МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Школа компьютерных наук

ОТЧЕТ

О РАЗРАБОТКЕ ПРОЕКТА ПО ДИСЦИПЛИНЕ

«Интеллектуальные системы поддержки принятия решений»

ТЕМА ПРОЕКТА

«Разработка сервиса для выявления непродуктивного общения в рабочих чатах

IT-компаний на основе анализа сообщений»

|  |  |  |
| --- | --- | --- |
| Выполнил  обучающийся 4 курса,  2021 года поступления |  | Потехин Илья Романович |
|  | (подпись) | (ФИО) |
| Выполнил  обучающийся 4 курса,  2021 года поступления |  | Кузьмин Артём Андреевич |
|  | (подпись) | (ФИО) |
| Научный руководитель  от института / школы |  | Захарова Ирина Гелиевна |
|  | (подпись) | (ФИО) |

**Оглавление**

[**Введение 3**](#_blw3xnidu9qg)

[**Глава 1. Материалы и модель классификации 6**](#_7z12hrfalkyn)

[Источники данных 6](#_wb6p8zyzinvm)

[Анализ данных 6](#_eihtcialliuh)

[Модель для классификации сообщений 10](#_ze8yird00w83)

[**Глава 2. Дообучение и оценка качества модели 12**](#_vl1d4kx0e55m)

[Критерии “непродуктивного общения” 12](#_mv2y34sdyk6x)

[Дообучение модели 13](#_3lldy45upvbw)

[Оценка качества модели 15](#_ah2wh2xbsdix)

[**Глава 3. Разработка бота 20**](#_jipeupuivxeq)

[Требования к системе 20](#_e09hectdhg7g)

[Практическая реализация 20](#_pdzj87msztoa)

[Генерация отчетов о продуктивности 28](#_cbb5v73iguj4)

[Тестирование 29](#_kw75hrg3u9y)

[Описание интерфейса работы чат-бота 32](#_1i5s4tflm4ek)

[**Заключение 34**](#_f1dakb5sp0o1)

[**Список литературы 36**](#_cplhe77sbch6)

# 

# 

# Введение

В условиях современного цифрового мира рабочие чаты стали неотъемлемой частью корпоративной культуры, предоставляя сотрудникам возможность оперативно обмениваться информацией и координировать свои действия в рамках проектов. Независимо от сферы, будь то производство, финансы или информационные технологии, команды используют рабочие чаты для совместной работы над проектами и решения повседневных задач. Однако, несмотря на очевидные преимущества, такие чаты часто становятся источником непродуктивного общения, которое негативно влияет на эффективность рабочих процессов. Проблема непродуктивного общения актуальна для различных сфер деятельности, включая информационные технологии, финансы и производство.  
  
 Под непродуктивным общением в рамках данного проекта подразумевается взаимодействие в рабочих чатах, характеризующееся отклонением от основной темы обсуждения и отвлекающее сотрудников от решения текущих задач. Например, в процессе обсуждения технического задания один из участников может начать говорить на личные темы, что отвлекает команду и замедляет выполнение задачи.

Согласно исследованию сервисов «Работа.ру» и «СберЗдоровье», до 40% рабочего времени сотрудников уходит на личные обсуждения в рабочих чатах, что приводит к замедлению выполнения задач и снижению общей продуктивности команды. Общение с коллегами составляет 44% среди основных отвлекающих факторов, замедляющих выполнение задач. Более 43% сотрудников тратят до 30 минут на посторонние вещи, а 37% — до часа ежедневно. Непродуктивное общение вынуждает 17% сотрудников задерживаться на работе, 12% — начинать день раньше, и 5% — работать в выходные дни. Такие показатели свидетельствуют о значительном негативном влиянии непродуктивного общения на рабочий процесс и личную эффективность сотрудников.

Актуальность данного исследования обусловлена растущей зависимостью современных компаний от цифровых инструментов для коммуникации и управления проектами. Эффективное использование рабочих чатов напрямую влияет на производительность труда и успешность реализации проектов. Автоматизация процесса анализа сообщений позволяет своевременно выявлять и устранять факторы, снижающие продуктивность, что способствует улучшению общей эффективности работы команды.

На сегодняшний день существуют исследования, направленные на улучшение рабочей эффективности с использованием цифровых технологий. Одним из таких направлений является автоматизация мониторинга и анализа сообщений в рабочих чатах с целью выявления отклонений от основной темы обсуждения. Встроенные функции в корпоративных мессенджерах, таких как Telegram, позволяют интегрировать интеллектуальные системы, которые могут автоматически классифицировать сообщения, отличая продуктивное общение от отвлечённых разговоров.

Целью данной работы является разработка телеграм-бота, использующего методы трансферного обучения на базе модели RuBERT для автоматической классификации сообщений в рабочих чатах как продуктивных или непродуктивных. Разработанное решение предполагает создание инструмента, способного анализировать содержание и контекст сообщений, основываясь на предварительно размеченном датасете, и предоставлять результаты классификации. Реализация этого инструмента направлена на повышение эффективности рабочих процессов, позволяя руководителям и командам выявлять отвлекающие факторы в общении, улучшать взаимодействие и принимать обоснованные управленческие решения.

Объектом исследования являются рабочие коммуникации в корпоративных чатах, а предметом исследования — автоматизированные методы анализа и классификации сообщений на продуктивные и непродуктивные с использованием моделей машинного обучения.

Задачи, поставленные для достижения цели:

* Определение содержания понятия "непродуктивного общения" и его признаков.
* Сбор и разметка датасета для дообучения модели.
* Разработка и обучение модели для классификации сообщений.
* Интеграция модели в телеграм-бота.
* Тестирование на данных.

# 

# Глава 1. Материалы и модель классификации

## Источники данных

Для дообучения модели классификации сообщений использовался датасет, состоящий из 1000 диалогов, размеченных как продуктивные и непродуктивные. Данный датасет был собран из различных источников, включая форумы, такие как Habr.com и YouTube. Часть сообщений была сгенерирована с использованием модели ChatGPT, которая использовалась для создания примеров диалогов, схожих с реальными обсуждениями в рабочих чатах. Это позволило дополнить датасет контекстно релевантными примерами для дообучения модели. В итоге датасет включает 5481 сообщение, размеченное по категориям продуктивного и непродуктивного общения.

Для тестирования системы были собраны диалоги из реальных рабочих чатов, которые предварительно были обезличены для обеспечения конфиденциальности. После этого данные были размечены в соответствии с выработанными критериями. Всего было собрано 243 сообщения.

## Анализ данных

### Баланс классов

Датасет имеет дисбаланс между классами. Из 5481 сообщения 70.46% составляют продуктивные сообщения (класс 1), а 29.54% — непродуктивные (класс 0). Это неравномерное распределение может повлиять на эффективность дообучения модели, поскольку большинство сообщений принадлежат к классу 1. В целях улучшения результатов были применены методы балансировки данных, такие как взвешенные функции потерь.

* Количество сообщений класса 0 (непродуктивные): 1619.
* Количество сообщений класса 1 (продуктивные): 3862.

### Длина сообщений

На рис. 1 показано распределение длин сообщений. Средняя длина сообщения в словах составляет 11.81, что говорит о том, что сообщения в основном краткие и лаконичные. Минимальная длина сообщения — 3 слова, максимальная — 136 слов. Медианная длина сообщения — 11 слов. Продуктивные сообщения, как правило, имеют более длинные тексты, что может указывать на более подробные и содержательные обсуждения рабочих вопросов.

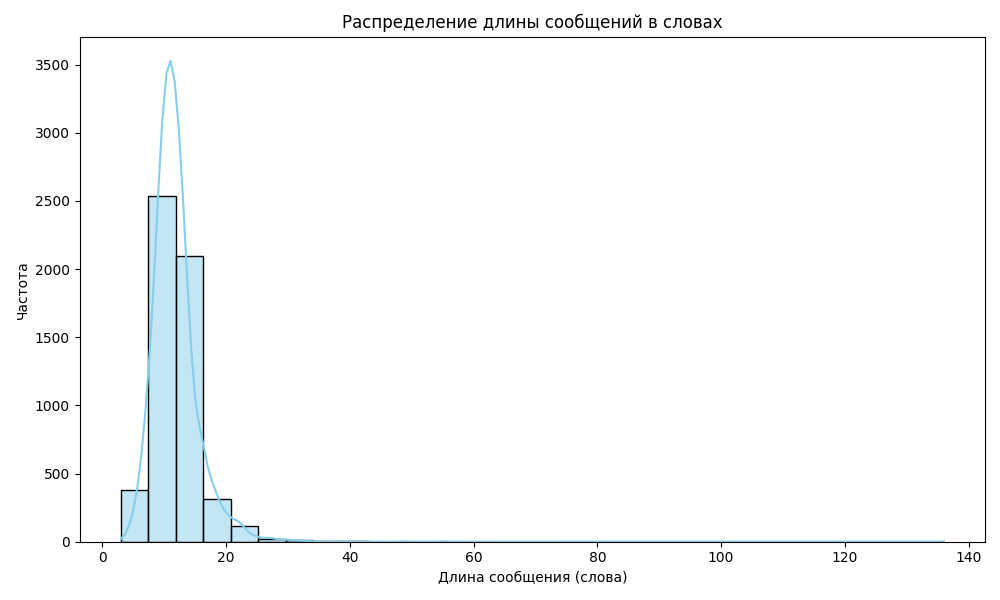


Рис. 1. Распределение длины сообщений в словах

Гистограмма показывает распределение длины сообщений по количеству слов. Большая часть сообщений имеет длину менее 20 слов, что подтверждает краткость и лаконичность сообщений в рабочих чатах.

Средняя длина непродуктивных сообщений (класс 0): 11.90 слов.

Средняя длина продуктивных сообщений (класс 1): 11.78 слов.

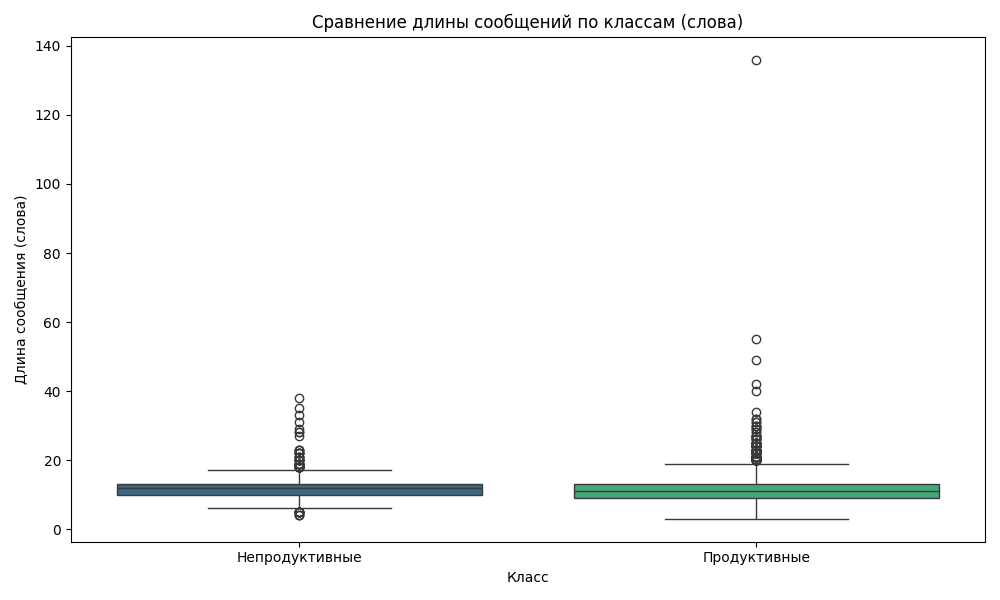


Рис. 2. Сравнение длины сообщений по классам (слова)

Рис. 2 демонстрирует различия в длине сообщений между продуктивными и непродуктивными классами. Видно, что продуктивные сообщения могут иметь значительно большую длину, что указывает на более подробные обсуждения рабочих вопросов.

### Длина сообщений в токенах

Анализ длины сообщений в токенах (рис. 3) показал, что средняя длина сообщения составляет 19.63 токена. Большинство сообщений имеют длину менее 50 токенов, что делает использование параметра max\_length=128 оптимальным для работы токенизатора. Минимальная длина сообщения в токенах — 7 токенов, максимальная — 234 токена.

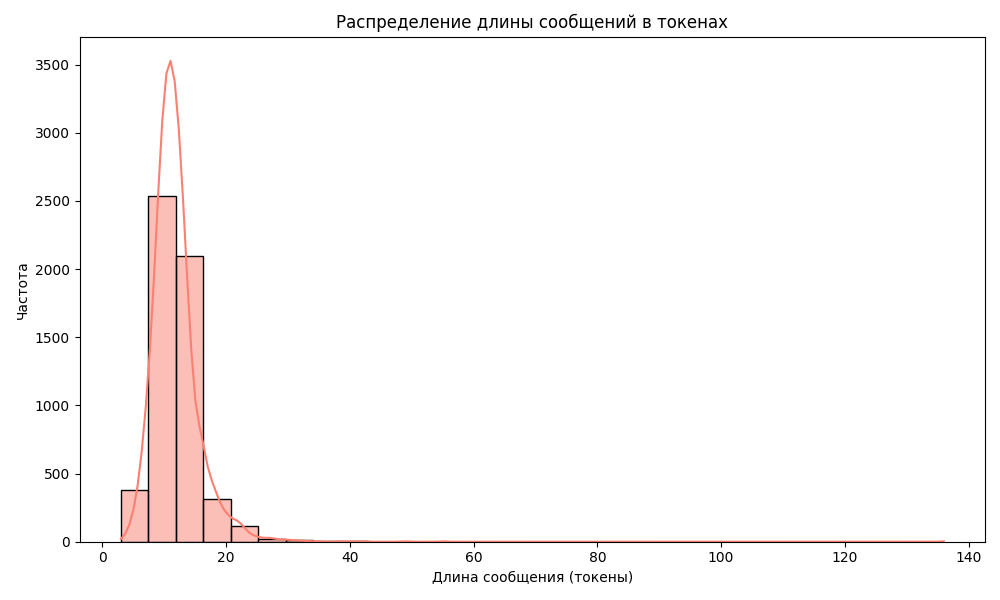


Рис. 3. Распределение длины сообщений в токенах

Гистограмма отображает распределение длины сообщений по количеству токенов. Большинство сообщений имеют длину менее 50 токенов, что подтверждает выбор параметра max\_length=128 для токенизатора.

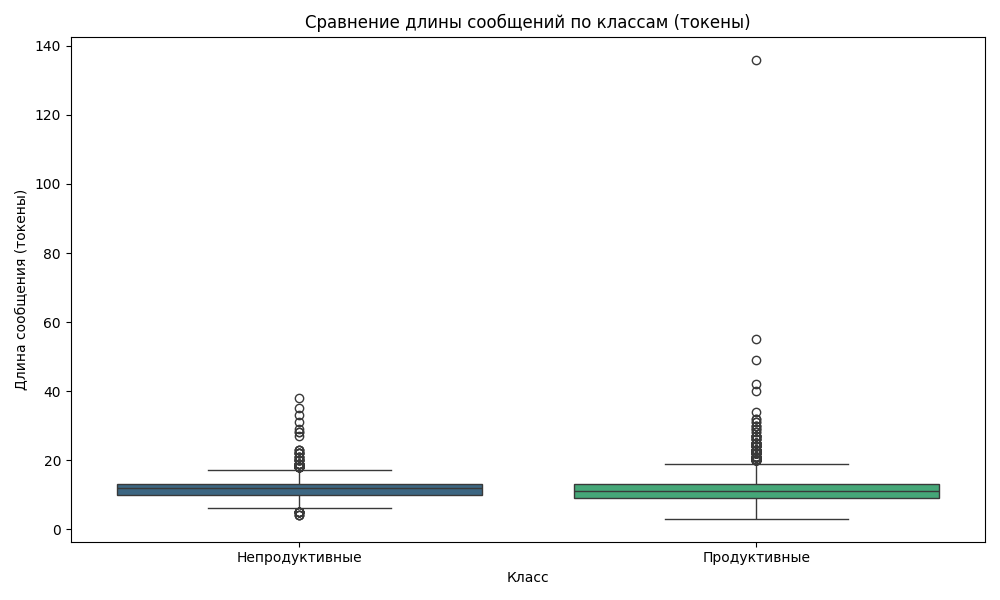


Рис. 4. Сравнение длины сообщений по классам (токены)

Рис. 4 показывает различия в длине сообщений в токенах между классами. Это позволяет увидеть, что продуктивные сообщения могут содержать значительно больше токенов, чем непродуктивные.

## Модель для классификации сообщений

Для задачи классификации сообщений на продуктивные и непродуктивные была выбрана модель RuBERT (Russian BERT), основанная на архитектуре BERT. Основная цель — разделение сообщений на два класса: продуктивные (класс 1) и непродуктивные (класс 0). Выбор RuBERT обусловлен её обучением на корпусах русскоязычных текстов, что позволяет учитывать морфологические и синтаксические особенности языка, характерные для русского текста.

Модель RuBERT имеет двунаправленную структуру, что означает её способность анализировать контекст сообщения одновременно слева направо и справа налево. Это свойство важно для задач, где требуется учитывать весь текст и понимать связи между словами в сообщении.

### Сравнение моделей

XLM-RoBERTa — мультиязычная модель, обученная на текстах на множестве языков, включая русский. Её преимущество заключается в универсальности, что делает её подходящей для задач на разных языках. Однако её узкая адаптация к особенностям русского языка ограничена, что снижает её эффективность в задачах, связанных исключительно с русскоязычными текстами.

RuGPT-3 — генеративная модель, разработанная для создания текстов. Её однонаправленная архитектура делает её менее подходящей для задач классификации, так как она анализирует текст только в одном направлении. Основное назначение этой модели — генерация текстов, а не их классификация.

RuRoBERTa — модель, аналогичная RuBERT, но с использованием обновлённых методик предобучения. На задачах обработки длинных текстов она показывает конкурентные результаты. Однако в задачах, где требуется анализ коротких текстов, таких как сообщения в рабочих чатах, RuBERT лучше учитывает специфику структуры данных.

Выбор RuBERT обоснован её специализацией на русском языке и эффективностью в задачах классификации, требующих анализа коротких текстов. Модель сочетает в себе баланс адаптации к языковым особенностям и универсальности архитектуры, что делает её оптимальным решением для анализа сообщений в рабочих чатах.

# 

# Глава 2. Дообучение и оценка качества модели

## Критерии “непродуктивного общения”

В рамках проекта была разработана система критериев для маркировки сообщений как продуктивных или непродуктивных. Основные признаки, по которым сообщения определяются как непродуктивные:

* Обсуждение отвлечённых тем
  + Сообщения, касающиеся личной жизни, хобби, отдыха, популярных развлечений, игр или сериалов, относятся к категории отвлечённых тем. Такие сообщения отвлекают внимание от основных рабочих задач и могут снизить фокус команды. Примеры включают: «На выходных ездил в горы, это было потрясающе!» или «Вчера весь вечер смотрел новый сезон сериала — невероятно захватывающе!».
* Смешение рабочих и отвлечённых тем
  + Сообщения, в которых продуктивная информация сочетается с отвлекающими элементами, также считаются непродуктивными. Это может затруднить восприятие основной информации и снизить эффективность коммуникации. Примером служат сообщения типа: «Запустил тесты для новой версии, а потом весь вечер играл с котом».
* Эмоциональные высказывания, не относящиеся к задачам
  + Сообщения, в которых выражаются личные переживания или эмоции без прямой связи с рабочими процессами, классифицируются как непродуктивные. Такие высказывания могут создавать напряжённую атмосферу или отвлекать от выполнения задач. Примером может служить сообщение: «Сегодня очень устал, ничего не могу сделать».
* Отсутствие конструктивности в рабочем контексте
  + Сообщения, содержащие общие или неопределённые фразы, которые не предлагают конкретных решений или не способствуют продвижению обсуждаемой задачи, также относятся к непродуктивным. Такие сообщения могут замедлить процесс принятия решений и выполнения задач. Примером является сообщение: «Ну, это как-то сложно, не знаю, что делать».

Разработка данных критериев позволила систематизировать подход к маркировке сообщений при разделении сообщений на продуктивные и непродуктивные. Наша гипотеза заключается в том, что применение этих критериев способствует улучшению рабочих коммуникаций и повышению общей эффективности команды.

## Дообучение модели

Основным этапом было дообучить модель для задачи бинарной классификации сообщений в рабочих чатах, направленная на разделение сообщений на продуктивные и непродуктивные. Модель RuBERT (Russian BERT), показала хорошие результаты при работе с русскоязычным текстом. Для классификации использовалась библиотека Hugging Face Transformers, а также инструменты из scikit-learn и PyTorch.

### Обработка данных

Для подготовки данных использовался предобработанный датасет из файлов формата JSON, содержащий размеченные сообщения. Каждое сообщение в датасете было предварительно очищено от лишних данных, таких как URL и email-адреса. Кроме того, для каждого сообщения была собрана информация о контексте, включая предыдущие и последующие сообщения из диалога. Эти данные стали входными для модели, что позволяет улучшить её понимание контекста и улучшить точность классификации.

### Архитектура модели

Была использована предобученная модель RuBERT, которую адаптировали для задачи классификации сообщений на продуктивные и непродуктивные. Для этого был добавлен к модели верхний классификационный слой с помощью AutoModelForSequenceClassification, чтобы она могла классифицировать сообщения на два класса. Затем была дообучена модель на нашем наборе данных, что позволило ей лучше понимать специфику сообщений из рабочих чатов. Чтобы учесть дисбаланс между продуктивными и непродуктивными сообщениями, в функции потерь (CrossEntropyLoss) были настроены весовые коэффициенты: больший вес был присвоен менее представленному классу (непродуктивным сообщениям), а меньший — более частому классу (продуктивным сообщениям). Это позволило модели уделять больше внимания обучению на менее представленном классе, обеспечивая равномерное качество классификации для обеих категорий.

### Процесс дообучения

Для дообучения модели был использован TrainingArguments из библиотеки Hugging Face Transformers. Модель обучалась в течение 10 эпох с использованием пакетов данных размером 8 сообщений для обучения и 16 для валидации. Для улучшения результатов была применена техника ранней остановки (EarlyStopping), которая позволяет остановить обучение при отсутствии улучшений на валидационном наборе данных.

Разделение данных на обучающий и валидационный наборы было выполнено с учётом пропорций классов (stratified split), что обеспечило более сбалансированную выборку для обучения.

## Оценка качества модели

### Оценка на валидационном наборе

По завершению дообучения была проведена оценка модели на валидационном наборе, который составил 10% от общего объема данных. Результаты показали высокую точность классификации:

Accuracy: 0.98

### Матрица ошибок

Матрица ошибок (confusion matrix) также показала хорошие результаты:

Непродуктивные — непродуктивные: 153

Продуктивные — непродуктивные: 1

Непродуктивные — продуктивные: 9

Продуктивные — продуктивные: 386

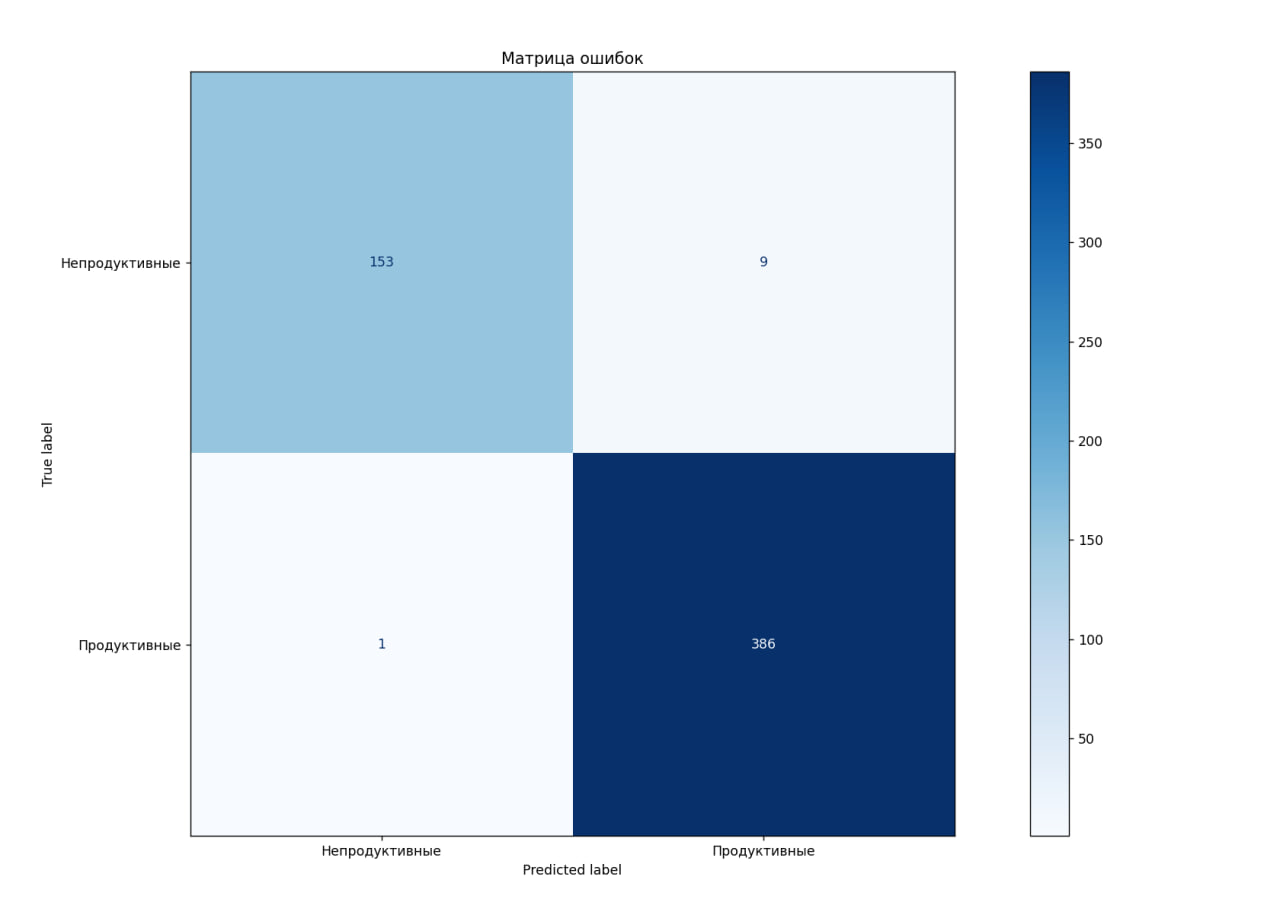


Рис. 5. Матрица ошибок на валидационной выборке

Визуализация матрицы ошибок на рис. 5 позволяет наглядно увидеть, что модель хорошо справляется с классификацией продуктивных сообщений, но иногда ошибается в классификации непродуктивных сообщений, что отражается в небольшом количестве ложных положительных (9 случаев).

### Оценка на тестовом наборе

**Всего:** 243 сообщений

**Accuracy:** 0.89

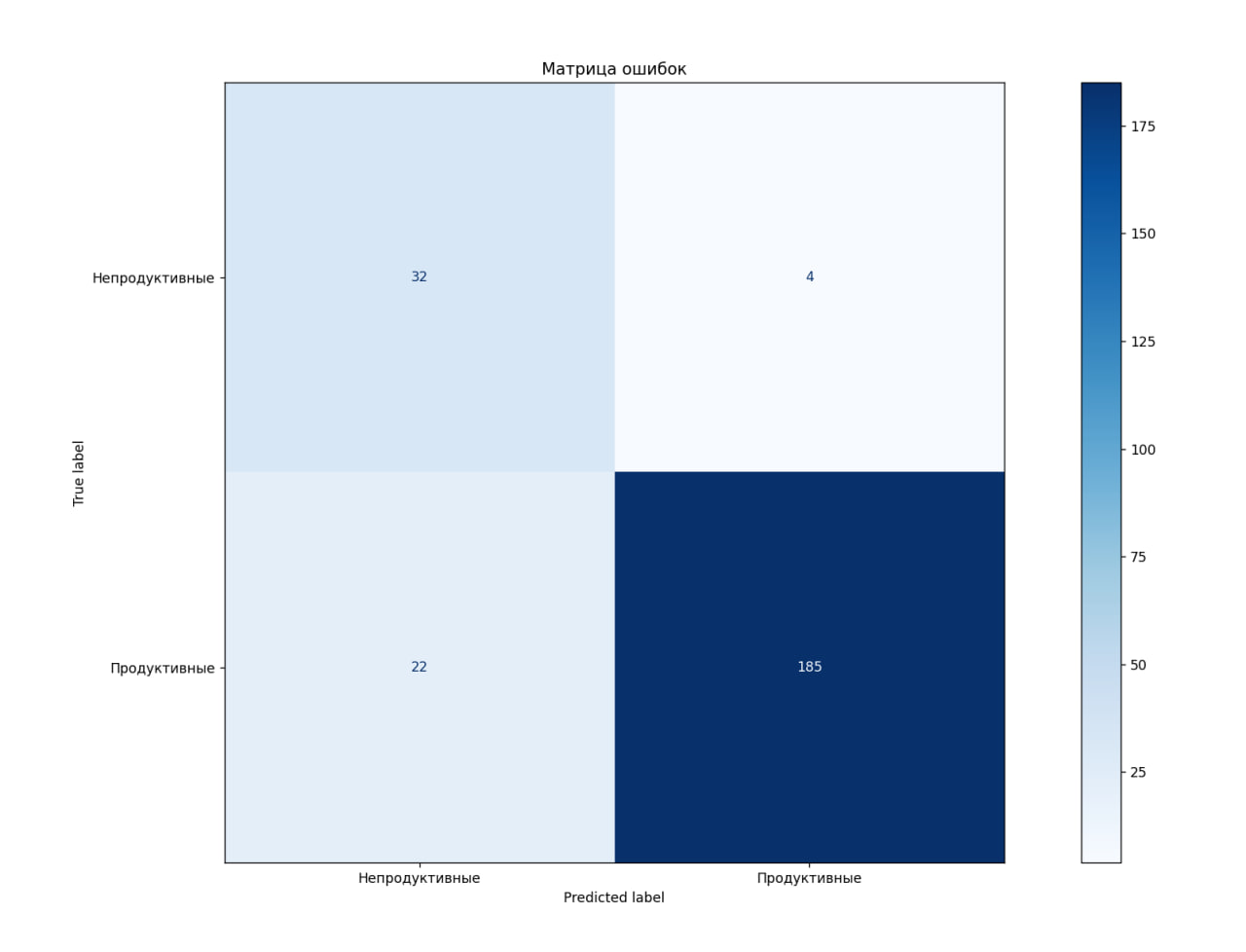
****

Рис. 6. Матрица ошибок на тестовой выборке

Пример диалога:

{

"id": 37,

"messages": [

{"message": "Кто готов взяться за обновление дизайна интерфейса?", "label": 1},

{"message": "Я могу, но нужно уточнить, что именно должно измениться.", "label": 1},

{"message": "Обновить все кнопки и добавить тёмную тему. Это главные задачи.", "label": 1},

{"message": "Иногда думаю, что нам бы подошла карьера дизайнеров. Работа программистов слишком скучна.", "label": 0},

{"message": "Дизайнеры тоже не скучают. Всё, приступаем к обновлению.", "label": 1},

{"message": "Понял. К утру всё будет готово.", "label": 1}

]

},

{

"id": 29,

"messages": [

{"message": "Нужно обсудить реализацию новой фичи. Вопросы по алгоритму есть у кого-то?", "label": 1},

{"message": "Да, есть. Как будет учитываться кэширование данных?", "label": 1},

{"message": "Мы можем использовать существующую библиотеку для кэширования. Это должно ускорить процесс.", "label": 1},

{"message": "Слушайте, я так устал от этих обсуждений. Хочу уже просто выходные.", "label": 0},

{"message": "Все устали, но мы почти у цели. Давайте соберёмся.", "label": 1},

{"message": "Ладно, тогда уточню ещё про интеграцию кэша с существующим API.", "label": 1}

]

},

### График потерь

В ходе дообучения были зафиксированы значения Loss на каждой эпохе. График, построенный по этим данным (рис. 7), продемонстрировал стабильное снижение потерь на протяжении всех эпох, что подтверждает успешную сходимость модели

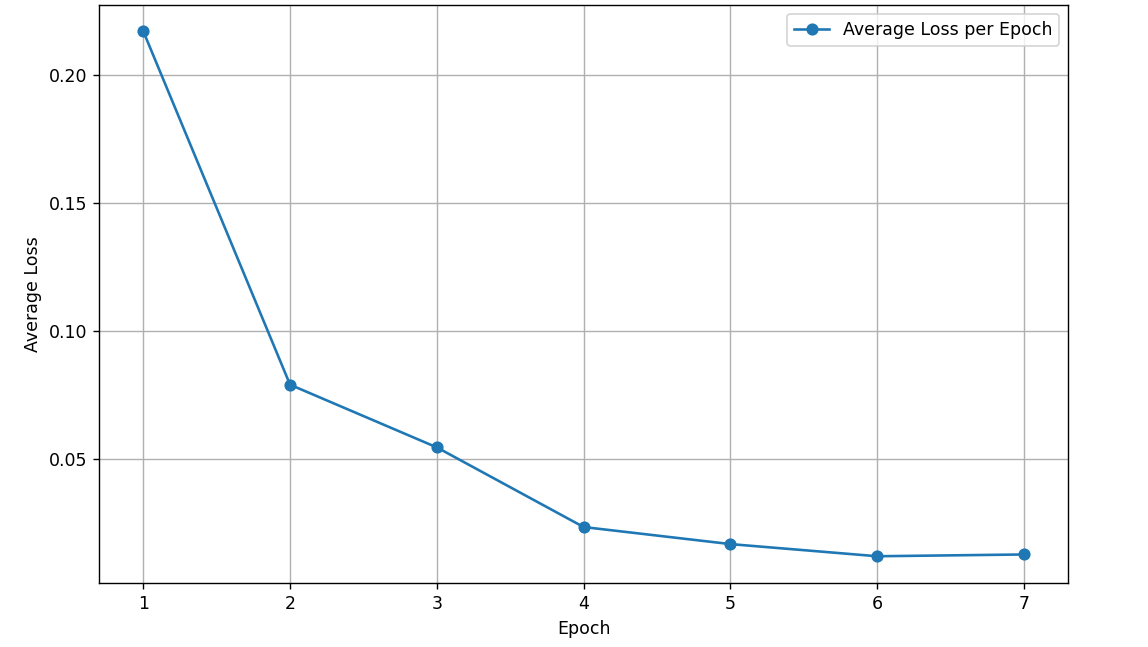


Рис. 7. График потерь при дообучении модели

# 

# Глава 3. Разработка бота

## Требования к системе

Цель системы заключается в автоматической классификации сообщений в рабочих чатах на продуктивные и непродуктивные. Для реализации поставленной задачи система должна интегрироваться с Telegram, обеспечивая взаимодействие с пользователями в режиме реального времени.

Система принимает сообщения из Telegram-чата, анализирует их с учетом контекста и классифицирует на два класса. На основе этой классификации система автоматически формирует отчеты о продуктивности общения, предоставляя пользователям возможность оценивать результаты и вносить обратную связь. Все обработанные сообщения сохраняются в базе данных для последующего анализа и улучшения работы модели.

Для обеспечения стабильной работы система должна обрабатывать большие объемы данных, предоставлять быстрый отклик API и быть масштабируемой для адаптации к изменяющимся нагрузкам. Помимо этого, важными аспектами являются устойчивость к сбоям, а также логирование и мониторинг работы для отслеживания состояния системы.

## Практическая реализация

### Общая структура системы

Система состоит из двух основных уровней: Telegram и Server. Каждый из уровней выполняет определённые функции для обеспечения классификации сообщений и взаимодействия с пользователем. Общая архитектура системы показана на рис. 8.

**Уровень Telegram:**

1. Интерфейс Telegram: Пользователь взаимодействует с Telegram-ботом, отправляя сообщения в интерфейсе мессенджера.
2. Telegram Bot API: API обеспечивает передачу сообщений от пользователя на сервер и получение ответа от системы. Этот компонент связывает Telegram с серверной частью системы.

**Уровень Server:**

1. Приложение для работы с Telegram API: Основной компонент, который обрабатывает запросы от Telegram Bot API, взаимодействует с базой данных и вызывает сервис классификации сообщений.
2. Сервис классификации сообщений: Включает модель машинного обучения на основе RuBERT, которая анализирует текст сообщения и определяет его продуктивность или непродуктивность. Модель классификации обрабатывает данные с учётом контекста, предоставляемого приложением.
3. База данных (PostgreSQL): Используется для хранения истории сообщений, меток классификации, а также для генерации аналитических отчётов. База данных взаимодействует с приложением, обеспечивая сохранность и доступ к данным.

**Взаимодействие компонентов:**

1. Пользователь отправляет сообщение через Telegram.
2. Telegram Bot API передаёт сообщение приложению на сервер.
3. Приложение взаимодействует с сервисом классификации, где сообщение анализируется моделью RuBERT.
4. Результаты классификации сохраняются в базе данных и отправляются обратно пользователю через Telegram Bot API.

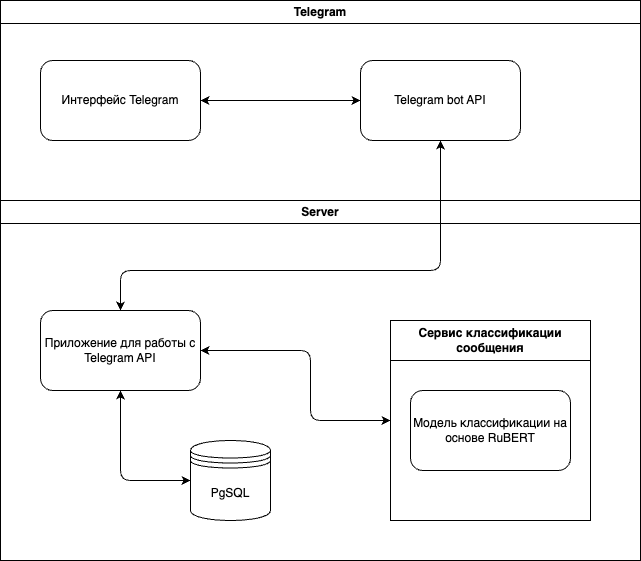
****

Рис. 8. Архитектура системы

**Сервис классификации сообщений**

Для обеспечения взаимодействия телеграм-бота с дообученной моделью классификации сообщений была разработана серверная часть на основе FastAPI. Данный компонент системы отвечает за прием запросов от бота, обработку сообщений, их классификацию и возврат результатов обратно в бот.

Основной задачей разработанного API является автоматизация процесса классификации сообщений, поступающих из рабочих чатов, с целью определения их продуктивности. Это достигается посредством следующих ключевых функций:

1. Инициализация приложения FastAPI:
   * API обеспечивает стабильную работу за счет обработки запросов через REST эндпоинты.
   * Эндпоинт /classify принимает данные в формате JSON, производит предобработку сообщений, выполняет классификацию и возвращает результат в том же формате.
2. Использование модели RuBERT с токенизатором DeepPavlov/rubert-base-cased:
   * Модель: Загружается предварительно обученная модель RuBERT, адаптированная для задач бинарной классификации сообщений. Модель загружается из сохраненного контрольного состояния, что позволяет использовать уже обученные веса для предсказаний.
   * Токенизатор: Используется токенизатор DeepPavlov/rubert-base-cased, который преобразует текстовые данные в формат, пригодный для обработки моделью. Токенизатор также загружается из предопределенного источника, обеспечивая согласованность с моделью.
   * После загрузки модель переводится в режим оценки (eval()), что отключает режим обучения и позволяет использовать модель исключительно для предсказаний.
3. Предобработка данных с учетом контекста:
   * Сбор контекста: Для каждого сообщения формируется контекст, включающий два предыдущих и два последующих сообщения из диалога. Это позволяет модели учитывать окружение сообщения, что повышает точность классификации.
   * Токенизация и предсказание: Собранный контекст токенизируется и передается в модель для предсказания. Результаты предсказания (метки классов) сохраняются вместе с оригинальными сообщениями.
   * Такой подход обеспечивает более глубокое понимание содержания и контекста сообщений, что важно для точной классификации.
4. Классификация и возврат результатов:
   * После обработки сообщения классифицируются и каждому из них присваивается метка: 1 (продуктивное) или 0 (непродуктивное). Эти метки возвращаются в формате JSON вместе с оригинальными сообщениями.

### Сервис для работы с Telegram bot API

Сервис написан на языке Go с использованием библиотеки go-telegram-bot-api. Он отвечает за взаимодействие с Telegram, обработку сообщений и их передачу в систему классификации. В сервисе реализован функционал получения обновлений, обработки сообщений, выполнения команд и передачи данных в базу.

Для выполнения периодических задач, таких как обработка сообщений и отправка их на классификацию, используется библиотека gocron. Планировщик настраивается через параметр CronSchedule, задаваемый в конфигурации. Основные этапы работы планировщика включают:

* Инициализация: Планировщик создается вызовом gocron.NewScheduler с указанием локальной временной зоны.
* Добавление задачи: Метод scheduler.Do назначает задачу для выполнения в соответствии с расписанием. Если параметр конфигурации RunImmediate установлен в true, задача запускается немедленно через scheduler.StartImmediately().Do.
* Запуск планировщика: Планировщик запускается в асинхронном режиме с помощью метода StartAsync, что позволяет ему выполнять задачи в фоновом режиме.

Для обработки входящих сообщений от пользователей используется отдельная горутина, которая запускает метод ProcessUpdatesFromBot сервиса WardenBotService. Этот метод прослушивает обновления через Telegram Bot API и обрабатывает команды и сообщения. Для этого используется метод GetUpdatesChan, который отслеживает все новые сообщения и команды. Если сообщение поступает из группового чата, оно сохраняется в базу данных. Сохраняется текст сообщения, дата отправки, имя пользователя, идентификатор чата и другие данные.

Если сообщение приходит в личный чат, бот проверяет, является ли это командой. Поддерживаются команды:

• /report [YYYY-MM-DD] — формирует запрос на создание отчета. Если дата не указана, используется предыдущий день.

• /help — показывает список доступных команд и краткую справку по их использованию.

Для обработки команды /report бот запрашивает у пользователя выбор чата, если пользователь является администратором нескольких чатов. Чаты выводятся списком кнопок, каждая из которых содержит название чата и его идентификатор. После выбора чата бот отправляет запрос на создание отчета, который затем отправляется пользователю.

Сообщения из групповых чатов ежедневно передаются на классификацию. Собираются сообщения за последний день, формируется из них список и этот список отправляется для классификации в сервис классификации через HTTP-запрос. Ответ от классификатора возвращается в виде меток для каждого сообщения. Затем эти метки обновляются в базе данных.

### Работа с базой данных

Сервис использует PostgreSQL для хранения данных, связанных с сообщениями и чатами. База данных служит для сохранения истории общения, управления данными о чатах и пользователей, а также для дальнейшего анализа и генерации отчетов. Структура данных представлена на рис. 9.

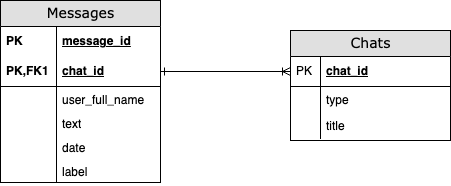


Рис. 9. Структура таблиц в базе данных

### Пример работы API для классификации сообщений

При отправке POST-запроса на эндпоинт /classify с телом запроса в формате JSON, содержащим список сообщений, API выполняет следующие действия:

1. Прием и валидация запроса: API принимает запрос и проверяет соответствие структуры данных модели MessagesRequest.

{

"messages": [

{"message": "Коллеги, для оптимизации маршрутов доставки газа нужны новые подходы. Что предложите?"},

{"message": "Используем трекинг через GPS и анализ данных в реальном времени. Это даст гибкость в логистике."},

{"message": "Можно подключить бигдату для прогнозирования пиковых нагрузок. Всё это легко крутится на джавочке."},

{"message": "Как думаете, стоит ли приглашать коллег на кофе? Иногда так хочется пообщаться неформально."},

{"message": "Конечно, почему бы и нет? Главное, чтобы это не влияло на работу."},

{"message": "Ещё стоит добавить аналитические отчёты в графану для визуализации маршрутов."}

]

}

2. Классификация сообщений: Для каждого сообщения из списка формируется контекст, выполняется его предобработка и классификация с использованием модели RuBERT.

3. Возврат результата: После обработки всех сообщений API возвращает список сообщений с присвоенными метками классов (0 —

непродуктивное, 1 — продуктивное).

{

"messages": [

{"message": "Коллеги, для оптимизации маршрутов доставки газа нужны новые подходы. Что предложите?", "label": 1},

{"message": "Используем трекинг через GPS и анализ данных в реальном времени. Это даст гибкость в логистике.", "label": 1},

{"message": "Можно подключить бигдату для прогнозирования пиковых нагрузок. Всё это легко крутится на джавочке.", "label": 1},

{"message": "Как думаете, стоит ли приглашать коллег на кофе? Иногда так хочется пообщаться неформально.", "label": 0},

{"message": "Конечно, почему бы и нет? Главное, чтобы это не влияло на работу.", "label": 0},

{"message": "Ещё стоит добавить аналитические отчёты в графану для визуализации маршрутов.", "label": 1}

]

}

## Генерация отчетов о продуктивности

Отдельной функцией разработанного чат-бота является возможность формирования отчётов для пользователей. После классификации сообщений пользователь может запросить персональный отчёт, который будет отправлен в его личные сообщения. Отчёт содержит данные о продуктивности общения в чате, выявленные непродуктивные сообщения. Такая функциональность позволяет участникам команды и руководителям оперативно анализировать взаимодействия и принимать меры для повышения продуктивности.

Отчёт содержит структурированную информацию, которая разделена на несколько ключевых блоков:

1. **Общая статистика**:  
   В этом разделе представлено общее количество сообщений, их деление на продуктивные и непродуктивные, а также процент непродуктивных сообщений. Это позволяет быстро оценить уровень продуктивности общения в чате.
2. **Примеры непродуктивных сообщений**:  
   Здесь приведены конкретные примеры сообщений, которые были классифицированы как непродуктивные. Это помогает участникам команды понять, какие темы или фразы отвлекают от основной работы, и скорректировать своё поведение.
3. **Пользователи с наиболее высоким количеством непродуктивных сообщений**:  
   Указывается список пользователей, которые отправили наибольшее количество непродуктивных сообщений. Этот раздел особенно полезен для руководителей, чтобы обратить внимание на индивидуальную продуктивность членов команды.
4. **Активность непродуктивных сообщений по времени**:  
   График или список, показывающий временные промежутки, в которые чаще всего появлялись непродуктивные сообщения. Это может помочь в выявлении периодов снижения продуктивности и планировании улучшений.
5. **Индикатор продуктивности**:  
   Итоговая оценка уровня продуктивности чата, которая позволяет быстро понять общее состояние коммуникаций.

## Тестирование

Тестирование являлось ключевым этапом разработки сервиса и было направлено на проверку его функциональности, стабильности и соответствия поставленным требованиям. Основное внимание было уделено функциональному, нагрузочному и интеграционному тестированию, что позволило убедиться в корректной работе системы на всех этапах её использования.

На первом этапе было проведено функциональное тестирование, чтобы проверить соответствие всех реализованных функций заявленным требованиям. Генерация отчетов была протестирована для различных временных периодов: текущий день, произвольная дата и диапазоны дат. Особое внимание уделялось корректности подсчета статистических данных, таких как общее количество сообщений, доля продуктивных и непродуктивных сообщений, а также процентное соотношение этих категорий. Дополнительно проверялась правильность формирования списка непродуктивных сообщений, идентификации их авторов и отображения временной активности в графическом виде. В ходе тестирования было установлено, что сервис корректно обрабатывает данные и предоставляет пользователям отчеты, соответствующие заявленным метрикам.

Вторым этапом стало интеграционное тестирование. Оно позволило убедиться в корректной работе всех компонентов системы. Проверялась работа взаимодействия Telegram-бота с серверной частью, включая передачу данных для классификации сообщений, их сохранение в базу данных и последующую генерацию отчетов. Были проведены тесты на обработку ошибок, таких как некорректный формат входящих данных или недоступность серверного API. В этих случаях система корректно возвращала пользователям сообщения об ошибках, что свидетельствует о высокой надежности разработанного решения.

Кроме того, особое внимание было уделено юзабилити-тестированию, чтобы убедиться в удобстве использования Telegram-бота. Пользователи проверяли функциональность команд, таких как /report и /help, а также генерацию отчетов, пример которого представлен на рис. 10, с указанием конкретных дат. По результатам тестирования были внесены небольшие изменения в интерфейс, такие как уточнение текстов подсказок и добавление форматирования в отчеты для лучшего восприятия данных.

Чат-бот был протестирован на данных имитирующие реальное рабочее взаимодействие, что позволило оценить его функциональность и практическую пользу.

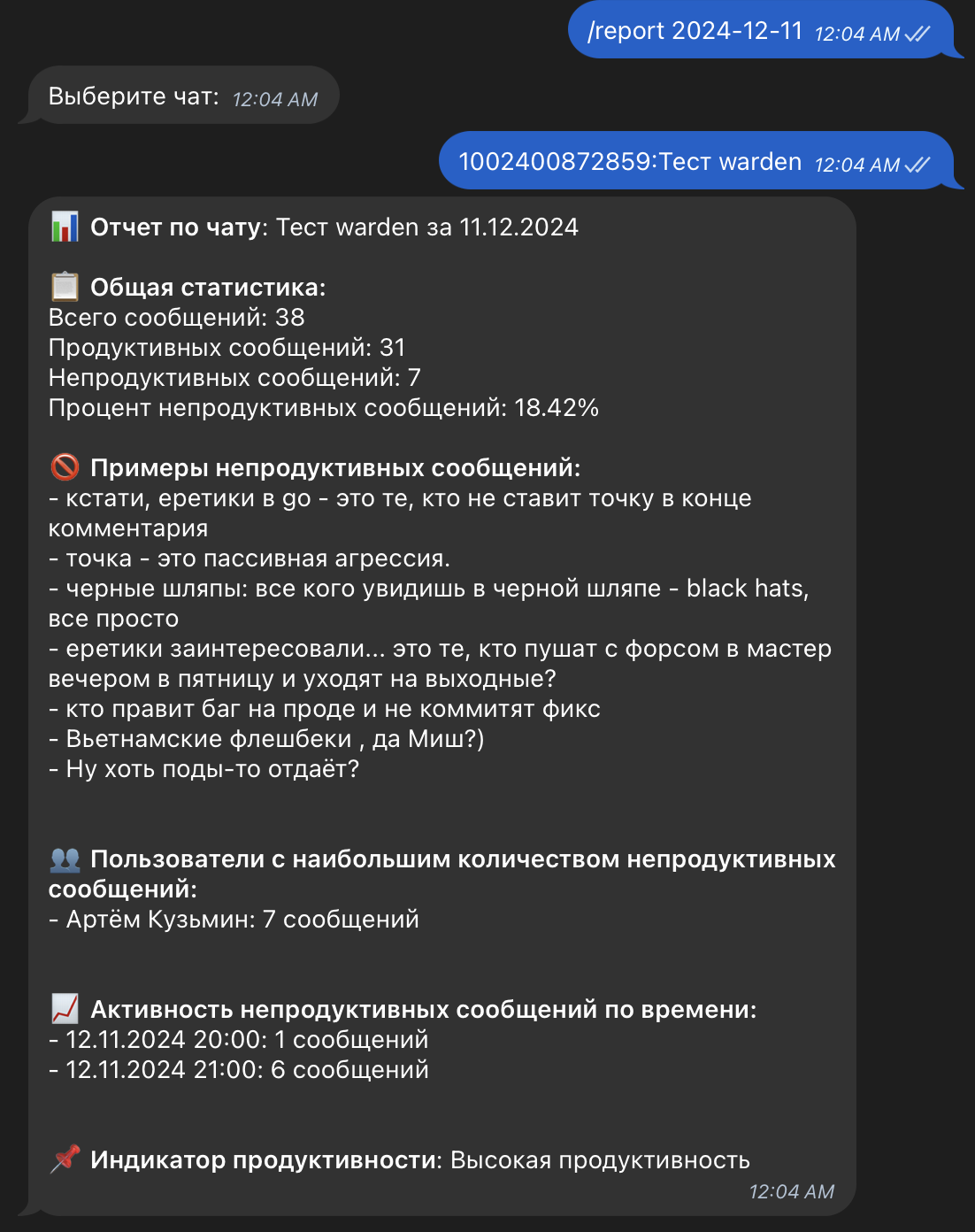


Рис. 10. Пример отчета, сформированного ботом

В результате проведенного тестирования прототип системы был признан соответствующим поставленным требованиям на этапе разработки. Все ключевые функции работают стабильно в рамках ограниченного набора сценариев, производительность системы на уровне прототипа отвечает заданным ожиданиям, а интерфейс обеспечивает базовый уровень удобства взаимодействия с пользователями. Для окончательного внедрения решения потребуется дальнейшая доработка, включающая масштабирование, расширенное тестирование и оптимизацию.

## Описание интерфейса работы чат-бота

Интерфейс работы чат-бота построен таким образом, чтобы пользователи могли взаимодействовать с системой через понятные команды и получать необходимые данные в удобной форме. На рис. 11 представлены два ключевых процесса работы чат-бота: обработка сообщений и генерация отчетов о продуктивности.

### 1. Обработка сообщений

* После подключения к заданным чатам бот начинает отслеживать все входящие сообщения через Telegram API.
* При появлении нового сообщения бот сохраняет его текст, дату, автора и идентификатор чата в базу данных.
* Бот автоматически классифицирует каждое сообщение как продуктивное или непродуктивное с использованием модели RuBERT.
* После классификации результат сохраняется в базе данных, а пользователь может запросить сводку через команду /report.

### 2. Генерация отчетов

* Пользователь отправляет команду /report и, при необходимости, выбирает нужный чат из списка кнопок.
* Бот формирует отчет
* Отчет отправляется пользователю в текстовом формате через Telegram.

### Пример взаимодействия

1. Пользователь вводит команду /report.
2. Если пользователь является администратором нескольких чатов, бот предлагает выбрать чат.
3. После выбора чата бот формирует и отправляет отчет, который отображается в формате текста с необходимой статистикой.

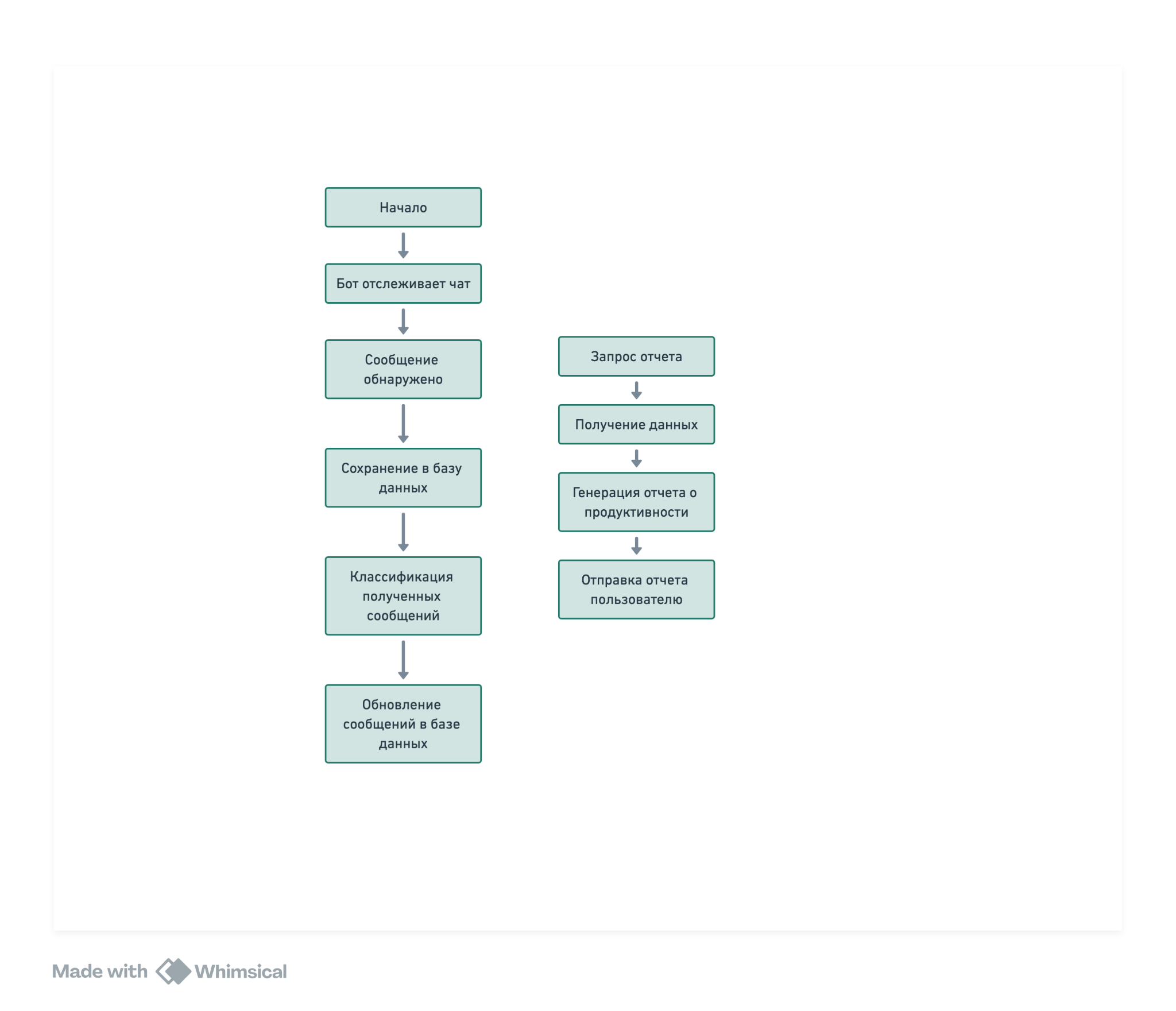


Рис. 11. Процессы взаимодействия с сервисом

# Заключение

В ходе реализации проекта был успешно разработан телеграм-бот для выявления непродуктивного общения в рабочих чатах IT-компаний на основе анализа сообщений.

Основным этапом проекта стало дообучение модели RuBERT для задачи бинарной классификации, которая смогла продемонстрировать высокую точность в разделении сообщений. Это подтверждает эффективность выбранного подхода и моделей машинного обучения в решении поставленной задачи.

Реализация API на базе FastAPI обеспечила интеграцию модели с телеграм-ботом, позволяя автоматизировать процесс классификации сообщений в режиме реального времени. Разработанные критерии классификации непродуктивного общения, такие как обсуждение отвлечённых тем, смешение рабочих и личных тем, эмоциональные высказывания и отсутствие конструктивности, подтвердили свою эффективность, поскольку модель классифицировала сообщения в соответствии с изначальной разметкой, основанной на этих критериях.

Тестирование бота подтвердило его стабильность и способность эффективно обрабатывать большие объёмы сообщений, обеспечивая своевременную маркировку и сохранение данных в базу.

В будущем планируется расширение функционала бота, включая добавление дополнительных критериев классификации. Также предусматривается направление проекта в область конфликтологии, что позволит более глубоко анализировать и разрешать потенциальные проблемы в коммуникациях внутри команд.

Разработанный проект представляет собой инструмент для мониторинга и анализа рабочих коммуникаций, который поможет руководителям и менеджерам оценивать содержание чатов сотрудников, выявлять непродуктивное общение и принимать более обоснованные управленческие решения.

# 

# Список литературы

1. Orhan M. A. et al. Technology distraction at work. Impacts on self-regulation and work engagement //Journal of Business Research. – 2021. – Т. 126. – С. 341-349. URL: https://www.sciencedirect.com/science/article/abs/pii/S0148296320308778 (дата обращения: 03.11.2024).
2. Darden D. C. Time wasting activities within the workplace (Don't be apart of them) //Available at SSRN 2623378. – 2015. URL:<https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2623378> (дата обращения: 07.10.2024).
3. SberDevices. ruT5, ruRoBERTa, ruBERT: как мы обучили серию моделей для русского языка. Habr, 2021. URL: <https://habr.com/ru/companies/sberdevices/articles/567776/>

(дата обращения: 07.10.2024).

1. Gazeta.ru. (2022). Россияне рассказали, на что чаще всего отвлекаются на работе. Газета.Ru. URL: <https://www.gazeta.ru/social/news/2022/06/09/17898182.shtml?updated>

(дата обращения: 07.11.2024).

1. Джурафски, Д., Мартин, Дж. (2009). Speech and Language Processing. 2-е изд. Прентис-Холл, 1024 с.
2. Telegram. Telegram Bot API. URL: <https://core.telegram.org/bots/api> (дата обращения: 05.11.2024)