sum_{i=1}^{r}\frac{\rho^{n+i}}{n!n^i}\right)^{-1}. \quad (1)$$

Здесь ρ, $n=1,2,,...$, $r=0,1,...$ – параметры распределения. В текущем прототипе мы предполагаем, что параметры *n* (число обслуживающих приборов) и *r* (число мест для

ожидания) заданы, поэтому остается только оценить параметр ρ. Ограничения, накладываемые на этот параметр:

$$\begin{cases} \rho > 0, \ r < \infty, \\ 0 < \rho < 1, \ r = \infty. \end{cases} \quad (2)$$

На основе системы (1) получено следующее уравнение относительно неизвестного параметра ρ:

$$\left(1 - \frac{1}{p_{\text{emp0}}}\right) + \sum_{i=1}^{n} \frac{1}{i!}\rho^i + \sum_{i=1}^{r} \frac{1}{n!n^i}\rho^{n+i} = 0 . \quad (3)$$

Здесь p_{emp0} – значение нулевого элемента эмпирического распределения.

В результате решения уравнения (3) получается $n + r$ корней. Неподходящие под условия (2) корни (комплексно-сопряженные, отрицательные, …) отбрасываются. В результате может не остаться подходящих корней, тогда оценить параметр нельзя и распределение исключается из списка кандидатов. Если остались подходящие корни, тогда выполняется построение оценок ряда распределения для каждого корня, а затем выбирается тот, для которого расстояние Колмогорова получилось наименьшим.

4. Реализация метода сравнения (близости) распределений

В качестве метода оценки близости распределений в текущей версии прототипа приложения используется расстояние Колмогорова [6]. Для каждого распределения, для которого получилось оценить параметры, вычисляется расстояние Колмогорова по формуле $D_n = \sup_{x \in \mathbb{R}} |F_n(x) - F(x)|$. Здесь $F_n(x)$ – эмпирическое распределение, $F(x)$ – оценка ряда распределения.

5. Пример работы прототипа

На основании вышеизложенного был реализован прототип приложения для поиска распределения, наиболее близкого к заданному эмпирическому. На вход программы подается эмпирический ряд распределения числа заявок в СМО конкретного типа, полученный в результате имитационного моделирования соответствующей СМО.

На рис. 3 показана аппроксимация распределения числа заявок в СМО типа $G/G/\infty$ для распределений: дискретного аналога нормального, пуассоновского, обобщенного отрицательного биномиального и геометрического. Вычислены параметры этих распределений и расстояния Колмогорова по отношению к заданному эмпирическому ряду. В результате сравнения сделан вывод, что наилучшая аппроксимация в данном случае – это обобщенное отрицательное биномиальное распределение (Negative binomial).

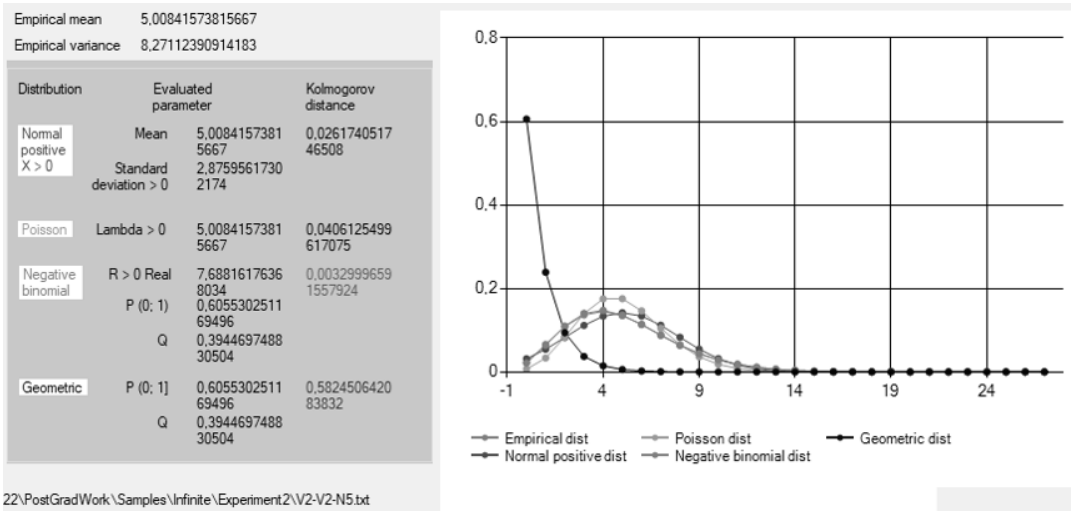


Рис. 3. Аппроксимация распределения числа заявок в СМО типа $G/G/\infty$.
Наилучшая аппроксимация – обобщенное отрицательное биномиальное распределение (Negative binomial)

На рис. 4 показана аппроксимация распределения числа заявок в СМО типа $M/M/N/\infty$ для следующих распределений: геометрического, квази-геометрического, распределения, основанного на формулах Эрланга. Вычислены параметры этих распределений и соответствующие расстояния Колмогорова. В результате сравнения значений расстояний Колмогорова сделан вывод, что в данном случае наилучшая аппроксимация – распределение, основанное на формулах Эрланга (Based on Erlang Functions).

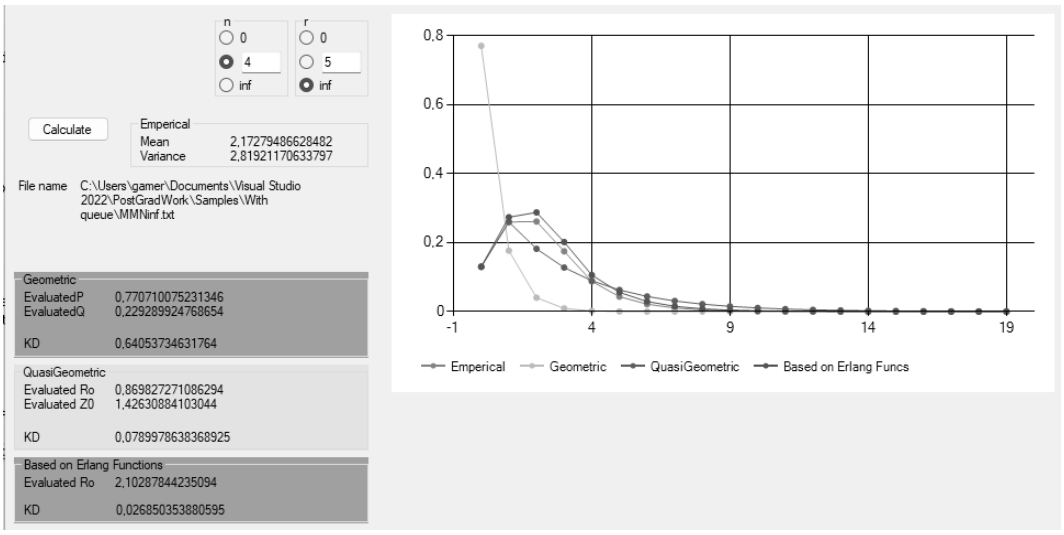


Рис. 4. Аппроксимация распределения числа заявок в СМО типа $M/M/N/\infty$.
Наилучшая аппроксимация – основанная на формулах Эрланга (Based on Erlang Functions)

Заключение

В качестве результата данной работы предложены формулы для оценки значений параметров следующих распределений вероятностей для аппроксимации заданного эмпирического ряда: пуассоновского, обобщенного отрицательного биномиального, гауссовской аппроксимации, геометрического, квази-геометрического, распределения, основанного на формулах Эрланга, формулах Поллачека – Хинчина. Разработана процедура поиска распределения, наиболее близкого к заданному.

Архитектура прототипа приложения спроектирована таким образом, чтобы в будущем можно было бы достаточно легко пополнять банк использующихся распределений, а также метрик для оценки близости между распределениями вероятностей. В итоге прототип приложения был реализован и протестирован.

ЛИТЕРАТУРА

1. *Прокудина Ю.А., Оруджов Э.А., Моисеев А.Н.* Концепция архитектуры программного комплекса SimQ // Материалы X-й Международной молодежной научной конференции "Математическое и программное обеспечение информационных, технических и экономических систем". Томск, 26–29 мая 2023 г. – Труды Томского государственного университета. – Т. 308: Серия физико-математическая. – Томск, 2023. – С. 155–159.

2. *Назаров А.А., Терпугов А.Ф.* Теория массового обслуживания: учебное пособие, – 2-е изд., испр. – Томск: Изд-во НТЛ, 2010. – 228 с.

3. *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.* Паттерны объектно-ориентированного проектирования. – СПб.: Питер, 2020 — 448 с.: ил.

4. *Прокудина Ю.А., Моисеев А.Н.* Распределение числа заявок в системах массового обслуживания с неограниченным числом приборов // Информационные технологии и математическое моделирование (ИТММ-2023): материалы XXII Международной конференции им. А.Ф. Терпугова, 4–9 декабря 2023 г. – Томск: Изд-во ТГУ, 2023. – С. 145–150.

5. *Моисеев А.Н., Назаров А.А.* Бесконечнолинейные системы и сети массового обслуживания. – Томск: Изд-во НТЛ, 2015. – 240 с.

6. *Лебедев А.В., Фадеева Л.Н.* Теория вероятностей и математическая статистика: учебное пособие. – 2-е изд., перераб. и доп. – М.: Эксмо, 2010 – 496 с.

7. *Тюрин Ю.Н., Макаров А.А.* Анализ данных на компьютере: учебное пособие. 4-е изд., перераб. – М.: ИД Форум, 2008 – 368 с., ил.

8. *Fedorova E.A.* Quasi-geometric and Gamma Approximation for Retrial Queueing Systems // Communications in Computer and Information Science. – 2014. – V. 487. – P. 123–136.

9. *Назаров А.А., Любина Т.В.* Немарковская динамическая RQ-система с входящим MMP-потоком заявок // Автоматика и телемеханика. – 2013. № 7. – С. 89–101.

СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ С НЕОГРАНИЧЕННЫМ ЧИСЛОМ УСТРОЙСТВ В СЛУЧАЙНЫХ СРЕДАХ

Тарасенко А.В., Моисеев А.Н.

Томский государственный университет

lsa328@yandex.ru

Введение

В данной работе рассматривается применение математических моделей, параметры которых меняются со временем случайным образом в зависимости от состояния внешнего случайного процесса. Такие модели находят широкое применение в различных приложениях. В частности, бесконечнолинейные системы массового обслуживания используются для аппроксимации систем с большим числом обслуживающих приборов [1]. Примерами таких систем могут быть сети банков, супермаркетов, call-центров и сервисов цифровой дистрибуции [2]. Важно отметить, что такие системы часто подвергаются внешним возмущениям, которые влияют на интенсивность входящего потока заявок и длительность времени обслуживания заявок. В зависимости от этих условий поток клиентов может быть более или менее интенсивным, а срок обработки заявки будет длиннее или короче. Таким образом, математические модели с учетом случайных изменений параметров позволяют анализировать и оптимизировать процессы в подобных системах.

1. Постановка задачи

Рассмотрим систему массового обслуживания (СМО) M/M/∞ (рис. 1), в которой входящий простейший поток и экспоненциальное время обслуживания зависят от "случайных сред", под которыми мы понимаем цепи Маркова $k(t)$ и $n(t)$ соответственно с непрерывным временем, принимающие значения $k(t)=\overline{1,K}$ и $n(t)=\overline{1,N}$. Параметр

входящего простейшего потока $\lambda = \lambda_k$ зависит от состояния среды $k(t)$, а параметр экспоненциального времени обслуживания $\mu = \mu_n$ – от состояния среды $n(t)$. Система ведёт себя как $M(\lambda_k)/M(\mu_n)/\infty$. Среды $n(t)$ и $k(t)$ определяются матрицами инфинитезимальных характеристик Q_1 и Q_2 соответственно. Случайный процесс $S\{i(t),k(t)n(t)\}$ является цепью Маркова. Обозначим через $P(i,k,n) = P\{i(t) = i, k(t) = k, n(t) = n\}$, $i \geq 0$, $k = \overline{1,K}$, $n = \overline{1,N}$ распределение вероятностей состояний рассматриваемого случайного процесса. В системе переход среды в новое состояние может повлиять на закон обслуживания заявок, находящихся в системе, или на параметры входящего потока и интенсивность обслуживания. Тогда для исследования такой системы предполагаем пять вариантов функционирования при переходе в новое состояние случайной среды:

1. Изменяется интенсивность обслуживания заявок, все заявки в системе начинают обслуживаться по новому закону.
2. Изменяется интенсивность обслуживания заявок, все текущие заявки в системе продолжают обслуживаться по старому закону.
3. Изменяется интенсивность входящего потока, все заявки в системе продолжают обслуживаться по старому закону.
4. Изменяется интенсивность входящего потока и интенсивность обслуживания заявок, все заявки в системе начинают обслуживаться по новому закону.
5. Изменяется интенсивность входящего потока и интенсивность обслуживания заявок, все заявки в системе продолжают обслуживаться по старому закону.

В данной работе будет рассмотрено решение задачи для пятого варианта функционирования (рис. 2). Вместо двух случайных сред $k(t)$ и $n(t)$ рассмотрим одну – $s(t)$, принимающую значения $s(t) = \overline{1,S}$. Цепь Маркова $s(t)$ определяется матрицей инфинитезимальных характеристик Q.

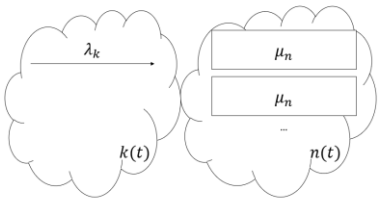


Рис. 1. СМО в случайных средах

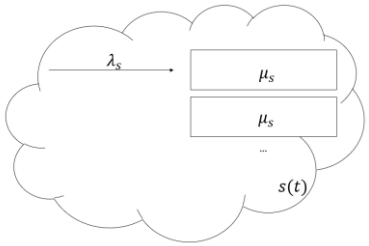


Рис. 2. Частный случай СМО с одной случайной средой

2. Система дифференциальных уравнений Колмогорова

Сначала составим систему дифференциальных уравнений Колмогорова для данной модели:

$$\begin{cases} \dfrac{\partial P(0,s,t)}{\partial t} = -P(0,s,t)\lambda_s + P(1,s,t)\mu_s + \sum\limits_{v=0}^{\infty} q_{vs} P(0,v,t), \\ \dfrac{\partial P(1,s,t)}{\partial t} = -P(1,s,t)\left(\lambda_s + \mu_s\right) + \lambda_s P(0,s,t) + 2P(2,s,t)\mu_s + \sum\limits_{v=0}^{\infty} q_{vs} P(1,v,t), \\ \ldots \\ \dfrac{\partial P(i,s,t)}{\partial t} = -P(i,s,t)\left(\lambda_s + i\mu_s\right) + \lambda_s P(i-1,s,t) + (i+1)P(i+1,s,t)\mu_s + \sum\limits_{v=0}^{\infty} q_{vs} P(1,v,t), \\ \ldots \end{cases} \quad (1)$$

Будем полагать, что в рассматриваемой СМО существует стационарный режим. Обозначим через $H(u,S) = \sum\limits_{i=0}^{\infty} e^{jui} P(i,s)$ частичную характеристическую функцию стационарного распределения вероятностей числа заявок в системе. Также введем следующие обозначения: $\mathbf{H}(u)$ – вектор, состоящий из элементов $H(u,1),H(u,2),\ldots,H(u,S)$, $\mathbf{\Lambda}$ – диагональная матрица с элементами $\lambda_1,\ldots,\lambda_S$ на главной диагонали, $\mathbf{M}$ – диагональная матрица с элементами $\mu_1,\ldots,\mu_S$ на главной диагонали. Тогда систему (1) можно записать в векторно-матричном виде

$$j\left(e^{-ju} - 1\right)\frac{d\mathbf{H}(u)}{du}\mathbf{M} = \mathbf{H}(u)\left[\left(e^{ju} - 1\right)\mathbf{\Lambda} + \mathbf{Q}\right]. \quad (2)$$

Начальное условие для решения векторно-матричного уравнения возьмём в виде

$$\mathbf{H}(0) = \mathbf{r}, \quad (3)$$

где $\mathbf{r}$ – вектор-строка стационарного распределения вероятностей состояний случайной марковской среды $s(t)$.

3. Решение задачи методом асимптотического анализа

Для решения задачи (2)–(3) воспользуемся методом асимптотического анализа. Решение будем искать в условии высокой интенсивности входящего потока и частой смены состояний среды [3]. Для этого введем следующие замены: $\mathbf{\Lambda} = \mathbf{\Lambda}N$, $\mathbf{Q} = \mathbf{Q}N$, где $N \to \infty$ – параметр, характеризующий высокую интенсивность входящего потока и частую смену состояний случайной среды.

Применяя метод асимптотического анализа, введем следующие обозначения: $\varepsilon = 1/N$, $u = \varepsilon w$, $\mathbf{H}(u) = \mathbf{F}_1(w,\varepsilon)$, подставим в (2), получим

$$j\left(e^{-j\varepsilon w} - 1\right)N\frac{d\mathbf{F}_1(w,\varepsilon)}{dw}\mathbf{M} = \mathbf{F}_1(w,\varepsilon)\left[\left(e^{j\varepsilon w} - 1\right)\mathbf{\Lambda} + \mathbf{Q}\right]. \quad (4)$$

Теорема 1. Асимптотическое решение $\mathbf{F}_1(w) = \lim\limits_{\varepsilon \to 0}\mathbf{F}_1(w,\varepsilon)$ уравнения (4) имеет вид $\mathbf{F}_1(w) = \mathbf{r}\exp\{jwk_1\}$, $k_1 = \dfrac{\mathbf{r\Lambda e}}{\mathbf{rMe}}$.

Отсюда можно записать выражение для характеристической функции числа заявок в системе в первом приближении в виде $\mathbf{H}(u) \approx \mathbf{F}_1(w) = \mathbf{r}\exp\{juk_1N\}$.

Найдём асимптотику второго порядка. Обозначим через $\mathbf{H}_2(u)$ векторную функцию, удовлетворяющую выражению $\mathbf{H}(u) = \mathbf{H}_2(u)\exp\{juk_1N\}$. Подставляя это выражение в (2), получим

$$\frac{1}{N}j\left(e^{-ju} - 1\right)\frac{d\mathbf{H}_2(u)}{du}\mathbf{M} = \mathbf{H}_2(u)\left[\left(e^{-ju} - 1\right)k_1\mathbf{M} + \left(e^{ju} - 1\right)\mathbf{\Lambda} + \mathbf{Q}\right].$$

Выполним здесь следующие замены: $1/N = \varepsilon^2$, $u = \varepsilon w$, $\mathbf{H}_2(u) = \mathbf{F}_2(w,\varepsilon)$. Получим

$$\varepsilon^2 j\left(e^{-j\varepsilon w}-1\right)\frac{\partial \mathbf{F}_2(w,\varepsilon)}{\partial w}\mathbf{M}=\mathbf{F}_2(w,\varepsilon)\left[\left(e^{-j\varepsilon w}-1\right)k_1\mathbf{M}+\left(e^{j\varepsilon w}-1\right)\mathbf{\Lambda}+\mathbf{Q}\right]. \quad (5)$$

Теорема 2. Асимптотическое решение уравнения (5) имеет вид

$$\mathbf{F}_2(w)=\mathbf{r}\exp\left\{\frac{(jw)^2}{2}Nk_2\right\},\ k_2=\frac{\mathbf{r\Lambda e}-k_1\mathbf{gMe}+\mathbf{g\Lambda e}}{\mathbf{rMe}}.$$

Здесь вектор-строка g является решением системы $\begin{cases}\mathbf{gQ}=k_1\mathbf{rM}-\mathbf{r\Lambda},\\ \mathbf{ge}=1.\end{cases}$

В результате получаем выражение для характеристической функции числа заявок в системе во втором приближении в виде: $\mathbf{H}(u)\approx\mathbf{F}_2(w)=\exp\left\{juk_1N+\frac{(ju)^2}{2}k_2N\right\}$.

Таким образом, в условии высокой интенсивности входящего потока и частого изменения состояний случайной среды стационарное распределение вероятностей числа заявок в системе можно аппроксимировать гауссовским распределением с математическим ожиданием $a=M\{i(t)\}=k_1N$ и дисперсией $\sigma^2=M\left\{(i(t)-a)^2\right\}=k_2N$.

4. Дискретизация распределения

Полученный результат в работе является функцией распределения нормальной величины. Для получения из нормального распределения закона распределения дискретной случайной величины числа заявок в системе следует выполнить его преобразование на аргумент из множества неотрицательных целых чисел. Для этого воспользуемся [1]

$$F_i=\frac{1}{1-G(-1/2)}\left[G(i+1/2)-G(i-1/2)\right], \quad (6)$$

где $i=\overline{0,\infty}$ и $G(x)$ – функция распределения нормальной случайной величины с математическим ожиданием a и дисперсией σ^2.

5. Численный анализ области применения асимптотических результатов

Смоделируем реализацию поведения системы с помощью имитационной модели. Для анализа точности полученных аппроксимаций законов распределения вероятностей числа занятых приборов смоделируем реализацию поведения модели с помощью дискретно-событийного подхода. Т.к. асимптотический анализ был проведён в условиях высокой интенсивности входящего потока и высокой частоты смены состояний случайной среды, то нужно определить погрешность аппроксимации при заданных значениях N параметра, характеризующего интенсивность. Для сравнения законов распределения вероятностей используем количественную метрику расстояние Колмогорова $d=\sup\limits_x\left|\sum\limits_{i=0}^{x}\left[P_i-F_i\right]\right|$, где $x=0,...,\infty$, P_i – относительные частоты эмпирического распределения, построенного на основе имитационного моделирования, а F_i – закон распределения, построенный на основе соответствующей аппроксимации (6). В качестве примера зададим следующие параметры модели:

$$\mathbf{\Lambda}=\begin{bmatrix}4&0&0\\0&3&0\\0&0&5\end{bmatrix},\ \mathbf{M}=\begin{bmatrix}4&0&0\\0&8&0\\0&0&2\end{bmatrix},\ \mathbf{Q}=\begin{bmatrix}-5&2&3\\0&-3&3\\2&1&-3\end{bmatrix}. \quad (7)$$

Согласно аппроксимации, при параметрах (7) математическое ожидание будет равно 1. На рис. 3 показаны графики сравнения эмпирического и асимптотического распределений числа заявок для различных значений N. В табл. 1 представлены значения расстояния Колмогорова между эмпирическим и асимптотическим распределением. По таблице можно определить, что начиная с $N = 50$ погрешность распределения составляет менее 0.05, что будем считать приемлемым результатом.

Таблица 1

Расстояние Колмогорова *d* между эмпирическим и асимптотическим распределениями при различных *N*

N	1	5	10	30	50	100
d	0.227	0.113	0.090	0.060	0.043	0.028

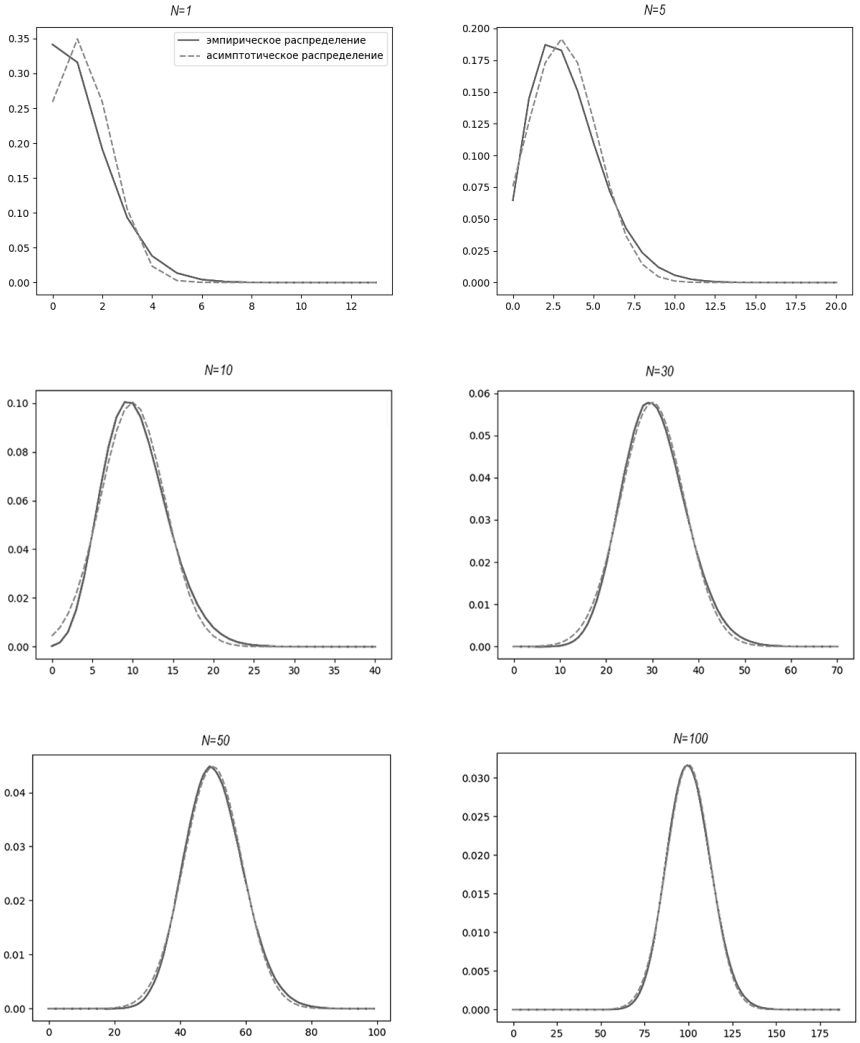


Рис. 3. Сравнение эмпирического и асимптотического распределений числа заявок в системе

Заключение

В работе рассмотрена система массового обслуживания с неограниченным числом приборов, функционирующая в случайной среде. В результате получено приближенное распределение вероятностей числа заявок в системе в асимптотическом условии высокоинтенсивного входящего потока и частой смены состояний среды. Была построена

имитационная модель и определено, при каких значениях N асимптотическое распределение достигает приемлемого результата.

ЛИТЕРАТУРА

1. *Моисеев А.Н., Назаров А.А.* Бесконечнолинейные системы и сети массового обслуживания. – Томск: Изд-во НТЛ, 2015. – 240 с.

2. *Wilkinson R.I.* Theories for toll traffic engineering in the USA. // The Bell System Technical Journal. – 1956. – № 2. – P. 421–507.

3. *Назаров А.А., Моисеева С.П.* Метод асимптотического анализа в теории массового обслуживания. – Томск: Изд-во НТЛ, 2006. – 112 с.

IV. МАТЕМАТИЧЕСКИЕ МЕТОДЫ И МОДЕЛИ ЦИФРОВОЙ ЭКОНОМИКИ

СРАВНЕНИЕ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ УЧАЩИХСЯ С ИСПОЛЬЗОВАНИЕМ СТАТИСТИЧЕСКОГО АНАЛИЗА И КЛАСТЕРНОГО ПОДХОДА

Барыкина А.О., Тарасенко В.Ф., Ерёмина Н.Л.
Томский государственный университет
stok1712@gmail.com, vtara54@mail.ru, 26051971@mail.ru

Введение

В современной образовательной парадигме для оценки уровня освоения учебного материала широко применяются методы тестирования. Данная методика представляет собой процедуру исследования, направленную на выявление и анализ степени выраженности определённых навыков, умений и знаний у участников тестирования посредством выполнения ими комплекса специализированных заданий. Такие задания, объединённые общей концепцией, принято называть тестами. По результатам тестирования можно получить информацию об уровне развития того или иного качества у испытуемого и сопоставить его с установленными стандартами. Тестовые методики доктора Реддина, используемые в процессе подготовки и повышения квалификации менеджеров, получили широкое распространение во многих странах мира. Каждый тест данного типа содержит 80 утверждений, и задачей испытуемого является оценка истинности или ложности каждого из них.

В XXI-м веке, характеризующемся быстрым развитием технологий, усилением конкуренции и непредсказуемыми изменениями в окружающей среде, успешность компаний зависит от наличия в их штате мотивированных, высококвалифицированных сотрудников, способных оперативно реагировать на внутренние и внешние изменения. Для эффективного управления персоналом менеджеры должны обладать знаниями о поведенческих особенностях индивидов и групп, проблемах коммуникации и мотивации. Это позволяет им организовать продуктивное взаимодействие внутри коллектива, оптимизировать организационную структуру в соответствии с требованиями внешней среды и, в конечном итоге, достигать поставленных перед организацией целей.

Одним из методов выявления проблем в управлении является опрос мнений менеджеров относительно значимости различных факторов, влияющих на эффективность управления. В данной работе рассматривается количественный подход к управлению, основанный на разработке и применении адекватных математических моделей проблемных ситуаций, что способствует повышению эффективности. Данный подход к управлению в сочетании с современными информационными технологиями, анализом больших данных и другими аналогичными инструментами в настоящее время обозначается термином "цифровая экономика".

1. Описание предметной области

Предметом исследования являются результаты тестов доктора Реддина под названием "Коммуникации в организации", "Информированность сотрудников", "Человеческие отношения". Предметная область настоящего исследования заключается в сравнительном анализе уровня осведомленности в области управления среди различных групп респондентов, потенциальных менеджеров, в течение пятнадцатилетнего периода исследования.

Тесты, разработанные доктором Реддином, предназначены для выявления у респондентов компетенций по ключевым аспектам управления и межличностных отно-

шений в организации, что может способствовать развитию персонала, улучшению коммуникации и повышению эффективности работы команды. Трехмерная 3D-модель лидерства, предложенная британским профессором, Уильямом Джеймсом Реддином, была изучена в контексте различных теорий стилей лидерства, основанных на ситуационных факторах, влияющих на эффективность организации. Эта модель включает в себя три измерения лидерства – цели, межличностностные отношения, управленческое поведение. Модель лидерства Реддина, представленная в данной работе, представляет собой интегрированный подход к оценке лидерских качеств и эффективности субъектов и включает такие аспекты, как цели, межличностные отношения и управленческое поведение, что позволяет получить более полную картину лидерских способностей и потенциала респондентов [1].

В рамках сравнительного анализа результатов тестов студентов, модель лидерства Реддина предполагает, что эффективность менеджера измеряется в исполнении целей, с учетом условий модели управления и непредвиденно возникающим изменениям. Респонденты, которые успешно прошли тесты, должны были продемонстрировать способность ставить ясные и достижимые цели, а также уметь адаптироваться к изменяющимся условиям и достигать поставленных целей. Человеческие отношения также являются одним из трех статистических измерений лидерства. Студенты, которые успешно прошли тесты, должны были продемонстрировать способность строить и поддерживать эффективные межличностные отношения с другими субъектами. Они должны были проявлять эмпатию, доверие и мотивацию, чтобы создать благоприятную рабочую атмосферу и повысить эффективность команды. Модель лидерства Реддина предполагает, что управленческое поведение зависит не только от поставленной цели, но и тесно связано с межличностными отношениями между субъектами и преподавателями. Респонденты должны были уметь адаптировать свое поведение к различным ситуациям и людям, чтобы достичь наилучших результатов.

Исследование включает результаты опроса студентов г. Томска, ожидающих будущую карьеру в области менеджмента В рамках исследования рассматривается временной промежуток с 2005-го по 2020-й год, с объемом выборки по каждому тесту: "Коммуникации в организации" – 4274 респондента, "Изменения, реформы, преобразования" – 4118 респондентов, "Менеджер и человеческие отношения" – 3728 респондентов. Информация, подлежащая дальнейшей обработке, представлена в виде прямоугольной таблицы данных из m столбцов и n строк, в которой строки – это объекты, а столбцы – характеристики объектов [2,3].

При сравнительном анализе результатов тестирования учащихся можно столкнуться с проблемой большого объема выборки респондентов. Большой объем выборки может привести к сложностям в анализе данных и их разумной интерпретации.

Для решения этих задач предлагается применение алгоритмов кластерного анализа и визуализации полученных результатов. Кластерный анализ позволяет объединять объекты на основе их сходства или различия по определенным параметрам, что облегчает обработку информации. Визуализация полученных результатов дает возможность наглядно представить данные и выявить основные тенденции и закономерности в них. В данной работе произведен сравнительный анализ результатов оригинальных тестов, разработанных Реддиным, применены методы кластерного анализа для разбиения респондентов на "естественные" группы внутри каждого временного интервала, которые формируются по основе "верных" (по мнению автора) ответов на вопросы этих тестов, а также в применены статистические методы обработки данных для поиска особенностей и закономерностей в выявленных группах.

2. Методы исследования

При кластеризации, разделения на группы, необходимо добиться такого результата, при котором, в одной группе будут похожие объекты, в разных группах объекты

будут различны [4]. Среди методов кластерного анализа выделяют два наиболее распространенных в применении: иерархический анализ и метод k-средних.

Для вычисления расстояния между объектами использовалось евклидово расстояние (1), а для измерения расстояния между кластерами был использован метод Варда (2):

$$d\left(X_i, X_j\right) = \sqrt{\sum_{k=1}^{z}\left(x_{ik} - x_{kj}\right)^2}, \quad (1)$$

где z – количество переменных, описывающие объект, x_{ik} –численное значение k-й переменной для i-го объекта, x_{kj} – численное значение k-й переменной для j-го объекта,

$$d(A,B) = \sum_{x\in A, y\in B}\frac{\partial^2(x,y)}{|A|+|B|} - \sum_{x,y\in A}\frac{\partial^2(x,y)}{|A|} - \sum_{x,y\in B}\frac{\partial^2(x,y)}{|B|}, \quad (2)$$

где $d^2(x,y)$ – квадрат евклидового расстояния, $|A|$ – количество элементов в кластере A, $|B|$ – количество элементов в кластере B.

В результате вычисления матрицы евклидовых расстояний построена дендрограмма для каждого теста в каждом периоде исследования, чтобы показать, как формируются кластеры и определить наиболее подходящее количество кластеров. На рис. 1 представлены дендограммы, все временные отрезки имели наибольшие скачки в расстояниях объединения при числе кластеров равного трем.

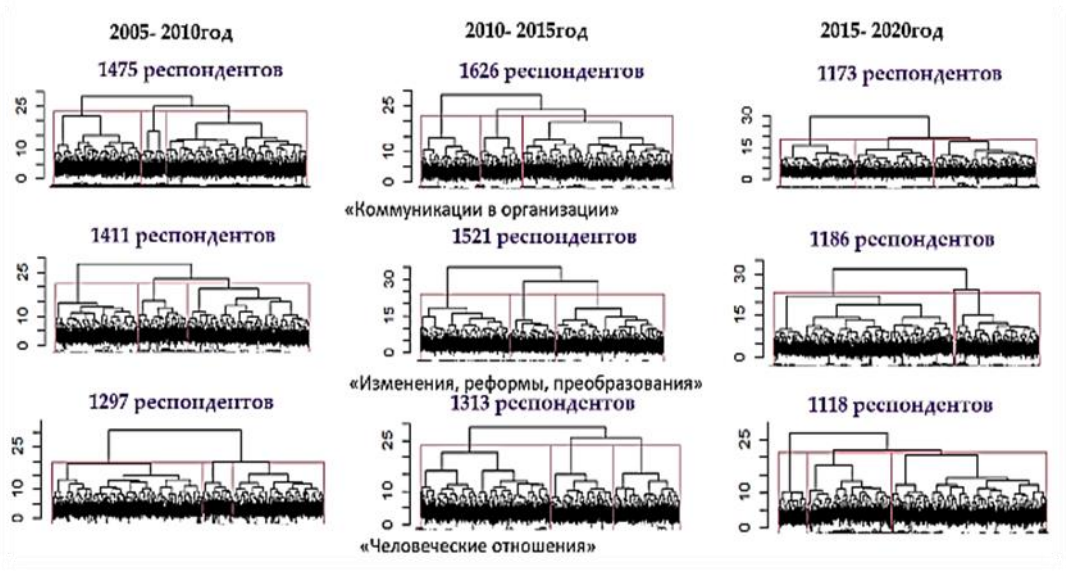


Рис. 1. Дендрограмма объединения кластеров за 2005–2020 гг. по результатам тестов "Коммуникации в организации", "Изменения, реформы, преобразования", "Человеческие отношения"

При визуализации оценки кластеризации с использованием метода многомерного шкалирования объекты (данные, относящиеся к разным кластерам) были представлены в виде точек в двумерном пространстве. Многомерное шкалирование дало возможность увидеть, насколько объекты в каждом кластере походили или отличались друг от друга в пространстве с уменьшенной размерностью. Каждый объект визуализируется с цветовой маркировкой, соответствующей его кластеру. На рис. 2–4 изображены результаты многомерного шкалирования иерархической кластеризации в каждом периоде.

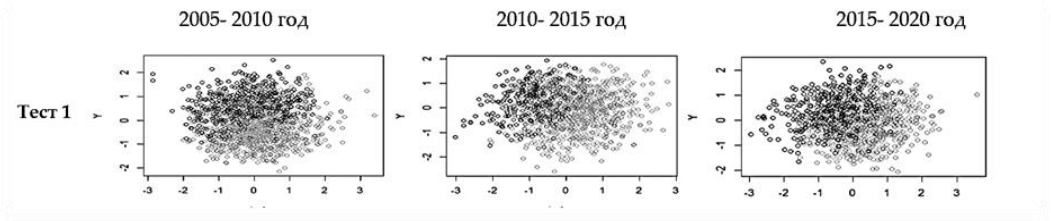


Рис. 2. Многомерное шкалирование результатов иерархической кластеризации для теста № 1

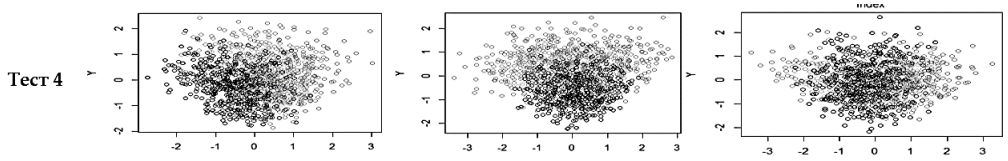


Рис. 3. Многомерное шкалирование результатов иерархической кластеризации для теста № 4

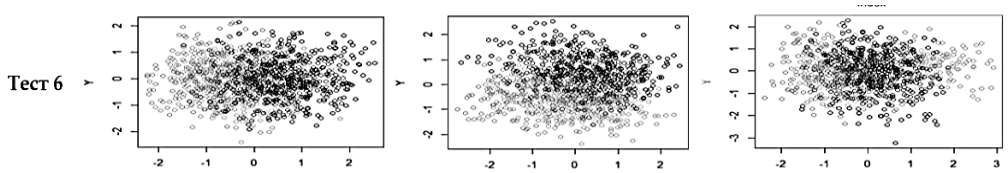


Рис. 4. Многомерное шкалирование результатов иерархической кластеризации для теста № 6

На рис. 5–7 изображены результаты многомерного шкалирования кластеризации методом *k*-средних в каждом периоде.

Рис. 5. Многомерное шкалирование результатов кластеризации методом *k*-средних для теста № 1

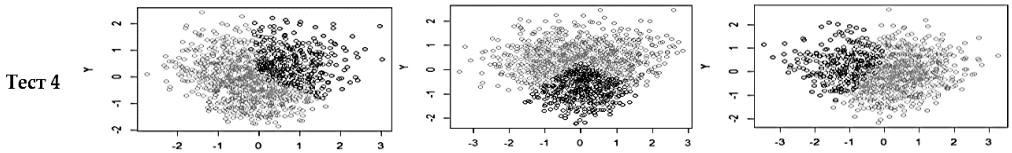


Рис. 6. Многомерное шкалирование результатов кластеризации методом *k*-средних для теста № 4

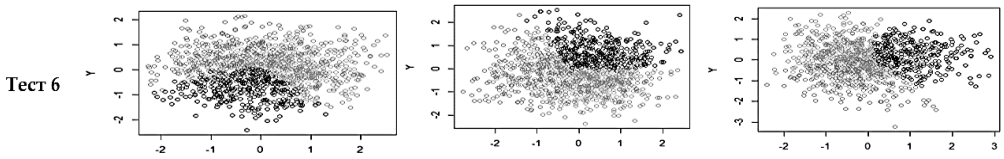


Рис. 7. Многомерное шкалирование результатов кластеризации методом *k*-средних для теста № 6

Использование многомерного шкалирования для визуализации оценки кластеризации позволило уяснить структуру и взаимоотношения между объектами в различных кластерах, что оказалось полезным при сравнении различных методов кластеризации и интерпретации результатов анализа данных. На основе представленных выше графиков можно сделать вывод о том, что оба метода кластеризации приводят к пересечениям

или наложениям кластеров, причем это пересечение явно выражено в иерархическом методе. Это свидетельствует о том, что применение метода *k*-средних на данных наиболее соответствует поставленной задаче кластеризации.

После анализа результатов визуализации для дальнейшего анализа принято решение об использовании кластеров, образовавшихся в результате алгоритма *k*-средних. Это решение принимается на основе такого фактора, как четкая разделимость кластеров.

Для оценки распределения использовался Критерий Шапиро – Уилка (Shapiro – Wilk test) для проверки гипотезы о том, что выборка была получена из генеральной совокупности с нормальным распределением [5]. Результаты проверки представлены на рис. 8–10.

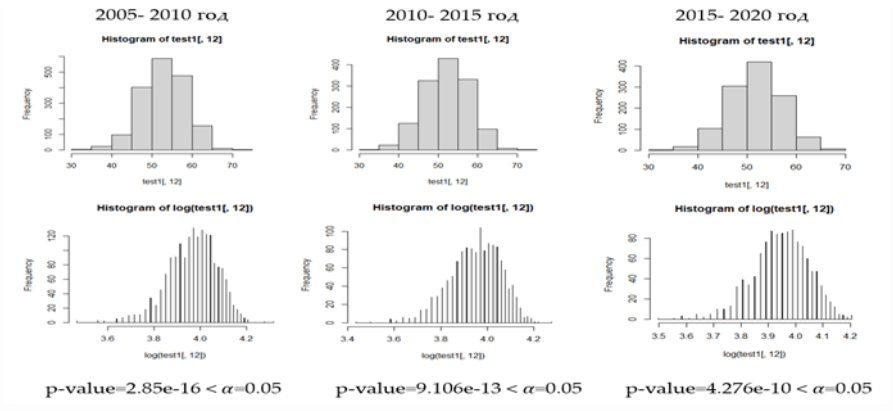


Рис. 8. Проверка гипотезы о нормальности распределения числа совпавших суждений теста № 1

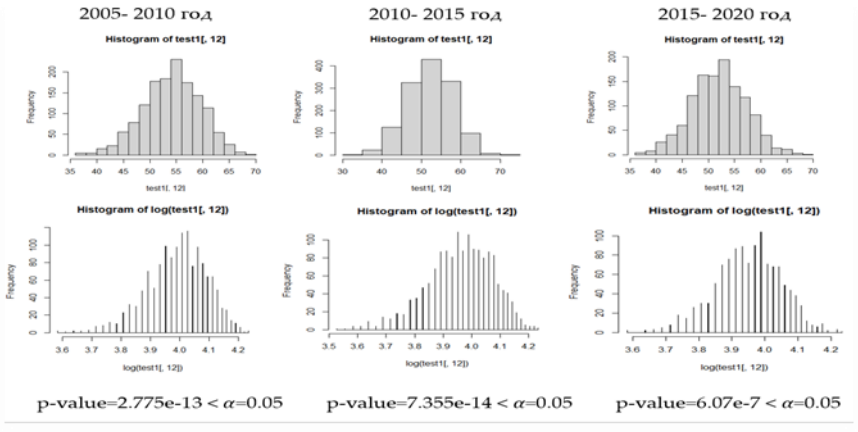


Рис. 9. Проверка гипотезы о нормальности распределения числа совпавших суждений теста № 4

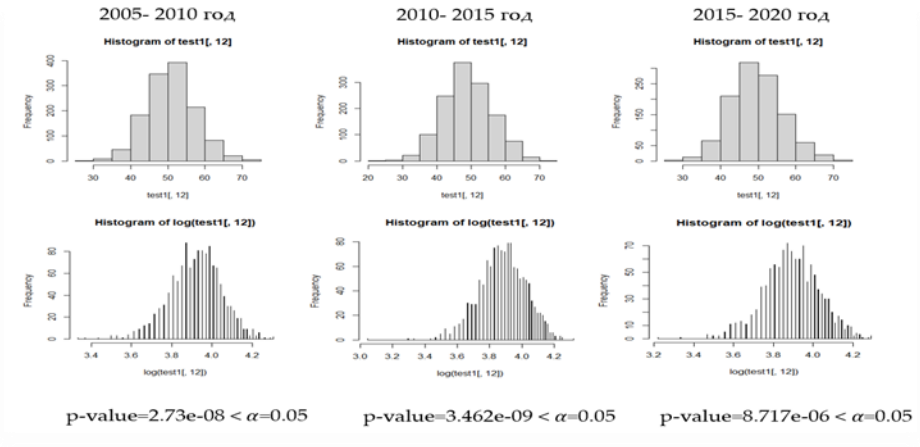


Рис. 10. Проверка гипотезы о нормальности распределения числа совпавших суждений теста № 6

Гипотеза не была принята для всех трех тестов во всех рассматриваемых временных периодах на уровне значимости 0.05 [6].

3. Описательные статистики

После проведения кластеризации посмотрим на описательные статистики по кластерам. Из табл. 1 видно, что средние значения показателей по группам отличаются. На основе доли совпадения ответов в каждом кластере, можно сделать выводы о том, насколько различны точки зрения будущих менеджеров по отношению к конкретным вопросам.

Таблица 1

Численность кластеров по числу совпадающих оценок

№	Год	Кластер 1		Кластер 2		Кластер 3	
		Среднее значение	Границы кластеров	Среднее значение	Границы кластеров	Среднее значение	Границы кластеров
Тест № 1	2005–2010	55.86	38–67	55.15	39–75	48.71	32–60
	2010–2015	57.45	46–72	50.33	31–62	49.75	33–62
	2015–2020	49.88	36–61	51.01	33–63	56.84	47–67
Тест № 4	2005–2010	55.86	38–67	55.15	39–75	48.71	32–60
	2010–2015	57.45	46–72	50.33	46–72	49.75	33–62
	2015–2020	49.88	36–61	51.01	33–63	56.84	47–67
Тест № 6	2005–2010	57.21	45–74	47.32	28–58	48.6	33–61
	2010–2015	44.8	21–58	56.89	47–75	47.56	33–60
	2015–2020	57.41	41–73	45.83	41–73	47.52	35–58

Низкая доля совпадения ответов с мнением автора указывает на неоднозначность или неопределенность мнений по этой теме. Определив, какие вопросы (номера ответов) имели высокую долю совпадения ответов с мнением автора, можно установить, какие вопросы были наиболее значимыми для анализа и принятия решений [7].

Заключение

Субъекты оценивают свое взаимодействие с реальностью, исходя из своего набора моделей этой реальности (своих знаний, информации, опыта взаимодействия с реальностью, представлений, предположений о реальности).

Когда субъекты одинаково оценивают что-либо, можно с высокой долей вероятности предположить, что у них модели этого чего-либо совпадают или близки. Ещё раз подчеркнем, что речь не идет о том, правильно или нет респондент оценил суждение. Речь только о том, совпала оценка респондента с оценкой авторов тестов или нет. Не-

совпадение оценок – это повод для обсуждения суждений в рамках дисциплины "Менеджмент".

По поводу любого суждения можно приводить доводы как "за", так и "против". Говорят, сколько менеджеров, столько и мнений. Поэтому, есть мнение [8], что менеджмент – самый сложный вид управления, т.к. менеджеры управляют самыми сложными по своей природе системами – социальными.

Наше исследование позволило сформулировать некоторые статистически обоснованные выводы на основе поставленных целей и задач. В ходе работы был выбран эффективный способ представления данных тестирования, осуществлен выбор метода кластерного анализа и оценена его применимость для работы с данными имеющегося типа и применение его к реальной выборке, осуществлён анализ полученных результатов.

Таким образом, были зафиксированы изменения в значениях описательных статистик в зависимости от периода исследования, что может свидетельствовать о непрерывных трансформациях Федеральных государственных образовательных стандартов или об изменениях социально-экономической обстановки в государстве, которые оказывают влияние на восприятие студентами тех или иных положений гуманитарных дисциплин в области управления социальными системами.

ЛИТЕРАТУРА

1. *Reddin W.J.* Managerial Effectiveness. – N.Y.: McGraw-Hill, 1970.

2. *Тарасенко В.Ф., Тарасенко Ф.П.* Подход к обучению менеджменту, ориентированный на использование методологии и технологии системного анализа // Труды VII международной научно-практич. конф. "Системный анализ в проектировании и управлении". 27 июня – 04 июля 2003. – Спб.: Издательство СпбГПУ, 2003.

3. *Дмитриев Ю.Г. Ерёмина Н.Л., Тарасенко В.Ф.* Детерминационный анализ опросов по тестам Реддина // Материалы международной научной конференции "Математическое и программное обеспечение информационных, технических и экономических систем", Томск, 28–30 мая 2020 г. – Томск, 2020. С. 206–210.

4. *Мандель И.Д.* Кластерный анализ. – М.: Финансы и статика, 1988.

5. *Мастицкий С.Э., Шитиков В.К.* Статистический анализ и визуализация данных с помощью R. – М.: ДМК Пресс, 2015.

6. *Барыкина А.О., Тарасенко В.Ф.* Исследование динамики результатов тестирования учащихся // Математическое и программное обеспечение информационных, технических и экономических систем. Материалы IX Международной молодежной научной конференции. Под общей редакцией И.С. Шмырина. – Томск : Издательство Томского государственного университета, 2022. – С. 179–183.

7. *Барыкина А.О., Тарасенко В.Ф.* Сравнительный анализ результатов тестирования учащихся // Математическое и программное обеспечение информационных, технических и экономических систем. Материалы X-й Международной молодежной научной конференции. Под общей редакцией И.С. Шмырина. – Под общей редакцией И.С. Шмырина. – Томск : Издательство Томского государственного университета, 2023 г. – С. 131–137.

8. *Тарасенко В.Ф.* Моделирование систем менеджмента. – Томск: Изд-во Томск. Гос. Ун-та систем управления и радиоэлектроники, 2018.

ОПТИМИЗАЦИЯ ЭКОНОМИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ С УЧЕТОМ АНОМАЛЬНОГО СПРОСА

Кривко Л.С.
Томский государственный университет
lyubov.krivko_tsu@mail.ru

Введение

При составлении стратегии компании необходимо выбрать метод для оптимизации ассортимента в целях минимизации затрат и максимизации прибыли. При этом важно анализировать влияние многих внешних факторов, одним из немаловажных является спрос конечного потребителя [1]. Его значительное колебание в большую или же в меньшую сторону могут существенно исказить результаты оптимизации, что может повлечь за собой убытки. Поэтому так необходимо уметь модифицировать выбранный метод под различные ситуации.

1. Теоретическая часть

Рассмотрим *КО* – коэффициент оборачиваемости [2], показывающий, сколько раз за год (месяц) возвращаются вложенные в оборотные средства (запасы) деньги. Его можно рассчитать по формуле

$$КО = \frac{R = \text{Выручка от прибыли}}{\overline{x} = \text{Средняя стоимость запасов}}.$$

Оборот *О* [3] определяется как отношение количества дней в рассматриваемом периоде к коэффициенту *КО*, что соответствует $O = \frac{365}{КО}$ и показывает, за сколько дней оборачиваются вложенные в запас деньги. Данные показатели рассматриваются в динамике. Чем выше коэффициент и ниже оборот, тем выше качество управления запасом [4].

Предлагается модифицировать классический метод расчета данных коэффициентов для применения в случае наличия аномального спроса. Для этого нам необходимо рассмотреть критерий для нахождения выброса, а именно критерий Граббса [5] $G = \frac{1}{\sigma}\max\limits_{i=\overline{1,n}}\left|x_i - \overline{x}\right|$, $G = \frac{1}{\sigma}\min\limits_{i=\overline{1,n}}\left|x_i - \overline{x}\right|$, где σ – среднеквадратическое отклонение (СКО), x_i – *i*-е значение выборки, $i = \overline{1,n}$, $\overline{x}$ – среднее значение выборки, *n* – объём выборки.

В случае, когда полученное значение критерия при заданном уровне значимости α больше табличного (критического) (табл. 1), наблюдение x_i признается выбросом.

Таблица 1

Табулированные значения критерия Граббса

n	**Уровень значимости α**				
	0.1	**0.05**	**0.025**	**0.001**	**0.0001**
3	1.148	1.153	1.155	1.155	1.155
4	1.425	1.463	1.481	1.492	1.499
5	1.602	1.672	1.715	1.749	1.780
6	1.729	1.882	1.887	1.944	2.011
7	1.828	1.938	2.020	2.097	2.201

После выявления выброса критерием Граббса [5] для модификации классического метода расчета *КО* и *О* нужно заменить классическое вычисление среднего значения стоимости запасов на усеченное среднее [6], вычисляемое по формуле $\overline{x}^* = \frac{1}{n-2}\sum\limits_{i=2}^{n-1} x_i$.

Тогда значения коэффициента оборачиваемости и оборота при наличии выбросов можно рассчитать, соответственно, по фермулам

$$КО^* = \frac{R}{\overline{x}^*}, \quad (1)$$

$$O^* = \frac{365}{КО^*}. \quad (2)$$

2. Практическая часть

Имеются данные [7] о продажах некоторых товаров в период с января по июль. Внимательно изучив табл. 2 исходных данных, можно отметить слегка завышенную мартовскую продажу товара "Два" и отсутствие майской продажи товара "Девять".

Таблица 2

Исходные данные

Наименование	Средняя цена за период, руб./ед.	Продажи, ед./мес.						
		Янв.	Февр.	Март	Апр.	Май	Июнь	Июль
Один	98	1202	1003	1123	989	1075	1102	1098
Два	250	119	127	165	116	122	107	93
Три	194	322	360	345	312	325	339	328
Четыре	350	100	110	90	100	105	95	100
Пять	200	103	112	106	98	89	109	112
Шесть	100	98	95	93	114	108	120	119
Семь	30	280	295	287	279	293	300	288
Восемь	286	20	25	23	19	18	15	16
Девять	100	50	60	70	30	0	40	60
Десять	230	18	18	19	20	19	22	21

Для расчёта коэффициента оборачиваемости по выручке и оборота необходимо использовать масштабированные данные, в которых есть информация о стоимости запасов для товаров "Два" и "Девять" (представлены в табл. 3).

Таблица 3

Масштабированные данные

Товар	"Два"		"Девять"	
Месяц	Объем реализации, тыс. руб./мес.	Стоимость запасов на конец месяца, тыс. руб.	Объем реализации, тыс. руб./мес.	Стоимость запасов на конец месяца, тыс. руб.
Январь	29 750	7 500	5 000	5 312.5
Февраль	31 570	7 812.5	6 000	5 625
Март	41 250	3 750	7 000	6 562.5
Апрель	29 000	7 187.5	3 000	4 687.5
Май	30 500	9 687.5	0	8 437.5
Июнь	26 750	8 427.5	4 000	3 750
Июль	23 250	7 187.5	6 000	5 626
Итого выручка, руб./мес.	212 250	–	31 000	–

2.1. Товар с завышенным спросом

Для начала рассчитаем коэффициент оборачиваемости и оборота до обнаружения и исключения выброса в выборке, соответствующей стоимости запасов товара "Два":

$$KO = \frac{R}{\overline{x}} = \frac{212\ 250}{7\ 366.07} \approx 29\,,\ O = \frac{212}{KO} = \frac{212}{29} \approx 7\,.$$

Прежде чем применить критерий Граббса к данной выборке, необходимо проверить ее на нормальность. Для этого воспользуемся критерием Шапиро – Уилка [8], который вычисляется по формуле

$$W = \frac{\left(\sum_{i=1}^{n/2} a_i \left(x_{n-i+1} - x_i\right)\right)^2}{\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2}\,.$$

В нашем случае $W = 0.873$, а значение $p\text{-value} = 0.195$, что при заданном уровне значимости $\alpha = 0.1$ позволяет утверждать о нормальности имеющейся выборки стоимости запасов на конец месяца, т.к. $p\text{-value} = 0.196 > \alpha = 0.01$.

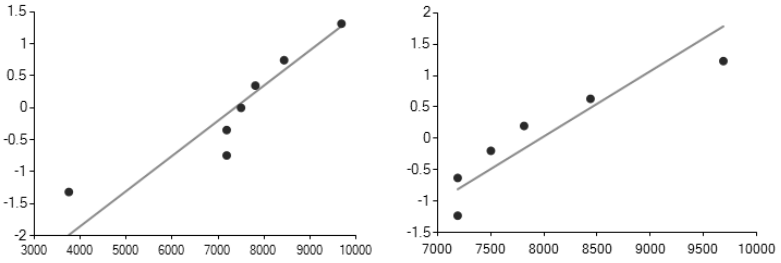
Теперь выдвинем гипотезу [9] относительно стоимости запасов на конец месяца товара "Два", имеющее распределение

x_2 = (7 500, 7 812.5, 3 750, 7 187.5, 9 687.5, 8 427.5, 7 187.5):

H_0: $x_{2,3}$ = 3 750 является выбросом, альтернативная же гипотеза H_1: $x_{2,3}$ = 3 750 не является выбросом.

Теперь рассчитаем критерий Граббса для нахождения нижнего выброса в заданной выборке: $G = \frac{1}{\sigma}\min_{i=1,n}|x_i - \bar{x}| = \frac{1}{1\,820.9}\min_{i=1,n}|3\,750 - 7\,366.07| = 1.986$.

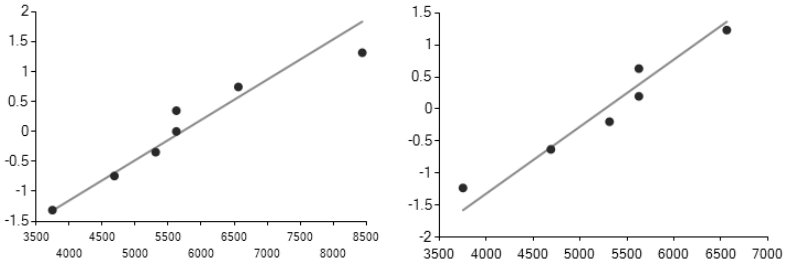
В нашем случае x_i = 3 750, среднее значение выборки $\bar{x}$ = 7 366.07, среднеквадратическое отклонение σ = 1 820.9, а значит G = 1.986. Табличный критерий Граббса при n = 7: G^* = 1.828. Получается, что $G = 1.986 < G^* = 1.828$, следовательно, мы принимаем гипотезу H_0 и считаем $x_{2,3}$= 3 750 выбросом.



а) с выбросом б) без выброса

Рис.1. Нормальные вероятностные графики стоимости запасов товара "Два".

Теперь проверим выборку без выброса на нормальность ранее упомянутым критерием Шапиро – Уилка. Получаем W = 0.850, а значение $p\text{-value} = 0.158 > \alpha = 0.01$ позволяет утверждать о нормальности полученной выборки, что видно и из графиков рис. 2.



а) с выбросом, б) без выброса

Рис. 2. Нормальные вероятностные графики стоимости запасов товара "Девять"

После этого рассчитаем модифицированный коэффициент оборачиваемости и оборота по ранее упомянутым формулам (1) и (2): $KO^* = \frac{R}{\bar{x}^*} = \frac{212\,250}{7\,625} \approx 28,,$

$O^* = \frac{212}{KO} = \frac{212}{28} \approx 8$. Здесь суммарный объем реализации за период остается прежним (R = 212 250), а среднее стоимость запаса $\bar{x}$ = 7 366.07 заменяется на урезанное среднее $\bar{x}^*$ = 7 625, которое на 258.93 единицы больше. Результаты вычисления модифицированного KO^* и O^* представлены в табл. 4.

Таблица 4

Коэффициент оборачиваемости и оборот товара "Два"

	Средняя стоимость запаса, тыс.руб	***KO* за 7 месяцев, раз**	***O*, дни**
С выбросом	7 366.07	29	7
Без выброса	7 625	28	8

2.2. Товар с заниженным спросом

Выполним аналогичные вычисления относительно стоимости запасов товара "Девять", у которого по табл. 2 наблюдается аномальный спрос, а точнее – его отсутствие в мае. Первым делом вычислим коэффициент оборачиваемости и оборота для товара "Девять":

$$KO = \frac{R}{\overline{x}} = \frac{31\,000}{5\,714.29} \approx 5\,,\; O = \frac{212}{KO} = \frac{212}{5} \approx 42\,.$$

Далее проверим выборку на нормальность, чтобы можно было воспользоваться критерием Граббса [5]. Как и ранее, применяем критерий Шапиро – Уилка [8]. В данной ситуации $W = 0.943$, а значение p-value = 0.673, что при заданном уровне значимости $\alpha = 0.1$ дает возможность говорить о нормальности исследуемой выборки стоимости запасов на конец месяца, т.к. $p\text{-value} = 0.673 > \alpha = 0.01$.

После этого выдвигаем гипотезу [9] относительно выборки стоимости запасов на конец месяца товара "Девять", имеющее распределение x_9 = (5 312.5, 5 625, 6 562.5, 4 687.5, 8 437.5, 3 750, 5 625): H_0: $x_{9,5}$ = 3 750 является выбросом, альтернативная гипотеза – H_1: $x_{9,5}$ = 3 750 не является выбросом.

Теперь рассчитаем критерий Граббса для нахождение нижнего выброса в заданной выборке: $G = \frac{1}{\sigma}\max\limits_{i=1,n}\left|x_i - \overline{x}\right| = \frac{1}{1\,484.7}\max\limits_{i=1,n}\left|8\,437.5 - 5\,714.29\right| = 1.834$, в данном случае значение статистики равно G = 1.834. Критическое значение критерия Граббса остается тем же: G^* = 1.828. Из этого следует, что $G = 1.834 < G^* = 1.828$, таким образом, мы принимаем гипотезу H_0 и считаем, что $x_{9,5}$ = 8 437.5 является выбросом.

Снова проверяем выборку на нормальность с помощью критерия Шапиро – Уилка, но уже исключив выброс. Получаем $W = 0.963$, а значение $p\text{-value} = 0.845 > \alpha = 0.01$ позволяет утверждать о нормальности полученной выборки, которая видна на графиках рис 3.

После этого вычислим модифицированный коэффициент оборачиваемости и оборота для товара "Девять": $KO^* = \frac{R}{\overline{x}^*} = \frac{31\,000}{5\,562.5} \approx 6\,,\; O^* = \frac{212}{KO} = \frac{212}{6} \approx 35\,.$

Сравнить итоги расчета классического и модифицированного KO^* и O^* для товара "Девять" можно по табл. 5

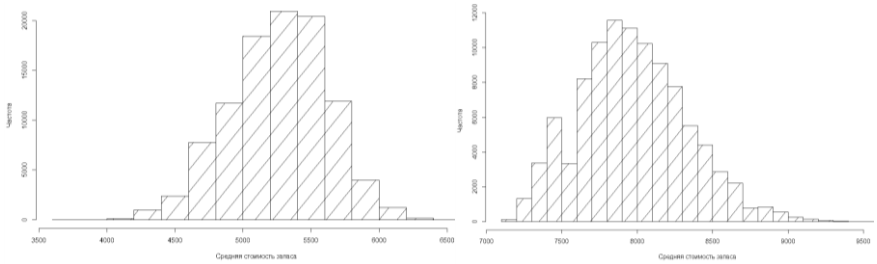
Таблица 5

Коэффициент оборачиваемости и оборот товара "Девять"

	Средняя стоимость запаса, тыс.руб	***KO* за 7 месяцев, раз**	***O*, дни**
С выбросом	5 714.29	5	42
Без выброса	5 562.5	6	35

2.3. Доверительные интервалы для *KO* и *O*

Чтобы найти доверительные интервалы [2] для средней стоимости запаса товара, а после и для коэффициента оборачиваемости и оборота, предлагается применить к данным без выброса бутстреп-метод [10]. Он основан на многократной генерации выборок из числовых значений исходной выборки. Доверительные интервалы будем строить с 90% уровнем доверия. Тогда полагается взять 5 000-е и 95 000-е значение [11] из полученного вариационного ряда средних значений (рис. 3).



а) товар "Два" б) товар "Девять"

Рис. 3. Распределения частот бутстреп статистик средней стоимости запасов

Теперь находим 5 000 и 95 000 среднее значения для товара "Два": $x^*_{(5\ 000)}$ = 7446.25 и $x^*_{(95\ 000)}$ = 8593.75. Тогда границы доверительных интервалов для коэффициента оборачиваемости и оборота вычислим следующим образом:

$$KO^{Л} = \frac{212\ 250}{x^*_{(95\ 000)}} = \frac{212\ 250}{8\ 593.75} \approx 24.7 \le KO \le KO^{П} = \frac{212\ 250}{x^*_{(5\ 000)}} = \frac{212\ 250}{7\ 446.25} \approx 28.5,$$

$$O^{Л} = \frac{212}{KO^{П}} = \frac{212}{28.5} \approx 7.44 \le O \le O^{П} = \frac{212}{KO^{Л}} = \frac{212}{24.7} \approx 8.58,$$

Таким же образом определяем 5 000 и 95 000 среднее значения для товара "Девять": $x^*_{(5\,000)}$ = 4635.75 и $x^*_{(95\,000)}$ = 5833.5, и вычисляем доверительные интервалы:

$$KO^{Л} = \frac{31\ 000}{x^*_{(95\,000)}} = \frac{31\ 000}{5\ 833.5} \approx 5.31 \le KO \le KO^{П} = \frac{31\ 000}{x^*_{(5\,000)}} = \frac{31\ 000}{4\ 635.75} \approx 6.69,$$

$$O^{Л} = \frac{212}{KO^{П}} = \frac{212}{6.69} \approx 31.69 \le O \le O^{П} = \frac{212}{KO^{Л}} = \frac{212}{5.31} \approx 39.93,$$

Итак, с уровнем доверия 90% истинные значения коэффициента оборачиваемости для товара "Два" имеют значения от 24.7 до 28.5, и значения оборота от 7.44 до 8.58. В случае товара "Девять" границы 90% доверительного интервала имеют значения 5.31 и 6.69 для коэффициента оборачиваемости, и 31.69 и 39.93 для оборота.

При этом мы видим, что значения коэффициента оборачиваемости и оборота с выбросом у обоих товаров выходят за границы полученного доверительного интервала, что видно из табл. 6.

Таблица 6

Сравнение результатов *КО* и *О* с выбросом и без

	Товар "Два"		**Товар "Девять"**	
	С выбросом	**Без выброса**	**С выбросом**	**Без выброса**
КО, раз	29	28	5	6
О, дни	7	8	42	35

Заключение

В работе осуществлен расчет показателей оборачиваемости запасов с учётом аномальных значений. Был рассмотрен методы вычисления выброса в критерий Граббса для дальнейшего применения. Также была предложена модификация метода расчёта коэффициента оборачиваемости и оборота путем применения усеченного среднего вместо среднего арифметического. Используя этот метод, менеджеры компании смогут по-новому оценить качество управления запасами и внести изменения для его повышения.

К выборкам с аномальным спросом был применен метод бутстрэп-моделирования для получения 90%-го доверительного интервала коэффициента оборачиваемости и оборота. Результаты бутстреп-моделирования позволили получить более точную оцен-

ку реализации товара с аномальным спросом, что немаловажно при оценке эффективности управления запасами.

ЛИТЕРАТУРА

1. *Зенкова Ж.Н., Кривко Л.С.* Выбор оптимальной стратегии развития предприятия сферы туризма и сервиса с помощью ABC-XYZ-анализа // Материалы международной конференции "Проблемы и решения организации международных транспортных коридоров и логистических центров в Узбекистане", 23–24 ноября 2023, Самарканд. – Самарканд: СамИСИ. –С. 376-379.

2. *Зенкова Ж.Н., Гуров Н.В.* Доверительные интервалы для показателей оборачиваемости оборотных средств предприятия с использованием робастного оценивания // XXI век: итоги прошлого и проблемы настоящего плюс.– 2017. – № 203 (36–37). –С. 52-57.

3. *Бадокин О.В., Лукинский В.В., Малевич Ю.В., Степанова А.С.*, Шульженко Т.Г. Управление запасами в цепях поставок. Учеб. пособие. – СПб.: СПбГИЭУ – 2010. – 372 с.

4. *Гуров Н.В., Зенкова Ж.Н.* Робастная оценка среднего в анализе оборачиваемости оборотных средств предприятия // Математическое и компьютерное моделирование в экономике, страховании и управлении рисками. Материалы V Международной молодежной научно-практической конференции (Саратов, 9–12 ноября 2016 г.). Саратов: Научная книга, 2016. – С. 47–51.

5. *Кривко Л.С., Зенкова Ж.Н.* Оптимизация ассортимента с учетом аномального спроса // Материалы IX Международной молодежной научной конференции "Математическое и программное обеспечение информационных, технических и экономических систем", Томск, 26–28 мая 2022 г., Том 307. Под общей редакцией И.С. Шмырина. Томск: ТГУ, 2022. С. 192–198.

6. *Zenkova Z.N., Kabanova T.V.* The ABC-XYZ Analysis Modified for Data with Outliers // Proceeding GOL'2018 The 4th IEEE International Conference on Logistics Operations Management. April 10–12, 2018, Le Havre, France. – P. 63–68. DOI: 10.1109/GOL.2018.8378073.

7. *Тюменцева Л.С., Зенкова Ж.Н.* Анализ продаж товара с учетом аномального спроса // Труды Томского государственного университета. Т. 305. Серия физико-математическая: Математическое и программное обеспечение информационных, технических и экономических систем: материалы Международной научной конференции. Томск, 28–30 мая 2020 г. / под общ. ред. И.С. Шмырина. Томск: Издательство Томского государственного университета, 2020. – С. 242–248.

8. *Кобзарь А.И.* Прикладная математическая статистика для инженеров и научных работников. М.: ФИЗМАТЛИТ – 2006. – 816 с.

9. *Шуленин В.П.* Математическая статистика. Ч. 1. Параметрическая статистика: учебник. – Томск: Изд-во НТЛ, 2012. – 540 с.

10. *Грибкова Н.В., Хэлмерс Р.* О состоятельности M⋘N-бутстреп-аппроксимации распределений усеченного среднего // Теория вероятностей и ее примения. – 55:1. – 2010. – С. 3–18.

11. *Зенкова Ж.Н., Мусони У., Андриевская А.А.* Статистическое оценивание показателей оборачиваемости оборотных средств с учетом дополнительной информации о квантиле. // Вестник Томского государственного университета, Управление, вычислительная техника и информатика. – №55. – 2021. – С. 35–44.

ОБ ЭФФЕКТИВНОСТИ КОМБИНИРОВАННОЙ ОЦЕНКИ РЕГРЕССИИ НА ОСНОВЕ ИНТЕГРАЛЬНОЙ СКО

Сорокин Р.Д.

Томский государственный университет

sorokinroman063@gmail.com

Введение

В построении математических моделей систем (задача идентификации), обнаружении зависимостей, сглаживании данных важное место занимает функция регрессии, являющаяся условным математическим ожиданием

$$r(x) = M\left(Y \mid X = x\right) = \frac{\int_{-\infty}^{+\infty} yf(x,y)dy}{g(x)},\ x \in R^m,$$

где $f(x,y)$ – совместная функция плотности случайного вектора (X,Y), описывающая взаимосвязь между объясняющей (входной) переменной $X \in R^m$ и переменной отклика (выходной переменной) $Y \in R^1$, $g(x) = \int_{-\infty}^{+\infty} f(x,y)dy$ – функция плотности вектора $X = \left(X_1,...,X_m\right)$. В зависимости от располагаемой исходной информации рассматриваются разные модели этой функции. Параметрические модели $r(x)$ исходят из предполо-

жения, что неизвестную зависимость можно описать известным семейством функций, заданных с точностью до конечного числа параметров. Непараметрические модели не предполагают такого описания и в этом смысле противоположны параметрическим. В [1] наряду с указанными моделями рассмотрены полу-параметрические модели, в [2] предложен метод локальной аппроксимации (МЛА) для синтеза регрессии.

Располагая наблюдениями над входной и выходной переменными $\{(X_i, Y_i)\}_{i=1}^{n}$ объема n необходимо построить оценку $r(x)$. Оценки строятся исходя из выбранной модели регрессии с соответствующими названиями: параметрические [2–4], непараметрические [1–9], полу-параметрические [1], МЛА-оценки [6] и т.д. Точность оценивания определяется разными критериями [1,6]: средняя квадратическая ошибка (СКО), интегральная средняя квадратическая ошибка (ИСКО), эмпирическая средняя квадратическая ошибка (ЭСКО) и др. Проблема повышения точности непараметрических оценок за счет привлечения дополнительной (априорной) информации рассматривалась в [7]. Выделим работы, в которых дополнительная информация выступает в виде априорной догадки [10–15]. Предполагается, что исследователь на основании своего опыта и знаний может задать конкретную функцию (возможно с точность до ряда параметров). Построение оценок линейных функционалов с учётом априорной догадки от распределений вероятностей и анализ их свойств рассматривалось в [10–15].

Цель работы заключается в построении оценки регрессии с учётом априорной догадки и анализе эффективности полученной оценки.

1. Комбинированные оценки регрессии и эффективность

Рассмотрим класс комбинированных оценок функции регрессии вида

$$\hat{R}(x;\lambda) = (1-\lambda)\cdot \hat{r}(x) + \lambda \cdot \phi(x)\,. \tag{1}$$

Здесь $\hat{r}(x)$ – исходная оценка функции регрессии, построенная по статистическим данным $\{X_i, Y_i\}_{i=1}^{n}$ объема n, $\phi(x)$ – априорная догадка, λ – весовой коэффициент, выбираемый из заданного критерия качества оценивания. Различные критерии качества регрессионных оценок изложены в [1,6]. В данной работе используется интегральная СКО (ИСКО)

$$\tilde{S}^2(\lambda) = \int_{R^m} M\left\{(1-\lambda)\cdot \hat{r}(x) + \lambda\cdot\phi(x) - r(x)\right\}^2 g(x)dx\,. \tag{2}$$

Из (2) находим оптимальное $\tilde{\lambda}$, доставляющее минимум ИСКО:

$$\tilde{\lambda} = \frac{\int_{R^m} M\left\{\left(\hat{r}(x)-r(x)\right)\cdot\left(\hat{r}(x)-\phi(x)\right)\right\} g(x)dx}{\int_{R^m} M\left\{\hat{r}(x)-\phi(x)\right\}^2 g(x)dx}\,. \tag{3}$$

При $\tilde{\lambda}$

$$\tilde{S}^2(\tilde{\lambda}) = \int_{R^m} M\left\{\hat{r}(x)-r(x)\right\}^2 g(x)dx - \frac{\left[\int_{R^m} M\left\{\left(\hat{r}(x)-r(x)\right)\cdot\left(\hat{r}(x)-\phi(x)\right)\right\} g(x)dx\right]^2}{\int_{R^m} M\left\{\hat{r}(x)-\phi(x)\right\}^2 g(x)dx}\,. \tag{4}$$

Формула (4) показывает, что ИСКО комбинированной оценки при оптимальном $\tilde{\lambda}$ не превосходит ИСКО $\int_{R^m} M\left\{\hat{r}(x)-r(x)\right\}^2 g(x)dx$ исходной оценки регрессии. Оценку $\hat{R}\left(x;\tilde{\lambda}\right)$ назовем оптимальной в классе (1) в смысле минимума ИСКО. Введём параметры $b(x)$, $D\hat{r}(x)$, $\Delta(x)$, где $b(x) = M\hat{r}(x) - r(x)$ – смещение исходной оценки,

$D\hat{r}(x) = M\left(\hat{r}(x) - M\hat{r}(x)\right)^2$ – её дисперсия, $\Delta(x) = r(x) - \phi(x)$ – отклонение априорной догадки от истинного значения регрессии. Возьмём интегралы от $b(x)$, $D\hat{r}(x)$, $\Delta(x)$ и получим интегральные параметры $\tilde{b}$, $\tilde{D}$, $\tilde{\Delta}$, где $\tilde{b} = \int_{R^m} b(x)g(x)dx$, $\tilde{D} = \int_{R^m} D\hat{r}(x)g(x)dx$, $\tilde{\Delta} = \int_{R^m} \Delta(x)g(x)dx$. При этом $\int_{R^m} M\left\{\hat{R}\left(x;\tilde{\lambda}\right) - r(x)\right\}g(x)dx = \tilde{b} + \tilde{\Delta} = \tilde{\tilde{\Delta}}$. Преобразовав (3) и (4) с учетом $\tilde{b}$, $\tilde{D}$, $\tilde{\Delta}$, получим

$$\tilde{\lambda} = 1 - \frac{\tilde{\Delta}\left(\tilde{b} + \tilde{\Delta}\right)}{\tilde{D} + \left(\tilde{b} + \tilde{\Delta}\right)^2},\ \tilde{S}^2(\tilde{\lambda}) = \tilde{D} + \tilde{b}^2 - \frac{\left(\tilde{D} + \tilde{b}\left(\tilde{b} + \tilde{\Delta}\right)\right)^2}{\tilde{D} + \left(\tilde{b} + \tilde{\Delta}\right)^2}. \quad (5)$$

Чтобы сравнить точность оценивания по ИСКО оптимальной комбинированной оценки $\hat{R}\left(x;\tilde{\lambda}\right)$ с исходной $\hat{r}(x)$, введём показатель эффективности, который обозначим через $\tilde{\upsilon}$:

$$\tilde{\upsilon} = \frac{\int_{R^m} M\left\{\hat{R}\left(x;\tilde{\lambda}\right) - r(x)\right\}^2 g(x)dx}{\int_{R^m} M\left\{\hat{r}(x) - r(x)\right\}^2 g(x)dx} = 1 - \frac{\left(\tilde{D} + \tilde{b}\left(\tilde{b} + \tilde{\Delta}\right)\right)^2}{\left(\tilde{D} + \left(\tilde{b} + \tilde{\Delta}\right)^2\right)\left(\tilde{D} + \tilde{b}^2\right)} = 1 - \frac{\left(\tilde{D} + \tilde{b}\tilde{\tilde{\Delta}}\right)^2}{\left(\tilde{D} + \tilde{\tilde{\Delta}}^2\right)\left(\tilde{D} + \tilde{b}^2\right)}. \quad (6)$$

Формула (6) показывает, что чем меньше значения $\tilde{\upsilon}$, тем выше эффективность оценки $\hat{R}\left(x;\tilde{\lambda}\right)$ в сравнении с исходной $\hat{r}(x)$. Представляет интерес исследовать поведение $\tilde{\upsilon}$ от параметров $\tilde{b}$, $\tilde{D}$, $\tilde{\Delta}$.

2. Анализ эффективности

Рассмотрим какое влияние оказывают параметры $\tilde{b}$, $\tilde{D}$, $\tilde{\Delta}$ на $\tilde{\upsilon}$ и $\tilde{\lambda}$. Различные случаи их взаимоотношений определены в (5) и (6).

2.1. Если $\tilde{b} = 0$ и $\tilde{D} > 0$, то $\tilde{\nu} = \sqrt{\tilde{\Delta}^2 / \tilde{D}}$. Тогда оптимальный весовой коэффициент

$$\tilde{\lambda} = 1 - \frac{\tilde{\Delta}^2}{\tilde{D} + \tilde{\Delta}^2} = 1 - \frac{\tilde{\nu}^2}{1 + \tilde{\nu}^2} = \frac{1}{1 + \tilde{\nu}^2}, \quad (7)$$

а эффективность

$$\tilde{\upsilon} = 1 - \frac{\tilde{D}}{\tilde{D} + \tilde{\Delta}^2} = 1 - \frac{1}{1 + \tilde{\nu}^2} = 1 - \tilde{\lambda}. \quad (8)$$

Из (7) видно, что для оценок $\hat{r}(x)$, где $\tilde{b} = 0$, весовой коэффициент $0 \le \tilde{\lambda} \le 1$. Когда $|\tilde{\Delta}| \mapsto 0$ ($\tilde{\nu} \to 0$), коэффициент $\tilde{\lambda} \to 1$, а $\tilde{\upsilon} \to 0$ (рис. 1–2). В таком случае априорная догадка сильнее воздействует на точность оценивания и $\hat{R}\left(x;\tilde{\lambda}\right)$ принимает вид априорной догадки $\phi(x)$.

Если $|\tilde{\Delta}| \mapsto \infty$, то $\tilde{\lambda} \to 0$, $\tilde{\upsilon} \to 1$, в таком случае априорная догадка при оценивании точности ослабевает и $\hat{R}\left(x;\tilde{\lambda}\right)$ стремится к исходной оценке $\hat{r}(x)$.

Пусть $\tilde{\Delta}$ фиксировано. Когда $\tilde{D} \to \infty$, коэффициент $\tilde{\lambda} \to 1$, а $\tilde{\upsilon} \to 0$, априорная догадка становится предпочтительней. Если $\tilde{D} \to 0$, то $\tilde{\lambda} \to 0$, $\tilde{\upsilon} \to 1$, комбинированная оценка переходит в исходную оценку.

2.2. Предположим, что в (5) и (6) $\tilde{\Delta} = 0$. Тогда $\tilde{\lambda} = 1$, $\tilde{\upsilon} = 0$, (рис. 1–6), $R\left(x,\tilde{\lambda}\right) = \phi(x)$.

2.3. Вернёмся к (5). Если $b \neq 0$, то весовой коэффициент может быть $\tilde{\lambda} < 0$, $\tilde{\lambda} > 1$ (рис. 3–6), когда $\tilde{\Delta}\tilde{b} > \tilde{D}$, $\tilde{\lambda} < 0$. Коэффициент $\tilde{\lambda} > 1$, когда $\tilde{b}$ не принадлежит интервалу $\left(-\tilde{\Delta}, \tilde{\Delta}\right)$, $\tilde{\Delta} > 0$.

2.4. Рассмотрим случай, когда $\tilde{b} + \tilde{\Delta} = 0$, при $\tilde{\tilde{\Delta}} = 0$ (рис. 7–8). Тогда из (5) и (6) следует $\tilde{\lambda} = 1$, но не следует, что $\tilde{\Delta} = 0$. Эффективность определяется $\upsilon = 1 - \tilde{D} \Big/ \left(\tilde{D} + \tilde{\Delta}^2\right)$.

3. Представление эффективности в графиках

Наглядным представлением эффективности является графическое изображение $\tilde{\upsilon}$ в зависимости от параметров $\tilde{b}$, $\tilde{D}$, $\tilde{\Delta}$.

3.1. При $\tilde{b} = 0$ из (7), (8) вытекают следующие графики (рис. 1,2):

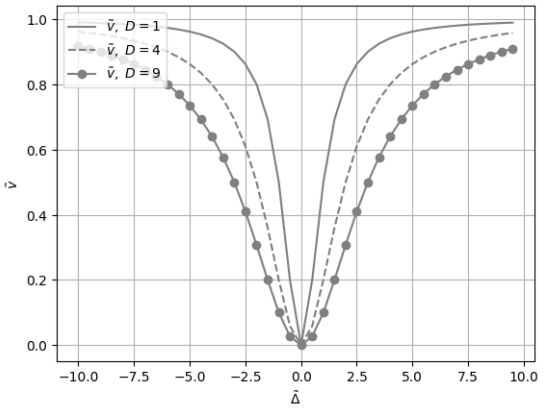


Рис. 1. Зависимость $\tilde{\upsilon}$ от $\tilde{\Delta}$ при фиксированных $\tilde{D}$ ($\tilde{b} = 0$)

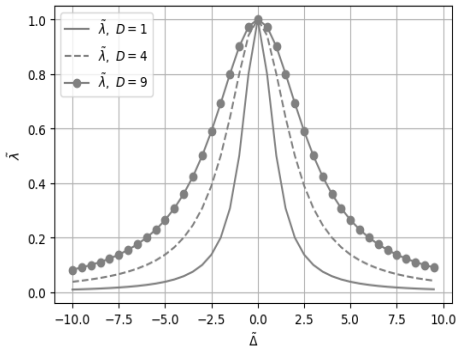


Рис. 2. Зависимость $\tilde{\lambda}$ от $\tilde{\Delta}$ при фиксированных $\tilde{D}$ ($\tilde{b} = 0$)

3.2. При $\tilde{b} \neq 0$ и $\tilde{b} = 0$, $\tilde{D} = 1$, из (7), (8) вытекают следующие графики (рис. 3,4):

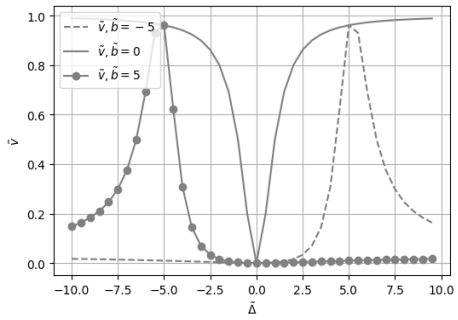


Рис. 3. Зависимость $\tilde{\upsilon}$ от $\tilde{\Delta}$ при фиксированных $\tilde{D}$

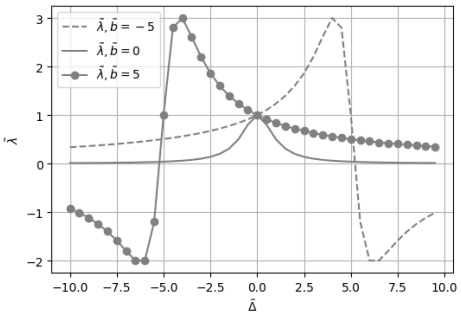


Рис. 4. Зависимость $\tilde{\lambda}$ от $\tilde{\Delta}$ при фиксированных $\tilde{D}$

3.3. При $\tilde{\Delta} \neq 0$ и $\tilde{\Delta} = 0$, $\tilde{D} = 1$, из (7), (8) вытекают следующие графики (рис. 5,6):

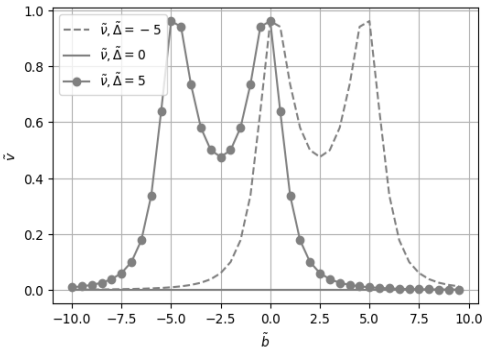


Рис. 5. Зависимость $\tilde{\upsilon}$ от $\tilde{b}$ при фиксированных $\tilde{\Delta}$

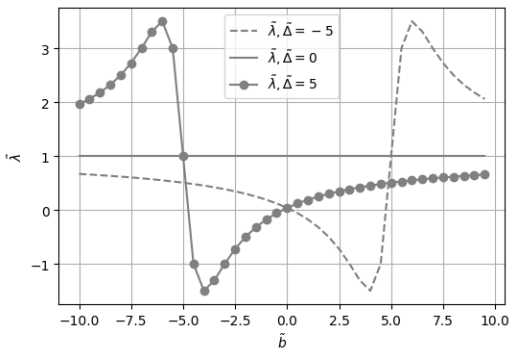


Рис. 6. Зависимость $\tilde{\upsilon}$ от $\tilde{b}$ при фиксированных $\tilde{\Delta}$

3.4. При $\tilde{b}+\tilde{\Delta}=0$, $\tilde{D}=1$, $\tilde{\lambda}=1$, $\tilde{\upsilon}=[0,...,1]$, из (7), (8) вытекают следующие графики (рис. 7,8):

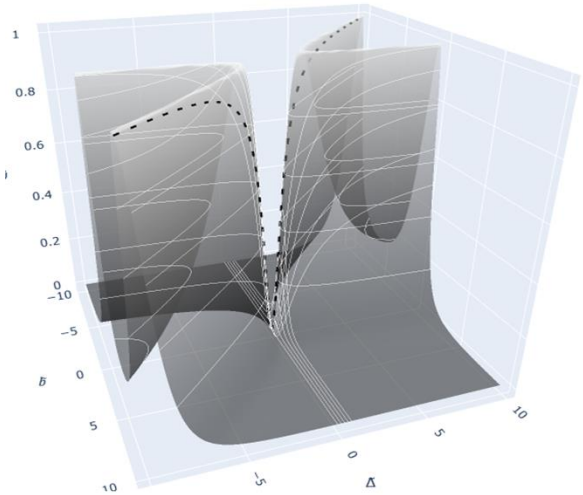


Рис. 7. Зависимость $\tilde{\upsilon}$ от $\tilde{b}$ и $\tilde{\Delta}$ при фиксированном $\tilde{D}$ (пунктир на графике – $\tilde{b}+\tilde{\Delta}=0$)

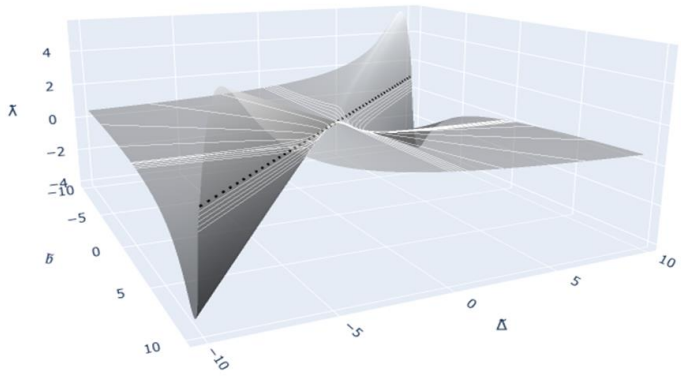


Рис. 8. Зависимость $\tilde{\lambda}$ от $\tilde{b}$ и $\tilde{\Delta}$ при фиксированном $\tilde{D}$ (пунктир на графике – $\tilde{b}+\tilde{\Delta}=0$)

Заключение

Описанный класс оценок неизвестной функции регрессии при оптимальном весовом коэффициенте позволяет уменьшить ИСКО комбинированной оценки за счет использования априорной догадки об исследуемой регрессии. Анализ эффективности показал, что комбинированные оценки могут иметь более высокую точность (меньшую ИСКО) при надлежащем взаимоотношении трёх интегральных параметров $\tilde{b}$, $\tilde{D}$, $\tilde{\Delta}$.

Некоторые соотношения этих параметров могут быть заранее известны, что позволит вычислить оптимальный весовой коэффициент. В иных случаях требуется строить адаптивные комбинированные оценки путем оценивания оптимального весового коэффициента по исходной выборке.

ЛИТЕРАТУРА

1. *Хардле В.* Прикладная непараметрическая регрессия: Пер. с англ. – М., Мир, 1993. 349 с.
2. *Watson G.S.* Smooth regression analysis // Sankhya. Indian J. Statist. – 1964. – Vol. A26. – P. 359–372.
3. *Надарая Э.А.* Об оценке регрессии // Теория вероятностей и ее применения. – 1964. – Т. 19, Вып.1. – С. 147–149.
4. *Gasser T., Muller H.-G.* Kernel Estimation of Regression Functions // Lect. Notes Math. – Vol. 757. – P. 23–68.

5. *Stone C.J.* Consistent Nonparametric Regression // Ann. Statist. – 1977. –Vol. 5. – № 4. – P. 595–645.

6. *Катковник В.Я.* Непараметрическая идентификация и сглаживание данных: метод локальной аппроксимации. – М.: Главная редакция физико-математической литературы, 1985. – 336 с.

7. *Дмитриев Ю.Г., Кошкин Г.М.* Использование дополнительной информации при непараметрическом оценивании функционалов плотности // Автоматика и телемеханика. – 1987. – № 10. – С. 47–59.

8. *Алексеев В.Г.* О непараметрических оценках кривых и поверхностей регрессии // Автоматика и телемеханика. – 1988. – № 7. – С. 81–87.

9. *Васильев В.А., Добровидов А.В., Кошкин Г.М.* Непараметрическое оценивание функционалов от распределений стационарных последовательностей. – М.: Наука, 2004. – 508 с.

10. *Скрипин С.В.* Свойства комбинированной оценки регрессии при конечных объемах выборок // Известия Томского политехнического университета. – 2008. Т. 313, № 5. – С. 10–14.

11. *Dmitriev Yu.G., Koshkin G.M., Lukov V.Yu.* Combined Identification Algorithms. Applied Methods of Statistical Analysis // Nonparametric Methods in Cybernetics and System Analysis – AMSA'2017, Krasnoyarsk, Russia, 18–22 September, 2017: Proceedings of the International Workshop. – Novosibirsk: NSTU publisher, 2017. – P. 19-27.

12. *Вилкина И.Ю., Дмитриев Ю.Г., Кошкин Г.М.,* Алгоритмы идентификации и прогнозирования для комбинированных моделей // Математическое и программное обеспечение информационных, технических и экономических систем: материалы Международной научной конференции. Томск, 28–30 мая 2020 г. под общ. ред. И.С. Шмырина. – Томск: Издательство Томского государственного университета, 2020. – С. 201–206.

13. *Dmitriev Yu.G, Tarassenko P.F., Ustinov Yu.K.* On Estimation of Linear Functional by Utilizing a Prior Guess // A. Dudin et al. (Eds.): ITMM-2014, CCIS 487. P. 82–90.

14. *Дмитриев Ю.Г., Тарасенко П.Ф.* О комбинированных оценках линейного функционала //Информационные технологии и математическое моделирование (ИТММ-2014) : Материалы XIII Международной начно-практической конференции им. А.Ф. Терпугова (20–22 ноября 2014 г.). Томск: Изд-во Том-го ун-та, 2014. Ч. 1. – С. 25–29.

15. *Dmitriev Yu.G, Tarassenko P.F.* On Adaptive Estimation Using a Prior Guess. Proceedings The International Workshop, Applied Methods of Statistical Analysis. Nonparametric Approach. Novosibirsk, Russia. Novosibirsk, 14–15 September, 2015. – NSTU publisher. – P. 49–55.

V. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, МАШИННОЕ ОБУЧЕНИЕ, БОЛЬШИЕ ДАННЫЕ

ПОИСК АРХИТЕКТУРЫ ИМПУЛЬСНОЙ НЕЙРОННОЙ СЕТИ ПРИ ПОМОЩИ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Волков В.К.
Томский государственный университет
viktor.volkov.mailbox@gmail.com

Введение

За последние пару десятилетий искусственные нейронные сети (ИНС) прошли большой путь модернизации, в результате которого были сформированы ключевые концепции, архитектуры и подходы, определяющие облик современных state-of-the-art (SOTA) моделей. Однако, как и у большинства технологий, у ИНС есть свои ограничения и проблемы. В частности, высокие затраты электроэнергии и времени на обучение, а также низкая эффективность по сравнению со скоростью обучения головного мозга человека. Данный факт стал одной из основ старта активных исследований нового типа ИНС – Импульсных нейронных сетей (ИмНС) [1]. Этот тип ИНС обладает совершенно иной моделью искусственного нейрона, работающей на основе единичных импульсов во времени. ИмНС характеризуются более низким энергопотреблением во время обучения и работы, такую модель можно развернуть на более широком классе устройств.

Поиск методов эффективного обучения и разработка подходов проектирования архитектуры подобных ИНС все еще остаются слабо изученными областями, поэтому представляют особый интерес, а их изучение обладает высокой актуальностью и имеет широкое поле для теоретических и экспериментальных исследований.

Для решения данной задачи неплохо себя зарекомендовали генетические алгоритмы – подвид эволюционных алгоритмов, порой превосходящие альтернативные методы [2]. Свое название генетический алгоритм получил из-за того, что в его основе лежат операции наследования, мутации, скрещивания и обмена генами, которые свойственны живым организмам и являются основой для их эволюции.

1. Импульсные нейронные сети

Одним из первых исследований, посвященных импульсным нейронным сетям, является работа [1], в которой вводится понятие поколений ИНС, подробно рассматриваются основы работы ИмНС и проводится сравнение с другими актуальными на тот момент видами ИНС. Автором предлагается разделить имеющиеся ИНС на три поколения. Первым поколением являются нейронные сети, основанные на модели МСР [3]. Ко второму поколению относят нейронные сети на основе нейронов, использующих функцию активации. Стоит заметить, что взаимодействие между биологическими нейронами происходит при помощи электрических импульсов [4]. Модель МСР, обладая более схожими с биологическими нейронами свойствами, не может рассматриваться как наиболее близкая, т.к. в ней момент появления импульсов на выходах нейронов сети строго синхронизирован. Накопленные экспериментальные данные позволили иначе посмотреть на процесс обмена информации в головном мозге, что стало мощным толчком развития третьего поколения ИНС, которые используют в своем составе модели искусственных нейронов, более точно отражающие реальную динамику процессов. Наглядное сравнение между поколениями ИНС представлено на рис. 1.

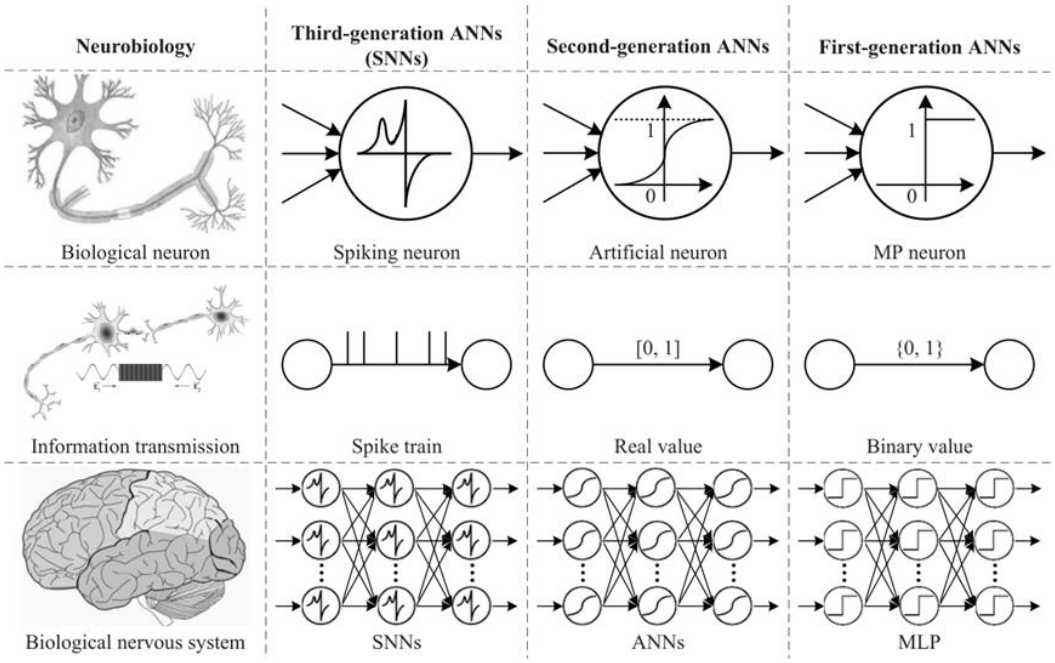


Рис. 1. Сравнение работы ИНС разных поколений [5]

В основе современных ИмНС зачастую используется модель нейрона "интегрировать и сработать с утечками". Ее сильными сторонами являются низкая сложность вычислений и возможность достаточно точно повторять динамику работы реальных нейронов. В данной модели нейрон представлен мембраной, на которой накапливается электрический заряд. При преодолении порогового значения генерируется выходной импульс и мембрана теряет большую часть своего заряда. Также существует механизм пассивной утечки, постепенно снижающий заряд на мембране. Поведение модели описывается соответствующим дифференциальным уравнением [6,7] однако для работы удобнее использовать его дискретное представление [8]:

$$U(t) = \beta U(t-1) + WX(t) - S_{\text{out}}(t-1)\theta,$$

где $U(t)$ – потенциал мембраны, $WX(t)$ – ток, попадающий на мембрану, $\beta = \exp\left(-1/\tau_m\right)$, τ_m – временная константа нейрона, $S_{\text{out}}(t-1)$ – функция сброса потенциала, θ – величина потенциала, которую потеряет мембрана в случае активации функции сброса. Функция $S_{\text{out}}(t) \in \{0,1\}$ где состояние $S_{\text{out}} = 1$ соответствует активированной функции, а $S_{\text{out}} = 0$ – не активированной. Данную функцию можно представить как:

$$S_{\text{out}}(t) = \Theta\left(U(t) - U_{th}\right), \tag{1}$$

где $\Theta(\cdot)$ – функция Хэвисайда, U_{th} – пороговое значение потенциала мембраны, при котором формируется импульс.

В процессе вычисления градиента для метода обратного распространения ошибки производная функции (1) обращается в 0 или бесконечность, делая дальнейшие вычисления невозможными: $\frac{dS}{dU} \in \{0,\infty\}$. Для устранения данной проблемы необходимо заменить функцию активации при вычислении производной следующим образом:

$$\frac{\partial S}{\partial U} \to \frac{\partial \tilde{S}}{\partial U} = \frac{1}{1 + (U\pi)^2}. \tag{2}$$

В (2) используется функция $\tilde{S} = \arctan(\pi U)/\pi$. Данный метод называется Surrogate Gradient Learning [9], благодаря ему обеспечивается возможность обучения многослойных ИмНС.

2. Поиск архитектуры ИНС

Одна из важных целей, стоящей перед разработчиками ИНС – поиск архитектуры, позволяющей добиться максимальной точности в решении поставленной задачи. В некоторых задачах такой поиск является еще более сложным, т.к. итоговая модель должна быть способна работать на оборудовании с различными ограничениями. Для этого ранее использовались ручной поиск и тестирование решений, однако современные автоматизированные способы поиска архитектуры позволяют находить более совершенные варианты [10,11].

Обучение с подкреплением и эволюционные алгоритмы (ЭА) являются наиболее старыми методами, применяемыми для решения данной задачи. Архитектуры, получаемые при помощи ЭА, могут превосходить по эффективности результаты работы обучения с подкреплением [2]. Генетический алгоритм (ГА) является эвристическим алгоритмом поиска и подвидом эволюционного алгоритма. Он используется для поиска оптимального решения в широком классе задач. Свое название алгоритм получил из-за того, что в его основе лежат операции наследования, мутации, скрещивания и обмена генами, которые свойственны живым организмам и являются основой для их эволюции. Общая схема работы представлена на рис. 2. Далее будет описана его реализация в рамках данной работы.

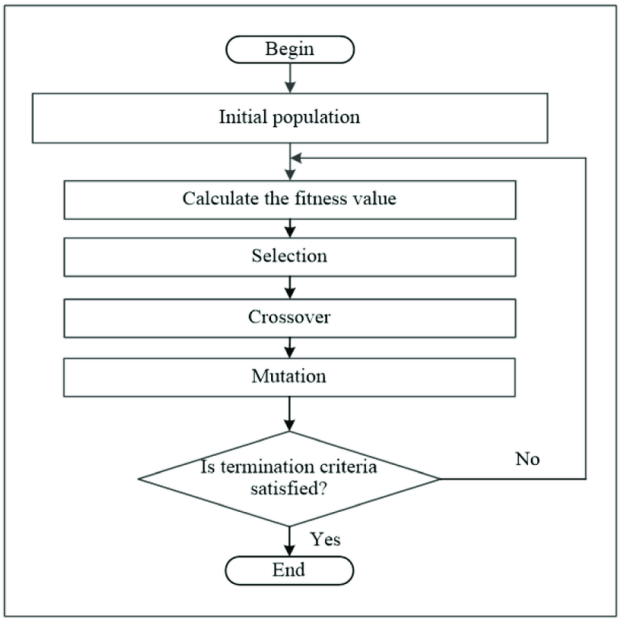


Рис. 2. Диаграмма работы генетического алгоритма [12]

Архитектура нейронной сети кодируется при помощи вектора, каждый из элементов которого соответствует какому-то гену. Данный вектор называется генотипом. Для начала работы алгоритма требуется генерация стартового набора генотипов. Чтобы оптимизировать вычисления, размер популяции постоянен. Следующим шагом является оценка параметра приспособленности, характеризующего качество оцениваемого решения. В данной работе такая оценка проводится путем обучения модели с заданной

архитектурой на обучающей выборке с последующей оценкой ее точности на тестовой части датасета.

Когда каждая из архитектур оценена, необходимо отобрать определенное количество лучших вариантов, которые будут использоваться в дальнейшем. Одним из наиболее простых вариантов является ранжирование с последующим отбором наиболее удачных решений. Генотипы отобранных архитектур используются для формирования промежуточной популяции – популяции потомков. Ее величина определяется как разница между целевым размером популяции и размером популяции отобранных архитектур. Для формирования популяции потомков служат операции кроссинговера и мутации. В ИНС прямого распространения параметры слоев могут быть записаны в виде вектора значений int32. Для проведения операции кроссинговера выбираются два случайных вектора, чьи элементы попарно кодируются в код Грэя [13]. Поскольку длина полученных последовательностей может отличаться, проводится выравнивание путем добавления незначащих нулей. После того, как последовательности выравнены, происходит разбиение в одинаковых местах на одинаковое количество частей (зачастую – две). Полученные части попарно группируются в наборы, после чего из каждого набора выбирается случайно одна часть. Таким образом, составляется новая последовательность, которая обратно декодируется в int32. Операция кроссинговера выполняется с заданной вероятностью и может не произойти; в таком случае для формирования нового генотипа используется соответствующий элемент одного из исходных векторов, выбираемого случайно. Операция мутации последовательно с определенной вероятностью декодирует элементы генотипа в коды Грэя. В полученной последовательности случайным образом меняются биты, после чего происходит декодирование. Работа оператора кроссинговера обеспечивает поиск новых архитектур, а оператор мутации вносит дополнительную случайность в данный процесс, препятствуя схождению ГА к локальному минимуму. После того, как поколение потомков сформировано, производится оценка полученных моделей, после чего из отобранных генотипов и генотипов потомков формируется новое поколение, в которое попадают лучшие архитектуры. Если в получившейся популяции присутствует модель с требуемым уровнем адаптации или выше, то алгоритм прекращает работу. Также работа ГА может быть остановлена при достижении максимального числа циклов.

3. Методика проведения эксперимента

Для проведения вычислений использовались бесплатные вычислительные мощности, предоставляемые сайтом Kaggle. В качестве видеоускорителя выбран GPU P100. Описанная выше реализация использует жесткие ограничения на размеры создаваемых популяций из-за необходимости соблюдения максимального времени работы. Параметры ГА представлены в табл. 1. В процессе работы проводилась запись среднего уровня адаптации в популяции и уровень адаптации лучшей модели в популяции.

Таблица 1

Параметры работы ГА

№	Датасет	Размер популяции	Процент отбора	Вид отбора	Максимальное число нейронов в слое	Минимальное число слоев	Максимальное число слоев	Число эпох
1	MNIST	50	50%	Отбор лучших	1200	3	6	20
2	MNIST	50	50%	Отбор лучших	1200	3	6	20
3	MNIST	50	50%	Отбор лучших	1200	3	6	20
4	CIFAR10	50	50%	Отбор лучших	1200	3	7	20
5	CIFAR10	50	50%	Отбор лучших	1200	3	7	20

Во всех экспериментах используется вероятность кроссинговера 65% и вероятность мутации 10%. Оператор кроссинговера – одноточечный.

4. Обзор результатов эксперимента

Первые несколько запусков реализуемого ГА были проведены на датасете MNIST. Результат замеров состояния популяции представлены на рис. 3. Для данного датасета измерения проводились 3 раза.

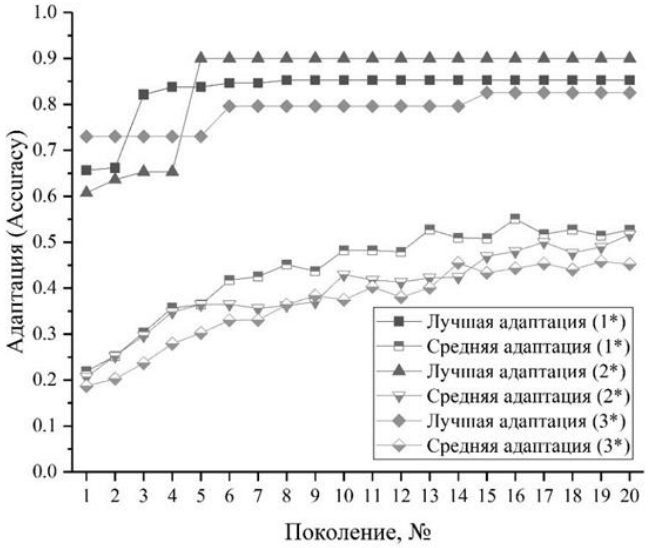


Рис. 3. Динамика изменения адаптации в популяции для работы ГА на датасете MNIST
(на легенде в скобках указан номер проводимого испытания)

На фоне экспериментов 1 и 2 заметно выделяется эксперимент 3. Для данного эксперимента характерна более низкая средняя адаптация, но уже после первого цикла ГА в популяции присутствует модель с адаптацией, превосходящей лучшие модели в остальных экспериментах. Другим важным отличием является динамика изменения качества моделей. На протяжении всех поколений в популяции не возникает модель, чье качество значительно отличается от лучшей модели предыдущего поколения, вследствие чего наблюдается два небольших перехода и алгоритм не выходит на плато. Несмотря на этот факт, итоговый уровень адаптации лучшей модели в данном эксперименте является самым низким. Среднее качество моделей в популяции также демонстрирует более низкий уровень на протяжении большинства эпох.

Из полученных данных видно, что алгоритм позволяет преодолеть порог точности в 90%, но целевое значение ~97% [14] не достигается. Стоит отметить, что в текущей реализации не используются полноценно динамические свойства ИмНС, поэтому подобные различия ожидаемы. Также после пятого поколения наблюдается тенденция снижения скорости роста среднего уровня адаптации в поколении и начинают наблюдаться колебания данной характеристики. Это выражается в виде отдельных резкого увеличения или уменьшения уровня средней адаптации с последующим возвратом значения до начального значения или несколько выше.

По полученным данным можно сделать вывод, что при заданных параметрах, основной этап активного улучшения решения приходится на первые пять поколений, после чего прогресс в улучшении начинает постепенно замедляться. Возможные механизмы возникновения данного эффекта будут описаны ниже.

В дальнейшем работа ГА была проверена на датасете CIFAR10, результаты представлены на рис. 4.

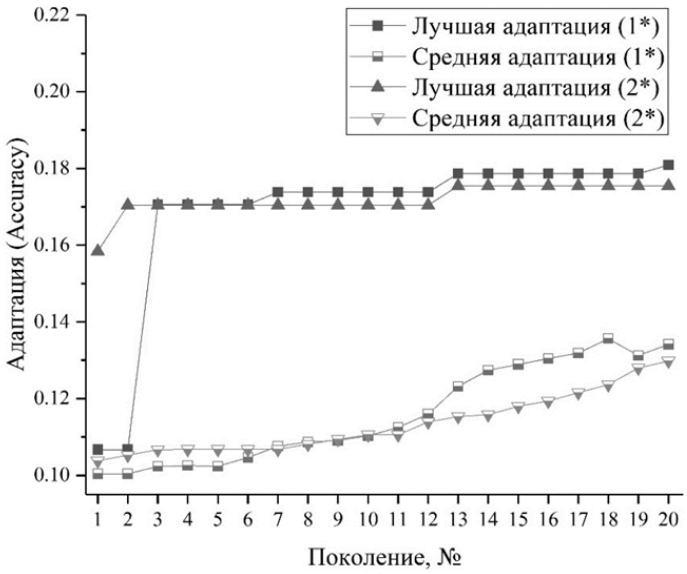


Рис. 4. Динамика изменения адаптации в популяции для работы ГА на датасете CIFAR10
(на легенде в скобках указан номер проводимого испытания)

Динамика улучшения средней адаптации и адаптации лучшей популяции в модели для данных экспериментов сильно отличается от того, что было получено на датасете MNIST. Прежде всего, стоит отметить присутствие нескольких переходов повышения качества лучшей модели в обоих экспериментах, алгоритм не выходит на плато. Присутствует заметный рост качества лучшей модели в эксперименте 1, при переходе от 2-го к 3-му поколению. Однако в дальнейшем в обоих экспериментах различие между значениями данной характеристик остается небольшим, а также присутствует практически идентичный рост при переходе с 12-го на 13-е поколение. Уровень средней адаптации в обоих популяциях также близок, особенно с 6-го по 12-е поколение, после чего заметно некоторое расхождение. Несмотря на более быстрый рост в эксперименте 1, в 19-м поколении наблюдается заметное снижение уровня средней адаптации, в то же время для эксперимента 2 подобного не возникает. Стоит отметить, что для ГА при работе на данном датасете менее свойственны флуктуация величины адаптации популяций, а также присутствует устойчивая тенденция роста среднего уровня адаптации, чего не наблюдается для датасета MNIST. Скорость схождения алгоритма заметно меньше, что снижает качество итогового результата.

Одной из наиболее очевидных причин, ограничивающих эффективность работы ГА, является малый размер популяции, из-за чего начальный набор моделей оказывает заметное влияние на скорость поиска оптимального решения. Влияние данного фактора дополнительно усиливается принципом работы механизма отбора потомков в новую популяцию. Поскольку размер популяции имеет ограниченный размер, количество детей, которые будут участвовать в процессе формирования новой популяции также ограничен, в том числе из-за текущих требований по времени выполнения поиска. Это приводит к сужению пространства поиска решений на этапе формирования набора потомков, что может дополнительно снижать скорость схождения ГА к оптимальному решению. Учитывая, что в формировании новой популяции участвуют только отобранные родители и ограниченный набор потомков, чье качество может оказаться хуже, чем у особей, не прошедших отбор, средняя адаптация может снижаться, что наглядно вид-

но в экспериментах на датасете MNIST. Для более сложных задач это менее выражено из-за меньшей величины изменений характеристик популяции и из-за невозможности находить заметно более качественное решение за один цикла работы. Позитивно на скорость схождения алгоритма влияет механизм переноса наиболее удачной модели из старой популяции в новую, стабилизирущий динамику характеристик популяции.

Заключение

В данной работе рассмотрена модель нейрона "интегрировать-и-сработать" с утечками, кодирование информации для ИмНС, метод обучения Surrogate Gradient Learning, а также принципы работы генетического алгоритма и его применение для поиска оптимальной архитектуры ИмНС. Работа генетического алгоритма продемонстрирована для ИмНС прямого распространения в задачах классификации для датасетов CIFAR и MNIST, выявлены особенности работы в текущей реализации. Полученные результаты указывают на то, что текущая реализация ГА успешно работает и справляется с поставленной задачей, однако присутствует ряд технических ограничений, снижающих качество получаемого результата. Планируется проведение необходимых доработок, повышающих общую производительность и исправляющих текущие недостатки. Основным направлением для модернизации является оптимизация процесса оценки уровня адаптации полученного решения, т.к. низкая производительность этой операции в значительной мере ограничивает возможности по исследованию работы ГА. Решение данной проблемы повысит скорость работы ГА, делая возможным проведение более длительных экспериментов, а также расширит возможности по настройке алгоритма, сняв значительные ограничения на допустимые значения параметров работы. Также будет улучшен процесс формирования популяции потомков путем внедрения нового алгоритма отбора, улучшающего размер пространства поиска, соблюдая баланс со временем, затрачиваемым на данный процесс. Помимо этого, будет проведен пересмотр логики работы этапа селекции в сторону сохранения исходной популяции, позволяя всем моделям участвовать за право попасть в новую популяцию. Данные меры направлены на ускорение сходимости ГА, а также на снижение флуктуаций характеристик популяции.

ЛИТЕРАТУРА

1. *Maass W.* Networks of spiking neurons: The third generation of neural network models // Neural Networks. – 1997. – V. 10. –№ 9. – P. 1659–1671.

2. *Real E. et al.* Regularized Evolution for Image Classifier Architecture Search // arXiv:1802.01548. arXiv, 2019.

3. *McCulloch W.S., Pitts W.A.* A logical calculus of the ideas immanent in nervous activity // Bulletin of Mathematical Biophysics. – 1943. – V. 5. – P. 115–133.

4. *Lovinger D.M.* Communication Networks in the Brain // 2008. – V. 31. – № 3.

5. *Wang X., Lin X., Dang X.* Supervised learning in spiking neural networks: A review of algorithms and evaluations // Neural Networks. – 2020. – V. 125. – P. 258–280.

6. *Gerstner W., Kistler W.M.* Spiking Neuron Models: Single Neurons, Populations, Plasticity.

7. *Burkitt A.N.* A Review of the Integrate-and-fire Neuron Model: I. Homogeneous Synaptic Input // Biol Cybern. – 2006. – V. 95. – № 1. – P.1–19.

8. *Eshraghian J.K. et al.* Training Spiking Neural Networks Using Lessons From Deep Learning // Proc. IEEE. – 2023. – V. 111. – № 9. – P. 1016–1054.

9. *Fang W. et al.* Deep Residual Learning in Spiking Neural Networks // arXiv:2102.04159. – arXiv, 2022.

10. *Zoph B., Le Q.V.* Neural Architecture Search with Reinforcement Learning // arXiv:1611.01578. – arXiv, 2017.

11. *Zoph B. et al.* Learning Transferable Architectures for Scalable Image Recognition // arXiv:1707.07012. – arXiv, 2018.

12. *Albadr M.A. et al.* Genetic Algorithm Based on Natural Selection Theory for Optimization Problems // Symmetry. – 2020. – V. 12. – № 11. – P. 1758.

13. *Charbonneau P.* An Introduction to Genetic Algorithms for Numerical Optimization // Technical Report NCAR/TN-450+IA. University Corporation for Atmospheric Research.

14. *Zhang A. et al.* Fast and robust learning in Spiking Feed-forward Neural Networks based on Intrinsic Plasticity mechanism // Neurocomputing. – 2019. – V. 365. – P. 102–112.

АВТОМАТИЧЕСКАЯ КЛАССИФИКАЦИЯ И СЕГМЕНТАЦИЯ ОПУХОЛЕЙ ГОЛОВНОГО МОЗГА НА СНИМКАХ МРТ

Закиев Я.Т.
Томский государственный университет
ya@stud.tsu.ru

Введение

По данным межконтинентального ракового регистра, объединяющего данные из 86 раковых регистров 5 континентов, заболеваемость первичными опухолями головного мозга (включая менингиомы) составляет 6–19 случаев на 100 тысяч мужского и 4–18 случаев на 100 тысяч женского населения. Данная мировая статистика собирается Международным агентством по изучению рака (International Agency for research on cancer) при участии Всемирной Организации Здравоохранения (источник – Информационный портал об опухолях мозга). С целью диагностики в наше время производится ручная сегментация МРТ-изображений.

Согласно [1], основные проблемы ручной сегментации МРТ изображений опухолей головного мозга на сегодняшний день в том, что она:

1. Требует больших затрат времени радиологов высокой квалификации.
2. Даже внутри одного единственного исследовательского центра нет единого стандарта границ сегментации опухоли, что говорит о субъективизации, также есть определенный процент ошибок. В целом этот пункт можно охарактеризовать, как человеческий фактор, который играет роль в диагностике опухолей головного мозга

Также отмечается, что постепенно методы машинного и глубинного обучения могут стать стандартом для этих целей.

В соответствии с этим целью данной работы стало создание и сравнение моделей машинного и глубинного обучения, способных предсказывать наличие опухоли на медицинских изображениях и производить их сегментацию, с получением модели на современном уровне показателей.

1. Основная часть

В первом эксперименте будет произведена предварительная сегментация изображения для попытки убрать линию черепа и здоровые мягкие ткани, как не несущие ползной информации и затрудняющие работу алгоритма. Далее просходит извлечение признаков с использованием матрицы GLCM (матрица смежности или матрица совместной встречаемости уровней яркости) и вычисление по ней статистических характеристик, которые подаются на вход ряду классических классификаторов. Во втором эксперименте сегментация будет самостоятельной задачей помимо классификации. К набору данных перед передачей нейросетевому классификатору Resnet50 с использованием Transfer learning применяется аугментация. Перед сегментацией нейросетью ResUnet с архитектурой coder-decoder к маскам опухолей была применена нормализация и центрирование.

Метод кластеризации данных Fuzzy C-Means (FCM). В нем один элемент данных может принадлежать двум или более кластерам. Метод FCM, разработанный Данном в 1973 г. [2] и улучшенный Бездеком в 1981 г. [3], часто используется в сегментации изображений. Использование алгоритма FCM на основе интенсивности, где FCM разделяет изображение на заранее заданное количество кластеров (K), дает нечеткую принадлежность (U), чтобы описать степень сходства одного пикселя с каждым кластером [4]. Это – обобщение k-Means, которое позволяет точкам данных принадлежать не только одному кластеру, но и иметь степень принадлежности к нескольким кластерам в виде числа от 0 до 1.

Алгоритм FCM

1. **Инициализация.** Значения центроидов кластеров и степени принадлежности (wij) инициализируются случайным образом, с условием, что сумма степеней принадлежности для каждой точки равна 1.

2. **Повторение шагов до сходимости.** Вычисляются новые центры кластеров (c_j) и обновляются степени принадлежности (w_{ij}) до тех пор, пока алгоритм не сойдется или не будет достигнуто максимальное количество итераций.

3. **Результат.** По завершении алгоритма каждая точка данных имеет степени принадлежности к каждому из k кластеров.

Матрица GLCM [5] предоставляет информацию о том, какие пары пикселей с какими уровнями яркости и в заданном направлении сдвига встречаются в изображении. Эти данные могут использоваться для вычисления упомянутых различных статистических характеристик текстуры изображения. Многие исследователи составляют различные комбинации из выделенных признаков в паттерны, принимающие различные значения на частях снимка с опухолью и без неё [6]. Части, имеющие опухоль (или не имеющие ее) при этом могут сильно отличаться друг от друга по структуре ткани (белое или серое вещество, костная ткань и др.). Строки и столбцы представляют уровни яркости в изображении. К примеру, в данном случае, у нас есть только два уровня яркости: 1 и 2.

Значения в ячейках матрицы указывают на количество раз, когда пиксели с заданными уровнями яркости встречаются друг с другом с заданным сдвигом (горизонтальным) и углом (0 градусов)

Далее для всех пар пикселей вычислено по шесть характеристик:

1. **Контраст (Contrast).** Контраст измеряет локальные изменения в матрице серого уровня соседства, вычисляется как сумма квадратов разницы между интенсивностями соседних пикселей.

2. **Непохожесть (Dissimilarity).** Непохожесть измеряет различия между интенсивностями соседних пикселей. Это просто среднее арифметическое расстояний между интенсивностями.

3. **Гомогенность (Homogeneity).** Гомогенность измеряет близость распределения элементов в GLCM к его диагонали.

4. **Энергия (Energy).** Энергия (или также называется униформностью) – это сумма квадратов элементов GLCM.

5. **Корреляция (Correlation).** Корреляция измеряет степень линейной зависимости между интенсивностями пикселей в изображении.

6. **Угловые моменты второго порядка (ASM).** ASM представляет собой сумму квадратов элементов GLCM и измеряет уровень детализации текстуры.

Поученные признаки классифицируются с помощью таких классических моделей машинного обучения, как : SVM, K-NN, LR, NB, DT, RF.

Сверточная нейронная сеть Resnet-50. Архитектура ResNet-50 (Residual Network с 50-ю слоями) (рис. 1) представляет собой глубокую нейронную сеть, которая была представлена в [7]. Эта архитектура разработана для решения проблемы затухающего градиента при обучении глубоких нейронных сетей. Принцип работы ResNet-50 основан на использовании блоков, называемых "residual blocks" (блоки с остаточным соединением). Они позволяют эффективно обучать очень глубокие сети, предотвращая проблемы с затухающими градиентами. Для классификации опухолей мозга в [8] применяется совместно с технологией Transfer Learning.

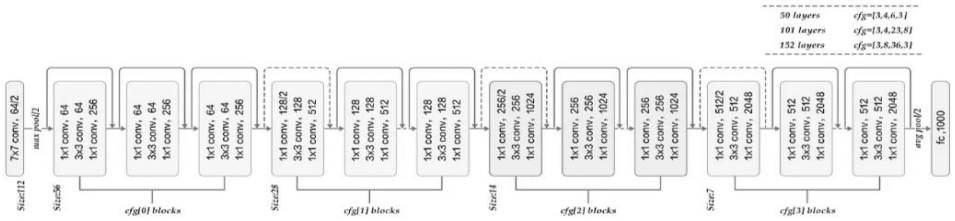


Рис. 1. Архитектура Resnet-50

ResUnet. В архитектуре используется стандартный кодер-декодер, идея которого заключается в плавном уменьшении размера изображения (кодирование) и затем его пошаговом увеличении (декодирование). Скип-коннекции позволяют передавать более низкоуровневые признаки напрямую в декодер, улучшая передачу информации между разными уровнями сети и помогая сохранять детали при восстановлении изображения.

Первые блоки – входная часть кодера (encoder); включает в себя сверточные слои (Conv2D), функции активации (ReLU), нормализацию пакета (BatchNormalization) и слой пулинга (MaxPool2D). Имеется два этих блока, которые постепенно уменьшают размерность данных (по высоте и ширине изображения) и увеличивают количество каналов признаков. Эти блоки обычно используется для захвата более низкоуровневых признаков в изображении, таких как грани, цвета и текстуры.

Далее следует серия блоков resblock, которые представляют собой блоки остаточной свертки (Residual Blocks), добавленные для улучшения процесса обучения и избежания проблемы затухающих градиентов. Эти блоки обычно служат для извлечения более высокоуровневых и абстрактных признаков в изображении. Разделение первых блоков кодера от последующих блоков resblock позволяет лучше контролировать уровень детализации и сложности признаков, извлекаемых на разных этапах кодирования. Это улучшает обучение и позволяет сети выделять как низкоуровневые, так и высокоуровневые признаки в изображении.

После блоков resblock в ResUNet следует декодер (decoder), который состоит из серии блоков, обратных по отношению к кодеру. Декодер постепенно увеличивает размерность данных и восстанавливает пространственное разрешение изображения.

Для каждого блока декодера используются транспонированные сверточные слои (Conv2DTranspose или Conv2D с upsampling), которые увеличивают размерность данных. Это позволяет восстановить пространственное разрешение изображения. После каждого такого слоя следует слой конкатенации (Concatenate), который объединяет выходные данные с соответствующими данными из сквозных соединений (skip connections), переданных из кодера. Это помогает передавать информацию о низкоуровневых признаках напрямую в декодер, улучшая сохранение деталей восстановленного изображения. Кроме того, после слоя конкатенации обычно добавляются блоки resblock в декодере, чтобы улучшить процесс обучения и извлечение признаков.

В конце декодера обычно добавляется выходной слой, который преобразует выходные данные в соответствии с требуемым форматом (например, для сегментации изображений это может быть один или несколько сверточных слоев с функцией активации, такой как сигмоида для двоичной сегментации или softmax для многоклассовой сегментации).

Итак, архитектура ResUNet состоит из кодера, включающего в себя блоки свертки и resblock для извлечения признаков, и декодера, включающего в себя блоки транспонированных сверток, конкатенации и дополнительные resblock для восстановления изображения. Эта архитектура позволяет эффективно решать задачи, такие как сегмен-

тация изображений, сохраняя при этом детали и контекст (рис. 2). Указанная архитектура применяется в работе для сегментации опухолей мозга в [9].

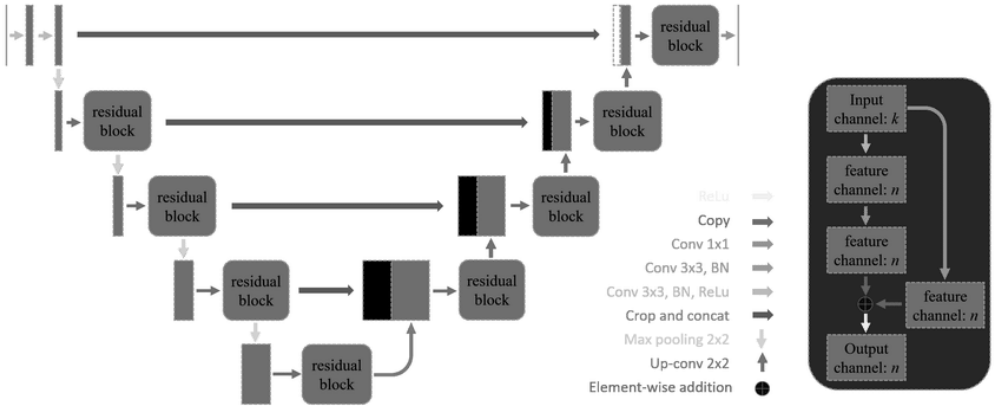


Рис. 2. Архитектура ResUnet

2. Описание алгоритма

Предлагаемая система обнаружения опухолей головного мозга с использованием традиционных алгоритмов машинного обучения состоит из следующих этапов:

1. Сегментация с помощью алгоритма Fuzzy C-means.
2. Извлечение признаков GLCM по 4-м направлениям соотношений пикселей: 0, 45, 90, 135 градусов и шагом 1.
3. Классификация с помощью традиционных классификаторов Logistic Regression\Support Machine vector\ Naive Bayes\ Decision Tree\ Random Forest\ K-nn

В эксперименте 1 изображения конвертируются в уровни серого и масштабируются до размера 256x256 пикселей с использованием библиотеки CV2. Затем применяются фильтры Гаусса для сглаживания возможного шума [10], применяется фильтр резкости для восстановления деталей [11] и подаются на вход модели кластеризации FCM. Для реализации FCM используется библиотека Pyclsustering. На выходе из алгоритма FCM будет также двумерная матрица, но уже с назначенными кластерами, которая покажет, какие пиксели относятся к каждому кластеру. Таким образом, мы выполняем сегментацию с целью убрать линию черепа и мягкие ткани, чтобы избавиться от неинформативных для нашей цели признаков. Затем с использованием матрицы GLCM при помощи библиотек Pandas и Skymage извлекается по шесть характеристик для каждого взаимоотношения четырех выбранных пар пикселей: Contrast, Dissimilarity, Homogeneity, Eneregy, Correlation, ASM, 24 признака на каждое изображение заносятся в Pandas Dataframe, который делится на обучающую и тестовую выборку в соотношении 80% и 20%, и далее подаются на вход классификаторов (SVM, LR,DT, RF, NB, K-NN с использованием библиотеки Sckitlearn), которые уже выдают нужные метки в зависимости от того, есть опухоль или её нет.

В эксперименте 2 набор данных разбивается на обучающий и тестовый в соотношении 85% и 15%. Производится аугментация обучающего набора путем дополнения обработанными изображениями с применением фильтра Гаусса с невысоким значением, для сглаживания малозначимых деталей и маловероятного шума и после фильтра повышения резкости, для повышения резкости и усиления границ. Также применяется случайный поворот по горизонтали, горизонтальный и вертикальный сдвиг, а также shearing со значением 20 градусов (случайные искажения по типу наклона или искривления). Затем получившийся набор передается нейросетевому классификатору на основе сверточной нейронной сети, предобученной на данных Imagenet, архитектуры Resnet 50, взятой с исключением последних слоев и добавлением слоя среднего пулин-

га (усредняем признаки по пространственным измерениям со значением ядра (3,3)), полносвязных слоев, слов нормализации пакета и слоев дропаута. Используем метод Transfer learning – fine tuning, добавляя новые собственные слои классификатора, чтобы адаптировать модель для решения конкретной задачи. Сегментируется изображение с помощью архитектуры ResU-net, реализованной в библиотеке Keras. Данная архитектура включает в себя кодер и декодер на residual – блоках с добавлением скип – коннекций между соответствующими слоями. Для сегментации набор применялся без упомянутой аугментации. К маскам и МРТ-изображениям применили центрирование и нормализацию данных: вычитается среднее значение всех пикселей маски, а затем полученное значение делится на стандартное отклонение. Это позволяет привести значения маски к более стабильной шкале, что часто применяется для улучшения процесса обучения и повышения стабильности модели. В декодере были применены слои Conv2DTranspose с обучаемыми параметрами.

ResUnet: Total params: 2733953, Trainable params: 2729569, Non-trainable params: 4384.

Resnet-50: Total params: 25752450, Trainable params: 25698818, Non-trainable params: 53632.

3. Результаты

Результаты экспериментов показаны в табл. 1–3 и рис. 3–7. Результаты, полученные в **эксперименте 1**, показывают что предложенный в нем подход требует доработки, хотя и была применена кроссвалидация и технология Greadresearch. Следует отметить низкие показатели Specificity у логистической регрессии и K-NN, следовательно, были частые случаи ложно позитивной классификации. Возможно, такой доработкой может быть использование совместно с признаками GLCM и других техник получения признаков, таких как контуры с вычислением геометрических характеристик: длины периметра, площади, соотношения сторон (aspect ratio), ограничивающего прямоугольника (bounding rectangle). Также можно применить признаки матриц GLRM (Gray-Level Run-Length Matrix) и HOG (Histogram of Oriented Gradients). GLRM представляет собой матрицу, которая отражает количество серий (run-length) пикселей с одинаковыми уровнями яркости в определенном направлении на изображении. HOG представляет собой метод, который вычисляет гистограмму направленных градиентов в каждом блоке изображения. Возможно комбинирование полученных по ним и другими методами признаков с предварительным применением фильтров изображений и без них. Наибольшая достигнутая точность – 84.31 с использованием Random Forest с применением фильтрации с подобранным значением ядра Гаусса 11. Результаты у K-nn выше были с большим значением ядра – 13. Фильтр резкости был постоянным, соответственно, матрица 3*3: [–1,–1,–1], [–1,9,–1], [–1,–1,–1].

В **эксперименте 2** технология Transfer learning на базе нейросети Resnet-50, предобученной на наборе данных Imagenet, показала себя значительно лучше. Были заменены слои top на настроенные под задачу бинарной классификации, и полученные метрики оказались от 97.79% и выше. График функции потерь показал незначительные колебания, тогда как метрика Accuracy имела большие колебания, но оба графика были близки к сходимости, переобучения не было. Колебания могут быть вызваны разнообразием форм опухолей; с одной стороны, это может помочь модели лучше обучиться на разнообразных сценариях и лучше обобщить свои знания, с другой стороны, может усложнить обучение из-за большего разнообразия объектов и различий между ними. Указанные осцилляции могут быть вызваны недостаточной регуляризацией – дропаутами, недостаточным количеством экспериментов со скоростью обучения Learning rate, выбором оптимизатора и других параметров. Семантическая сегментация ResUnet показала также хороший результат по метрикам от 94.17%, графики ее и функции потерь получены весьма стабильные, близкие к сходимости. Результаты сегментации отобра-

жают область опухоли, близкую к оригинальной маске. В дальнейших работах планируется получить более высокие результаты за счет использования набора данных большего объема и аккуратной работой с сохранением соотношения объектов и фона при аугментации, выбора метода предобработки, с учетом сохранения важных деталей и контекста изображений, а также использование длинных соединений для передачи информации от более глубоких слоев к более поверхностным слоям, что позволяет сохранять и использовать более детальную информацию о контексте изображения на различных уровнях абстракции и может улучшить качество сегментации объектов на изображении [12]

Таблица 1

Результаты эксперимента 1 – классификации классическими моделями

Классификатор	Accuracy	Recall	Precision	Specificity	Jaccard_score
LR	68.63%	96.0%	60.97 %	53,85%	64.86%
SVM	78.43%	92.0%	71.88%	65.38%	67.64%
Random Forest	84.31%	100.0%	5.75 %	69.23%	86.20 %
Naive Bayes	76.47%	88.0%	70.96%	65.38%	64.70%
Decision Tree	78.43 %	92.0 %	71.86 %	65.38%	67.64%
K-nn	76.47%	100.0%	67.56%	53.84%	80.66%

Таблица 2

Результаты эксперимента 2 – классификации нейросетевым классификатором

Классификатор	Accuracy	Recall	Precision	f1	Specificity
Resnet50	97.79%	98.0%	98.0%	98.0%	98.25%

Таблица 3

Результаты эксперимента 2 – семантическая сегментация ResUnet

Семантическая сегментация	Tversky	Focal Tversky	Precision	Recall	f1	IOU	Dice coef.
ResUnet	94.17%	11.75 %	100%	95.1%	95. 08%	90.62%	95.08%

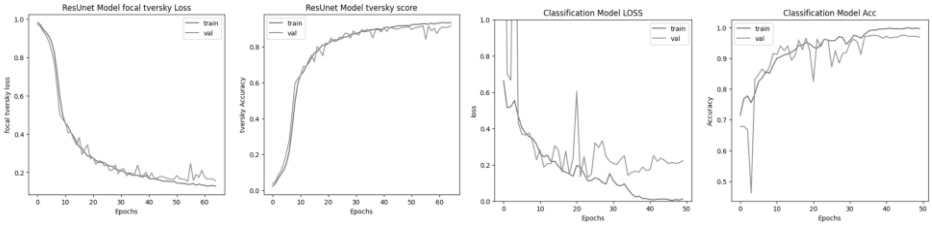


Рис. 3. График потерь categorical crossentropy и метрики accuracy Resnet-50 слева и график потерь Focal Tversky и метрики Tversky ResUnet справа

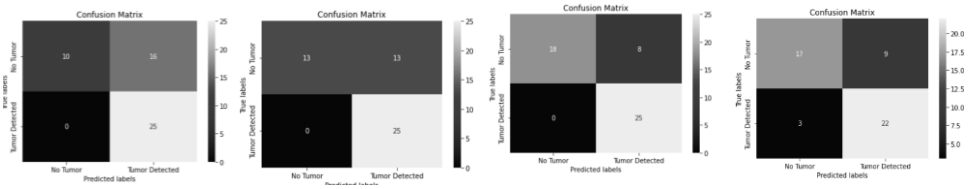


Рис. 4. Confusion matrix для LR, SVM, RF, NB

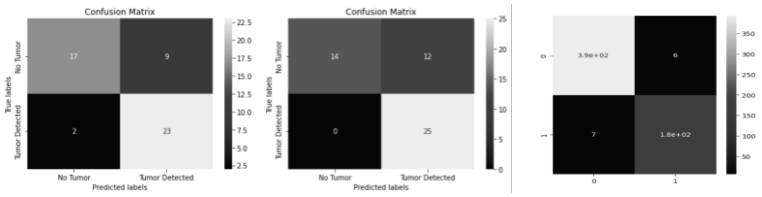


Рис 5. Confusion matrix для DT, K-NN, Resnet-50

Далее сравним лучшие результаты с другими работами. Результаты классификации взяты из [13] и [14].

Таблица 4

Сравнениие с результатами классификации других работ посвященных классификации опухолей мозга

Авторы	Название модели	Точность
Wo`zniak et al. (2021)	CNN with classic architecture	95.09
Ayadi et al. (2021)	CNN	94.74
Ghassemi et al. (2020)	GAN+CNN	95.6
Badža and Barjaktarović (2020)	CNN	96.56
Swati et al. (2019)	AlexNet, VGG16, VGG19	94.82
Sultan et al. (2019)	CNN	96.13
Pashaei, Sajedi, and Jazayeri (2018)	CNN	93.68
Ismael and Abdel-Qader (2018)	Neural Network	91.9
Afshar, Mohammadi, and Plataniotis (2018)	Capsule Network	86.56
Atika Akter (2024) [9]	Deep CNN	96.7
Resnet-50	Resnet- 50	97.79

Таблица 5

Сравнениие с результатами других работ посвященных сегментации опухолей мозга

Работа	Dice coeffecient
W. Chen et al., 2019	83.90%
Aboelenein et al., 2020	86.5%
Ben naceur et al., 2020	86.2%
Zhou et al., 2020	86.88%
Sun et al., 2021	90%
Nanda et al., 2023	93.0%
Z.Zhu et al. 2023	86.93%
Y.wang et al. 2023	83.24%
Sangeet Kumar et.al (brats2018) [14]	94.80%
Sangeet Kumar et.al (brats2021) [14]	95.01%
ResUnet	95,08%

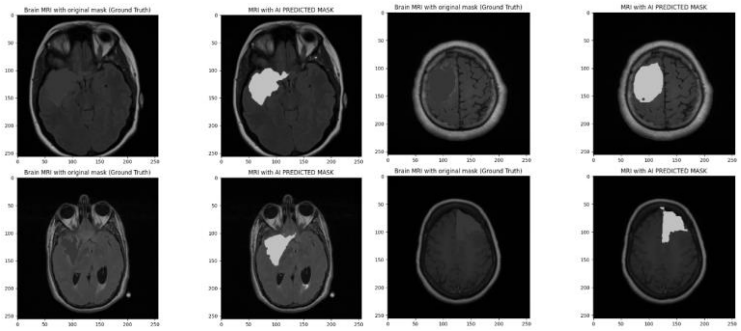


Рис. 6. Результаты применения сегментации к оригиналу
в сравнении с оригинальной маской, наложенной на оригинал

Заключение

В ходе проделанной работы были получены результаты классификации, включающие в себя 6 классических моделей и одну нейросетевую на базе Resnet-50. Решение задачи нейросетевой сегментации было произведена на основе применения ResUnet. Проведено сопоставление результатов проведенных численных экспериментов с известными аналогами. Сопоставление показало, что исследованный в данной работе нейросетевой классификатор попадает на первое место из рассмотренных аналогов, нейросетевая сегментация на базе ResUnet отличается от первого места незначительно, их показатели на современном уровне.

ЛИТЕРАТУРА

1. *Далечина А.В., Беляев М.Г., Тюрина А.Н., Золотова С.В., Пронин И.Н., Голанов А.В.* АО "Деловой центр нейрохирургии" (Центр "Гамма-нож"), Москва, Россия Сколковский институт науки и технологий, Москва, Россия ЗНМИЦ нейрохирургии им. акад. Н.Н. Бурденко, Москва, Россия © Коллектив авторов, 2019.

2. *Dunn J.C.* A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters // Journal of Cybernetics. – 1973. – 3.– P. 32–57.

3. *Bezdek J.C.* Pattern Recognition with Fuzzy Objective Function Algoritms. – Plenum Press, New York, 1981.

4. *Deng W., Xiao W., Deng H, Liu J.* MRI Brain Tumor Segmentation With Region Growing Method Based On The Gradients And Variances Along And Inside Of The Boundary Curve // 3rd International Conference on Biomedical Engineering and Informatics , 2010.

5. *Тымчук А.И.* О выборе уровней серого в задаче текстурной сегментации изображений на основе матриц яркостной зависимости // Кибернетика и программирование. – 2018. – №. 3. – С. 1–9.

6. *Sompong C., Wongthanavasu S.* MRI brain tumor segmentation using GLCM cellular automata-based texture feature //2014 International Computer Science and Engineering Conference (ICSEC). – IEEE, 2014. – P. 192–197.

7. *He K. et al.* Deep residual learning for image recognition // Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – P. 770–778.

8. *Balaji G., Sen R., Kirty H.* Detection and Classification of Brain tumors Using Deep Convolutional Neural Networks // arXiv preprint arXiv:2208.13264. – 2022.

9. *Kumar P.S. et al.* Brain tumor segmentation of the FLAIR MRI images using novel ResUnet // Biomedical Signal Processing and Control. – 2023. – Т. 82. – С. 104.

10. *Li Q. et al.* Medical image classification with convolutional neural network // 13th international conference on control automation robotics & vision (ICARCV). – IEEE, 2014. – P. 844–848.

11. *Balaji G., Sen R., Kirty H.* Detection and Classification of Brain tumors Using Deep Convolutional Neural Networks // arXiv preprint arXiv:2208.13264. – 2022.

12. *Pradeep K.R. et al.* Improved machine learning method for intracranial tumor detection with accelerated particle swarm optimization // Journal of Healthcare Engineering. – 2022.

13. *Akter A. et al.* Robust clinical applicable CNN and U-Net based algorithm for MRI classification and segmentation for brain tumor // Expert Systems with Applications. – 2024. – V. 238. – P. 122.

14. *Kumar S., Biswal B.* MAEU-NET: A novel supervised architecture for brain tumor segmentation // International Journal of Imaging Systems and Technology. – 2024. – V. 34. – №. 2. – P. e22988.

ОБУЧЕНИЕ АГЕНТОВ В ВИРТУАЛЬНОЙ СРЕДЕ KUKADIVERSEOBJECTENV

Залогин Н.Е.

Томский политехнический университет

zalogin.nikita@mail.ru

Введение

Обучение с подкреплением (Reinforcement Learning, RL) стало мощным инструментом для обучения интеллектуальных агентов выполнению различных задач в сложных средах. В последние годы все больше растет интерес к применению методов RL в робототехнике с целью обеспечения автономности и адаптивности к окружающей среде.

Целью работы является реализация и сравнение алгоритмов Proximal Policy Optimization (PPO) [1], а также трех модификаций PPO – PPOv1, PPOv2, PPOv3. Алгоритмы реализованы и протестированы с использованием среды KukaDiverseObjectEnv [2] на физическом движке PyBullet [3]. Эти алгоритмы широко известны и демонстрируют некоторую эффективность в различных областях, но все

еще требуют дальнейшего исследования и сравнения для повышения их производительности и применимости в сложных средах робототехники.

Также проведен анализ результатов работы каждого алгоритма на основе метрик, таких как скорость обучения, количество взаимодействий со средой и процент успешного выполнения задачи. Это позволит сравнить эффективность и применимость каждого алгоритма в среде KukaDiverseObjectEnv. Полученные результаты будут полезны для оптимизации алгоритмов обучения с подкреплением в сложных средах робототехники.

1. Виртуальная среда обучения

KukaDiverseObjectEnv – виртуальная среда симуляции на платформе PyBullet, разработанная для обучения агентов манипулированию объектами. Пространство действий в среде имеет 3 параметра – по одному действию для перемещения по каждой из осей *x* и *y* размерностью от –1.0 до 1.0, а также одно действие управления захватом. При каждом шаге в среде манипулятор автоматически опускается. Т.о., агенту необходимо захватить и поднять любой объект из контейнера. В данной задаче награда является бинарной и выдается, только если один из объектов находится выше заданной высоты к концу эпизода. Входными данными, предоставляемыми агенту, являются трехканальные изображения состояния среды размерностью (48,48,3), пример такого изображения приведен на рис. 1.

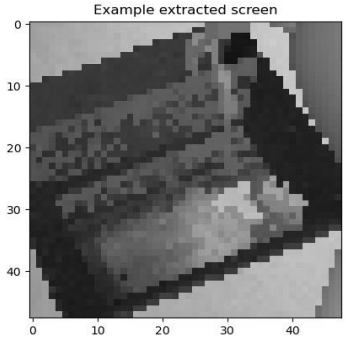


Рис. 1. Пример изображения с виртуальной камеры робота

Благодаря редкой двоичной награде, графическим входным данным и сложности задачи в целом, среда становится достаточно интересной для исследования.

Для тестирования и оценки алгоритмов обучения агентов был установлен критерий ранней остановки – достижение средней награды в 50 за 100 эпизодов, что эквивалентно 50% точности.

2. Описание алгоритмов

Алгоритм Proximal Policy Optimization (PPO) основан на политике (on-policy), и использует actor-critic архитектуру. PPO применяет две ключевые идеи для обеспечения стабильности и эффективности обучения. Первая идея – это обновление политики на основе отношения вероятностей между старой и новой политиками, чтобы ограничить ее изменение. Отношение политик вычисляется ([4]) по формуле

$$r(\theta) = \frac{\pi_\theta(a \mid s)}{\pi_{\theta_{old}}(a \mid s)},$$

где π – политика, θ – параметры политики, a – действие в среде, s – состояние среды, $\pi(a|s)$ определяет вероятность выбора действия a в состоянии s. Это означает, что ее обновление происходит малыми шагами и избегает значительных изменений, которые могут приводить к нестабильности обучения. Вторая идея – это использование значе-

ний преимуществ (advantages) для выравнивания обновлений политики, в противовес абсолютным вероятностям. Это способствует более сбалансированному и стабильному обучению.

Без ограничения на расстояния между новой и старой политиками максимизация целевой функции привела бы к нестабильности с чрезвычайно большими обновлениями параметров и большими коэффициентами политики. PPO накладывает ограничение, заставляя оставаться в пределах небольшого интервала. Вычисляется целевая функция с этими изменениями ([4]) по формуле

$$J^{\text{CLIP}}(\theta) = \mathrm{E}\left(\min(r(\theta)\hat{A}_{\theta_{\text{old}}}(s,a), \text{clip}\left(r(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{\theta_{\text{old}}}(s,a)\right),$$

где $A(s,a)$ – функция преимуществ, ε – гиперпараметр.

В дополнение к ограниченному вознаграждению целевая функция дополняется членом ошибки при оценке значения и энтропией, чтобы стимулировать достаточное исследование среды. В итоге целевая функция ([4]) будет вычисляться по формуле

$$J^{\text{CLIP}}(\theta) = \mathrm{E}\left[J^{\text{CLIP}}(\theta) - c_1\left(V_0(s) - V_{\text{target}}\right)^2 + c_2 H\left(s, \pi_\theta(.)\right)\right],$$

где c_1 и c_2 – константные гиперпараметры, $V(s)$ – ожидаемый возврат состояния s.

Топология нейронной сети PPO представлена на рис. 2.

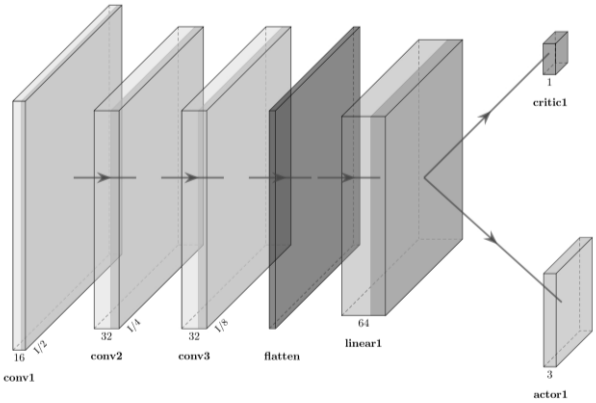


Рис. 2. Топология нейронной сети агента PPO

PPOv1 представляет собой первую модификацию PPO, специально адаптированную для взаимодействия со средой KukaDiverseObjectEnv. В данной модификации, помимо структурных изменений в обучающем цикле, произведены изменения в пространствах наблюдений и действий. Входные данные увеличены с (48, 48,3) до (84,84,3), а количество доступных действий увеличено до пяти: по два действия для перемещения по каждой из осей x и y, а также одно действие для управления захватом. Изменения также затронули архитектуру нейронной сети. Дополнительные скрытые слои были добавлены для сетей actor-critic. Кроме того, в сверточных слоях изменены размеры ядра и шаг, а также применена пакетная нормализация (BatchNorm2d). В качестве метода инициализации весов нейронной сети использован метод равномерного распределения Ксавье (xavier-uniform). Данные изменения должны стабилизировать и ускорить обучение агента. Топология нейронной сети PPOv1 представлена на рис. 3.

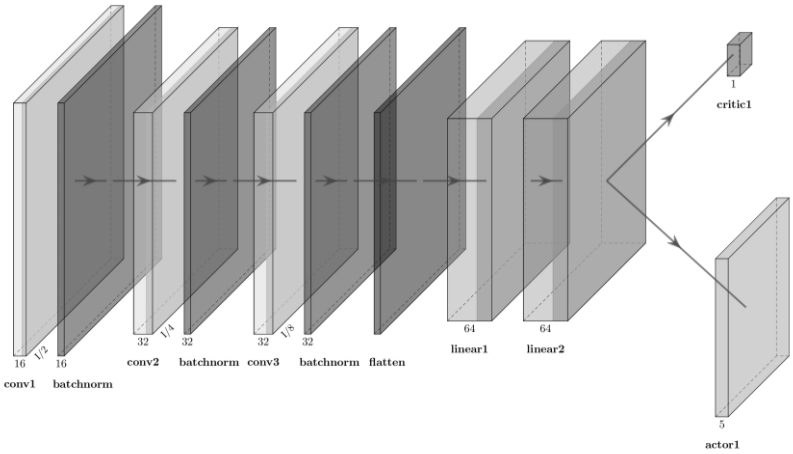


Рис. 3. Топология нейронной сети агента PPOv1

PPOv2, вносит дополнительные изменения к PPOv1, включая синхронное параллельное выполнение действий в среде. Этот подход существенно ускоряет процесс сбора траекторий действий в среде и, следовательно, улучшает производительность обучения агента в целом. Реализация параллельных сред выполнена с использованием библиотеки "multiprocessing", при этом число параллельных сред составляет 20, что соответствует количеству потоков процессора на рабочей станции.

PPOv3 продолжает развитие PPOv2. В этой версии алгоритма были оптимизированы многие гиперпараметры благодаря использованию модульной системы обучающего цикла. В частности, главными изменениями стали уменьшение размера пакета данных и увеличение скорости обучения (Learning rate), что позволило сократить длительной эпизодов обучения и ускорить процесс обновления весов модели, что также ускорило обучение. Важным изменением является увеличение глубины слоев с большим количеством нейронов для сверточные сети, включая общие слои (shared layers) и слои actor-critic. Топология нейронной сети PPOv3 представлена на рис. 4.

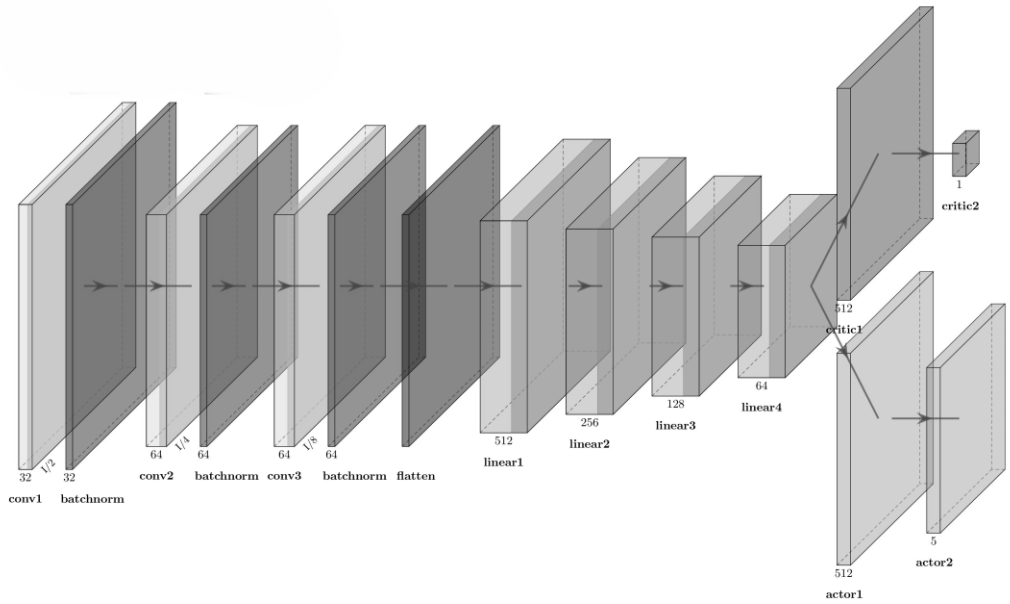


Рис. 4. Топология нейронной сети агента PPOv3

3. Результаты

По ходу обучения агентов были проведены измерения средней награды за 100 эпизодов, результаты которых представлены на рис. 6,7.

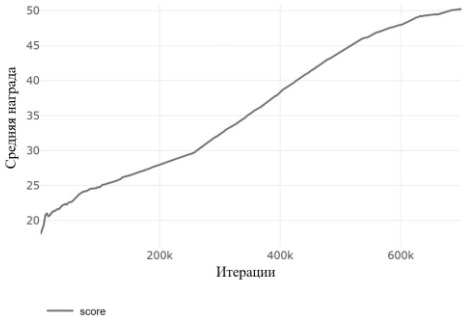


Рис. 6. Средняя награда агента PPO

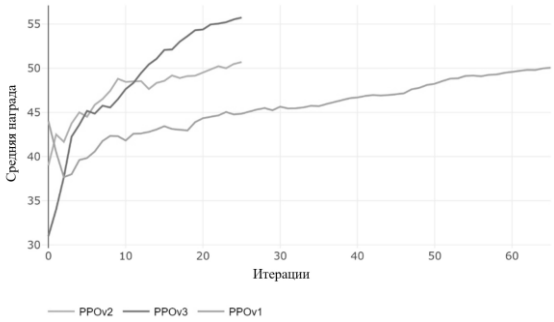


Рис. 7. Средняя награда агентов PPOv1, PPOv2, PPOv3

В дополнение к графикам средней награды, были зафиксированы показатели конечной средней награды, количество шагов обучения и время обучения, приведенные в табл. 1.

Таблица 1

Сравнение метрик обучения всех полученных агентов

Агент	Средняя награда	Шаги обучения	Время обучения
PPO	50.202	696320	5:12:04.500
PPOv1	50.690	65 (66560)	1:52:07.842
PPOv2	50.061	26 (26624)	0:35:12.836
PPOv3	50.067	13 (13312)	0:09:34.629

Из результатов, представленных на рис. 6,7 и табл. 1, можно сделать вывод о том, что стандартный алгоритм PPO достигает поставленной цели, но для этого требуется значительное количество времени. Последующие три модификации смогли значительно сократить время обучения агента.

Для тестирования полученных агентов использовалось по 1000 эпизодов среды в тестовом режиме. Результаты тестирования приведены в табл. 2.

Таблица 2

Тестирование обученных агентов

Агент	PPO	PPOv1	PPOv2	PPOv3
True episode	49.7%	50.2%	49.1%	52.4%
False episode	50.3%	49.8%	50.9%	47.6%

Из табл. 2 можно сделать вывод, что все агенты решают поставленную задачу с заданной точностью в окрестности 50% за 1000 тестовых эпизодов.

Заключение

Стандартный алгоритм PPO показал хорошие результаты и продемонстрировал способность решать задачу с поставленной точностью, но он имеет значительные проблемы с вычислительной сложностью. За счет изменения нейросети и дополнительных оптимизаций в PPOv1, время обучения для рассматриваемой задачи было сильно сокращено. Благодаря параллелизации сред в PPOv2, удалось еще сильнее сократить время обучения. Модификация PPOv3 также сокращает время обучения. Все реализованные агенты решают поставленную задачу, достигая точности в районе 50% за 1000 тестовых эпизодов.

В будущем предполагается дальнейшее улучшение алгоритма PPO и общего процесса обучения для достижения лучших результатов и решения проблемы переобучения. Апробация алгоритмов LSTM [5] и xLSTM [6] для данной задачи. Замена среды обучения на уже существующие варианты или разработка авторской системы симуляции среды может открыть новые направления исследований в области обучения с подкреплением.

ЛИТЕРАТУРА

1. *Schulman J. et al*. Proximal policy optimization algorithms // arXiv preprint arXiv:1707.06347. – 2017.

2. bulletphysics/bullet3 bullet/kuka_diverse_object_gym_env.py – Текст : электронный // bulletphysics – 2020. – URL: https://github.com/bulletphysics/bullet3/blob/master/examples/pybullet/gym/pybullet_envs/bullet/kuka_diverse_object_ gym_env.py

3. PyBullet // PyBullet – 2024. – URL: https://pybullet.org/wordpress/

4. *Weng L.* Policy Gradient Algorithms // Lil'Log – 2018. – URL: https://lilianweng.github.io/posts/2018-04-08-policy-gradient/#ppo

5. *Hochreiter S., Schmidhuber J.* Long short-term memory // Neural computation. – 1997. – V. 9. – №. 8. – P. 1735–1780.

6. *Beck M. et al.* xLSTM: Extended Long Short-Term Memory // arXiv preprint arXiv:2405.04517. – 2024.

МОДЕЛЬ ГРАФОВОЙ НЕЙРОННОЙ СЕТИ С МЕХАНИЗМОМ ВНИМАНИЯ ДЛЯ ЗАДАЧИ СЕМАНТИЧЕСКОЙ СЕГМЕНТАЦИИ ОБЛАКОВ ТОЧЕК

Кадыров Р.А.
Томский политехнический университет
rak32@tpu.ru

Введение

Современные технологии в области компьютерного зрения и машинного обучения становятся ключевыми элементами в решении сложных задач визуального анализа сцен, таких как классификация объектов на сцене и сегментация отдельных объектов сцены.

Облака точек представляют собой многомерный источник информации, получаемый с лазерных датчиков, таких как LIDAR [1], позволяющий расширить возможности для восприятия окружающей среды при помощи компьютерного зрения.

Алгоритмы обработки облаков точек находят применение в различных областях. Например, в архитектуре и градостроительстве [2]они используются для создания точных 3D-моделей зданий и ландшафтов. С увеличением получаемого объема данных возникает потребность в эффективном и быстром анализе.

Семантическая сегментация представляет собой актуальное направление в исследованиях компьютерного зрения. В контексте уличных сцен, где разнообразие объектов и их взаимодействие являются сложными задачами для традиционных методов, применение машинного обучения становится неотъемлемым инструментом для достижения точных и эффективных результатов.

Одним из способов обработки объектов, представленных в виде облаков точек, является их преобразование в графовую структуру. Активное применение моделей машинного обучения для обработки графовых структур ведется с 2009 г. [3–6]. Появились модели, взаимодействующие с облаками точек как с графами, позволяя использовать расстояние между отдельными точками как важный признак [7,8], что подтверждает актуальность разработки эффективного способа обработки облаков точек на основе графовых структур.

Целью данной работы является развитие идеи применения графовых нейронных сетей для сегментации облаков точек, в частности, улучшение современных моделей при помощи адаптированного механизма внимания.

1. Постановка задачи и выбор моделей

С развитием моделей, основанных на сверточных слоях [9] появился большой интерес в том, чтобы адаптировать эти методы для геометрических данных. Однако геометрические данные, в частности, облака точек, имеют сильно отличающуюся от изображений структуру, поэтому первые работы в этой области предпринимали попытки как-то структурировать геометрические данные [10]. Впрочем, с развитием глубоких нейронных сетей была разработана модель PointNet [11], открывшая область геометрических глубоких нейронных сетей.

Для задачи сегментации и классификации облаков точек существует модель, которая способна осуществлять преобразование облаков точек в граф для осуществления обобщения информации на основе графовой свертки – Dynamic Graph Convolutional Network (DGCNN) [7].

В этой работе рассматривается развитие этой модели – Linked Dynamic Graph Convolutional Network (LDGCNN) [8]. Архитектура модели представлена на рис. 1.

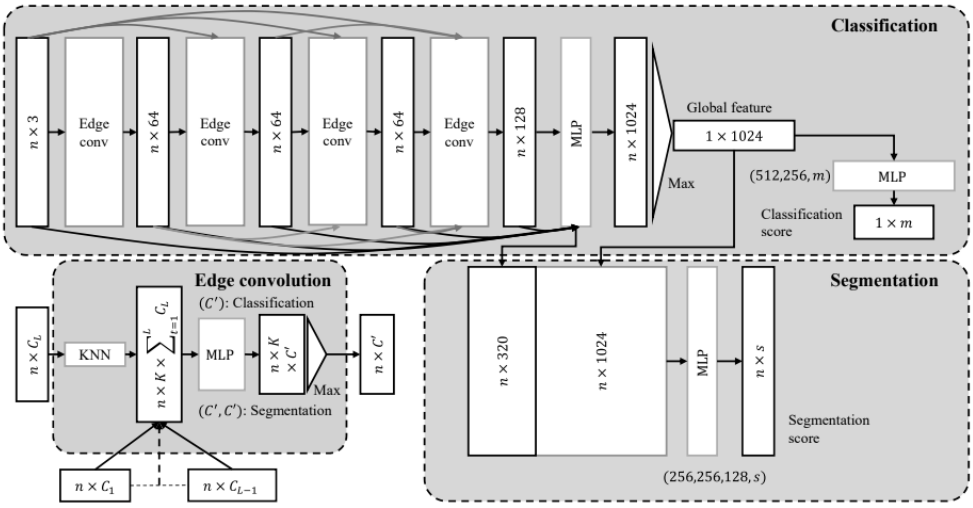


Рис. 1. Архитектура модели LDGCNN

Отличительными особенностями этих моделей является динамическое построение графов. В свою очередь, в модели LDGCNN [8] признаки с разных слоев дополнительно передаются на более глубокие слои. Главной особенностью этих моделей является слой "EdgeConv", принцип его работы состоит в следующем:

1. Для каждой точки строится граф путем соединения ближайших точек и признаков с других слоев при помощи метода *k*-ближайших соседей. Все ноды также имеют направленные грани, указывающие на эти ноды (self-loop edges).
2. Далее для каждой построенной грани высчитываются признаки (путем применения нелинейной функции с обучаемыми параметрами).
3. Применяется "свертка по граням". Эта операция агрегирует признаки с прилегающих граней путем сложения или взятия максимума и обновляет признаковое пространство вершины.

Благодаря обновлению признакового пространства каждой точки, с каждым последующим слоем ближайшими соседями к каждой точке оказываются наиболее "осмысленные" точки, т.е. более подходящие структурно.

Рассчитанные признаки далее могут использоваться как для предсказания класса облака точек, так и для сегментации облака точек на разные классы.

В свою очередь, механизм внимания, впервые примененный в графовых моделях, был представлен в работе авторов Petar Veličković et al., представивших модель Graph

Attention Network (GAT) [12]. Схема работы механизма внимания в графовых моделях может быть представлена в виде следующего алгоритма:

1. На входе имеются наборы признаков всех вершин $h = \left\{\vec{h}_1, \vec{h}_1, \ldots, \vec{h}_N\right\}$.

2. Далее идет обучаемое линейное преобразование в виде однослойного персептрона без функции активации, представленного в виде матрицы весов *W*. Признаки вершин умножаются на эту весовую матрицу.

3. Получившиеся представления конкатенируются и перемножаются с вектором весов *a* для получения коэффициентов внимания $e_{ij} = a\left(W\vec{h}_i, W\vec{h}_j\right)$.

Вектор весов *a* представляет собой однослойный персептрон с функцией активации LeakyReLU на выходе.

4. Для сравнения показателей внимания между вершинами полученные коэффициенты нужно нормализовать, и, чтобы определить, какой узел важнее для какого узла, требуется привести коэффициенты к одинаковому масштабу. Для этого на коэффициенты внимания применяется функция активации Softmax.

Для улучшения результатов применяется не один такой механизм, а множественные "головы внимания". Другими словами, совокупность "механизмов внимания" с разными параметрами, работающие параллельно [13]. Такой механизм позволяет распределить внимание на разных признаках, при этом каждая отдельная "голова внимания" фокусируется на чем-то своем, и, тем самым, расширяется захватываемый моделью контекст признаков.

2. Архитектура разработанной модели

Основное нововведение разработанной модели состоит в замене обычной графовой свертки в модели LDGCNN [8] на графовую свертку с использованием "механизма внимания" (рис. 2), описанную выше. Т.о., это должно улучшить обобщающую способность модели за счет возможности выделять наиболее важные признаки. Механизм работы сверточного слоя с "механизмом внимания" "Attention EdgeConv" – нижний блок на рис. 2.

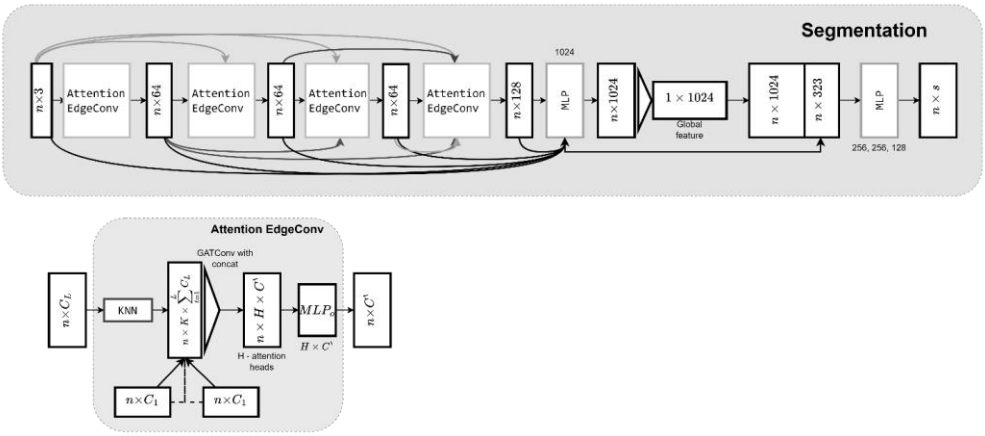


Рис. 2. Архитектура модели LDGCNN с использованием механизма внимания

Также, используя алгоритм К-ближайших соседей, из облака точек формируется граф. Выходом графовой свертки с механизмом внимания является количество выходных признаков, умноженное на количество "голов внимания", используемых в механизме внимания. Для агрегации результатов с каждой "головы внимания" используется однослойный персептрон. Т.о., на выходе такой свертки будет аналогичный со вход-

ным набор данных с обновленным набором признаков для каждой точки из облака точек.

3. Выбор набора данных

Для апробации разработанной модели в задаче сегментации был выбран набор данных "LARGE-SCALE 3D SHAPE RECONSTRUCTION AND SEGMENTATION FROM SHAPENET CORE55" [14], являющийся сокращенным вариантом набора данных – "ShapeNetCore" [15]. Он содержит облака точек различных объектов с разметкой в виде сегментации отдельных частей объектов.

Набор данных содержит 16881 облаков точек как с 16-ю глобальными категориями (самолет, стул, стол и т.п.), так и от 2-х до 6-ти категорий частей объектов для задачи сегментации. Разделение производилось на обучающую и валидационную выборки размерами 14007 и 2874, соответственно. Примеры объектов из набора данных приведены на рис. 3.

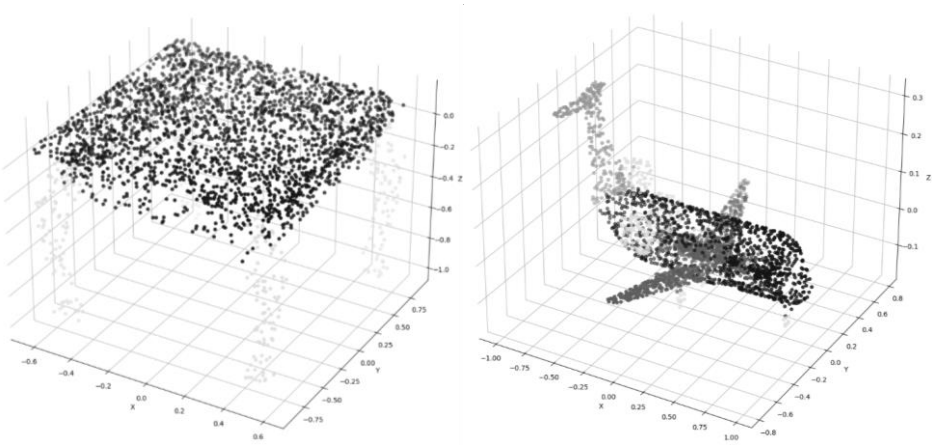


Рис. 3. Примеры облаков точек из использованного набора данных (правый – стол, левый – самолет)

4. Описание измеряемых метрик

Для оценки результатов использовались описанные ниже метрики.

Точность – количество точек, правильно отнесенных к классу, к количеству точек этого класса.

Метрика "Intersection over Union", также известная как Jaccard index, — число от 0 до 1, показывающее, насколько у двух объектов (эталонного и измеряемого) совпадает внутренний "объем" [16].

Формально, для двух непустых множеств A и B, функция IoU определяется как $\mathrm{IoU}(A,B)=\frac{|A\cap B|}{|A\cup B|}$. В качестве функции потерь использовалась "Negative Log-Likelihood Loss" (NLL loss). Она используется для оценки различий между распределением вероятностей, предсказанным моделью, и фактическими метками данных.

В ее основе – функция логарифмического правдоподобия (Log-Likelihood), она вычисляет логарифм вероятности предсказанного класса для каждого примера данных. Например, для бинарной классификации такая функция будет выглядеть как $LL=\sum_{i=1}^{N} y_i \log(p_i)+(1-y_i)\log(1-p_i)$, где N – количество примеров данных, y_i – фактическая метка класса для i-го примера, p_i – предсказанная вероятность принадлежности i-го примера к классу.

Соответственно, т.к. это функция правдоподобия, то она должна максимизироваться для достижения требуемого результата. Но если взять отрицательный логарифм, то потребуется минимизировать функцию, и получится $NLL = -\sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$.

5. Результаты тестирования

Для сравнения тестировалась реализация модели LDGCNN и ее модифицированная версия с использованием внимания, реализованные средствами PyTorch Geometric [17]. Конфигурации процесса обучения приведены в табл. 1.

Таблица 1

Конфигурации моделей

	LDGCNN	**LDGCNN с использованием механизма внимания**
Количество эпох	50	50
Размер батча	10	10
Тип функции оптимизации	Adam	Adam
Скорость обучения	0.0001	0.0001
Тип функции потерь	negative log likelihood loss	negative log likelihood loss
Кол-во соседних точек для объединения в граф	20	20
Функция агрегации	Max	–
Кол-во голов внимания	–	4

Т.о., были получены следующие результаты (таб. 2 и рис. 4):

Таблица 2

Результаты обучения

Название метрики	**LDGCNN**	**LDGCNN с использованием механизма внимания**
Точность	0.853	0.797
IoU	0.678	0.636
Значение функции потерь	0.548	0.638

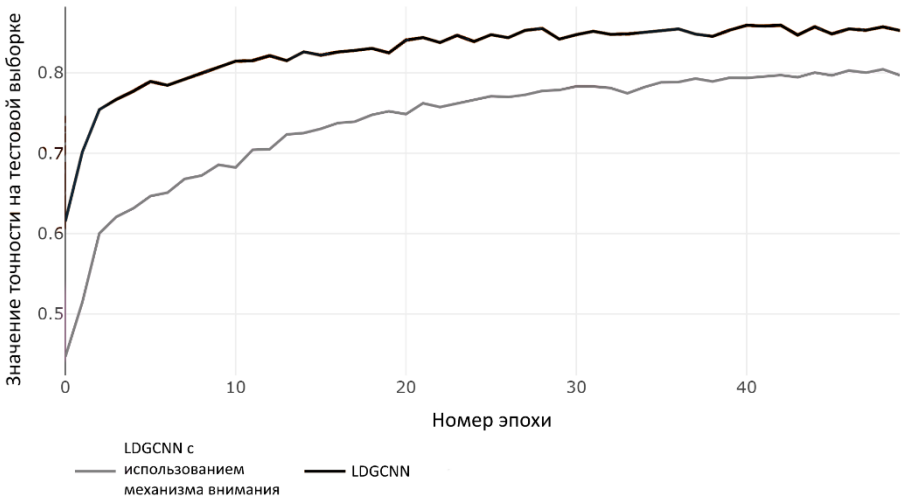


Рис. 4. График изменения точности обученных моделей от эпохи обучения

По результатам можно увидеть, что разработанная модель достаточно близка по основным метрикам к использованной в качестве базы модели LDGCNN, что, в целом, говорит о работоспособности и потенциале модели. Представленные результаты являются промежуточными, далее планируется предпринять шаги по улучшению метрик и скорости работы модели, а также исследовать влияние аугментации данных на качество предсказания.

Заключение

В результате работы была предпринята попытка внедрения механизма внимания в модель сегментации облаков точек. Полученная модель была обучена на подготовленном наборе данных, было произведено сравнение основных метрик с моделью, взятой в качестве основы. Т.о., удалось достичь близких метрик, и следующими шагами планируется улучшить результаты работы модели для повышения метрик.

ЛИТЕРАТУРА

1. Лидар // Википедия: сайт. – URL: https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%B4%D0%B0%D1%80 (дата обращения: 23.03.2024)
2. Облако точек. Как мы развиваем цифровые технологии в строительстве // Habr: сайт. – 2019. – URL: https://habr.com/ru/companies/jetinfosystems/articles/464593/ (дата обращения: 23.03.2024)
3. *Franco S.* The Graph Neural Network Model // IEEE Transactions on Neural Networks. – 2009. – V. 20. – №. 1. – P. 61–80.
4. *Kipf T.N., Welling M.* Semi-Supervised Classification with Graph Convolutional Networks. // ArXiv. – 2016. – 1609.02907v4
5. Графовые нейронные сети // ИТМО: сайт. – 2022. – URL: https://neerc.ifmo.ru/wiki/index.php?title=Графовые_нейронные_сети (дата обращения: 23.03.2024)
6. Графовые нейронные сети // Школа Анализа данных: сайт. – URL: https://education.yandex.ru/handbook/ml/article/grafovye-nejronnye-seti (дата обращения: 23.03.2024)
7. *Wang Y., Sun Y., Liu Z., Sarma S.E., Bronstein M.M., Solomon J.* Dynamic Graph CNN for Learning on Point Clouds // ArXiv. – 2019. – 1801.07829
8. *Zhang K., Hao M., Wang J., de Silva C.W., Fu C.* Linked Dynamic Graph CNN: Learning on Point Cloud via Linking Hierarchical Features // ArXiv. – 2019. –1904.10014;
9. *Krizhevsky A.* ImageNet Classification with Deep Convolutional Neural Networks // Communications of the ACM. – 2017. – V. 60. – № 6. – P. 84–90.
10. *Qi C.R., Liu W., Wu Ch., Su H., Guibas L.J.* Frustum PointNets for 3D Object Detection from RGB-D Data // ArXiv. – 2018. – 1711.08488;
11. *Qi C.R., Su H., Mo K., Guibas L.J.* PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation // ArXiv. – 2017. – 1612.00593
12. *Velickovi˘c P., Cucurul G., Casanova A., Romero A., Lio P., Bengio Y.* Graph Attention Networks // ArXiv. – 2018. – 1710.10903
13. *Ashish V., Noam S., Niki P., Jakob U., Llion J., Aidan N.G., Lukasz K., Illia P.* Attention Is All You Need // ArXiv. – 2017. – 1706.03762
14. Large-Scale 3D Shape Reconstruction and Segmentation from Shapenet Core55 // ShapeNet: сайт. – 2017. – URL: https://shapenet.cs.stanford.edu/iccv17/ (дата обращения: 23.03.2024)
15. ShapeNetCore // Papers With Code: сайт. – URL: https://paperswithcode.com/dataset/shapenetcore (дата обращения: 23.03.2024)
16. Intersection over Union (IoU) for object detection // pyimagesearch: сайт. – 2016. URL: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (дата обращения: 23.03.2024)
17. PyG Documentation // Pytorch Geometric: сайт. – URL: https://pytorch-geometric.readthedocs.io/en/latest/ (дата обращения: 23.03.2024)

ГЕНЕРАЦИЯ МЕДИЦИНСКОГО ЗАКЛЮЧЕНИЯ ДЛЯ РЕНТГЕНОВСКИХ СНИМКОВ ГРУДНОЙ КЛЕТКИ ПРИ ПОМОЩИ НЕЙРОННЫХ СЕТЕЙ

Пан А.Э.
Томский государственный университет
tolikpan0403@gmail.com

Введение

В настоящее время технологии искусственного интеллекта, включая машинное обучение, искусственные нейронные сети и глубокое обучение, приобрели огромную популярность. Они предоставляют возможности для улучшения эффективности и точности выполнения задач. Актуальность развития машинного обучения в целом и в медицине в частности, состоит в том, что применение данных технологий способствует повышению скорости, качества и надежности выполнения целого спектра задач. С развитием информационных технологий и возросшим доступом к медицинским данным появилась возможность использовать автоматизированные методы обработки медицинских изображений и текстов для повышения эффективности и точности диагностики.

В частности, процесс чтения и интерпретации радиологических изображений обычно осуществляется квалифицированными специалистами, такими как радиологи и патологоанатомы. Эти специалисты составляют в письменном виде текстовые отчеты, описывающие результаты их анализа каждой анатомической области, представленной на изображении, и указывающие на ее нормальные, аномальные или потенциально аномальные характеристики.

Однако для менее опытных медицинских специалистов, особенно тех, кто работает в малонаселенных районах с низким уровнем медицинского обслуживания, составление отчетов по медицинским изображениям требует значительных усилий. Например, для правильной интерпретации изображений магнитно-резонансной томографии (МРТ) мозга необходимо обладать глубоким пониманием анатомии и функции мозга, а также способностью распознавать патологические изменения, которые могут быть неочевидными для неподготовленного наблюдателя.

В странах с высокой плотностью населения, таких как Индия или Китай, каждый день врачам приходится заполнять огромное число отчетов, количество которых исчисляется сотнями, что отнимает у них много времени. Процесс написания такого отчета обычно занимает около 5–10 минут. В результате как опытным, так и неопытным медицинским специалистам составление отчетов по визуализации медицинских изображений представляет собой трудоемкий процесс.

Актуальность данного исследования обусловлена несколькими факторами. Во-первых, современные технологии и методы обработки медицинских изображений и текстов имеют огромный потенциал для улучшения диагностики и лечения пациентов. С помощью автоматизированных методов можно обнаруживать и анализировать патологии на изображениях с большей точностью и скоростью, что может привести к более раннему выявлению заболеваний и принятию более верного решения дальнейшего лечения. Во-вторых, с появлением больших объемов медицинских данных и электронных медицинских записей становится все сложнее эффективно управлять и анализировать эту информацию. Т.е. у врачей и исследователей возникает потребность в применении современных технологий для автоматизации процесса анализа и интерпретации данных. В-третьих, использование машинного обучения и глубокого обучения позволяет создавать модели, способные обучаться на больших наборах данных и принимать решения, аналогичные решениям врачей. Это может значительно повысить эффективность диагностики и помочь в принятии взвешенных решений о лечении пациентов.

1. Рентгенологическая диагностика

На сегодняшний день существует множество заболеваний, которые представляют серьезную угрозу для здоровья людей. Среди них имеются и заболевания органов дыхания, которые были включены в список 10-ти основных причин смертности по всему миру Всемирной организацией здравоохранения в 2011-м году. Следовательно, важно проводить тщательную и своевременную диагностику заболеваний органов дыхания, поскольку от этого часто зависит жизнь пациента [1].

Важное открытие Вильгельма Рентгена в 1895-м году сыграло ключевую роль в различных областях науки. Не только физики, но и химики, биологи и геологи воспользовались его результатами. Рентгенография уже более ста лет служит в медицине и до сих пор остается одним из наиболее информативных диагностических методов. Она используется специалистами различных областей медицины для обнаружения патологических изменений в различных частях и органах человеческого тела.

Исследование проводится с использованием медицинских рентгеновских аппаратов. Рентгеновские лучи, которые они генерируют, проходят через тело человека и фиксируются системой. Современные цифровые системы для рентгенографии оснащены чувствительным детектором, который мгновенно передает рентгеновское изображение на компьютерный монитор. Рентгенограммы, в сущности, представляют собой изображения с негативным оттенком, где более светлые области определяются как затемнения. Например, области инфильтрата в легких, которые являются плотными и более светлыми на фоне "темных" легочных тканей, могут быть описаны врачом как тени. Перелом кости проявляется как более темный "разлом" на светлом "фоне" кости. Теневые изображения, полученные с помощью рентгенографии, предоставляют врачу информацию о состоянии различных органов (таких как легкие, сердце, желудок, лимфатические узлы, кости, позвоночник и др.), а также помогают выявить различные патологические состояния, включая участки воспаления, деструкцию, дистрофию, опухоли и аномалии развития органов.

Для достижения максимальной информативности часто проводят рентгенографию в нескольких проекциях: прямой (передняя и задняя), боковой (левая или правая). При необходимости выполняются дополнительные методы обследования, включая позиционирование пациента в различных положениях: горизонтальном (для обнаружения гидроторакса, который может изменять своё положение при изменении ориентации тела), в наклоне назад (для выполнения лордотической рентгенографии, обеспечивающей лучшую визуализацию верхушек лёгких, например, для исключения опухоли Панкоста), на выдохе (для улучшения диагностики пневмоторакса), а также в косых проекциях [1].

2. Компьютерное зрение и обработка естественного языка

Компьютерное зрение (англ. Computer Vision, CV) – это междисциплинарная область, находящаяся на стыке искусственного интеллекта, компьютерных наук и обработки изображений. Целью данной области является разработка систем, способных обрабатывать и интерпретировать визуальную информацию (изображения и видео) аналогично человеческому зрению.

Задачи компьютерного зрения многообразны и охватывают широкий спектр, начиная от базовых функций, таких как обнаружение краев и сегментация изображения, до более сложных задач, включающих распознавание объектов, отслеживание движения, восстановление сцены и анализ текстур. Решение этих задач предполагает глубокое понимание контекста, семантики, геометрических свойств и структурной организации изображения, что делает компьютерное зрение сложной и перспективной областью исследований [2].

Согласно исследованию технологического портала TAdviser, с 2018-го по 2023-й год наблюдается 5-кратный рост объема отечественного рынка решений в данной сфере, достигнув отметки до 38 млрд. рублей. Наиболее крупную долю составляют решения в области видеонаблюдения и безопасности (32%), промышленности (17%), медицины (14%) и торговли (10%).

Современные алгоритмы, основанные на сверточных нейронных сетях и визуальных трансформерах, демонстрируют очень впечатляющую эффективность в решении широкого круга задач [3]. Однако, несмотря на эти достижения, компьютерное зрение сталкивается с рядом трудностей. Изображения и видеоданные представляют собой сложные многомерные структуры, подверженные шумам, искажениям, изменениям освещения и перспективы. Анализ сцен, содержащих множество объектов, которые могут частично перекрываться или взаимодействовать друг с другом непредсказуемым образом, представляет значительную сложность для современных систем компьютерного зрения.

Обработка естественного языка (англ. Natural Language Processing, NLP) – это стремительно развивающаяся область искусственного интеллекта и компьютерной лингвистики, занимающаяся анализом, пониманием, изучением человеческого языка, а также разработкой методов взаимодействия компьютеров с ним. Это область, где совмещаются компьютерные науки, искусственный интеллект и лингвистика с целью создания систем, способных понимать, интерпретировать и генерировать текст так же, как это делают люди на естественном языке. Это включает в себя понимание не только буквального значения слов, но и контекста, в котором они используются, а также способность улавливать нюансы и подтексты.

Задачи обработки естественного языка многообразны и включают в себя широкий спектр областей, как машинный перевод, распознавание речи, генерация текста, извлечение информации, анализ тональности текста, автоматическое реферирование и многие другие. Эти задачи требуют понимания контекста, семантики, синтаксиса и структуры языка.

Актуальность обработки естественного языка обусловлена стремительным ростом объема текстовых данных, доступных в цифровом виде. Это позволяет автоматизировать обработку информации, извлекая ценные знания из огромных массивов текста, которые невозможно анализировать вручную.

На сегодняшний день обработка естественного языка находит применение в самых различных областях:

– бизнес (анализ отзывов клиентов, автоматизация обслуживания клиентов, маркетинговые исследования);
– наука (анализ научных публикаций, автоматизация исследований);
– медицина (обработка медицинских записей, диагностика заболеваний);
– образование (автоматическая оценка сочинений, персонализированное обучение).

С развитием технологий машинного обучения и глубокого обучения, возможности обработки естественного языка значительно расширились. Современные модели, такие как трансформеры и предварительно обученные модели языкового моделирования, такие как GPT и BERT, показывают впечатляющие результаты в широком спектре задач обработке естественного языка [4].

3. Сверточные и рекуррентные нейронные сети

Сверточные нейронные сети (англ. Convolutional Neural Network, CNN) представляют собой тип глубоких нейронных сетей, разработанных для анализа и обработки структурированных данных, в частности, для работы с изображениями. Они были вдохновлены биологическими процессами зрения животных, где клетки рецепторы в глазу реагируют на различные участки визуального поля.

Главная идея CNN заключается в использовании сверточных слоев для автоматического извлечения иерархии признаков из входных данных. Сверточные слои применяют фильтры к входным данным с тем, чтобы выделить различные характеристики, такие как границы, текстуры и формы объектов.

Сверточные нейронные сети состоят из нескольких ключевых компонентов (слоев), таких как сверточные слои, слои субдискретизации и полносвязные слои. В совокупности они обеспечивают их способность эффективно анализировать и обрабатывать визуальные данные (рис. 1).

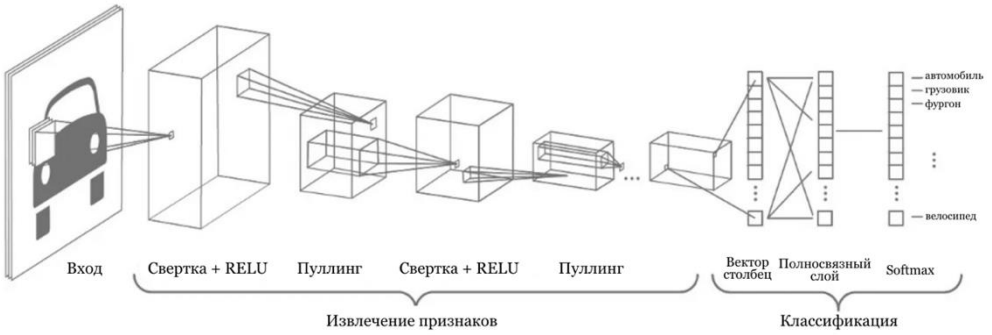


Рис. 1. Пример простой сверточной нейронной сети для задачи классификации

Сверточные слои (англ. convolutional layers) являются основой сверточных нейронных сетей. Они работают путем применения набора фильтров к входным данным. Эти фильтры представляют собой матрицы весов, которые скользят по входным данным (например, изображению) с определенным шагом (шаг свертки), вычисляя взвешенную сумму значений пикселей. Результатом этой операции является активационная карта (feature map), которая представляет собой карту активации (рис. 2) для каждого фильтра. Она показывает, насколько хорошо или плохо у нас активировался фильтр. Сверточные слои позволяют сети извлекать локальные признаки изображения, такие как границы, углы и текстуры. Обычно они содержан несколько слоев сверток.

Фильтры (или ядра свертки) представляют собой матрицы весов, которые применяются к входным данным в процессе свертки (рис. 2). Каждый фильтр обучается находить определенные паттерны или признаки в данных. Например, один фильтр может находить горизонтальные границы, а другой – вертикальные границы. Если паттерн присутствует на изображении, то карта активации после применения соответствующего фильтра будет содержать большие числовые значения.

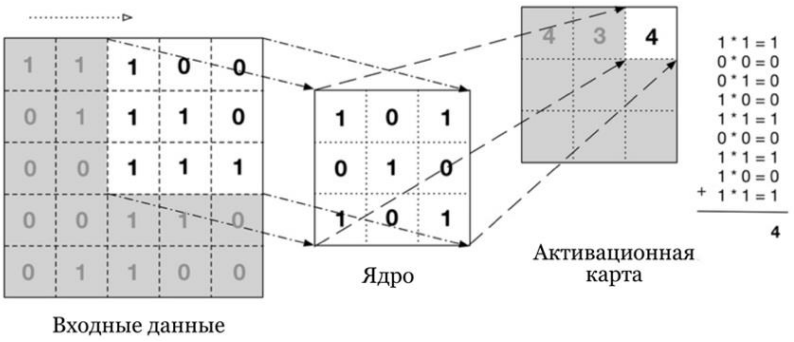


Рис. 2. Процесс свертки ядра с изображением

CNN широко применяются в обработке изображений, но также могут быть использованы в других областях, таких как обработка звука и анализ текста. Эти сети демон-

стрируют выдающиеся результаты в распознавании образов, классификации объектов, сегментации изображений и других задачах компьютерного зрения.

Рекуррентные нейронные сети (англ. Recurrent Neural Networks, RNN) представляют собой класс нейронных сетей, способных обрабатывать последовательности данных и учитывать их контекст. Они обладают возможностью использовать информацию о предыдущих состояниях для обработки текущего входа. Это делает их особенно эффективными для анализа временных рядов, обработки естественного языка, генерации текста и других задач, где важно учитывать последовательную структуру данных.

Основной особенностью рекуррентных нейронных сетей является наличие обратных связей, благодаря которым они могут сохранять информацию о предыдущих состояниях и использовать ее при обработке последующих входов (рис. 3). Это позволяет модели учитывать долгосрочные зависимости в данных и делает их более мощными и гибкими в сравнении с традиционными нейронными сетями.

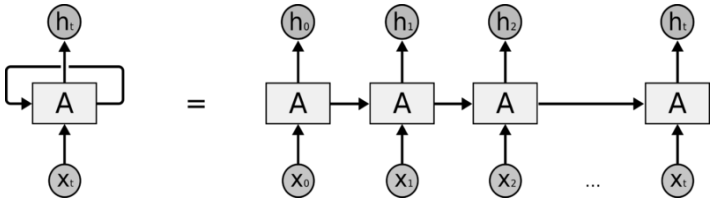


Рис. 3. Рекуррентная нейронная сеть и ее представление в развернутом виде

Однако у RNN есть свои ограничения, включая проблему затухающих градиентов при обучении на длинных последовательностях, а также сложности в передаче информации на большие временные расстояния из-за потери информации в процессе обновления состояний. Для решения этой проблемы были разработаны более продвинутые архитектуры, такие как LSTM и GRU.

4. Трансформерная архитектура

Трансформерная архитектура представляет собой инновационный подход к обработке последовательных данных в области глубокого обучения. Она была впервые представлена в статье "Attention is All You Need" [5] компанией Google в 2017 г. и стала ключевым компонентом в решении таких задач, аналогичным рекуррентным нейронным сетям, как машинный перевод, суммаризация текста, генерация текста и другие. Трансформер стал основой для многих современных моделей обработки естественного языка, таких как BERT, GPT-2, GPT-3 и др.

Отличительной особенностью трансформеров от рекуррентных нейросетей является их способность обрабатывать входные данные параллельно, а не последовательно [3]. Например, при обработке текста ему не требуется дожидаться завершения обработки начала текста для того, чтобы приступить к его концу. Это позволяет эффективнее параллелизировать вычисления и обучаться быстрее.

Трансформер состоит из нескольких стековых слоев, каждый из которых содержит две основные компоненты: механизм внимания и полносвязные нейронные сети. Механизм внимания включает в себя несколько подмодулей, таких как механизм самовнимания и механизм межслойного внимания (англ. multi-head attention). Слои трансформера также содержат нормализацию по слоям (англ. layer normalization) и применение функций активации, таких как функция выпрямленного линейного блока.

Механизм внимания (англ. attention mechanism) позволяет модели фокусироваться на различных частях входных данных, присваивая каждой соответствующий вес в зависимости от её важности для конкретной задачи.

Трансформеры используют многоуровневые входные представления, которые позволяют модели адаптироваться к различным уровням абстракции в данных. Это позво-

ляет модели извлекать более сложные и информативные признаки из входных последовательностей.

Энкодер и декодер, в совокупности – автокодировщик или автоэнкодер, представляют собой основные компоненты трансформерной архитектуры (рис. 4).

Рис. 4. Архитектура простого автокодировщика

Энкодер преобразует входную последовательность в скрытое представление (латентное пространство, latent space), обогащенное информацией о структуре и содержании входных данных. Он состоит из нескольких идентичных слоев, каждый из которых включает в себя два основных компонента: слой самовнимания и полносвязные слои с функцией активации. Слой самовнимания позволяет модели сфокусироваться на различных частях входных данных, в то время как полносвязные слои выполняют преобразование скрытого представления.

Декодер генерирует выходную последовательность на основе скрытого пространства, созданного энкодером, и предыдущих выходов. Он также состоит из нескольких слоев, но включает в себя дополнительный слой внимания, который позволяет модели учитывать связь между предыдущими и текущими состояниями.

Визуальные трансформеры (англ. Vision Transformer, ViT) представляет собой инновационную архитектуру в области компьютерного зрения, основанную на идеях трансформеров, применяемых в обработке естественного языка. Они были представлены в 2021 г. в статье "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" [6], достигнув высоких результатов в задачах классификации изображений. По словам авторов статьи, применение трансформерной архитектуры в компьютерном зрении было вдохновлено успехами аналогичных моделей в обработке естественного языка: применение общего подхода, такого как самовнимание, позволяет избежать необходимости включения в архитектуру специфичных особенностей задачи (т.н. индуктивного смещения, inductive bias) при достаточном объеме обучающих данных, времени обучения и числе параметров модели. Устройство визуального трансформера представлено на рис. 5.

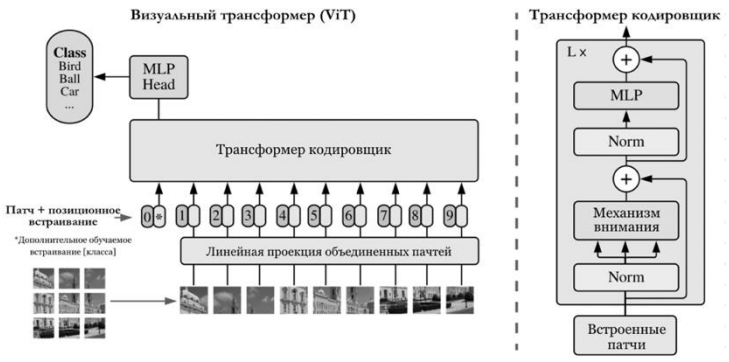


Рис. 5. Устройство визуального трансформера (слева), блок кодировщика (справа)

Сначала входное изображение разбивается на небольшие участки, называемые патчами или сегментами. Каждый патч представляет собой квадратную или прямоугольную область пикселей из исходного изображения. Затем каждый сегмент преобразуется в вектор фиксированной размерности с помощью обучаемого линейного слоя

векторизации. Он конвертирует каждый патч пикселей в векторное представление, которое затем будет использоваться в архитектуре трансформера. После преобразования патчей в каждый вектор добавляются позиционные эмбеддинги. Они представляют собой информацию о расположении каждого патча в исходном изображении и позволяют модели учитывать его пространственную структуру.

Каждый векторизованный патч, включая позиционные эмбеддинги, подается на вход трансформерной архитектуре. Механизм внимания трансформера используется для извлечения взаимосвязей и зависимостей между патчами изображения, как если бы они были частями последовательности. Внутри трансформера каждый патч обрабатывается несколькими слоями энкодера. Каждый слой энкодера состоит из механизма внимания и полносвязных слоев, которые выполняют преобразование и агрегацию информации между патчами.

После обработки всех патчей трансформером получается глобальное скрытое представление изображения, которое содержит информацию о всем изображении в целом и в дальнейшем может использоваться в рамках исследуемой задачи.

В 2021 г. исследователи из Microsoft Research Asia во главе с Лю Цзе (Ze Lui) представили новую модель под названием SWIN (англ. Shifted WINdow – сдвинутое окно) Transformer [7]. Эта модель, которая является трансформером общего назначения, была специально адаптирована для решения задач компьютерного зрения. SWIN-трансформер демонстрирует более высокую производительность по сравнению с визуальным трансформером ViT и архитектурами на основе самовнимания, использующими сверточные нейронные сети.

В трансформерной архитектуре SWIN начальный слой аналогичен тому, что используется в визуальном трансформере – исходное изображение разбивается на патчи и проецируется с помощью линейного слоя (рис. 6). Отличие заключается в том, что в SWIN на первом слое размер патчей составляет 4×4, что позволяет обрабатывать более детализированный контекст.

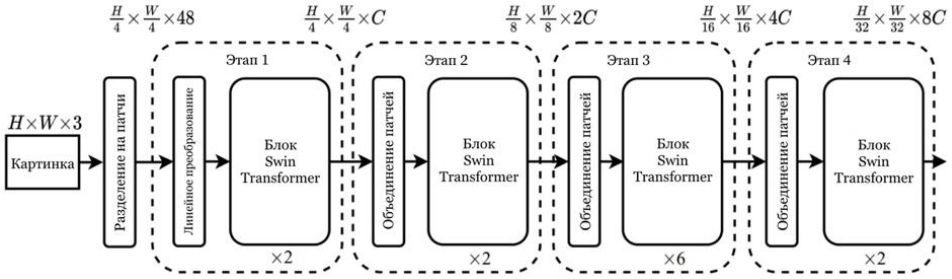


Рис. 6. Архитектура SWIN Transformer

Затем следуют несколько слоев объединения патчей (Patch Merging) и блок SWIN Transformer. Слой объединения патчей соединяет признаки соседних (в окне 2×2) элементов и уменьшает размерность, создавая более высокоуровневое представление. Т.о., после каждого этапа формируются карты признаков, содержащие информацию на различных пространственных масштабах. Это позволяет получить иерархическое представление изображения, которое может быть полезно для последующей сегментации, обнаружения объектов и т.д.

GPT-2 (англ. Generative Pre-trained Transformer 2 – генеративный предварительно обученный трансформер) – это большая языковая модель, разработанная компанией OpenAI в 2019 г. [8]. Она была предварительно обучена на наборе данных из 8 миллионов веб-страниц (40 ГБ текста). Он был частично выпущен в феврале 2019 г., а полная версия модели с 1.5 миллиардами параметров была выпущена 5 ноября 2019 г.

Данная модель была предоставлена свобода изучения языковых структур и паттернов без привязки к определённой функции. GPT-2 была создана как "прямое масштабирование" GPT-1 с десятикратным увеличением количества параметров и размера обучающего набора данных. Это общецелевой обучающийся алгоритм, и его способность выполнять различные задачи была следствием его общей способности точно предсказывать следующий элемент в последовательности, что позволило ему переводить тексты, отвечать на вопросы о теме из текста, суммировать фрагменты из большего текста и генерировать текстовый вывод на уровне, иногда неотличимом от человеческого.

GPT-2, как и его предшественник GPT-1 и его преемники GPT-3 и GPT-4, имеет архитектуру генеративного предобученного трансформера, реализующую глубокую нейронную сеть, в частности – модель трансформера, которая использует механизмы внимания. Они позволяют модели выборочно сосредотачиваться на сегментах входного текста, которые она предсказывает как наиболее релевантные. Эта модель позволяет значительно увеличить параллелизацию и превосходит предыдущие показатели для моделей на основе рекуррентных, сверточных нейросетей и длинной цепи краткосрочных элементов LSTM [9].

Сейчас GPT-2 доступна в открытом доступе на платформе Hugging Face, что делает ее доступной для исследователей и разработчиков [10]. Это значит, что GPT-2 может быть дополнительно обучена на своих собственных данных, например, на медицинских текстах, чтобы улучшить ее способность генерировать медицинские заключения. Это позволяет адаптировать модель для более точной генерации заключений, учитывая специфику конкретных заболеваний.

5. Обработка данных и метрики качества

В качестве набора данных использовалась открытая коллекция рентгенограмм грудной клетки Университета Индианы, доступная на публичной веб-платформе Kaggle. Эта база данных включает в себя широкий набор рентгеновских снимков грудной клетки, включая снимки пациентов с различными патологиями, такими как пневмония, туберкулез, рак легких и заболевания сердца. Помимо снимков пациентов с различными заболеваниями и аномалиями, набор данных содержит рентгенограммы полностью здоровых пациентов.

Исходные изображения были загружены в необработанном стандарте DICOM. Для дальнейшей работы каждое из них было преобразовано в формат png с использованием следующей постобработки:

1. Верхние/нижние 0.5% значений пикселей DICOM были обрезаны. Это позволило исключить очень темные или очень яркие пиксели, которые могут содержать шум и не нести полезной информации.
2. Произведено линейное масштабирование значений пикселей DICOM для соответствия диапазону 0–255. Это обеспечило стандартизацию изображений и облегчило их дальнейшую обработку.
3. Для соответствия ограничениям набора данных Kaggle, размер изображений был изменен до 2048 пикселей по короткой стороне, а длинная сторона масштабировалась пропорционально.

Набор данных содержит 7470 рентгеновских снимков человеческой грудной клетки, как в прямой (фронтальной), так и в боковой (латеральной) проекциях

Для обеспечения систематизации и стандартизации описания медицинских изображений в наборе данных используется следующая структура.

Сравнения (Comparison) предназначены для сравнения текущих изображений с предыдущими снимками пациента, что позволяет выявить изменения, оценить динамику заболевания и эффективность применяемой терапии. Сравнение может осуществляться с помощью анализа анамнеза пациента и сопоставления рентгенологических изображений, полученных в разные моменты времени.

Показания (Indication) указывают на причину выполнения рентгенологического исследования. Данный атрибут содержит информацию о симптомах, предполагаемых заболеваниях и целях проведения исследования. Индикация помогает определить, какие конкретные данные и патологии врачи искали при анализе снимка.

Обнаружения (Findings) содержат конкретные данные, выявленные при анализе рентгенограммы грудной клетки. В этом разделе описываются структура грудной клетки, размеры органов, выявленные изменения (опухоли, инфекции, травмы или другие патологии). Результаты представляются в ясной и точной форме для обеспечения полного понимания состояния пациента.

Заключение (Impression) представляет собой обобщенное заключение о рентгенологическом исследовании. Включает в себя диагнозы, предположения и рекомендации специалиста, основанные на результатах исследования. Заключение может содержать информацию о вероятных диагнозах, рекомендованных лечебных мероприятиях и необходимости проведения дальнейших исследований для уточнения состояния пациента.

Медицинские предметные рубрики (MeSH, Medical Subject Headings) – это стандартизированный тезаурус, разработанный и обновляемый Национальной медицинской библиотекой США, используемый для индексирования и систематизации медицинских и научных публикаций. В описании медицинских изображений MeSH применяется для указания конкретных терминов, относящихся к патологиям, органам и процедурам.

Таблица 1

Пример отчета для рентгенограммы грудной клетки

Наименование признака	**Описание на английском языке**	**Описание на русском языке**
Comparison	Chest radiographs XXXX	Рентгенограмма грудной клетки XXXX
Indication	XXXX-year-old male, chest pain	Мужчина XXXX лет, боль в груди
Findings	The cardiomediastinal silhouette is within normal limits for size and contour. The lungs are normally inflated without evidence of focal airspace disease, pleural effusion, or pneumothorax. Stable calcified granuloma within the right upper lung. No acute bone abnormality	Кардиомедиастинальная тень в пределах нормы по размеру и контурам. Легкие нормально раздуваются, без признаков очаговых воздушных заболеваний, плеврального выпота или пневмоторакса. Стабильная кальцинированная гранулема в верхней доле правого легкого. Острой костной патологии не выявлено.
Impression	No acute cardiopulmonary process	Острого кардиопульмонального процесса не выявлено.
MeSH	Calcified Granuloma/lung/upper lobe/right	Кальцинированная гранулема/легкое/верхняя доля/правая

Рентгеновские снимки грудной клетки, полученные в клинических условиях, характеризуются вариабельностью размеров, разрешения, контрастности и наличия артефактов. Предварительная обработка направлена на стандартизацию данных и устранение шумов, что повышает эффективность обучения моделей глубокого обучения [11].

Стандартные методы нормализации включают:

– масштабирование: значения пикселей приводятся к диапазону от 0 до 1;
– стандартизация: значения пикселей масштабируются с использованием среднего значения и стандартного отклонения;
– масштабирование размеров: изображения масштабируются до единого размера, например, 224×224 пикселей, с сохранением соотношения сторон;
– интерполяция: применяется для изменения разрешения изображения при необходимости.

Аугментация данных используется для увеличения объема обучающей выборки и улучшения обобщающей способности моделей [12]. Она предполагает применение раз-

личных трансформаций к изображениям: вращение, сдвиг, изменение масштаба, отражение, изменение яркости.

Извлечение признаков – следующий этап после предварительной обработки, представляющий собой преобразование исходных данных изображения в компактное представление, содержащее наиболее значимую информацию для решения поставленной задачи. В контексте сверточных нейронных сетей и трансформеров извлечение признаков осуществляется автоматически через слои сети.

Предварительная обработка текста играет важную роль в стандартизации и очистке данных. Этот процесс, также известный как нормализация, включает ряд этапов, направленных на приведение текстовых данных к форме, пригодной для использования системами обработки естественного языка и интеллектуальными системами, основанными на машинном и глубоком обучении. Целью предварительной обработки является удаление ненужного содержимого из текстовых документов и получение чистых данных, оптимизированных для анализа [4].

Одним из первых этапов предварительной обработки является очистка текста. Данный этап включает:

– удаление ненужных символов: специальных символов, знаков препинания, чисел;
– приведение слов к нижнему регистру: обеспечивает единообразие представления слов и упрощает анализ;
– учет специфики медицинской терминологии: сохранение медицинских аббревиатур и терминов, важных для анализа.

Важно учитывать особенности медицинской терминологии и специфические требования к текстовым данным в контексте медицинского исследования. Предварительная обработка должна быть адаптирована к конкретным задачам и типам данных, используемых в исследовании.

После проведения очистки текста следующим этапом предварительной обработки является токенизация, удаление стоп-слов и стемминг. Эти методы позволяют структурировать текстовые данные и выделить ключевые элементы для последующего анализа.

Оценка качества текстовых данных является критически важным аспектом в задачах, связанных с автоматическим созданием медицинских заключений, т.к. точность и достоверность выводов могут существенно повлиять на клинические решения.

Метрика BLEU (Bilingual Evaluation Understudy) является одной из первых и наиболее популярных метрик для автоматической оценки качества машинного перевода и текстовой генерации. Она основана на вычислении сходства между сгенерированным текстом и одним или несколькими эталонными текстами (референсами). BLEU измеряет количество совпадающих n-грамм (последовательностей из n слов) между этими текстами, и выдает оценку от 0 до 1, где 1 – идеальное совпадение.

Метрика ROUGE (Recall-Oriented Understudy for Gisting Evaluation), в отличие от BLEU, акцентирует внимание на полноте (recall), а не только на точности. ROUGE включает несколько вариантов, такие как ROUGE-N (на основе n-грамм), ROUGE-L (учитывает длину наибольшей общей подпоследовательности). Как и BLEU, выдает оценку от 0 до 1.

METEOR (Metric for Evaluation of Translation with Explicit Ordering) разработана для преодоления некоторых ограничений BLEU, таких как нечувствительность к порядку слов и морфологическим вариациям. Она выдает оценку от 0 до 1. METEOR учитывает точность и полноту, а также использует парафразные соответствия и синонимы для повышения точности оценки.

BERTScore представляет собой более современную метрику, основанную на глубинных языковых моделях, таких как BERT. В отличие от традиционных метрик, BERTScore использует эмбеддинги слов, чтобы вычислить сходство между сгенерированным и эталонным текстами на уровне смысловых представлений. Это позволяет

BERTScore учитывать контекстуальные и семантические аспекты текста, что делает ее особенно полезной для задач, где важны не только синтаксическое, но и семантическое соответствие. BERTScore, как и предыдущие метрики, выдает оценку от 0 до 1.

Использование этих метрик должно быть дополнено экспертной оценкой для получения полного представления о качестве сгенерированного текста.

6. Моделирование

В данной работе были обучены несколько вариантов моделей для автоматической генерации описания рентгенограммы человека. Одной из первых рассмотренных вариантов является модель, основанная на архитектуре VGG16. Она состоит из трех основных компонентов, взаимодействующих для предсказания следующего слова в последовательности:

Экстрактор (англ. extract – извлекать) признаков изображений. Используется предварительно обученная 16-слойная модель VGG16, обученная на наборе данных ImageNet.

Процессор последовательностей. Входные текстовые последовательности фиксированной длины (125 слов) обрабатываются слоем встраивания слов, отображающим каждое слово в векторное представление.

Третья часть модели представляет собой декодер. Он объединяет векторы признаков из экстрактора изображений и процесса последовательностей путем сложения соответствующих элементов.

Несмотря на реализацию предложенной архитектуры, результаты модели на основе VGG16 оказались неудовлетворительными, что обусловлено несколькими факторами. Наблюдалась тенденция к генерированию повторяющихся фрагментов текста, преимущественно утверждая об отсутствии патологий у исследуемого пациента. Это приводило к искажению описания медицинских изображений и, как следствие, к неправильной классификации.

В связи с возникшими проблемами при использовании модели VGG16 с предобученными весами ImageNet, было решено провести эксперименты с другой моделью – CheXNet.

CheXNet представляет собой сверточную нейронную сеть, состоящую из 121-го слоя. Данная нейронная сеть была обучена на крупнейшем общедоступном наборе данных ChestX-ray14, содержащем более 100000 фронтальных рентгеновских изображений с 14-ю различными заболеваниями [13]. Модель CheXNet продемонстрировала некоторое улучшение по сравнению с моделью VGG16, но результаты, по-прежнему, оставались неудовлетворительными. В табл. 2 представлены значения метрик качества для обеих моделей, рассчитанные на выборке из 50-ти случайных пациентов.

Таблица 2

Оценка метрик качества для моделей VGG16 и CheXNet (LSTM)

Метрика	VGG16	CheXNet
BLEU-1	0.285	0.304
BLEU-2	0.210	0.220
BLEU-3	0.189	0.197
BLEU-4	0.145	0.158
ROUGE-1	0.212	0.286
ROUGE-2	0.151	0.251
ROUGE-L	0.161	0.222
METEOR	0.134	0.277

В следующей попытке создания модели генерации медицинского заключения была использована архитектура Inceptionv3, также предварительно обученной на наборе данных ImageNet.

Ниже представлено предсказание модели и сгенерированное заключение для случайного пациента, рентгеновский снимок которого изображен на рис. 7.

Рис. 7. Рентгенограмма пациента, на которой проводилось тестирование

Исходное медицинское заключение и перевод на русский язык для рис. 7 представлено в табл. 3.

Таблица 3

Исходное медицинское заключение

Исходное заключение	Перевод на русский язык
the heart is normal in size atherosclerotic calcifications of the aorta the mediastinum is stable there is again soft tissue density projected over the right mid chest patients known large breast mass the appearance is grossly stable to decreased from prior study the lateral projection is suboptimal as patient could not raise there is no pleural effusion	Сердце нормального размера. Атеросклеротические кальцификации аорты. Средостение стабильно. Снова наблюдается мягкотканная плотность, проецируемая на среднюю часть правой стороны грудной клетки. У пациента известна большая масса в груди. Внешний вид в целом стабилен или снижен по сравнению с предыдущим исследованием. Боковая проекция субоптимальна, т.к. пациент не мог поднять руку. Отсутствует плевральный выпот.

В табл. 4 представлено сгенерированное заключение по рентгенограмме грудной клетки и его перевод.

Таблица 4

Сгенерированное медицинское заключение (Inceptionv3)

Сгенерированное заключение	Перевод на русский язык
the heart and pulmonary vasculature are within normal limits the lungs are clear the aorta is normal in size the lungs are clear there is no focal consolidation no pleural effusion or pneumothora no acute bony findings or infiltrate there is no focal airspace consolidation is seen or focal air within the right middle and lateral views the interval there is no focal airspace consolidation there are no acute bony abnormality or focal consolidation the lungs are hyperinflated but clear and cardiopulmonary silhouette is stable the lungs are clear or significant pleural effusion there are no	Сердце и легочные сосуды в пределах нормы. Легкие чистые. Аорта нормального размера. Легкие чистые. Отсутствует очаговая консолидация. Отсутствует плевральный выпот или пневмоторакс. Отсутствуют острые костные изменения или инфильтрация. Нет очаговой воздушной консолидации, видимой на средних и боковых проекциях. Нет очаговой воздушной консолидации. Нет острых костных аномалий или очаговой консолидации. Легкие гипервентилированы, но чистые. Кардиопульмональный силуэт стабилен. Легкие чистые. Нет существенного плеврального выпота.

Как можно заметить, в сгенерированном описании также наблюдается повторение текста. Проведем сравнение текстов. Совпадения:

– Размер сердца: оба описания указывают на нормальный размер сердца.

– Состояние аорты: оба описания упоминают аорту, в исходном – наличие кальцификации, в сгенерированном – нормальный размер.

– Легкие: оба описания указывают на отсутствие плеврального выпота и очаговой консолидации.
– Стабильность: оба описания указывают на стабильность медиастинума и кардиопульмонального силуэта.

Различия:

– Описание изменений в легких: исходное описание упоминает мягкотканную плотность в правой части груди, связанную с известной массой молочной железы, тогда как сгенерированное описание указывает на наличие воздушных полостей в правой части грудной клетки.
– Латеральная проекция: исходное описание отмечает субоптимальное качество латеральной проекции из-за невозможности пациента поднять руку, в сгенерированном описании эта информация отсутствует.
– Дополнительные сведения: сгенерированное описание содержит информацию о легочных сосудах, гипервентиляции легких, отсутствии пневмоторакса и острых костных патологий, которой нет в исходном описании.

Сгенерированное описание в целом соответствует исходному описанию, но содержит некоторые неточности и дополнения. Модель Inceptionv3 успешно распознала основные элементы рентгеновского снимка, такие как размер сердца, состояние аорты, наличие плеврального выпота и очаговой консолидации для данного примера. Однако она не смогла передать специфику мягкотканной плотности в правой части грудной клетки и не уловила информацию о качестве латеральной проекции.

Также было исследована возможность применения сверточной нейронной сети CheXNet на основе пайплайна предыдущей модели. В этой реализации используется управляемый рекуррентный блок GRU. Полученные результаты для пациента, рентгенограмма которого изображена на рис. 7, представлены в табл. 5.

Таблица 5

Сгенерированное медицинское заключение (CheXNet)

Сгенерированное заключение	Перевод на русский язык
cardiomediastinal silhouette is normal limits bibasilar atelectasis or hemothora no pneumothora or pleural effusion the heart size is within normal limits pulmonary vascularity is normal there is no pleural effusion there is unremarkable visualized bony findings or pleural effusions or pleural effusion or focal alveolar consolidation no evidence of	Кардиомедиастинальный силуэт в пределах нормы. Бибазальная ателектазия или гемоторакс. Отсутствует пневмоторакс или плевральный выпот. Сердце нормальных размеров. Легочные сосуды нормальные. Отсутствует плевральный выпот. Не выявлено изменений в костях, плевральном выпоте или очаговой альвеолярной консолидации. Нет признаков...

Сравнивая исходный текст (табл. 3) и сгенерированный (табл. 5), можно выявить следующие сходства:

– Размер сердца: оба описания указывают на нормальный размер сердца.
– Стабильность: оба описания указывают на стабильность медиастинума и кардиомедиастинального силуэта.
– Отсутствие плеврального выпота: оба описания указывают на отсутствие плеврального выпота.

Различия:

– Описание изменений в легких: исходное описание упоминает мягкотканную плотность в правой части груди, связанную с известной массой молочной железы, тогда как сгенерированное описание указывает на бибазальную ателектазию или гемоторакс.
– Состояние аорты: исходное описание упоминает атеросклеротические кальцификации аорты, в сгенерированном описании эта информация отсутствует.
– Дополнительные сведения: сгенерированное описание содержит информацию о легочных сосудах, отсутствии пневмоторакса, незначительных костных изменениях и альвеолярной консолидации, которой нет в исходном описании.

Анализируя полученный текст, можно прийти к выводу, что сгенерированное описание не соответствует исходному описанию. Модель не смогла правильно распознать основные элементы рентгеновского снимка и не сохранила контекст исходного описания. Модель произвела несколько неверных утверждений, таких как "бибазальная ателектазия или гемоторакс", которой нет в исходном тексте.

Модель Inceptionv3 выдала более точное описание с медицинской точки зрения, хотя и добавила некоторые дополнительные сведения. Модель CheXNet содержит ошибки в диагностике, не смогла распознать ключевые элементы исходного описания.

Еще один из вариантов реализации модели для генерации медицинского отчета заключался в создании визуально-языковой модели с применением трансформерной архитектуры. Она состоит из двух ключевых компонентов.

Визуальный энкодер, как и в предыдущих вариантах, предназначен для извлечения признаков из изображений, здесь используется модель Swin Transformer. Выходной слой классификации удаляется, в результате чего получается 768-мерный вектор признаков для каждого изображения.

Языковая модель GPT-2 используется для генерации описаний изображений. Она принимает на вход вложения изображений и токенов, объединенных вместе, а затем генерирует отчет. Перед началом основного обучения GPT-2 подвергается дообучению на корпусе текстовых отчетов. Данный этап позволяет улучшить способность GPT-2 генерировать связные, логически обоснованные медицинские заключения, адаптированные к специфике предметной области.

Модель проекции представляет собой линейный слой с активационной функцией "гиперболический тангенс". Он проецирует 768-мерные эмбеддинги изображений на 768-мерные эмбеддинги, совместимые с языковой моделью GPT-2. Т.о. с помощью модели проекций векторные представления изображений приводятся к тому же пространству вложений, что и вложения токенов GPT-2.

Чтобы оценить способность GPT-2 генерировать медицинские отчеты, был проведен эксперимент, в котором в качестве входных данных использовались первые пять слов из реальных медицинских заключений. GPT-2 использует свою способность предсказывать следующий токен, чтобы сгенерировать полный отчет, который по стилю и структуре похож на реальные медицинские заключения.

Ниже в табл. 6 приведены некоторые примеры результатов языковой модели.

Таблица 6

Сгенерированный текст моделью GPT-2 после тонкой настройки

Исходное заключение	Сгенерированное заключение	Перевод на русский язык
The cardiac and mediastinal silhouettes are unremarkable. The lungs are well expanded and clear. There are no focal air space opacities. There is no pneumothorax or effusion. The bony structures of the thorax are intact with no evidence of acute osseous abnormality.	The cardiac and mediastinal silhouettes are normal in size and contour. The lungs are clear, without evidence of infiltrate. There is no pneumothorax or effusion	Сердечная и средостенная тени имеют нормальные размеры и контуры. Легкие прозрачны, без признаков инфильтрации. Нет пневмоторакса или выпота.
The lungs are clear bilaterally. Specifically, no evidence of focal consolidation, pneumothorax, or pleural effusion.. Cardio mediastinal silhouette is unremarkable. Visualized osseous structures of the thorax are without acute abnormality	The lungs are clear bilaterally. Specifically, no evidence of focal consolidation, pneumothorax, or pleural effusion.. Cardio mediastinal silhouette is unremarkable. Visualized osseous structures of the thorax are without acute abnormality	Легкие прозрачны с обеих сторон. Отсутствуют признаки очаговой консолидации, пневмоторакса или плеврального выпота. Сердечная и средостенная тени без патологических изменений. Визуализируемые костные структуры грудной клетки без признаков острого повреждения.
Stable mild thoracic levoscoliosis. Stable cardiomegaly. Multiple scattered round calcific densities XXXX represent old granulomatous disease. No pneumothorax or pleural effusion.	Stable mild thoracic levoscoliosis. Stable cardiomegaly. No focal consolidation, pneumothorax or large pleural effusion	Стабильная легкая левосторонняя сколиоз грудного отдела позвоночника. Стабильная кардиомегалия. Отсутствуют признаки очаговой консолидации, пневмоторакса или

Исходное заключение	Сгенерированное заключение	Перевод на русский язык
No focal consolidation. Moderate degenerative changes of the thoracic spine.		крупного плеврального выпота.

Результаты показали, что GPT-2 способна генерировать тексты, похожие на реальные медицинские отчеты, но присутствуют ошибки в отражении клинических отклонений. Например, в отчете, сгенерированном по подсказке "Stable mild thoracic levoscoliosis..." отсутствует описание множественных кальцификатов, упомянутых в исходном заключении. Но, в целом, результаты исследования демонстрируют потенциал использования GPT-2 для генерации медицинских отчетов.

В табл. 7 представлено сгенерированное медицинское заключение визуально-языковой моделью.

Таблица 7

Сгенерированное медицинское заключение (Визуально-языковая модель)

Сгенерированное заключение	Перевод на русский язык
Heart size is normal. Lungs are clear. No pneumonia, effusions, edema, pneumothorax, adenopathy, nodules or masses.	Размеры сердца в норме. Легкие чистые. Отсутствуют признаки пневмонии, выпота, отека, пневмоторакса, аденопатии, узелков или образований.

Сгенерированный отчет демонстрирует существенные отличия от эталонного. Модель корректно определила нормальный размер сердца, однако не смогла выявить атеросклеротические кальцификаты аорты и мягкотканное образование в правой половине грудной клетки. Можно заключить, что сгенерированный отчет представляет собой обобщенное описание без учета специфических наблюдений, присутствующих в эталонном отчете. Подобное качество описания наводит на мысль, что проблемы с генерацией отчетов по изображениям могут быть связаны с визуальной компонентой модели, а именно с SWIN Transformer.

В табл. 8 представлены сводные значения метрик качества рассмотренных моделей в данной работе.

Таблица 8

Метрики качества для рассмотренных моделей

Метрика	VGG16	VGG16 (CheXNet)	Inceptionv3	DenseNet121 (CheXNet)	Vision Language Model
BLEU-1	0.285	0.304	0.303	0.342	0.202
BLEU-2	0.210	0.220	0.199	0.223	0.114
BLEU-3	0.189	0.197	0.131	0.146	0.074
BLEU-4	0.145	0.158	0.084	0.093	0.049
ROUGE-1	0.212	0.286	0.467	0.422	0.407
ROUGE-2	0.151	0.251	0.191	0.131	0.174
ROUGE-L	0.161	0.222	0.344	0.303	0.293
METEOR	0.134	0.277	0.287	0.272	0.213
BERTScore	–	–	0.825	0.836	0.882

Из данной таблицы видно, что модели Inceptionv3 и DenseNet121 (CheXNet) показали наилучшие результаты

Заключение

В ходе исследования были разработаны и протестированы модели для генерации медицинских заключений по рентгеновским снимкам грудной клетки с использованием нейронных сетей. Основные этапы включали выбор и настройку моделей, таких как VGG16, Inceptionv3 и CheXNet, а также разработку визуально-языковой модели на основе Swin Transformer и GPT-2.

Результаты показали, что модели, использованные для генерации заключений, обладают различной степенью точности и полноты. Inceptionv3 продемонстрировала лучшую способность к распознаванию основных медицинских элементов рентгеновских снимков, таких как размер сердца и состояние легких, но она имела трудности с интерпретацией более сложных и специфических диагнозах. CheXNet, хотя и обладала высоким уровнем распознавания однословных фраз, также не всегда корректно интерпретировала более сложные медицинские данные. Визуально-языковая модель, включающая Swin Transformer и GPT-2, показала потенциал в генерации текстов, близких к реальным медицинским отчетам, но также имела свои недостатки, такие как пропуск важных клинических деталей.

В целом, результаты подтверждают, что использование нейронных сетей для автоматической генерации медицинских заключений имеет значительный потенциал, но требует дальнейшего улучшения и доработки для достижения клинически приемлемого уровня точности и полноты. Высокие значения BERTScore указывают на значительную семантическую схожесть с исходными текстами, что подтверждает потенциал моделей в генерации осмысленных медицинских отчетов.

Для повышения точности и полноты генерируемых заключений необходимо проводить дальнейшие исследования и оптимизацию моделей, включая

– расширение обучающего набора данных,
– доработку архитектуры моделей для лучшего понимания и интерпретации сложных медицинских данных,
– интеграцию с существующими клиническими информационными системами, что позволит эффективно использовать разработанные модели в медицинской практике,
– тестирование моделей в реальных клинических условиях для оценки их практической применимости и выявления дополнительных областей для улучшения.

Для визуально-языковой модели, помимо SWIN Transformer, можно использовать другие современные и мощные архитектуры трансформеров и моделей компьютерного зрения, включая: DETR (Detection Transformer), ConvNeXt и U-Net.

Несмотря на то, что модель GPT-2 показала довольно неплохие результаты после тонкой настройки, можно рассмотреть использование более современных и мощных языковых моделей, особенно если имеются высокопроизводительные вычислительные мощности, к которым, в рамках исследования, у меня не было доступа. Например, GPT-3, T5 (Text-to-Text Transfer Transformer), BART (Bidirectional and Auto-Regressive Transformers), PaLM (Pathways Language Model), а также использование LLaMA2 и недавно представленной LLaMA3.

Внедрение этих моделей и их адаптация для медицинских задач требуют дополнительных исследований и тестирования, однако они могут стать основой для создания высокоэффективных систем автоматической генерации медицинских заключений.

ЛИТЕРАТУРА

1. Рентгенография грудной клетки // Википедия : свободная энциклопедия. [Б. м.]. – URL: https://ru.wikipedia.org/wiki/Рентгенография_грудной_клетки (дата обращения: 03.05.2024). – Текст : электронный.

2. *Gupta A., Mannem P.* From Image Annotation to Image Description. – 2012. – P. 196–204.

3. *Raschka S., Liu Yu., Mirjalili V., Dzhulgakov D.* Machine Learning with PyTorch and Scikit-Learn. – Packt Publishing, 2022. – 770 p.

4. *Sarkar D.* Text Analytics with Python / Dipanjan Sarkar. – Berkeley, CA : Apress, 2019.

5. *Vaswani A., Shazeer N., Parmar N. [et al.]* Attention Is All You Need // Text : electronic. – 2017. – URL: https://arxiv.org/abs/1706.03762 (date accessed: 11.05.2024).

6. *Dosovitskiy A., Beyer L, Kolesnikov A. [et al.]* An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale // Text : electronic. – 2020. – An Image is Worth 16x16 Words. – URL: https://arxiv.org/abs/2010.11929 (date accessed: 11.05.2024).

7. *Liu Z., Lin Yu., Cao Yu. [et al.]* Swin Transformer: Hierarchical Vision Transformer using Shifted Windows // – Text : electronic. – 2021. – Swin Transformer. – URL: https://arxiv.org/abs/2103.14030 (date accessed: 11.05.2024).

8. *Radford A., Wu J., Child R. [et al.]* Language models are unsupervised multitask learners // OpenAI blog. – 2019. – V. 1. – № 8. – P. 9.
9. GPT-2 // Википедия : свободная энциклопедия. [Б. м.]. – URL: https://en.wikipedia.org/wiki/GPT-2 (дата обращения: 12.05.2024). – Текст : электронный.
10 HF Canonical Model Maintainers. gpt2 / HF Canonical Model Maintainers. – Текст : электронный. – URL: https://huggingface.co/gpt2 (дата обращения: 12.05.2024).
11. *Vasuki P.A., Kanimozhi J., Devi M.B.* Survey on image preprocessing techniques for diverse fields of medical imagery // 2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE) 2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE). – Karur, India : IEEE. – 2017. – P. 1–6.
12. *Chaki J.A., Dey N.* Beginner's Guide to Image Preprocessing Techniques // – Boca Raton : Taylor & Francis, a CRC title, part of the Taylor & Francis imprint, a member of the Taylor & Francis Group, the academicdivision of T&F Informa, plc, 2019. Series: Intelligent signalprocessing and data analysis : CRC Press, 2018.
13. *Rajpurkar P., Irvin J., Zhu K. [и др.].* CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning // 2017. – CheXNet. – URL: https://arxiv.org/abs/1711.05225 (дата обращения: 17.05.2024).

РАЗРАБОТКА СИСТЕМЫ РАСПОЗНАВАНИЯ ДОРОЖНЫХ ЗНАКОВ НА ИЗОБРАЖЕНИЯХ И ВИДЕО

Рудов В.А., Приступа А.В., Скворцов А.В.
Томский государственный университет
tsu.rudov.vladislav@gmail.com

Введение

В настоящее время вопросы обеспечения безопасности и эффективности транспортных коммуникаций являются одними из наиболее значимых и актуальных в сфере дорожного хозяйства Российской Федерации. Развитие технологий и увеличение автомобильного трафика требуют не только усовершенствованной инфраструктуры, но и эффективного управления и обслуживания дорожных систем. В данном контексте особую важность приобретает процесс технического учёта и паспортизации дорог.

Проведение данных мероприятий не только предоставляет возможность для сбора и систематизации объективной информации о техническом состоянии дорог и дорожных сооружений, но и обеспечивает основу для дальнейшего рационального планирования строительства, реконструкции, ремонта и содержания [1].

На сегодняшний день, для осуществления данных мероприятий всё чаще применяются технологии фото-видеосъёмки совместно с системами GPS и ГЛОНАСС. Однако анализ таких данных остаётся трудоёмким процессом, который зачастую выполняется вручную. Таким образом, актуальность данного исследования обусловлена совершенствованием процесса анализа подобных данных путём ускорения его выполнения и повышения эффективности.

Настоящая работа направлена на разработку единого алгоритма, последовательно решающего как задачу распознавания дорожных знаков, так и задачу их позиционирования в криволинейной системе координат исследуемой дороги [2].

Процесс распознавания дорожного знака логически можно разделить на два основных этапа:

1) детектирование области предполагаемого расположения знака;
2) классификация обнаруженного знака в соответствии с его номером по стандарту (ГОСТ [3]) и значением определяемой величины (скорости, массы, уклона и др.).

Существующие методы решения задачи детектирования знаков дорожного движения можно разделить на 3 основные категории [4]:

– эвристические алгоритмы,
– методы на основе классических методов машинного обучения,
– методы на основе глубоких свёрточных нейронных сетей (ГСНС).

Каждый из этих подходов имеет свои преимущества и недостатки. Тем не менее, для решения поставленных задач целесообразнее всего использовать методы, основанные на ГСНС. Это обосновывается следующими факторами:

1) ГСНС обладают более высокой точностью по сравнению с эвристическими и классическими методами,
2) ГСНС устойчивы к различным типам оптических искажений, возникающих в различных дорожных сценариях,
3) обучаясь на больших объёмах данных, ГСНС способны выявлять сложные патерны и признаки, характерные для различных видов дорожных знаков.

В рамках данного исследования для детектирования области расположения дорожных знаков используется ГСНС архитектуры YOLOv8 [5], а для классификации обнаруженных регионов – ГСНС архитектуры MobileNetV3 [6].

Для достоверной оценки точности полученного решения использовались реальные данные, собранные в ходе проездов по трассе мобильной дорожной лаборатории.

1. Категоризация дорожных знаков

С целью повышения эффективности работы метода детектирования, из множества всех существующих знаков дорожного движения российского образца выделены десять подмножеств, элементы которых обладают схожими визуальными (цветовыми и геометрическими) признаками. Список данных подмножеств, названных "сериями", представлен в табл. 1.

Таблица 1

Список выделенных серий знаков дорожного движения

Наименование серии	Визуальный признак	Описание признака
Ограничения (Prohibitory)		Круг с красной каймой
Предупреждения (Warning)		Треугольник с красной каймой
Информационные (Information)		Синий квадрат
Предписания (Mandatory)		Синий круг
Запреты (Stop)		Красный круг
Приоритет (Priority)		Жёлтый ромб с белой каймой
Помеха (Yield)		Перевёрнутый треугольник с красной каймой
Сервисные (Service)		Синий прямоугольник с белым наполнением
Направления (LaneDirection)		Синий прямоугольник с белыми стрелками
Отмены (Cancel)		Белый круг с чёрной чертой

Т.к. состав серий в первую очередь определяется визуальными признаками дорожных знаков, он не всегда совпадает с составом групп, описанных ГОСТ [3]. Таким образом, серия "Сервисные" включает в себя все дорожные знаки группы "Сервис", а также несколько знаков из группы "Особые предписания" с номерами по стандарту 5.16, 5.17 и 5.18.

Данный подход позволяет построить легко расширяемый алгоритм распознавания знаков, т.к. система сначала определяет серию знака на основе его визуальных характеристик, а затем классифицирует внутри этой серии.

2. Предлагаемый алгоритм

Предложенный алгоритм распознавания дорожных знаков формирует последовательный процесс обработки входного кадра и обеспечивает работу процессов обнаружения, классификации дорожных знаков, отслеживания объектов в видеопотоке, а так-

же расчёта их местоположений, выраженных километровыми отметками. На рис. 1 представлена схема работы данного алгоритма.

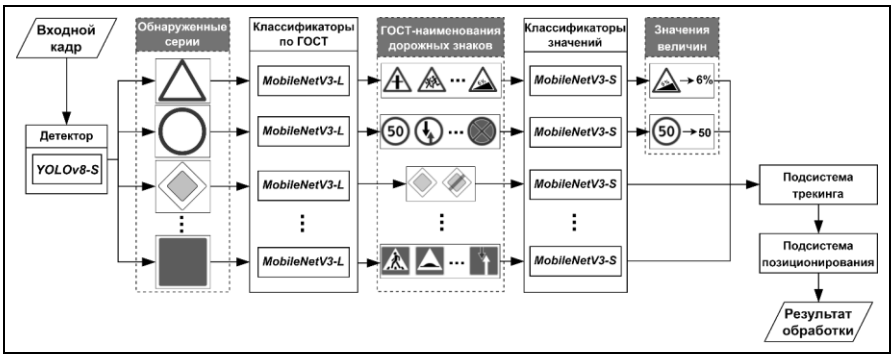


Рис. 1. Схема работы алгоритма распознавания дорожных знаков

Разработанный алгоритм состоит из пяти ключевых этапов.

1. **Детектирование и предварительная классификация регионов.** Совместно с выделением областей, предположительно содержащих дорожные знаки, осуществляется их предварительная классификация в соответствии с предложенным списком серий. Эту задачу решает специально обученная модель нейронной сети архитектуры YOLOv8–S [5].
2. **Определение номера по стандарту (ГОСТ).** В зависимости от присвоенной серии, обнаруженные дорожные знаки классифицируются в соответствии с множеством их наименований по ГОСТ [3]. За этот процесс отвечают специально обученные модели нейронной сети архитектуры MobileNetV3–Large [6].
3. **Определение значений величин.** Для знаков, определяющих какую-либо величину, производится классификация в соответствии с конечным множеством её допустимых значений. Эту задачу решает набор обученных моделей нейронной сети архитектуры MobileNetV3–Small [6].
4. **Объединение множественных обнаружений.** С целью идентификации и объединения множества обнаружений одного и того же знака в видеопотоке в один связный трек применяется подсистема трекинга.
5. **Позиционирование дорожных знаков.** На финальном этапе, производится расчёт местоположения дорожного знака в криволинейной системе координат дороги [2]. Расчёт основан на пропорциональном изменении размера ограничивающих рамок обнаруженного знака и известных координатах точек съёмки в ходе движения автомобиля.

Результатом работы системы является список распознанных дорожных знаков, содержащий требуемую информацию о каждом из них. Ключевыми характеристиками являются номер знака по стандарту и его местоположение.

Предложенный алгоритм легко расширяем. При необходимости распознавания нового дорожного знака, уже принадлежащего одной из детектируемых серий, достаточно переобучить только одну модель классификации, не внося изменений в остальные, что экономит время и вычислительные ресурсы.

3. Используемые наборы данных

В качестве основы для обучения и тестирования моделей нейронных сетей выбрана Российская база изображений дорожных знаков (Russian Traffic Sign Dataset, RTSD) [7]. Однако в процессе анализа выявлено значительное количество некорректно размеченных кадров. Ограничивающие рамки на данных кадрах имели неверные координаты расположения, указывали на отличный от действительности класс знака или вовсе отсутствовали.

По этой причине проведены мероприятия, направленные на исправление исходной разметки:

– переразмечено ~5000 кадров, содержащих ~8720 ограничивающих рамок дорожных знаков;
– исключено ~12000 кадров неудовлетворительного качества, на которых дорожные знаки были визуально неразличимы.

Дополнительно подготовлено ~7000 кадров из видеорядов дорог Томской области.

4. Детектирование дорожных знаков

В ходе подготовки набора данных для обучения и тестирования модели детектирования произведена группировка знаков в соответствии с выделенным списком серий, а также разделение на тренировочную и тестовую выборки. На рис. 2 представлено исходное распределение серий набора данных детектирования.

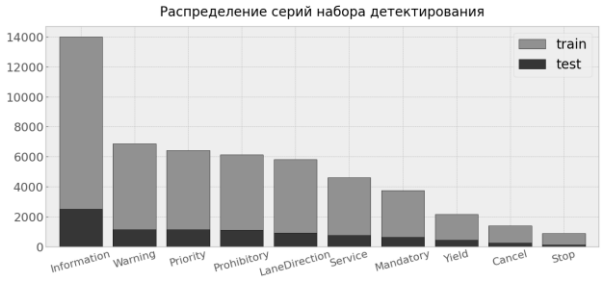


Рис. 2. Исходное распределение серий набора данных детектирования

Для устранения проблем, связанных с дисбалансом классов тренировочной выборки, применялись методы субдискретизации и передискретизации данных. Аугментация выполнялась с применением цветовых и геометрических преобразований, а также деформаций и искажений. Эти методы позволили создать множество разнообразных вариаций исходных изображений, тем самым обогатив тренировочную выборку.

В результате применения описанных методов удалось добиться однородности распределения серий тренировочной выборки. Общий объём выборки составил более 80 000 изображений. Итоговое распределение серий набора данных детектирования представлено на рис. 3.

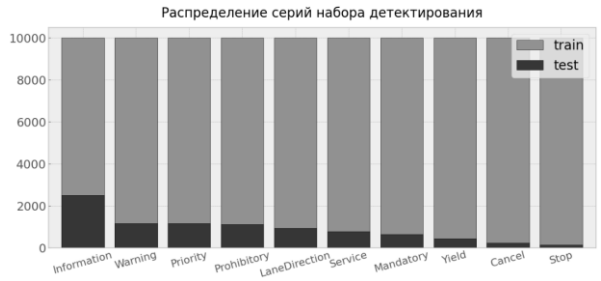


Рис. 3. Итоговое распределение серий набора данных детектирования

В ходе множества экспериментов проведено обучение моделей детектирования YOLOv8–S [5] с различными значениями гиперпараметров. Основной целью экспериментов являлась максимизация метрики средней точности (mAP). В табл. 2 представлены эмпирические значения гиперпараметров, полученные при обучении наиболее производительной модели.

Таблица 2

Используемые гиперпараметры обучения модели детектирования

Параметр	Значение
epochs	100
batch	32
imgsz	832
optimizer	AdamW
lr0	0.001
lrf	0.01
hsv_h	0.0015
hsv_s	0.7
hsv_v	0.4
degrees	5
scale	0.25
albumentations	Blur(p=0.01,blur_limit=(3,7)), MedianBlur(p=0.01,blur_limit=(3,7)), CLAHE(p=0.01, clip_limit=(1,4.0));

В табл. 3 представлены значения метрик точности модели, рассчитанных на тестовой выборке.

Таблица 3

Метрики обученной модели детектирования

Метрика	Значение
mAP@[0.5]	0.993
mAP@[0.5, 0.95]	0.885
Precision	0.989
Recall	0.986
F1-score	0.987

Таким образом, обученная модель обеспечивает довольно высокую точность работы, что позволяет успешно использовать её в реальных условиях.

5. Классификация дорожных знаков по ГОСТ

Эффективное обучение моделей классификации дорожных знаков по ГОСТ предполагает тщательную подготовку необходимых наборов данных. Алгоритм их формирования включал следующие ключевые этапы: извлечение изображений знаков из кадров RTSD, присвоение соответствующих меток классификации и разделение на тренировочную и тестовую выборки. На рис. 4 представлено исходное распределение классов одного из сформированных набора данных – "Предписания" (Mandatory).

Рис. 4. Исходное распределение набора данных "Предписания" классификации по стандарту

С целью устранения неравномерного распределения классов применялись аналогичные методы субдискретизации и передискретизации данных.

Т.к. набор RTSD не содержит все существующие виды дорожных знаков, решено воспользоваться методом синтетической генерации образцов. Этапы работы алгоритма

генерации включают подготовку векторных образцов, аугментацию с применением различных преобразований, а также добавление реалистичного контекста окружения. На рис. 5 представлена поэтапная демонстрация работы описанного алгоритма.

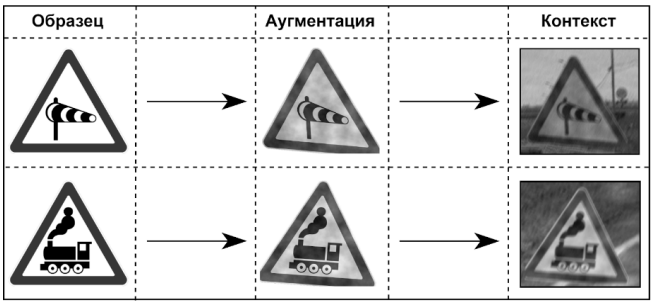


Рис. 5. Поэтапная демонстрация работы алгоритма синтетической генерации дорожных знаков

К базовым изображениям применялись различные случайные трансформации:
– аффинные преобразования и изменение перспективы,
– изменение яркости, оттенка и насыщенности цветов,
– добавление случайного градиентного шума Перлина [8].

Данные преобразования позволяют создать разнообразные вариации изображений знаков, похожие на изображения из реальных дорожных сцен. На рис. 6 представлено итоговое распределение классов набора данных "Предписания".

Рис. 6. Итоговое распределение набора данных "Предписания" классификации по стандарту

Используемая архитектура — MobileNetV3–Large [6] — взята из открытого репозитория HuggingFace [9]. Адаптация данной архитектуры включала корректировку размерности выходного полносвязного слоя, в соответствии с размерностью множества классификации.

Значения гиперпараметров обучения, такие как количество эпох, размер партии, входной размер изображений и др., были подобраны эмпирическим путём в результате множества экспериментов. Основной целью экспериментов являлась максимизация метрики точности классификации — F-меры (F1-score). Полученные значения представлены в табл. 4.

Таблица 4

Используемые гиперпараметры обучения моделей классификации по ГОСТ

Параметр	Значение
epochs	50
batch	64
imgsz	224
optimizer	AdamW
learning_rate	0,001

Параметр	Значение
loss	categorical_crossentropy

Таким образом, в результате проведённых работ, подготовлено 8 моделей нейронной сети, обеспечивающих классификацию дорожных знаков по ГОСТ для 10 выделенных серий.

В ходе оценки качества обученных моделей, рассчитаны значения ключевых метрик точности классификации: меткость (Accuracy), F-мера (F1-score) и площадь под ROC-кривой (ROC-AUC). Полученные значения представлены в табл. 5.

Таблица 5

Метрики обученных моделей классификации по ГОСТ

Наименование модели (группы знаков)	Значение метрики		
	Accuracy	**F1-score**	**ROC-AUC**
Ограничения (Prohibitory)	0.998	0.997	0.999
Предупреждения (Warning)	0.996	0.997	0.999
Информационные (Information)	0.998	0.997	0.999
Предписания (Mandatory)	0.994	0.996	0.997
Запреты-Помеха-Приоритет (Stop-Yield-Priority)	0.998	0.997	0.999
Сервисные (Service)	0.997	0.996	0.998
Направления (LaneDirection)	0.993	0.996	0.997
Отмены (Cancel)	0.998	0.995	0.997

Модели успешно справляются с задачей классификации дорожных знаков по ГОСТ и могут успешно применяться в реальных условиях с высоким уровнем эффективности и надежности.

6. Классификация значений дорожных знаков

Основываясь на распространённости дорожных знаков на дорогах общего пользования, а также на определённом в ГОСТ 32945–2014 [10] множестве допустимых значений величин, сформированы конечные множества классификации значений для 11-ти дорожных знаков. Итоговый список множеств представлен в табл. 6.

Таблица 6

Список множеств классификации значений величин

Наименование знака	Наименование множества	Величина	Множество значений
Ограничение максимальной скорости	MaxSpeedLimit	Скорость	{5; 10; 20; 30; 40; 50; 60; 70; 80; 90; 100; 110; 120; 130}
Конец ограничения максимальной скорости	EndMaxSpeedLimit		
Ограничение минимальной скорости	MinSpeedLimit		
Конец ограничения минимальной скорости	EndMinSpeedLimit		
Рекомендуемая скорость	RecommendedSpeed		
Крутой спуск	SteepDescent	Уклон	{4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14}
Крутой подъём	SteepAscent		
Ограничение массы	WeightLimit	Масса	{1,5;…;3,5}∪{4;…;35}
Ограничение массы на ось	AxleWeightLimit		{1,5;…;3,5}∪{10;…;35}
Ограничение высоты	HeightLimit	Линейные размеры	{2,0;…;5,0}
Ограничение ширины	WidthLimit		{1,8;…;3,5}

С помощью описанных методов сэмплирования и генерации синтетических данных подготовлены сбалансированные наборы для обучения и тестирования моделей. В качестве архитектуры использовалась легковесная вариация MobileNetV3–Small [6]. Эмпирические значения гиперпараметров обучения моделей представлены в табл. 7.

Таблица 7

Используемые гиперпараметры обучения моделей классификации значений величин

Параметр	Значение
epochs	35
batch	64
imgsz	224
optimizer	AdamW
learning_rate	0,0001
loss	categorical_crossentropy

В ходе оценки качества обученных моделей, рассчитаны значения метрик точности классификации. Полученные значения представлены в табл. 8.

Таблица 8

Метрики обученных моделей классификации значений величин

Наименование модели	Значение метрики		
	Accuracy	F1-score	ROC-AUC
MaxSpeedLimit	0.997	0.996	0.998
EndMaxSpeedLimit	0.998	0.998	0.999
MinSpeedLimit	0.996	0.996	0.998
EndMinSpeedLimit	0.996	0.997	0.998
RecommendedSpeed	0.999	0.999	0.999
SteepDescent	0.969	0.965	0.984
SteepAscent	0.971	0.974	0.991
WeightLimit	0.994	0.982	0.995
AxleWeightLimit	0.989	0.981	0.993
HeightLimit	0.994	0.994	0.997
WidthLimit	0.990	0.991	0.998

Модели демонстрируют высокий уровень точности и готовы к применению в реальных условиях с высоким уровнем эффективности и надёжности.

7. Позиционирование распознанных дорожных знаков

В контексте задачи инвентаризации и паспортизации дорог важно не только определить сам факт наличия объекта дорожной инфраструктуры, но и рассчитать его местоположение относительно начала дороги. Алгоритм расчёта местоположения распознанного дорожного знака основан на анализе его обнаружений на последовательности кадров и включает в себя следующие шаги:

1. Расчёт приблизительных расстояний от мест съёмки.

Для каждой пары кадров, представленной в последовательности обнаружений, производится оценка приблизительных расстояний до обнаруженного знака (D_j) по формуле $D_j = \frac{H_i \times \Delta_{ij}}{H_i - H_j}$, где D_j — расстояние от j-го кадра до обнаруженного объекта (в метрах), Δ_{ij} – расстояние между местами съёмки i-го и j-го кадров (в метрах), H_i – высота ограничивающей рамки на i-м кадре (в пикселях);

2. Расчёт множества предполагаемых позиций.

Затем рассчитываются предполагаемые позиции знака в криволинейной системе координат дороги [2] (L). Для этого к соответствующим координатам кадров p_j добавляются рассчитанные расстояния: $L = \{D_j + p_j\}$, где p_j – координаты места сьёмки j-го кадра в системе координат дороги.

3. Выбор серединного (медианного) значения.

Из множества предполагаемых координат знака выбирается медианное значение, определяющее итоговое местоположение распознанного знака (P): $P = median(L)$. Выбор медианы обусловлен устойчивостью метода к возможным экстремальным значениям (выбросам).

Таким образом, зная расстояния между кадрами и высоты соответствующих ограничивающих рамок, можно рассчитать местоположение обнаруженного дорожного знака в криволинейной системе координат дороги [2]. Однако следует учитывать, что данный метод оценивает приблизительное значение, т.к. результат может быть искажён ввиду различного рода погрешностей.

8. Оценка точности разработанной системы

Для достоверной оценки точности разработанной системы распознавания дорожных знаков, проведено её тестирование в условиях реальной дорожной среды на участке трассы Р–398 "Томск – Колпашево". Анализируемый видеоряд подготовлен в ходе проезда по трассе мобильной дорожной лаборатории и содержит 35643 кадра, снятых в светлое время суток с разрешением 1719x1719 пикселей (нестандартный размер кадра обусловлен особенностями видеооборудования). В табл. 9 представлены результаты тестирования.

Таблица 9

Результаты обработки реальных данных

Показатель	Количество
Распознанные знаки (True Positive)	597
Нераспознанные знаки (False Negative)	7
Ложные срабатывания (False Positive)	53
Общий итог	**657**

Рассчитанные значение метрик точности системы представлены в табл. 10.

Таблица 10

Итоговые метрики точности разработанной системы

Наименование метрики	Значение
Accuracy	**0.908**
F1-score	**0.952**
Precision	0.918
Recall	0.988

Дополнительно, произведены замеры времени обработки системой одного кадра с использованием различных аппаратных ресурсов: видеокарты Nvidia GeForce RTX 3060 12Gb и процессора Intel Core i5–12500. Полученное сравнение быстродействия представлено в табл. 11.

Таблица 11

Быстродействие системы

Оборудование / Этап работы	Nvidia GeForce RTX 3060 12Gb, миллисекунд	Intel Core i5–12500, миллисекунд
Детектирование	18–31	136–162
Классификация	2–8	8–32
Трекинг	1–4	1–4
Общее время	**21–43**	**129–198**

На основании результатов тестирования, разработанная система распознавания дорожных знаков обеспечивает довольно высокую точность и скорость работы и готова к практическому применению в условиях реальной дорожной среды.

Заключение

В данной работе предложен подход к распознаванию дорожных знаков, который значительно повышает эффективность технического учёта и паспортизации дорожных сооружений.

Обучена модель YOLOv8-Small [5], обеспечивающая высокую точность детектирования дорожных знаков на изображениях и видео реальных дорожных сцен.

Обучено 8 моделей MobileNetV3-Large [6], обеспечивающих эффективную классификацию дорожных знаков в соответствии с множеством наименований по стандарту (ГОСТ [3]).

Обучено 11 моделей MobileNetV3-Small [6], обеспечивающих эффективную классификацию дорожных знаков в соответствии с конечными множествами значений определяемых величин.

Предложен подход к позиционированию дорожных знаков в криволинейной системе координат дороги [2], основанный на пропорциональном изменении размера ограничивающих рамок обнаруженного знака и известных координатах точек съёмки.

Разработана система, способная распознавать 170 различных знаков дорожного движения российского образца. Оценка точности системы показала её эффективность и готовность к практическому применению в условиях реальной дорожной среды.

ЛИТЕРАТУРА

1. Венская конвенция о дорожном движении. – 1968. – 19 с.

2. *Скворцов А.В.* Геоинформатика: Учебное пособие. Томск: Изд-во Том. ун-та, – 2006. – 336 с.

3. ГОСТ Р 52290–2004. Технические средства организации дорожного движения. Знаки дорожные. Общие технические требования. – 194 с.

4. *Shakhuro V., Faizov B., Konushin A.* Rare Traffic Sign Recognition using Synthetic Training Data // Proceedings of the 3rd International Conference on Video and Image Processing (ICVIP '19). Association for Computing Machinery. – 2020. – P. 23–26.

5. Ultralytics YOLOv8 [Электронный ресурс] // URL: https://docs.ultralytics.com/models/yolov8/#overview

6. Searching for MobileNetV3 [Электронный ресурс] // URL: https://arxiv.org/abs/1905.02244

7. *Shakhuro V.I., Konushin A.S.* Russian traffic sign images dataset // Computer Optics. – 2016. – V. 40. – № 2. – P. 294 – 300.

8. Perlin Noise Improve Adversarial Robustness [Электронный ресурс] // URL: https://arxiv.org/pdf/2112.13408 (дата обращения: 12.04.2024)

9. Hugging Face – The AI community building the future [Электронный ресурс] // URL: https://huggingface.co/ (дата обращения: 12.04.2024)

10. ГОСТ 32945–2014. Дороги автомобильные общего пользования. Знаки дорожные. Технические требования [Электронный ресурс] // URL: https://irtechnologies.ru/assets/gost-32945-2014.pdf?ysclid=lw4gmhdhsp562853948 (дата обращения: 18.04.2024)

МЕТОДЫ ПОДГОТОВКИ ДАННЫХ ДЛЯ ОБУЧЕНИЯ НЕЙРОСЕТИ В ЗАДАЧЕ РЕГУЛИРОВКИ ЛОКАЛЬНОЙ ЯРКОСТИ ИЗОБРАЖЕНИЯ

Сапегин А.А.

Томский политехнический университет

aas271@tpu.ru

Введение

Одной из главных проблем изображений, полученных при естественном освещении, является неравномерное распределение яркости. Ввиду ограниченного диапазона интенсивности каждого пикселя (от 0 до 255) перепад яркостей может быть настолько мал, что не будет замечен человеком. Чаще всего такие ситуации возникают, когда камера фиксирует область с избытком света либо с его недостатком. Это может привести к потере деталей, а также затруднить обработку изображения и его анализ.

Степень потери информации зависит от светочувствительности матрицы камеры. Чем выше данный показатель, тем труднее на полученном изображении распознать объекты в светлых областях. Соответственно, чем меньше светочувствительность, тем выше потери в темных областях. Т.о., даже если у камеры имеется возможность регулировки данного показателя, велика вероятность, что в общей композиции будут одновременно присутствовать обе категории областей, в которых теряется информация. Это

означает, что изображение так или иначе будет необходимо подвергать дополнительной обработке, представляющую собой регулировку яркости и контраста. Наиболее эффективны методы контекстуальной обработки, т.е. когда учитываются определенные особенности изображения (это может быть, например, распределение яркости изображения целиком или его областей).

1. Описание анализируемого подхода

Регулировка яркости или контрастности с использованием нейросети является ярким представителем контекстуальной обработки. Главным преимуществом такого подхода перед алгоритмическими методами является имитация когнитивных функций. Иными словами, сети оперируют нечётной логикой и анализируют одновременно множество факторов в контексте друг с другом. Такой подход полезен тем, что можно сформировать структурно сложный, но, при этом, универсальный паттерн, который будет выдавать корректный результат при огромном количестве "но" и "если", которые не надо пытаться учитывать разрабочку. Наиболее распространенными являются полносвязные нейронные сети, однако для обработки изображений сверточные нейронные сети являются наиболее предпочтительным вариантом.

Сверточные нейронные сети – это специализированный тип нейронных сетей, которые обычно используются для анализа данных с пространственной структурой, таких как изображения или видео. Они обладают несколькими сверточными слоями, которые помогают распознавать паттерны и структуры в данных. Главными преимуществами перед своим полносвязным аналогом является способность автоматически извлекать признаки из входных данных без необходимости ручного определения характеристик и эффективность в работе с данными пространственной структуры, такими как изображения, где каждый пиксель имеет значения, зависящие от его контекста.

Исходя из этих преимуществ, можно предположить, что сеть можно обучить для решения задачи восстановления неравномерно освещенных изображений. Суть экспериментального подхода заключается в использовании модификации архитектуры сверточной сети Super-Resolution Convolutional Neural Network (SRCNN).

Оригинальная архитектура сети показана на рис. 1. Обработку изображения нейросетью можно условно разделить на три этапа (по числу уровней): извлечение признаков (преобразование фрагментов изображения в многомерные вектора), нелинейное преобразование (нелинейное отображение каждого многомерного вектора на другой многомерный вектор), реконструкция (получение значения яркости центрального пикселя на основе полученных признаков) [1].

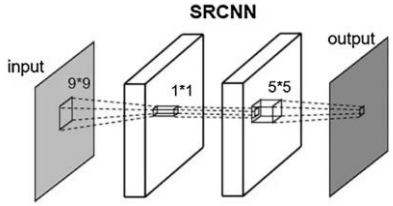


Рис. 1. Архитектура SRCNN

Оригинальная архитектура сети подразумевает обработку одного канала изображения, модификация же принимает на вход три изображения: оригинал и две его обработанные версии. В качестве методов обработки для этих двух каналов выступают Contrast-Limited Adaptive Histogram Equalization (CLAHE) [2] и Single Scale Retinex [3] вместе с CLAHE.

CLAHE является модификацией метода адаптивного гистограммного вырывания. Гистограмма изображения представляет собой график плотностного распределения яркостей пикселя. Обычно, пиксель чёрно-белого изображения может принимать 256

(от 0 до 255) значений. Т.о., для изображения строится столбчатый график, где каждый длина каждого столбца показывает, сколько писклей каждого оттенка имеется на изображении. Метод выравнивания гистограммы сводится к тому, чтобы изменить яркости пикселей таким образом, чтобы на график распределения яркостей был приближен к равномерному распределению [4]. На рис. 2 показан пример глобального выравнивания гистограммы, которое применяется ко всему изображению.

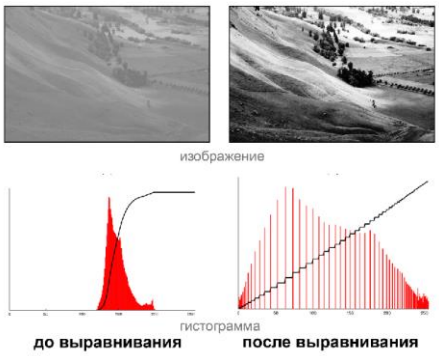


Рис. 2. Глобальное выравнивание гистограммы

Адаптивное выравнивание гистограммы является модификацией этого метода. Ее суть заключается в том, что выравнивание происходит не для изображения целиком, а для отдельных его участков (окон). Данный метод часто ругают за большой риск появления шумов. CLAHE призван устранить и эту проблему. Метод устанавливает ограничения, из-за которых обрабатываемое окно имеет меньший контраст, что дает на выходе более сглаженное изображение [2].

Подход SSR представляет собой разницу между исходным изображением в данном пикселе (x,y) и средним значением по центру этого пикселя. Для вычисления этого значения используется фильтры верхних частот (чаще всего, Гауссово размытие [3]). Подход можно выразить математически:

$$SSR(x,y) = \log_{10}\left(I(x,y)\right) - \log_{10}\left(I_{\text{m}}(x,y)\right) \quad (1)$$

где $I(x,y)$ – яркость пикселя оригинального изображения на координатах (x,y), $I_m(x,y)$ – яркость пикселя изображения после обработки фильтром.

Т.к. результат разности двух логарифмов не соответствует диапазону яркости пикселей на изображении, чаще всего прибегают к тому, что полученный массив в виде двухмерной матрицы подвергается нормализации (min-max scaling). Т.к. результат работы SSR получается довольно тусклым, для формирования третьего входного изображения также используется CLAHE (рис. 3).

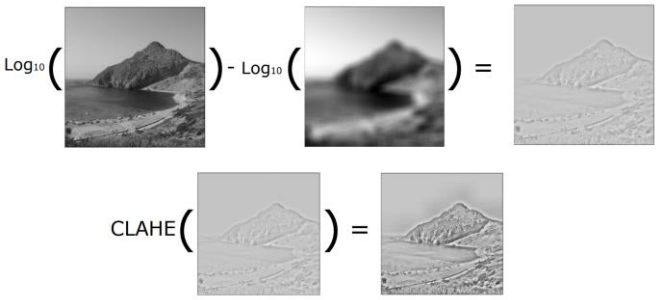


Рис. 3. Пример формирования одного из входных изображения

2. Экспериментальная часть

Для оценки результатов работы сетей была выбрана метрика Мунтеану – Роса (MR), основанная на том свойстве, что для человеческого восприятия большее значение имеет перепад яркости в соседних пикселях, чем значение яркости в каждом пикселе. В то же время учитывается, что равномерность гистограммы изображения также положительно сказывается на восприятии изображения [5].

Саму же метрику было необходимо подкорректировать путем умножения ее на модифицированную версию индекса структурного сходства (SSIM) [6]. Необходимость такого шага заключается в том, что, помимо исправления яркости в конкретной области, некоторые обученные модели имели свойство полностью удалять объемные тени объектов. Из-за этого появлялся эффект барельефа, что сильно мешало восприятию изображения. Т.о., умножение MR на измененный SSIM помогало штрафовать результат обработки за удаление объемных теней.

Целью экспериментов было выявление наиболее эффективного способа аугментации входных данных при обучении.

Рассматривались два варианта: полностью искусственно задавать слепые зоны для входных изображений и использовать изображения с реальными тенями и искусственно задавать только светлые участки. Во втором случае использовались изображения из базы данных для обучения нейросетей более сложной архитектуры по удалению теней. Т.о., на вход подавались аугментации изображений с тенями, на которые наложили засветы, а в качестве эталона – изображения без теней.

Формирование слепых зон также, в свою очередь, делится на два подхода: либо менять яркость полностью всего изображения, но в таком случае каждый раз брать именно малый фрагмент 64x64 от целого, либо прибегнуть к использованию масок. Этот подход заключался в добавлении областей случайной формы, в которых была бы слишком высокая, либо слишком низкая яркость, на исходное изображение большего разрешения.

Рассматривалась два способа по ухудшению яркости. В одном случае это было изменение среднего показателя яркости в этих областях и занижение контрастности. Во втором же случае применялась гамма-коррекция. Соответственно, в случае, когда рассматривались изображения с реальными тенями, ухудшение яркости происходило только путем его увеличения. Результаты этих сравнений результатов работы обученных т.о. сетей показаны в табл. 1.

Таблица 1

Сравнение способов задания слепых зон

Искусственные тени и засвет				**Реальные тени, искусственный засвет**			
Маска		**Все изображение**		**Маска**		**Все изображение**	
Яркость и контраст	Гамма-коррекция	Яркость и контраст	Гамма-коррекция	Яркость и контраст	Гамма-коррекция	Яркость и контраст	Гамма-коррекция
217	238	228	245	233	246	241	260

Из таблицы видны тенденции: использование реальных теней лучше, чем попытки искусственного затенения, случайные маски менее эффективны, чем большой набор малых фрагментов, а также, что ухудшение яркости путем гамма-коррекции более эффективно, чем манипуляции со средним значением и уменьшением контраста.

Помимо этого происходила проверка гипотезы о том, что можно добиться улучшения качества обработки сетью изображений путем модификации процесса обработки SSR, результат работы которого подается на вход.

Формула (1) демонстрирует общий вид подхода SSR, где Im представляет собой лишь обработанную версию изображения. И, несмотря на то, что практически всегда это размытый с помощью фильтра Гаусса оригинал, в теории может подойти любая обработка, в результате которой появляется эффект слияния и размытия границ. В ка-

честве эксперимента были выбраны 3 подхода помимо классического Гаусса, которые дают похожие эффекты: метод сегментации MeanShift, медиальный фильтр (Median Blur) и билатеральный фильтр (Bilateral Filter). Подразумевается, что на местах, где контраст между объектами высокий, на изображении после обработкой одним из этих фильтров они будут четче, чем после размытия Гаусса (рис. 4). Было выдвинуто предположение, что данное свойство должно как-то улучить обработку SSR, а следовательно, и работу сети.

Рис. 4. Пример работы методов

В табл. 2 отражены результаты тестирования подходов. Обучение каждого происходило с использованием реальных теней, изменения яркости – с помощью гамма-коррекции для большого набора малых кусков 64x64 из изображения.

Таблица 2

Сравнение способов реализации SSR

Gaussian	MeanShift	Median Blur	Bilateral Filter
260	262	258	262

Как видно из таблицы, имеются минимальные улучшения. На рис. 5 можно обратить внимание, как при использовании MeanShift и Bilateral Filter сеть пытается выделить трубу на фоне крыши.

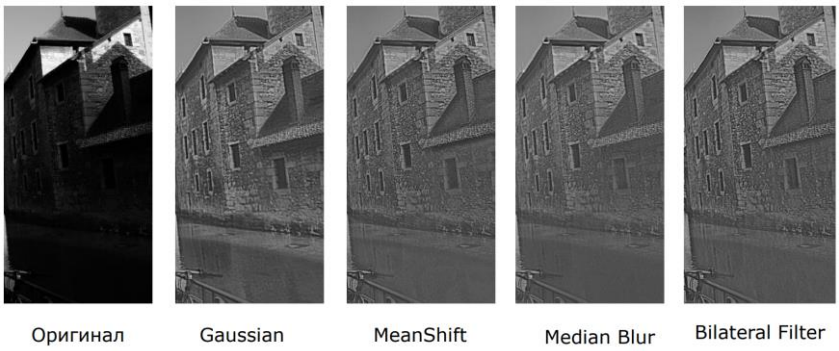


Рис. 5. Пример работы сетей для разных SSR

Заключение

В результате проведенных исследований был выявлен оптимальный подход к нормализации яркостных характеристик с помощью использования модифицированной архитектуры сверточной нейронной сети SRCNN. Данный способ позволяет добиться высокой степени различимости объектов при достаточно низком риске возникновения артефактов, свойственных изображениям после процедуры нормализации яркости. Использование датасета с настоящими тенями все еще выигрывает у подхода с полностью искусственной аугментацией данных.

Использование иных подходов помимо использования Гауссового фильтра при обработке SSR, таких как MeanShift и Bilateral Filter, также могут давать более качествен-

ный результат. Однако улучшения минимальны, в то время как Гауссово размытие вычислительно более простая операция, что, т.о., нивелирует достоинства нестандартных подходов.

ЛИТЕРАТУРА

1. *Dong C., Loy C.C., He K., Tang X.* Image super-resolution using deep convolutional networks // IEEE transactions on pattern analysis and machine intelligence. – 2015. – P. 295–307.

2. *Кокошкин А.В., Коротков В.А., Новичихин Е.П.* Модификация метода CLAHE для компенсации влияния гидрометеоров // Журнал Радиоэлектроники. – 2017. – 10.

3. *Jobson D. J., Rahman Z., Woodell G.A.* A multiscale retinex for bridging the gap between color images and the human observation of scenes // IEEE Trans. Image Process. – 1997. – V. 6. – № 7. – P 965–976.

4. *Гонсалес Р., Вудс Р.* Цифровая обработка изображений. – Москва: Техносфера, 2005. – 1072 с.

5. *Munteanu C., Rosa A.* Gray-scale image enhancement as an automatic process driven by evolution // IEEE Trans. on Systems, Man, and Cybernetics – part B: Cybernetics. – 2004. – V. 34. – № 2.

6. *Wang Z., Bovik A.C., Sheikh H.R., Simoncelli E.P.* Image Quality Assessment: From Error Visibility to Structural Similarity // IEEE Trans. Image Process. – 2004. – V. 13. – № 4. – P. 2–15.

VI. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ПРИЛОЖЕНИЕ ПО ВЫБОРУ МЕСТА ОБЩЕСТВЕННОГО ПИТАНИЯ

Басаргина А.С., Шкуркин А.С.
Томский государственный университет
basargina.nastia@yandex.ru, shkurkin@mail.tsu.ru

Введение

Современная динамика городской жизни предъявляет высокие требования к эффективности и удобству в различных сферах, включая общественное питание. С ростом числа ресторанов, кафе и других заведений выбор подходящего места для приема пищи становится не только вопросом вкусовых предпочтений, но и вопросом удобства, доступности и информированности. Определение где и что поесть может стать задачей, требующей значительных усилий, особенно в условиях городской суеты и разнообразия предложений.

Актуальность выбранной темы обусловлена рядом ключевых факторов, которые определяют запросы и потребности современного общества: рост индустрии общественного питания, требования к удобству и информированности, развитие мобильных технологий, значимость культурных и кулинарных предпочтений. Все эти факторы подчеркивают актуальность исследования, направленного на создание приложения для выбора места приема пищи. Это приложение не только упрощает процесс принятия решений в сфере общественного питания, но и соответствует современным ожиданиям и запросам потребителей в контексте удобства, информированности и культурных предпочтений.

В данной работе рассматривается создание мобильного приложения, которое не только соответствует текущим потребностям пользователей в выборе места для приема пищи, но и предлагает дополнительную ценность в виде подробной информации о блюдах и заведениях. В конечном итоге, исследование должно привести к созданию инновационного и успешного приложения, способного сделать процесс выбора и опыт потребителей в сфере общественного питания более удобным и информативным.

1. Ключевые моменты развития общественного питания

В истории развития ресторанного бизнеса в России можно выделить несколько ключевых моментов, которые существенно повлияли на его тенденции и направления развития. Появление собственных сайтов и приложений, а также переход на доставку еды стали важными этапами в этом процессе.

В начале 2000-х годов в России начался активный рост интернет-пользователей [1,2]. Стремительное развитие технологий и доступ к сети Интернет стали поворотным моментом для ресторанного бизнеса. Владельцы заведений начали осознавать важность онлайн-присутствия для привлечения клиентов и повышения лояльности. С появлением собственных сайтов ресторанов и кафе клиенты получили возможность ознакомиться с меню, условиями заказа и распорядком работы заведения в любое удобное время. Это способствовало расширению аудитории и укреплению связи с постоянными клиентами.

Особенно важным моментом в развитии ресторанного бизнеса в России был переход на доставку еды [3]. Сервисы доставки еды, такие как Delivery Club, Яндекс.Еда, Uber Eats и другие, стали популярными в конце 2000-х и начале 2010-х годов. Этот тренд изменил облик ресторанной индустрии, позволив заведениям достигать новых клиентов, не имея собственного зала для посетителей. Рестораны начали активно интегрировать свои меню в эти сервисы, а также разрабатывать собственные службы до-

ставки. Этот шаг также способствовал увеличению конкуренции в индустрии и стимулировал рестораторов к поиску новых способов привлечения и удержания клиентов.

История развития ресторанного бизнеса в России демонстрирует, как изменения в потребительском поведении и технологические инновации влияют на эту отрасль, стимулируя рестораторов к адаптации и поиску новых решений для удовлетворения потребностей своих клиентов.

2. Динамика числа заведений

В 2023 г. число заведений общепита в России [4,5] выросло на 2.15%, достигнув отметки в 326 000, как сообщил генеральный директор Infoline И. Федяков на конференции "Horeca–2024: новый ландшафт рынка". Этот рост сопровождался и увеличением доходов заведений общественного питания на 14%.

Несмотря на уход крупных иностранных компаний, начался положительный тренд еще в прошлом году. В 2018 г. в России действовало 333 000 точек общепита, а к 2019-му году их количество сократилось до 328 000. В последующие два года число заведений продолжало снижаться, достигнув 308 000 и 303 000 в 2020-м и 2021-м гг. соответственно. Однако в 2023 г. произошел рост, приведший к увеличению количества заведений до 326 000 (рис. 1).

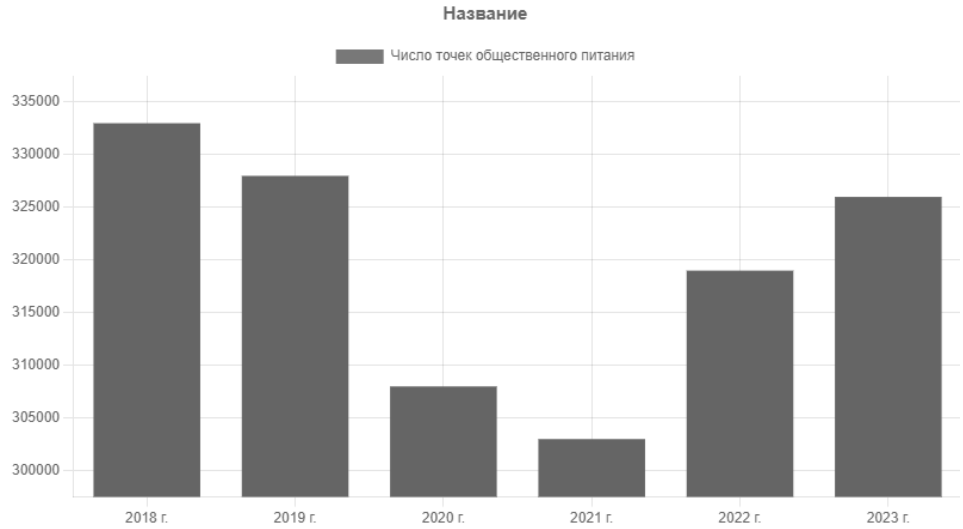


Рис. 1. Динамика численности точек общественного питания в России в 2018–2023 гг.

Прошлый год также принес изменения в структуре рынка общественного питания. Наблюдалось сокращение количества сетевых заведений традиционно прибыльных форматов, таких как суши-шопы (–21%), фастфуд (–4%) и кофейни (–5%). В то же время количество заведений формата Casual/fine dining (кафе и рестораны) увеличилось на 14%, пиццерий – на 13% и пекарен – на 8%.

После ухода Mcdonald’s сеть "Вкусно – и точка" активно начала открывать новые рестораны, добавив 111 точек к своей сети. Это стало самым высоким темпом открытия среди лидеров.

Кроме того, "Шоколадница" (включая бренды "Шоколадница", "Кофе Хауз", "Ваби Саби", Panda Express) открыла 62 новых заведения, а Dodo Brands ("Додо пицца", "Дринкит", "Донер 42") добавила 41 точку к своей сети.

Эти изменения отражают динамику развития ресторанной индустрии в России, подчеркивая рост и изменения в структуре рынка.

3. Анализ оборота

Сфера общественного питания в России становится объектом растущего интереса для различных компаний, не связанных непосредственно с ресторанным бизнесом. Вместе с традиционными рестораторами на рынок вступают ритейлеры, агрохолдинги, операторы автозаправочных станций и другие нестандартные участники. Это создает новые вызовы и возможности для развития отрасли [6].

Согласно данным Росстата и оценкам Infoline, оборот компаний в сфере общественного питания в 2023-м году продемонстрировал значительный рост. По данным Росстата, оборот увеличился на 12.3%, достигнув отметки в 2.83 трлн. р., в то время как по оценке Infoline рост составил 15%, достигнув отметки в 4 – 4.2 трлн. р. [7] (рис. 2).

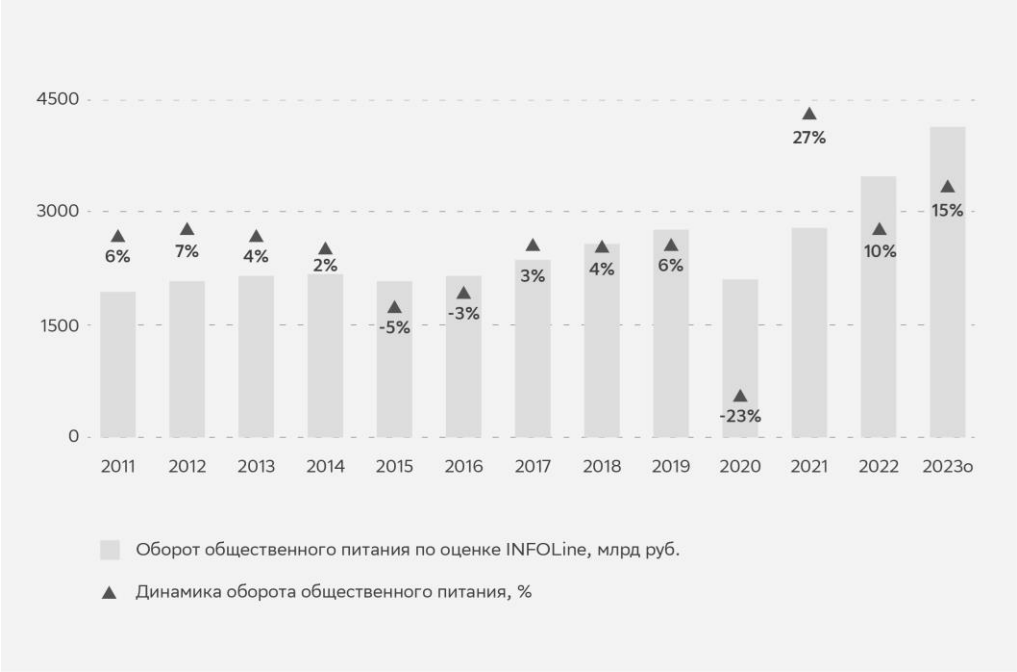


Рис. 2. Динамика оборота компаний из сферы общественного питания в России, млрд. р.

Основной вклад в рост оборота вносят заведения быстрого обслуживания (QSR) и Grab&Go, такие как бургерные, пекарни и столовые. Аналитика "Ромир" показывает, что аудитория российского ресторанного рынка за последний год выросла на 10%, а средний чек увеличился на 5%. Это свидетельствует о стремлении потребителей к быстрому и доступному питанию.

Усиливающаяся конкуренция между фастфудом и розничной торговлей приводит к тому, что супермаркеты открывают свои собственные кофейни, ресторанчики и бургерные. Ритейлеры и агрохолдинги также входят в ресторанный бизнес, диверсифицируя свою деятельность. Такие тенденции способствуют появлению новых моделей бизнеса и инноваций в сфере общественного питания.

Российские рестораторы активно внедряют цифровые технологии для оптимизации процессов обслуживания и улучшения клиентского опыта. Разработка онлайн-сервисов, использование интерактивных технологий и внедрение предсказывающей эксплуатации позволяют ресторанным компаниям эффективно адаптироваться к изменяющимся условиям рынка и удовлетворять потребности современного потребителя.

Внутренний туризм в России переживает активный рост, что стимулирует развитие ресторанного сектора. По словам С. Миронова, прирост туристического потока напря-

мую влияет на спрос на рестораны в туристических центрах, создавая благоприятные условия для их развития.

С увеличением туристического трафика и строительством новых дорог возрастает спрос на придорожные заведения общественного питания. Н. Попов отмечает, что модульные рестораны представляют собой эффективное решение для быстрого открытия точек питания в удаленных районах. Это позволяет рестораторам оперативно реагировать на изменения в рыночной ситуации и быстро расширять свою сеть.

Анализ оборота общественного питания в России позволяет выявить ключевые тенденции и факторы, влияющие на развитие отрасли. Повышение спроса на быстрое и доступное питание, увеличение конкуренции и внедрение инноваций в цифровую сферу будут определяющими факторами развития ресторанного бизнеса в ближайшей перспективе.

4. Описание разрабатываемого приложения

Цель разрабатываемого приложения: обеспечить пользователям быстрый и удобный доступ к информации о ресторанах, кафе, закусочных и других заведениях общественного питания, а также помочь им выбрать оптимальное место для приема пищи в соответствии с их предпочтениями и потребностями.

Основные функции приложения:

– *Поиск заведений.* Поиск по названиям блюд и ингредиентам, включая фильтры по кухням, ценовым категориям, рейтингам и фильтры по ингредиентам блюд.
– *Информация о блюдах.* Представление информации о блюдах, включая калории, состав и цену блюда. (Информативная страница для каждого блюда: в поисковике будет видна краткая информацию блюда, а на странице блюда будет видна подробная информация, включая состав, калорийность). В краткой версии видны название блюда, цена, адрес заведения и кнопка "Подробнее".
– *Интеграция с картами.* API карта для визуального отображения расположения заведений, интеграции с навигацией.
– *Регистрация профиля.* Создание персональной учетной записи для хранения информации.
– *Отзывы и рейтинги.* Пользователи могут оставлять отзывы о посещенных заведениях и ставить им оценки, что поможет другим пользователям принять более обоснованное решение.
– *Список избранного/любимого.* Добавление в избранное для сохранения информации о понравившемся блюде/заведении.
– *Информативная страница заведений.* Представление информации о заведении: график работы, меню, адрес, дополнительные ссылки на сайт заведения/социальные сети.
– *Рекомендательная система по предпочтениям.* На главной странице приложения отображаются рекомендации на основе списка избранного, составленного пользователем или на основе поисковых запросов.

5. Дизайн приложения

Проектирование пользовательского интерфейса [8] является одним из ключевых этапом в разработке любого мобильного или веб-приложения. Основная цель этого этапа – создать удобный и интуитивно понятный интерфейс, который обеспечит положительный опыт взаимодействия пользователей с приложением.

Для создания эффективного дизайна приложения необходимо учитывать следующие требования:

– *Удобство использования.* Интерфейс должен быть интуитивно понятным и легким в навигации, чтобы пользователи могли быстро находить нужную информацию.

– *Эстетическая привлекательность.* Визуальное оформление должно быть современным и привлекательным, чтобы привлекать пользователей и удерживать их внимание.
– *Функциональность.* Все функции приложения должны быть легко доступны, а взаимодействие с приложением – максимально простым и понятным.
– *Адаптивность.* Дизайн должен корректно отображаться на различных устройствах и экранах разных размеров.

Главный экран приложения (рис. 3) должен предоставлять пользователям возможность поиска заведений и блюд по ключевым запросам. Основные элементы главного экрана включают:

– *Поисковая строка.* Позволяет пользователям вводить ключевые слова для поиска заведений или блюд.
– *Фильтры.* Предоставляют возможность фильтрации результатов по различным параметрам, таким как цена, тип кухни, местоположение, рейтинг, ингредиенты блюда.
– *Категории.* Разделы, такие как "Пицца", "Роллы" и т.д., "Рекомендуемые блюда" помогают пользователям ориентироваться в приложении.

Рис. 3. Дизайн главной страницы приложения

Заключение

Разработка приложения для выбора места общественного питания демонстрирует важность и востребованность цифровых решений в современном обществе. В условиях быстрого развития технологий и растущих ожиданий потребителей, такое приложение не только удовлетворяет текущие потребности пользователей, но и открывает новые возможности для бизнеса. Продолжение работы над проектом и его дальнейшее развитие будут способствовать улучшению сервиса в индустрии общественного питания и повышению удовлетворенности клиентов.

Разработка данного приложения имеет несколько значимых аспектов для индустрии общественного питания:

– *Повышение конкурентоспособности.* Приложение помогает заведениям общественного питания привлекать новых клиентов и удерживать существующих за счет улучшенного пользовательского опыта.
– *Удобство для пользователей.* Пользователи получают инструмент, который значительно упрощает процесс поиска и выбора места для приема пищи, экономит время и обеспечивает доступ к актуальной информации.
– *Стимулирование инноваций.* Внедрение цифровых технологий в ресторанный бизнес стимулирует заведения к инновациям и улучшению качества обслуживания.

ЛИТЕРАТУРА

1. *Ткаченок М.А.* Анализ влияния электронного бизнеса на ресторанный бизнес // 2012. – URL: https://scienceforum.ru/2012/article/2012002320

2. *Магомедов М.Г.* Применение интернет-технологий в сфере общественного питания // Молодой ученый. – 2016. – № 27.2 (131.2). – С. 20-21.
3. Активно развивающийся рынок доставки продуктов и готовой еды продолжает набирать обороты // 2019. – URL: https://www.dp.ru/a/2019/03/28/Appetiti_dostavki_rastut
4. Число работающих точек общепита в России почти вернулось к допандемийным показателям // 2024. – URL: https://retailer.ru/chislo-rabotajushhih-tochek-obshhepita-v-rossii-pochti-vernulos-k-dopandemijnym-pokazateljam/
5. За два года ресторанов, кафе и баров в России стало больше почти на 10% // 2024. – URL: https://rg.ru/2024/02/24/za-dva-goda-restoranov-kafe-i-barov-v-rossii-stalo-bolshe-pochti-na-10.html
6. Как меняется российский ресторанный рынок // 2024. – URL: https://sber.pro/publication/svoi-produkti-konkurentsiya-v-onlaine-i-modulnii-fastfud-kak-menyaetsya-rossiiskii-restorannii-rinok/
7. Розничная торговля и общественное питание // 2024. – URL: https://rosstat.gov.ru/statistics/roznichnayatorgovlya
8. *Скотт Б., Нейл Т.* Проектирование веб-интерфейсов. – 2010.

РАЗРАБОТКА И ИНТЕГРАЦИЯ ДОПОЛНИТЕЛЬНЫХ МОДУЛЕЙ ДЛЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ ОРГАНИЗАЦИИ ВНЕУЧЕБНОЙ ДЕЯТЕЛЬНОСТИ

Зенкина Е.С., Моисеев А.Н.
Томский государственный университет
lizazenkina@ya.ru

Введение

"Информационная система поддержки воспитательной деятельности обучающихся" – система, которая разрабатывалась для муниципального бюджетного общеобразовательного учреждения [1]. Программный комплекс для сопровождения внеучебной деятельности – продолжение данной работы, призванное сделать систему доступной для использования не только учебными заведениями среднего образования, но также и для использования высшими учебными заведениями. Данная работа была поделена на две части.

Данная часть отвечает за внедрение в имеющуюся систему новых модулей, которые позволили бы осуществлять задачи, уникальные для высших учебных заведений. Кроме того, новые модули должны расширять часть имеющегося функционала, реализованного для школ, но требующего расширения для вузов, а именно – функций администратора и функций составления отчетов.

Система должна иметь блок, отвечающий за расселение студентов в общежития, где должна быть возможность обрабатывать данные о расселении, оперировать необходимыми документами и т.д. Это должно упростить работу с данными о расселении в общежития. Аналогичный блок должен быть реализован для контроля над оказанием материальной помощи. Также должен быть реализован блок конструктора отчетов для более гибкой работы над отчетами в сравнении с уже имеющейся реализацией. Блок администрирования должен быть расширен исходя из необходимости внедрения в систему функционала, уникального именно для администратора системы в части вузов.

Начальный этап разработки данной системы был представлен в [2].

1. Проектирование

В модель предметной области и схему базы данных были добавлены элементы, относящиеся к оказанию материальной помощи и расселению в общежития. Новые сущности, добавленные в модель предметной области и в схему базы данных, представлены на рис. 1 и 2 соответственно. Часть, которая относится к расселению в общежития – это договор найма жилого помещения, соглашение к договору найма, комната, общежитие. Часть, которая относится к оказанию материальной помощи – заявление на оказания материальной помощи, основание, необходимый документ, первоочередное право на заселение. Также на схеме отражены новые роли: "Ответственный за оказание материальной помощи", "Ответственный за расселение", "Администратор ВУЗа".

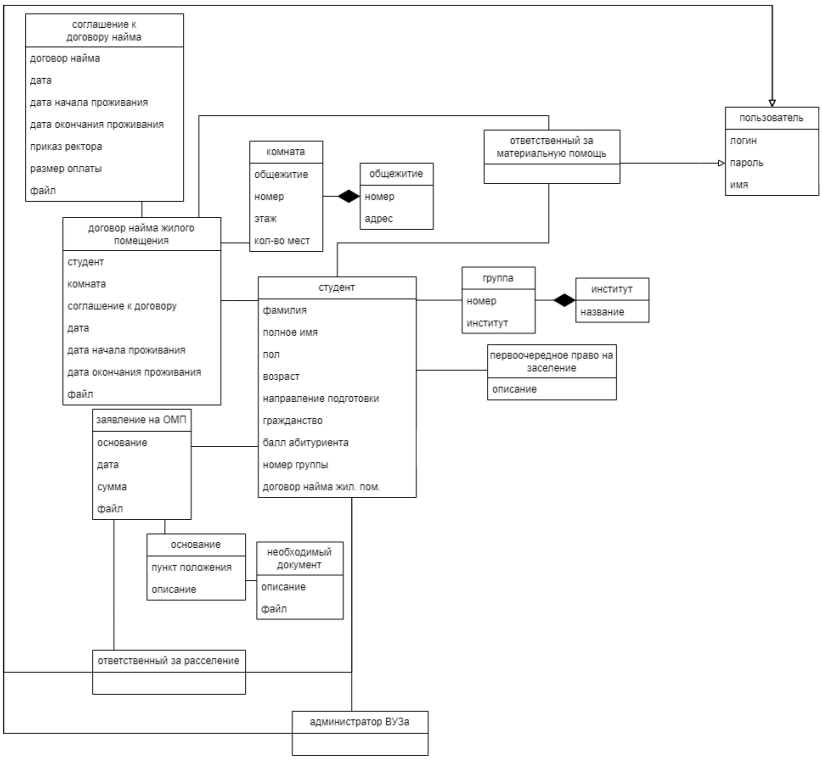


Рис. 1. Расширенная модель предметной области: часть, соответствующая добавленным в систему элементам

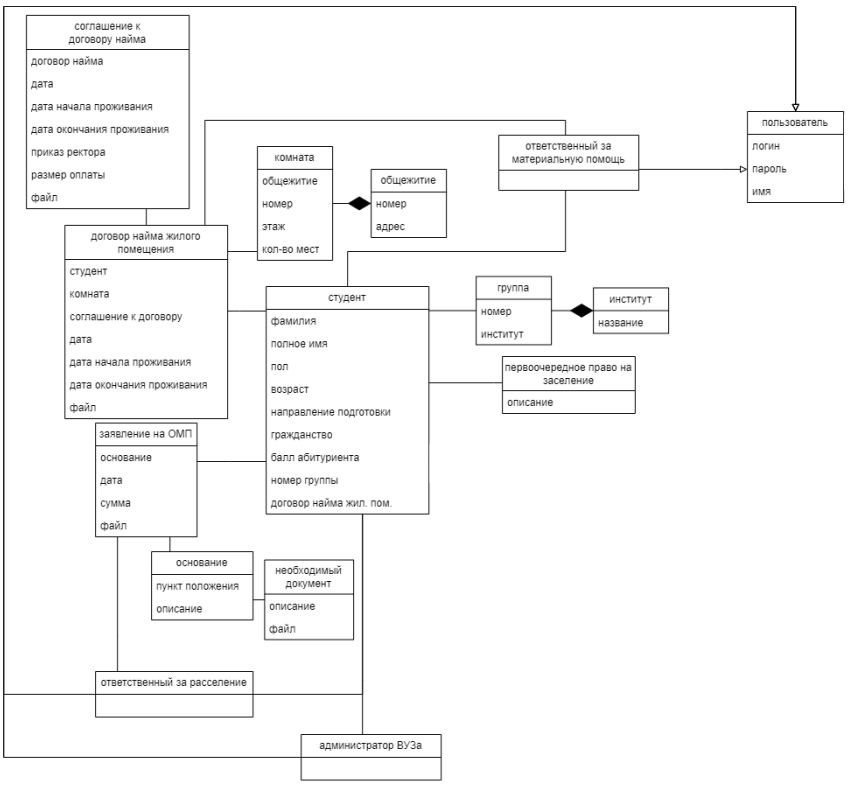


Рис. 2. Расширенная схема базы данных: таблицы, соответствующие добавленным в систему элементам

Кроме того, в систему были добавлены элементы, относящиеся к конструктору отчетов. Один из этих элементов – атрибуты, по каждому из которых различными способами составляются отчеты. Эти способы в дальнейшем называются критериями. Новые элементы, введенные в модель предметной области и схему базы данных, представлены на рис. 3 и 4 соответственно.

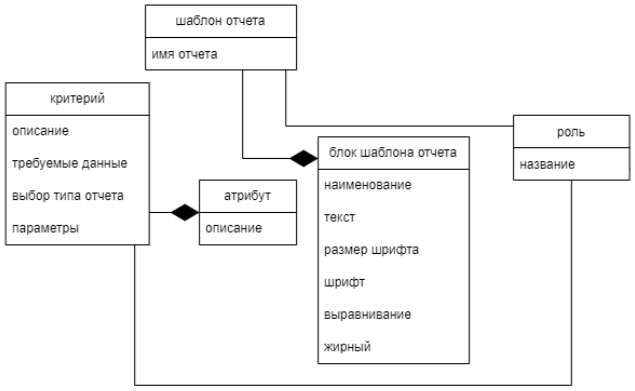


Рис. 3. Модель предметной области. Добавленные в систему сущности

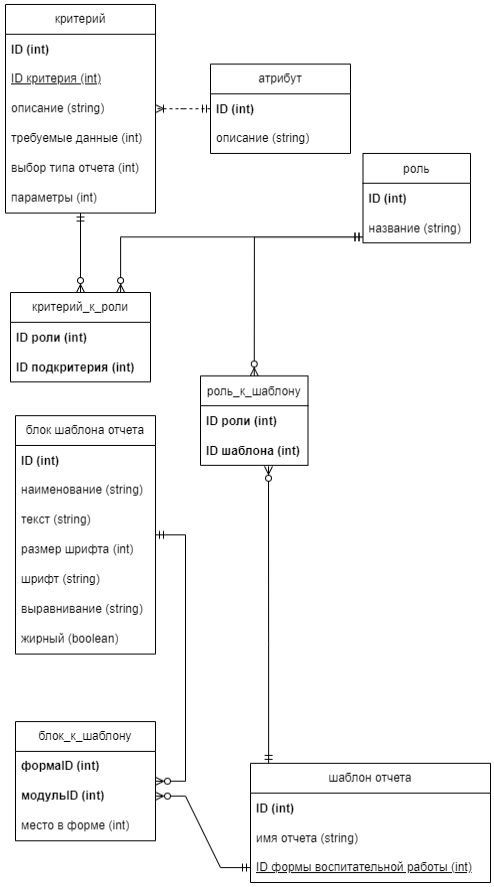


Рис. 4. Схема базы данных. Добавленные в систему таблицы

Одна из основных функций системы – создание отчетов с помощью конструктора отчетов. С целью более подробно рассмотреть процесс создания отчета пользователем, была построена схема взаимодействий для роли "Администратор ВУЗа", реализующей

вариант использования сценария, отвечающий за создание отчета. Схема взаимодействий изображена на рис. 5.

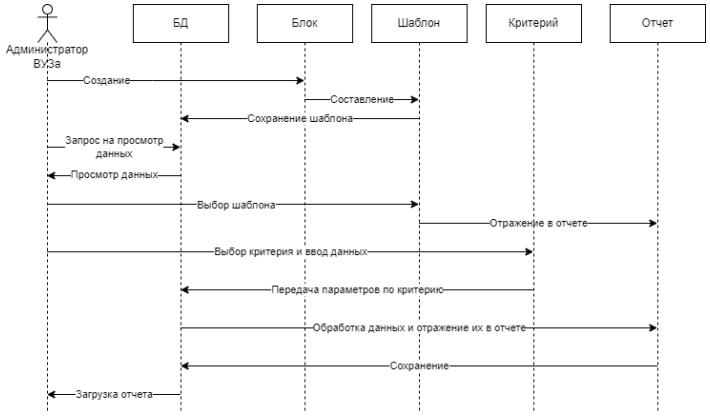


Рис. 5. Схема взаимодействий. Сценарий создания отчета для роли "Администратор ВУЗа"

Пользователь на странице для работы с отчетами создает блоки, из которых затем составляет шаблон. После составления шаблон сохраняется в базу данных. Пользователь может обратиться к базе данных и просмотреть существующие в системе отчеты и критерии, после чего выбрать шаблон (который позже будет отражен в тексте отчета) и критерий. Для критерия необходимо настроить параметры. Далее эти параметры передаются на сервер, где в соответствии с ними и с запросами, соответствующими критерию, будет произведен выбор элементов из базы данных. После этого полученные данные передаются клиенту, где вместе с шаблоном отражаются в получившемся отчете. Отчет сохраняется в базе данных, затем загружается на компьютер пользователя в виде файла.

2. Реализация функций контроля за расселением в общежития и оказанием материальной помощи

Страница контроля за оказанием материальной помощи изображена на рис. 6.

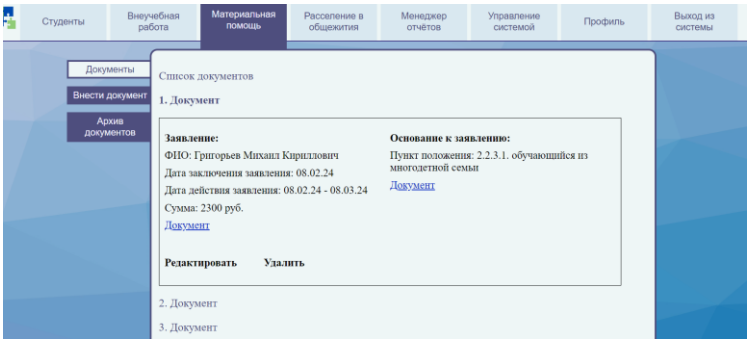


Рис. 6. Страница контроля за оказанием материальной помощи

Вкладки "Материальная помощь" и "Расселение в общежития" похожи. В обоих случаях по первой боковой вкладке производится просмотр информации по существующим документам. Здесь также можно в модальном окне посмотреть изображения документов. Документы отсортированы в системе по дате создания. Как только дата окончания действия документа становится меньше текущей даты, элемент больше не отображается на данной вкладке. По вкладке "Внести документ" производится внесение нового документа в систему. Ключевые данные документа вносятся пользователем самостоятельно. Загрузка файлов необходима для того, чтобы эти данные подтвержда-

лись документом и при необходимости была возможность их подтвердить. Без загрузки документов невозможно внести данные этих документов в систему. Вкладка "Архив документов" содержит документы, дата окончания действия которых меньше текущей даты. С точки зрения обращения к базе данных [3], отображение действующих документов отличается от вкладки архива документов только тем, что на одной отображаются документы до определенной даты, а на другой – после определенной даты. Документы в архиве редактировать нельзя. Данные архива документов удаляются из системы по прошествии определенного промежутка времени.

3. Конструктор отчетов

В изначальной системе пользователь мог выбрать несколько отчетов из предложенных. В новой версии необходимо было доработать [4] данную функцию: расширить функционал в виде добавления новых наборов данных, связанных с высшим учебным заведением; дать пользователю возможность самостоятельно составлять документ на основе шаблонов, динамически добавляемых в систему.

Атрибуты и критерии – статичные сущности, существующие в системе изначально. Каждый критерий имеет свой, заранее прописанный, запрос к базе данных, и итоговый результат у одного запроса при одном и том же состоянии базы данных может поменяться только при изменении входных данных, список которых у каждого критерия также задан заранее. В процессе обработки данных обращение к серверу осуществляется через fetch-запросы. Данные, подающиеся на вход серверу, могут отличаться не только значениями, но и количеством и типом этих значений. После получения данных осуществляется их обработка. Запросы для каждого критерия составлены заранее, динамически меняются только значения в этих запросах. В итоге сервер возвращает данные (например, числовые, если речь идет о подсчете), а на клиенте происходит отражение этих данных удобным для пользователя способом.

4. Установка уровней доступа к системе по ролям

"Администратор" обладает самыми широкими полномочиями, ему доступны все возможные вкладки. Функции, уникальные для школы, остались, но "Администратор ВУЗа" не сможет обратиться к ним через доступный ему пользовательский интерфейс.

"Ответственный за материальную помощь" – роль, которой доступны вкладки "Студенты", "Материальная помощь", "Менеджер отчетов", "Профиль".

"Ответственный за расселение" – роль, которой доступны вкладки "Студенты", "Расселение в общежития", "Менеджер отчетов", "Профиль". Ответственный за материальную помощь и ответственный за расселение, кроме того, во вкладке "Студенты" могут пользоваться только функциями, предусмотренными их ролями, а именно – добавлять, редактировать и удалять информацию, связанную с материальной помощью или расселением соответственно. Таким же образом происходит работа с менеджером отчетов – пользователям доступно создание отчетов по атрибутам, предусмотренным только их ролями.

5. Детализированные права доступа к системе

Кроме доступа ко вкладкам, в системе существует также доступ к критериям и шаблонам. Например, "Администратору" доступны все критерии и шаблоны, в то время как для новых ролей установлены ограничения, поэтому им недоступна часть отчетов. Изначально существующие уровни доступа ролей к критериям поменять невозможно, но возможно установить уровни доступа в необходимом порядке к любым критериям для новой роли, созданной в рамках системы. Шаблонов в системе изначально нет, но при их создании можно сделать их доступными для всех ролей или только для определенных. Также уровни доступа реализованы на уровне отчетов. Администратору доступны все сформированные отчеты в системе. Остальным ролям, изначально суще-

ствующим в системе, отчеты, касающиеся внеучебной работы, недоступны. На уровне документов доступ организован следующим образом: всем пользователям с ролью "Ответственный за материальную помощь" доступны документы, относящиеся к материальной помощи, и всем пользователям с ролью "Ответственный за расселение" – документы, относящиеся к расселению в общежития. Функция удаления элементов из архива доступна только администратору.

6. Создание новых ролей и установка уровней доступа

Администратор обладает возможностью создать новые роли для системы. Это нужно для того, чтобы при необходимости сделать процесс контроля над внеучебной деятельностью более гибким. При создании роли можно настроить уровни доступа в нескольких разделах: основной раздел, внеучебная работа, материальная помощь, расселение в общежития, управление системой. Кроме возможности создания, редактирования и удаления ролей, на данной вкладке возможен просмотр пользователей с определенными ролями и назначение пользователей на роли. Также в рамках этой вкладки происходит установка временного промежутка, по истечении которого информация, находящаяся в архиве, автоматически уничтожается. Роли, изначально существующие системе, не подлежат редактированию и удалению.

Заключение

В результате работы были выполнены все поставленные задачи:

– Скорректированы существующие и добавлены новые модели проекта в соответствии с изменениями.
– Добавлены специфические для высших учебных заведений роли.
– Реализованы функции контроля за расселением в общежития и функции контроля за оказанием материальной помощи.
– Реализован пользовательский интерфейс в частях системы, специфических для высших учебных заведений, в соответствии с ролями пользователей.
– Установлены уровни доступа к системе в соответствии с ролями.
– Реализован конструктор отчетов.

В итоге текущая функциональность разработанной программной системы позволяет использовать ее в высших учебных заведениях. Кроме того, система стала более гибкой в использовании. В данный момент система готова к процессу внедрения.

ЛИТЕРАТУРА

1. *Игнатьев В.Д., Зенкина Е.С., Моисеев А.Н., Кравец А.А.* Начальный этап разработки информационной системы поддержки воспитательной деятельности в учебном заведении // Труды Томского государственного университета. – Т. 308. Серия физико-математическая: Математическое и программное обеспечение информационных, технических и экономических систем : материалы X-й Международной молодёжной научной конференции. Томск, 26–29 мая 2023 г. / под общ. ред. И.С. Шмырина. – Томск : Издательство Томского государственного университета, 2023. – С. 149–155.

2. *Зенкина Е.С., Игнатьев В.Д., Спиридонова А.А.* Информационная система организации воспитательной деятельности // Фундаментальные и прикладные исследования в физике, математике и информатике. Материалы симпозиума в рамках XVII (XLIX) Международной научной конференции студентов, аспирантов и молодых ученых. – Вып. 23. – Издательство: Кемеровский государственный университет (Кемерово), Кемерово, 2022. – С. 135–138.

3. *Вайнберг П.Н., Грофф Д.Р., Оппель Э. Дж.* SQL: Полное руководство. 3-е изд. пер. с англ. – М.: ООО "И.Д. Вильямс", 2015. – 960 с.: ил.

4. *Кантелон М., Мек Б., Янг А.* Node.js в действии. 2-е изд. пер. с англ. – СПб.: Питер, 2018. – 432 с.: ил.

МЕТОДЫ ПРОЕКТИРОВАНИЯ ОТКАЗОУСТОЙЧИВЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Зоркин А.С.
Томский государственный университет
somnoynadno@stud.tsu.ru

Введение

Эксплуатация и развертывание приложений в промышленном окружении всегда сопровождается определенными рисками. Например, сервис может не выдержать резкого повышения нагрузки, в ходе его работы может быть внезапно выявлен критичный дефект или во время проведения плановых работ исполнитель может случайно совершить ошибку, приведя систему в неработоспособное состояние.

Эти и многие другие причины могут впоследствии привести к крупным сбоям и длительной недоступности бизнес-процессов, что может иметь разного рода последствия – от простого недовольства пользователей и репутационных потерь до финансовой катастрофы предприятия. В настоящее время эта проблема стала особо актуальной по причине того, что информационные технологии являются одной из самых больших инвестиций человечества, и длительные простои могут поставить под угрозу не только финансовые показатели компаний, но и жизни людей, если речь идет о системах здравоохранения. К тому же не так много исследований предлагают системный подход к обеспечению надежности.

Поэтому целью данной работы становится выявление и описание методов проектирования отказоустойчивых информационных систем с использованием аппарата управления рисками, который может помочь построить работу над надежностью более систематически.

1. Анализ рисков разработки ПО

Риск – это возможная реализация некоторого неблагоприятного события. В нашей предметной области под таким событием понимается сбой информационной системы, вследствие которого страдает ее доступность или целостность хранимых данных.

Обратимся к классическому циклу разработки ПО, состоящему из семи стадий (рис. 1).

Рис. 1. Риски возникновения сбоев на цикле разработки ПО

Зафиксируем здесь важный тезис: пусть сбои обычно происходят на двух последних этапах цикла разработки, их причины часто кроются в самых первых. Например, если в ходе эксплуатации была выявлена критическая неисправность, её появление может

быть вызвано тем, что QA-инженер не выявил ее на этапе тестирования, а архитектор не заложил способы снижения последствий в архитектуру на этапе проектирования системы.

Триггером сбоя в разобранной выше ситуации выступает выпуск новой версии приложения и срабатывание в ней дефектной функциональности. Триггер у сбоя всегда один, в то время как причин может быть множество. Основываясь на статистических и научных исследованиях [1,2], можно выделить 8 основных причин сбоев. К ним относятся ошибки в аналитике и контрактах, некачественная архитектура системы, некачественный программный код, некачественные процессы автоматизации, проблемы в коммуникации сотрудников, ошибка в ходе ручного процесса, воздействие других сервисов, невозможность доработки системы.

Некачественно спроектированное и реализованное в программном коде приложение очень сложно сделать отказоустойчивым. Распространенным ложным убеждением является то, что нанятый инженер по надежности сможет сделать любой программный компонент высокодоступным. На деле, если программный компонент не включает в себя код, позволяющий ему переживать некоторого рода отказы, или содержит слишком большое количество программных дефектов, то инженер вряд ли сможет повлиять на количество сбоев, которые из-за этого возникают, однако он может существенно снизить время реагирования и исправления инцидентов.

Основным последствием каждого сбоя является то, что для пользователя будет нарушена доступность, конфиденциальность или консистентность его данных, либо то, что он не сможет в нужный момент воспользоваться конкретной услугой, предоставляемой системой. Наибольшее влияние несут сбои, приводящие к разрушению или нарушению консистентности хранимых данных, по той причине, что их восстановление может занять длительное время, а в крайних случаях потерянные данные и вовсе могут быть не восстановлены. Это прямые финансовые и репутационные риски: клиенты полностью доверяют данные выбранной компании и ожидают, что их данные будут гарантированно защищены от повреждения или потери.

Нарушение конфиденциальности данных обычно имеет меньшее влияние по той причине, что большинству данных, за исключением персональных и платежных, зачастую не нужна высокая конфиденциальность. Защита критических данных дополнительно регулируется на законодательном уровне, поэтому без должного уровня защищенности информационная система может быть не допущена к эксплуатации регулятором, что значительно снижает риски, связанные с нарушениями информационной безопасности.

Временная недоступность данных или бизнес-процессов, пусть и является неприятной для клиентов и также ведет к репутационным и финансовым потерям, на самом деле наименее критична с точки зрения возможных последствий. Большинство кратковременных сбоев клиенты даже не замечают, потому что они неотличимы от проблем с интернет-соединением, которые довольно часто сопровождают рядовых пользователей.

Исходя из тезисов выше, основные триггеры сбоев можно разложить по матрице рисков таким образом, что наиболее приоритетными для минимизации и устранения будут являться риски повреждения или потери данных, т.к. они имеют немалую вероятность и наиболее опасные последствия, в то время как прочие риски будут менее существенными (рис. 2).

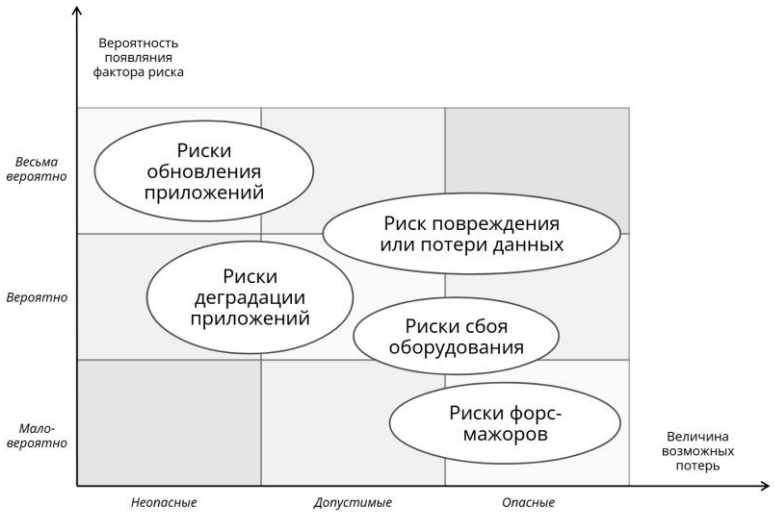


Рис. 2. Оценка рисков эксплуатации по матрице вероятности и последствий

Опираясь на частоту возникновения сбоев с определенными причинами и триггерами, а также оценив влияния и возможные последствия от большинства сбоев, мы можем построить некоторую универсальную риск-стратегию, подходящую для большинства типовых проектов в области ИТ (рис. 3).

Рис. 3. Визуализация риск-стратегии в виде пирамидальной диаграммы

Фундаментом для любой надежной информационной системы являются процессы резервного копирования, мониторинга и реагирования на инциденты. Без их наличия двигаться дальше просто нецелесообразно, потому что при отсутствии мониторинга любой сбой рискует быть попросту незамеченным, а без резервного копирования будет невозможно восстановить данные по состоянию на время до возникновения инцидента.

После того как система будет застрахована от потери или повреждения данных, стоит переходить к обеспечению процессов безопасной поставки новых версий приложений и автоматизации процессов сборки, тестирования и развертывания. Чем чаще происходит выпуск новых версий, тем выше вероятность появления ошибок в ходе обновления. Большую роль в обеспечении надежности играет наличие качественного процесса тестирования, однако даже при его наличии система несет риск появления ошибок в ходе выполнения работ и ручных действий. Поэтому ключевой задачей здесь является создание инструментария для быстрого отката релиза или выведения трафика с пострадавшей части системы.

Следующими шагами к улучшению метрик отказоустойчивости является ликвидация любых возможных точек отказа и внедрение механизмов плавной деградации. Это необходимо для снижения рисков возникновения каскадных сбоев, когда из-за недоступности одного компонента может пострадать множество других.

Финальным этапом развития является воспитание SRE-культуры (от англ. Site Reliability Engineering) внутри команды, которая заключается в том, что каждый ее участник знает ценности создания надежного ПО и соблюдает некоторые практики, которые позволяют это поддерживать [3]. К таким практикам относятся соблюдение регламентов, проведение дежурств, поддержание культуры создания постмортемов, а также высокое внимание к нефункциональным требованиям системы, касающимся ее производительности, сопровождаемости и масштабируемости.

2. Подходы к построению отказоустойчивых сервисов

Одной из основных задач архитекторов и разработчиков программного обеспечения является снижение единых точек отказа таким образом, что каждая часть системы некоторым образом резервируется: либо в архитектуру добавляются новые равноценные программные компоненты, позволяя системе распределять нагрузку между ними, либо вводится дополнительная логика выполнения процессов, позволяющая пережить временную недоступность отдельных узлов системы.

Такого рода паттерны и практики уже широко известны в кругах разработчиков и архитекторов: механизм их работы и специфика применения описаны в [4–7].

Классическим примером структурного резервирования является внедрение в инфраструктуру балансировщиков нагрузки, распределяющих входящий трафик по нескольким репликам одного приложения (рис. 4). В случае сбоя отдельно взятого узла, балансировщик нагрузки способен временно вывести с него трафик, ожидая его восстановления. Многие алгоритмы балансировки способны учитывать текущую загрузку сервисов во время принятия решения о маршрутизации запросов.

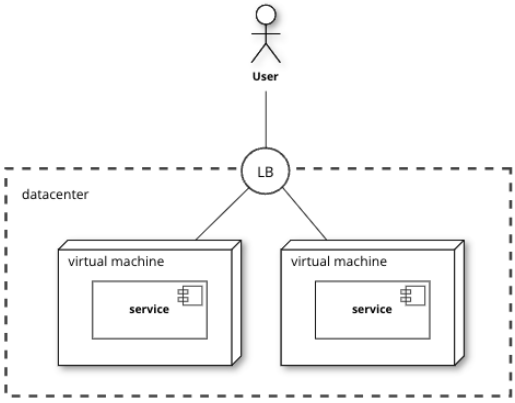


Рис. 4. Балансировка нагрузки на две реплики сервиса в пределах одного датацентра

Распределенная архитектура и асинхронное взаимодействие между сервисами значительно повышают отказоустойчивость системы, т.к. в них уменьшается кол-во потенциальных точек отказа системы. Однако асинхронная обработка запросов подходит не для каждого бизнес-процесса, а распределенные хранилища данных создают дополнительные затраты в обслуживании. В частности, следствием CAP-теоремы [8] является невозможность обеспечения единовременных свойств согласованности, доступности и устойчивости к разделению для распределенных систем хранения данных, что стоит учитывать на этапе проектирования системы.

Методы временного (использование резерва времени) и функционального (методы альтернативной логике) резервирования [9] позволяют повысить эффективность работы синхронных запросов для сервиса, и они реализуются путем усложнения логики работы сервиса путем внедрения ограничений на время обработки запроса, ограничений на предельное кол-во вызовов метода, дополнительных попыток вызова метода (а также паттерна "Переключатель"), механизмов проверки жизнеспособности сервиса (англ. healthcheck).

Плавная деградация сервиса намного предпочтительней полной недоступности сервиса. Она может быть реализована путем отключения части функциональности, не являющейся критичной для обеспечения работоспособности основных бизнес-процессов, приоритизацией пользовательских запросов или ухудшением качества генерируемого сервисом контента. Во многих системах может быть допустимо использование данных из кэша или реализация fallback-сценариев (рис. 5), обработка запросов по которым наступает в случае невозможности выполнения основного сценария. Примером такого сценария может являться обращение к партнерским сервисам при невозможности вызова внутреннего сервиса, реализующего основную функциональность системы.

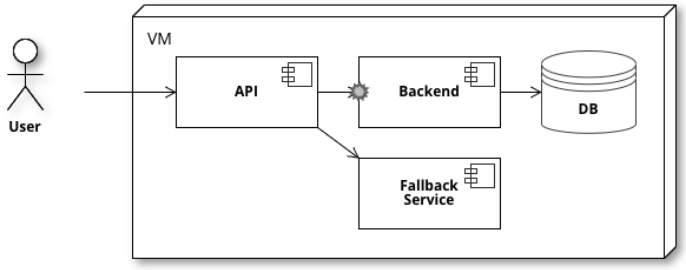


Рис. 5. Визуализация принципа работы fallback-сценария для обработки запроса

Многие структурные шаблоны проектирования и архитектурные фасады, к которым можно отнести API Gateway, Backend-for-Frontend, а также Sidecar (рис. 6), также помогают повысить отказоустойчивость системы, выполняя контроль доступа ко внутренним сервисам и обеспечивая мониторинг интеграций. Они играют критическую роль для систем, в которые сложно или невозможно вносить доработки.

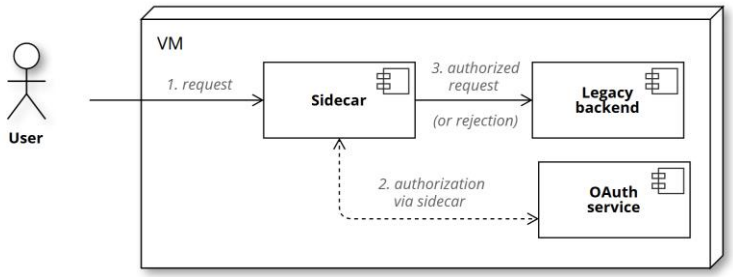


Рис. 6. Визуализация принципа работы авторизующего sidecar-сервиса

Заключение

В результате работы был проведен анализ рисков внедрения и эксплуатации приложений, а также разработана риск-стратегия, учитывающая основные причины и триггеры сбоев и позволяющая внедрять практики повышения надежности в логичной по-

следовательности. Для выделенных групп рисков описаны некоторые базовые методы, позволяющие минимизировать влияния недоступности системы на клиента или повысить скорость восстановления нормального режима функционирования.

Более подробно данные практики (с подробным разбором их основных преимуществ и недостатков), а также иные методы повышения надежности, включающие изменения в процессах тестирования, обновления и обслуживания программных компонентов, описываются в [10].

ЛИТЕРАТУРА

1. *Hoodat H., Rashidi H.* Classification and analysis of risks in software engineering // World Academy of Science, Engineering and Technology. – 56. P. 446–452.

2. Annual Outage Analysis // Uptime Institute. – 2024. – URL: https://uptimeinstitute.com/resources/research-and-reports/annual-outage-analysis-2023.

3. *Beyer B., Jones Ch., Petoff J., Murphy N.R.* Site Reliability Engineering: How Google Runs Production Systems. 1st. ed. – O'Reilly Media, Inc., 2016.

4. *Indrasiri K., Suhothayan S.* Design Patterns for Cloud Native Applications. – O'Reilly Media, Inc., 2021.

5. *Wilder B.* Cloud Architecture Patterns. – O'Reilly Media, Inc., 2012.

6. *Kleppmann M.* Designing Data-intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. – O'Reilly Media, Inc., 2017.

7. *Richards M., Ford N.* Fundamentals of Software Architecture: An Engineering Approach. 1st. ed. – O'Reilly Media, Inc., 2020.

8. *Brewer E.* Towards robust distributed systems. – DOI: 10.1145/343477.343502 // PODS, 2000.

9. *Ефремов А.А.* Теория надежности : конспект лекций. – Томск : Изд-во Том. полит. ун-та, 2015.

10. *Зоркин А.С.* Методы проектирования отказоустойчивых информационных систем: магистерская диссертация по направлению подготовки: 09.04.04 – Томск: [б.и.], 2024. – URL: https://vital.lib.tsu.ru/vital/access/manager/Repository/vital:21310 (дата обращения: 01.07.2024).

РЕАЛИЗАЦИЯ ОТЧЕТОВ И РОЛЕЙ В ЯДРЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ОРГАНИЗАЦИИ ВНЕУЧЕБНОЙ ДЕЯТЕЛЬНОСТИ

Игнатьев В.Д., Моисеев А.Н.
Томский государственный университет
ignatevvd@gmail.com, moiseev.tsu@gmail.com

Введение

"Информационная система поддержки воспитательной деятельности обучающихся" – система, которая разрабатывалась для муниципального бюджетного общеобразовательного учреждения [1]. Программный комплекс для сопровождения внеучебной деятельности – продолжение данной работы с целью сделать систему доступной для использования не только учебными заведениями среднего образования, но также и для использования высшими учебными заведениями. Начальный этап разработки данной системы был представлен в [2].

В данной работе представлена реализация и внедрение модулей, которые имеют общий функционал как для школы, так и для вуза. Разрабатываемая система должна иметь блок, ответственный за составление новых шаблонов отчётов, которые могут быть использованы повторно при необходимости. Также требуется доработать систему ролей, чтобы один пользователь мог выполнять несколько ролей одновременно.

1. Проектирование

В системе есть ряд атрибутов, по которым составляются отчеты воспитательного события. Данные отчеты стандартны для каждой определенной формы воспитательной работы и имеют ряд критериев, которым должны подчиняться. Например, есть критерий "фотография", по которому можно выяснить необходимо ли для закрытия данного события загружать в систему фотографии с проведенного воспитательного события.

Предполагается, что конструктор отчетов воспитательных событий должен быть реализован следующим образом: пользователь с правами администратора задает атри-

бут из предложенных, затем выбирает критерий из списка тех, что соответствуют этому атрибуту и выбирает отчет воспитательного события, который нужно будет составить. Таких отчетов воспитательного события можно выбрать несколько. Кроме того, если есть необходимость встроить информацию из отчетов в документ, имеющий, например, заголовок или другие составные части, отчеты можно встроить в определенный шаблон. Для этого администратор может создать блоки и из них собрать шаблон, а пользователь может выбрать шаблон из предложенных.

Для реализации работы конструктора отчетов воспитательных событий описанным выше образом, база данных была соответствующим образом расширена: были добавлены новые таблицы (рис. 1).

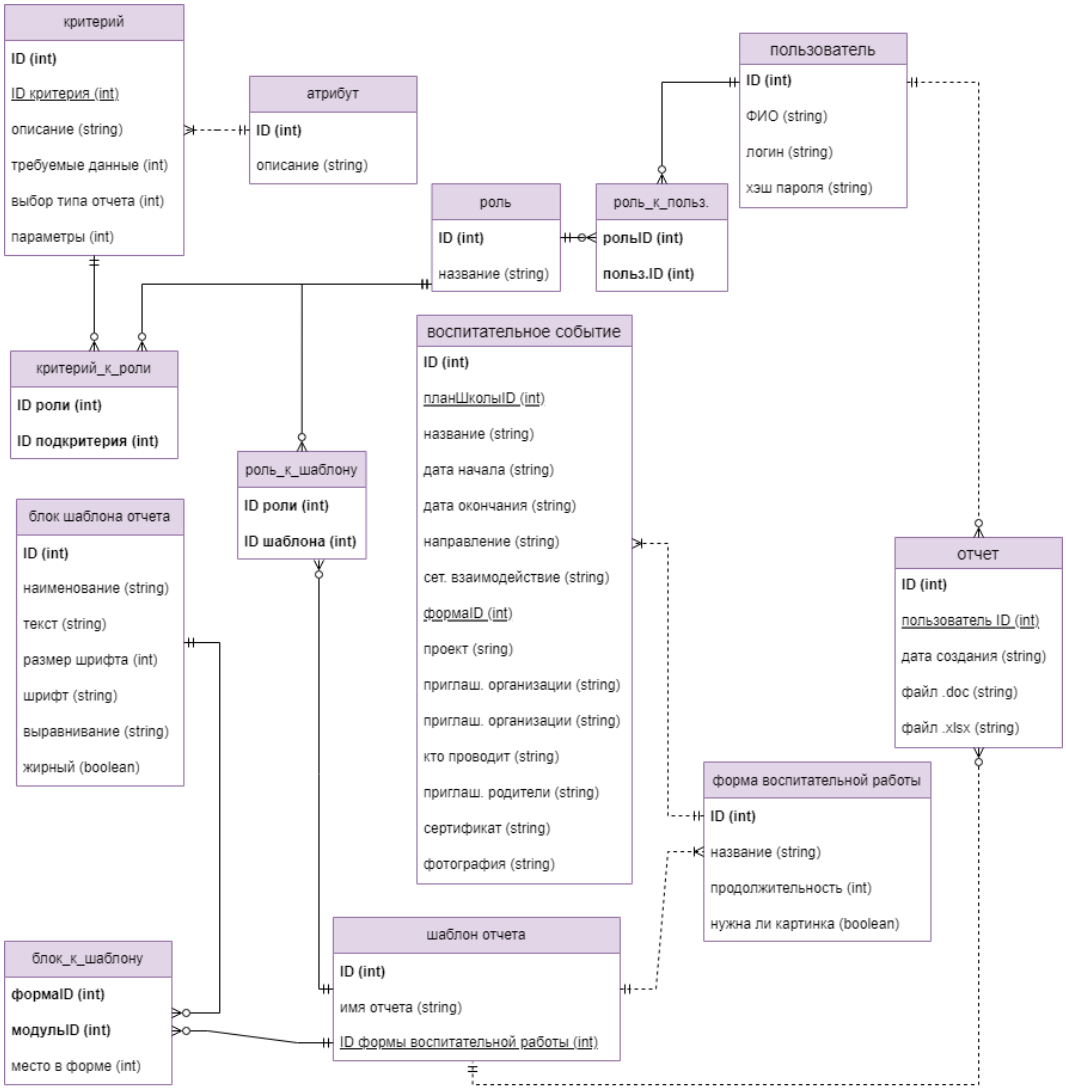


Рис. 1. Схема базы данных. Добавленные в систему таблицы

Таблица "Атрибут" отражает информацию об атрибутах. Описание атрибута отражает суть этого атрибута. Атрибут можно назвать "группой", которая обобщает несколько отчетов для более структурированного ориентирования между отчетами. Данный параметр был вынесен в отдельную таблицу в связи с тем, что в дальнейшем есть вероятность усложнения данной таблицы и увеличения количества её связей.

Таблица "Критерий" – это информация о критериях (отчётах воспитательного события). Описание критерия раскрывает его суть и помогает пользователю понять, какой отчёт он может составить.

Поле "Требуемые данные" указывает, какие данные нужно ввести при выборе этого атрибута: 0 – данные не требуются, 1 – требуются числовые данные.

Например, если пользователь хочет составить отчёт о количестве комнат в общежитии с определённым числом свободных мест, ему нужно будет указать, сколько мест должно быть в комнатах.

Поле "Выбор типа отчёта" определяет, какой отчёт будет составлен: 0 – автоматический количественный отчёт, 1 – существует возможность выбора между количественным и процентным отчётом (относительно общего числа).

Поле "Параметры" содержит значение 0 или 1 и показывает, сможет ли пользователь выбирать параметры группировки. Если значение – 0, то ничего не происходит. Если значение – 1, то при настройке атрибута пользователю будет доступен селектор с соответствующими условиями атрибута элементами из базы данных.

Таблица "Критерий к роли" содержит информацию о критерии и реализует возможность отображать критерии в соответствии с их уровнем доступа.

Таблица "Блок шаблона отчета" содержит часть шаблона отчета, которая подразумевает наличие одного заранее определенного отрезка текста с определенным шрифтом, уровнем жирности, выравниванием и.т.д.

Таблица "Блок к шаблону" реализует связь "многие ко многим" между блоком шаблона отчета и самим шаблоном отчета. Также данная таблица имеет поле "Место в форме", хранящее позицию блока в шаблоне отчета.

Таблица "Шаблон отчета" содержит в себе имя отчета, связь с формой воспитательной работы, что позволяет в дальнейшем с помощью данного шаблона генерировать уже готовые отчеты.

Таблица "Роль к шаблону" реализует связь "многие ко многим" между ролью и шаблоном отчета, позволяющую устанавливать уровни доступа к отчетам различных ролей.

Таблица "Отчет" хранит в себе дату создания отчета, а также пути к файлам внутри файловой системы на сервере. Данные пути задаются сервером автоматически посредством генерации GUID для каждого отдельного файла и присваиванием этого значения имени соответствующего файла. Файлы располагаются в файловой системе сервера, и, в случае необходимости, пользователь может получить к ним доступ.

Таблица "Форма воспитательного события" содержит поля названия (наример, "классный час"), продолжительность события, выраженную в днях, а также необходимость наличия картинки. В случае, если необходимость картинки для события отсутствует, по истечении срока продолжительности воспитательного события оно будет автоматически закрыто и прекращено.

Таблица "Воспитательное событие" содержит всю информацию, касающуюся воспитательного события, которое может быть как в школе, так и в ВУЗе. Поле "План школы" необходимо для связи события с планом школы, если это – школа. Поле "Название" хранит название события в строковом формате. Поле "Направление" содержит строковое название направления данного события. Поле "Сертификат" необходимо для подтверждения специалистом по безопасности данного события. Поле "Фотография" необходимо для внесения фотографии к данному событию, если таковая необходима согласно таблице "Форма воспитательной работы".

2. Конструктор отчетов

В предыдущей версии системы была возможность формировать отчёты о воспитательных мероприятиях на основе фиксированного набора данных, предоставленного заказчиком. В новой версии мы хотим улучшить эту функцию: дополнить её новыми

данными, связанными с высшим учебным заведением, а также дать пользователям возможность создавать документы с помощью шаблонов, которые динамически добавляются в систему. Перед сохранением документа в базе данных или его загрузкой пользователь сможет увидеть, как он будет выглядеть. Также необходимо предусмотреть возможность скачать документ сразу или отложить на более позднее время. Web-страница [3] конструктора отчётов, предназначенная для выполнения этих функций, показана на рис. 2.

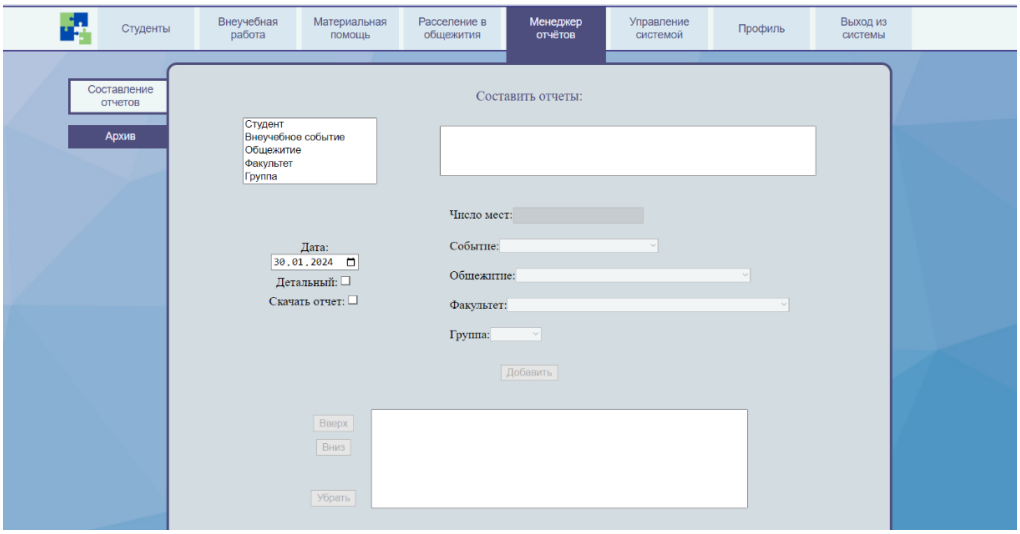


Рис. 2. Конструктор отчетов

Ниже приведен список атрибутов, из которых могут составляться отчеты, и виды отчетности по этим атрибутам.

1. Ученик.
   – Общее число учеников (можно вывести список ФИО).
   – Число учеников, у которых есть ПФДО.
   – Число учеников, проживающих в полной семье.
   – Число учеников, проживающих в неполной семье.
   – Число учеников обучающихся на дополнительном образовании.
   – Число учеников проживающих в семье с тяжелым материальным состоянием.
2. Внеучебное событие.
   – Общее число событий.
   – Число учеников, участвующих в событии (численный или процентный от общего количества; можно вывести список ФИО).
3. Класс.
   – Общее число учеников в классе (можно вывести список их ФИО).
   – Число школьников, имеющих сертификат ПФДО (численный или процентный от общего количества; можно вывести список ФИО).
   – Число учеников, проживающих семьях с тяжелым материальным состоянием (численный или процентный от общего количества; можно вывести список ФИО).
4. Факультет.
   – Общее число обучающихся на факультете (можно вывести список ФИО).
   – Число обучающихся, которые имеют сертификат ПФДО (численный или процентный от общего количества; можно вывести список ФИО).

– Число обучающихся, проживающих семьях с тяжелым материальным состоянием (численный или процентный от общего количества; можно вывести список ФИО).

Под списком атрибутов находятся поля настройки отчетности по атрибуту. Пользователь может видеть все параметры, но доступны для редактирования из них только те, которые относятся к выбранному атрибуту и той роли, в которой работает пользователь – они становятся доступными при выборе соответствующего им атрибута. Пример использования конструктора отчетов показан на рис. 3.

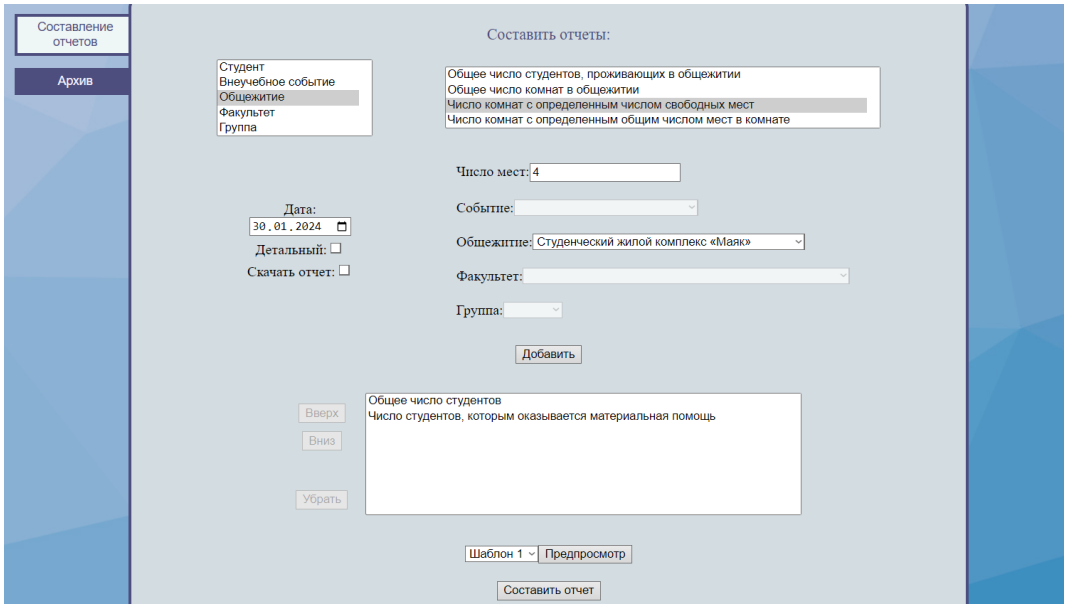


Рис. 3. Пример формирования отчета в конструкторе отчетов

В системе, помимо атрибутов, присутствует механизм шаблонов, представляющий собой структуру, сформированную из текстовых блоков, в которой заранее определены места для размещения отчётов. Пользователю предоставляется возможность выбора шаблона, обеспечивающего размещение в начале страницы полного наименования учебного заведения, в середине – отведённое пространство для размещения отчётной информации, а в конце – указание года составления документа. Этот подход гарантирует, что, независимо от выбранного типа и количества отчётов, конечный документ будет содержать наименование в начале, отчёты в середине и дату в конце. Прежде чем выбрать шаблон, пользователь может ознакомиться с его содержимым в специальном окне (рис. 4). Процесс выбора шаблона также осуществляется через специальный интерфейс. Отчёты могут быть созданы как с использованием шаблона, так и без него.

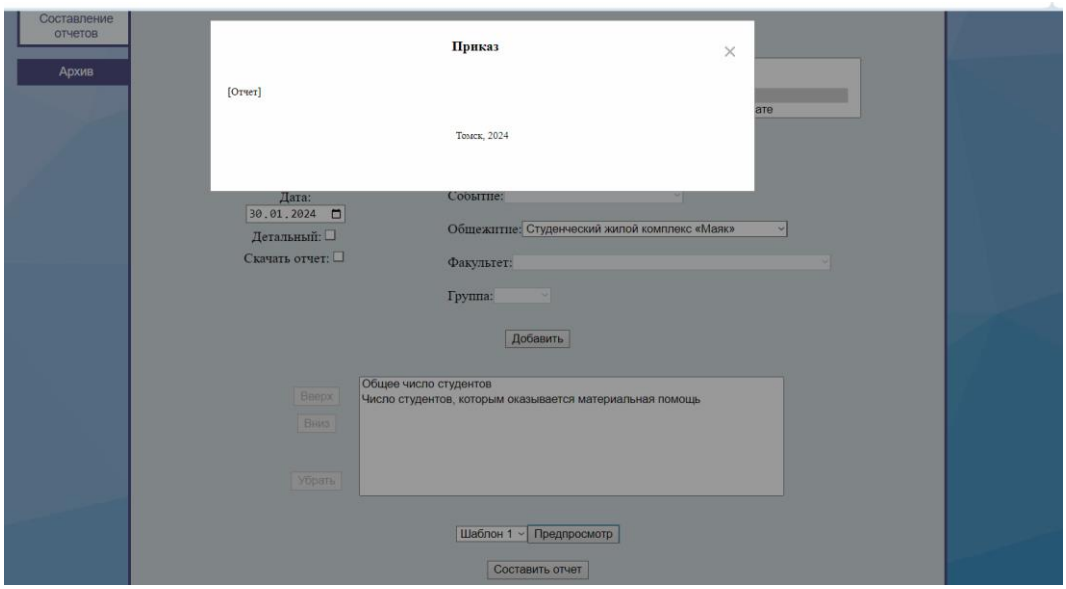


Рис. 4. Предпросмотр шаблона отчета в модальном окне

Под всеми перечисленными элементами страницы располагается кнопка "Составить отчет", после нажатия на которую пользователь увидит предварительный вид документа (рис. 5).

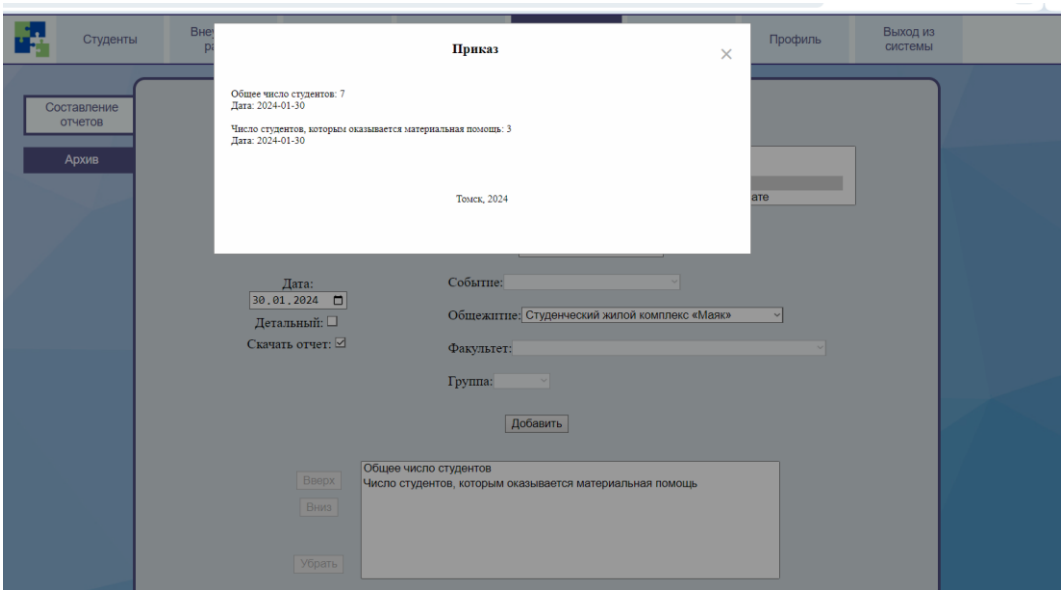


Рис. 5. Предварительный вид документа

На рис. 6 представлен вид документа в случае, если он был скачан сразу после формирования отчета.

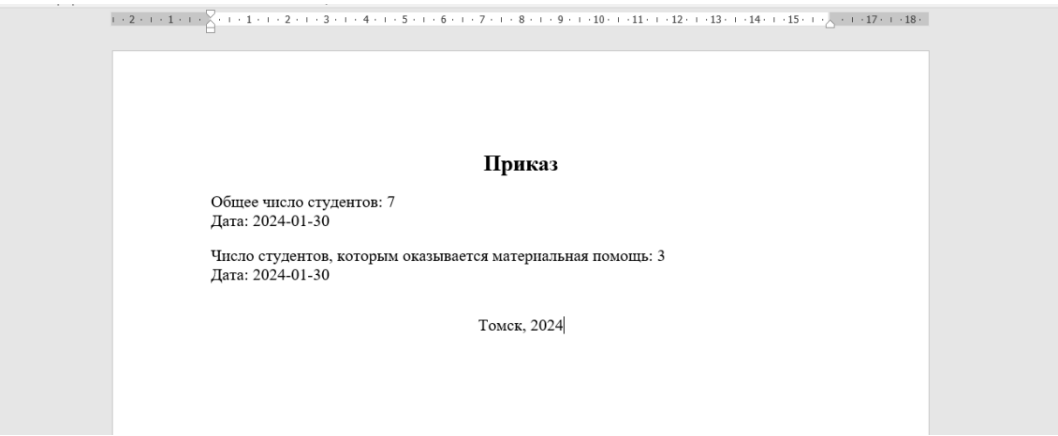


Рис. 6. Сформированный отчет

С помощью кнопок "Подтвердить" и "Отмена" пользователь может либо подтвердить генерацию отчета, либо вернуться назад к его редактированию. В случае нажатия кнопки "Подтвердить" пользователь отправляет запрос на сервер, который в свою очередь проверяет токен пользователя, и, если тот является валидным и пользователь имеет право на генерацию подобного отчета, составляет HTML-версию файла из блоков отчета и конвертирует их в docx-файл посредством плагина LibreOffice.

3. Установка уровней доступа к системе по ролям

Как упоминалось ранее, в базу данных [4] был добавлен ряд таблиц, которые позволяют заместителю по воспитательной работе создавать роли и разграничивать их области видимости (рис. 1). Это сделано для того, чтобы каждый сотрудник имел доступ только к тем данным, которые ему необходимы для работы.

Например, классный руководитель не сможет создать или увидеть отчёты, которые были созданы заместителем по безопасности жизнедеятельности. Это обеспечивает конфиденциальность и безопасность данных.

Для реализации этой функции [5] были созданы отдельные таблицы ролей, а также связей "роль к шаблону" и "роль к критерию". Эти таблицы позволяют точно определить, какие данные доступны каждой роли и как они связаны друг с другом. Такой подход позволяет эффективно управлять доступом к информации и обеспечивать безопасность данных в школе или ВУЗе. Это особенно важно в условиях, когда необходимо обеспечить конфиденциальность чувствительной информации и предотвратить несанкционированный доступ к ней.

Заключение

В процессе работы были выполнены поставленные задачи: расширена схема базы данных, установлены базовые уровни доступа к системе в соответствии с ролями, реализован конструктор отчётов, реализовано создание шаблонов и блоков отчётов для роли администратора. Таким образом был доработан существующий программный комплекс для сопровождения внеучебной деятельности путём внедрения в него модулей, позволяющих использовать систему в высших учебных заведениях.

ЛИТЕРАТУРА

1. *Игнатьев В.Д., Зенкина Е.С., Моисеев А.Н., Кравец А.А.* Начальный этап разработки информационной системы поддержки воспитательной деятельности в учебном заведении // Материалы X-й Международной молодёжной научной конференции "Математическое и программное обеспечение информационных, технических и экономических систем", Томск, 26-29 мая 2023 г. – Томск: Изд-во Том. гос. ун-та, 2023. – С. 149–155.

2. *Зенкина Е.С., Игнатьев В.Д., Спиридонова А.А.* Информационная система организации воспитательной деятельности // Фундаментальные и прикладные исследования в физике, математике и информатике. Материалы

симпозиума в рамках XVII (XLIX) Международной научной конференции студентов, аспирантов и молодых ученых. Вып. 23. – Кемерово: КемГУ, 2022. – С. 135–138.

3. *Вайс Р., Хортон А.* Разработка веб-приложений в ReactJS. Пер. с англ. – М.: Питер, 2007. – 254 с.: ил.

4. *Вайнберг П.Н., Грофф Д.Р., Оппель Э.Дж.* SQL: Полное руководство. 3-е изд. Пер. с англ. – М.: ООО "И.Д. Вильямс", 2015. – 960 с.: ил.

5. *Кантелон М., Мек Б., Янг А.* Node.js в действии. 2-е изд. Пер. с англ. – СПб.: Питер, 2018. – 432 с.: ил.

РАЗВИТИЕ ОБОЛОЧКИ ЭКСПЕРТНЫХ СИСТЕМ GURU

Бабанов А.М., Пустовой А.Е.
Томский государственный университет
babanov2000@mail.ru, ai.pustovoy284@gmail.com

Введение

В 70-х годах прошлого века основной акцент в искусственном интеллекте был сделан на разработке "экспертных систем" (называемых также "системами, основанными на знаниях"), которые были способны при наличии соответствующих знаний в проблемной области достичь или превзойти производительность людей- экспертов при решении узко определенных задач.

Первая экспертная система, Dendral интерпретировала выходные данные массового спектрометра столь же точно, как и опытные химики. Хотя успех системы Dendral позволил сообществу исследователей по искусственному интеллекту убедиться в важности представления знаний, формальные средства представления, используемые в Dendral, были в высшей степени специализированными и относящимися только к данной проблемной области химии. Со временем у исследователей появилось стремление к стандартизации формальных средств представления знаний, что позволило бы упростить процесс создания новых экспертных систем [1].

Так появились многочисленные модели представления знаний, одной из которых была логическая модель продукций. Продукционные системы принадлежат к числу самых первых систем прямого логического вывода, получивших широкое распространение.

1. Продукционные экспертные системы

Экспертные системы (ЭС), основанные на продукциях, позволяют представить знания экспертов в виде множества правил "если …, то …". С их помощью даже неподготовленный человек способен проанализировать типичную ситуацию и сделать профессиональный вывод. В ходе консультации ЭС находит очередной вопрос, задает его человеку, принимает ответ, агрегирует его с предыдущими ответами и в ходе логического вывода по продукциям определяет следующую текущую цель, которая приведет к очередному вопросу. На основании полученных ответов ЭС придет к определенному решению, выраженному в виде некоторого значения целевой переменной, причем ЭС не бездумно задает всегда одни и те же вопросы в фиксированном порядке, а определяет разумную траекторию экспертизы в зависимости от полученных ею ответов.

Типичными примерами таких ЭС являются многочисленные системы постановки медицинских диагнозов. Они учитывают многочисленные симптомы, показатели анализов, другие характеристики пациентов. Опытный врач выразил в продукциях ЭС свои профессиональные правила диагностики, и теперь от менее опытного коллеги требуется только передать особенности конкретного случая, отвечая на вопросы системы. В дальнейшем такие медицинские ЭС мы будем рассматривать в качестве объекта наших рассуждений.

2. Оболочки экспертных систем

Наряду с ЭС, раз и навсегда жестко фиксирующими набор правил, существуют т.н. оболочки экспертных систем (ОЭС). Здесь уместна аналогия с системами управления

базами данных (СУБД). Каждая СУБД предоставляет модель данных – совокупность правил структуризации данных и правил задания свойств допустимых данных. Все эти правила материализуются в виде команд языка определения данных. Задача проектировщика базы данных (БД) сводится к описанию своей предметной области (ПрО) с использованием правил и грамматики предложенных команд СУБД. После этого система БД настроена и можно использовать ее для хранения данных об объектах этой ПрО.

Аналогично и ОЭС предлагает проектировщику ЭС некоторую модель представления знаний (в отличие от систем БД, ЭС относятся к системам искусственного интеллекта, а в них общепринятым считается использование термина "знания" и язык, на котором опытный эксперт должен представить свой опыт анализа ситуаций ПрО. После проектирования БД и ЭС в обоих случаях их пользователи могут удовлетворять свои информационные потребности – манипулировать данными и проводить консультации. Т.о., универсальные программные продукты СУБД и ОЭС позволяют использовать их в самых различных ПрО.

3. Оболочка экспертных систем GURU

Авторам известна лишь одна реализация ОЭС GURU [2], представляющая собой приложение для операционной системы MS DOS. Предполагается ее автономное использование одним пользователем (либо в роли эксперта, либо в роли пользователя ЭС). Каждая ЭС представляет собой единый набор правил, сформированный одним экспертом и оформленный в виде файла. Желающий провести консультацию выбирает соответствующую ЭС из нескольких хранящихся на диске компьютера. Запустив консультацию, пользователь отвечает на вопросы в течение одного сеанса и в конце получает на экране результат – значение целевой переменной. Ответы и результат не сохраняются. В простейших случаях этой функциональности достаточно.

К положительным характеристикам GURU относятся:

– мощные механизмы прямого и обратного логического вывода, определяющие ход консультации на основании полученных ответов и порождающие иллюзию разумного поведения;
– многочисленные настройки логического вывода, позволяющие тонко определить поведение системы;
– богатые возможности аппарата для работы с нечеткими значениями и правилами, позволяющего оперировать неточными, размытыми знаниями.

4. Модель знаний GURU

Модель знаний GURU удобнее всего описывать, опираясь на грамматику текстового описания ЭС, сохраняемого в исходном файле. Перед консультацией это описание преобразуется в исполняемый вид.

Поскольку эта грамматика нигде не опубликована, пришлось ее синтезировать по известным описаниям и примерам применения. Т.о., в ходе анализа прототипа был выполнен программный реинжиниринг – по описанию и поведению системы был восстановлен проектный артефакт, который составит ядро будущей грамматики ЭС и будет совершенствоваться в ходе разработки.

Грамматика дается в нотации Бэкуса – Наура.

Сначала определим общеупотребимые нетерминальные символы.

Оператор OUTPUT предназначен для показа пользователю значений (в т.ч. и нечетких) переменных экспертизы и текстовых констант. Каждое значение нечеткой переменной снабжается фактором уверенности (certainty factor – CF) в том, что эта переменная принимает указанное значение. Значения CF изменяются от 0 (абсолютная уверенность в том, что этого значения нет) до 100 (абсолютная уверенность в том, что это значение есть).

<output> ::= OUTPUT: {<output data>,…}

<output data> ::= <expertise variable name> | <text>

Операция присвоения значения переменным среды и экспертизы позволяет задавать как четкие, так и нечеткие значения.

<assignment> ::= <variable name> = <value>

<variable name> ::= <expertise variable name> | <environment variable name>

<value> ::= <single value> | {<fuzzy value>,…}

<fuzzy value> ::= <single value> CF 1..100

Оператор INPUT предназначен для ввода пользователем значения переменной экспертизы в ответ на предложенный в виде текстовой константы вопрос. В отличие от операции вывода, в системе GURU допустим ввод только одиночного четкого значения (с фактором уверенности CF, равным 100) для каждой переменной, т.е. модель знаний GURU не предполагает сомнений пользователя.

<input> ::= INPUT: <expertise variable name> <type> WITH <text>

Далее последовательно определяются грамматические правила собственно ЭС, начиная с самых глобальных нетерминальных символов. Вся ЭС описывается последовательностью блоков, как показано в первом правиле.

<expert system> ::= <definition> <initialization> <rule block> <variable block> <completion>

Блок определения задает целевую переменную экспертизы.

<definition> ::= GOAL: <expertise variable name>

Блок инициализации предназначен для указания команд, которые будут выполнены в начале экспертизы до начала работы механизма вывода.

<initialization> ::= INITIAL: {<initialization command>,…}

<initialization command> ::= <output> | <assignment> | <input>

Основным блоком ЭС является блок правил, которые в основном и определяют ее базу знаний.

<rule block> ::= {<rule>,…}

<rule> ::= [{<auxiliary element>,…}] [<ready>] IF: <premise> THEN: <conclusion> [<reason>] [<used variables>]

Вспомогательные элементы задают стратегию использования правила механизмом вывода и комментарий к нему.

<auxiliary element> ::= <priority> | <cost> | <test> | <comment>

<priority> ::= PRIORITY: 1..100

<cost> ::= COST: 1..100

<test> ::= TEST: {S | E | P}

<comment> ::= COMMENT: <text>

Следующий нетерминал определяет набор команд, выполняемых при обращении к правилу (но до проверки его посылки).

<ready> ::= READY: {<ready command>,…}

<ready command> ::= <output> | <assignment>

Посылка правила должна удовлетворять нетерминалу <premise> Он определяет грамматику логического выражения, атомами которого являются операции сравнения.

<premise> ::= < logical expression >

<logical expression> ::= <logical operand> | NOT <logical operand> | <logical operand> <binary logical operator> <logical operand>

<logical operand> ::= (<logical expression>) | (<comparison expression>)

<binary logical operator> ::= AND | OR

<comparison expression> ::= {<expertise variable name> | <function name>(<expertise variable name>)} <comparison operator> <single value>

<comparison operator> ::= > | < | >= | <= | == | <> | !=

Если посылка правила оценивается как истина, система выполняет действия заключения.

<conclusion> ::= {<action>,…}

<action> ::= <expertise variable name> <operator> <value> | <output>

<operator> ::= +=, -=, =

Нетерминальный символ REASON задает текст, объясняющий пользователю, почему система применила это правило.

<reason> ::= REASON: <text >

В последнем блоке правила определяются переменные экспертизы, значения которых либо нужны для проверки посылки, либо изменяются в заключении правила.

<used variables> ::= <needs> | <changes>

<needs> ::= NEEDS: {<expertise variable name>,…}

<changes> ::= CHANGES: {<expertise variable name>,…}

Блок переменных предназначен для объявления переменных экспертизы, задания способов (пользователь или механизм вывода) и стратегии определения их значений, а также принципов работы с нечеткими значениями.

<variable block> ::= {<variable>,…}

<variable> ::= VAR: <expertise variable name> [{<variable command>,…}]

<variable command> ::= <find> | <label> | <when> | <cf type> | <rigor> | <limit>

<find> ::= FIND: {<find command>,…}

<find command> ::= <assignment> | <input>

<label> ::= LABEL: <text>

<when> ::= WHEN: {F | L | N}

<cf type> ::= CF TYPE: <cf type value><cf type value>

<cf type value> ::= {M | P}

<rigor> ::= RIGOR: {M | C | A}

<limit> ::= LIMIT: <number>

Завершающий блок ЭС определяет команды, которые система выполняет по окончании экспертизы, когда получено значение целевой переменной. Именно здесь указывается команда вывода этого значения.

<completion> ::= DO: {<completion command>,…}

<completion command> ::= <assignment> | <output>

5. Направления развития модели знаний GURU

Оболочка GURU хорошо решала задачу создания ЭС и была в свое время достаточно популярна. Однако продукционную модель представления знаний и механизмы логического вывода можно использовать и в более сложных ситуациях.

Во-первых, ЭС может представлять собой совокупность нескольких взаимосвязанных частей (экспертных подсистем), подготовленных различными узкими специалистами, но в целом решающих одну задачу. Это напоминает ситуацию, когда врач-терапевт направляет пациента на консультации к специалистам и впоследствии учитывает их мнение в окончательном диагнозе.

Во-вторых, в сложных случаях, когда одному человеку затруднительно отвечать на вопросы ЭС, требующие специальные знания, можно устраивать что-то, напоминающее консилиум. При этом система при необходимости посылает вопрос нужному специалисту и продолжает консультацию, получив его ответ. Т.о., консультация становится многопользовательской.

В-третьих, хотелось бы расширить спектр возможных видов ответов. Помимо ввода четкого значения неплохо было бы предусмотреть:

– ввод нечеткого значения;

– выбор из предложенных альтернатив (простой или множественный);

– ввод нескольких значений;

– упорядоченный ввод нескольких значений;
– упорядочивание предложенных альтернатив.

В-четвертых, хорошо было бы иметь возможность откладывать консультацию, сохранять полученные ответы, промежуточные и окончательные результаты в долговременном хранилище для последующего возобновления консультации, анализа результатов и, возможно, изменения ответов и хода вывода.

Предложенные перспективы открываются, если спроектировать и реализовать расширенную ОЭС продукционного типа, взяв в качестве прообраза GURU и добавив следующее:

– распределенная архитектура;
– централизованное хранилище ЭС и данных консультаций;
– взаимосвязи между ЭС и возможность обращения к другим консультациям в ходе текущей консультации;
– персонализация консультаций;
– расширение видов возможных ответов пользователя.

6. Архитектура расширенной оболочки ЭС

Проект системы можно представить в виде диаграммы компонентов в нотации UML [3] (рис. 1). Расширенная оболочка ЭС представляет собой трехуровневую клиент-серверную систему.

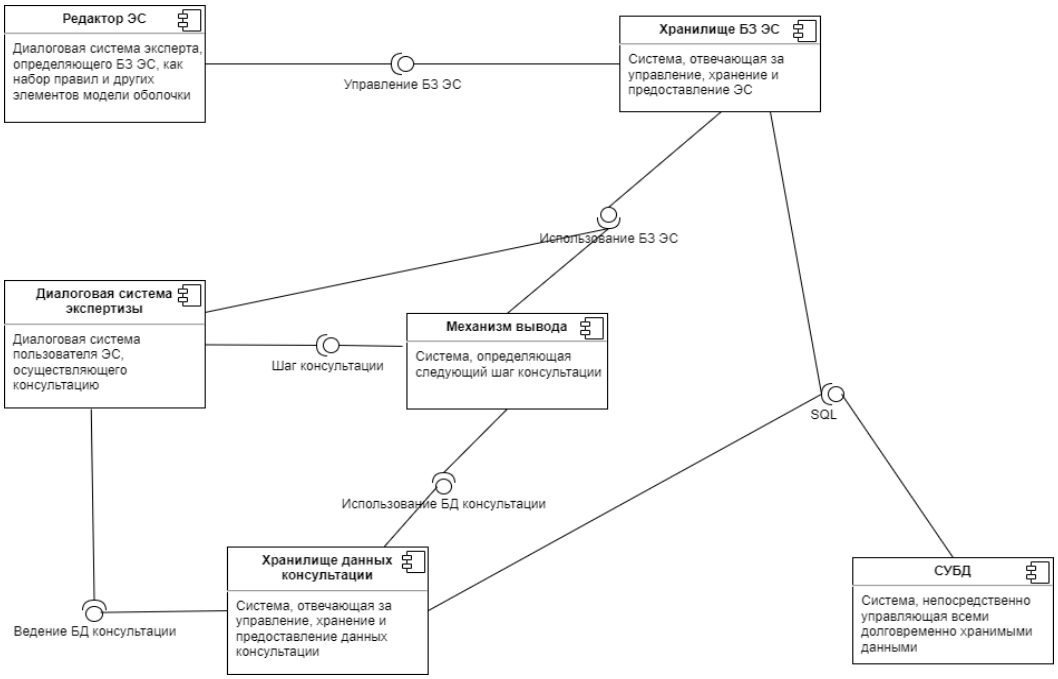


Рис. 1. Диаграмма компонентов расширенной оболочки ЭС

Ее фронтенд составляют диалоговые инструменты эксперта ("Редактор ЭС") и пользователя ЭС ("Диалоговая система экспертизы"). Редактор ЭС предназначен для первоначального создания и последующего изменения базы знаний ЭС. С помощью второго компонента осуществляется запуск и управление экспертизой, предъявление вопросов пользователю и ввод его ответов. Предполагается использование для этих компонентов различных платформ – десктопное приложение, веб-приложение, мобильное приложение.

Промежуточный слой представляет собой сервер приложений, одновременно обслуживающий несколько клиентских приложений. Здесь представлены основные "ин-

теллектуальные" части системы. С одной стороны, это элементы, отвечающие за долговременное хранение и предоставление всей информации, касающейся определения ЭС и состояний всех текущих и проведенных экспертиз. С другой стороны, именно здесь реализованы механизмы прямого и обратного логического вывода. На сервере приложений функционируют компоненты "Хранилище БЗ ЭС", "Хранилище данных консультации" и "Механизм вывода". Они реализуют показанные на диаграмме интерфейсы, как с диалоговыми компонентами, так и между собой.

Наконец, бэкенд системы – это система управления базами данных (СУБД), предназначенная для управления данными на диске. В случае реляционной СУБД языком интерфейса с компонентами-хранилищами является SQL. Второй вариант интерфейса – использование объектно-ориентированной технологии доступа к данным, обеспечивающей отображение объектов оперативной памяти в кортежи отношений реляционной БД и наоборот.

7. Представление БЗ ЭС и БД консультаций

Одной из особенностей расширенной оболочки ЭС является хранение ЭС и данных консультаций в БД, что предполагает проектирование схемы этой БД. На рис. 2 представлена предварительная каркасная ER-схема определения ЭС (в модели знаний GURU). Эта схема составляет остов (каркас) полной ER-схемы ЭС. Чтобы не усложнять рисунок, в схеме показаны только основные элементы и опущены атрибуты.

На схеме в нотации Чена [4] представлены основные множества сущностей и множества связей, а также ограничения целостности на множества связей. Хорошо видно, как нетерминальные символы грамматики отображаются в структуры данных, предназначенные для хранения соответствующих элементов ЭС.

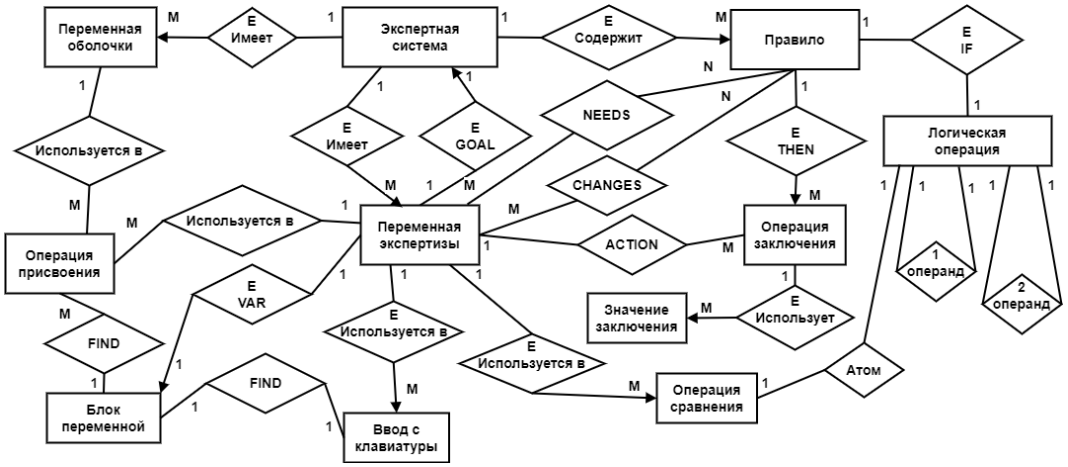


Рис. 2. Каркасная ER-диаграмма данных ЭС

Развитие модели знаний и реализация хранения данных консультаций повлечет за собой расширение схемы.

Заключение

Статья представляет собой "декларацию о намерениях" разработать расширенную оболочку ЭС, являющуюся развитием ОЭС GURU. Прекрасно осознавая положительные стороны прообраза, авторы видят ее недостатки и намечают пути их устранения. В статье представлены первые аналитические и проектные артефакты – архитектура будущей системы, модель знаний GURU в виде грамматики ЭС и схема БД ЭС.

ЛИТЕРАТУРА

1. *Russell S., Norvig P.* Artificial Intelligence: A Modern Approach, 3th ed. – Pearson, 2009. – Pp. 1152.

2. *Еремеев А.П.* Проектирование экспертных систем средствами инструментальной системы GURU. – М.: Издательство МЭИ, 1996. – 52 с.

3. *Rumbaugh J., Jacobson I., Booch G.* Unified Modeling Language Reference Manual, The Second Edition. – Addison-Wesley, 2004. – Pp. 721.

4. *Chen P.P.* The Entity-Relationship Model – Toward a Unified View of Data // ACM Trans. Database Systems. – 1976. – V. 1. – № 1. – P. 9–36.

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СМО БЕСПРОВОДНЫХ СЕТЕЙ СВЯЗИ С МИГРАЦИЕЙ АБОНЕНТОВ

Салимзянов Р.Р., Моисеев А.Н.

Томский государственный университет

rsalimzyanov@yahoo.com

Введение

Особенностью сетей связи пятого поколения является использование миллиметровых волн более высоких частот, что может обеспечить высокую скорость и низкую задержку передачи сигнала. При высоких частотах миллиметровых волн распространение радиосигнала существенно слабее, чем в низких диапазонах, что приводит к необходимости более точного моделирования сетей связи нового поколения. Для математического моделирования подобных систем используют аппарат теории массового обслуживания (ТМО), в частности – асимптотический анализ [1]. Результаты, полученные данным методом, являются приближенными и требуют оценки области применимости. Для этого возникает необходимость разработки программ имитационного моделирования систем массового обслуживания (СМО).

Ранее [2] была рассмотрена абонентская сеть, состоящая из двух узлов, пользователи находящиеся внутри радиуса действия сети могут перемещаться от одного узла сети к другому, не прерывая свое обслуживания. В случае, если абонент выходит за пределы распространения сигнала своего узла и не может подключиться к другому узлу, абонент покидает систему.

1. Описание математической модели

В качестве математического описания подобной системы была предложена бесконечно линейная СМО [3], на вход которой поступает простейший поток заявок с интенсивностью λ. Время обслуживания заявок является произвольной функцией распределения $B(x)$. Входящая заявка занимает любой свободный прибор и с вероятностью ν_1 или ν_2 переходит в первое или второе состояние соответственно. Заявка может перейти из состояния 1 в состояние 2 и из состояния 2 в состояние 1 за время Δt с вероятностью $\alpha_{12}\Delta t$ и $\alpha_{21}\Delta t$ соответственно, либо с вероятностью $\alpha_{10}\Delta t$ для состояния 1 и $\alpha_{20}\Delta t$ для состояния 2 заявка может покинуть систему, не завершив обслуживание. По окончании обслуживания заявка также покидает систему.

Для описанной выше СМО были получены аппроксимации распределений вероятностей числа заявок, находящихся в первом и втором состоянии. Полученные результаты могут быть применены лишь при определённом асимптотическом условии.

2. Общая модель имитационного моделирования

Для реализации имитационной модели рассматриваемой системы, был выбран дискретно-событийный подход [4]. Программа имитационного моделирования состоит из трех модулей (рис. 1):

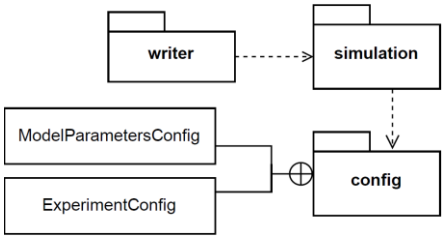


Рис. 1. Общая архитектура приложения

1. Модуль config необходим для считывания и передачи параметров имитационного моделирования из конфигурационного файла с расширением .yml.
2. В модуле simulation реализован алгоритм имитационного моделирования.
3. Модуль writer позволяет сохранить результаты проведения эксперимента в формате .xlsx.

Каждый модуль программы работает независимо друг от друга и может быть изменен на аналогичный, например, в случае изменения формата ввода входных данных или вывода результатов работы программы.

Рассмотрим диаграмму классов [5] имитационной модели модуля simulation для исследуемой системы (рис. 2).

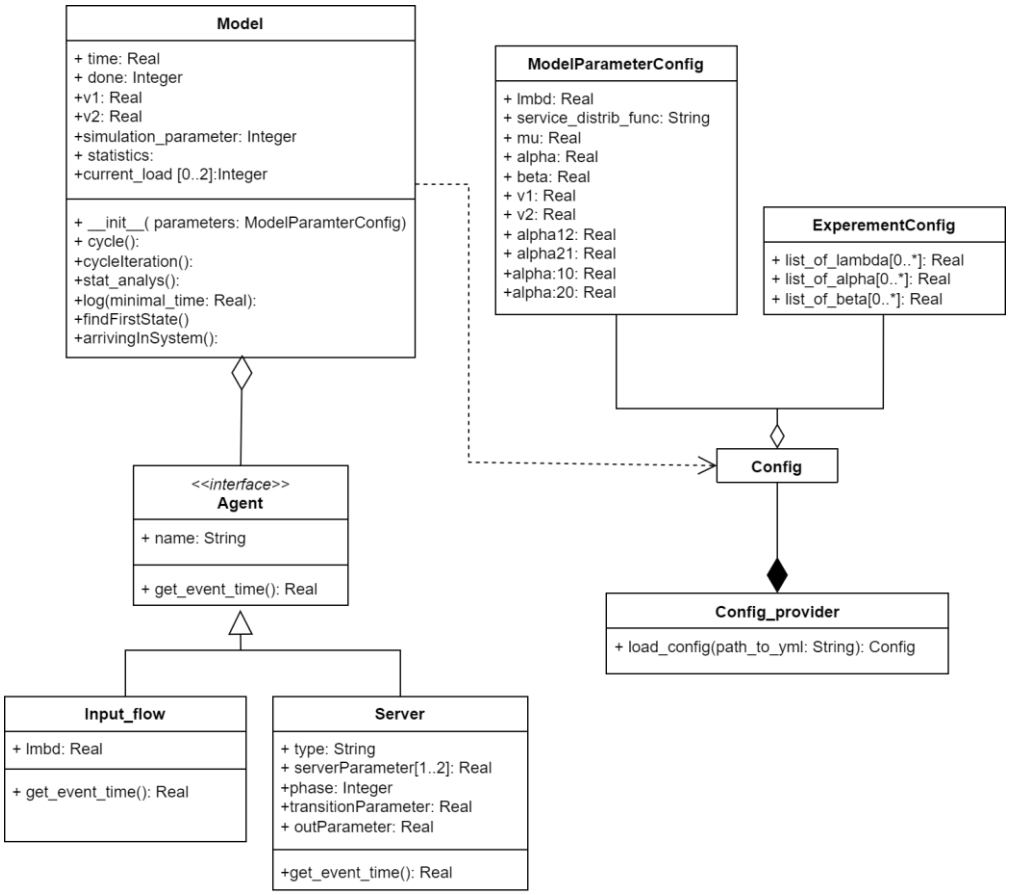


Рис. 2. Диаграмма классов программы имитационного моделирования (модули simulation и config)

Модуль simulation является основой программы имитационного моделирования и состоит из четырех классов.

Класс Model содержит основную логику проведения цикла имитационного моделирования и позволяет собирать статистику изменений состояния системы. Также класс Model реализует основные методы имитационного моделирования, они отвечают за прибытие новой заявки, определение состояния, в которое перейдет заявка и переход между состояниями.

Agent – интерфейс, а классы Server и Input_flow реализуют данный интерфейс. Agent содержит атрибут name, который отвечает за идентификацию класса и принимает значения 'SERVER' или 'INPUT_FLOW' для классов Server и Input_flow соответственно. Метод get_event_time() является абстрактным методом вычисления моментов времени наступления событий.

Класс Server реализует поведение одного прибора системы, позволяет получить информацию о том, какое событие случится в каждом приборе в ходе имитационного моделирования.

В методе cycle() класса Model реализован основной алгоритм имитационного моделирования (рис. 3).

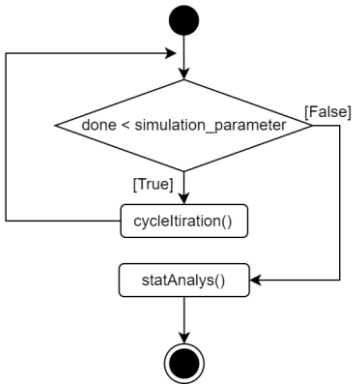


Рис. 3. Обобщенный алгоритм имитационного моделирования. Метод cycle() класса Model

Метод cycle() в цикле вызывает функцию cycleIteration(), которая отвечает за один шаг моделирования системы. Цикл продолжается до тех пор, пока атрибут done не станет равным числу, которое задал пользователь. Атрибут done обозначает число выполненных заявок в системе, т.е. число выходов заявки из системы в результате истечения времени обслуживания. По завершении цикла запускается метод statAnalys(), который необходим для расчета числовых характеристик системы.

Алгоритм метода cycleIteration() класса Model изображен на рис. 4. В случае наступления события входящего потока, создается заявка и определяется, в какой узел она попадет. Для этой заявки генерируется время изменения ее состояния. Если событие происходит с заявкой, которая находится на обслуживании, предполагается изменение состояния заявки и переопределение ее параметров в зависимости от ее текущего состояния, например, в случае, если заявка покинет систему, то ее необходимо удалить из общего списка заявок (agents).

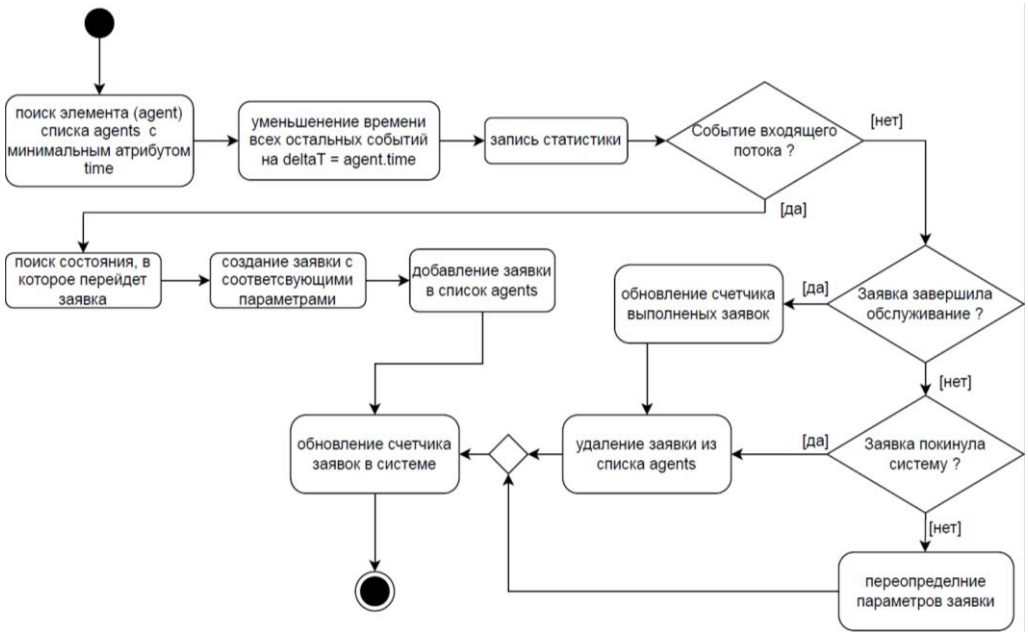


Рис. 4 Алгоритм метода cycleIteration() класса Model

3. Обработка пришедшей заявки

Каждая вновь пришедшая заявка может попасть в первое или второе состояние системы. У пользователя есть выбор функции распределения времени обслуживания. Для корректного создания заявок в программе имитационного моделирования был разработан метод arrivingInSystem() (рис. 5).

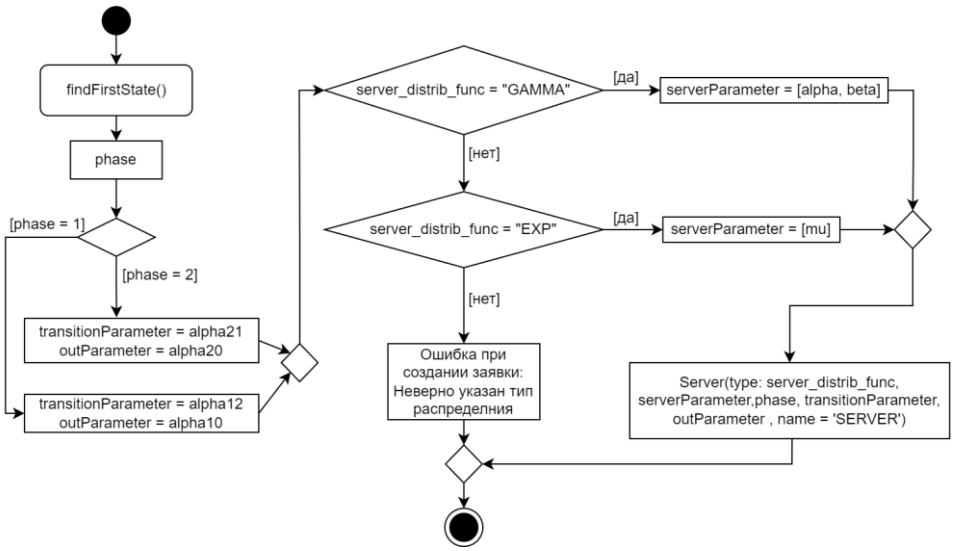


Рис. 5. Алгоритм метода arrivingInSystem() класса Model

В начале работы алгоритма вызывается функция findFirstState(), которая возвращает номер состояния, в котором окажется заявка (1 или 2). От номера состояния зависят значения атрибутов класса Server: transitionParameter и outParameter. Далее в зависимости от параметров, которые указал пользователь для проведения эксперимента, будет создана заявка с соответствующим видом и параметрами распределения времени обслуживания заявки.

Функция findFirstState() определяет номер состояния, в которое попадет заявка. Определение номера происходит с помощью сравнения указанных пользователем вероятностей попадания в 1-е или 2-е состояние со случайно сгенерированной величиной [6]. Для этого генерируется случайная величина согласно равномерному закону распределения в интервале от 0 до 1. Если случайная величина меньше или равна $1-v_1$, то заявка попадет во 2-е состояние, если условие не выполняется – то в 1-е.

4. Пример работы программы

Перед запуском программы имитационного моделирования пользователю необходимо определить параметры системы и записать их в конфигурационный файл с расширением .yml; пример такого файла показан на рис. 6.

```yaml
model_parameters:
  lambda: 23
  type_of_service: gamma # gamma, exp
  mu: 0.1 # for exp
  alpha: 0.75 # for gamma  a
  beta: 0.75 #for gamma  k
  v1: 0.3
  v2: 0.7
  alpha12: 1.8
  alpha21: 1
  alpha10: 1
  alpha20: 0.1
experiments:
  list_of_lambda: None
  list_of_alphas: None
  list_of_beta: None
```

Рис. 6. Пример файла config.yml

Для запуска программы необходимо указать два параметра:

1. output-path – путь до директории, в которой будут храниться результирующие файлы. Если такой директории нет, программа создаст ее по указанному пути.
2. config – путь к файлу .yml с параметрами системы. По умолчанию принимает значение config.yml.

Пример запуска программы представлен на рис. 7.

```
pypy .\main.py --config config.yml --output-path out_dir
```

Рис. 7. Пример запуска программы имитационного моделирования

В результате в указанной папке появятся файлы .xlsx (количество зависит от параметров experiments). Каждый файл содержит параметры эксперимента, среднее число заявок на обслуживании в 1-м и 2-м состояниях (mean1 и mean2) и распределения вероятностей числа заявок в системе в 1-м и 2-м состояниях (рис. 8).

P1_simula	P2_simulated						
0,013023	3,3E-05	lmbd	23		mean 1 : 4.313747302155588		
0,057073	0,000255	service_d	gamma		mean 2 : 10.644096070961798		
0,124981	0,001231	mu					
0,17944	0,004699	alpha	0,75				
0,193904	0,012706	beta	0,75				
0,166598	0,026992	v1	0,3				
0,11943	0,048107	v2	0,7				
0,073631	0,07348	alpha12	1,8				
0,039882	0,09786	alpha21	1				
0,018923	0,115121	alpha10	1				
0,008244	0,12243	alpha20	0,1				
0,003237	0,118621						
0,001102	0,105038						
0,000363	0,086434						

Рис. 8. Пример результирующего файла

5. Оценка точности имитационной модели

Для того, чтобы оценить точность работы имитационной модели, было выполнено два независимых запуска программы для разного необходимого числа обслуженных заявок N. В результате запусков были получены эмпирические функции распределения вероятностей (ФРВ) числа заявок в системе и посчитаны значения расстояний Колмогорова между парой соответствующих ФРВ по формуле $\Delta = \max_i \left|F_1(i) - F_2(i)\right|$, где $F_1(i)$ и $F_2(i)$ – эмпирические ФРВ числа заявок в системе, полученные путем запуска двух независимых итераций имитационного моделирования для одинаковых параметров системы.

В табл. 1 отображена зависимость расстояния Колмогорова от числа обслуженных заявок в имитационной модели. Можно сделать вывод, что с увеличением необходимого числа обслуженных заявок для каждой итерации, точность имитационного моделирования увеличивается и при $N = 10^6$ составляет 0.0004, что можно считать приемлемой погрешностью.

Таблица 1

Расстояние Колмогорова для первого и второго состояний при разном числе обслуженных заявок *N*

N	**100**	**1000**	**1000000**
Δ_1	0.0303	0.0067	0.0004
Δ_2	0.0665	0.0384	0.0004

Заключение

В результате была разработана программа имитационного моделирования абонентской сети с миграцией абонентов. Программа может быть масштабирована для схожих систем и сетей массового обслуживания. На основе проведенных численных экспериментов было определено значение числа обслуженных заявок для дальнейшего сравнения результатов, полученных в ходе имитационного моделирования с теоретическими результатами.

ЛИТЕРАТУРА

1. *Моисеева С.П., Назаров А.А.* Метод асимптотического анализа в теории массового обслуживания : учеб. пособие – Томск : Изд-во НТЛ, 2006. – 109 с.

2. *Салимзянов Р.Р., Моисеев А.Н.* Уравнение локального баланса для распределения вероятностей числа клиентов в узлах IAB-сети // Системы управления, информационные технологии и математическое моделирование : материалы V Всерос. науч.-практ. конф. с междунар. Участием, Омск, 25–26 апр. 2023 г. – Омск, 2023. – С. 284 289.

3. *Moiseev A., Salimzyanov R.* Asymptotic analysis of mathematical model of subscriber communication network based on IAB technology // Информационные технологии и математическое моделирование (ИТММ-2023) : материалы XXII Междунар. конф. имени А.Ф. Терпугова, Томск, 4–9 дек. 2023 г. – Томск, 2023. – Ч. 1 –С. 158–163.

4. *Эльберг М.С., Цыганов Н.С.* Имитационное моделирование : учеб. пособие. – Красноярск : Сиб. федер. ун-т, 2017. – 128 с.

5. *Рамбо Дж., Блаха М.* UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд. – СПб.: Питер. – 2007. – 544 с.

6. *Войтишек А.В.* Основы метода Монте-Карло. – Новосибирск: НГУ, 2010. – 108 с.

АВТОРСКИЙ УКАЗАТЕЛЬ

СОДЕРЖАНИЕ

Научное издание

МАТЕРИАЛЫ
XI-й Международной молодёжной научной конференции
«МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННЫХ, ТЕХНИЧЕСКИХ И ЭКОНОМИЧЕСКИХ СИСТЕМ»

Томск, 24–27 мая 2024 г.

Под общей редакцией И.С. Шмырина

Издание подготовлено в авторской редакции

Подписано в печать 02.12.2024 г. Формат 70×108 1/16
Печ. л. 20,2; усл. печ. л. 26,3.
Тираж 500 экз. Заказ № 6139.

Отпечатано на оборудовании
Издательства Томского государственного университета
634050, г. Томск, пр. Ленина, 36
Тел. 8+(382-2) 52-98-49
Сайт: http://publish.tsu.ru; e-mail: rio.tsu@mail.ru

ISBN 978-5-907890-15-2