МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Доцент кафедры ИИТиМОИ
ФГБОУ ВО «ЮУрГГПУ», к.п.н.

_____ Л.С. Носова

«___»_____ 2024 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка модели автоматической оценки заявок некоммерческих организаций на финансирование от Правительства Челябинской области

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.04.04.2024.308-1469.ВКР

Научный руководитель,
доцент кафедры СП, к.п.н.
_____ О.Н. Иванова

Автор работы,
студент группы КЭ-229
_____ А.А. Зелениченко

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы магистра
студенту группы КЭ-229
Зелениченко Арсению Алексеевичу,
обучающемуся по направлению
09.04.04 «Программная инженерия»
(магистерская программа «Искусственный интеллект и инженерия данных»)

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка модели автоматической оценки заявок некоммерческих организаций на финансирование от Правительства Челябинской области.
2. **Срок сдачи студентом законченной работы:** 20.05.2024 г.
3. **Исходные данные к работе**

3.1. Гольдберг Й. Нейросетевые методы в обработке естественного языка: руководство. // ДМК Пресс, 2019. – 282 с.

3.2. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение. // ДМК Пресс, 2018. – 652 с.

3.3. Wang F., Beladev M., Kleinfeld O., Frayerman E., Shachar T., Fainman E., Assaraf K., Mizrachi S. Text2Topic: Multi-Label Text Classification System for Efficient Topic Detection in User Generated Content with Zero-Shot Capabilities [Электронный ресурс] // arXiv.org, 2024. Дата обновления: 23.10.2023 г. URL: https://arxiv.org/abs/2310.14817 (дата обращения: 10.02.2024 г.).

3.4. Gomaa W., Fahmy A. A Survey of Text Similarity Approaches. // International journal of Computer Applications, 2013. – Vol. 68, № 13. – 13–18 pp.

4. Перечень подлежащих разработке вопросов

4.1. Изучить предметную область, провести обзор научной литературы.

4.2. Подготовить набор данных для обучения, спроектировать архитектуры нейронных сетей и математическую модель.

4.3. Обучить нейронные сети, оценить результаты их работы.

4.4. Реализовать программную систему для оценки заявок на основе нейросетевых технологий.

4.5. Протестировать программную систему.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.п.н. О.Н. Иванова

Задание принял к исполнению А.А. Зелениченко

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современной России происходят быстрые социальные и экономические изменения, поэтому государственные социальные учреждения не всегда успевают или в недостаточной степени удовлетворяют потребности общества [1]. В связи с этим крайне важно предоставить людям возможность активно участвовать в формировании гражданских обществ и решении насущных социальных проблем, проявляющихся в форме гражданских инициатив [2]. Одним из наиболее эффективных и современных способов поддержки социальных инициатив является создание некоммерческих организаций и их финансирование при помощи грантовой системы [3].

В Челябинской области наиболее известным грантооператором, осуществляющим финансирование социальных проектов, является «Фонд поддержки инициатив Южного Урала», созданный при поддержке губернатора Челябинской области А.Л. Текслера в 2020 году. Фонд поддерживает как некоммерческие организации, так и физических лиц. Только лишь во втором конкурсе 2023 года грантами губернатора было поддержано 74 проекта на сумму более 107 млн. рублей [4].

В течение года Фондом проводится два конкурса и одна стажировка для физических лиц и некоммерческих организаций в 12 категориях. При этом наиболее трудоемким процессом проведения конкурса является оценка необходимости реализации проектов участников конкурса вследствие [5]:

- обеспечения высокого уровня открытости принятия решений;
- широкого круга категорий конкурса;
- значительного числа участников конкурса в каждой категории;
- большого объема анализируемой информации;
- сложных качественных критериев оценивания для категории.

Актуальность

В связи с вышеперечисленными сложностями возникает необходимость в реализации специализированной системы, которая будет оценивать проекты участников конкурса согласно положению о проведении конкурса

с целью исключения ошибок при заполнении заявок. Также система должна выявлять наиболее значимые и перспективные проекты для последующей оценки экспертами, что позволит значительно ускорить время проверки всех проектов конкурса и сэкономить административные ресурсы фонда.

Также потребность в разработке системы автоматической оценки подтверждается запросом на разработку функционала Фондом, что свидетельствует о насущности рассматриваемой проблемы.

Новизна

На текущий момент в государственных информационных системах применение технологии искусственного интеллекта находится на довольно низком уровне [6]. При этом, учитывая разнородность критериев, определяемых положениями о проведении конкурсов, оценивание проектов является нетривиальной задачей, требующей особого подхода к проектированию архитектуры нейронной сети и применения различных моделей машинного обучения.

Постановка задачи

Целью выпускной квалификационной работы является разработка модели автоматической оценки заявок некоммерческих организаций на финансирование от Правительства Челябинской области. Для выполнения поставленной цели необходимо решить следующие задачи:

1) изучить предметную область, провести обзор научной литературы и существующих аналогов, осуществить декомпозицию задачи;

2) провести анализ критериев заявки и подготовить набор данных для обучения модели;

3) проанализировать и выбрать наиболее подходящие типы нейросетевых моделей и подходы оценки заявок на грант с помощью нейросетевых технологий;

4) спроектировать архитектуры нейронных сетей, математическую модель и программную систему для оценки заявок;

5) реализовать и обучить нейронные сети, математическую модель для оценки заявок, оценить результаты их работы;

6) реализовать программную систему для оценки заявок на основе нейросетевых технологий, контейнеризировать программную систему;

7) протестировать программную систему, проверить переносимость программной системы на целевую платформу Linux.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения, списка литературы и приложения. Объем работы составляет 62 страницы, объем списка литературы – 50 источников.

В первой главе рассматривается предметная область исследуемой задачи, проводится обзор теоретических работ и аналогов разрабатываемого решения, рассматривается этический аспект использования технологии искусственного интеллекта.

Во второй главе описываются процедуры и инструменты для предварительной обработки данных, декомпозиция задачи на подзадачи, описание архитектур, которые могут быть использованы для решения подзадач. Также в главе проводится концептуальное рассмотрение и формирование требований к подзадачам, описание математической модели и проектирование программной системы.

В третьей главе содержится реализация синтаксического анализатора для сбора данных, разработка функций предобработки и визуализации данных, программная реализация нейронных сетей для классификации текстов и выявления семантического сходства, математической модели. Также в главе рассмотрена реализация клиентской и серверной части и принцип контейнеризации программной системы.

В четвертой главе описывается тестирование основных функций программной системы.

В приложении содержится структура заявки на получение гранта.

1. ОБЗОР ЛИТЕРАТУРЫ

1.1. Обзор литературы

Социальные проекты отличаются своей организационной структурой, предлагаемыми достижениями и непредсказуемостью получаемых результатов. Поэтому для реализации или продолжения развития социально значимых инициатив некоммерческих организаций и группами активных граждан наиболее острым является вопрос получения финансирования [7].

Существует множество источников финансирования социальных инициатив, такие как фандрайзинг, эндаумент, субсидии, краудфандинг и другие, однако наиболее эффективным инструментом поддержки является выделение финансовой помощи в виде грантов [7].

При этом предоставление грантов, как правило, осуществляется на конкурсной основе, что влечет за собой обработку множества заявок с различными критериями. Отметим, что на текущий момент не существует единой нормативно-правовой базы для проведения конкурсов, что приводит к усложнению процесса подачи заявок грантополучателями [8].

Грантодателю же необходимо обработать большое количество заявок от потенциальных грантополучателей, оценить необходимость и эффективность предложенных проектов, выбрать наиболее приоритетные и актуальные проекты. Также выделение грантовых средств несет сопутствующие риски, поскольку на основании проведенного анализа грантодатель выносит вердикт и надеется, что исполнитель надлежаще выполнит обещанные проекты в надлежащий срок без запроса дополнительного финансирования. Поэтому процесс оценки документов является чрезвычайно важным и требует привлечения специалистов различного уровня [8].

В работе [9], анализирующей проблему экспертного оценивания грантов в научно-исследовательской сфере (гранты РФФИ и РГНФ), автор отмечает важность рассмотренных ранее проблем, особенно упоминая важность экспертной оценки, которая основывается на принципиальном противоре-

чии между субъективным и объективным подходами в оценке проекта экспертом. Немаловажным является трудоемкость подготовки экспертного заключения, состоящего из суммы баллов всех критериев, оцениваемых в заявке, средний срок реализации которого составляет 12 дней. При этом автор отмечает, что некорректно заполненные заявки вследствие влияния человеческого фактора могут сказаться на процессе рассмотрении заявки.

В работе [9] анализируется три вида обработки заявок (рисунок 1), на основании которых авторы рекомендуют использовать гибридный метод и предлагают собственную схему сопровождения заявки, где искусственный интеллект контролирует:

– формально-логический контроль, заключающийся в проверке правильности реализации формата в записях, нахождении и выделении дублирующихся заявок, анализе правильности заявки запросами семантического ядра и лингвистическим контролем;

– регистрацию заявок;

– экспертизу, заключающуюся в определении нейронной сетью соответствия проекта тематике, актуальности, плану поставленных задач, в проведении анализа приложенной информации об участниках и их научного потенциала.

Рисунок 1 – Методы сопровождения грантовых заявок

Предложенная методика анализа заявок использует технологии обработки естественного языка, а именно составляет семантическое ядро и проводит лингвистический анализ текста. В качестве примера авторами указан отбор по грантам с помощью лингвистического подхода на основе словаря LIWC5 [9], основанного на частоте повторения слов в заявлении.

В исследовательской работе [10] проводится проектирование модуля оценки актуальности заявок автоматизированной системы распределения грантов (АСРГ). В рамках данного модуля предполагается выполнение двух этапов: подготовительного и аналитического (рисунок 2).

Рисунок 2 – Схема анализа заявки в АСРГ [10]

В аналитическом этапе исследуемой работы рассмотрим задачу отраслевой идентификации, которая подробно описана в работе [11] и решается при помощи математической модели и статистического метода TF-IDF, исключая использование машинного обучения или нейронных сетей, поскольку автор работы считает, что задача должна быть строго формализована и может быть выполнена лишь при помощи математических операций с базой термов [11, с.4]. Однако фактическая реализация данного модуля, как и всей системы АСРГ, отсутствует в открытых источниках.

Следовательно, можно заключить, что теоретический аспект исследуемой задачи проработан в достаточной степени. Внедрение системы с технологией искусственного интеллекта позволит экспертам исключить несоответствия данных заявления, упростить механизм взаимодействия и сократить время работы с заявкой. При этом механизм оценки системы напрямую зависит от структуры заявления, платформы и схемы проведения конкурса.

1.2. Описание предметной области

Согласно политике цифровой трансформации, проводимой в Челябинской области, взаимодействие грантополучателя и грантодателя в процессе подготовки, экспертизы и приема отчетности по итогам реализации проектов осуществляется в электронном виде [5].

Сценарий проведения конкурсов состоит из четырех этапов.

1. Заполнение и подача заявок в электронном виде, проверка на соответствие критериям конкурса и регистрация заявок.
2. Экспертная оценка заявки в составе не менее двух экспертов в соответствии с положением о проведении конкурса.
3. Формирование рейтинга проектов на основании результатов экспертной оценки объединенного экспертного совета.
4. Подведение итогов конкурса, установка пороговых значений рейтинга для победы в конкурсе, утверждение списка победителей.

Направления, по которым проводится конкурс среди некоммерческих организаций и физических лиц, представлены в таблице 1.

Таблица 1 – Типы конкурсов

№	Наименование
1	Социальное обслуживание, социальная поддержка и защита граждан
2	Охрана здоровья граждан, пропаганда здорового образа жизни
3	Поддержка семьи, материнства, отцовства и детства
4	Поддержка молодежных проектов, реализация которых охватывает виды деятельности, предусмотренные статьей 31.1 Федерального закона от 12 января 1996 года № 7-ФЗ «О некоммерческих организациях»
5	Поддержка проектов в области науки, образования, просвещения
6	Поддержка проектов в области культуры и искусства
7	Сохранение исторической памяти
8	Защита прав и свобод человека и гражданина, в том числе защита прав заключенных
9	Охрана окружающей среды и защита животных
10	Укрепление межнационального и межрелигиозного согласия
11	Развитие общественной дипломатии и поддержка соотечественников
12	Развитие институтов гражданского общества

Отметим, что структура полей в рамках одного конкурса для всех категорий одинакова и структурирована по категориям, представленным на рисунке 3.

Рисунок 3 – Структура заявки на участие в конкурсе

Согласно предметной области можно выделить ключевые аспекты:

– внедрение разрабатываемого решения между первым и вторым этапом схемы проведения конкурса;

– типы конкурсов могут использоваться в качестве классов для задачи классификации;

– структура заявки на участие в конкурсе имеет фиксированный формат, что позволяет применять решение ко всем конкурсам фонда.

1.3. Обзор аналогов

В работе [12] представлена исследовательская модель оценки заявок на предоставление грантов федеральным ВУЗам и государственным учреждениям из бюджета города Москвы с использованием алгоритма логистической регрессии и метода векторизации «Word-bag». В результате работы авторы пришли к результату, что проведенное исследование и полученная модель с точностью 68% станет прототипом для создания рекомендательной системы. Также авторы отмечают, что существуют потенциальные пути развития модели: использование иных алгоритмов и дополнительных параметров исходного набора для анализа.

В работе [13] разработана система классификации текстов по научным специальностям с помощью алгоритма логистической регрессии и метода векторизации «TF-IDF». Поскольку автором указывается, что в наборе данных фактически используется четыре признака и модель «TF-IDF» из-за отсутствия вычислительных мощностей, точность 87% может быть увеличена на несколько процентов. В качестве выходных данных система выводит процент уверенности предсказания для каждого класса.

В работах [10, 11] разработана часть автоматизированной системы распределения грантов, в которой с помощью математической модели оцениваются заявки на проектное финансирование. Данные работы наиболее плотно коррелируют с поставленной задачей, однако, следует заметить, что автор использует строгое концептуальное математическое обоснование для

определения актуальности и новизны заявки [14]. При этом автором работ указывается, что система показывает очень высокие результаты, однако подтверждающих данных ни в одной из работ не приводится.

В качестве иностранного аналога можно выделить сервис SmartSimple cloud + AI [15], который позволяет оценивать грантовые заявки. Суть сервиса заключается в проверке содержания заявки в соответствии с критериями и требованиями программы, выставлении оценки и формирования краткого отчета на основе архитектуры GPT, объясняющего причину выставления оценки. При этом в официальных источниках не указано, каким образом происходит анализ заявки, а также не обозначены обучающий датасет и точность предсказания, следовательно, оценить сервис в полной мере не представляется возможным. Результаты обзора аналогов представлены в таблице 2.

Таблица 2 – Сравнительный обзор аналогов

Задача	**Работа**	**Количество признаков**	**Размерность датасета**	**Векторизация**	**Точность**
Бинарная классификация	[12]	4	2903	Мешок слов	66%
Многоклассовая классификация	[13]	6	10685	TF-IDF	87%
Бинарная классификация	[10]	–	–	–	–
Классификация	[15]	–	–	–	–

На основании полученных результатов можно сделать вывод, что в отечественных разработках комплексная оценка заявок на финансирование с использованием нейронных сетей не реализовывалась, прорабатывались лишь теоретические аспекты по реализации данного функционала, что подкрепляет актуальность разработки программной системы.

Этический аспект

Использование технологии искусственного интеллекта позволяет экспертам и членам комиссии получить такие преимущества, как:

– беспристрастная оценка заявлений;

– быстрота, качество и скорость принятия решений;

– исключение человеческого фактора.

Однако существуют риски возникновения этических проблем:

– объективность / прозрачность;

– справедливость / дискриминация;

– предвзятость / отсутствие человеческого суждения в процессах принятия решений;

– подотчетность.

Так, например, в [16] указывается, что в государственных ведомствах, крупных научных изданиях и журналах запрещается использование больших языковых моделей, поскольку они не обеспечивают конфиденциальность данных (передаваемые данные включаются в обучающую выборку), а обзоры, написанные с помощью технологии искусственного интеллекта, будут содержать ошибки и неточности по отношению к тезисам и суждениям, которые не были заложены в модель.

Следовательно, разрешение и поддержка рассмотренных этических проблем заключается в обеспечении прозрачности алгоритмов принятия решений, наличии четких критериев оценки в соответствии с нормативно-правовыми актами, регулярном контроле на предмет предвзятости, привлечении сторон к ответственности за решения модели.

В результате выполнения главы была теоретически обоснована необходимость создания помощника для оценки заявок на грант с применением технологии искусственного интеллекта, было приведено описание предметной области и рассмотрены аналоги разрабатываемого продукта, которые частично удовлетворяют требованиям поставленной задачи.

2. ПРОЕКТИРОВАНИЕ

Согласно результатам анализа, проведенного в первой главе, задача оценки заявок на грант является нетривиальной, что приводит к необходимости дополнительного и тщательного исследования при проектировании системы. В связи с этим, а также с целью систематизации и структурирования излагаемого материала, произведем декомпозицию задачи.

Декомпозиция задачи

Для того чтобы оценить релевантность заявки, необходимо решить следующие подзадачи:

1) определение тематики заявки;
2) определение степени корреляции текстов;
3) реализация математической модели оценки релевантности заявки.

Прикладная область задачи: обработка естественного языка (Natural Language Processing, NLP) – область искусственного интеллекта, которая фокусируется на проблемах, связанных с пониманием и обработкой человеческого языка: восприятие синтаксической структуры и контекста, анализ настроений, машинный перевод, идентификация и классификация объектов в тексте, извлечение информации, обобщение текста [17].

Сведем каждую подзадачу к некоторому абстрактному классу задач.

1. «Определение тематики заявки» – многоклассовая классификация текста (multiclass text classification): классификация экземпляров в один из множества счетных классов на основе некоторой вероятности принадлежности к одному классу [17, с.38].

2. «Определение степени корреляции текстов» – семантическое сходство текста (text similarity): определение полноты связи двух фрагментов текста по смыслу или контексту на основе семантических отношений между словами или фразами [17, с.148].

3. «Реализация математической модели релевантности гранта» – концептуальное проектирование математической модели: строгое опреде-

ление алгоритма, критериев и правил количественной оценки объекта на основе совокупности гипотез о свойствах объекта и массива зависимых качественных входных параметров.

Этапы обработки текста

В общем случае задача обработки текста подразделяется на этапы, представленные на рисунке 4.

Рисунок 4 – Этапы обработки текста (в общем случае)

Для решения подзадач необходимо определить инструменты реализации этапов обработки текста.

1. Этап «Подготовка»: вид токенизатора, методы, правила очистки текста, морфологический анализатор, вид векторизации текста.
2. Этап «Модель»: архитектура модели и метрики.

2.1. Описание и анализ датасета

В работе был использован самостоятельно собранный набор данных на основе федерального ресурса «Оценка результатов проектов» [18], содержащих данные о 25000 реализованных проектах за 2020–2023 годы. Сбор

данных осуществлялся с помощью специально разработанного синтаксического анализатора [19]. Структура атрибутов, содержащихся в наборе данных, представлена в форме таблицы 3.

Таблица 3 – Структура атрибутов датасета

№	Наименование	Атрибут	Описание поля	Среднее кол-во токенов
1	Ссылка	Reference	Прямая ссылка на карточку проекта ресурса «оценка результатов проектов»	–
2	Направление	Category	Тематика заявки в соответствии с положением о проведении конкурса	3
3	Название	Application Name	Наименование заявки	7
4	Краткое описание	Short Description Raw	Краткий обзор, описывающий целевую аудиторию, ключевые аспекты, ожидаемые результаты проекта	256
5	Дата начала	Start Date	Дата начала реализации проекта	1
6	Дата окончания	Final Date	Дата окончания реализации проекта	1
7	Социальная значимость	Social Description Raw	Потенциальное положительное влияние проекта на общество или конкретное сообщество	308
8	Целевые группы	Social Groups Raw	Категории получателей результатов реализации проекта	13
9	Цель	Aims Raw	Качественная формулировка того, чего заявитель стремится достичь посредством проекта	25
10	Задачи	Tasks Raw	Последовательность шагов, которые необходимо предпринять заявителю для достижения цели	57
11	Качественные результаты	Quality Result	Конкретные качественные изменения, которые произошли в результате реализации проекта	111
12	Оценка результатов	Evaluation	Оценка результатов реализации проекта согласно целям и задачам, а также полученного социального эффекта	196

Согласно таблице 3 определим анализируемые пары атрибутов:

– в рамках подзадачи «Многоклассовой классификации текста»: «Направление – Название», «Направление – Краткое написание», «Направление – Цель», «Направление – Задачи»;

– в рамках подзадачи «Семантического сходства текста»: «Социальная значимость – Цель», «Социальная значимость – Задачи», «Социальная значимость – Качественные результаты».

2.2. Предобработка данных

Поскольку алгоритмы машинного обучения и нейронные сети не могут работать с данными на естественном языке, необходимо определить процедуры и методы предварительной обработки данных [20].

Далее необходимо проанализировать каждый из этапов схемы предобработки данных, представленной на рисунке 5.

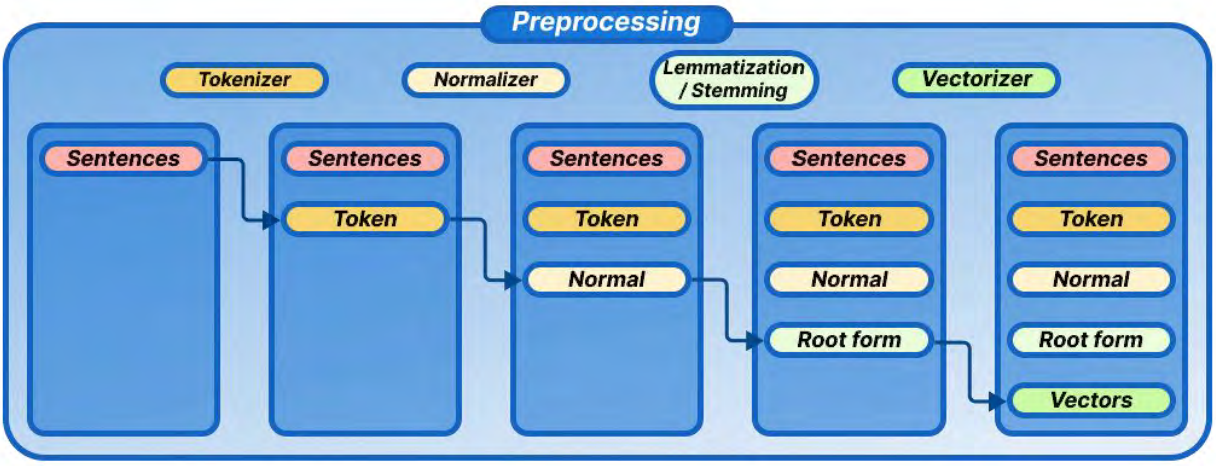

Рисунок 5 – Этапы предобработки данных

Токенизация (tokenization) [21]

Токенизация – процесс сегментации (разбиения) текста на более мелкие единицы, называемые токенами, которые могут быть словами, фразами, символами или другими значимыми синтаксическими элементами.

Целью токенизации является преобразование неструктурированных текстовых данных в формат, наиболее пригодный для обработки и алгоритмического анализа.

Нормализация (normalization) [21]

Нормализация – серия операций, в результате которых текст приводится к стандартной словарной форме. Нормализация включает:

– преобразование текста к нижнему регистру (lowercasing);
– удаление знаков препинания и чисел (punctuation removal);
– обработку сокращений (abbreviations processing);
– удаление стоп слов (stop-words removal);
– группировку нескольких слов в один токен (multiword grouping);

– приведение к корневой форме слова (reduction to the root form): лемматизация и стемминг;

– другие методы нормализации.

Целью нормализации является уменьшение избыточности и улучшение целостности данных за счет устранения дублирующейся и незначимой информации, а также эффективная организация данных.

Лемматизация (lemmatizing) [21]

Лемматизация – процесс приведения слов к корневой форме (лемме) путем рассмотрения значения и контекста слова, в результате которого снижается вариативность одного и того же слова.

Стемминг (stemming) [21]

Стемминг – процесс приведения слов к корневой форме (стем) путем:

– удаления префиксов, суффиксов, окончаний;

– усечения окончаний;

– поиска по заданному словарю.

Сравнительный анализ методов приведения к корневой форме слова представлен в таблице 4.

Таблица 4 – Сравнительный анализ лемматизации и стемминга

№	Сравнительные параметры	Технология	
		Лемматизация	Стемминг
1	Быстрота вычисления	–	+
2	Точность вычисления	+	–
3	Независимость от морфологических правил языка	–	+
4	Сохранение контекста и семантического значения	+	–
5	Простота реализации	–	+
6	Использование лингвистических правил	+	–
7	Возможное создание несуществующих слов	–	+
8	Подходит для языков с богатым словоизменением	+	–

На основании таблицы 4 можно сделать вывод, что наиболее подходящим методом приведения к корневой форме слова в рамках исследуемой задачи является лемматизация, так как в анализе используются атрибуты с сильными контекстуальными и семантическими зависимостями, а также потому, что лемматизация имеет более точное приведение к корневой форме.

Векторизация (vectorization) [21]

Векторизация – процесс преобразования текстовых данных в числовые векторы, которые могут быть использованы в качестве входных данных для моделей машинного обучения и нейронных сетей.

Существуют следующие техники векторизации [21]: bag of words, TF-IDF, word embeddings.

Bag of words

Мешок слов (bag of words) – простой метод обработки естественного языка, при котором текст представляется как неупорядоченный набор слов без учета грамматики и порядка слов. Мешок слов легко реализовать и понять, но при этом техника игнорирует порядок слов и контекст, плохо справляется со словами, которых нет в словаре, а также требует большого объема памяти из-за разряженного матричного представления.

TF-IDF

TF-IDF (частота термина – обратная частота документа) – статистическая мера, используемая в обработке естественного языка для оценки важности слова в документе по отношению к коллекции документов или корпусу текстов. Мера решает проблему «зашумления» важных редко встречающихся слов слишком часто повторяющимися словами, которые не несут смысловой нагрузки, такие как предлоги и союзы. Техника хорошо справляется с синонимами и общеупотребляемыми словами, однако игнорирует контекст, порядок слов, семантику, а для эффективной работы ей требуется достаточно большой корпус текстов.

Word embeddings

Встраивание слов (word embeddings) – векторное представление слов в непрерывном векторном пространстве, полученное из больших текстовых корпусов. Встраивание слов позволяет уловить смысловые связи между словами, уменьшить размерность и разреженность текстовых данных, но, при этом, препятствует интерпретируемости отдельных измерений в пространстве встраивания.

Word2Vec

Word2Vec [22] – метод обработки естественного языка в многомерном векторном пространстве, который использует локальную информацию в пределах заданных размеров окон для создания вложений слов, ограничивая семантическое поле окружающими словами в исходном предложении. Метод имеет два основных способа реализации (рисунок 6).

1. Skip-Gram – модель, которая предсказывает контекстные слова по заданному слову, эффективна для редких слов с небольшими наборами данных. Основной целью модели является изучение весов скрытого слоя по мере того, как корректируются спрогнозированные веса окружающих слов.

2. CBOW (Continuous bag of words) – модель, которая предсказывает целевое слово на основе слов его контекста, полезна для обучения на больших наборах данных, работает быстрее по сравнению с Skip-gram.

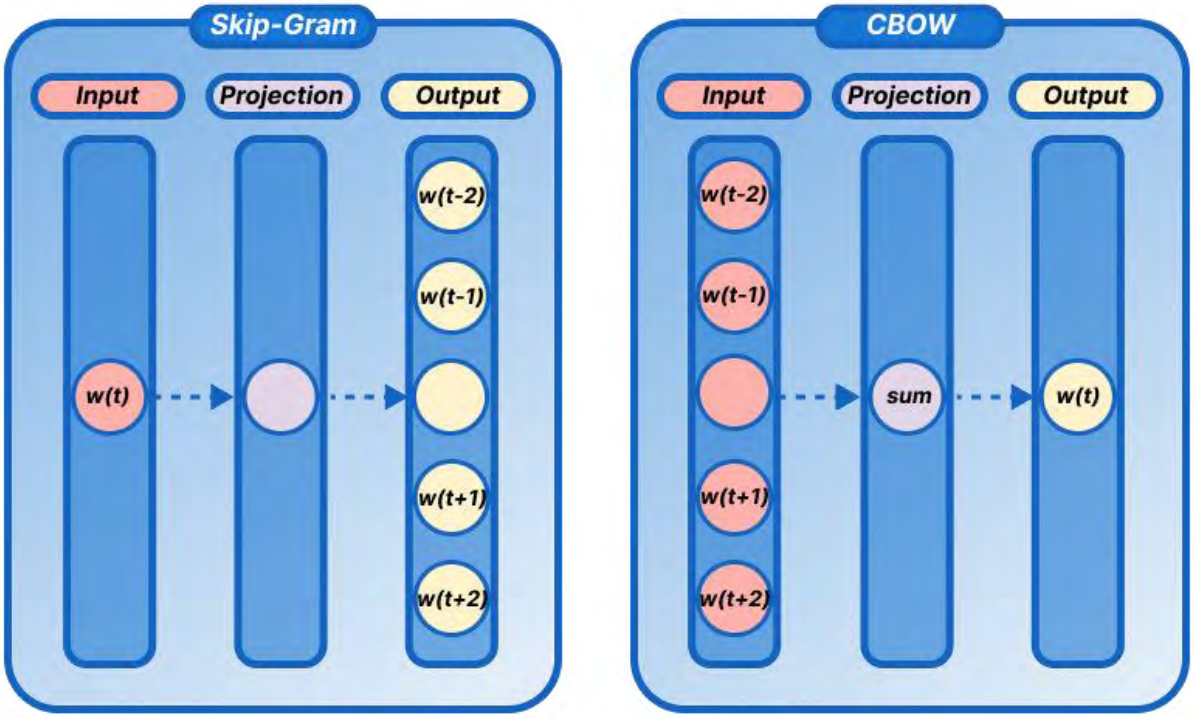

Рисунок 6 – Архитектуры Skip-gram и CBOW

GloVe

GloVe (Global Vectors for Word Representation) [23] – модель встраивания слов, которая изучает векторные представления слов на основе глобаль-

ной статистики совместного появления слов в корпусе текста. Модель минимизирует разницу между произведением векторов слов и логарифмом вероятности их совместного появления методом стохастического градиентного спуска, создавая представления, фиксирующие важные линейные подструктуры векторного пространства слов.

FastText

FastText [24] – библиотека с открытым исходным кодом для эффективного изучения представлений слов и классификации предложений, разработанная лабораторией Facebook AI Research. К основной модели Word2Vec добавлена модель символьных N-грамм, в результате чего эмбеддинги слов получаются не напрямую, а через комбинацию векторов более низкого уровня. Эта комбинация позволяет использовать меньше данных для обучения, поскольку слово само по себе становится контекстом.

Преимущества и недостатки методов векторизации сведены в сравнительную таблицу 5, на основании которой можно сделать вывод, что наиболее подходящим методом является встраивание слов в связи с эффективной фиксацией семантических отношений во вложениях слов.

Таблица 5 – Сравнение технологий векторизации [22–24]

№	Сравнительные параметры	Технология				
		Bag of words	**TF-IDF**	**Word2Vec**	**GloVe**	**FastText**
1	Простота архитектуры	+	+	+	+	–
2	Быстрота обучения	+	+	–	+	–
3	Обучение на уровне слов	–	–	+	+	+
4	Осмысленные эмбеддинги	–	–	+	+	+
5	Работает на редких словах	–	–	–	–	+
6	Наличие информации о контексте	–	–	–	–	–
7	Обработка больших текстовых корпусов	–	–	–	+	+

Выбор инструментов предобработки данных

Инструменты, выбранные для реализации этапа обработки текста, представлены в таблице 6.

Таблица 6 – Инструменты для реализации этапа обработки текста

№	Стадия	Задача «Многоклассовая классификации текста»	Задача «Семантическое сходство текста»
1	Предобработка данных	Модуль stopwords библиотеки NLTK [25], regex и стандартные функции языка python	
2	Лемматизация	Морфологический анализатор pymorphy3 [26]	
3	Токенизация	Модуль токенизации библиотеки NLTK / Autotokenizer	
4	Векторизация	Word embeddings	
5	Оценка качества обучения	Accuracy [27]	
6	Функция потерь	Categorical entropy	Triplet Loss
7	Тип обучения	Обучение с учителем	Обучение без учителя

Следующим шагом является определение типа модели для каждой задачи. Для этого рассмотрим и изучим основные архитектуры, которые могут использоваться для реализации поставленных задач.

2.3. Выбор архитектуры модели

Выбор архитектуры нейронной сети зависит от следующих факторов:

– сложность задачи и размер набора данных;

– доступность вычислительных ресурсов;

– специфика манипулируемого языка.

Рассмотрим основные архитектуры и опишем преимущества и недостатки архитектур в рамках исследуемых подзадач.

CNN

Сверточная нейронная сеть (CNN) – архитектура ИНС, предназначенная для обработки пространственных отношений с использованием сверточных слоев. В рамках прикладной области CNN применяется для решения задач классификации текста (обнаружение спама), анализа настроений. На вход сеть принимает embedding слов, далее применяются сверточные фильтры для улавливания локальных шаблонов в тексте, операции уменьшения размерности признаков, на выходе используется полносвязный слой и функция активации softmax (рисунок 7). Однако использование CNN сопряжено

с проблемами вычислений и интерпретации, а производительность может быть недостаточной, когда данные ограничены [28].

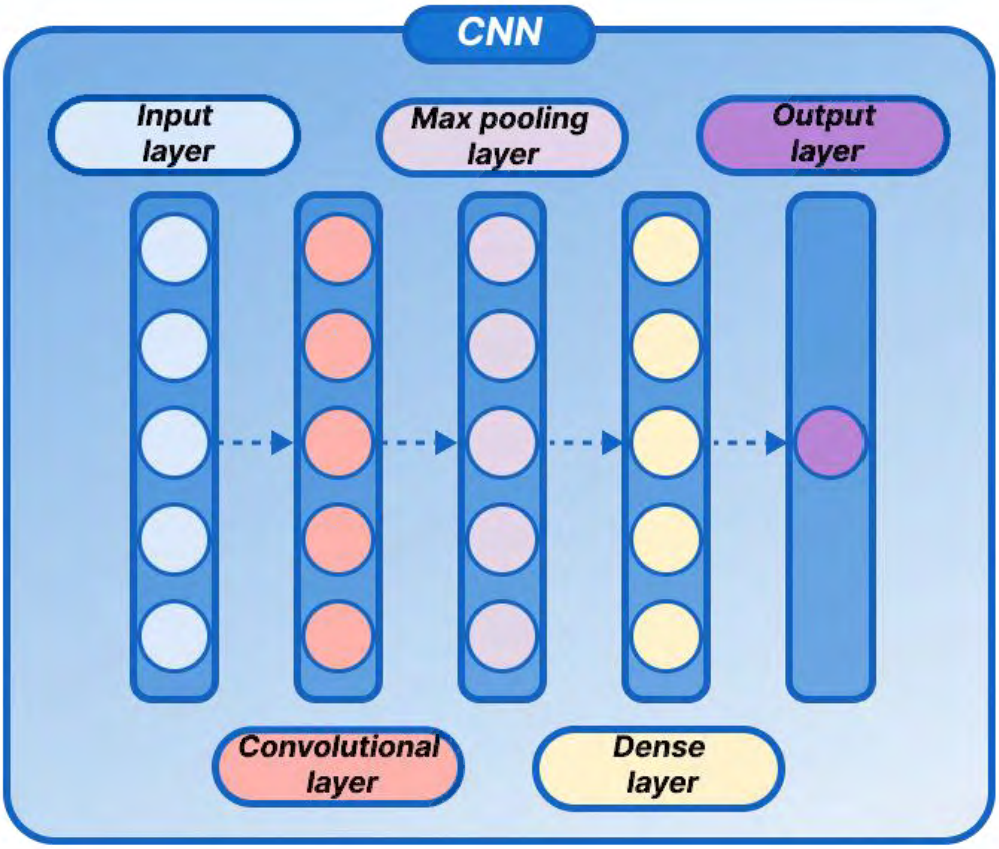

Рисунок 7 – Архитектура CNN

RNN

Рекуррентная нейронная сеть (RNN) – архитектура ИНС, предназначенная для обработки последовательных данных или данных временного характера. В отличие от традиционных ИНС, RNN поддерживает внутреннее состояние, которое позволяет обрабатывать последовательности различной длины (рисунок 8).

В рамках прикладной области RNN применяется для решения задач классификации текста при помощи обучения с обратным распространением ошибки во времени и схожести текстов при помощи фиксации зависимостей в тексте. Однако на их производительность могут оказывать влияние вычислительная сложность и ограничения памяти [29].

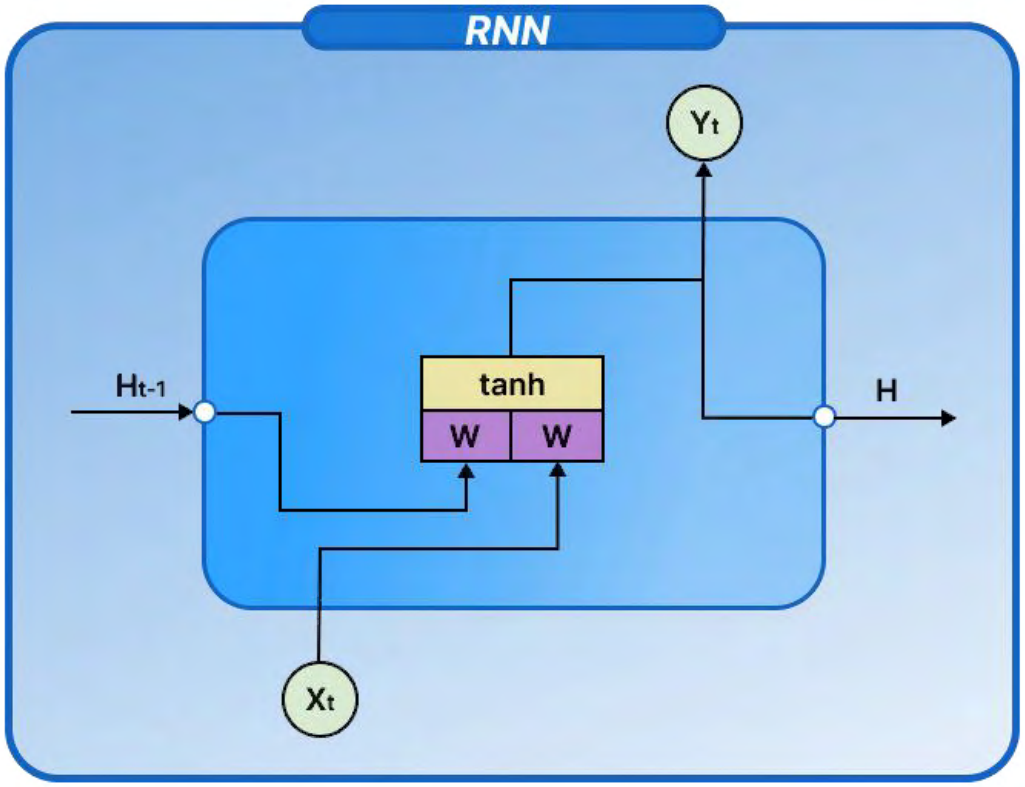

Рисунок 8 – Архитектура RNN

LSTM / GRU

Долгосрочная краткосрочная память (LSTM) или вентилируемый рекуррентный блок (GRU): варианты архитектуры RNN, предназначенные для решения проблемы исчезающего градиента и учета долгосрочных зависимостей (рисунок 9).

В LSTM баланс между долговременной и кратковременной памятью имеет решающее значение для эффективного выполнения задачи классификации и схожести текстов. Долговременная память обеспечивает базу знаний, а кратковременная память помогает в непосредственной обработке и манипулировании информацией [30].

GRU, в свою очередь, представляет механизм шлюзования, который позволяет информации течь напрямую от входа к скрытым состояниям, минуя повторяющиеся соединения, что улучшает поток информации по сети и помогает фиксировать долгосрочные зависимости во входной последовательности. Однако GRU требует тщательной настройки параметров и может быть более сложным по сравнению с RNN [31].

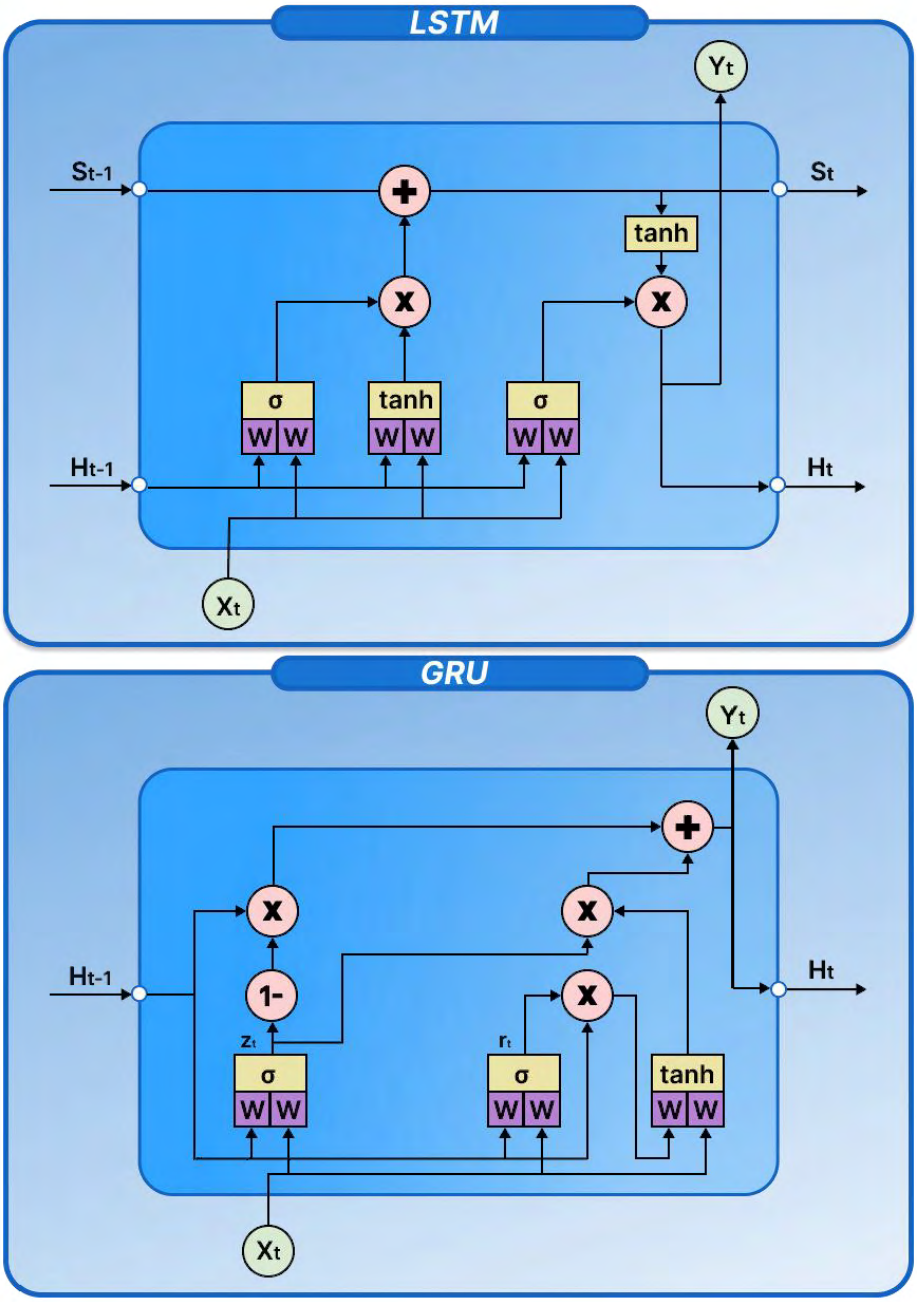

Рисунок 9 – Архитектура LSTM и GRU

Transformers

Трансформеры (transformers) – тип архитектуры ИНС, которые предназначены для решения различных задач обработки естественного языка (NLP), включая классификацию текста. Они построены на механизме внимания, позволяющем улавливать контекстуальные связи между словами и улучшать понимание смысла предложений (рисунок 10). Однако трансформеры могут быть дорогостоящими в вычислительном отношении и требовать значительного количества размеченных данных [32].

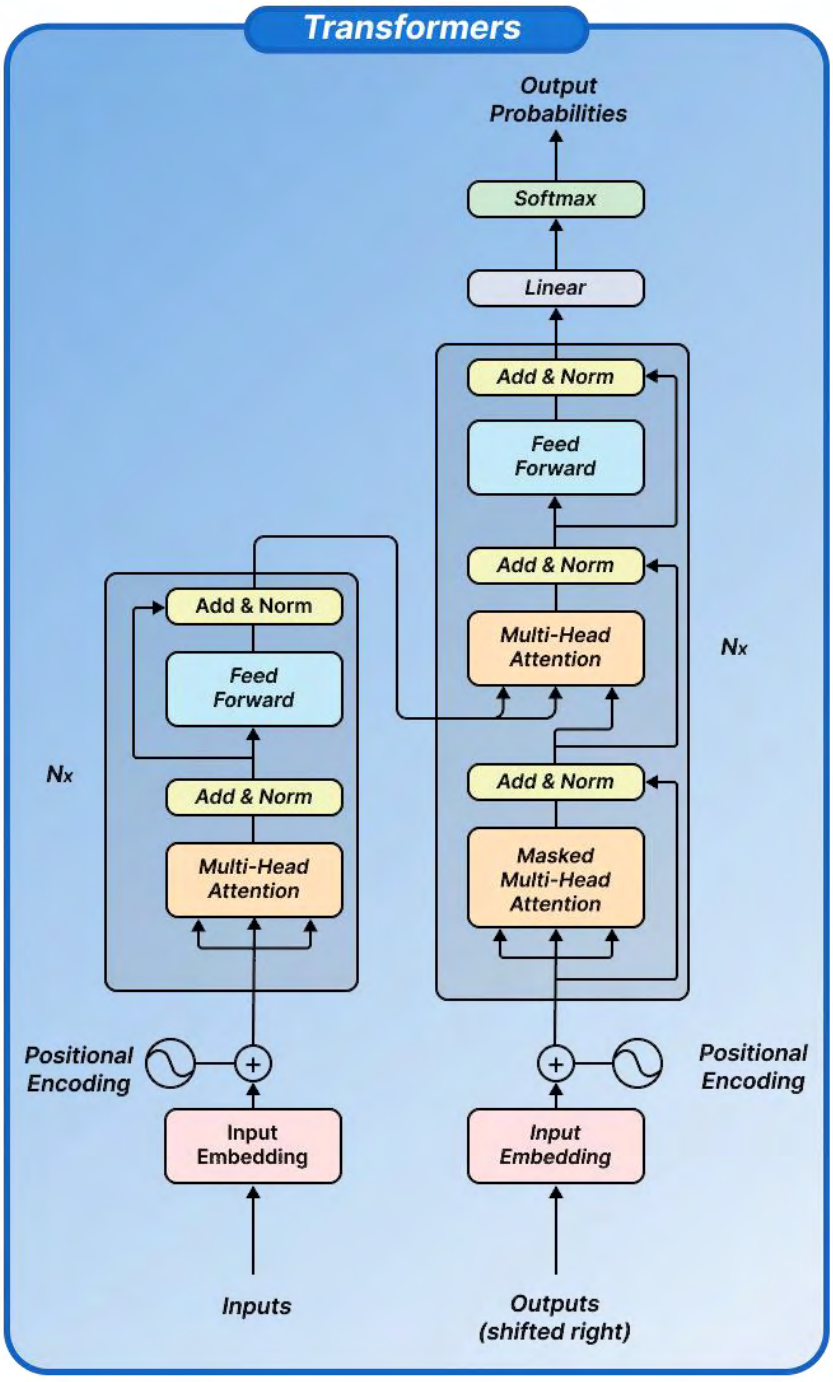

Рисунок 10 – Архитектура Transformers

Решение задач классификации и семантического сходства текстов может быть осуществлено при помощи языковых моделей GPT и BERT. В контексте рассматриваемых задач наиболее целесообразным является выбор языковой модели BERT вследствие ее двунаправленной природы.

BERT (Bidirectional Encoder Representations from Transformers) [33] – языковая модель, основанная на композиции кодировщиков трансформера,

для каждого из которых применяется двустороннее внимание, что позволяет модели учитывать контекст с обеих сторон от рассматриваемого токена.

Двустороннее внимание делает архитектуру подходящей для задач предсказания текста относительно всего объема входных данных, например, при классификации предложений.

Siamese neural network

Сиамские нейронные сети (siamese neural network) – тип архитектуры ИНС, используемый для задач изучения и сравнения сходства. Они состоят из сетей-близнецов, которые имеют одинаковые веса и учатся создавать аналогичные вложения для аналогичных входных данных (рисунок 11).

Сиамские нейронные сети имеют несколько преимуществ для задач классификации тестов, таких как обработка несбалансированных данных и обучение в несколько этапов. Однако они также имеют недостатки, связанные с вычислительной сложностью, требованиями к помеченным парам, интерпретируемостью и переоснащением [34].

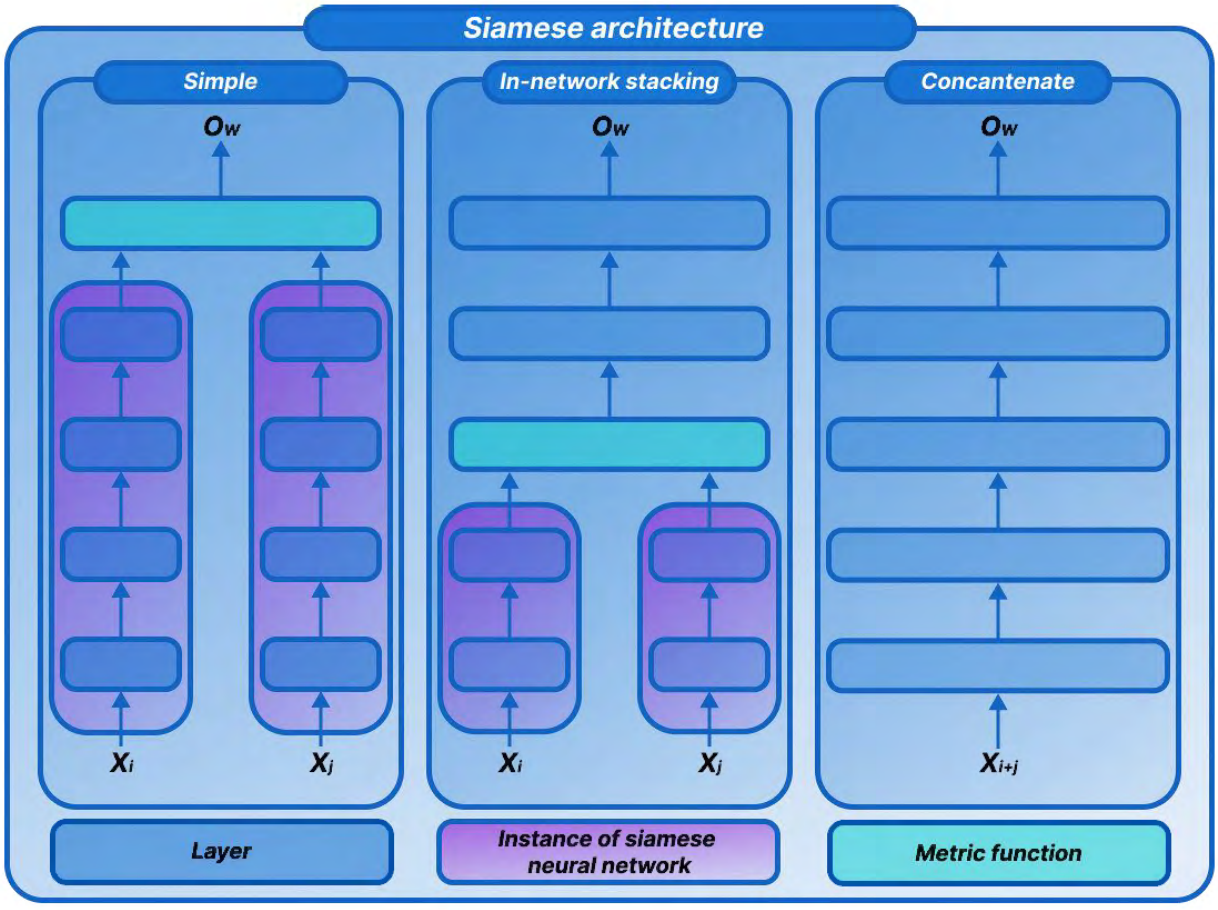

Рисунок 11 – Архитектура сиамских нейронных сетей

Сведем преимущества и недостатки рассмотренных архитектур нейронных сетей в рамках задачи классификации текста в таблицу 7.

Таблица 7 – Сравнение архитектур нейронных сетей в рамках задачи классификации текста

№	Сравнительные параметры	Технология				
		CNN	RNN	LSTM	GRU	BERT
1	Устойчивость к проблемам исчезновения и взрыва градиента	–	–	–	–	+
2	Обучение на небольшом наборе данных	–	–	–	–	+
3	Быстрое обучение	+	+	+	+	–
4	Захват контекстной информации	–	+	+	+	+
5	Фиксирование долгосрочных зависимостей	–	+	+	+	+
6	Низкие вычислительные затраты	+	+	+	+	–
7	Высокая интерпретируемость	+	+	–	–	–

Согласно таблице 7 и приведенного ранее теоретического материала можно заключить, что:

– для задачи «многоклассовой классификации текста» наиболее подходящими архитектурами являются LSTM, GRU, а также языковые модели, основанные на архитектуре трансформер (BERT, GPT);

– для задачи «семантического сходства текста» наиболее подходящей архитектурой является Siamese neural network, поскольку архитектура позволяет обрабатывать несбалансированные наборы данных и устойчива к изменениям наборов данных.

2.4. Задача классификации текстов

Классификация текста – фундаментальная задача NLP, целью которой является автоматический анализ и категоризация (присвоение) предопределенных меток (категорий) текстовым данным на основе их содержимого.

Задача классификации текстов подразделяется на типы:

– бинарная (binary classification) – два класса объектов, объект может принадлежать только одному классу;

– многоклассовая (multiclass classification) – несколько классов объектов, объект может принадлежать только одному классу;

– многозначная (multilabel classification) – несколько классов объектов, объект может принадлежать нескольким классам.

Поскольку задача классификации текста относится к типу многоклассовой (multiclass classification), перечислим основные архитектуры, которые могут быть использованы для обучения:

– машинное обучение: Logistic Regression, Naive Bayes, k-Nearest Neighbor, Decision Tree, Random Forest, Extreme Gradient Boosting, Support Vector Machine и другие;

– нейронные сети: RNN, CNN, LSTM, GRU и другие;

– трансформеры с моделью классификации: BERT, GPT и другие.

Из представленных архитектур наиболее перспективными являются CNN, LSTM, GRU и языковая модель BERT, в связи с чем необходимо провести дополнительные исследования и эксперименты для выявления наиболее результативной архитектуры.

На основании анализа можно определить требования к задаче:

– количество классов: 12 классов атрибута «Направление» исходного датасета (согласно таблице 1);

– исследуемые архитектуры: RNN, LSTM, GRU, BERT;

– датасет: 25000 записей, сформированные объединением данных из столбцов «Краткое написание», «Цель», «Задачи».

2.5. Задача семантического сходства

В классической формулировке задача семантического сходства текстов включает представление текстов в виде числовых векторов в многомерном пространстве, а затем измерение сходства между этими векторами с помощью специальных мер. Поскольку исследуемая проблема проработана теоретически, рассмотрим научные работы с целью выявления наиболее оптимального подхода к реализации задачи.

В работе [35], рассматривающей алгоритмы сравнения схожести текстов на основе нейросетевых алгоритмов, была проведена серия опытов с

собственными и предобученными моделями. В результате автор пришел к выводу, что наиболее рациональным решением является использование модели paraphrase multilingual MiniLM L12 v2 благодаря относительно низкой размерности векторов внутреннего представления и высокой корреляции с целевой переменной. Также автор рекомендует провести тонкую настройку гиперпараметров модели при дообучении (слои: два скрытых слоя, функция активации: ReLU, оптимизатор: моменты Нестерова, мера схожести: евклидово расстояние).

В работе [36], рассматривающей определение схожести специализированных текстов на естественном языке, автор реализовал собственную модель на основе сиамской нейронной сети с тремя подсетями LSTM и тремя полносвязными (Dense) слоями, использовав технологию векторизации текста Word2Vec. В результате автор методом кросс-валидации получил результат со средней точностью 0,8564.

Исходя из рассмотренных работ, можно сделать вывод, что создание и обучение модели с нуля не является целесообразным, поскольку в свободном доступе имеются предобученные модели, которые были обучены на большом корпусе текстов и могут быть тонко настроены (fine-tuning) в рамках текущей задачи, что позволит сэкономить временные и вычислительные ресурсы. Использование языковой модели BERT обосновано тем, что языковая модель может лучше всего находить семантические зависимости в тексте среди всех рассмотренных моделей.

В пособии по использованию архитектуры Transformer [37, с.160] приведен алгоритм и пример настройки языковой модели BERT для регрессии пар предложений на основе датасета STS-B, который может быть использовать в качестве опорного.

Поскольку заказчиком не был предоставлен датасет для задачи семантического сходства текста, тонкая настройка модели не предоставляется возможной. В связи этим было принято решение использовать базовую предобученную модель paraphrase multilingual MiniLM L12 v2.

На основании анализа можно определить требования к задаче:

– модель: paraphrase-multilingual-MiniLM-L12-v2, основанная на технологии SentenceTransformers и языковой модели BERT [38];

– мера схожести: косинусное расстояние [37].

2.6. Математическая модель

Поскольку заказчиком не было представлено иной концепции, то будем считать, что:

– на вход математической модели подаются векторы-результаты из двух нейронных сетей;

– в качестве результата работы нейронной сети находится сумма средних значений векторов с весовыми коэффициентами 0,5.

Предложенные условия соответствуют математической модели, описываемой формулой (1), входные и выходные параметры представлены в таблице 8.

$$out(\bar{x}, \bar{y}) = \frac{1}{2n}\sum_{i=1}^{n} f(x_i) + \frac{1}{2m}\sum_{j=1}^{m} g(y_j), \quad (1)$$

где $f(x)$ – функция вызова нейронной сети, решающей подзадачу «многоклассовой классификации текста»;

$g(y)$ – функция вызова нейронной сети, решающей подзадачу «семантического сходства текста»;

x_i – параметры, подаваемые на вход функции $f(x)$, $i = \overline{0,n}$;

y_j – параметры, подаваемые на вход функции $g(y)$, $j = \overline{0,m}$;

n – количество параметров для функции $f(x)$;

m – количество параметров для функции $g(y)$.

Таблица 8 – Входные и выходные параметры модели

№	Параметр	Значение	Количество параметров
1	Вход (нейронная сеть 1)	0, 1	3
2	Вход (нейронная сеть 2)	[0, 1]	3
3	Выход	[0, 1]	1

2.7. Проектирование программной системы

Разрабатываемая программная система описывается схемой, представленной на рисунке 12.

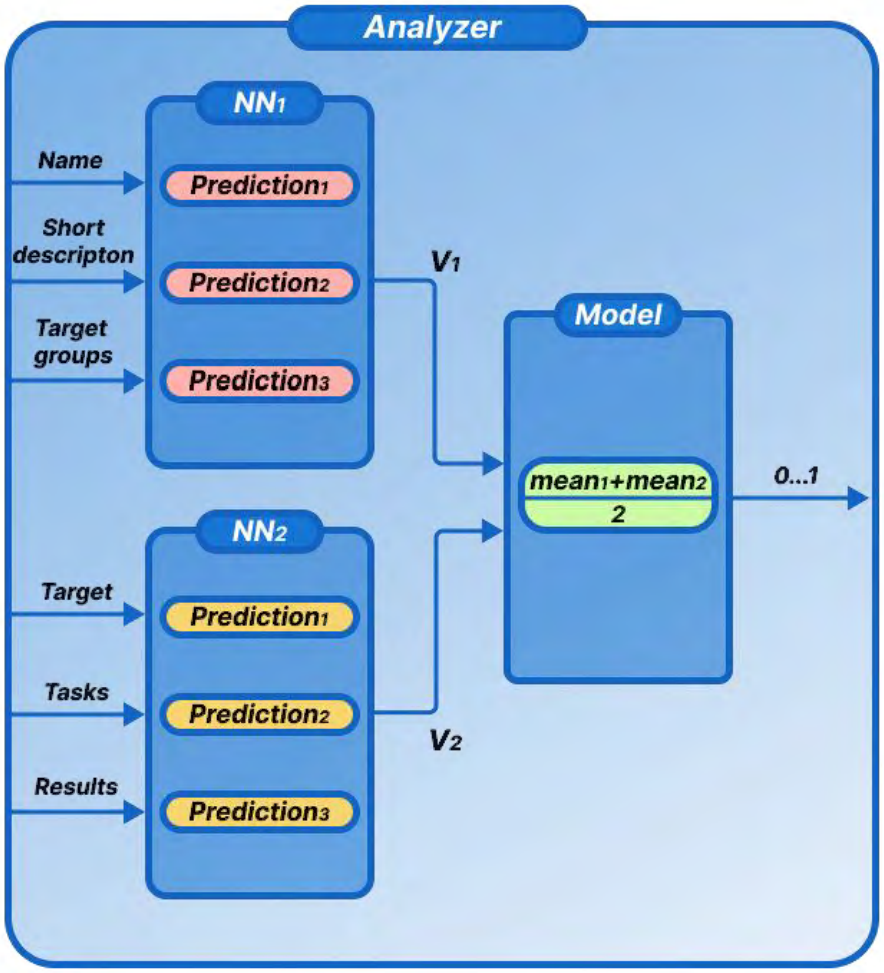

Рисунок 12 – Функциональная схема программной системы

Функциональные требования к программной системе:

– пользователь должен иметь возможность получить предсказание;

– пользователь должен иметь возможность получить статистику по предсказаниям;

– пользователь должен иметь возможность аутентификации.

Нефункциональные требования к программной системе:

– система должна быть написана на языке Python3;

– система должна использовать библиотеку PyTorch [39];

– система должна использовать предобученные нейронные сети для задачи «Семантического сходства текста»;

– система должна иметь API для взаимодействия со сторонними сервисами;

– система должна иметь серверную и клиентскую части;

– система должна быть кроссплатформенной.

API-методы для взаимодействия с программной системой представлены в таблице 9.

Таблица 9 – Входные и выходные параметры модели

№	Параметр	Входное значение	Выходное значение
1	Auth	{ login, password }	accessToken
2	Predict	{ category, applicationName, shortDescripton, socialGroups, aims, tasks, socialDescripton, qualityResult }	result
3	Statistic	–	{ applicationName, score, label, result }

Аппаратные требования к разрабатываемой системе:

– целевая операционная система: Linux;

– процессор (количество ядер): 4 ядра;

– оперативная память: 4 Гб;

– свободное дисковое пространство: 10 Гб.

Входные и выходные параметры нейронных сетей представлены в таблице 10.

Таблица 10 – Входные и выходные параметры нейронных сетей

№	Направление	Тип задачи	Значение	Эталон / сравнение
1	Вход	Многоклассовая классификация текстов	shortDescriptionRaw, tasksRaw, aimsRaw	category
2	Вход	Семантическое сходство текстов	socialDescriptionRaw, tasksRaw, aimsRaw	similarity
3	Выход		[0, 1]	

В результате выполнения главы были подобраны наиболее оптимальные методы обработки данных, архитектуры для реализации моделей, а также спроектирована математическая модель для оценки заявок и описаны требования к разрабатываемой программной системе.

3. РЕАЛИЗАЦИЯ

3.1. Сбор датасета

Источником данных для сбора датасета, как было указано ранее, является федеральный ресурс «оценка результатов проектов» [18].

Проанализировав запросы, отправляемые клиенту от сервера, было выявлено, что WEB-ресурс имеет API, на основе которого можно осуществить сбор данных. Формат ответа API по заявке представлен в листинге 1 и листинге 2 приложения.

Для автоматизации обработки данных был написан асинхронный скрипт, позволяющий получить все заявки с федерального ресурса [18, 40]. Фрагмент извлечения атрибутов из заявки для дальнейшего анализа представлен в листинге 1.

Листинг 1 – Функция извлечения атрибутов из заявки

```
@handle_errors()
async def parse_project(client: httpx.AsyncClient, i: int, project_id: str
= None, url: str = URL_BASE_ITEM) -> dict:
    result = {}
    item_columns = ['applicationName', 'shortDescriptionRaw',
         'socialGroupsRaw', 'aimsRaw', 'tasksRaw', 'startDate',
'finalDate']
    result_columns = ['qualityResult', 'evaluation']

    async with sem:
        response = await client.get(url +
f'/item?applicationId={project_id}')
        response.raise_for_status()

    data = response.json()
    contest = data.get('contestDirectionTenant')
    result['id'] = i
    result['url'] = f'https://оценка.гранты.рф/award/project/{project_id}'
    if not contest:
        result['category'] = 'None'
    else:
        result['category'] = contest.get('name')
    for item in item_columns:
        result[item] = data.get(item)

    async with sem:
        response = await client.get(url +
f'/results?applicationId={project_id}')
        response.raise_for_status()

    data = response.json()
    for item in result_columns:
        result[item] = data.get(item)

    print(f'HTTP code 200 for project: {i}')
    return result
```

3.2. Предобработка данных

Характеристики собранного датасета:

– формат: «.json»;

– количество атрибутов (столбцов): 12;

– размерность: 25000.

Методы предобработки данных представлены в таблице 11.

Таблица 11 – Методы предобработки данных

№	Метод	Реализация
1	Исключение экземпляров, которые содержат пустые значения	Функция df.dropna
2	Удаление html атрибутов	Пакет BeautifulSoup [41]
3	Удаление классов, содержащих менее 150 экземпляров	Функция df.drop по условию
4	Удаление числовых значений	Регулярное выражение
5	Перевод слов в нижний регистр	Функция lower
6	Удаление знаков препинания	Регулярное выражение
7	Удаление цифр	Регулярное выражение
8	Удаление слов-дубликатов	Регулярное выражение
9	Удаление дублирующих пробелов	Регулярное выражение
10	Удаление стоп-слов	Пакет nltk.stopwords [25]

Функция предобработки датасета представлена в листинге 2.

Листинг 2 – Функция предобработки датасета

```
def preprocess(path: str = PATH) -> pd.DataFrame:
    """
    Preprocesses input json file according to preprocessing rules
    :param path: file path
    :return: dataframe of preprocessed json file
    """
    df = json_to_pandas(path)

    # Remove missing values
    df.dropna(inplace=True)

    # Get value_counts of each category
    counts = df['category'].value_counts()

    # Define columns needed to preprocess
    dateColumns = ['startDate', 'finalDate']
    htmlColumns = ['shortDescriptionRaw', 'socialGroupsRaw', 'aimsRaw',
'tasksRaw', 'qualityResult', 'evaluation']
    removeLabels = counts[counts < 150].index

    # Extract dates
    for column in dateColumns:
        df[column] = df[column].str[:10] # ISO 8601

    # Remove html text from
```

```
    for column in htmlColumns:
        df[column] = df[column].apply(lambda x: BeautifulSoup(x,
'html.parser').get_text())

    # Remove emissions
    for label in removeLabels:
        df.drop(df.loc[df['category'] == label].index, inplace=True)

    for column in htmlColumns:
        # Remove links
        df[column] = [re.sub(r'^https?:\/\/.*[\r\n]*', '', i,
                     flags=re.MULTILINE)
                      for i in tqdm(df[column])]

        # Reduce words to lowercase
        df[column] = [i.lower() for i in tqdm(df[column])]

        # Remove punctuation marks
        df[column] = [re.sub(r"[^\w\d\s]+", " ", i)
                      for i in tqdm(df[column])]
        # Remove numbers
        df[column] = [re.sub("\d+", "", i) for i in tqdm(df[column])]

        # Remove duplicate words
        df[column] = [re.sub(r'[x]{2,}', "", i) for i in tqdm(df[column])]

        # Remove double spaces
        df[column] = [re.sub(' +', ' ', i) for i in tqdm(df[column])]

    return df
```

Стемминг

Стемминг осуществлялся при помощи библиотеки pymorphy3 [26], время выполнения стемминга – 5543 секунды, функция стемминга представлена в листинге 3.

Листинг 3 – Функция стемминга

```
def stemming(df: pd.DataFrame, columns: list[str]) -> pd.DataFrame:
    """
    Stemming method for dataframe
    :param df: input dataframe
    :param columns: columns of dataframe need to be stemmed
    :return: dataframe with stemmed columns
    """
    stop_words = stopwords.words("russian")
    morph = pymorphy3.MorphAnalyzer()

    tqdm.pandas()

    for column in columns:
        df[f'preprocessed_{column}'] = df.progress_apply(
            lambda row: stemming_process(row[column], stop_words, morph),
axis=1)

    return df
```

Лемматизация

Лемматизация осуществлялась при помощи библиотеки pymystem3 [42], время выполнения лемматизации – 1722 секунды, функция лемматизации представлена в листинге 4.

Листинг 4 – Функция лемматизации

```
def lemmatize(df: pd.DataFrame, columns: list[str]) -> pd.DataFrame:
    """
    Lemmatizing method for dataframe
    :param df: input dataframe
    :param columns: columns of dataframe need to be lemmatized
    """
    noise = list(punctuation) + stopwords.words('russian')
    smart_vectorizer = CountVectorizer(stop_words=noise)
    tok = smart_vectorizer.build_tokenizer()
    mystem = Mystem()
    for column in columns:
        df[f'preprocessed_{column}'] = list(
            map(lambda text: " ".join((tok(str(mystem.lemmatize(text))))),
tqdm(df[column])))
    return df
```

3.3. Визуализация данных

Для исследования распределения признаков, идентификации выбросов и получения представления о зависимостях между различными атрибутами была проведена визуализация предобработанных данных.

С целью выявления проблемы дисбаланса классов в датасете на рисунке 13 представлено распределение экземпляров по классам.

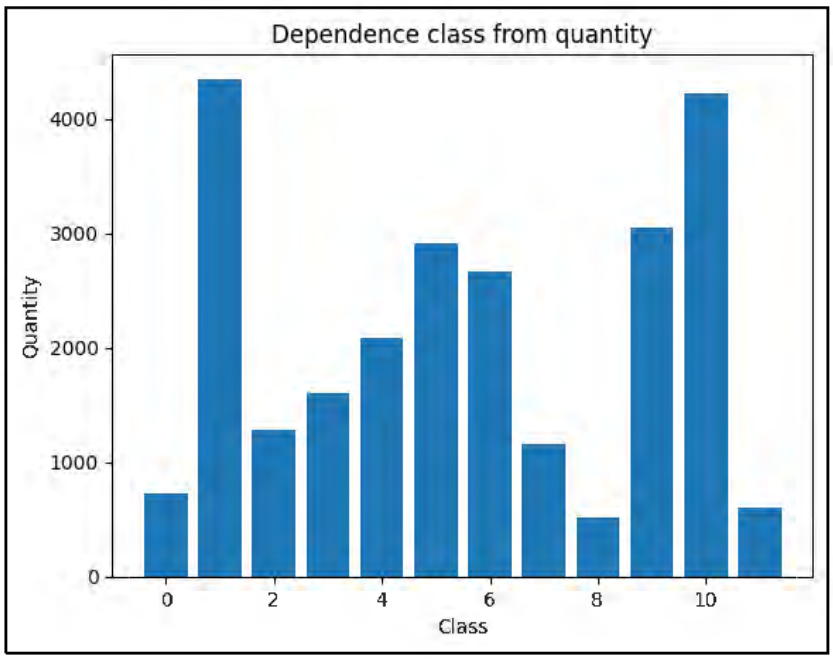

Рисунок 13 – Распределение экземпляров по классам

С целью проверки корректности подбора пар атрибутов и получения оценки для выбора модели на рисунке 14 представлено распределение среднего количества токенов по классам.

Приведем сокращения атрибутов, используемые при построении:

– aN – название;
– shDr – краткое описание;
– sGr – целевые группы;
– ar – цель;
– tr – задачи;
– socDr – социальная значимость;
– qr – качественные результаты;
– er – оценка результатов.

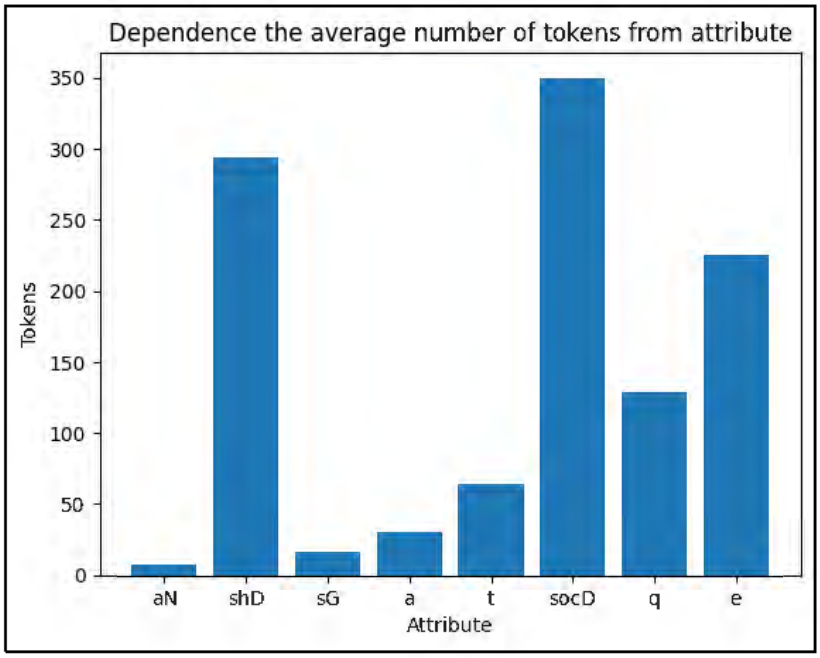

Рисунок 14 – Распределение среднего количества токенов по атрибутам

С целью изучения выбросов в наборе данных и проведения дальнейшей оптимизации модели на рисунке 15 представлено частотное распределение токенов в атрибутах.

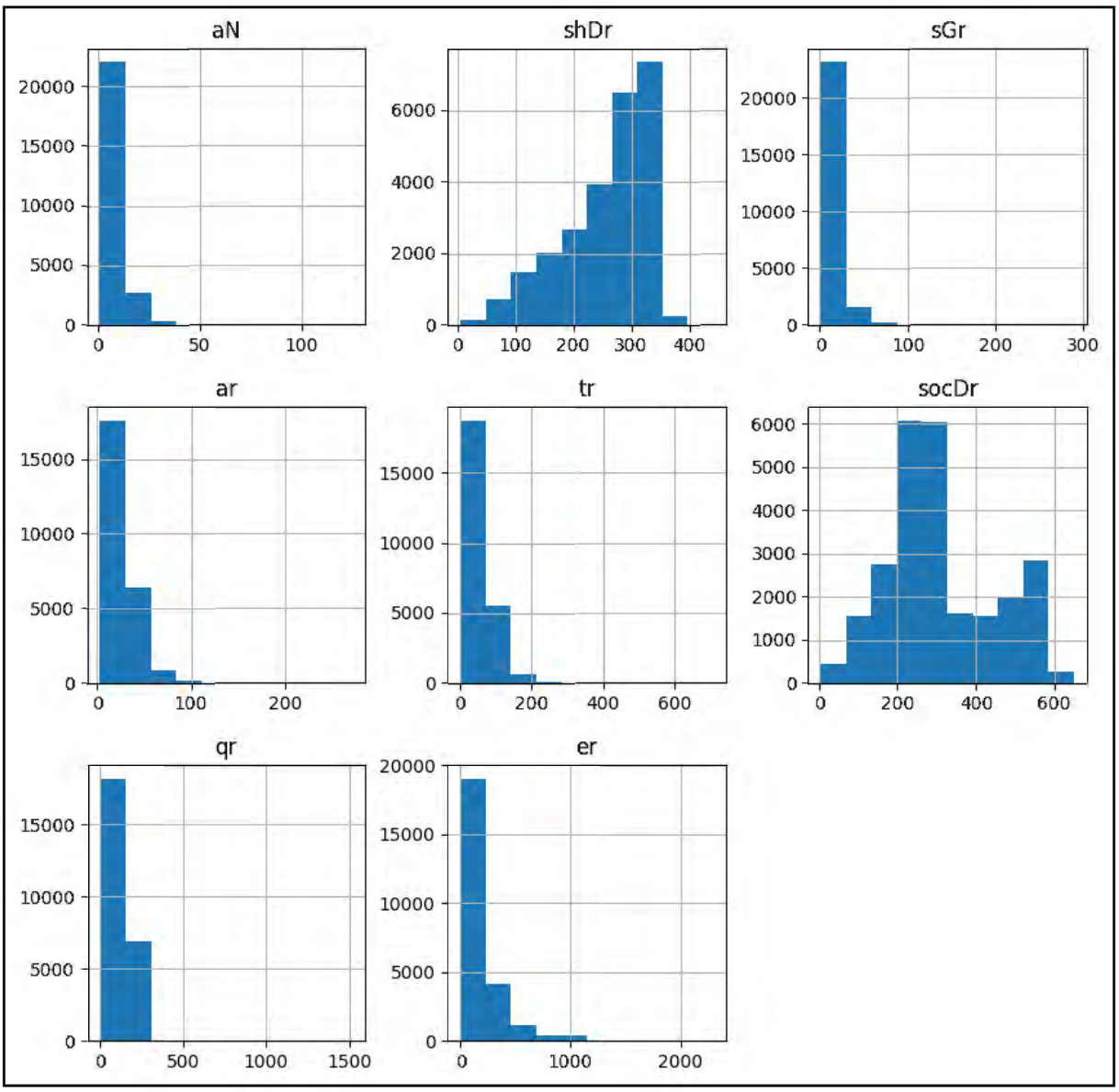

Рисунок 15 – Частотное распределение токенов в атрибутах

В результате анализа графиков было выявлены следующие риски:

– дисбаланс классов – модель отдает предпочтение классу большинства, что может быть решено при помощи нахождения весов классов, ансамблевых методов или повторной дискретизации;

– различное количество токенов атрибутов – модель склонна к переобучению (overfitting) при большом и недообучению (underfitting) при малом количестве токенов, что может быть решено при помощи тонкой настройки гиперпараметра «максимальное количество токенов», а также при помощи конкатенации нескольких атрибутов.

3.4. Реализация нейронных сетей и математической модели

Задача многоклассовой классификации текста

Для решения задачи многоклассовой классификации текста были проведены эксперименты с моделью RandomForestClassifier и архитектурами CNN, LSTM, GRU, BERT.

RandomForestClassifier

Подбор параметров для модели RandomForestClassifier был осуществлен с помощью инструмента GridSearchCV с входными параметрами:

1) n_estimators – количество деревьев в лесу: 500, 1000, 1500;
2) min_samples_split – минимальное количество выборок, необходимое для разделения внутреннего узла: 20, 30, 40;
3) max_depth – максимальная глубина оценок регрессии: 3, 5, 7.

В результате для модели получены гиперпараметры: n_estimators – 1000, min_samples_split – 30, max_depth – 3.

В ходе обучения модели RandomForestClassifier была получена accuracy 0,67 на валидационной выборке.

CNN

В ходе вычислительных экспериментов были подобраны наиболее оптимальные параметры модели, представленные в листинге 5.

Листинг 5 – Архитектура CNN

```
Model: "CNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 500, 64)           6400000

 conv1d (Conv1D)             (None, 496, 250)          80250

 global_max_pooling1d        (None, 250)               0

 dense (Dense)               (None, 128)               32128

 dense_1 (Dense)             (None, 12)                1548

=================================================================
Total params: 6513926 (24.85 MB)
Trainable params: 6513926 (24.85 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

В результате обучения модели с архитектурой CNN была получена accuracy 0,72 на валидационной выборке, график обучения модели представлен на рисунке 16.

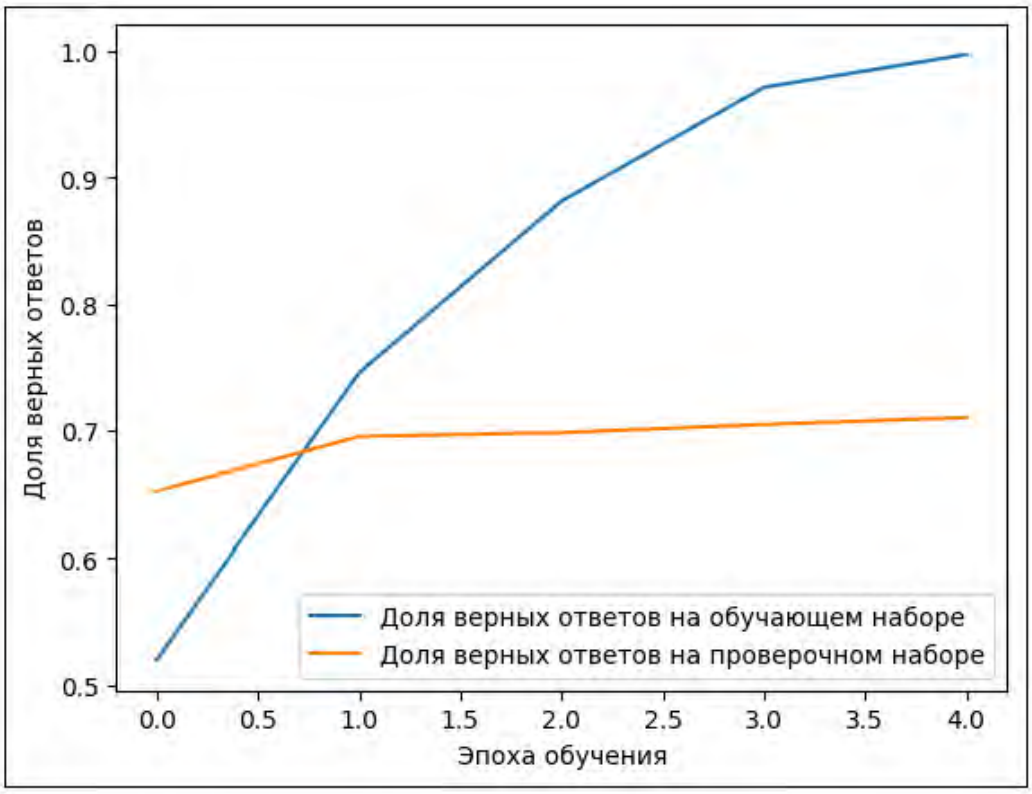

Рисунок 16 – Результаты обучения модели с архитектурой CNN

LSTM

В ходе вычислительных экспериментов были подобраны наиболее оптимальные параметры модели, представленные в листинге 6.

Листинг 6 – Архитектура LSTM

```
Model: "LSTM"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 500, 32)           3200000

 lstm (LSTM)                 (None, 16)                3136

 dense_2 (Dense)             (None, 12)                204

=================================================================
Total params: 3203340 (12.22 MB)
Trainable params: 3203340 (12.22 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

В результате обучения модели с архитектурой LSTM была получена accuracy 0,60 на валидационной выборке, график обучения представлен на рисунке 17.

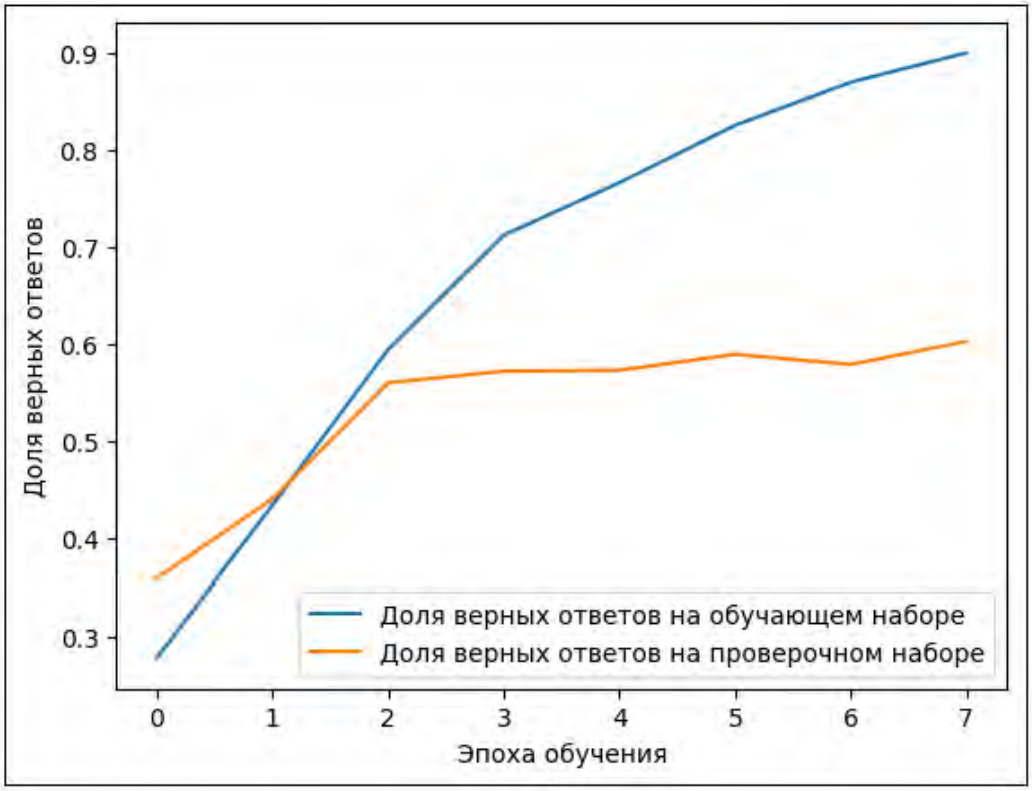

Рисунок 17 – Результаты обучения модели с архитектурой LSTM

GRU

В ходе вычислительных экспериментов были подобраны наиболее оптимальные параметры модели, представленные в листинге 7.

Листинг 7 – Архитектура GRU

```
Model: "GRU"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 500, 16)           1600000

 gru (GRU)                   (None, 16)                1632

 dense_3 (Dense)             (None, 12)                204

=================================================================
Total params: 1601836 (6.11 MB)
Trainable params: 1601836 (6.11 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

В результате обучения модели с архитектурой LSTM была получена accuracy 0,60 на валидационной выборке, график обучения представлен на рисунке 18.

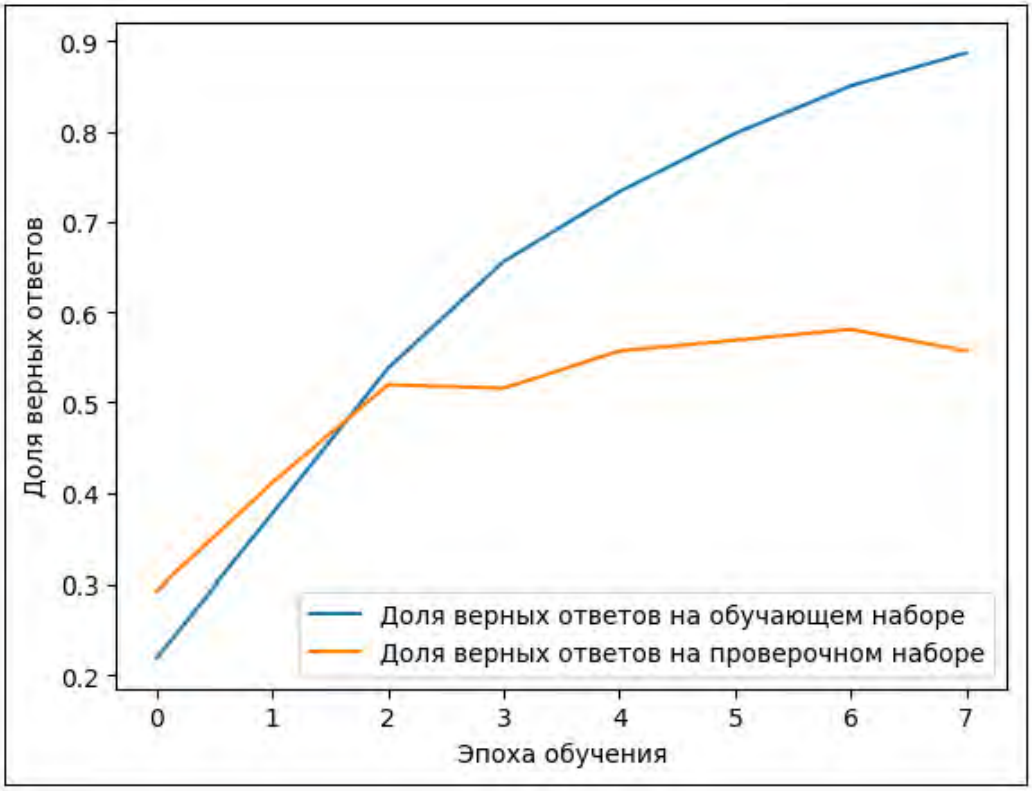

Рисунок 18 – Результаты обучения модели с архитектурой GRU

BERT

Для языковой модели BERT была выбран русскоязычный энкодер предложений rubert-tiny-2 [43] по следующим причинам:

– фокус на специфику русского языка;

– небольшой размер и высокая эффективность на устройствах с ограниченными ресурсами;

– высокая способность обобщения в различных задачах NLP.

Характеристики модели [43]:

– размер модели: 29,4M параметров;

– словарь модели: 80000 токенов;

– максимальная длина текста: 2048 токенов.

Для обучения использовался гибкий и эффективный фреймворк pytorch [39]. Архитектура модели представлена в листинге 8.

Листинг 8 – Архитектура BERT

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
BertForSequenceClassification                                      --
├─BertModel: 1-1                         (6, 312)                  --
│    └─BertEmbeddings: 2-1               (6, 768, 312)             --
│    │    └─Embedding: 3-1               (6, 768, 312)             26,154,336
│    │    └─Embedding: 3-2               (6, 768, 312)             638,976
│    │    └─Embedding: 3-3               (1, 768, 312)             624
│    │    └─LayerNorm: 3-4               (6, 768, 312]             624
│    │    └─Dropout: 3-5                 (6, 768, 312)             --
│    └─BertEncoder: 2-2                  (6, 768, 312)             --
│    │    └─ModuleList: 3-6                                        2,301,552
│    └─BertPooler: 2-3                   (6, 312)                  --
│    │    └─Linear: 3-7                  (6, 312)                  97,656
│    │    └─Tanh: 3-8                    (6, 312)                  --
├─Dropout: 1-2                           (6, 312)                  --
├─Linear: 1-3                            (6, 12)                   3,756
==========================================================================
Total params: 29,197,524
Trainable params: 29,197,524
Non-trainable params: 0
==========================================================================
```

В результате обучения модели с архитектурой BERT была получена accuracy 0,75 на валидационной выборке и 0,74 на тестовой выборке, график обучения и матрица ошибок представлены на рисунках 19 и 20.

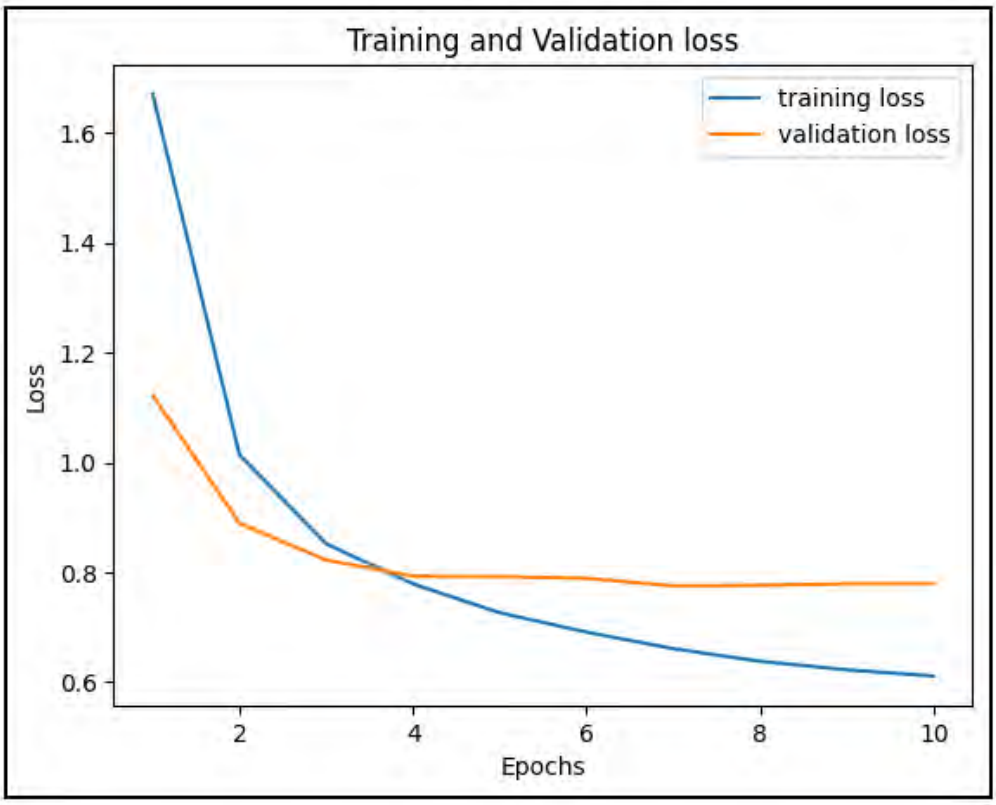

Рисунок 19 – Результаты обучения модели с архитектурой BERT

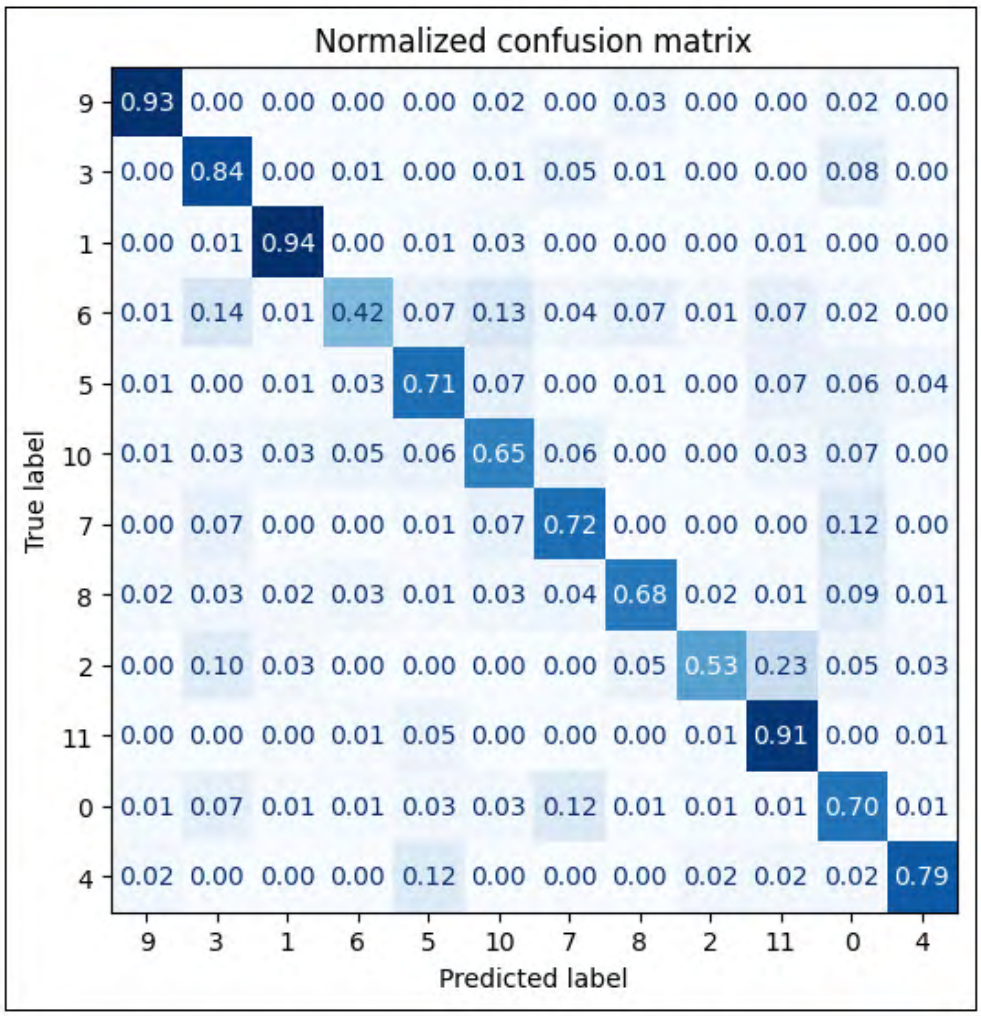

Рисунок 20 – Матрица ошибок модели с архитектурой BERT

Текстовый отчет, показывающий основные метрики модели классификации с архитектурой BERT, представлен на рисунке 21.

	precision	recall	f1-score	support
9	0.93	0.83	0.88	69
3	0.84	0.82	0.83	388
1	0.94	0.85	0.90	117
6	0.42	0.68	0.52	84
5	0.71	0.69	0.70	165
10	0.65	0.71	0.68	215
7	0.72	0.63	0.67	229
8	0.68	0.77	0.72	84
2	0.53	0.66	0.58	32
11	0.91	0.83	0.86	259
0	0.70	0.69	0.70	321
4	0.79	0.71	0.75	48
accuracy			0.75	2011
macro avg	0.73	0.74	0.73	2011
weighted avg	0.76	0.75	0.75	2011

Рисунок 21 – Отчет по метрикам модели с архитектурой BERT

Конфигурация

Поскольку наилучшие результаты при обучении были получены на архитектуре BERT, что подтвердило теоретические выводы, возьмем предложенную архитектуру в качестве базовой.

Опишем конфигурацию модели BERT для обучения:

– разделение обучающей / тестовой / валидационной выборки: 0,7/0,2/0,1;

– количество экземпляров в обучающей выборке: 17595;

– количество экземпляров в тестовой выборке: 5027;

– количество экземпляров в валидационной выборке: 2513;

– максимальная длина токенов (max_lenght): 500;

– взвешенность классов (class_weights): да;

– оптимизатор (optimizer): AdamW;

– шедулер (scheduler): get_linear_schedule_with_warmup, get_cosine_schedule_with_warmup из пакета transformers;

– скорость обучения (learning_rate): 1e-5;

– размер выборки (batch_size): 8;

– число шагов разогрева (warmup_steps): 500;

– сокращение весов (weight_decay): 1e-3.

Результаты обучения всех нейронных сетей с лучшими показателями, полученными в ходе экспериментов, представлены в таблице 12.

Таблица 12 – Результаты обучения

№	Tokenizer	Architecture	Epochs	Batch size	Loss	Accuracy
1	TfidfVectorizer/ Word2Vec/AutoTokenizer rubert-tiny2	Random forest	–	32	–	0,6716
2	Tensorflow keras.preprocessing.text	CNN	5	16	0,0166	0,7162
3	Tensorflow keras.preprocessing.text	LSTM	8	16	0,2872	0,6022
4	Tensorflow keras.preprocessing.text	GRU	8	16	0,3111	0,5808
5	**AutoTokenizer rubert-tiny2**	**BERT**	**3**	**8**	**0,7877**	**0,7458**

Метод для вызова нейронной сети для классификации текста представлен в листинге 9.

Листинг 9 – Метод вызова нейронной сети для классификации текста

```
class MultiLabelClassification:
    def __init__(self):
        self.model =
BertForSequenceClassification.from_pretrained(model_path,
num_labels=num_labels)
        self.tokenizer = BertTokenizerFast.from_pretrained(model_path)
        self.device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
        self.model.to(self.device)
        self.model.eval()

    def get_class(self, text: str) -> str:
        inputs = self.tokenizer(text, padding=True, truncation=True,
max_length=max_length, return_tensors="pt")
        outputs = self.model(**inputs)
        probs = outputs[0].softmax(1)
        prediction = inverted_dict[target_names[probs.argmax()]]
        return prediction
```

Задача семантического сходства текстов

Поскольку в рамках работы была использована готовая модель, то напишем функцию, позволяющую взаимодействовать с моделью. Метод для вызова нейронной сети представлен в листинге 10.

Листинг 10 – Метод вызова нейронной сети для решения задачи семантического сходства текстов

```
class SentenceSimilarity:
    def __init__(self):
        self.tokenizer = AutoTokenizer.from_pretrained(model_name)
        self.model = BertForSTS()
        self.device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
        self.model.to(self.device)
        self.model.load_state_dict(torch.load(PATH, self.device))
        self.model.eval()

    def get_similarity_score(self, sentence_pair: list[str]) -> float:
        test_input = self.tokenizer(sentence_pair, padding='max_length',
max_length=max_length, truncation=True,
return_tensors="pt").to(self.device)
        test_input['input_ids'] = test_input['input_ids']
        test_input['attention_mask'] = test_input['attention_mask']
        del test_input['token_type_ids']
        output = self.model(test_input)
        sim = torch.nn.functional.cosine_similarity(output[0], output[1],
dim=0).item()

        return round(sim, 2)
```

Математическая модель

Представим программную реализацию математической модели, описанной во второй главе, и код ее вызова в листинге 11.

Листинг 11 – Метод вызова математической модели

```
def get_prediction(grand_application: GrantApplication, session: Session =
Depends(get_session)) -> JSONResponse:
    similarities = [
        pipe.get_similarity_score([
            grand_application.socialDescription,
            grand_application.aims]),
        pipe.get_similarity_score([
            grand_application.socialDescription,
            grand_application.tasks]),
        pipe.get_similarity_score([
            grand_application.socialDescription,
            grand_application.socialGroups])
    ]

    classifications = [
        pipe2.get_class(grand_application.applicationDescription),
        pipe2.get_class(grand_application.aims),
        pipe2.get_class(grand_application.tasks)
    ]

    label = int(Counter(classifications).most_common(1)[0][0])
    score = sum(similarities) / len(similarities)

    encoded_label = transform_label(label)
    label_trues = 0

    for item in classifications:
        if item == grand_application.category:
            label_trues += 1

    result = (label_trues/len(classifications) + score)/2

    content = jsonable_encoder(Response(label=encoded_label, score=score,
result=result))

    return JSONResponse(content=content)
```

3.5. API и контейнеризация

Серверная часть

Для организации работы серверной части программной системы был разработан следующий функционал:

- запись результатов вычислений в базе данных;
- аутентификация администратора через JWT (JSON Web Token);
- метод получения всех предсказаний;
- реализация интерфейса для взаимодействия с клиентом.

При реализации функционала использованы инструменты:

– СУБД Postgres [44];

– графический клиент pgAdmin;

– фреймворк FastAPI [45].

Клиентская часть

Для организации работы клиентской части программной системы были использованы следующие инструменты:

– менеджер пакетов npx [46];

– JavaScript-библиотека ReactJS [47];

– библиотека «Chakra-UI» [48].

Приведем скриншоты основных страниц программной системы.

На рисунке 22 представлена форма ввода заявления с целью получения оценки по заявке на грант.

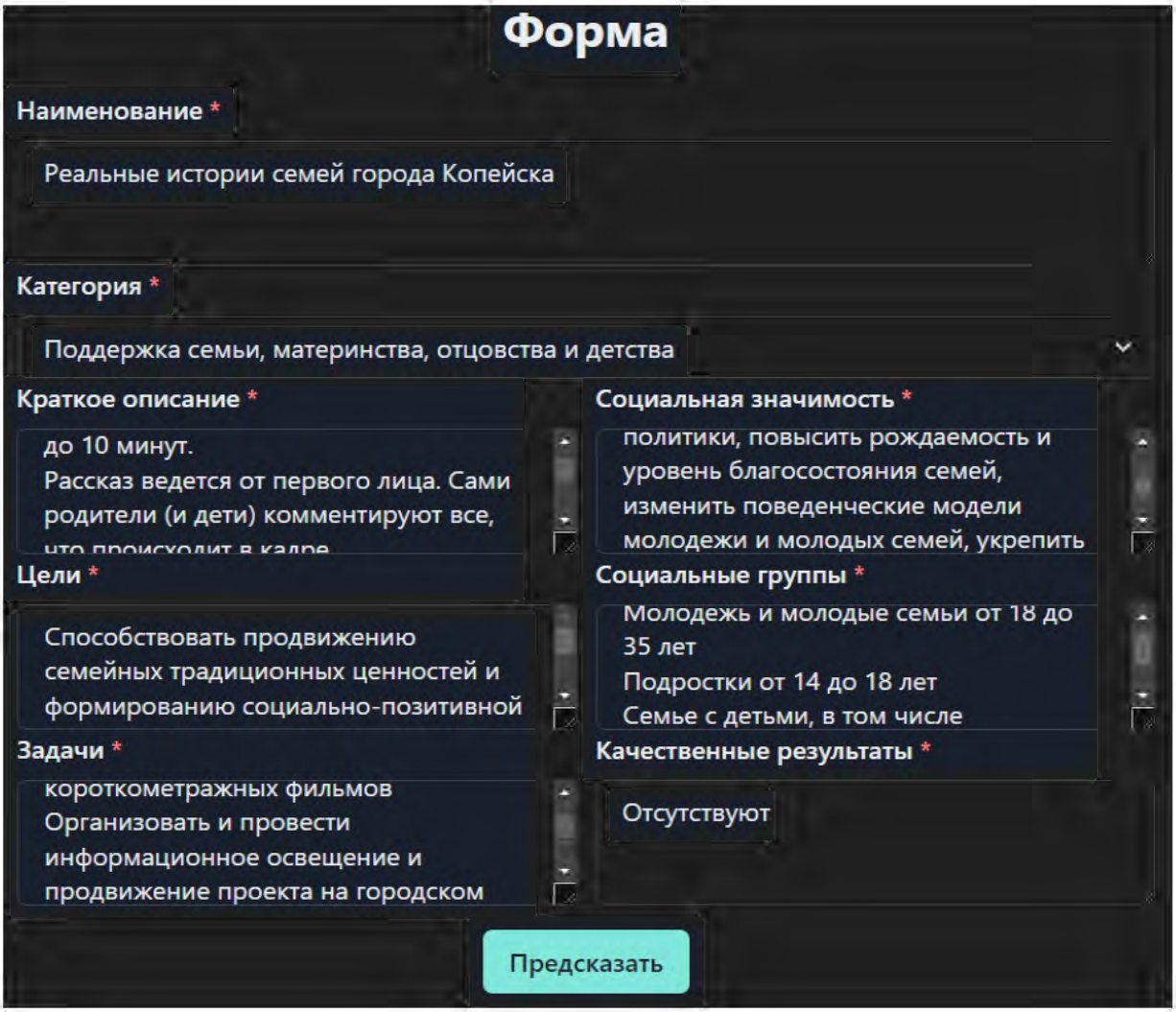

Рисунок 22 – Форма ввода заявления

На рисунке 23 представлена таблица с запросами, выполненными пользователями в программной системе.

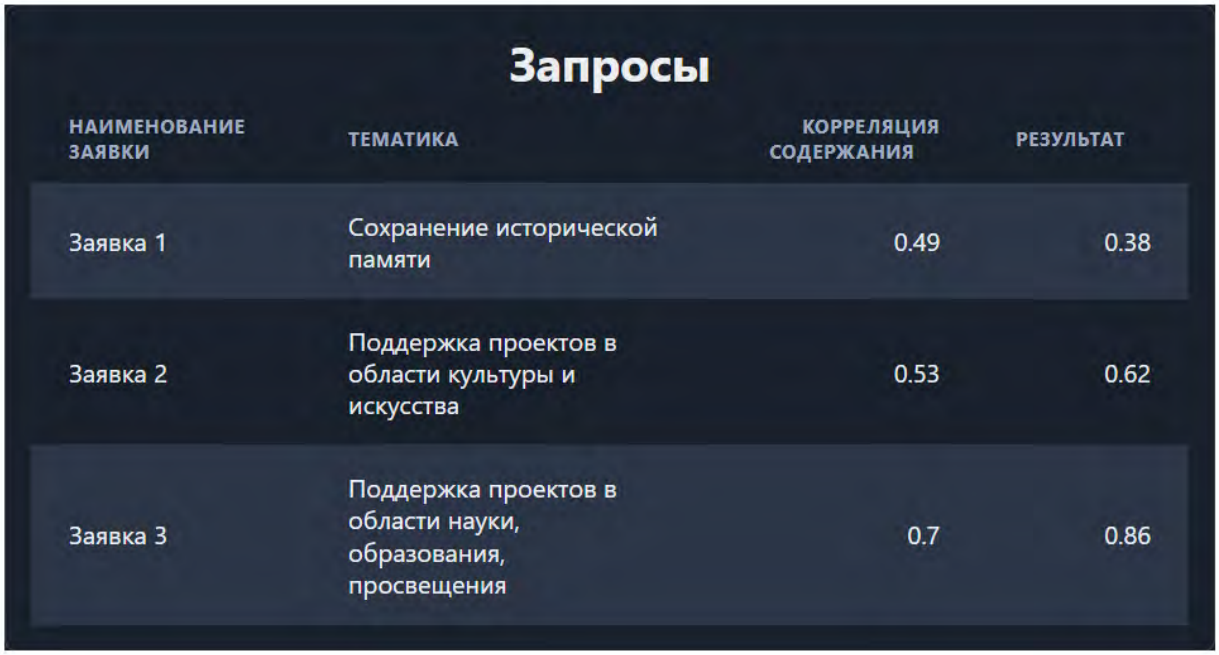

Рисунок 23 – Страница с запросами

На рисунке 24 представлена форма авторизации в системе.

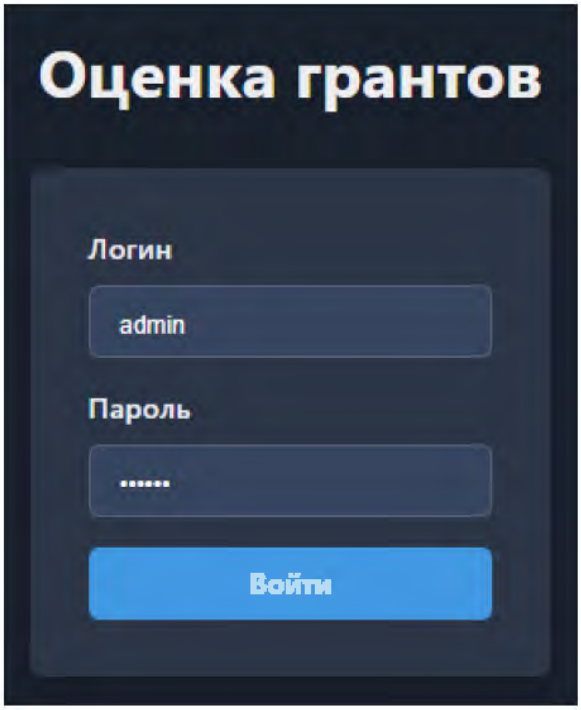

Рисунок 24 – Страница авторизации

Контейнеризация

Для воспроизводимости программной системы были реализованы dockerfile для клиентской и серверной части, а также docker-compose.yaml для автоматизации развертывания приложения (листинг 12) [49].

Листинг 12 – Docker-compose.yaml

```
version: "3.7"
services:
  postgres:
    image: postgres:14-alpine
    volumes:
      - postgres-data:/var/lib/postgresql/data
    environment:
      POSTGRES_PASSWORD: password
      POSTGRES_USERS: postgres
      POSTGRES_DB: grant
    restart: always
  pgadmin4:
    image: dpage/pgadmin4:latest
    restart: unless-stopped
    environment:
      PGADMIN_DEFAULT_EMAIL: user@example.com
      PGADMIN_DEFAULT_PASSWORD: password
    ports:
      - 49999:80
    volumes:
      - pgadmin:/var/lib/pgadmin
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    command: 'bash -c "cd /app && python -m core"'
    environment:
      db_host: postgres
      db_database: grant
      db_password: password
    ports:
      - 8000:8080
    restart: unless-stopped
    expose:
      - 8000
  frontend:
    build:
      context: ./frontend/
      dockerfile: ./Dockerfile
    restart: unless-stopped
    ports:
      - 3000:3000
    expose:
      - 3000
volumes:
  postgres-data:
  pgadmin:
```

В результате выполнения главы была разработана программная система для оценки грантов [50].

4. ТЕСТИРОВАНИЕ

Результаты тестирования реализованной программной системы приведены в таблице 13.

Таблица 13 – Результаты тестирования программной системы

№	Входные данные	Ожидаемый результат	Тест пройден
1	Аутентификация пользователя в системе	Пользователь переводится на главную страницу	Да
2	Переход пользователя в раздел «Предсказание»	Пользователь перешел в раздел «Предсказание»	Да
3	Пользователь ввел данные в форму и нажал кнопку «предсказать»	Пользователь получил предсказание	Да
4	Переход пользователя в раздел «Статистика»	Пользователь перешел в раздел «Статистика» и получил таблицу со статистикой	Да
5	Пользователь нажал кнопку «Выйти»	Пользователь вышел из системы	Да
6	Пользователь обновил страницу через 60 минут	Пользователь попал на страницу аутентификации	Да
7	Администратор разворачивает систему на машине с иной ОС с помощью docker-compose	Система разворачивается и готова к работе	Да

В результате тестирования не было обнаружено ошибок. Программная система показывает приемлемую скорость работы при запуске на машине с минимальными рекомендуемыми характеристиками.

ЗАКЛЮЧЕНИЕ

Оценка заявок на финансирование некоммерческих организаций является важным, но трудоемким и ресурсозатратными этапом проведения конкурса. Задача оценки грантов может быть решена при помощи использования технологии искусственного интеллекта, которая позволит значительно ускорить время проверки и выявить наиболее перспективные проекты.

В рамках данной работы была разработана модель автоматической оценки заявок некоммерческих организаций на финансирование от Правительства Челябинской области. При этом были решены следующие задачи.

1. Изучена предметная область, проведен обзор научной литературы.
2. Подготовлен набор данных для обучения, спроектированы архитектуры нейронных сетей и математическая модель.
3. Обучены нейронные сети, оценены результаты их работы.
4. Реализована программная система для оценки заявок на основе нейросетевых технологий.
5. Проведено тестирование программной системы.

В настоящий момент производится опытное тестирование разработанной модели. В будущем планируется продолжить разработку и улучшение программной системы, в частности обучить нейронную сеть для сравнения семантического сходства на основе размеченных данных заказчика и улучшить пользовательский интерфейс.

ЛИТЕРАТУРА

1. Никовская Л.И. Гражданские инициативы и модернизация России. / Л.И. Никовская, В.Н. Якимец, М.А. Молокова // Москва: Ключ-С, 2011. – 336 с.

2. Авдалян А.В. Гражданские инициативы в современной России. // Государственное управление. Электронный вестник, 2016. – № 57. – С. 201–218.

3. Дзусова С.С. О некоторых аспектах финансирования некоммерческих организаций: отечественный и зарубежный опыт. // Финансовая жизнь, 2018. – № 1. – С. 68–72.

4. Дан старт очередному конкурсу грантов губернатора для НКО и активных граждан. [Электронный ресурс] URL: https://minobr74.ru/press/item/11484 (дата обращения: 10.02.2024 г.).

5. Главная страница информационного ресурса «Фонд поддержки гражданских инициатив Южного Урала». [Электронный ресурс] URL: https://грантыгубернатора74.рф (дата обращения: 10.02.2024 г.).

6. Указ Президента Российской Федерации от 10.10.2019 г. № 490 «О развитии искусственного интеллекта в Российской Федерации». [Электронный ресурс] URL: http://www.consultant.ru/document/cons_doc_LAW_335184/ (дата обращения: 10.02.2024 г.).

7. Рахова М.В. Особенности управления социальными проектами: вопросы финансирования. / М.В. Рахова, О.А. Новикова // Экономика и управление: проблемы, решения, 2022. – № 5. – С. 132–138.

8. Провалинский Д.И. Некоторые проблемы реализации отечественных грантов и пути их оптимизации. // Правовая политика и правовая жизнь, 2017. – № 3. – С. 139–145.

9. Бубнова Е.Ю. Экспертная оценка грантов: проблемы и пути развития. // Вестник ВГУ. Серия: Экономика и управление, 2022. – № 2. – С. 112–122.

10. Сироткин А.В. Оценка актуальности заявки на проектное финансирование в автоматизированной системе распределения грантов. / А.В. Сироткин, В.К. Копченко // Современные наукоемкие технологии, 2021. – № 12–1. – С. 109–113.

11. Сироткин А.В. Отраслевая идентификация заявок в автоматизированной экспертной системе распределения грантов. / А.В. Сироткин, О.А. Старикова // Современные наукоемкие технологии, 2019. – № 7. – С. 99–103.

12. Рогожин Д.К. Разработка и применение модели оценки заявок на предоставление грантов федеральным вузам и федеральным государственным учреждениям из бюджета города Москвы на основе методов машинного обучения для автоматической обработки текстов. / Д.К. Рогожин, Е.В. Павлова // Инжиниринг предприятий и управление знаниями: Сборник научных трудов XXV Российской научной конференции. – Москва: РЭУ имени Г.В. Плеханова, 2022. – С. 235–241.

13. Гусев П.Ю. Разработка системы классификации текстов по научным специальностям с применением методов машинного обучения. // Вестник НГУ, 2021. – № 1. – С. 39–47.

14. Сироткин А.В. Модель определения новизны заявки на проектное финансирование методами терминологического анализа // Современные наукоемкие технологии, 2020. – № 4–2. – С. 239–244.

15. Introducing AI-Assisted Application Screening: Transform your grant review process with intelligent pre-screening. [Электронный ресурс] URL: https://www.smartsimple.com/blog/introducing-ai-assisted-application-screening (дата обращения: 10.02.2024 г.).

16. Funding agencies say no to AI peer review. [Электронный ресурс] URL: https://www.science.org/content/article/science-funding-agencies-say-no-using-ai-peer-review (дата обращения: 10.02.2024 г.).

17. Гольдберг Й. Нейросетевые методы в обработке естественного языка: руководство. // ДМК Пресс, 2019. – 282 с.

18. Оценка результатов проектов. Главная страница. [Электронный ресурс] URL: https://оценка.гранты.рф (дата обращения: 10.03.2024 г.).

19. Чирков А.Н. Сравнительное исследование библиотеки beautiful-soup4 и selenium для парсинга веб-страниц. // Научно-технические инновации и веб-технологии. – 2022. – № 1. – С. 74–78.

20. Акжолов Р.К. Предобработка текстов для решения задач NLP. // Вестник науки, 2020. – № 3. – С. 24.

21. Ганегедара Т. Обработка естественного языка с TensorFlow: руководство. // ДМК Пресс, 2020. – 382 с.

22. Milokov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space [Электронный ресурс] // arXiv.org, 2013. Дата обновления: 07.09.2013 г. URL: https://arxiv.org/abs/1301.3781 (дата обращения: 10.03.2024 г.).

23. Pennington J., Socher R., Manning C. GloVe: Global Vectors for Word Representation [Электронный ресурс] // ACL Anthology, 2014. URL: https://aclanthology.org/D14-1162/ (дата обращения: 10.03.2024).

24. Joulin A, Grave E., Bojanowski P., Mikolov T. Bag of Tricks for Efficient Text Classification [Электронный ресурс] // arXiv.org, 2016. Дата обновления: 09.08.2016 г. URL: https://arxiv.org/abs/1607.01759 (дата обращения: 10.03.2024 г.).

25. NLTK: пакет библиотек. [Электронный ресурс] URL: https://github.com/nltk/nltk (дата обращения: 10.03.2024 г.).

26. Pymorphy3: Morphological analyzer. [Электронный ресурс] URL: https://github.com/no-plagiarism/pymorphy3 (дата обращения: 10.03.2024 г.).

27. Гудфеллоу Я. Глубокое обучение. / Я. Гудфеллоу, И. Бенджио, А. Курвилль // ДМК Пресс, 2018. – 652 с.

28. Yoon K. Convolutional Neural Networks for Sentence Classification [Электронный ресурс] // arXiv.org, 2014. Дата обновления: 03.09.2014 г. URL: https://arxiv.org/abs/1408.5882 (дата обращения: 10.03.2024 г.).

29. Pengfei L., Xipeng Q., Xuanjing H. Recurrent Neural Network for Text Classification with Multi-Task Learning [Электронный ресурс] // arXiv.org, 2016. Дата обновления: 17.05.2016 г. URL: https://arxiv.org/abs/1605.05101 (дата обращения: 10.03.2024 г.).

30. Hochreiter S. Long short-term memory. / S. Hochreiter, J. Schmidhuber. // Neural computation, 1997. – Vol. 9. – P. 1735–1780.

31. Cho K. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches [Электронный ресурс] // arXiv.org, 2014. Дата обновления: 07.10.2014 г. URL: https://arxiv.org/abs/1409.1259 (дата обращения: 10.03.2024 г.).

32. Chi S., Xipeng Q., Yige X., Xuanjing H. How to Fine-Tune BERT for Text Classification? [Электронный ресурс] // arXiv.org, 2019. Дата обновления: 05.02.2020 г. URL: https://arxiv.org/abs/1905.05583 (дата обращения: 10.03.2024 г.).

33. Devlin J., Chang M., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Электронный ресурс] // arXiv.org, 2018. Дата обновления: 24.05.2019 г. URL: https://arxiv.org/abs/1810.04805 (дата обращения: 10.03.2024 г.).

34. Bruck T., Pouly M. Estimating Text Similarity based on Semantic Concept Embeddings [Электронный ресурс] // arXiv.org, 2024. Дата обновления: 09.01.2024 г. URL: https://arxiv.org/abs/2401.04422 (дата обращения: 10.03.2024 г.).

35. Цыденов, С.Б. Алгоритмы сравнения схожести текстов на основе нейросетевых алгоритмов: Магистерская диссертация. – Томск, 2023. – 94 с.

36. Кралько, Е.В. Определение схожести специализированных текстов на естественном языке: Магистерская диссертация. – Москва, 2018. – 52 с.

37. Йылдырым, С. Осваиваем архитектуру Transformer. / С. Йылдырым, М. Асгари-Ченаглу. // ДМК Пресс, 2022. – 320 с.

38. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks [Электронный ресурс] // arXiv.org, 2019. Дата обновления: 27.08.2019 г. URL: https://arxiv.org/abs/1908.10084 (дата обращения: 10.03.2024 г.).

39. PyTorch 2.3.0 documentation. [Электронный ресурс] URL: https://pytorch.org/docs/stable/index.html (дата обращения: 10.04.2024 г.).

40. Grant parser source code. [Электронный ресурс] URL: https://github.com/Zelar2/parser (дата обращения: 01.05.2024 г.).

41. Beautiful Soup 4.12.0 documentation. [Электронный ресурс] URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата обращения: 10.04.2024 г.).

42. Pymystem 3.0.1 documentation. [Электронный ресурс] URL: https://pythonhosted.org/pymystem3/ (дата обращения: 10.04.2024 г.).

43. Rubert-tiny2: a small Russian BERT-based encoder [Электронный ресурс] URL: https://huggingface.co/cointegrated/rubert-tiny2 (дата обращения: 10.04.2024 г.).

44. Postgres 16 documentation. [Электронный ресурс] URL: https://www.postgresql.org/docs/ (дата обращения: 10.04.2024 г.).

45. FastAPI 0.110.0 documentation. [Электронный ресурс] URL: https://fastapi.tiangolo.com/ (дата обращения: 10.04.2024 г.).

46. Npx CLI Commands. [Электронный ресурс] URL: https://docs.npmjs.com/cli/v8/commands/npx (дата обращения: 10.04.2024 г.).

47. React documentation. [Электронный ресурс] URL: https://legacy.reactjs.org/tutorial/tutorial.html (дата обращения: 10.04.2024 г.).

48. Chakra-ui v2.8.2 documentation. [Электронный ресурс] URL: https://v2.chakra-ui.com/docs (дата обращения: 10.04.2024 г.).

49. Docker documentation. [Электронный ресурс] URL: https://docs.docker.com (дата обращения: 10.04.2024 г.).

50. Project source code. [Электронный ресурс] URL: https://github.com/Zelar2/DiplomaProject (дата обращения: 01.05.2024 г.).

ПРИЛОЖЕНИЕ. Формат ответа API по заявке

Листинг 1 – Структура заявки в разделе «О проекте»

```
{
    "publicId": "guid",
    "oldApplicationId":"int"
    "number": "string",
    "applicantFullName": "string",
    "regionId": "guid",
    "region": {...},
    "areaCityId": "guid",
    "areaCity": {...},
    "contestSubDirectionTenantId": "guid",
    "contestSubDirectionTenant": {...},
    "contestDirectionId": "guid",
    "contestDirectionTenantId": "guid",
    "contestDirectionTenant": {...},
    "competitionId": "guid",
    "competition": {...},
    "rating": "float",
    "applicationName": "string",
    "shortDescriptionRaw": "string",
    "aimsRaw": "string",
    "tasksRaw": "string",
    "socialGroupsRaw": "string",
    "socialDescriptionRaw": "string",
    "coFinance": "float",
    "requestedSum": "float",
    "totalSum": "float",
    "totalCoFinance": "float",
    "totalPayment": "float",
    "totalSumResult": "float",
    "inn": "string",
    "ogrn": "string",
    "organizationWebSite": "string",
    "createdDate": "timestamp",
    "confirmDate": "timestamp",
    "startDate": "timestamp",
    "finalDate": "timestamp",
    "tenantId": "guid",
    "deletedDate": "timestamp",
    "isBlocked": "boolean",
    "blockType": "string",
    "isPreNominated": "boolean",
    "directionName": "string",
    "overallRate":"int"
    "votes": "string",
    "votesCount":"int"
    "isOwner": "boolean",
    "hasOwnVote": "boolean",
    "canNominate": "boolean",
    "ownRate":"int"
    "rate": "float",
    "periodInDays": "int",
    "voting": {...},
    "shareModel": {...},
    "requestedCoFinance": "float",
    "requestedToCoFinancePercent": "int",
    "totalPaymentCoFinance": "float",
    "totalPaymentToCoFinancePercent": "int",
    "id": "guid"
}
```

Листинг 2 – Структура заявки в разделе «Основные итоги»

```
{
    "application": "string",
    "applicationId": "guid",
    "qualityResult": "string",
    "evaluation": "string",
    "unplannedResult": "string",
    "disadvantages": "string",
    "conclusion": "string",
    "quantitativeIndicators": [
        {
            "reportSummary": {...},
            "reportSummaryId": "int",
            "name": "string",
            "expectedResult": "int",
            "totalResult": "int",
            "rate": "float",
            "order": "int",
            "oldUid": "guid",
            "id": "int",
            "createdDate": "timestamp",
            "modifiedDate": "timestamp",
            "deletedDate": "timestamp",
            "createdBy": "guid",
            "modifiedBy": "guid",
            "deletedBy": "guid",
            "version": "string"
        }
    ],
    "id": "int"
}
```