МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

ОПРЕДЕЛЕНИЕ КАТЕГОРИИ ТЕКСТОВОЙ ИНФОРМАЦИИ С ИСПОЛЬЗОВАНИЕМ МАШИННО - ОБУЧАЕМЫХ СИСТЕМ

Студент _____ 11.06.2026 Педай Никита Денисович
(подпись, дата) (ФИО полностью)

Руководитель _____ 11.06.2026 Дорошенко И.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ 11.06.2026 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент зав. каф. "Программирование" ПензГТУ _____ 11.06.26 Марьинских А.И.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.26 № 14)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 23.06.2026 № 25)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«20» 09 2024 г.

ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

1. Студент Педай Никита Денисович гр. 24ВВИм1 факультета ВТ

2. Тема работы Определение категории текстовой информации с использованием машинно – обучаемых систем

Тема утверждена приказом ПГУ № 920/о от "20" 09 2024 г.

3. Руководитель магистерской диссертации к.т.н., доцент кафедры ВТ ПГУ Дорошенко И.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Выполнить исследование и разработать программный модуль интеллектуальной классификации текстовых сообщений для системы обмена сообщениями. Реализовать обработку входных текстовых данных, определение категории сообщения, формирование результата классификации и применение параметров фильтрации. Предусмотреть выявление полезного контента, спама, рекламы и оскорбительных сообщений. Программный модуль должен обеспечивать взаимодействие с клиентской частью приложения и хранение результатов обработки.

5. Перечень подлежащих разработке вопросов
Требуется провести анализ методов автоматической классификации текстовой информации и существующих систем обнаружения нежелательного контента. Исследовать возможности применения нейросетевых моделей для анализа коротких сообщений. Разработать архитектуру системы, алгоритм предварительной обработки и классификации текста, механизм фильтрации и структуру взаимодействия программных компонентов. Сформировать экспериментальный набор данных, реализовать основные модули системы, провести функциональное тестирование и оценить качество классификации.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области автоматической классификации текстовых сообщений	10%	20.09.24-15.11.2024	
2.	Анализ требований к интеллектуальной системе классификации сообщений	10%	16.11.2024–20.12.2024	
3.	Изучение литературы, методов классификации и существующих программных решений	10%	21.12.2024–31.01.2025	
4.	Проектирование архитектуры системы и алгоритма обработки текстовых сообщений	15%	01.02.2025–15.03.2025	
5.	Формирование экспериментального набора данных и реализация программного модуля классификации	30%	16.03.2025–31.01.2026	
6.	Функциональное и экспериментальное тестирование разработанной системы	10%	01.02.2026–28.02.2026	
7.	Анализ результатов исследования и оценка качества классификации сообщений	10%	01.03.2026–30.04.2026	
8.	Оформление пояснительной записки	5%	01.05.2026–20.05.2026	

Дата выдачи задания "20" 09 2024 г.

Руководитель магистерской диссертации _____ Дорошенко И.Н.

(подпись, дата)

Задание к исполнению принял студент _____ Педай Н.Д.

(подпись, дата)

Работу к защите допустить
Декан факультета _____ 15.06.2026 Фионова Л.Р.

РЕФЕРАТ

Магистерская диссертация содержит 81 страницы, 14 рисунков, 11 таблиц, 20 источников.

КЛАССИФИКАЦИЯ ТЕКСТОВОЙ ИНФОРМАЦИИ, МАШИННОЕ ОБУЧЕНИЕ, ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА, НЕЙРОННЫЕ СЕТИ, CNN, МЕССЕНДЖЕР, ФИЛЬТРАЦИЯ СООБЩЕНИЙ, POSTGRESQL

Целью работы является разработка серверного модуля автоматической классификации текстовых сообщений для мессенджера с применением методов машинного обучения. Разрабатываемый модуль предназначен для предварительной обработки текста, определения категории сообщения и формирования параметров фильтрации контента.

В результате выполненной работы спроектирована архитектура интеллектуальной системы, реализованы серверный программный интерфейс, модуль предварительной обработки, свёрточная нейронная сеть и подсистема хранения результатов классификации.

Для обучения и тестирования модели сформирован набор данных объёмом 100000 сообщений, включающий категории «полезный контент», «спам», «реклама» и «оскорбления». Проведённая экспериментальная проверка подтвердила работоспособность разработанного решения и возможность применения модуля в системах обмена сообщениями.

					ПГУ 1.090401.18.001 ПЗ		
Изм.	Лист	№ докум	Подпись	Дата			
Разраб.		Педай Н.Д.		11.06.26	Определение категории текстовой информации с использованием машинно-обучаемых систем. Пояснительная записка	Лит.	Лист 2 / Листов 99
Провер.		Дорошенко И.Н.		11.06.26			
Н. Контр.		Бычков А.С.		11.06.26		ФВТ, каф. ВТ гр. 24ВВИм1	
Утв.		Митрохин М.А.		11.06.26			

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 6
ВВЕДЕНИЕ 7
1 Анализ предметной области 9
1.1 Анализ современных подходов к автоматической классификациитекстовой информации 9
1.2 Анализ современных систем анализа и модерации сообщений 11
1.2.1 Механизмы фильтрации Gmail 12
1.2.2 Механизмы противодействия спаму в Telegram 13
1.2.3 Microsoft Purview Communication Compliance 14
1.2.4 Google Perspective API 14
1.2.5 Сравнительный анализ рассмотренных систем 15
1.2.6 Вывод 16
1.3 Особенности применения методов искусственного интеллекта для анализа текстовых сообщений 16
1.4 Формирование информационной модели классифицируемых сообщений 18
1.5 Постановка задачи 21
2 Проектирование модуля классификации текстовых сообщений 24
2.2 Алгоритм классификации и фильтрации сообщений 26
2.3 Архитектура серверной части 29
2.4 Алгоритм классификации сообщений 32
2.5 Логическая структура системы 34
3.2 Формирование и подготовка экспериментального набора данных 45
3.3 Реализация модулей системы 49
3.4 Результаты тестирования программной реализации 62
3.4 Экспериментальная оценка качества классификации 72
ЗАКЛЮЧЕНИЕ 78
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 79
ПРИЛОЖЕНИЕ А 79

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей работе применяются следующие сокращения и обозначения:

БД - база данных

ИИ - искусственный интеллект

СУБД - система управления базами данных

API (Application Programming Interface) - программный интерфейс приложения

CNN (Convolutional Neural Network) - свёрточная нейронная сеть

HTTP (HyperText Transfer Protocol) - протокол передачи гипертекста

JSON (JavaScript Object Notation) - текстовый формат обмена структурированными данными

NLP (Natural Language Processing) - обработка естественного языка

REST (Representational State Transfer) - архитектурный стиль взаимодействия web-сервисов

SQL (Structured Query Language) - язык структурированных запросов

ВВЕДЕНИЕ

Двадцатые годы двадцать первого века ознаменовались революцией в области искусственного интеллекта. С каждым годом идет активное увеличение возможностей больших языковых моделей, которые позволяют делегировать повседневные задачи людей на ИИ-ассистентов, а также открывают новые варианты использования программных продуктов путем внедрения этих инструментов в разрабатываемые приложения.

Современные системы обмена сообщениями обеспечивают передачу значительных объёмов текстовой информации. Наряду с полезными сообщениями в цифровых коммуникациях распространяются рекламные материалы, спам и тексты, содержащие оскорбительные выражения. Рост интенсивности информационного обмена усложняет ручную обработку содержимого и повышает потребность в автоматизированном определении характера передаваемой информации.

Методы машинного обучения позволяют выделять признаки текстовых сообщений и относить поступающие данные к установленным категориям. В отличие от решений, основанных только на словарях и правилах, нейросетевые модели способны учитывать особенности построения фраз и устойчивые сочетания слов. Для обработки коротких сообщений оптимально применение свёрточной нейронной сети, сочетающей приемлемое качество классификации с невысокими требованиями к вычислительным ресурсам.

Разрабатываемая система предназначена для определения категории текстовой информации в мессенджере. Результаты классификации могут использоваться для маркировки сообщений, формирования предупреждений и применения настроек фильтрации. Автоматическая оценка содержимого не ограничивает передачу информации, а предоставляет дополнительные сведения о характере сообщения. Поэтому основными задачами магистерской диссертации являются:

– анализ существующих методов классификации текстовой информации,

– исследование систем обнаружения нежелательного содержимого,

– проектирование архитектуры интеллектуальной системы анализ результатов,

– формирование и подготовка экспериментального набора данных,

– реализация модулей предобработки, классификации и фильтрации сообщений,

– проведение тестирования и оценка качества полученных результатов.

Объектом исследования является процесс автоматической обработки и классификации текстовой информации в системах обмена сообщениями.

Предметом исследования являются методы машинного обучения, применяемые для определения категории текстовых сообщений.

Целью работы является проектирование, разработка и исследование интеллектуальной системы определения категории текстовой информации в мессенджере.

Практическая значимость работы заключается в возможности применения разработанной системы определения категории информации в мессенджерах, корпоративных коммуникационных платформах и программных комплексах, предусматривающих автоматизированный анализ и фильтрацию текстовых сообщений.

1 Анализ предметной области

1.1 Анализ современных подходов к автоматической классификации текстовой информации

В современных информационных системах значительная часть передаваемых данных представлена в текстовом виде. Особенно высокие объёмы текстовой информации обрабатываются средствами электронных коммуникаций, включая социальные сети, корпоративные платформы и мессенджеры. Рост количества сообщений приводит к необходимости применения автоматизированных методов анализа текстового контента.

Автоматическая классификация текстовой информации представляет собой процесс отнесения текстового сообщения к одной из заранее определённых категорий на основании его содержимого. Подобные методы применяются при решении задач фильтрации нежелательного контента, тематической классификации документов, анализа обращений пользователей и интеллектуальной обработки сообщений [1, 2].

На протяжении развития технологий обработки естественного языка было предложено несколько основных подходов к решению задачи классификации текстов. К ним относятся методы на основе правил, статистические методы анализа текста, алгоритмы машинного обучения и современные нейросетевые модели.

Методы, основанные на правилах, используют словари ключевых слов, регулярные выражения и заранее определённые шаблоны сообщений. Основным преимуществом данного подхода является простота реализации и высокая скорость работы. Вместе с тем эффективность подобных решений существенно снижается при изменении структуры текста и появлении новых способов представления информации.

Статистические методы основаны на анализе частоты встречаемости слов и выражений в тексте. Наиболее известными являются модели Bag-of-Words и TF-IDF. Такие подходы позволяют формировать числовое представление текстовой информации и использовать его для дальнейшей

обработки. Однако данные методы ограниченно учитывают смысловые связи между словами [4].

Следующим этапом развития стали методы машинного обучения, использующие размеченные наборы данных для обучения модели классификации. Наиболее распространёнными алгоритмами являются байесовский классификатор, логистическая регрессия и метод опорных векторов. Подобные модели позволяют повысить качество классификации по сравнению с правилами и статистическими подходами.

Наиболее активно в настоящее время развиваются методы глубокого обучения. Использование нейронных сетей позволяет автоматически выделять признаки текстовой информации и адаптироваться к особенностям предметной области. Для обработки коротких сообщений широкое распространение получили сверточные нейронные сети, обеспечивающие высокую скорость работы и достаточно высокую точность классификации.

Сравнение основных подходов к автоматической классификации текстовой информации представлено в таблице 1.

Таблица 1 - Сравнение подходов к классификации текстовой информации

Подход	Основные преимущества	Основные недостатки
Правила и словари	Простота реализации, высокая скорость обработки	Низкая гибкость, необходимость постоянного обновления правил
Статистические методы	Невысокие вычислительные затраты, простота применения	Ограниченный учёт контекста текста
Методы машинного обучения	Более высокое качество классификации	Требуется формирование обучающей выборки

Продолжение таблицы 1

Подход	Основные преимущества	Основные недостатки
Сверточные нейронные сети (CNN)	Высокая точность, эффективная работа с короткими сообщениями	Необходимость обучения модели
Трансформерные модели	Высокое качество анализа контекста	Значительные вычислительные затраты

Как следует из таблицы 1, различные подходы обладают собственными преимуществами и ограничениями. Для задач обработки сообщений мессенджеров важными требованиями являются высокая скорость работы, возможность обработки большого количества сообщений и достаточная точность классификации.

Дополнительной особенностью рассматриваемой задачи является необходимость анализа сообщений, которые могут содержать признаки спама, рекламного контента, оскорбительной лексики либо иной информации, требующей дополнительного внимания пользователя. В данном случае интеллектуальная система выступает в роли инструмента анализа текста и формирования уведомлений, тогда как окончательное решение о дальнейших действиях остаётся за пользователем.

С учётом требований к производительности и качеству классификации в качестве основы разрабатываемого прототипа целесообразно рассматривать использование сверточной нейронной сети, обеспечивающей эффективную обработку коротких текстовых сообщений и возможность дальнейшего развития системы с применением методов искусственного интеллекта.

1.2 Анализ современных систем анализа и модерации сообщений

Рост объёма цифровых коммуникаций сопровождается увеличением количества нежелательных сообщений. В электронной почте, мессенджерах, социальных сетях и корпоративных платформах распространяются массовые

рассылки, рекламные материалы, мошеннические обращения и тексты, содержащие оскорбительные выражения. Ручная проверка такого потока информации требует значительных временных затрат, поэтому современные коммуникационные сервисы применяют автоматизированные средства анализа содержимого.

Назначение существующих средств обработки текста зависит от области применения. Почтовые сервисы преимущественно решают задачу обнаружения спама и фишинговых писем. Мессенджеры ограничивают массовую рассылку и нежелательную активность учётных записей. Корпоративные решения контролируют нарушения правил деловой коммуникации и передачу конфиденциальных сведений. Специализированные программные интерфейсы формируют вероятностные оценки токсичности текстов.

Для сопоставления с разрабатываемой системой рассмотрены механизмы Gmail, Telegram, Microsoft Purview Communication Compliance и Google Perspective API. Перечисленные решения охватывают разные направления анализа: фильтрацию спама, контроль массовых рассылок, распознавание оскорбительных выражений и выявление чувствительной информации.

1.2.1 Механизмы фильтрации Gmail

Gmail содержит встроенные средства обнаружения нежелательной и потенциально опасной корреспонденции. Проверка входящих писем выполняется с учётом текста, характеристик отправителя, структуры письма, присутствия ссылок и признаков массовой рассылки. Сообщения, распознанные как нежелательные, помещаются в отдельный раздел почтового ящика.

Дополнительная защита связана с выявлением фишинговых обращений и писем, отправленных со скомпрометированных учётных записей. При обнаружении подозрительных признаков интерфейс отображает предупреждение о возможной угрозе. Получателю предоставляется

возможность сообщить о фишинге либо подтвердить ошибочность срабатывания фильтра.

Функции Gmail ориентированы на защиту электронной почты от спама, мошенничества и вредоносных действий. Отдельное распознавание рекламного содержимого не выполняется. Коммерческое предложение может быть отнесено к спаму при наличии признаков нежелательной массовой рассылки, однако категория «реклама» в результатах проверки не формируется. Автоматическая оценка оскорбительных выражений также не относится к основному назначению почтового сервиса.

1.2.2 Механизмы противодействия спаму в Telegram

Telegram применяет несколько механизмов ограничения нежелательной активности. Жалобы на сообщения и учётные записи используются при выявлении массовых рассылок, мошеннических обращений и навязчивой рекламы. После подтверждения нарушения для учётной записи может вводиться ограничение на отправку сообщений адресатам, не добавившим отправителя в список контактов.

Для крупных супергрупп предусмотрен автоматический антиспам-режим. После включения режима сообщения проверяются встроенными алгоритмами Telegram. Записи, распознанные как спам, удаляются автоматически. Администратор супергруппы может сообщить об ошибочном удалении, что позволяет учитывать ложные срабатывания при последующей работе фильтра.

Встроенные средства Telegram направлены на предотвращение массовых рассылок и коммерческого преследования. Разделение сообщений на самостоятельные категории «реклама», «оскорбления» и «полезный контент» в открытой документации не предусмотрено. Контроль агрессивных высказываний осуществляется правилами платформы, жалобами участников и действиями администраторов сообществ.

1.2.3 Microsoft Purview Communication Compliance

Microsoft Purview Communication Compliance относится к средствам контроля корпоративных коммуникаций. Программное решение используется для анализа сообщений Microsoft Teams, электронной почты и других каналов Microsoft 365. Проверка выполняется на основании заранее настроенных политик организации.

Политики позволяют обнаруживать оскорбительные и угрожающие высказывания, признаки домогательства, материалы недопустимого характера и сведения конфиденциального содержания. Сообщение, соответствующее условиям политики, регистрируется в системе и передаётся уполномоченному сотруднику для проверки.

Назначение Microsoft Purview отличается от назначения обычного спам-фильтра. Основное внимание уделяется соблюдению требований организации, снижению внутренних рисков и предотвращению передачи защищаемой информации. Анализ рекламных материалов и массовых рассылок не относится к основным функциям Communication Compliance.

1.2.4 Google Perspective API

Google Perspective API представляет собой программный интерфейс для автоматизированного анализа комментариев и коротких текстовых сообщений. В основе сервиса находятся модели машинного обучения, формирующие числовую оценку предполагаемого влияния комментария на ход обсуждения.

Основным анализируемым показателем является токсичность текста. Результат обработки представляется числом в диапазоне от 0 до 1. Увеличение значения соответствует повышению вероятности того, что сообщение будет воспринято как грубое, неуважительное или способное нарушить конструктивное общение.

Вероятностная оценка может применяться для предварительной проверки комментариев, формирования предупреждений перед публикацией и определения приоритета ручной модерации. Окончательное решение о

скрытии либо удалении текста принимается программной системой, использующей Perspective API.

Perspective API не выполняет полноценную многоклассовую категоризацию. Сервис не разделяет сообщения на полезный контент, рекламу и спам. Основная область применения связана с оценкой токсичности и близких характеристик текстовой коммуникации.

1.2.5 Сравнительный анализ рассмотренных систем

Сравнение функциональных возможностей представлено в таблице 2.

Таблица 2 - Сравнение современных систем анализа сообщений

Система	Выявление спама	Выявление рекламы	Выявление оскорбительного содержимого	Анализ чувствительной информации
Gmail	Да	Частично	Нет	Не заявлено
Telegram	Да	Частично	Не заявлено	Нет
Microsoft Purview	Нет	Нет	Да	Да
Google Perspective API	Нет	Нет	Да	Нет
Разрабатываемая система	Да	Да	Да	Нет

Gmail и Telegram не выделяют рекламные сообщения в самостоятельную категорию. Коммерческие рассылки могут быть отнесены к спаму при наличии признаков массового или нежелательного распространения. Microsoft Purview ориентирован на контроль корпоративной переписки и передачу защищаемой информации, тогда как Google Perspective API предназначен преимущественно для оценки токсичности текста.

В разрабатываемой системе используются категории «полезный контент», «спам», «реклама» и «оскорбления». Выделение рекламы в

отдельный класс позволяет отличать коммерческое предложение от массовой нежелательной рассылки. Результаты классификации применяются для формирования меток, предупреждений и параметров отображения сообщений.

Состав категорий определялся с учётом доступности открытых размеченных данных. Персональные данные и сведения ограниченного распространения в экспериментальную выборку не включались, поскольку их использование требует отдельного порядка хранения и обработки.

1.2.6 Вывод

Проведённый анализ показал, что рассмотренные программные решения ориентированы на отдельные направления обработки сообщений и не обеспечивают одновременного определения всех категорий, принятых в настоящей работе. Это обосновывает разработку собственной системы классификации текстовой информации, позволяющей различать полезный контент, спам, рекламу и оскорбления.

Выбранная структура классификатора допускает последующее расширение. При наличии соответствующей размеченной выборки в состав распознаваемых классов могут быть включены персональные данные, сведения ограниченного распространения и другие виды текстовой информации.

1.3 Особенности применения методов искусственного интеллекта для анализа текстовых сообщений

Развитие методов машинного обучения изменило подходы к автоматической обработке текстовой информации. Системы, построенные на словарях и заранее определённых правилах, способны обнаруживать только известные конструкции и требуют постоянного обновления используемых шаблонов. Модели, обучаемые на размеченных данных, формируют признаки непосредственно в процессе обучения и лучше учитывают разнообразие способов выражения одной и той же мысли.

Обработка сообщений мессенджеров имеет ряд особенностей. Короткий объём текста ограничивает количество доступных контекстных признаков, а

использование сокращений, разговорной лексики, ошибок, эмодзи и нестандартной пунктуации усложняет применение формальных правил. Дополнительная трудность связана с тем, что близкие по структуре сообщения могут относиться к различным категориям. Например, рекламное предложение и спам-рассылка нередко содержат ссылки, призывы к действию и упоминания скидок, но различаются характером распространения и назначением текста.

Для классификации текстовой информации применяются рекуррентные, свёрточные и трансформерные нейронные сети [4, 5, 9]. Рекуррентные модели обрабатывают последовательность слов с учётом ранее полученного контекста, однако последовательный характер вычислений ограничивает возможность их распараллеливания. Трансформерные архитектуры позволяют учитывать удалённые смысловые связи и обеспечивают высокое качество при решении многих задач обработки естественного языка, но требуют большего объёма вычислительных ресурсов и памяти.

В настоящей работе используется свёрточная нейронная сеть. Выбор данной архитектуры связан с особенностями коротких сообщений и необходимостью обнаружения локальных сочетаний слов, характерных для отдельных категорий. Свёрточные фильтры позволяют выделять устойчивые фрагменты текста, после чего сформированные признаки используются при определении итогового класса. Архитектура также допускает обработку сообщений с небольшой задержкой и не требует применения крупной языковой модели.

Классификация выполняется на основании содержимого сообщения без набора вручную заданных правил для каждого класса. На выходе модели формируются вероятностные оценки, соответствующие предусмотренным категориям. Итоговый класс определяется по наибольшему значению, а уровень уверенности может использоваться при формировании предупреждения или информационной метки.

Результат анализа не служит основанием для безусловной блокировки

текста. Категория сообщения передаётся клиентской части и учитывается при применении настроек отображения. Сообщение может быть показано без изменений, отмечено соответствующей меткой либо скрыто при наличии установленного правила фильтрации.

Применение обучаемой модели позволяет отказаться от жёсткой зависимости от словарей и текстовых шаблонов. Качество классификации при этом определяется составом обучающей выборки, корректностью разметки, параметрами предобработки и архитектурой нейронной сети. Указанные факторы учитываются при формировании набора данных и экспериментальной проверке разработанной системы.

1.4 Формирование информационной модели классифицируемых сообщений

Разработка системы определения категории текстовой информации требует установления состава входных данных, промежуточного представления и результата обработки. Информационная модель определяет сведения, передаваемые между компонентами системы, и устанавливает связь исходного сообщения с результатом классификации.

Основным информационным объектом является текстовое сообщение, поступающее из клиентского приложения. Для выполнения классификации используется содержимое сообщения, а служебные атрибуты обеспечивают идентификацию записи и её связь с системой обмена сообщениями.

Структура входных данных представлена в таблице 3.

Таблица 3 – Структура входных данных

Параметр	Описание	Тип данных
message_id	Уникальный идентификатор сообщения	Целое число
sender_id	Идентификатор отправителя	Целое число
message_text	Исходный текст сообщения	Строка

Продолжение таблицы 3

Параметр	Описание	Тип данных
send_time	Дата и время отправки	Дата и время

Основным анализируемым полем является message_text. Идентификатор сообщения используется для сопоставления исходной записи с результатом классификации. Сведения об отправителе и времени передачи относятся к служебным атрибутам и не передаются нейросетевой модели в качестве входных признаков.

После получения сообщения формируется подготовленное представление текста. Предварительная обработка включает нормализацию регистра, удаление технических символов, упорядочивание пробельных последовательностей и токенизацию. Токены преобразуются в числовые идентификаторы в соответствии со словарём, сформированным по обучающей выборке.

Максимальная длина входной последовательности составляет 200 токенов. Сообщения большей длины сокращаются, а последовательности меньшего размера дополняются служебными значениями. Подготовленное представление используется только при выполнении нейросетевой модели и не заменяет исходный текст сообщения.

Дополнительные характеристики, включая длину текста и наличие ссылки, могут рассчитываться при подготовке экспериментального набора данных. Указанные значения применяются при анализе состава корпуса и не относятся к обязательным полям входного запроса.

Результатом работы модели является набор вероятностных оценок для предусмотренных категорий. Категория с наибольшим значением принимается в качестве итогового результата, а соответствующая вероятность используется как уровень уверенности классификатора.

Выходной информационный объект содержит:

– идентификатор сообщения message_id,

– определённую категорию predicted_category,

– уровень уверенности confidence_score,

– признак предупреждения warning_flag,

– текстовую метку category_tag,

– рекомендуемое действие display_action.

Поля predicted_category и confidence_score формируются нейросетевой моделью. Параметры warning_flag, category_tag и display_action определяются после сопоставления полученной категории с настройками отображения.

Состав категорий и соответствующие параметры отображения представлены в таблице 4.

Таблица 4 - Структура результата классификации

Категория	Характеристика сообщения	Текстовая метка	Предупреждение
Полезный контент	Нейтральное или информационное сообщение	Целое число	Отображение без дополнительной маркировки
Спам	Массовая или нежелательная рассылка	Нежелательный контент	Отображение или скрытие по настройкам
Реклама	Коммерческое предложение или продвижение услуг	Нежелательный контент	Отображение или скрытие по настройкам
Оскорбления	Сообщение с агрессивными или уничижительными выражениями	Нежелательный контент	Отображение или скрытие по настройкам

Результат классификации не приводит к автоматическому удалению или блокированию сообщения. Параметр отображения определяет способ представления текста в клиентском приложении и учитывает установленные правила фильтрации.

Связь между информационными объектами имеет последовательный характер. Исходное сообщение преобразуется в числовую

последовательность, после чего обученная модель формирует вероятностные оценки категорий. Определённая категория сопоставляется со справочником и параметрами фильтрации. На заключительном этапе формируется структурированный результат, передаваемый клиентскому приложению.

Информационная модель процесса классификации представлена на рисунке 1.

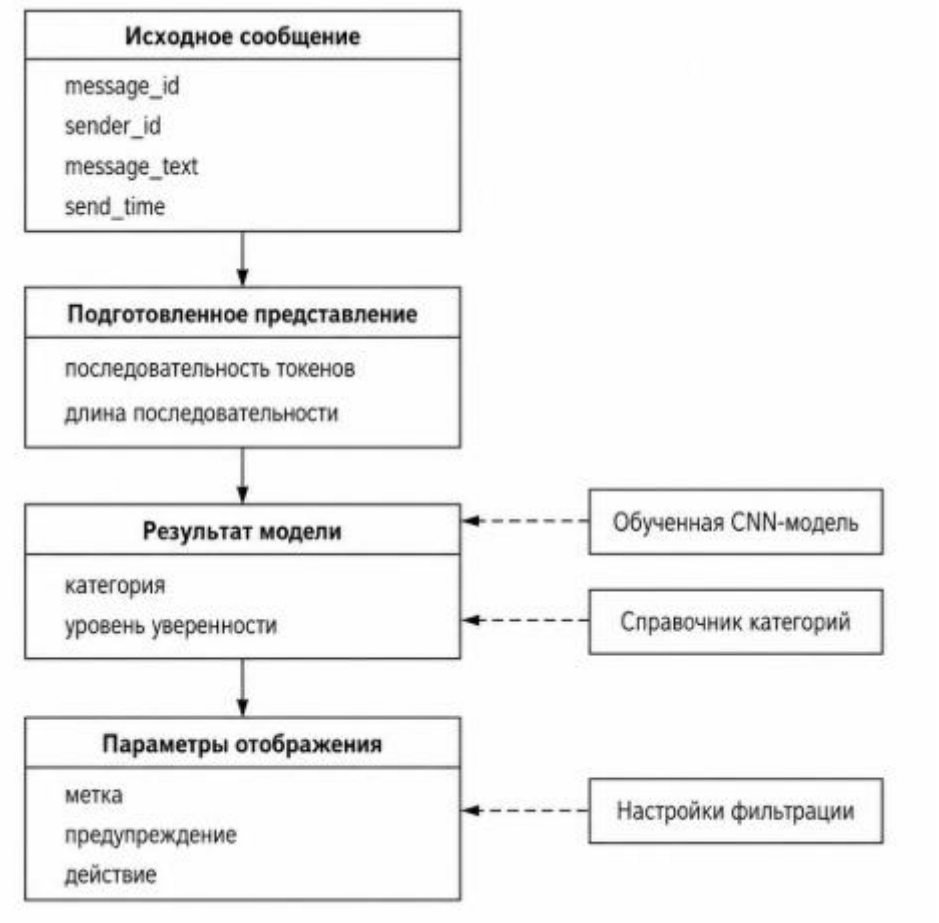

Рисунок 1 - Информационная модель процесса классификации сообщений

Предложенная структура разделяет исходные данные, подготовленное представление, результат нейросетевой модели и параметры отображения. Расширение перечня категорий может выполняться без изменения состава входного сообщения и общей последовательности обработки.

1.5 Постановка задачи

Анализ методов классификации текстовой информации и существующих программных решений показал, что системы, основанные на

словарях и фиксированных правилах, имеют ограниченную применимость при обработке сообщений мессенджеров. Изменение лексики, использование сокращений и разнообразие формулировок требуют постоянной корректировки шаблонов. Применение обучаемой модели позволяет формировать признаки непосредственно на основе размеченных текстовых данных.

Задача исследования заключается в разработке интеллектуальной системы определения категории текстовой информации. Система должна выполнять предварительную обработку сообщения, преобразование текста в числовое представление, классификацию и формирование результата, пригодного для использования в приложении обмена сообщениями.

В состав распознаваемых классов входят полезный контент, спам, реклама и оскорбления. Выбранные категории позволяют проверить возможность разграничения нейтральных сообщений, массовых нежелательных рассылок, коммерческих предложений и текстов с признаками агрессивного речевого поведения.

Для решения поставленной задачи необходимо:

– определить структуру входных данных и результата классификации,

– разработать архитектуру системы определения категории информации,

– подготовить алгоритм предварительной обработки текстовых сообщений,

– сформировать экспериментальный набор данных,

– реализовать нейросетевой классификатор,

– разработать механизм формирования параметров отображения сообщения,

– организовать взаимодействие программных компонентов,

– провести функциональное тестирование и оценить качество классификации.

Результатом исследования должна стать программная система,

обеспечивающая автоматическое определение категории текстового сообщения и передачу сведений о результатах анализа клиентскому приложению. Полученная категория может использоваться для маркировки, формирования предупреждения или применения настроек фильтрации.

2 Проектирование модуля классификации текстовых сообщений

2.1 Общая архитектура системы

Проектируемая система предназначена для определения категории текстового сообщения и передачи результата анализа клиентскому приложению. Классификация выполняется средствами нейросетевой модели, размещённой в серверной части. Клиентское приложение используется для отправки текста, получения результата и применения параметров отображения.

При проектировании выбрана клиент-серверная архитектура. Такое построение отделяет интерфейс обмена сообщениями от вычислительных операций, связанных с подготовкой текста и запуском классификатора. Обновление модели, изменение правил формирования результата и расширение перечня категорий выполняются на серверной стороне и не затрагивают основной механизм работы клиентского приложения.

Архитектура системы включает три уровня:

– клиентский уровень,

– серверный уровень,

– уровень хранения данных.

Клиентский уровень представлен приложением мессенджера. На этом уровне формируется текст сообщения, выполняется передача запроса и отображается полученный результат. Клиентская часть не производит вычисления, связанные с нейросетевой классификацией, и использует серверный сервис как внешний компонент анализа текста.

Серверный уровень содержит программный интерфейс, средства проверки запроса, модуль предварительной обработки, обученную CNN-модель, механизм фильтрации и компонент формирования ответа. REST API принимает входные данные и передаёт их в последовательность обработки. Валидация исключает запросы с отсутствующим текстом, неверной структурой или недопустимым объёмом сообщения.

Обучение свёрточной нейронной сети выполняется отдельно от серверной части программного комплекса. Для подготовки модели используется язык Python и средства TensorFlow/Keras. После завершения обучения модель экспортируется в формат ONNX и размещается на сервере вместе со словарём токенов и параметрами предварительной обработки. Классификация входящих сообщений выполняется средствами ONNX Runtime в среде Node.js.

После проверки текст поступает в модуль предварительной обработки. Выполняются нормализация, токенизация и преобразование сообщения в числовую последовательность фиксированной длины. Подготовленные данные передаются CNN-классификатору, выполняемому средствами ONNX Runtime. Загруженная модель формирует вероятностные оценки для предусмотренных категорий, после чего итоговый класс определяется по наибольшему значению выходного вектора.

Результат классификации сопоставляется с настройками отображения. На этом этапе формируются текстовая метка, признак предупреждения и рекомендуемое действие. Подготовленные сведения включаются в JSON-ответ и возвращаются клиентскому приложению.

Уровень хранения данных включает базу PostgreSQL и хранилище артефактов обученной модели. В базе данных размещаются сообщения, результаты анализа и параметры фильтрации. В отдельном хранилище располагаются CNN-модель в формате ONNX, словарь токенов и параметры предварительной обработки, необходимые для формирования входной последовательности.

Обмен между клиентской и серверной частями выполняется посредством программного интерфейса. Запрос содержит исходный текст и служебные атрибуты сообщения. Ответ включает определённую категорию, уровень уверенности классификатора и параметры отображения. Единый формат обмена упрощает подключение системы к приложению мессенджера

и допускает дальнейшее использование классификатора в других программных комплексах.

Общая архитектура системы представлена на рисунке 2.

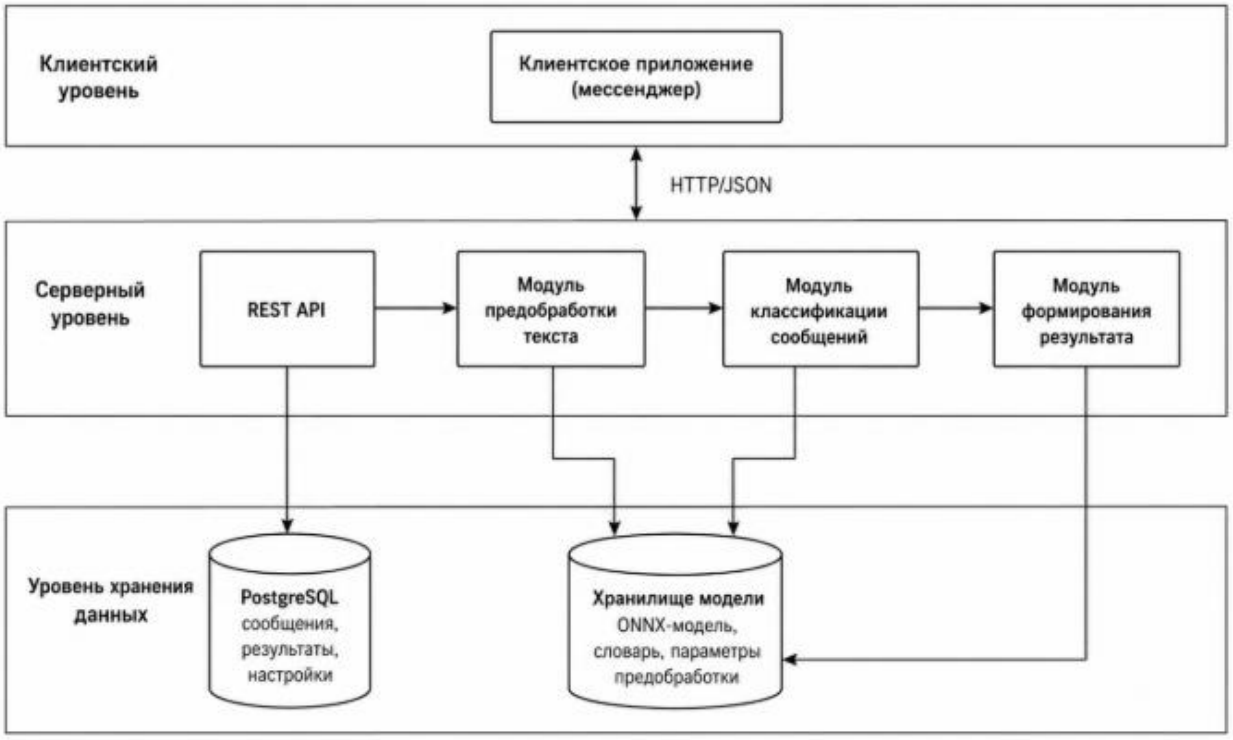

Рисунок 2 – Общая архитектура системы классификации текстовых сообщений

Разделение компонентов по уровням снижает связанность программной реализации. Нейросетевая модель, средства хранения и правила обработки результата могут изменяться независимо при сохранении структуры входного запроса и ответа. Принятая архитектура служит основой для последующего проектирования серверных компонентов и алгоритма классификации текстовых сообщений.

2.2 Алгоритм классификации и фильтрации сообщений

Алгоритм классификации и фильтрации определяет общий порядок обработки текстовой информации от момента поступления сообщения до передачи результата клиентскому приложению. Выполнение вычислительных операций осуществляется на серверной стороне, что обеспечивает единый способ подготовки данных и исключает зависимость результатов анализа от характеристик устройства, на котором запущен мессенджер.

Обработка начинается после передачи текстового сообщения через программный интерфейс. Сервер принимает запрос, проверяет наличие обязательных полей и допустимую длину текста. Сообщения, не соответствующие установленной структуре, не передаются на последующие этапы. После успешной проверки текст направляется в модуль предварительной обработки.

На этапе подготовки исходное сообщение приводится к форме, используемой классификатором. Выполняются нормализация текста, токенизация и преобразование токенов в числовую последовательность. Подробный состав операций предобработки и внутренний порядок работы нейросетевой модели приведены в разделе 2.4.

Подготовленная последовательность передаётся CNN-модели, которая формирует оценки принадлежности сообщения к предусмотренным классам. Категория определяется по наибольшему значению выходного вектора. В состав классов входят полезный контент, спам, реклама и оскорбления.

После определения категории выполняется сопоставление результата с параметрами отображения. Данный этап не изменяет решение классификатора, а определяет способ использования полученной информации. Для сообщения может быть сформирована текстовая метка, предупреждение или указание на скрытие в соответствии с установленными настройками.

Завершающей операцией является подготовка структурированного ответа. В ответ включаются идентификатор сообщения, определённая категория, уровень уверенности классификатора и параметры отображения. Сформированные данные возвращаются клиентскому приложению и используются при выводе сообщения в интерфейсе.

Сведения о выполненной классификации могут сохраняться в базе данных. Накопленные результаты используются для проверки работы программных компонентов, анализа распределения категорий и последующей оценки качества модели.

Укрупнённая схема выполнения алгоритма представлена на рисунке 3.

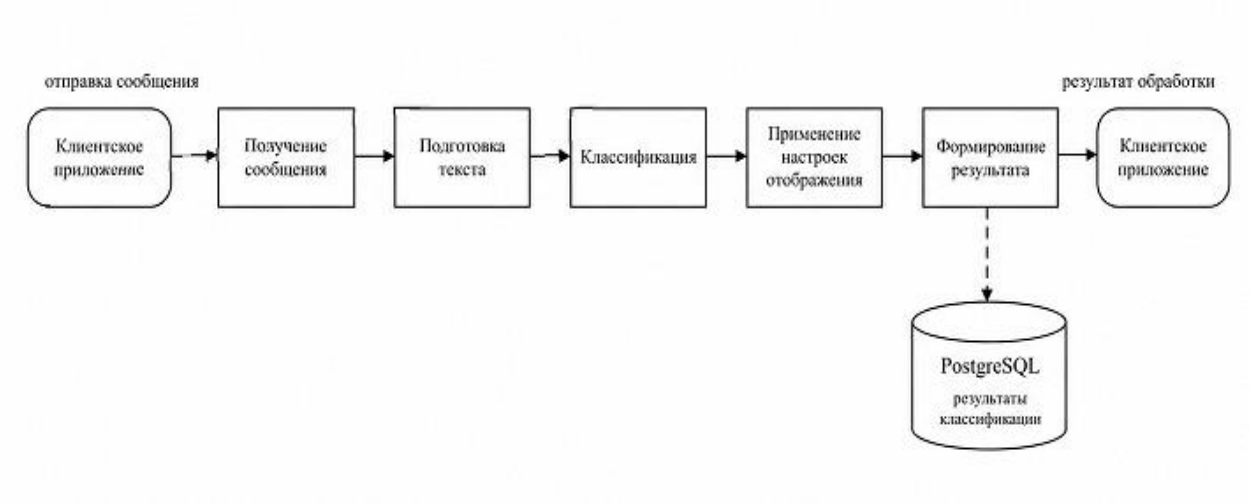

Рисунок 3 – Укрупнённая схема выполнения алгоритма классификации и фильтрации сообщений

Основные этапы алгоритма приведены в таблице 5.

Таблица 5 – Основные этапы алгоритма классификации и фильтрации

Этап	Назначение
Получение сообщения	Прием текстовых данных
Предобработка	Подготовка текста к анализу
Классификация	Определение категории сообщения
Применение настроек	Формирование параметров отображения
Формирование результата	Подготовка ответа клиентскому приложению
Сохранение данных	Регистрация результата классификации

Представленный порядок операций разделяет получение данных, работу нейросетевой модели и интерпретацию результата. Такое построение позволяет изменять параметры классификатора и правила отображения независимо друг от друга.

2.3 Архитектура серверной части

Центральным компонентом серверной части является модуль классификации сообщений. Обучение свёрточной нейронной сети выполняется на языке Python с использованием TensorFlow и Keras. После завершения обучения модель экспортируется в формат ONNX. На сервере классификатор загружается средствами ONNX Runtime и используется для определения вероятностей принадлежности сообщения к предусмотренным категориям. Такое разделение позволяет выполнять обучение в специализированной среде и использовать готовую модель непосредственно в серверном приложении. При проектировании серверной части использован модульный подход. Его применение позволяет разделить основные функции между самостоятельными компонентами, каждый из которых отвечает за отдельный этап обработки сообщений. Такая организация упрощает сопровождение программного комплекса, создаёт условия для поэтапной модернизации системы и позволяет изменять внутреннюю реализацию отдельных модулей без нарушения общей схемы взаимодействия.

Структура серверной части представлена на рисунке 4.

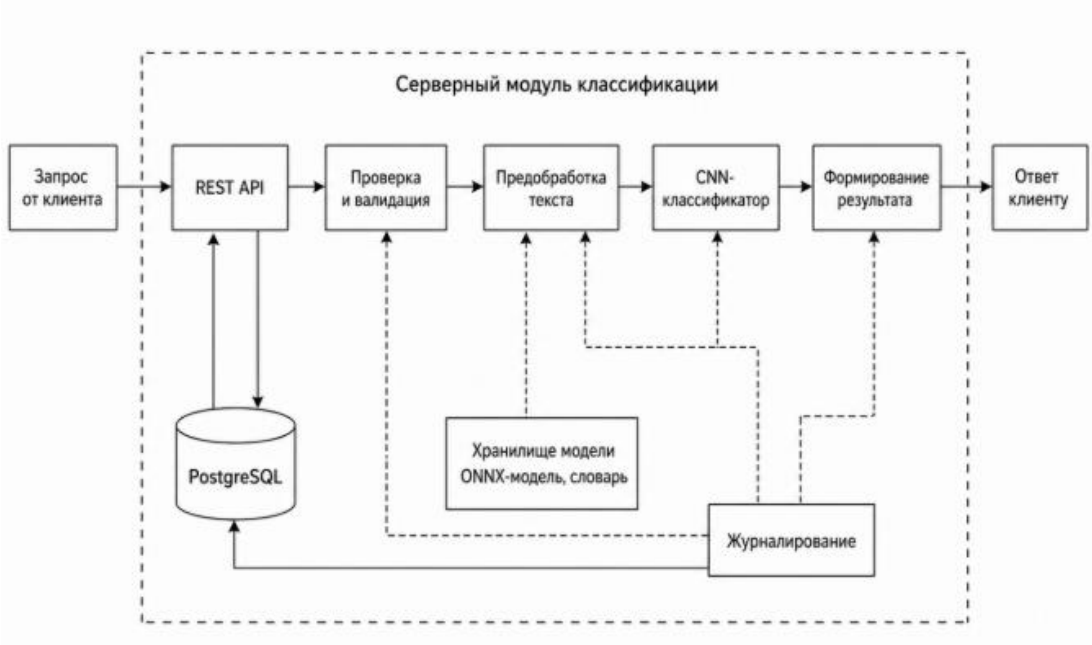

Рисунок 4 - Архитектура серверного модуля классификации

Входной точкой серверного уровня является REST API. Указанный компонент обеспечивает приём запросов от клиентского приложения,

первичную обработку передаваемых данных и маршрутизацию сообщений во внутренний контур анализа. Использование программного интерфейса прикладного уровня позволяет организовать стандартизированный обмен данными между клиентской и серверной частями системы, а также обеспечить возможность дальнейшего расширения функциональности без изменения базовых принципов взаимодействия.

После поступления сообщения управление передаётся модулю предобработки текста. Назначение данного компонента состоит в приведении текстовой информации к форме, пригодной для дальнейшего анализа средствами машинного обучения. На этом этапе выполняются нормализация текста, очистка от служебных символов, устранение избыточных пробельных конструкций, токенизация и преобразование сообщения в числовое представление. Подготовленные данные выступают входной информацией для классификатора.

Ключевым компонентом серверной части является модуль классификации сообщений. В рамках разработанной системы данный модуль реализуется на основе свёрточной нейронной сети. Выбор такой архитектуры связан с её пригодностью для анализа коротких текстовых сообщений и способностью выделять устойчивые локальные признаки, характерные для отдельных категорий содержимого. Результатом работы классификатора является определение наиболее вероятной категории сообщения и вычисление уровня уверенности модели.

Полученные результаты передаются в модуль формирования результата. На данном этапе выполняется интерпретация итогов классификации с учётом правил отображения, действующих в клиентской части системы. Здесь формируются сведения, необходимые для маркировки сообщения, вывода предупреждения или применения параметров фильтрации. Таким образом, серверная часть не только определяет категорию текста, но и подготавливает данные к практическому использованию в пользовательском интерфейсе.

Отдельное место в архитектуре занимает подсистема хранения данных. Её назначение связано с сохранением результатов классификации, служебных параметров обработки и иной информации, необходимой для последующего анализа работы системы. Централизованное хранение данных создаёт основу для накопления статистики, проверки качества классификации и дальнейшего совершенствования модели.

Для контроля функционирования программного комплекса используется подсистема журналирования. В журнале фиксируются сведения о поступлении запросов, прохождении отдельных этапов обработки и возможных ошибках. Наличие такого механизма позволяет отслеживать состояние серверной части, упрощает поиск причин сбоев и способствует повышению надёжности программного решения.

Основные компоненты серверного уровня и их назначение приведены в таблице 6.

Таблица 6 - Основные компоненты серверной части

Компонент	Назначение
REST API	Приём запросов и передача результатов обработки
Модуль предобработки	Подготовка текстовых данных к классификации
CNN-модель	Определение категории сообщения
Модуль формирования результата	Подготовка параметров отображения сообщения
Хранилище данных	Сохранение результатов обработки
Журналирование	Регистрация событий и контроль работы системы

Предложенная архитектура серверной части обеспечивает последовательное выполнение всех операций, необходимых для автоматического определения категории текстовой информации. Модульное построение создаёт возможность независимого развития отдельных

компонентов и допускает замену используемой модели машинного обучения без переработки клиентской части и основных механизмов обмена данными. Это позволяет рассматривать разработанную серверную часть как устойчивую основу для дальнейшего развития интеллектуальной системы классификации сообщений.

2.4 Алгоритм классификации сообщений

Алгоритм классификации сообщений является ключевым элементом разработанной интеллектуальной системы, поскольку именно на данном этапе выполняется определение категории текстовой информации на основе содержимого сообщения. Назначение алгоритма состоит в последовательной обработке входных данных, их преобразовании в форму, пригодную для машинного анализа, и получении итогового результата классификации.

Работа алгоритма начинается после поступления текстового сообщения в серверный модуль. На начальном этапе выполняется проверка корректности входных данных. Контролируется наличие обязательных параметров, соответствие допустимому формату и соблюдение ограничений по длине сообщения. Выполнение данной процедуры позволяет исключить обработку некорректных записей и обеспечить устойчивость функционирования программного комплекса.

После первичной проверки сообщение передаётся в модуль предварительной обработки текста. На этом этапе текст приводится к унифицированному виду, необходимому для последующего анализа. Предварительная обработка включает перевод символов в нижний регистр, удаление служебных и специальных символов, нормализацию пробельных последовательностей и токенизацию. Далее выполняется ограничение длины текстовой последовательности и преобразование сообщения в числовое представление, используемое в качестве входа нейросетевой модели.

Подготовленные данные передаются CNN-классификатору, выполняемому средствами ONNX Runtime. Классификатор использует обученную модель в формате ONNX и формирует вероятностные оценки для

предусмотренных категорий сообщений. Использование данной архитектуры обусловлено возможностью выявления локальных текстовых признаков, характерных для различных типов сообщений. В процессе обработки входной последовательности модель выделяет информативные сочетания слов и устойчивые языковые конструкции, после чего формирует внутреннее представление сообщения, необходимое для его отнесения к одному из заданных классов.

В рамках исследования рассматриваются четыре категории сообщений: полезный контент, спам, реклама и оскорбления. Для спама характерны повторяющиеся шаблоны, типовые призывы к действию и признаки массовой рассылки. Рекламные сообщения отличаются наличием коммерческих предложений, упоминаний товаров и услуг, сведений об акциях и скидках. Сообщения категории оскорблений содержат агрессивные или уничижительные высказывания. К полезному контенту относятся сообщения, не имеющие признаков перечисленных категорий и соответствующие обычным сценариям текстового взаимодействия.

Результатом работы модели является набор вероятностных оценок, отражающих степень принадлежности сообщения к каждому из рассматриваемых классов. На основании наибольшего значения вероятности определяется итоговая категория. Дополнительно фиксируется уровень уверенности модели, который используется при формировании итогового результата и позволяет оценивать надёжность принятого решения.

Полученный результат передаётся в модуль формирования ответа, где подготавливаются данные для последующей передачи клиентскому приложению. В состав выходной структуры включаются определённая категория сообщения, уровень уверенности классификации и служебные параметры, необходимые для дальнейшего отображения результата. Прикладное использование полученной категории, включая маркировку сообщения или применение параметров фильтрации, осуществляется на последующих этапах функционирования системы.

Следует отметить, что разработанный алгоритм не предназначен для самостоятельного ограничения передачи текстовой информации. Его назначение заключается в автоматическом определении категории сообщения и формировании результата анализа, который может быть использован в составе интеллектуальной системы обработки текстовых данных.

Общая последовательность выполнения операций алгоритма представлена на рисунке 5.

Рисунок 5 – Алгоритм классификации текстового сообщения

2.5 Логическая структура системы

Логическая структура определяет состав функциональных подсистем и связи, возникающие при обработке текстового сообщения. В отличие от общей

архитектуры, отражающей уровни программного комплекса, логическое представление показывает распределение операций между компонентами без привязки к конкретному способу их программной реализации.

В составе системы выделены пять подсистем:

– получения данных,

– предобработки текста,

– классификации,

– хранения информации,

– формирования результата.

Подсистема получения данных обеспечивает приём запросов от клиентского приложения. В её состав входят программный интерфейс и средства проверки входных параметров. Перед началом анализа контролируются структура запроса, наличие текста и соблюдение установленного ограничения длины сообщения. После успешной проверки данные передаются подсистеме предобработки.

Назначение подсистемы предобработки состоит в подготовке текста к использованию нейросетевой моделью. Исходное сообщение нормализуется, разбивается на токены и преобразуется в числовую последовательность. Результат обработки имеет формат и размерность, соответствующие входу обученного классификатора.

Подсистема классификации формирует вероятностные оценки принадлежности сообщения к предусмотренным категориям. В её составе используется предварительно обученная свёрточная нейронная сеть, экспортированная в формат ONNX. Выполнение модели на серверной стороне осуществляется средствами ONNX Runtime. Итоговая категория определяется по максимальному значению выходного вектора, после чего фиксируется соответствующий уровень уверенности.

Подсистема формирования результата сопоставляет определённую категорию с параметрами отображения. На основании настроек фильтрации формируются текстовая метка, признак предупреждения и рекомендуемое

действие. Подготовленные сведения включаются в структурированный ответ и передаются клиентскому приложению.

Подсистема хранения информации обеспечивает сохранение сообщений, результатов классификации, параметров фильтрации и служебных событий. Размещение данных в PostgreSQL позволяет связывать исходное сообщение с полученной категорией и использовать накопленные записи при проверке работы программных компонентов. CNN-модель в формате ONNX, словарь токенов и параметры предварительной обработки размещаются в отдельном хранилище артефактов.

Основной поток данных проходит последовательно через подсистемы получения, предобработки, классификации и формирования результата. Обращения к настройкам и хранилищам имеют вспомогательный характер и не изменяют порядок выполнения классификации.

Логическая структура интеллектуальной системы представлена на рисунке 6.

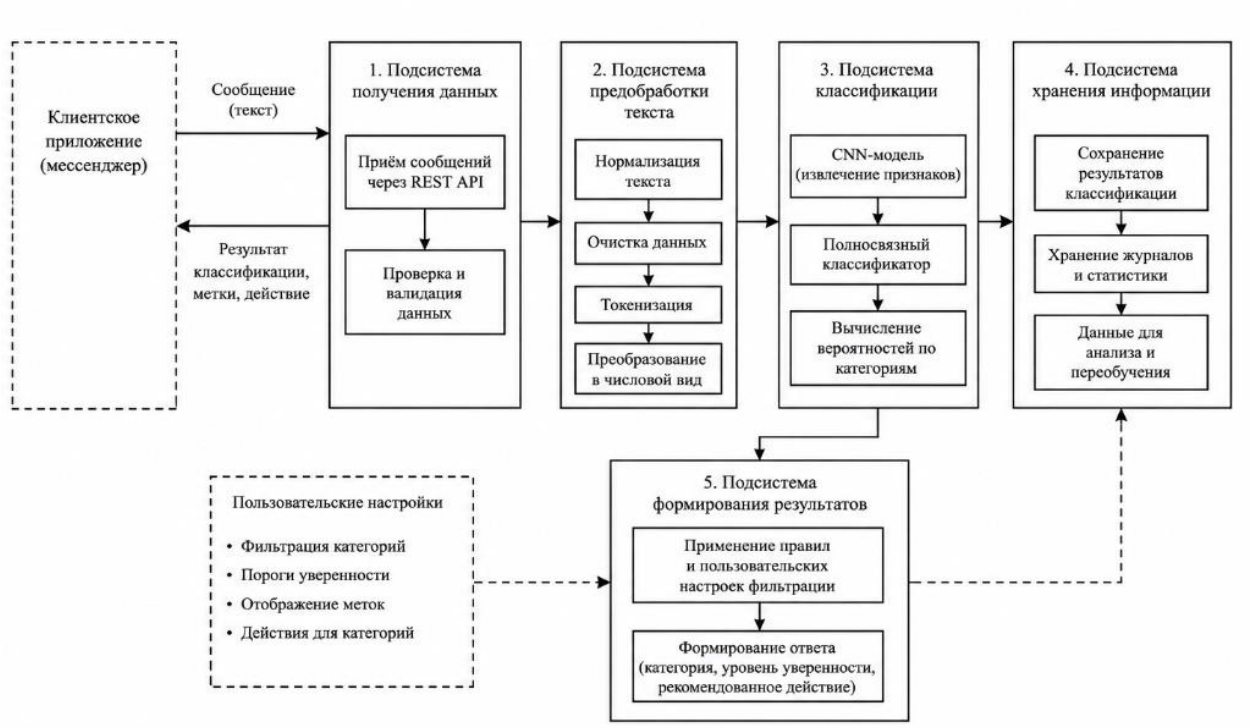

Рисунок 6 – Логическая структура интеллектуальной системы классификации сообщений

Предложенное разделение закрепляет назначение каждого функционального компонента и исключает смешение операций подготовки текста, классификации и интерпретации результата. Изменение одной подсистемы может выполняться без пересмотра общей логики обработки при сохранении форматов передаваемых данных.

2.6 Организация взаимодействия компонентов системы

Организация взаимодействия компонентов определяет порядок передачи данных между клиентским приложением, серверными модулями и подсистемой хранения. В отличие от логической структуры, отражающей состав функциональных подсистем, данный раздел рассматривает последовательность вызовов и состав информации, передаваемой при обработке одного текстового сообщения.

Обмен данными построен по клиент-серверной схеме. Клиентское приложение формирует сообщение, отправляет запрос и отображает полученный результат. Операции предварительной обработки и классификации выполняются на серверной стороне, что обеспечивает единый порядок анализа независимо от характеристик пользовательского устройства.

После отправки сообщения клиентское приложение формирует HTTP-запрос в формате JSON. В состав запроса входят идентификатор сообщения, исходный текст, служебные атрибуты и параметры отображения. Запрос передаётся REST API, являющемуся единой точкой взаимодействия с серверной частью.

REST API передаёт полученные сведения модулю валидации. На данном этапе проверяются структура запроса, наличие обязательных полей, типы значений и допустимая длина текста. При обнаружении ошибки обработка прекращается, а клиентскому приложению возвращается сообщение о причине отклонения запроса. Корректный текст направляется в модуль предварительной обработки.

Модуль предварительной обработки выполняет нормализацию регистра, очистку текста, токенизацию и ограничение длины последовательности.

Полученные токены преобразуются в числовые идентификаторы. Результатом работы является числовая последовательность длиной не более 200 токенов, передаваемая CNN-классификатору.

При запуске серверной части из хранилища загружаются CNN-модель в формате ONNX, словарь токенов и параметры предварительной обработки. Средствами ONNX Runtime создаётся рабочая сессия, повторно используемая при классификации последующих сообщений. Повторное чтение файла модели при обработке каждого запроса не выполняется.

CNN-классификатор получает числовую последовательность и формирует вероятностные оценки для категорий «полезный контент», «спам», «реклама» и «оскорбления». Категория с наибольшей оценкой принимается в качестве результата классификации. Вместе с категорией в модуль формирования результата передаётся уровень уверенности.

Модуль формирования результата обращается к параметрам фильтрации, сохранённым в PostgreSQL. На основании определённой категории и установленных настроек формируются текстовая метка, признак предупреждения и рекомендуемый способ отображения сообщения. Параметры фильтрации не изменяют результат классификации и используются только при подготовке ответа клиентскому приложению.

Сформированные сведения передаются REST API в виде внутреннего объекта ответа. После преобразования в формат JSON клиентскому приложению возвращаются идентификатор сообщения, определённая категория, уровень уверенности и параметры отображения.

Результаты обработки сохраняются в PostgreSQL через отдельный модуль доступа к данным. В базе регистрируются идентификатор сообщения, исходный текст, определённая категория, вероятностные оценки и время классификации. Служебные события и ошибки направляются в подсистему журналирования.

Состав данных, передаваемых между основными компонентами, приведён в таблице 7.

Таблица 7 - Взаимодействие компонентов системы

Компонент	Назначение	Преподаваемые данные
Клиентское приложение	REST API	Текст сообщения и служебные атрибуты
REST API	Модуль валидации	Объект входного запроса
Модуль валидации	Модуль предобработки	Проверенный текст сообщения
Модуль предобработки	CNN- классификатор	Числовая последовательность длиной до 200 токенов
Хранилище модели	CNN- классификатор	ONNX-модель, словарь токенов и параметры предобработки
CNN-модель	Модуль формирования результата	Категория и вероятностные оценки
PostgreSQL	Модуль формирования результата	Параметры фильтрации
Модуль формирования результата	REST API	Структура итогового ответа
REST API	Клиентское приложение	Результат классификации в формате JSON
Серверные компоненты	PostgreSQL	Сообщение и результат обработки
Серверные компоненты	Подсистема журналирования	Служебные события и ошибки

Схема обмена данными между компонентами представлена на рисунке 7.

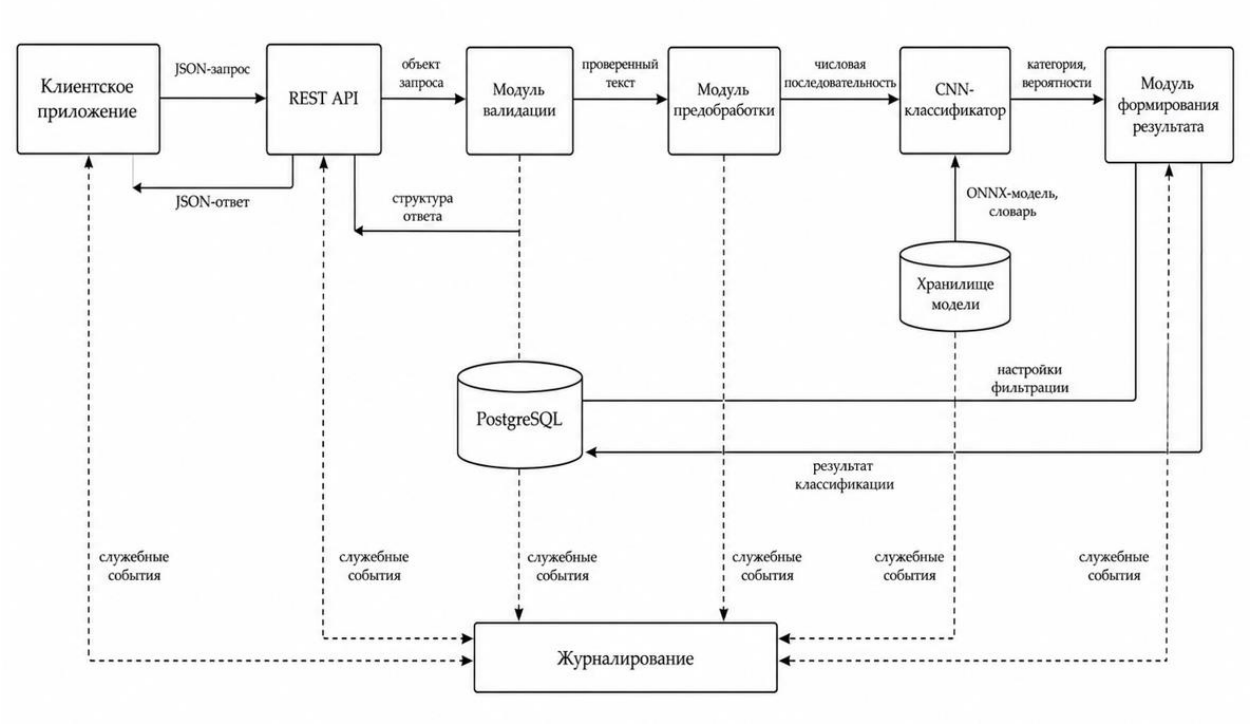

Рисунок 7 - Схема обмена данными между компонентами системы

Принятый порядок взаимодействия закрепляет назначение программных компонентов. Клиентская часть не зависит от внутреннего устройства классификатора, а серверные модули обмениваются данными посредством установленных структур. Изменение модели, способа хранения или правил отображения может выполняться при сохранении формата входного запроса и ответа.

3 Программная реализация и экспериментальная оценка системы классификации текстовых сообщений

3.1 Выбор технологий реализации

Программная реализация системы классификации текстовых сообщений включает подготовку экспериментального набора данных, обучение нейросетевой модели, организацию серверной обработки запросов, выполнение классификации и сохранение полученных результатов. Состав программных средств определялся с учётом архитектуры, разработанной во второй главе, требований к обработке сообщений и необходимости разделения процессов обучения и эксплуатации модели.

Серверная часть реализована на языке JavaScript в среде Node.js. Выбранная среда обеспечивает обработку сетевых запросов, выполнение операций предварительной подготовки текста и взаимодействие с подсистемой хранения данных. Серверный модуль принимает сообщение от клиентского приложения, передаёт его в последовательность классификации и формирует результат в виде структурированного ответа.

Взаимодействие между клиентской и серверной частями организовано посредством REST API. Клиентское приложение передаёт текст сообщения и служебные параметры в составе HTTP-запроса. После завершения обработки сервер возвращает определённую категорию, уровень уверенности классификатора и параметры отображения. Для представления запросов и ответов применяется формат JSON.

Подготовка экспериментального набора данных и обучение свёрточной нейронной сети выполняются на языке Python. Такое разделение связано с наличием специализированных средств для обработки табличной информации, построения нейросетевых моделей и расчёта показателей качества. Среда обучения не входит в серверный контур и используется на этапе формирования итоговой версии классификатора.

Для реализации CNN-модели применяются TensorFlow и Keras. С их помощью задаются слои нейронной сети, параметры свёрточных операций,

функция потерь и порядок обновления весовых коэффициентов. После завершения обучения выполняется проверка модели на валидационной и тестовой выборках. Состояние классификатора, показавшее наилучшее значение выбранного показателя качества, используется для последующего экспорта.

Обученная нейронная сеть преобразуется в формат ONNX. Данный формат содержит структуру модели и рассчитанные в процессе обучения весовые коэффициенты. Вместе с моделью сохраняются словарь токенов, перечень категорий, максимальная длина последовательности и параметры предварительной обработки. Наличие перечисленных артефактов обеспечивает одинаковое преобразование текстов на этапах обучения и серверной классификации.

Выполнение обученной модели на серверной стороне осуществляется средствами ONNX Runtime. При запуске серверного приложения создаётся рабочая сессия и загружается файл модели. Полученная сессия сохраняется в памяти и повторно используется при обработке последующих сообщений. Повторная загрузка модели для каждого запроса не выполняется.

Серверный модуль предварительной обработки использует словарь и параметры, подготовленные при обучении. Текст нормализуется, разбивается на токены и преобразуется в последовательность числовых идентификаторов. Максимальная длина входной последовательности составляет 200 токенов. Сообщения большей длины сокращаются, а более короткие последовательности дополняются служебными значениями до требуемого размера.

Для хранения сообщений, параметров фильтрации и результатов классификации применяется PostgreSQL. Серверная организация СУБД обеспечивает одновременную работу нескольких компонентов, поддержку транзакций и контроль целостности записей. В базе данных сохраняются исходный текст сообщения, определённая категория, уровень уверенности, вероятностные оценки и время обработки.

Обращение к PostgreSQL выполняется через отдельный модуль доступа к данным. Выделение данного компонента исключает размещение запросов к базе данных внутри процедур предобработки и классификации. Для повторного использования соединений применяется пул подключений, сокращающий количество операций открытия и закрытия соединения при последовательной обработке сообщений.

Подготовка экспериментальных таблиц выполняется средствами Pandas и NumPy. Библиотеки используются при загрузке исходных корпусов, удалении дублирующихся записей, преобразовании категорий и формировании обучающей, валидационной и тестовой выборок. Расчёт Accuracy, Precision, Recall, F1-меры и матрицы ошибок выполняется средствами Scikit-learn.

Выбранный набор программных средств обеспечивает разделение процессов подготовки модели и её применения. Обучение классификатора выполняется в среде Python, а обработка входящих сообщений производится серверной частью в среде Node.js. Передача модели в формате ONNX исключает необходимость запуска обучающей среды на сервере и позволяет использовать полученные весовые коэффициенты непосредственно при классификации новых сообщений.

Основные программные средства, используемые при реализации системы, представлены в таблице 8.

Таблица 8 - Средства и технологии программной реализации

Программное средство	Назначение
JavaScript	Реализация серверных компонентов
Node.js	Среда выполнения серверной части
REST API	Взаимодействие с клиентским приложением
Python	Подготовка данных и обучение нейросетевой модели
TensorFlow, Keras	Построение и обучение CNN-модели
ONNX	Формат хранения обученной модели
ONNX Runtime	Выполнение классификатора на серверной стороне
PostgreSQL	Хранение сообщений, настроек и результатов
Pandas	Обработка и объединение исходных наборов данных
NumPy	Выполнение операций с числовыми массивами
Scikit-learn	Разделение выборки и расчёт показателей качества
JSON	Представление запросов и результатов обработки

Выбранный набор программных средств обеспечивает разделение процессов подготовки модели и её применения. Обучение классификатора выполняется в среде Python, а обработка входящих сообщений производится серверной частью в среде Node.js. Передача модели в формате ONNX исключает необходимость запуска обучающей среды на сервере и позволяет использовать полученные весовые коэффициенты непосредственно при классификации новых сообщений.

3.2 Формирование и подготовка экспериментального набора данных

Обучение системы классификации текстовых сообщений требует размеченного корпуса, содержащего примеры всех категорий, предусмотренных постановкой задачи. Состав выборки должен учитывать особенности коротких сообщений, различия между нейтральной перепиской, рекламными предложениями, массовыми рассылками и текстами с признаками оскорбительного содержания.

Экспериментальный набор формировался путём объединения открытых текстовых корпусов и дополнительно подготовленных сообщений. Для категории спама использовались материалы SMS Spam Collection, включающие обычные SMS-сообщения и нежелательные рассылки. Категория оскорблений формировалась на основе открытых русскоязычных корпусов токсичных комментариев. В состав рекламной категории включались тексты коммерческих объявлений, сведения о скидках, акциях, товарах и услугах. Полезный контент представлен нейтральными сообщениями, организационной перепиской и информационными уведомлениями.

Часть исходных файлов загружалась с использованием программного интерфейса Kaggle API. Авторизация выполнялась посредством персонального токена, размещённого вне программного кода. Полученные наборы сохранялись локально и далее обрабатывались средствами Pandas.

Исходные наборы различались по структуре, наименованию полей и способу обозначения целевых классов. Перед объединением названия столбцов приводились к единому виду, а исходные метки преобразовывались в категории, принятые в работе: «полезный контент», «спам», «реклама» и «оскорбления».

На этапе очистки исключались записи с отсутствующим или пустым текстом. Дублирующиеся сообщения удалялись до разделения корпуса на выборки. Данная операция предотвращает попадание одинаковых текстов одновременно в обучающую и тестовую части, способное привести к завышению итоговых показателей качества.

Источники и назначение данных представлены в таблице 9.

Таблица 9 - Источники данных экспериментальной выборки

Категория	Характеристика сообщения	Текстовая метка
Полезный контент	Нейтральные сообщения из открытых диалоговых корпусов и SMS-наборов	Целое число
Спам	SMS Spam Collection и открытые наборы нежелательных рассылок	Нежелательный контент
Реклама	Открытые тексты объявлений и коммерческих предложений	Нежелательный контент
Оскорбления	Русскоязычные корпуса токсичных комментариев	Нежелательный контент

Дополнительная проверка выполнялась для сообщений, расположенных на границе категорий «спам» и «реклама». Коммерческое предложение относилось к рекламе при отсутствии выраженных признаков массовой нежелательной рассылки. Сообщения с повторяющимися призывами, подозрительными ссылками и навязчивым характером распространения относились к спаму. Записи с неоднозначной разметкой исключались из итогового корпуса.

Фрагмент объединения и очистки исходных данных представлен в листинге 1.

Листинг 1 - Объединение и очистка исходных наборов данных

```
import pandas as pd

CATEGORY_MAP = {
    "ham": "полезный контент",
    "safe": "полезный контент",
    "spam": "спам",
    "advertisement": "реклама",
    "toxic": "оскорбления",
}


def prepare_dataset(dataframes: list[pd.DataFrame]) ->
pd.DataFrame:
```

```
    dataset = pd.concat(dataframes, ignore_index=True)

    dataset = dataset.rename(
        columns={
            "text": "message_text",
            "label": "message_category",
        }
    )

    dataset = dataset.dropna(
        subset=["message_text", "message_category"]
    )

    dataset["message_text"] = (
        dataset["message_text"]
        .astype(str)
        .str.strip()
    )

    dataset = dataset[
        dataset["message_text"].str.len() > 0
    ]

    dataset["message_category"] = (
        dataset["message_category"]
        .map(CATEGORY_MAP)
    )

    dataset = dataset.dropna(
        subset=["message_category"]
    )

    dataset = dataset.drop_duplicates(
        subset=["message_text"]
    )

    return dataset.reset_index(drop=True)
```

Увеличенная доля полезного контента соответствует характеру обычной пользовательской переписки, в которой нейтральные сообщения преобладают над нежелательными. Количество записей остальных классов выровнено, что обеспечивает достаточное представление каждой категории при обучении классификатора.

После объединения, очистки и выравнивания количества записей был сформирован набор объёмом 100 000 сообщений. Распределение данных по категориям приведено в таблице 10.

Таблица 10 - Распределение сообщений по категориям

Категория	Характеристика сообщения
Полезный контент	40000
Спам	20000
Реклама	20000
Оскорбления	20000
Всего	100000

Увеличенная доля полезного контента соответствует характеру обычной пользовательской переписки, в которой нейтральные сообщения преобладают над нежелательными. Количество записей остальных классов выровнено, что обеспечивает достаточное представление каждой категории при обучении классификатора.

Подготовленный корпус разделялся на обучающую, валидационную и тестовую выборки. Обучающая часть содержала 70 000 сообщений. Валидационная и тестовая части включали по 15 000 записей. Разделение выполнялось стратифицированным способом с сохранением исходного соотношения категорий.

Обучающая выборка использовалась для изменения весовых коэффициентов нейронной сети. Валидационная часть применялась для контроля процесса обучения и выбора состояния модели. Тестовая выборка не участвовала в настройке классификатора и использовалась только при итоговой оценке качества.

Фрагмент разделения корпуса представлен в листинге 2.

Листинг 2 - Формирование обучающей, валидационной и тестовой выборок

```
from sklearn.model_selection import train_test_split

train_data, temporary_data = train_test_split(
    dataset,
```

```
    test_size=0.30,
    stratify=dataset["message_category"],
    random_state=42,
)

validation_data, test_data = train_test_split(
    temporary_data,
    test_size=0.50,
    stratify=temporary_data["message_category"],
    random_state=42,
)
```

Словарь токенов формировался только по обучающей выборке. Такой порядок исключает использование сведений из валидационной и тестовой частей при подготовке модели. Наиболее часто встречающимся токенам присваивались числовые идентификаторы. Для неизвестных слов и дополнения последовательностей предусматривались отдельные служебные значения.

Максимальная длина входной последовательности составляла 200 токенов. Более длинные сообщения сокращались до установленного размера. Последовательности меньшей длины дополнялись нулевыми значениями. Словарь токенов, максимальная длина и параметры нормализации сохранялись вместе с обученной CNN-моделью и использовались при классификации сообщений на серверной стороне.

Подготовленный корпус содержит достаточное количество примеров всех рассматриваемых категорий. Разделение данных на независимые выборки обеспечивает контроль процесса обучения и позволяет оценить качество модели на сообщениях, не использованных при настройке её параметров.

3.3 Реализация модулей системы

Программная реализация системы построена в соответствии с архитектурой, разработанной во второй главе. Серверная часть выполняет приём и проверку запросов, подготовку текста, запуск обученного классификатора, применение параметров фильтрации и сохранение результата. Подготовка экспериментальных данных и обучение свёрточной нейронной сети выполняются отдельно в среде Python.

Разделение процессов обучения и эксплуатации модели связано с различием выполняемых задач. Средства Python применяются для формирования словаря, построения CNN-модели и настройки её весовых коэффициентов. После завершения обучения модель преобразуется в формат ONNX. Серверная часть получает готовую модель, словарь токенов и параметры предварительной обработки, после чего использует их при анализе новых сообщений.

В составе программной реализации выделены следующие модули:

– обработка входного запроса,

– предварительная обработка текста,

– обучение CNN-модели,

– выполнение классификатора на сервере,

– применение параметров фильтрации,

– формирование ответа,

– сохранение результатов в PostgreSQL.

Каждый модуль выполняет отдельную операцию и обменивается с другими компонентами посредством установленных структур данных. Такой порядок исключает размещение логики классификации непосредственно в обработчике сетевого запроса и упрощает проверку отдельных частей системы.

3.3.1 Реализация обработки входного запроса

Обработка сообщения начинается с получения объекта запроса от клиентского приложения. В состав объекта входят идентификатор сообщения, исходный текст и параметры фильтрации. Перед запуском классификатора проверяются наличие обязательных полей, тип переданных значений и допустимая длина текста.

Максимальный размер исходного сообщения ограничен 4096 символами. Данное ограничение применяется до токенизации и предотвращает передачу в последовательность обработки текстов,

значительно превышающих размеры сообщений, характерных для рассматриваемой предметной области.

После успешной проверки запрос последовательно передаётся модулям предобработки, классификации и формирования результата. Основная функция обработки представлена в листинге 3.

Листинг 3 - Обработка запроса классификации сообщения

```
export async function processClassificationRequest(
    requestData,
    userId
) {
    const request = validateClassificationRequest(requestData);

    const preparedMessage = preprocessMessage(
        request.messageText
    );

    const classification = await classifyMessage(
        preparedMessage.tokenIds
    );

    const displayParameters = applyFilterSettings(
        classification,
        request.filters
    );

    await saveClassificationResult({
        messageId: request.messageId,
        userId,
        originalText: request.messageText,
        normalizedText: preparedMessage.normalized,
        classification,
        displayParameters,
    });

    return buildClassificationResponse(
        request.messageId,
        classification,
        displayParameters
    );
}
```

Функция не содержит внутренней реализации нейронной сети или операций с базой данных. Её назначение состоит в координации последовательности вызовов и передаче результатов между компонентами.

Проверка входного объекта выполняется отдельной функцией. Фрагмент валидации приведён в листинге 4.

Листинг 4 - Проверка входных данных

```
function validateClassificationRequest(data) {
    if (!data || typeof data !== 'object') {
        throw new Error('Некорректная структура запроса');
    }

    if (typeof data.messageText !== 'string'
            || !data.messageText.trim()) {
        throw new Error('Текст сообщения не указан');
    }

    if (data.messageText.length > 4096) {
        throw new Error(
            'Превышена допустимая длина сообщения'
        );
    }

    return {
        messageId: data.messageId,
        messageText: data.messageText.trim(),
        filters: data.filters ?? {
            hiddenCategories: [],
            showTags: true,
        },
    };
}
```

Размещение проверки до этапа предобработки исключает выполнение нейросетевой модели для пустых и некорректных запросов.

3.3.2 Реализация модуля предварительной обработки текста

Модуль предварительной обработки преобразует исходное сообщение в числовую последовательность, соответствующую входу CNN-модели. На сервере применяются те же правила нормализации и тот же словарь, которые использовались при подготовке обучающей выборки.

Текст приводится к нижнему регистру, нормализуется по стандарту Unicode, очищается от управляющих символов и разбивается на токены. Символ ё заменяется на е, что уменьшает количество различных вариантов представления одинаковых слов. Последовательности пробелов сокращаются до одного символа.

Каждый токен сопоставляется с числовым идентификатором из словаря. Для слов, отсутствующих в словаре, используется специальный идентификатор неизвестного токена. Максимальная длина входной

последовательности составляет 200 элементов. Короткие последовательности дополняются нулевыми значениями.

Реализация модуля представлена в листинге 5.

Листинг 5 - Предварительная обработка текстового сообщения

```
const MAX_SEQUENCE_LENGTH = 200;
const PAD_TOKEN_ID = 0;
const UNKNOWN_TOKEN_ID = 1;

export function preprocessMessage(text, vocabulary) {
    const normalized = text
        .normalize('NFKC')
        .toLowerCase()
        .replace(/ё/g, 'е')
        .replace(/[\u200B-\u200D\uFEFF]/g, '')
        .replace(/\s+/g, ' ')
        .trim();

    const tokens = normalized.match(
        /[а-яа-z0-9]+/gi
    ) ?? [];

    const tokenIds = tokens
        .slice(0, MAX_SEQUENCE_LENGTH)
        .map(token =>
            vocabulary[token] ?? UNKNOWN_TOKEN_ID
        );

    while (tokenIds.length < MAX_SEQUENCE_LENGTH) {
        tokenIds.push(PAD_TOKEN_ID);
    }

    return {
        normalized,
        tokenIds,
    };
}
```

Правила предобработки не включают удаление всех коротких слов и стоп-слов. Такие элементы могут участвовать в формировании локальных сочетаний, используемых свёрточными фильтрами. Исключение выполняется только для технических и управляющих символов, не несущих содержательной информации.

Словарь токенов формируется по обучающей выборке и сохраняется в формате JSON. Использование словаря, построенного только на обучающих

данных, предотвращает попадание сведений из тестовой выборки в процесс подготовки классификатора.

3.3.3 Реализация и обучение CNN-модели

Свёрточная нейронная сеть реализована средствами TensorFlow и Keras. Архитектура построена по принципу TextCNN и ориентирована на выделение локальных сочетаний токенов различной длины.

На вход модели передаётся последовательность из 200 числовых идентификаторов. Слой Embedding преобразует каждый идентификатор в вектор размерностью 128. Далее используются три параллельных свёрточных слоя с размерами ядра 3, 4 и 5. Такой набор позволяет выделять признаки, соответствующие сочетаниям различной длины.

После свёрточных операций выполняется глобальное максимальное объединение признаков. Полученные векторы объединяются, проходят через слой регуляризации Dropout и передаются выходному слою Softmax. Количество выходных нейронов равно четырём по числу рассматриваемых категорий.

Архитектура модели приведена в листинге 6.

Листинг 6 - Построение CNN-модели классификации сообщений

```
import tensorflow as tf
from tensorflow.keras import layers, Model

MAX_LENGTH = 200
EMBEDDING_DIM = 128
NUM_CLASSES = 4


def build_message_cnn(vocabulary_size: int) -> Model:
    inputs = layers.Input(
        shape=(MAX_LENGTH,),
        dtype="int32",
        name="token_ids",
    )

    embeddings = layers.Embedding(
        input_dim=vocabulary_size,
        output_dim=EMBEDDING_DIM,
        mask_zero=False,
        name="embedding",
    )(inputs)
```

```
    branches = []

    for kernel_size in (3, 4, 5):
        features = layers.Conv1D(
            filters=128,
            kernel_size=kernel_size,
            activation="relu",
            name=f"conv_{kernel_size}",
        )(embeddings)

        features = layers.GlobalMaxPooling1D(
            name=f"pool_{kernel_size}"
        )(features)

        branches.append(features)

    merged = layers.Concatenate()(branches)
    merged = layers.Dropout(0.5)(merged)

    outputs = layers.Dense(
        NUM_CLASSES,
        activation="softmax",
        name="category_probabilities",
    )(merged)

    return Model(
        inputs=inputs,
        outputs=outputs,
        name="message_cnn",
    )
```

Для обучения применяется оптимизатор Adam с коэффициентом обучения 0,001. В качестве функции потерь используется категориальная перекрёстная энтропия. Выбор данной функции соответствует многоклассовой постановке задачи, при которой каждому сообщению назначается одна итоговая категория.

Перечень классов задаётся в следующем порядке:

```
CATEGORY_LABELS = [
    "полезный контент",
    "спам",
    "реклама",
    "оскорбления",
]
```

Порядок категорий сохраняется при подготовке целевых меток и обработке выходного вектора. Изменение порядка после обучения привело бы к неправильному сопоставлению вероятностных оценок с названиями классов.

Процесс настройки модели представлен в листинге 7.

Листинг 7 - Обучение CNN-модели

```
model = build_message_cnn(
    vocabulary_size=len(vocabulary)
)

model.compile(
    optimizer=tf.keras.optimizers.Adam(
        learning_rate=0.001
    ),
    loss="categorical_crossentropy",
    metrics=["accuracy"],
)

checkpoint = tf.keras.callbacks.ModelCheckpoint(
    filepath="best_message_cnn.keras",
    monitor="val_loss",
    save_best_only=True,
)

early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    patience=3,
    restore_best_weights=True,
)

history = model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=20,
    callbacks=[checkpoint, early_stopping],
)
```

Валидационная выборка используется для контроля изменения функции потерь и прекращения обучения при отсутствии улучшения. Тестовые сообщения в данном процессе не участвуют. Итоговая проверка выполняется после выбора лучшего состояния модели.

После завершения обучения структура сети и рассчитанные весовые коэффициенты преобразуются в формат ONNX. Вместе с моделью сохраняются словарь, перечень категорий и максимальная длина последовательности. Полученные файлы передаются в серверную часть системы.

3.3.4 Выполнение CNN-модели на серверной стороне

При запуске серверной части создаётся сессия ONNX Runtime. Загрузка выполняется один раз, после чего сессия сохраняется в памяти и используется при обработке следующих запросов.

На вход модели передаётся тензор размерности 1 × 200, содержащий идентификаторы токенов одного сообщения. Выход модели представляет собой массив из четырёх вероятностных оценок. Итоговая категория определяется по индексу максимального значения.

Фрагмент серверного классификатора приведён в листинге 8.

Листинг 8 - Выполнение ONNX-модели

```
import * as onnx from 'onnxruntime-node';

const CATEGORY_LABELS = [
    'полезный контент',
    'спам',
    'реклама',
    'оскорбления',
];

let modelSession;

export async function loadClassificationModel(
    modelPath
) {
    modelSession = await onnx.InferenceSession.create(
        modelPath,
        {
            executionProviders: ['cpu'],
        }
    );
}

export async function classifyMessage(tokenIds) {
    if (!modelSession) {
        throw new Error(
            'Модель классификации не загружена'
        );
    }

    const inputData = BigInt64Array.from(
        tokenIds,
        value => BigInt(value)
    );

    const inputTensor = new onnx.Tensor(
```

```
            'int64',
            inputData,
            [1, 200]
        );

        const output = await modelSession.run({
            token_ids: inputTensor,
        });

        const probabilities = Array.from(
            output.category_probabilities.data
        );

        const predictedIndex = probabilities.indexOf(
            Math.max(...probabilities)
        );

        return {
            category: CATEGORY_LABELS[predictedIndex],
            confidence: probabilities[predictedIndex],
            probabilities,
        };
}
```

Тип входного тензора должен соответствовать типу, указанному при экспорте модели. В приведённом варианте используются 64-разрядные целые значения. Несоответствие типа или размерности входа приводит к прекращению выполнения модели, поэтому данные параметры сохраняются вместе с остальными характеристиками классификатора.

3.3.5 Реализация механизма фильтрации

Механизм фильтрации выполняется после определения категории и не изменяет результат нейросетевой модели. Его назначение состоит в выборе способа отображения сообщения в соответствии с пользовательскими настройками.

Категория «полезный контент» не требует маркировки. Для спама, рекламы и оскорблений формируется текстовая метка. Если определённая категория включена в список скрываемых, клиентскому приложению передаётся действие hide. В остальных случаях сообщение отображается с предупреждением либо без дополнительной обработки.

Реализация представлена в листинге 9.

Листинг 9 - Применение параметров фильтрации

```
const CATEGORY_TAGS = {
    'полезный контент': null,
    'спам': 'Спам',
    'реклама': 'Реклама',
    'оскорбления': 'Оскорбительное содержимое',
};

export function applyFilterSettings(
    classification,
    settings
) {
    const category = classification.category;
    const warningFlag =
        category !== 'полезный контент';

    const hiddenCategories =
        settings.hiddenCategories ?? [];

    const shouldHide =
        hiddenCategories.includes(category);

    let displayAction = 'show';

    if (shouldHide) {
        displayAction = 'hide';
    } else if (warningFlag && settings.showTags) {
        displayAction = 'show_warning';
    }

    return {
        warningFlag,
        categoryTag: CATEGORY_TAGS[category],
        displayAction,
    };
}
```

Параметр displayAction носит рекомендательный характер. Фактическое скрытие или отображение выполняется клиентской частью после получения ответа.

3.3.6 Формирование ответа клиентскому приложению

Выходной объект формируется после завершения классификации и применения параметров фильтрации. Его структура соответствует информационной модели, приведённой в пункте 1.4.

Листинг 10 - Формирование результата классификации

```
export function buildClassificationResponse(
    messageId,
```

```
    classification,
    displayParameters
) {
    return {
        message_id: messageId,
        predicted_category:
            classification.category,
        confidence_score: Number(
            classification.confidence.toFixed(4)
        ),
        warning_flag:
            displayParameters.warningFlag,
        category_tag:
            displayParameters.categoryTag,
        display_action:
            displayParameters.displayAction,
    };
}
```

Пример сформированного ответа:

```
{
  "message_id": 105,
  "predicted_category": "реклама",
  "confidence_score": 0.94,
  "warning_flag": true,
  "category_tag": "Реклама",
  "display_action": "show_warning"
}
```

Использование единой структуры ответа позволяет изменять внутреннюю реализацию классификатора без переработки клиентского механизма отображения.

3.3.7 Сохранение результатов в PostgreSQL

Результаты классификации сохраняются в отдельной таблице PostgreSQL. Запись содержит идентификатор сообщения, пользователя, исходный и нормализованный текст, итоговую категорию, вероятностные оценки, версию модели и параметры отображения.

Структура таблицы представлена в листинге 11.

Листинг 11 - Создание таблицы результатов классификации

```
CREATE TABLE message_classifications (
    id UUID PRIMARY KEY,
    message_id BIGINT NOT NULL,
    user_id BIGINT NOT NULL,
    original_text TEXT NOT NULL,
    normalized_text TEXT NOT NULL,
    predicted_category VARCHAR(64) NOT NULL,
    confidence_score DOUBLE PRECISION NOT NULL,
```

```
    probabilities JSONB NOT NULL,
    warning_flag BOOLEAN NOT NULL,
    category_tag VARCHAR(64),
    display_action VARCHAR(32) NOT NULL,
    model_version VARCHAR(32) NOT NULL,
    classified_at TIMESTAMPTZ NOT NULL
        DEFAULT CURRENT_TIMESTAMP
);
```

Хранение полного распределения вероятностей в поле JSONB позволяет выявлять сообщения с близкими оценками нескольких категорий. Поле model_version используется при сравнении результатов, полученных различными состояниями классификатора.

Сохранение выполняется через отдельный модуль доступа к данным. Фрагмент операции записи приведён в листинге 12.

Листинг 12 - Сохранение результата в PostgreSQL

```
export async function saveClassificationResult(data) {
    const query = `
        INSERT INTO message_classifications (
            id,
            message_id,
            user_id,
            original_text,
            normalized_text,
            predicted_category,
            confidence_score,
            probabilities,
            warning_flag,
            category_tag,
            display_action,
            model_version
        )
        VALUES (
            gen_random_uuid(),
            $1, $2, $3, $4, $5, $6,
            $7, $8, $9, $10, $11
        )
    `;

    const values = [
        data.messageId,
        data.userId,
        data.originalText,
        data.normalizedText,
        data.classification.category,
        data.classification.confidence,
        JSON.stringify(
            data.classification.probabilities
        ),
```

```
        data.displayParameters.warningFlag,
        data.displayParameters.categoryTag,
        data.displayParameters.displayAction,
        '1.0.0',
    ];

    await database.query(query, values);
}
```

Параметризованный запрос исключает непосредственную подстановку входных значений в SQL-строку. Операции хранения отделены от модулей предобработки и классификации, поэтому изменение структуры базы данных не требует изменения нейросетевой модели.

Реализованные компоненты образуют последовательный контур обработки текстового сообщения. Серверная часть получает и проверяет запрос, преобразует текст в числовую последовательность, запускает ONNX-модель, применяет настройки отображения, сохраняет результат и формирует ответ клиентскому приложению. Разделение обучения и серверного выполнения позволяет обновлять классификатор без изменения общего порядка взаимодействия программных компонентов.

3.4 Результаты тестирования программной реализации

Функциональное тестирование проводилось после объединения модулей предварительной обработки, классификации, фильтрации, формирования ответа и сохранения результатов. Цель проверки заключалась в подтверждении соответствия программной реализации требованиям, сформулированным при проектировании системы.

Тестирование выполнялось на персональном компьютере с процессором Intel Core i5-10400F без использования графического ускорителя. CNN-модель запускалась средствами ONNX Runtime на центральном процессоре. Серверная часть, PostgreSQL и клиентское приложение функционировали в локальной вычислительной среде.

3.4.1 Методы функционального тестирования

Проверка проводилась методом функционального тестирования, при котором оценивалось соответствие результата заданному сценарию без

анализа внутреннего состояния отдельных программных компонентов. Дополнительно применялись методы граничных значений и разбиения входных данных на классы эквивалентности.

В качестве классов эквивалентности рассматривались корректные сообщения четырёх категорий, пустой текст, сообщения предельной длины, последовательности с неизвестными токенами и запросы с различными параметрами фильтрации. Метод граничных значений использовался при проверке ограничения исходного текста и последовательности длиной 200 токенов.

Проверка выполнялась на нескольких уровнях:

– тестирование предварительной обработки текста,

– проверка выполнения CNN-модели,

– тестирование механизма фильтрации,

– проверка структуры серверного ответа,

– контроль сохранения результата в PostgreSQL,

– интеграционная проверка полного цикла обработки.

Для каждого сценария определялись входные данные, ожидаемый результат и фактически полученное состояние системы. Тест считался пройденным при совпадении полученного результата с установленными требованиями.

3.4.2 Проверка модуля предварительной обработки

При тестировании предварительной обработки контролировались приведение текста к нижнему регистру, замена символа «ё», удаление управляющих символов, нормализация пробелов и преобразование токенов в числовые идентификаторы.

Отдельно проверялись сообщения, содержащие слова, отсутствующие в словаре. Для таких элементов использовался идентификатор неизвестного токена. Последовательности длиной менее 200 элементов дополнялись нулевыми значениями, а более длинные сообщения сокращались до установленного размера.

Результаты проверки подтвердили, что модуль формирует последовательности одинаковой длины и использует параметры, сохранённые на этапе обучения модели. Исходный текст при этом сохраняется отдельно и не заменяется нормализованным представлением.

3.4.3 Проверка CNN-классификатора

Проверка классификатора выполнялась на сообщениях всех четырёх категорий. Для каждого входного текста контролировались размер выходного вектора, диапазон вероятностных оценок и соответствие итоговой категории индексу максимального значения.

Выходной вектор содержал четыре значения, соответствующие категориям «полезный контент», «спам», «реклама» и «оскорбления». Сумма значений после применения функции Softmax составляла единицу с учётом допустимой погрешности вычислений.

Также проверялась загрузка модели при запуске серверной части. Сессия ONNX Runtime создавалась один раз и повторно использовалась при обработке последующих сообщений. Такой порядок исключает повторное чтение файла модели и уменьшает задержку при последовательной классификации.

3.4.4 Проверка механизма фильтрации и формирования ответа

Механизм фильтрации тестировался при различных сочетаниях категории и пользовательских настроек. Для полезного контента формировалось действие обычного отображения без предупреждающей метки. Для остальных категорий выводилась соответствующая информационная отметка.

При включении категории в список скрываемых значению display_action присваивалось значение hide. При отключённом скрытии и разрешённом выводе меток формировалось значение show_warning. Результат классификации при этом не изменялся.

Серверный ответ проверялся на наличие обязательных полей:

– идентификатор сообщения,

– определённая категория,

– уровень уверенности,

– признак предупреждения,

– текстовая метка,

– рекомендуемое действие.

Полученные ответы соответствовали структуре информационного объекта, установленной в пункте 1.4.

3.4.5 Проверка сохранения результатов

После завершения обработки выполнялась проверка записи результата в PostgreSQL. Контролировалось сохранение идентификатора сообщения, исходного и нормализованного текста, итоговой категории, уровня уверенности, вероятностных оценок, параметров отображения и версии модели.

Дополнительно проверялась возможность получения ранее сохранённого результата по идентификатору сообщения. Значения, возвращённые из базы данных, соответствовали результату, сформированному серверным модулем.

Применение параметризованных запросов обеспечивало передачу текстовых значений без непосредственного включения пользовательского ввода в SQL-строку.

3.4.6 Результаты функционального тестирования

Результаты тестирования подтвердили корректную работу основных программных компонентов и их взаимодействие в составе единого контура обработки. Некорректные запросы отклонялись до выполнения нейросетевой модели, а корректные сообщения последовательно проходили этапы предобработки, классификации, применения настроек и сохранения результата.

Основные сценарии и результаты проверки представлены в таблице 11.

Таблица 11 - Результаты функционального тестирования

№	Проверяемый сценарий	Ожидаемый результат	Результат
1	Передача нейтрального сообщения	Определение категории «полезный контент»	Соответствует
2	Передача сообщения с признаками массовой рассылки	Нежелательный контент	Соответствует
3	Передача коммерческого предложения	Нежелательный контент	Соответствует
4	Передача текста с признаками оскорбления	Нежелательный контент	Соответствует
5	Передача пустого сообщения	Отказ в обработке до запуска модели	Соответствует
6	Превышение допустимой длины текста	Возврат сообщения об ошибке	Соответствует
7	Последовательность длиннее 200 токенов	Сокращение до установленной длины	Соответствует
8	Использование неизвестных слов	Применение служебного идентификатора	Соответствует
9	Категория включена в список скрываемых	Формирование действия hide	Соответствует
10	Отображение предупреждающей метки	Формирование действия show_warning	Соответствует
11	Формирование серверного ответа	Наличие всех обязательных полей	Соответствует
12	Завершение классификации	Сохранение результата в PostgreSQL	Соответствует

Для иллюстрации работы клиентской части в данном подразделе следует разместить снимки интерфейса приложения.

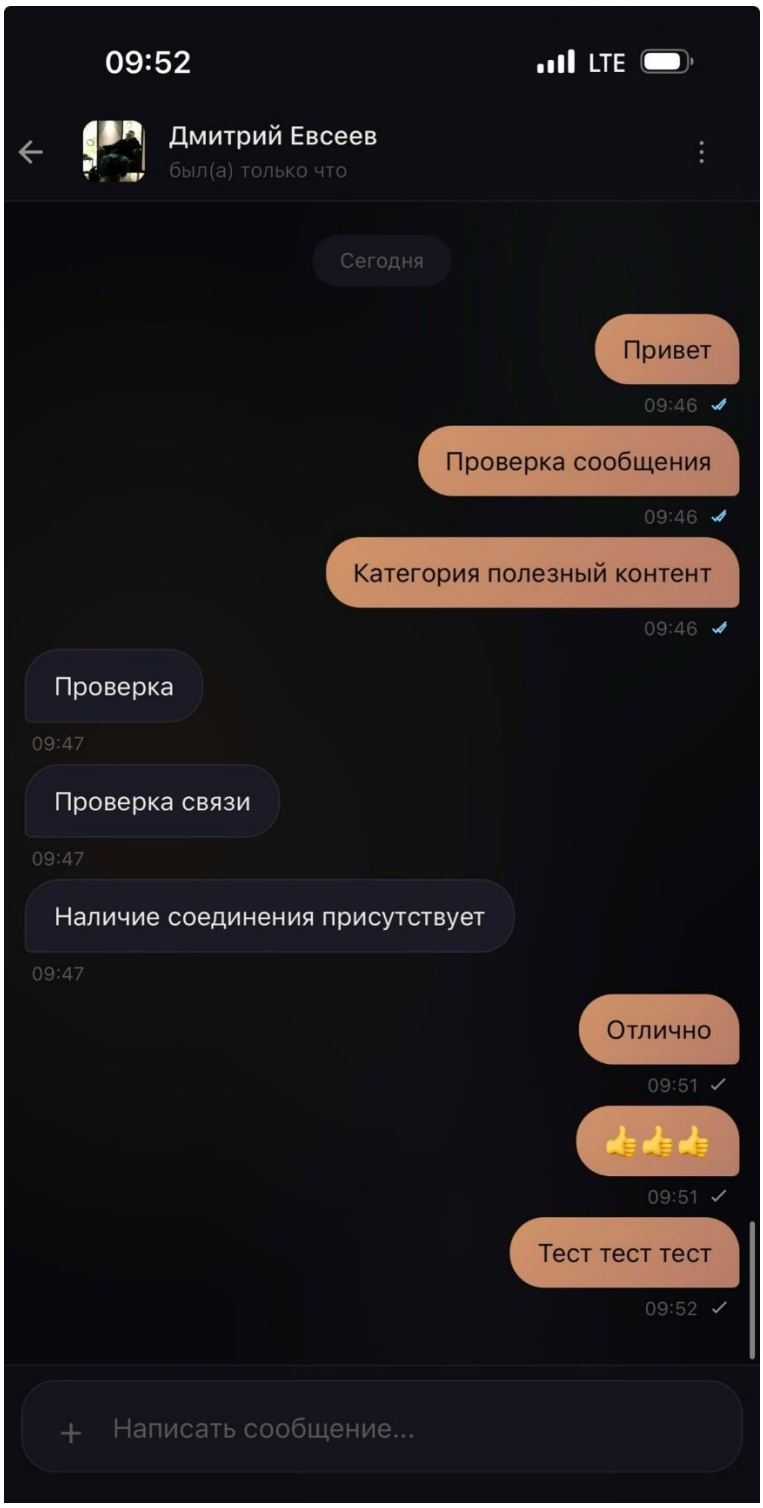

Рисунок 8 – Отображение результата классификации полезного сообщения

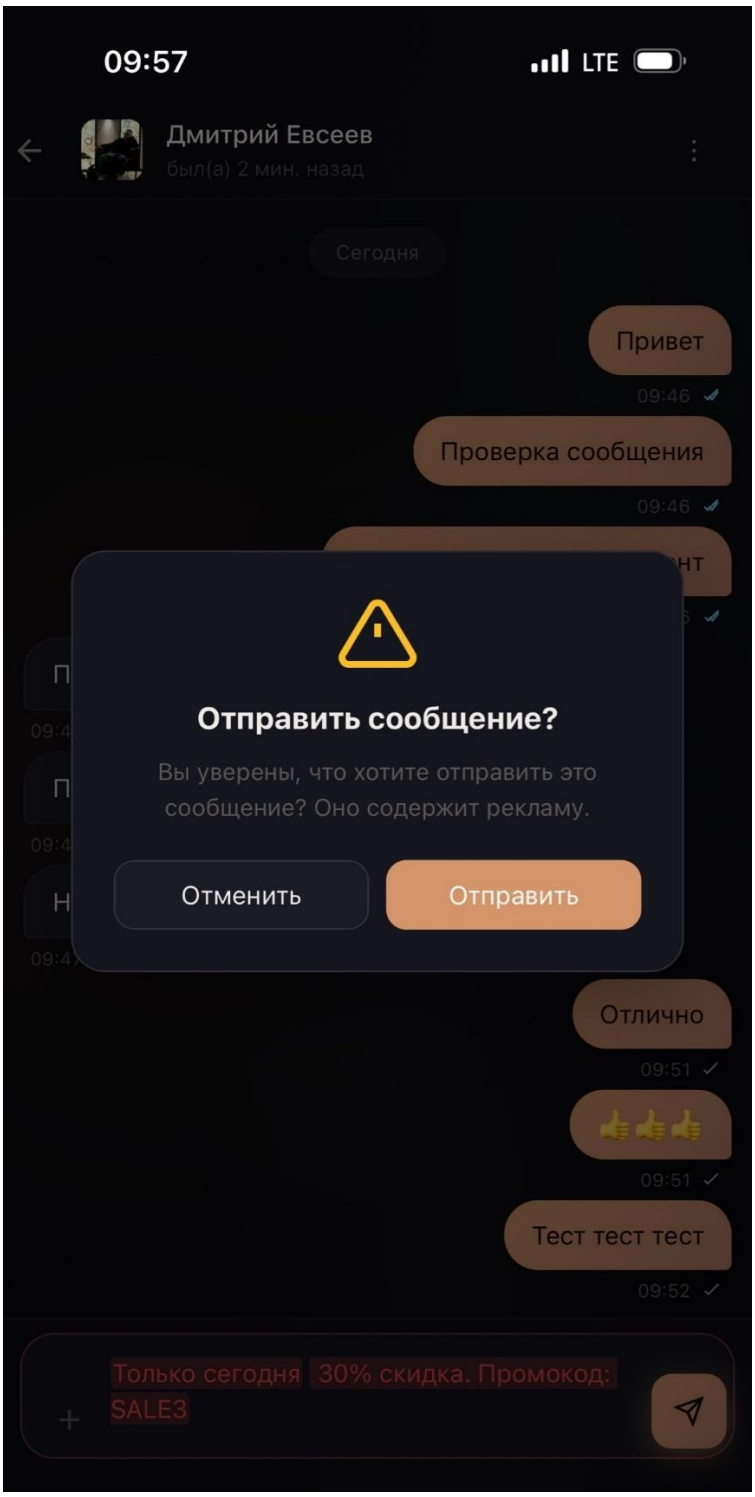

Рисунок 9 - Предупреждение при обнаружении рекламного сообщения

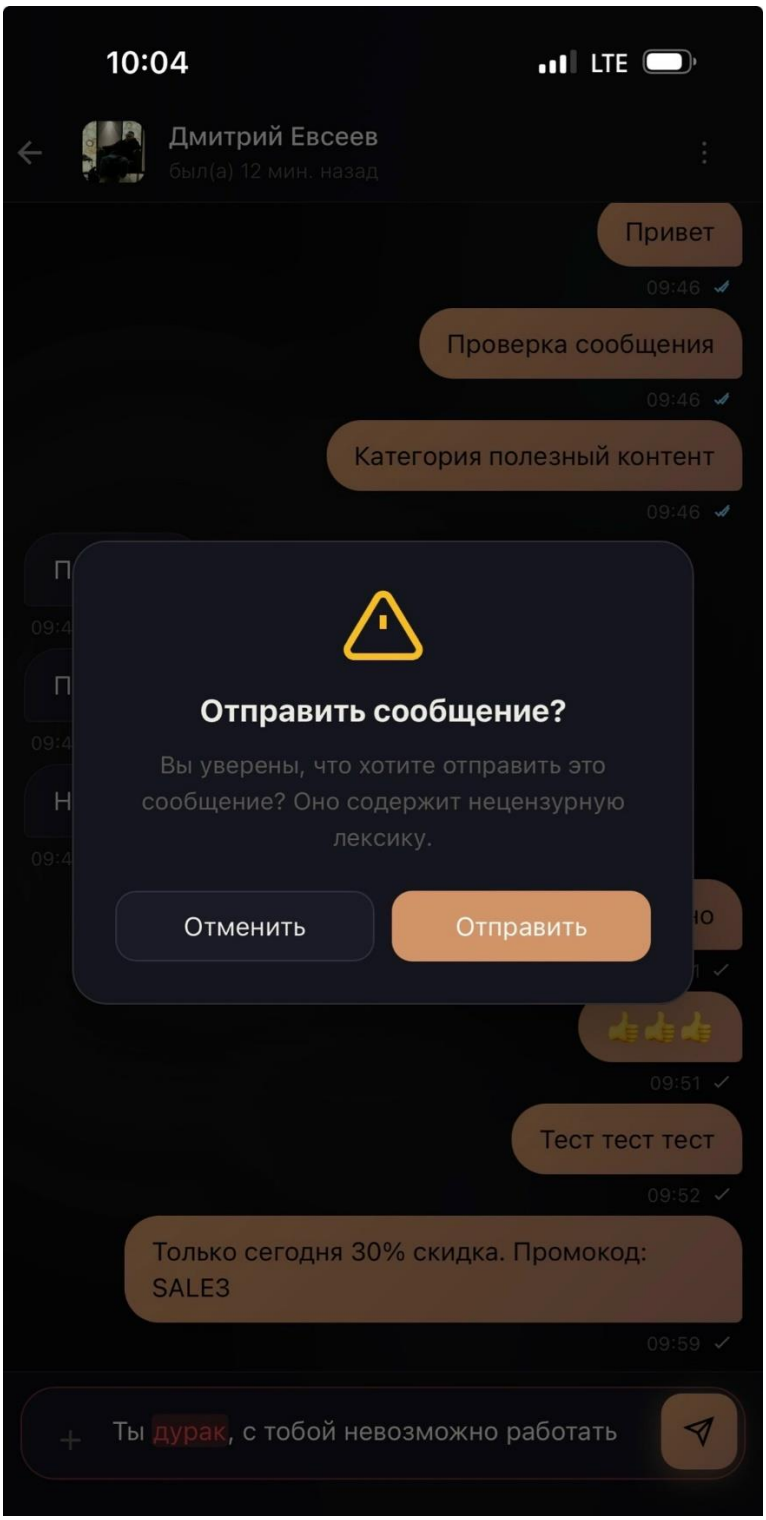

Рисунок 10 - Отображение сообщения с признаками оскорбительного содержания

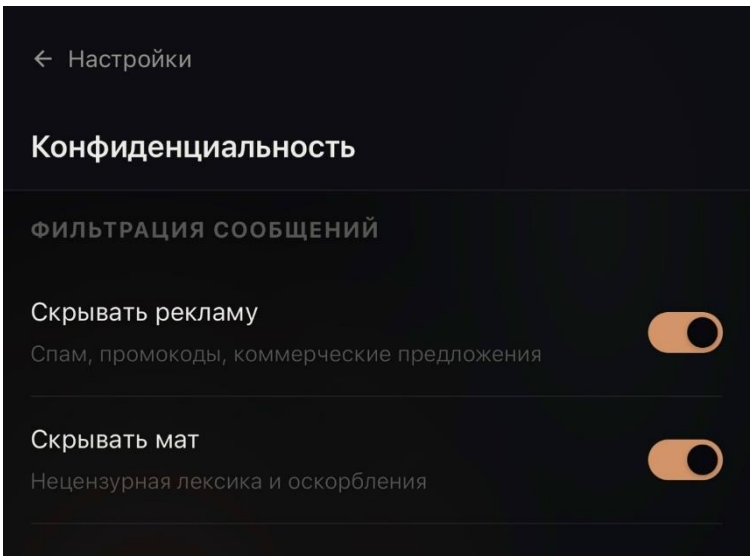

Рисунок 11 - Применение пользовательской настройки скрытия сообщения

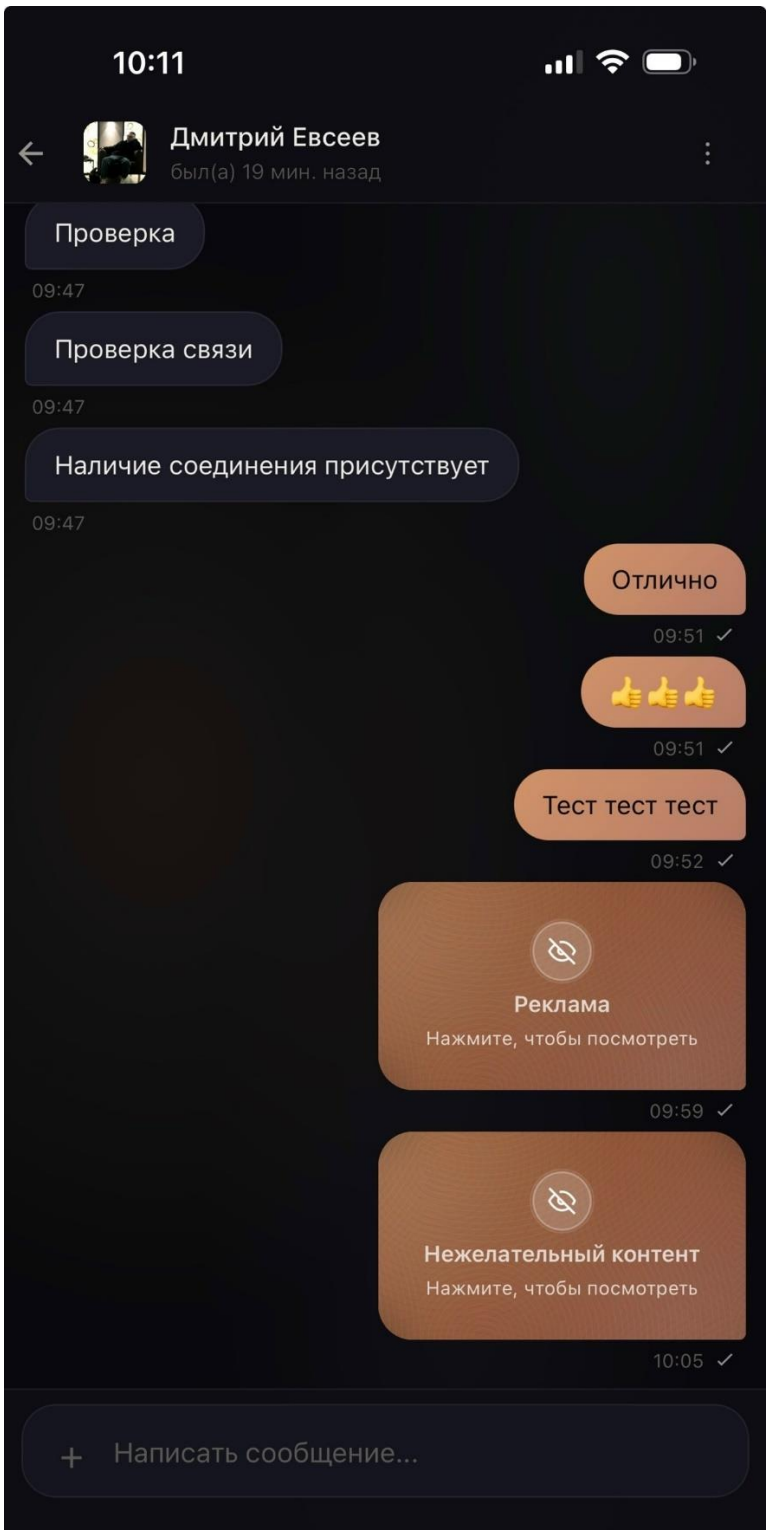

Рисунок 12 - Результат выполнения запроса классификации сообщения

Функциональная проверка подтвердила готовность разработанных модулей к экспериментальной оценке. Следующий этап предусматривает расчёт показателей качества классификации на тестовой выборке и анализ характерных ошибок модели.

3.4 Экспериментальная оценка качества классификации

Экспериментальная оценка проводилась после завершения обучения и выбора состояния модели по результатам валидации. Цель проверки заключалась в определении качества классификации сообщений, не использованных при настройке весовых коэффициентов нейронной сети.

Тестовая выборка содержала 15 000 сообщений. Соотношение категорий соответствовало распределению исходного корпуса. Тестовые данные не применялись при формировании словаря, выборе количества эпох и настройке параметров модели.

Перед классификацией каждое сообщение проходило нормализацию, токенизацию, преобразование токенов в числовые идентификаторы и приведение последовательности к длине 200 элементов. Такой порядок соответствовал подготовке обучающей и валидационной выборок. Использование одинаковых правил обработки исключило различия между данными, применяемыми при обучении модели и при её итоговой проверке.

CNN-модель формировала четыре вероятностные оценки, соответствующие категориям «полезный контент», «спам», «реклама» и «оскорбления». Итоговой считалась категория, получившая наибольшее значение выходного вектора.

Для оценки классификатора использовались показатели Accuracy, Precision, Recall и F1-мера. Accuracy отражает долю сообщений, для которых модель правильно определила категорию, относительно общего объёма тестовой выборки.

Precision показывает, какая часть сообщений, отнесённых моделью к определённой категории, действительно принадлежит данному классу. Высокое значение показателя соответствует небольшому количеству ложных срабатываний.

Recall характеризует долю сообщений определённой категории, которые были обнаружены классификатором. Снижение данного показателя

свидетельствует о наличии сообщений, ошибочно отнесённых к другим классам.

F1-мера учитывает одновременно Precision и Recall. Данный показатель используется для обобщённой оценки качества классификации в условиях, когда требуется учитывать как ложные срабатывания, так и пропущенные сообщения.

Для многоклассовой задачи Precision, Recall и F1-мера рассчитывались отдельно для каждой категории. После этого определялось среднее значение показателя по четырём классам. Макроусреднение обеспечивает равный вклад каждой категории в итоговую оценку независимо от количества содержащихся в ней сообщений.

Расчёт показателей выполнялся средствами Scikit-learn. Фрагмент программного кода представлен в листинге 13.

Листинг 13 - Расчёт показателей качества классификации

```
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix,
)

accuracy = accuracy_score(
    true_labels,
    predicted_labels,
)

precision = precision_score(
    true_labels,
    predicted_labels,
    average="macro",
)

recall = recall_score(
    true_labels,
    predicted_labels,
    average="macro",
)

f1 = f1_score(
    true_labels,
    predicted_labels,
```

```
    average="macro",
)

matrix = confusion_matrix(
    true_labels,
    predicted_labels,
)
```

Результаты экспериментальной оценки представлены в таблице 12.

Показатель	Значение
Accuracy	0,92
Precision	0,91
Recall	0,91
F1-мера	0,91
Среднее время обработки сообщения	4,1 мс

Значение Accuracy показывает, что категория была правильно определена для 92 % сообщений тестовой выборки. Precision, равный 0,91, указывает на небольшую долю сообщений, ошибочно отнесённых к выбранным категориям. Аналогичное значение Recall означает, что классификатор обнаружил основную часть сообщений каждого рассматриваемого класса.

Значение F1-меры составило 0,91. Близость показателей Precision, Recall и F1-меры свидетельствует о сопоставимом соотношении ложных срабатываний и пропущенных сообщений. Выраженного смещения классификатора в сторону одного способа принятия решения не выявлено.

Графическое представление основных показателей качества приведено на рисунке 13.

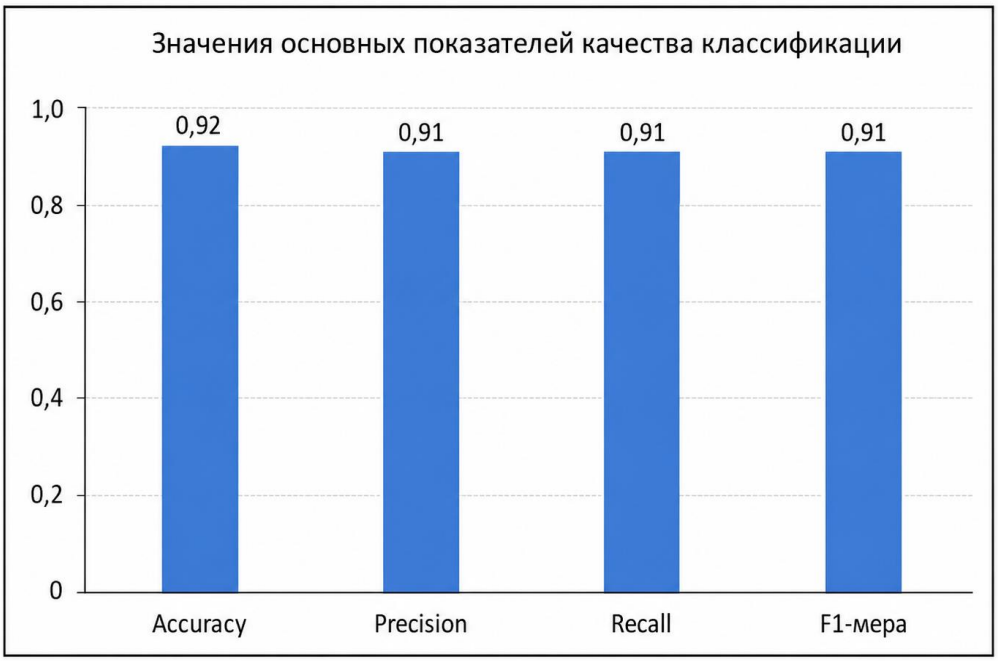

Рисунок 13 - Значения основных показателей качества классификации

Диаграмма содержит значения Accuracy, Precision, Recall и F1-меры. Единый масштаб по вертикальной оси позволяет сопоставить полученные результаты и показать незначительные различия между рассчитанными показателями.

Наиболее устойчиво определялись полезные сообщения и тексты с признаками оскорбительного содержания. Полезный контент не содержит выраженных признаков нежелательных категорий, а сообщения с оскорблениями часто включают устойчивые лексические сочетания, распознаваемые свёрточными фильтрами.

Основная часть ошибок была связана с разграничением спама и рекламы. Сообщения обеих категорий могут содержать ссылки, сведения о скидках, названия товаров и призывы к выполнению определённого действия. При этом отнесение текста к спаму нередко зависит от массовости и навязчивого характера распространения, которые невозможно установить только по содержимому одного сообщения.

Отдельные ошибки возникали при обработке коротких текстов, состоящих из нескольких слов. Небольшое количество токенов ограничивает состав признаков, доступных классификатору, и повышает влияние каждого

отдельного слова на результат. Неоднозначные сообщения могли получать близкие вероятностные оценки нескольких категорий.

Распределение правильных и ошибочных решений представлено на рисунке 14.

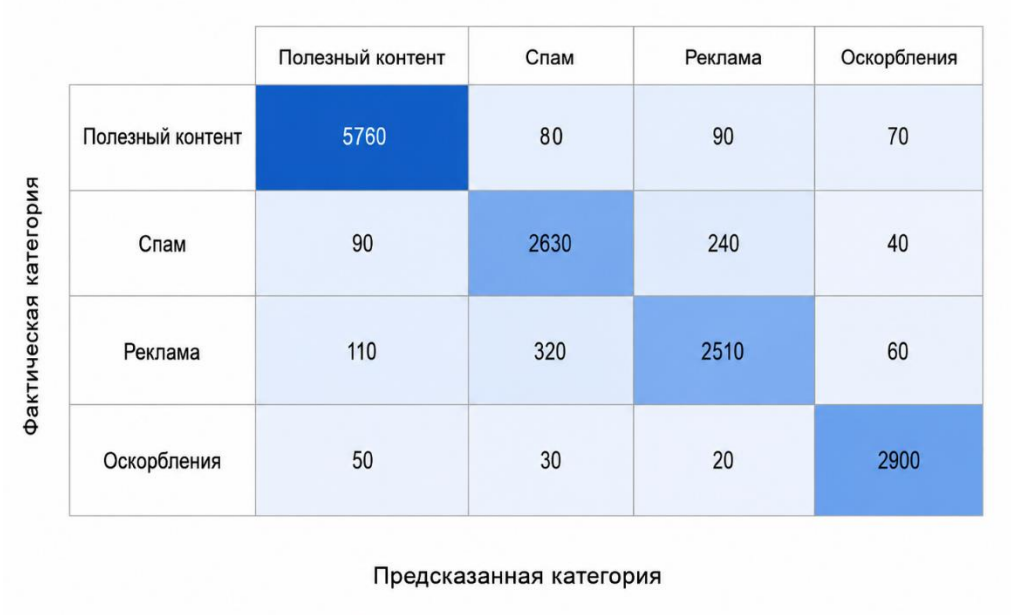

Рисунок 14 - Матрица ошибок классификации текстовых сообщений

По горизонтальной оси матрицы указаны категории, определённые моделью, по вертикальной оси - фактические категории тестовой выборки. Значения главной диагонали соответствуют правильно классифицированным сообщениям. Остальные ячейки показывают количество ошибочных отнесений между отдельными классами.

Среднее время обработки одного сообщения составило 4,1 мс. Измерение выполнялось на персональном компьютере с процессором Intel Core i5-10400F без использования графического ускорителя. Указанное время включает предварительную обработку текста и выполнение CNN-модели средствами ONNX Runtime. Задержка сетевой передачи данных при расчёте не учитывалась.

Модель загружается при запуске серверной части и остаётся в памяти в течение работы приложения. Время повторной инициализации ONNX Runtime не входит в продолжительность обработки каждого сообщения. Полученная

скорость позволяет выполнять классификацию без заметной задержки при отображении результатов в клиентском приложении.

Проведённая оценка подтвердила возможность применения CNN-модели для классификации коротких текстовых сообщений. Повышение качества может быть достигнуто за счёт расширения экспериментального корпуса, уточнения разметки категорий «спам» и «реклама», а также учёта дополнительных сведений о характере распространения сообщения.

ЗАКЛЮЧЕНИЕ

В результате выполнения магистерской работы был проведён анализ методов автоматической классификации текстовой информации. Рассмотрены словарные, статистические и нейросетевые подходы, применяемые при обработке сообщений. Для решения поставленной задачи выбрана свёрточная нейронная сеть, обеспечивающая распознавание локальных признаков короткого текста и не требующая значительных вычислительных ресурсов при использовании обученной модели.

Также рассмотрены существующие средства анализа и модерации сообщений. Анализ рассмотренных решений показал, что большинство из них предназначено для выявления отдельных видов нежелательного содержимого. В разработанной системе предусмотрено определение трех категорий нежелательного содержимого: «спам», «реклама» и «оскорбления».

Результатом работы стала система определения категории текстовой информации, выполняющая подготовку сообщения, классификацию содержимого и формирование параметров его отображения. Для обучения модели сформирован набор из 100 000 сообщений. Проверка на тестовой выборке показала Accuracy 0,92 и F1-меру 0,91. Среднее время обработки одного сообщения составило 4,1 мс.

Разработанная система может применяться в мессенджерах, корпоративных средствах коммуникации и других программных продуктах, в которых требуется автоматический анализ текстового содержимого. Полученная категория позволяет маркировать сообщения и учитывать пользовательские настройки фильтрации без автоматического ограничения их передачи.

Разработанная система обладает возможностями дальнейшего расширения без изменения основных принципов её функционирования. Дальнейшее развитие включает в себя увеличение перечня распознаваемых категорий, дополнение обучающей выборки пограничными примерами спама и рекламы, а также учёт контекста переписки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Воронцов К. В. Машинное обучение: курс лекций // MachineLearning.ru [САЙТ]. - URL: https://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_(курс_лекций,_К.В.Воронцов) (дата обращения: 10.05.2026).

2. Воробьев Е. В., Пучков Е. В. Классификация текстов с помощью сверточных нейронных сетей // КиберЛенинка [САЙТ]. - 2017. - URL: https://cyberleninka.ru/article/n/klassifikatsiya-tekstov-s-pomoschyu-svertochnyh-neyronnyh-setey (дата обращения: 10.06.2026).

3. Черкасов А. Н., Туркин Е. А. Выбор оптимальной архитектуры искусственной нейронной сети для задачи классификации текстов // КиберЛенинка [САЙТ]. - 2021. - URL: https://cyberleninka.ru/article/n/vybor-optimalnoy-arhitektury-iskusstvennoy-neyronnoy-seti-dlya-zadachi-klassifikatsii-tekstov (дата обращения: 12.04.2026).

4. Смирнова О. С. Выбор топологии нейронных сетей и их применение для классификации коротких текстов // КиберЛенинка [САЙТ]. - 2016. - URL: https://cyberleninka.ru/article/n/vybor-topologii-neyronnyh-setey-i-ih-primenenie-dlya-klassifikatsii-korotkih-tekstov (дата обращения: 11.04.2026).

5. Sebastiani F. Machine Learning in Automated Text Categorization // ACM Computing Surveys. - 2002. - Vol. 34, № 1. - P. 1-47. - URL: https://dl.acm.org/doi/10.1145/505282.505283 (дата обращения: 15.03.2026).

6. Kim Y. Convolutional Neural Networks for Sentence Classification // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. - 2014. - P. 1746-1751. - URL: https://aclanthology.org/D14-1181/ (дата обращения: 20.05.2026).

7. Minaee S., Kalchbrenner N., Cambria E., Nikzad N., Chenaghlu M., Gao J. Deep Learning Based Text Classification: A Comprehensive Review // ACM Computing Surveys. - 2021. - Vol. 54, № 3. - URL: https://arxiv.org/abs/2004.03705 (дата обращения: 7.03.2026).

8. Goodfellow I., Bengio Y., Courville A. Deep Learning. - Cambridge: MIT Press, 2016. - 800 p. - URL: https://www.deeplearningbook.org/ (дата обращения: 10.05.2026).

9. Avoid and report phishing emails // Gmail Help [САЙТ]. - URL: https://support.google.com/mail/answer/8253 (дата обращения: 21.03.2026).

10. Spam FAQ // Telegram [САЙТ]. - URL: https://telegram.org/faq_spam (дата обращения: 24.03.2026).

11. Native antispam system // Telegram API [САЙТ]. - URL: https://core.telegram.org/api/antispam (дата обращения: 25.03.2026).

12. Communication Compliance // Microsoft Learn [САЙТ]. - URL: https://learn.microsoft.com/en-us/purview/communication-compliance-solution-overview (дата обращения: 26.03.2026).

13. Perspective API Documentation // Perspective API [САЙТ]. - URL: https://developers.perspectiveapi.com/s/docs (дата обращения: 12.04.2026).

14. Keras: The high-level API for TensorFlow // TensorFlow [САЙТ]. - URL: https://www.tensorflow.org/guide/keras (дата обращения: 2.05.2026).

15. ONNX: Open Neural Network Exchange // ONNX [САЙТ]. - URL: https://onnx.ai/ (дата обращения: 19.05.2026).

16. ONNX Runtime Node.js binding // ONNX Runtime [САЙТ]. - URL: https://onnxruntime.ai/docs/get-started/with-javascript/node.html (дата обращения: 20.05.2026).

17. PostgreSQL Documentation // PostgreSQL [САЙТ]. - URL: https://www.postgresql.org/docs/ (дата обращения: 22.05.2026).

18. Metrics and scoring: quantifying the quality of predictions // Scikit-learn [САЙТ]. - URL: https://scikit-learn.org/stable/modules/model_evaluation.html (дата обращения: 24.05.2026).

19. SMS Spam Collection // UCI Machine Learning Repository [САЙТ]. - URL: https://archive.ics.uci.edu/dataset/228/sms+spam+collection (дата обращения: 22.04.2026).

20. Russian Language Toxic Comments // Kaggle [САЙТ]. - URL: https://www.kaggle.com/datasets/blackmoon/russian-language-toxic-comments (дата обращения: 22.05.2026).

ПРИЛОЖЕНИЕ А

(обязательное)

Определение категории текстовой информации с помощью машинно - обучаемых систем

Листинг программы

Файл train_model.py

```
import json
import numpy as np
import pandas as pd
import tensorflow as tf
import tf2onnx

from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix,
)
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, Model
from tensorflow.keras.preprocessing.sequence import
pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer


MAX_LENGTH = 200
VOCABULARY_SIZE = 50000
CATEGORIES = [
    "полезный контент",
    "спам",
    "реклама",
    "оскорбления",
]

category_indexes = {
    category: index
    for index, category in enumerate(CATEGORIES)
}


dataset = pd.read_csv(
    "messages_dataset.csv"
)

dataset = dataset.dropna(
    subset=["message_text", "message_category"]
)

dataset["message_text"] = (
    dataset["message_text"]
    .astype(str)
    .str.strip()
)

dataset = dataset[
    dataset["message_text"].str.len() > 0
```

```
]

dataset = dataset.drop_duplicates(
    subset=["message_text"]
)


train_data, temporary_data = train_test_split(
    dataset,
    test_size=0.30,
    stratify=dataset["message_category"],
    random_state=42,
)

validation_data, test_data = train_test_split(
    temporary_data,
    test_size=0.50,
    stratify=temporary_data["message_category"],
    random_state=42,
)


tokenizer = Tokenizer(
    num_words=VOCABULARY_SIZE,
    oov_token="<UNKNOWN>",
    lower=True,
)

tokenizer.fit_on_texts(
    train_data["message_text"]
)


def encode_texts(texts):
    sequences = tokenizer.texts_to_sequences(
        texts
    )

    return pad_sequences(
        sequences,
        maxlen=MAX_LENGTH,
        padding="post",
        truncating="post",
    )


train_x = encode_texts(
    train_data["message_text"]
)

validation_x = encode_texts(
    validation_data["message_text"]
)
```

```
test_x = encode_texts(
    test_data["message_text"]
)

train_y = train_data[
    "message_category"
].map(category_indexes).to_numpy()

validation_y = validation_data[
    "message_category"
].map(category_indexes).to_numpy()

test_y = test_data[
    "message_category"
].map(category_indexes).to_numpy()


inputs = layers.Input(
    shape=(MAX_LENGTH,),
    dtype=tf.int32,
    name="token_ids",
)

embeddings = layers.Embedding(
    VOCABULARY_SIZE,
    128,
)(inputs)

branches = []

for kernel_size in (3, 4, 5):
    branch = layers.Conv1D(
        128,
        kernel_size,
        activation="relu",
    )(embeddings)

    branch = layers.GlobalMaxPooling1D()(
        branch
    )

    branches.append(branch)

features = layers.Concatenate()(
    branches
)

features = layers.Dropout(0.5)(
    features
)

outputs = layers.Dense(
```

```
    len(CATEGORIES),
    activation="softmax",
    name="category_probabilities",
)(features)

model = Model(
    inputs,
    outputs,
)

model.compile(
    optimizer=tf.keras.optimizers.Adam(
        learning_rate=0.001
    ),
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"],
)

model.fit(
    train_x,
    train_y,
    validation_data=(
        validation_x,
        validation_y,
    ),
    batch_size=64,
    epochs=20,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor="val_loss",
            patience=3,
            restore_best_weights=True,
        )
    ],
)


probabilities = model.predict(
    test_x
)

predicted_y = np.argmax(
    probabilities,
    axis=1,
)

print(
    "Accuracy:",
    accuracy_score(test_y, predicted_y),
)

print(
    "Precision:",
```

```
    precision_score(
        test_y,
        predicted_y,
        average="macro",
    ),
)

print(
    "Recall:",
    recall_score(
        test_y,
        predicted_y,
        average="macro",
    ),
)

print(
    "F1-мера:",
    f1_score(
        test_y,
        predicted_y,
        average="macro",
    ),
)

print(
    "Матрица ошибок:\n",
    confusion_matrix(
        test_y,
        predicted_y,
    ),
)


with open(
    "artifacts/vocabulary.json",
    "w",
    encoding="utf-8",
) as file:
    json.dump(
        tokenizer.word_index,
        file,
        ensure_ascii=False,
    )

with open(
    "artifacts/metadata.json",
    "w",
    encoding="utf-8",
) as file:
    json.dump(
        {
            "max_length": MAX_LENGTH,
```

```
            "categories": CATEGORIES,
            "model_version": "1.0.0",
        },
        file,
        ensure_ascii=False,
    )


signature = (
    tf.TensorSpec(
        shape=(None, MAX_LENGTH),
        dtype=tf.int32,
        name="token_ids",
    ),
)

tf2onnx.convert.from_keras(
    model,
    input_signature=signature,
    opset=17,
    output_path=(
        "artifacts/message_cnn.onnx"
    ),
)
```

Файл classification.js

```
import fs from 'node:fs/promises';
import * as onnx from 'onnxruntime-node';


let vocabulary;
let metadata;
let modelSession;


export async function initializeClassifier() {
    vocabulary = JSON.parse(
        await fs.readFile(
            './artifacts/vocabulary.json',
            'utf-8'
        )
    );

    metadata = JSON.parse(
        await fs.readFile(
            './artifacts/metadata.json',
            'utf-8'
        )
    );

    modelSession =
        await onnx.InferenceSession.create(
            './artifacts/message_cnn.onnx',
```

```
            {
                executionProviders: ['cpu'],
            }
        );
}


function normalizeText(text) {
    return text
        .normalize('NFKC')
        .toLowerCase()
        .replace(/ё/g, 'е')
        .replace(/[\u200B-\u200D\uFEFF]/g, '')
        .replace(/\s+/g, ' ')
        .trim();
}


function encodeMessage(text) {
    const normalized = normalizeText(text);

    const tokens = normalized.match(
        /[а-яа-z0-9]+/gi
    ) ?? [];

    const vocabularySize =
        metadata.vocabulary_size ?? 50000;

    const tokenIds = tokens
        .slice(0, metadata.max_length)
        .map(token => {
            const tokenId = vocabulary[token];

            if (
                tokenId === undefined
                || tokenId >= vocabularySize
            ) {
                return 1;
            }

            return tokenId;
        });

    while (
        tokenIds.length < metadata.max_length
    ) {
        tokenIds.push(0);
    }

    return {
        normalized,
        tokenIds,
    };
```

```
}


export async function classifyMessage(text) {
    if (!modelSession) {
        throw new Error(
            'Модель классификации не загружена'
        );
    }

    const preparedMessage = encodeMessage(text);

    const inputTensor = new onnx.Tensor(
        'int32',
        Int32Array.from(
            preparedMessage.tokenIds
        ),
        [1, metadata.max_length]
    );

    const output = await modelSession.run({
        [modelSession.inputNames[0]]:
            inputTensor,
    });

    const probabilities = Array.from(
        output[
            modelSession.outputNames[0]
        ].data
    );

    const confidence = Math.max(
        ...probabilities
    );

    const predictedIndex =
        probabilities.indexOf(confidence);

    return {
        normalizedText:
            preparedMessage.normalized,
        category:
            metadata.categories[predictedIndex],
        confidence:
            Number(confidence.toFixed(4)),
        probabilities,
    };
}


export function buildClassificationResponse(
    messageId,
    classification,
```

```
    filterSettings = {}
) {
    const warningFlag =
        classification.category
        !== 'полезный контент';

    const hiddenCategories =
        filterSettings.hiddenCategories ?? [];

    let displayAction = 'show';

    if (
        hiddenCategories.includes(
            classification.category
        )
    ) {
        displayAction = 'hide';
    } else if (warningFlag) {
        displayAction = 'show_warning';
    }

    const categoryTags = {
        'полезный контент': null,
        'спам': 'Спам',
        'реклама': 'Реклама',
        'оскорбления':
            'Оскорбительное содержимое',
    };

    return {
        message_id: messageId,
        predicted_category:
            classification.category,
        confidence_score:
            classification.confidence,
        warning_flag: warningFlag,
        category_tag:
            categoryTags[
                classification.category
            ],
        display_action: displayAction,
    };
}
```

Файл storage.js

```
import pg from 'pg';

const { Pool } = pg;


const database = new Pool({
    host: process.env.DB_HOST,
    port: Number(
        process.env.DB_PORT ?? 5432
```

```
    ),
    database: process.env.DB_NAME,
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
});


export async function saveClassificationResult(
    data
) {
    const query = `
        INSERT INTO message_classifications (
            message_id,
            original_text,
            normalized_text,
            predicted_category,
            confidence_score,
            probabilities,
            warning_flag,
            category_tag,
            display_action,
            model_version
        )
        VALUES (
            $1, $2, $3, $4, $5,
            $6::jsonb, $7, $8, $9, $10
        )
    `;

    const values = [
        data.messageId,
        data.originalText,
        data.classification.normalizedText,
        data.classification.category,
        data.classification.confidence,
        JSON.stringify(
            data.classification.probabilities
        ),
        data.response.warning_flag,
        data.response.category_tag,
        data.response.display_action,
        '1.0.0',
    ];

    await database.query(
        query,
        values
    );
}
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Определение категории текстовой информации с помощью машинно - обучаемых систем

Презентация