

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего
образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»
Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
на тему
**ЧАТ-БОТ ДЛЯ РАСПОЗНАВАНИЯ
НАСТРОЕНИЯ ПОЛЬЗОВАТЕЛЯ И
ОКАЗАНИЯ ЭМОЦИОНАЛЬНОЙ
ПОДДЕРЖКИ**

Студент
(подпись, дата)

(ФИО полностью)

Репин Илья Андреевич

Руководитель
(подпись, дата)

(фамилия, инициалы)

Коннов Н.Н.

Нормоконтролёр
(подпись, дата)

(фамилия, инициалы)

Бычков А.С.

Рецензент кат. 03.02.01
(должность, место работы) (подпись, дата)

15.06.2024
(фамилия, инициалы)

Семенов А.В.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.24 № 11)

Заведующий кафедрой
(подпись)

(фамилия, инициалы)

М.А. Митрохин

Работа защищена с отметкой отлично (протокол заседания ГЭК от 14.06.2024 № 24)

Секретарь ГЭК

(подпись)

Е.И. Гурин
(фамилия, инициалы)

Пенза 2024

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 13 » сентября 2022 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Репин Илья Андреевич гр. 22ВВим1 факультета вычислительной техники
2. Тема работы «Чат-бот для распознавания настроения пользователя и оказания эмоциональной поддержки»

Тема утверждена приказом ПГУ №1338/о от "29" декабря 2022 г.

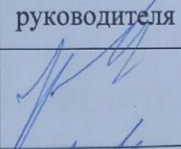
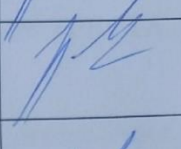
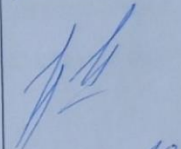
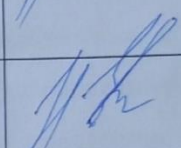
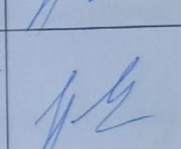
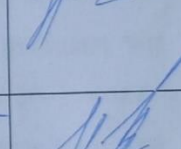
3. Руководитель магистерской диссертации д.т.н., профессор Коннов Н.Н.
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки: создание модуля чат-бота для распознавания настроения пользователя и оказания эмоциональной поддержки.

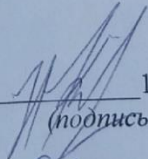
Разработать автоматизированное клиент-серверное программное обеспечение с использованием искусственного интеллекта для распознавания эмоций пользователя по его сообщениям и оказания эмоциональной поддержки.

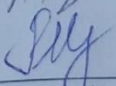
5. Перечень подлежащих разработке вопросов:
 1. разработка авторизации в веб приложении,
 2. разработка чата для взаимодействия с пользователем,
 3. обучение модели нейронной сети для распознавания настроения,
 4. использование обученной нейронной модели для определения состояния пользователя по его сообщениям.

6. Календарный график выполнения работы

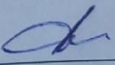
№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Проанализировать предметную область	10	21.09.22 - 21.11.22	
2	Провести анализ существующих аналогов	10	22.11.22 - 22.01.23	
3	Определиться с инструментами разработки для реализации веб-приложения и обучения модели нейронной сети	10	23.01.23 - 23.04.23	
4	Изучить методы реализации авторизации в приложении	10	24.04.23 - 28.04.23	
5	Рассмотреть существующие типы моделей нейронной сети и выбрать наиболее подходящий под цели данной работы	10	29.04.23 - 05.05.23	
6	Создать и обучить выбранную модель нейронной сети	10	05.05.23 - 09.05.23	
7	Оценить точность работы обученной модели нейронной сети, проверив корректность её работы	10	10.05.23 - 15.05.23	
8	Оформление ВКР	30	16.05.23 - 31.05.23	

Дата выдачи задания "13" сентября 2024 г.

Руководитель магистерской диссертации  13.09.2024 Коинов Н.Н.
(подпись, дата)

Задание к исполнению принял студент  13.09.2024 Репин И.А.
(подпись, дата)

Работу к защите допустить
Декан факультета

 А.Р. Пирогова
(подпись, дата)
14.06.2024

Реферат

Работа содержит 72 листа, 28 рисунков, 4 таблиц, 7 источников, 2 приложения

ЧАТ-БОТ, МАШИННОЕ ОБУЧЕНИЕ, WEB, TYPESCRIPT, REACT,
НЕЙРОННЫЕ СЕТИ, ОПРЕДЕЛЕНИЕ ЭМОЦИЙ, ОБРАБОТКА ТЕКСТА
разработка авторизации в веб приложении

Целью ВКР является сервис, содержащий модуль для определения настроения пользователя для чат-бота, позволяющий определять его эмоциональное состояние по сообщению.

Проект включает в себя разработку чата для взаимодействия с пользователем, обучение модели нейронной сети для распознавания настроения, а также ее использование для определения состояния пользователя по его сообщениям.

В процессе разработки был проведен теоретический анализ предметной области, обоснован выбор программного обеспечения для разработки чат-бота, изучены этапы подготовки текстовых данных для подачи их в модель машинного обучения, проведен анализ моделей машинного обучения и дана оценка их точности, также дано описание итогового приложения, которое реализует пользовательский интерфейс для взаимодействия с ботом.

В результате выполнения магистерской работы был разработан чат-бот позволяющий определять эмоциональное состояние пользователя по сообщению. Осуществлено функциональное тестирование разработанного программного продукта, которое показало корректность его работ.

ПГУ1.09.04.01.12.001 ПЗ				
Изм.	Лист	№ докум	Подпись	Дата
Разраб.		Репин И.А.		13.06.24
Провер.		Коннов Н.Н.		13.06.24
Н. Контр.		Бычков А.С.		13.06.24
Утв.		Митрохин М.А.		13.06.24

Чат-бот для распознавание настроения пользователя и оказания эмоциональной поддержки Пояснительная записка			Лит.	Лист	Листов
				2	72
ФВТ, каф. ВТ гр. 21ВВИм1					

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	9
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	10
__1 Анализ предметной области	14
___1.1 Анализ аналогов.....	16
___1.2 Анализ инструментов разработки	20
___2.1 Анализ методов реализации авторизации в приложении	28
__3 Анализ видов нейронных сетей	32
___3.1 Описание существующих типов нейронных сетей	32
__4 Обучение модели нейронной сети.....	43
___4.1 Выбор датасета	43
___4.2 Описание датасета.....	44
___4.3 Процесс обучения	45
___4.3.1 Анализ данных	45
___4.3.2 Предобработка данных	46
___4.3.3 Обучение модели	48
__5 Результаты реализации приложения	49
___5.1 Описание архитектуры приложения	49
___5.2 Реализация веб-версии приложения.....	50
___5.2.2 Реализация чата в приложении с использованием сокетов и Socket.IO .	56
ЗАКЛЮЧЕНИЕ	59
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	60
ПРИЛОЖЕНИЕ А Листинг программы	61
ПРИЛОЖЕНИЕ Б Презентация.....	67

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей магистерской диссертации применяются следующие сокращения и обозначения

БД – база данных

ИИ – искусственный интеллект

СУБД – система управления базами данных

ЯП – язык программирования

API (Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой

IDE (Integrated development environment) – интегрированная среда разработки

ID (Identifier) – идентификатор объекта

JSON (JavaScript Object Notation) – текстовый формат обмена данными

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

ВВЕДЕНИЕ

Искусственный интеллект становится всё более неотъемлемой частью жизни современного человека и используется им в различных сферах деятельности. Он используется в медицине, финансах, образовании и многих других. Одним из наиболее интересных направлений применения ИИ является распознавание эмоций человека по тексту. Данная функция распознавания эмоций по тексту может быть полезной для создания более эффективных систем коммуникации, анализа настроения пользователей и улучшения качества обслуживания клиентов.

Распознавание эмоций по тексту является сложной задачей, требующей глубокого понимания контекста и семантики текста. Однако с развитием технологий машинного обучения и нейронных сетей эта задача становится всё более решаемой. В последние годы были разработаны различные методы и алгоритмы, которые позволяют автоматически определять эмоции человека на основе его текстовых сообщений.

Использование ИИ для распознавания эмоций человека по его сообщениям в сфере медицины и оказания психологической помощи может стать мощным инструментом, который позволит более точно и эффективно реагировать на потребности пациентов, что показывает актуальность данной темы.

В современном мире люди все чаще обращаются за помощью к специалистам в области медицины и психологии. Однако, несмотря на все усилия, иногда бывает сложно понять, что именно чувствует пациент, какие эмоции он испытывает и как это влияет на его состояние. Именно здесь на помощь приходит искусственный интеллект.

Данная магистерская диссертация преследует следующие цели:

- разработка авторизации в веб приложении,
- разработка чата для взаимодействия с пользователем,
- обучение модели нейронной сети для распознавания настроения,

– использование обученной нейронной модели для определения состояния пользователя по его сообщениям.

Для реализации преследуемых магистерской диссертацией целей необходимо:

- проанализировать предметную область,
- провести анализ существующих аналогов,
- определиться с инструментами разработки для реализации веб-приложения и обучения модели нейронной сети,
- изучить методы реализации авторизации в приложении,
- рассмотреть существующие типы моделей нейронной сети и выбрать наиболее подходящий под цели данной работы,
- создать и обучить выбранную модель нейронной сети,
- оценить точность работы обученной модели нейронной сети, проверив корректность её работы.

Благодаря поставленным задачам, в ходе выполнения работы, данная магистерская диссертация была разделена на следующие разделы:

Вступительный раздел, в котором анализируется предметная область, существующие аналоги, инструменты разработки, а также реализации авторизации в веб-приложении. В финале каждого пункта данного раздела описаны результаты анализа и выбора инструментов разработки и реализации.

Во втором разделе рассматриваются типы моделей нейронных сетей.

В третьем разделе рассказывается про процесс обучения модели нейросети и оценивание её работы.

В заключительном разделе показаны результаты реализации приложения и его архитектура.

В результате проделанной работы, продуктом данной диссертации будет являться реализованный модуль для определения настроения пользователя, созданный на основе выбранных решений, найденных путем проведения исследований и анализа предметной области.

1 Анализ предметной области

В последние годы наблюдается стремительный рост интереса к использованию чат-ботов, которые представляют собой автоматизированные системы, способные взаимодействовать с пользователями через текстовые или голосовые сообщения. Чат-боты могут выполнять различные функции, такие как предоставление информации, помощь в решении задач, общение с клиентами и т.д. Основными причинами роста популярности чат-ботов являются:

- удобство и доступность: Чат-боты доступны 24/7, что позволяет пользователям получать помощь и информацию в любое время и в любом месте. Это особенно важно для людей, живущих в удаленных районах или работающих в ночное время,

- экономия времени и ресурсов: Чат-боты способны обрабатывать большое количество запросов одновременно, что снижает нагрузку на сотрудников и ускоряет процесс решения задач,

- персонализация: Чат-боты могут адаптироваться к потребностям и предпочтениям пользователей, предлагая им индивидуальные рекомендации и услуги,

- интеграция с другими технологиями: Чат-боты легко интегрируются с различными платформами и системами, такими как CRM, ERP и другие, что делает их еще более эффективными и полезными.

Однако, несмотря на все преимущества, чат-боты имеют некоторые ограничения, связанные с их способностью понимать и интерпретировать запросы пользователей. В частности, они могут столкнуться с трудностями при распознавании контекста, эмоций и настроения пользователя. Для решения этой проблемы используются технологии искусственного интеллекта (ИИ), которые позволяют чат-ботам лучше понимать и реагировать на запросы пользователей.

Одним из ключевых аспектов использования ИИ в чат-ботах является способность распознавания настроения пользователя по его сообщениям. Это позволяет чат-боту адаптировать свой ответ в зависимости от эмоционального состояния пользователя, делая взаимодействие более естественным и эффективным.

Например, если пользователь выражает недовольство или раздражение, чат-бот может предложить ему дополнительные варианты решения проблемы или предоставить дополнительную информацию. Если же пользователь проявляет интерес или любопытство, чат-бот может предоставить более подробную информацию или предложить дополнительные услуги.

Таким образом, использование ИИ для распознавания настроения пользователя является важным шагом в развитии чат-ботов. Оно позволяет сделать взаимодействие с ними более персонализированным и эффективным, а также повысить уровень удовлетворенности пользователей.

В ходе исследования было выявлено, что использование чат-ботов с ИИ-помощником для распознавания настроения пользователей становится все более популярным в различных отраслях, таких как здравоохранение, образование, финансы, туризм и других. Это связано с тем, что чат-боты с ИИ могут предоставлять более точные и персонализированные ответы, что повышает уровень удовлетворённости клиентов и эффективность работы компаний.

В целом, использование чат-ботов с ИИ для распознавания настроения пользователей является перспективным направлением развития технологий искусственного интеллекта. Оно позволяет создавать более удобные и эффективные системы взаимодействия с клиентами, которые могут адаптироваться к их потребностям и предпочтениям.

Это особенно важно в условиях растущей конкуренции на рынке, где компании стремятся предоставлять клиентам наилучший сервис и опыт взаимодействия.

Однако стоит отметить, что внедрение таких систем требует тщательной проработки и тестирования, чтобы обеспечить корректное и этичное использование данных о пользователях. Также необходимо учитывать возможные риски и ограничения, связанные с использованием технологий искусственного интеллекта, такие как возможность предвзятости или неправильного толкования данных.

1.1 Анализ аналогов

Woebot — это приложение для психологической поддержки, которое использует искусственный интеллект для предоставления консультаций и помощи пользователям. Оно разработано с целью помочь людям справиться со стрессом, тревогой и депрессией.

Woebot работает на основе алгоритмов машинного обучения и искусственного интеллекта, которые позволяют ему анализировать поведение пользователя и предоставлять персонализированные рекомендации и поддержку. Приложение также предлагает различные упражнения и техники для улучшения эмоционального состояния, такие как медитация, дыхательные упражнения и другие практики. Интерфейс Woebot представлен на рисунке 1.

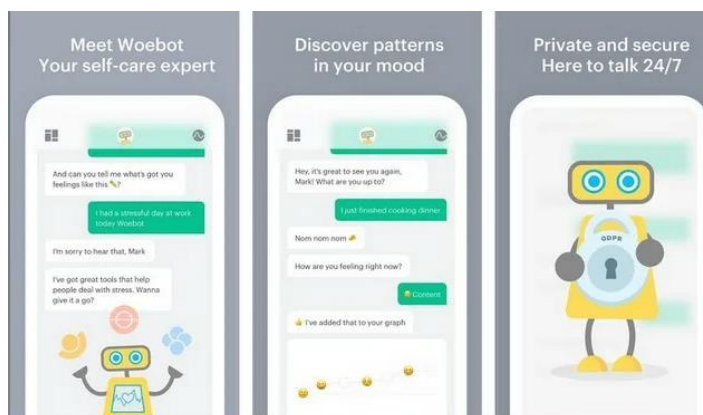


Рисунок 1 – Интерфейс приложения Woebot

Talkspace — это приложение, предназначенное для оказания психологической помощи и поддержки. Оно предоставляет возможность пользователям получать консультации от лицензированных психологов и терапевтов через онлайн-платформу. Talkspace предлагает широкий спектр услуг, включая индивидуальные и групповые сеансы, а также поддержку в решении различных проблем, таких как депрессия, тревога, стресс и семейные конфликты.

Приложение также предоставляет инструменты для отслеживания прогресса и оценки эффективности терапии. Пользователи могут общаться с терапевтами через чат, видеозвонки или голосовые сообщения. Это позволяет им получать помощь в удобное время и в удобном формате. Интерфейс Talkspace представлен на рисунке 2.

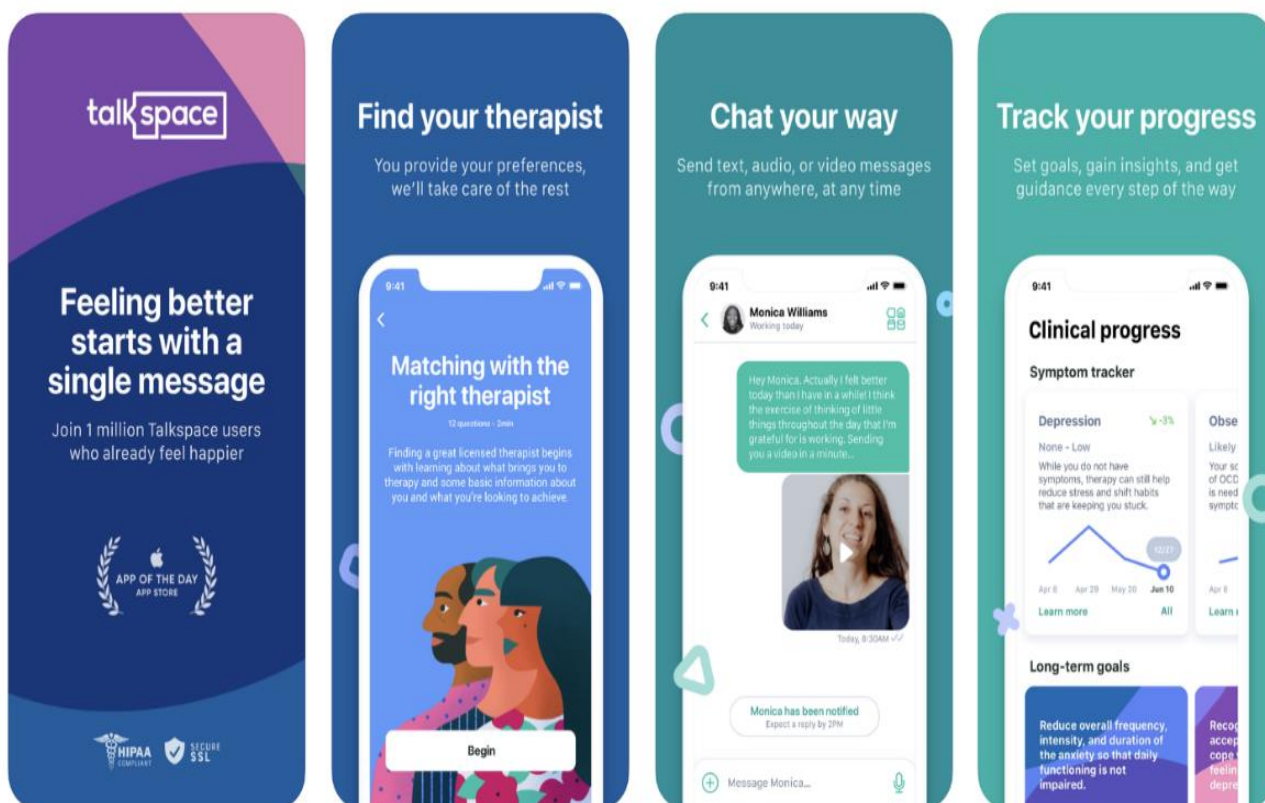


Рисунок 2 – Интерфейс сайта Talkspace

Headspace — ещё одно популярное приложение для медитации и управления стрессом. Оно предлагает разнообразные программы медитации, дыхательных упражнений и других практик, направленных на снижение стресса и улучшение концентрации.

Приложение помогает пользователям научиться правильно дышать, расслабляться и сосредотачиваться. Оно также предоставляет ежедневные напоминания о необходимости практики медитации и других упражнений. Headspace подходит для людей, желающих улучшить своё эмоциональное состояние и повысить уровень осознанности. Интерфейс Headspace представлен на рисунке 3.

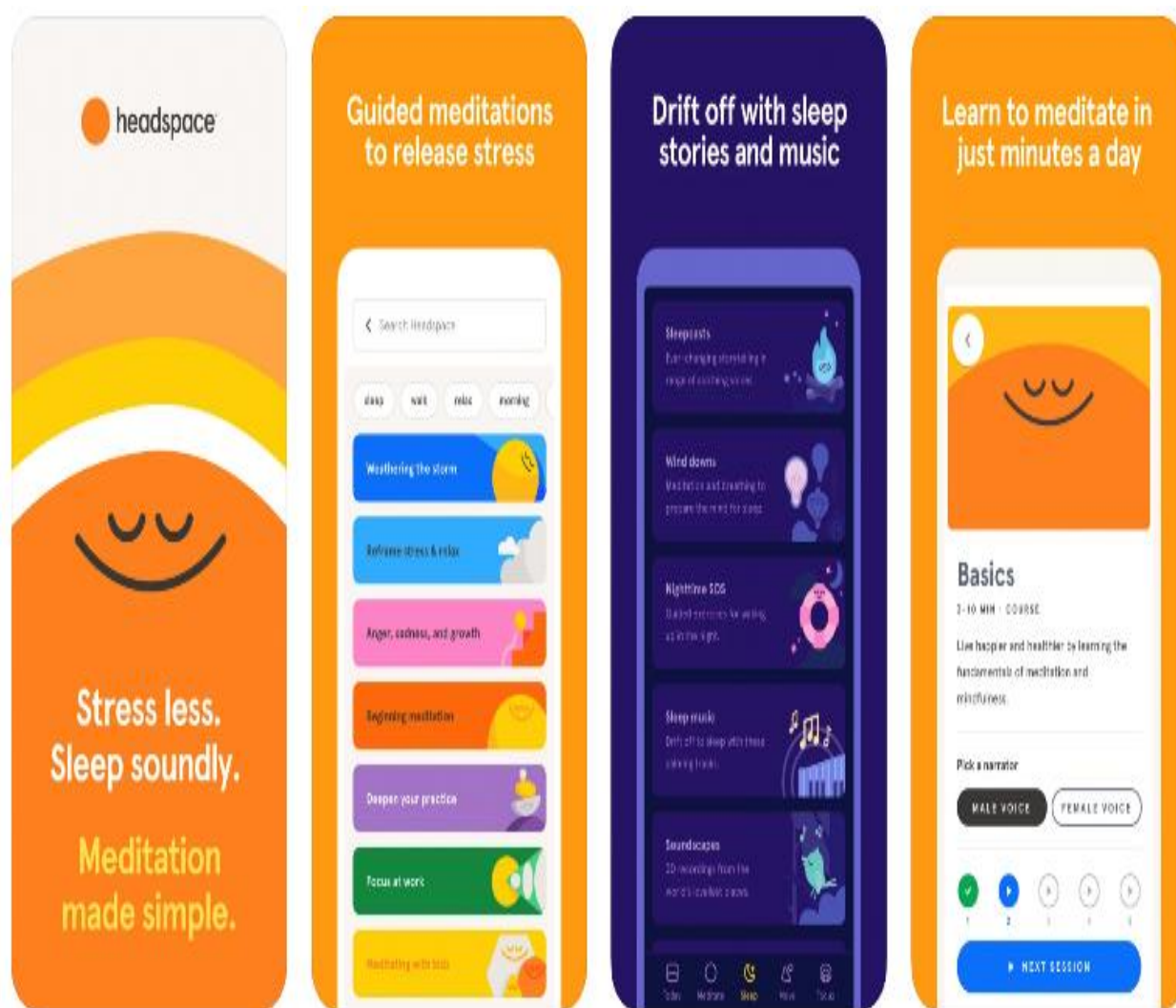


Рисунок 3 - Интерфейс приложения Headspace

Все сильные и слабые стороны данных приложений приведены в

таблице 1.

Таблица 1 – Сильные и слабые стороны рассматриваемых аналогов

Приложение	Сильные стороны	Слабые стороны
Woebot	Использование ИИ для анализа поведения пользователя и предоставления персонализированных рекомендаций. Разнообразие упражнений и техник для улучшения эмоционального состояния.	Ограниченность в выборе тем консультаций. Отсутствие возможности прямого общения с психологом.
Talkspace	Широкий спектр услуг, включая индивидуальные и групповые сеансы. Возможность отслеживания прогресса и оценки эффективности терапии. Общение с терапевтами через чат, видеозвонки или голосовые сообщения.	Зависимость от качества работы конкретного терапевта. Необходимость оплаты за каждую сессию.
Headspace	Разнообразные программы медитации, дыхательных упражнений и других практик. Ежедневные напоминания о необходимости практики медитации. Улучшение концентрации и снижение стресса.	Отсутствие индивидуального подхода. Ограниченность функций без платной подписки.

Из таблицы видно, что у каждого приложения есть свои плюсы и минусы. Woebot предлагает широкий спектр упражнений и техник, но ограничен в выборе тем для консультаций.

Talkspace предоставляет возможность общения с лицензированными специалистами, но зависит от их профессионализма и требует оплаты за каждую консультацию. Headspace помогает улучшить концентрацию и снизить стресс, но не имеет индивидуального подхода и ограничивает функции бесплатной версией.

Выбор приложения зависит от индивидуальных потребностей и предпочтений пользователя. Если необходима помощь в решении конкретных проблем, то лучше выбрать Talkspace. Для улучшения общего эмоционального состояния и снижения стресса подойдёт Headspace. А Woebot может быть полезен для тех, кто хочет получить поддержку и рекомендации по управлению эмоциями. Однако важно помнить, что эти приложения не заменяют профессиональную психологическую помощь и консультацию специалистов.

1.2 Анализ инструментов разработки

Анализ инструментов разработки приложения — это важный этап в процессе создания программного обеспечения, который позволяет определить наиболее подходящие инструменты для реализации проекта.

Анализ инструментов помогает разработчикам выбрать оптимальные решения для достижения поставленных целей и задач, а также обеспечить высокое качество конечного продукта. В ходе анализа разработчики могут оценить функциональность, производительность, удобство использования и другие характеристики инструментов, чтобы принять обоснованное решение о том, какие из них использовать в проекте.

1.2.1 Выбор фреймворка для создания веб-приложения

В ходе исследования было выявлено, что существует несколько популярных фреймворков для создания веб-приложений. Они отличаются своей функциональностью, производительностью и простотой использования.

– React – это JavaScript-библиотека для создания пользовательских интерфейсов. Она позволяет разработчикам создавать интерактивные и динамические веб-приложения. React широко используется для разработки одностраничных приложений (SPA) и мобильных приложений. Он также имеет большое сообщество разработчиков, которые создают различные расширения и инструменты для упрощения работы с фреймворком.

– Angular – это фреймворк для создания веб-приложений на основе TypeScript. Angular предоставляет разработчикам набор инструментов для создания сложных и масштабируемых приложений. Он имеет модульную структуру, которая позволяет легко добавлять или удалять функции в приложениях. Angular также поддерживает разработку мобильных приложений и имеет большое количество готовых компонентов для различных задач.

– Vue.js – это ещё один популярный фреймворк, который позволяет создавать веб-приложения на основе JavaScript. Vue.js отличается своей лёгкостью и гибкостью, что делает его привлекательным для разработчиков. Он позволяет быстро создавать прототипы и небольшие приложения, а также легко интегрируется с другими библиотеками и фреймворками. Vue.js также имеет активное сообщество разработчиков и множество готовых компонентов.

Все достоинства и недостатки фреймворков приведены в таблице 2.

Таблица 2 – достоинства и недостатки рассматриваемых фреймворков

Фреймворк	Сильные стороны	Слабые стороны
React	Популярность и большое сообщество разработчиков, что обеспечивает поддержку и наличие множества готовых компонентов. Возможность создания интерактивных и динамических веб-приложений. Лёгкость интеграции с другими библиотеками и фреймворками.	Сложность в изучении для новичков. Необходимость дополнительных библиотек для некоторых функций.
Angular	Поддержка разработки сложных и масштабируемых приложений. Модульная структура, позволяющая легко добавлять или удалять функции в приложении. Наличие большого количества готовых компонентов для различных задач.	Более сложный синтаксис по сравнению с React. Ограничения в использовании сторонних библиотек.
Vue.js	Лёгкость и гибкость, позволяющие быстро создавать прототипы и небольшие приложения. Простота интеграции с другими библиотеками и фреймворками. Активное сообщество разработчиков и множество готовых компонентов.	Ограниченные возможности для создания сложных приложений. Отсутствие некоторых функций, доступных в других фреймворках.

Поскольку React является популярным и широко используемым фреймворком для создания веб-приложений благодаря своей гибкости, простоте интеграции с другими инструментами и наличию большого сообщества разработчиков, было принято решение использовать именно его. Он предоставляет разработчикам возможность создавать интерактивные и динамические веб-приложения, а также имеет множество готовых компонентов и расширений.

1.2.2 Выбор языка программирования веб-версии

TypeScript — это типизированный язык программирования, надмножество JavaScript, разработанное компанией Microsoft. TypeScript добавляет в JavaScript статическую типизацию, классы, интерфейсы и другие функции, которые делают код более структурированным и безопасным. Это позволяет разработчикам писать более сложные приложения с меньшим количеством ошибок.

Основные особенности TypeScript:

- статическая типизация: TypeScript требует явного указания типов переменных, параметров функций и возвращаемых значений. Это помогает избежать ошибок и упрощает отладку,
- поддержка объектно-ориентированного программирования: TypeScript поддерживает классы, наследование, полиморфизм и другие концепции объектно-ориентированного программирования,
- интеграция с другими инструментами и фреймворками: TypeScript легко интегрируется с различными инструментами и фреймворками, такими как React, Angular и Vue.js,
- компиляция в JavaScript: TypeScript компилируется в чистый JavaScript, который может выполняться в любом браузере или среде выполнения.

JavaScript — это язык программирования, разработанный для создания интерактивных веб-приложений. JavaScript является одним из самых популярных языков программирования благодаря своей простоте, гибкости и доступности.

Основные особенности JavaScript:

- динамическая типизация: JavaScript не требует явного указания типов переменных и параметров функций. Это делает код более простым и гибким, но также может привести к ошибкам,
- поддержка различных парадигм программирования: JavaScript поддерживает процедурное, функциональное и объектно-ориентированное программирование,
- большое сообщество разработчиков: у JavaScript большое и активное сообщество разработчиков, которое создаёт множество готовых решений и библиотек,
- простота изучения и использования: JavaScript имеет простой и понятный синтаксис, что делает его доступным для начинающих разработчиков.

Достоинства и недостатки выбранных языков представлены в таблице 3.

Таблица 3 – достоинства и недостатки языков программирования

Языки программирования	Сильные стороны	Слабые стороны
TypeScript	Строгая типизация, которая помогает избежать ошибок и упрощает отладку. Поддержка объектно-ориентированного и функционального программирования. Интеграция с другими инструментами и фреймворками.	Более сложный синтаксис по сравнению с JavaScript. Необходимость дополнительной настройки и конфигурации.
JavaScript	Простота изучения и использования. Большое сообщество разработчиков и множество готовых решений. Поддержка различных парадигм программирования.	Отсутствие строгой типизации, что может привести к ошибкам. Ограниченные возможности для создания сложных приложений.

Поскольку TypeScript является мощным инструментом для разработки веб-приложений благодаря своей строгой типизации и поддержке объектно-ориентированного программирования, а также позволяет создавать более надёжные и масштабируемые приложения и прекрасно интегрируется с уже выбранным фреймворком, было решено использовать именно его.

1.2.3 Выбор языка для обучения модели нейронной сети

Поскольку Выбор языка для обучения модели нейронной сети зависит от нескольких факторов, таких как доступность данных, сложность задачи и предпочтения разработчика. Вот некоторые из наиболее популярных языков для создания и обучения моделей:

- Python – это один из самых популярных языков программирования для машинного обучения и искусственного интеллекта. Он имеет множество библиотек и фреймворков, которые упрощают процесс разработки и позволяют быстро создавать прототипы моделей. Python также имеет большое сообщество разработчиков, что делает его отличным выбором для начинающих,

- R – язык программирования, который широко используется в статистическом анализе и машинном обучении. R предоставляет множество функций и пакетов для анализа данных и построения моделей. Он также является открытым исходным кодом, что позволяет разработчикам свободно использовать и модифицировать код,

- Julia – относительно новый язык программирования, предназначенный для научных вычислений и машинного обучения. Julia сочетает в себе скорость С-подобных языков с простотой использования Python. Это делает его привлекательным выбором для разработчиков, которым требуется высокая производительность и гибкость,

- C++ – мощный язык программирования, используемый для создания высокопроизводительных приложений. C++ может быть использован для разработки сложных моделей машинного обучения, требующих высокой скорости работы. Однако, C++ требует более глубокого понимания алгоритмов и структур данных, что может затруднить его использование для начинающих разработчиков,

– Java – объектно-ориентированный язык программирования, широко используемый для разработки корпоративных приложений. Java также может быть использован для создания моделей машинного обучения, особенно в контексте больших данных и распределённых систем,

– JavaScript – популярный язык для веб-разработки, который также используется для создания моделей машинного обучения в браузере. JavaScript предоставляет доступ к мощным библиотекам и фреймворкам, таким как TensorFlow.js, которые позволяют создавать и обучать модели прямо в браузере,

– MATLAB – коммерческий язык программирования, разработанный для математических вычислений и анализа данных. MATLAB предоставляет множество инструментов для создания и анализа моделей машинного обучения. Однако, MATLAB может быть дорогим для использования, и его функциональность может быть ограничена по сравнению с другими языками.

Выбор языка для обучения модели нейронной сети является важным этапом в процессе разработки. Среди множества доступных языков программирования, **Python** выделяется как один из наиболее популярных и подходящих вариантов для создания и обучения моделей машинного обучения.

Python обладает рядом преимуществ, которые делают его привлекательным выбором для разработчиков:

– большое сообщество разработчиков: Python имеет огромное сообщество разработчиков, что обеспечивает хорошую поддержку и ресурсы. Это позволяет быстро находить ответы на вопросы, получать помощь и делиться опытом с другими разработчиками,

– множество библиотек и фреймворков Python предлагает широкий спектр библиотек и фреймворков для машинного обучения, таких как TensorFlow, PyTorch, Scikit-learn и другие. Эти инструменты упрощают процесс разработки и позволяют быстро создавать прототипы моделей,

- простой и понятный синтаксис: Синтаксис Python является простым и понятным, что делает его доступным для начинающих разработчиков. Это также способствует быстрой разработке и отладке кода,

- гибкость и расширяемость: Python предоставляет гибкость и возможность расширения функциональности за счёт использования сторонних библиотек и модулей. Это позволяет адаптировать код под конкретные требования проекта.

Благодаря данным преимуществам было решено использовать именно его.

1.3 Анализ методов реализации авторизации в приложении

В современном мире, где информационные технологии играют важную роль, безопасность данных становится одним из ключевых аспектов. Одним из способов обеспечения безопасности является авторизация пользователей в приложении. На данный момент самыми популярными способами реализации аутентификации в приложениях являются следующие методы:

- базовая аутентификация,
- OAuth 2.0,
- OpenID Connect,
- JWT.

Базовая аутентификация является одним из самых простых и распространённых методов авторизации. Она основана на использовании имени пользователя и пароля для проверки подлинности пользователя. Этот метод используется во многих веб-приложениях и API.

OAuth 2.0 это протокол авторизации, который позволяет пользователям получать доступ к ресурсам приложения без необходимости предоставлять свои учётные данные.

Вместо этого пользователь получает токен доступа, который он может использовать для доступа к ресурсу.

OpenID Connect – это расширение протокола OAuth 2.0, которое предоставляет дополнительные функции, такие как аутентификация и авторизация. OIDC позволяет пользователям использовать один и тот же токен для доступа к различным сервисам.

JWT – это формат токена, который может использоваться для передачи информации о пользователе между различными сервисами. JWT состоит из трёх частей: заголовка, полезной нагрузки и подписи.

Все достоинства и недостатки данных методов реализации авторизации вынесены в таблицу 4.

Таблица 4 – достоинства и недостатки методов авторизации

Метод	Описание	Преимущества	Недостатки
Базовая аутентификация (Basic Authentication)	Простой метод, основанный на использовании имени пользователя и пароля для проверки подлинности пользователя.	Простота реализации, низкая стоимость внедрения, подходит для небольших приложений.	Слабая защита от атак, пароль может быть украден или взломан, не подходит для крупных приложений с большим количеством пользователей.

Продолжение таблицы 4

OAuth 2.0	<p>Протокол авторизации, который позволяет пользователям получать доступ к ресурсам приложения без необходимости предоставлять свои учётные данные. Вместо этого пользователь получает токен доступа, который он может использовать для доступа к ресурсу.</p>	<p>Высокая степень защиты, возможность использования различных типов токенов, поддержка различных сценариев использования.</p>	<p>Сложность реализации, необходимость настройки сервера авторизации, может быть сложно интегрировать с существующими системами.</p>
OpenID Connect (OIDC)	<p>Расширение протокола OAuth 2.0, которое предоставляет дополнительные функции, такие как аутентификация и авторизация. OIDC позволяет пользователям использовать один и тот же токен для доступа к различным сервисам.</p>	<p>Высокий уровень безопасности, удобство для пользователей, совместимость с другими стандартами.</p>	<p>Сложность настройки, требуется дополнительный сервер авторизации.</p>

Продолжение таблицы 4

JWT (JSON Web Token)	Формат токена, который может использоваться для передачи информации о пользователе между различными сервисами. JWT состоит из трёх частей: заголовка, полезной нагрузки и подписи.	Лёгкость интеграции, гибкость в настройке, безопасность на уровне транспортного слоя.	Риск перехвата токена, ограниченный срок действия.
----------------------	--	---	--

В результате анализа данных представленных в таблице 4, а также в виду ограниченности финансовых ресурсов, было решено использовать метод JWT токенов, поскольку JWT обеспечивает лёгкость интеграции, гибкость в настройке и безопасность на уровне транспортного слоя. Однако он также имеет некоторые слабые стороны, такие как риск перехвата токена, что весьма просто нивелируется добавлением времени жизни токена.

Важно отметить, что выбор метода JWT не является единственным фактором, определяющим безопасность приложения. Необходимо также обеспечить защиту данных на всех уровнях, включая шифрование, аутентификацию и авторизацию. Только комплексный подход к обеспечению безопасности позволит создать надёжное и безопасное приложение.

2 Анализ видов нейронных сетей

Прежде чем приступать к созданию нейронной сети для распознавания настроения пользователя, важно разобраться в существующих моделях нейронных сетей и методах их обучения, поскольку это позволит выбрать наиболее оптимальные варианты для удовлетворения целей данной работы.

Как же работает нейросеть? Нейронная сеть, являющаяся математической моделью, состоит из огромного количества взаимосвязанных элементов, которые передают информацию друг другу, имитируя мозга человека. Эти элементы называют нейронами, по аналогии с элементами в голове человека.

В основе работы нейросетей лежит принцип обучения. ИИ обучается на большом количестве данных, чтобы научиться делать определённые выводы или прогнозы. Это позволяет ей решать сложные задачи, такие как распознавание образов, предсказание погоды или анализ медицинских данных.

Обучение нейронных сетей происходит с помощью алгоритмов, которые позволяют им находить закономерности в данных. После обучения нейронная сеть может быть использована для решения новых задач, аналогичных тем, на которых она была обучена.

Существует несколько видов нейронных сетей, каждый из которых имеет свои особенности и области применения и описан далее в работе.

2.1 Описание существующих типов нейронных сетей

Для достижения целей, поставленных данной работой, необходимо изучить существующие типы нейросетей и выбрать самый оптимальный и подходящий.

На данный момент выделяют семь типов нейросетей.

2.1.1 Однослойные нейронные сети

Однослойные нейронные сети (ОН) представляют собой простейший тип нейросетей, который состоит из одного слоя нейронов. Эти сети используются для решения простых задач классификации и регрессии, где требуется определить принадлежность объекта к определённому классу или предсказать значение непрерывной переменной. ОН имеют простую структуру и могут быть быстро обучены на основе данных. Однако они не обладают высокой точностью и сложностью, поэтому их применение ограничено простыми задачами.

ОН состоит из входного слоя, скрытого слоя (который в данном случае является единственным) и выходного слоя. Входной слой принимает входные данные, которые затем передаются через скрытый слой, где происходит обработка информации. Выходной слой выдаёт результат обработки входных данных.

Для обучения ОН используется метод обратного распространения ошибки. Этот метод позволяет корректировать веса связей между нейронами таким образом, чтобы минимизировать ошибку между ожидаемым и фактическим результатом. После обучения сеть может быть использована для классификации новых объектов или прогнозирования значений переменных.

Несмотря на свою простоту, ОН сети могут быть полезны в некоторых случаях. Например, они могут использоваться для предварительной обработки данных перед подачей их в более сложные модели. Также они могут применяться для быстрого прототипирования и проверки гипотез о том, как работает модель.

К сожалению, ОН сети не подходят для сложных задач, таких как распознавание образов или прогнозирование временных рядов.

Для этих задач необходимо использовать более сложные многослойные нейронные сети. Алгоритм ОН представлен на рисунке 4.

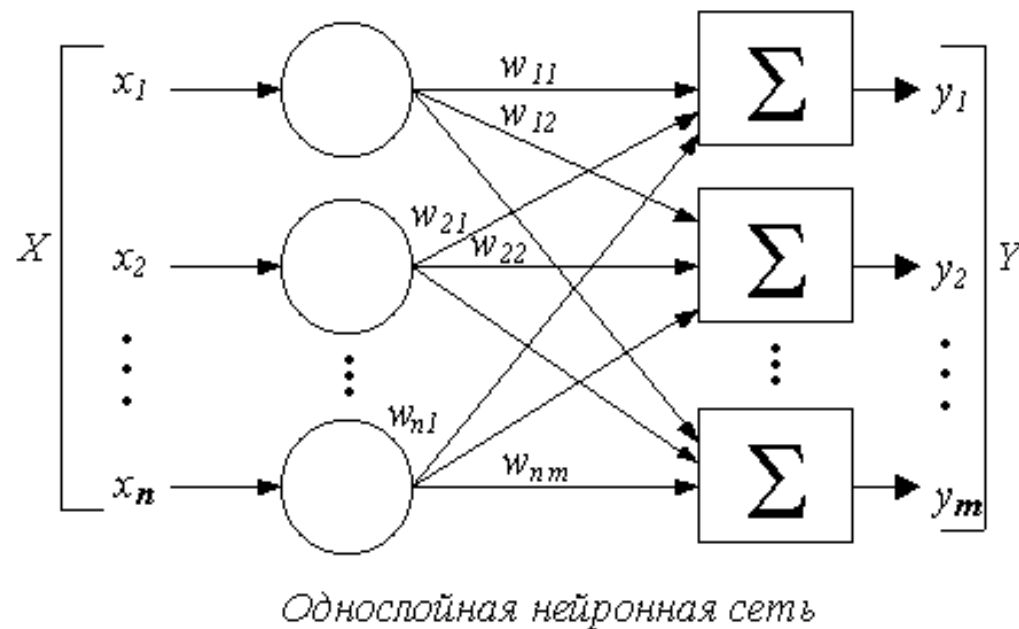


Рисунок 4 – Алгоритм однослойных нейронных сетей

2.1.2 Многослойные нейронные сети

Многослойные нейронные сети (МН) являются более сложной и мощной моделью, чем однослойные. Они состоят из нескольких слоёв нейронов, включая входной, скрытый и выходной слой. В результате проведённой работы было установлено, что многослойные нейронные сети могут решать более сложные задачи, такие как распознавание образов, прогнозирование временных рядов и другие.

МН имеют несколько преимуществ перед однослойными. Во-первых, они могут обрабатывать более сложную информацию, так как каждый слой может выполнять свою функцию. Например, первый слой может выделять основные признаки изображения, второй слой — классифицировать эти признаки, а третий — принимать решение о том, к какому классу относится изображение. Во-вторых, многослойные сети могут обучаться на более сложных данных, таких как изображения или аудиозаписи.

Это позволяет им достигать более высокой точности в задачах, требующих глубокого понимания данных.

Если говорить о недостатках данного типа нейросетей, то во-первых, их обучение может быть более сложным и длительным, чем у ОН. Это связано с тем, что каждый слой должен быть обучен отдельно, и это может потребовать больше времени и ресурсов. Во-вторых, МН могут быть подвержены переобучению, то есть они могут запоминать данные, на которых они были обучены, вместо того чтобы обобщать их.

Это может привести к снижению точности на новых данных. Чтобы избежать переобучения, необходимо использовать методы регуляризации, такие как L1 или L2 регуляризация. Алгоритм работы МН рассмотрен на рисунке 5.

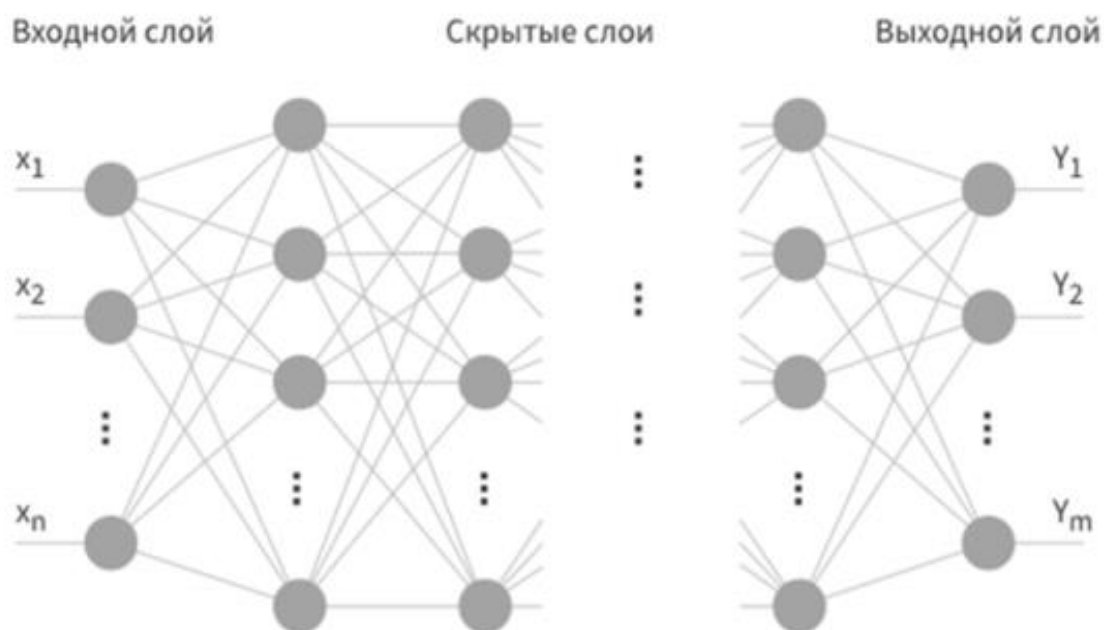


Рисунок 5 – Алгоритм многослойных нейронных сетей

В общем и целом, многослойные нейронные сети являются весьма полезным инструментом для решения задач повышенной сложности.

Они могут использоваться для распознавания образов, прогнозирования временных рядов, обработки естественного языка и других достижения других целей. Но стоит внимательно подходить к выбору архитектуры, обучению и тестированию.

2.1.3 Рекуррентные нейронные сети

Рекуррентные нейронные сети (RNN) – это особый вид нейронных сетей, которые способны обрабатывать последовательности данных. Они широко используются в таких областях, как обработка естественного языка, анализ временных рядов и другие задачи, где важна информация о предыдущих состояниях системы.

В ходе изучения информации данном типе нейросетей, стало ясно, что RNN имеют уникальную способность запоминать и использовать информацию о прошлых событиях для принятия решений в настоящем. Это делает их особенно полезными в задачах, где необходимо учитывать контекст или временные зависимости.

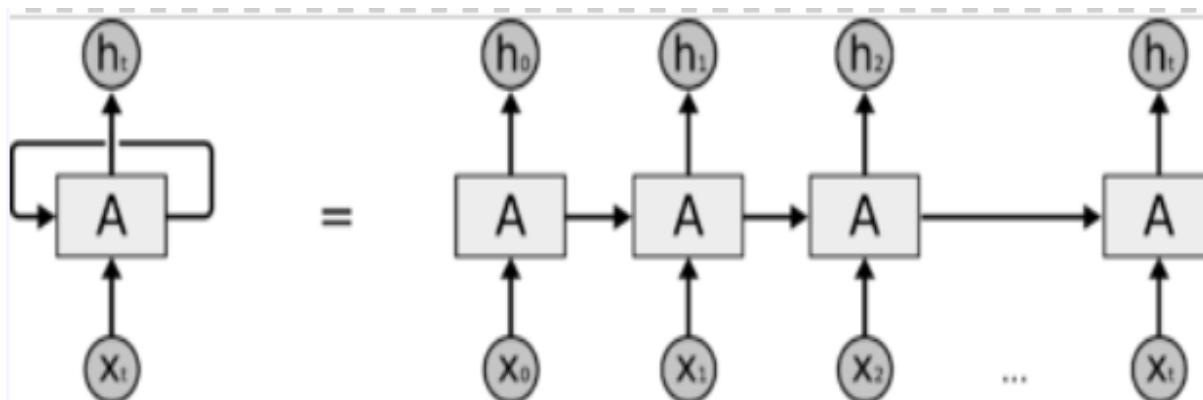
Ключевой особенностью RNN является наличие обратной связи, которая позволяет им сохранять информацию о предыдущих входных данных.

Это достигается с помощью специальных элементов, называемых ячейками памяти, которые могут хранить информацию на протяжении нескольких шагов.

В результате RNN могут обрабатывать длинные последовательности данных, сохраняя при этом информацию о контексте. Алгоритм работы RNN представлен на рисунке 6.

Рисунок 6 – Алгоритм RNN

Однако RNN также имеют некоторые ограничения. Во-первых, они



могут быть подвержены проблеме затухания градиента, когда информация о прошлых состояниях теряется со временем. Это может привести к снижению точности модели. Чтобы решить эту проблему, были разработаны различные методы, такие как использование LSTM (Long Short-Term Memory) или GRU (Gated Recurrent Unit).

К сожалению, RNN требуют большего количества вычислений по сравнению с другими типами нейронных сетей. Это связано с необходимостью обработки каждой позиции последовательности отдельно. Однако современные вычислительные мощности позволяют эффективно обучать и использовать RNN даже для больших объёмов данных.

Таким образом, RNN могут быть полезны при работе с последовательностями данных. Они могут использоваться в различных областях, таких как распознавание речи, машинный перевод и генерация текста.

2.1.4 Свёрточные нейронные сети

Свёрточные нейронные сети (CNN) представляют собой особый тип нейронных сетей, которые специализируются на обработке изображений. Они широко используются в таких областях, как компьютерное зрение, распознавание объектов и классификация изображений.

CNN обладают рядом преимуществ перед другими типами нейронных сетей. Они способны эффективно выделять признаки из изображений, такие как края, углы и текстуры. Это позволяет им создавать более точные модели для классификации и распознавания изображений. Также, CNN имеют меньшую чувствительность к изменениям в масштабе, повороте и освещении, что делает их более устойчивыми к вариациям входных данных.

CNN является использует так называемые «свёрточные» слои. Эти слои выполняют операцию свёртки над входными данными, которая заключается в умножении каждого пикселя изображения на матрицу весов и суммировании результатов. В результате свёртки получается новое изображение, которое содержит информацию о признаках исходного изображения. Пример свёртки двух матриц показан на рисунке 7.

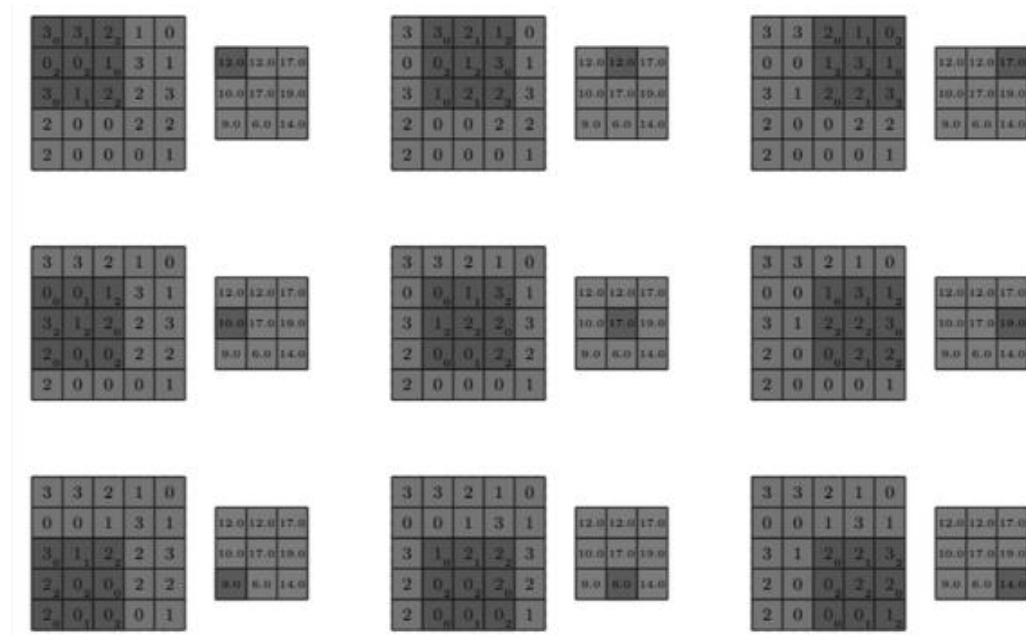


Рисунок 7 – Пример свёртки двух матриц

После свёрточного слоя обычно следуют слои «пулинга» (или субдискретизации), которые уменьшают размерность выходных данных с целью уменьшения количества параметров модели и предотвращения переобучения. Затем следуют полносвязные слои, которые преобразуют выходные данные в вектор признаков, который может быть использован для классификации или регрессии.

У CNN есть и недостатки. Они требуют большого количества данных для обучения, чтобы достичь высокой точности и могут быть сложными для интерпретации, поскольку процесс свёртки и пулинга приводит к потере информации о местоположении признаков в исходном изображении. Тем не менее, эти ограничения не умаляют важности CNN в области компьютерного зрения.

2.1.5 Рекуррентные нейронные сети

Существует вид нейронных сетей, которые состоят из двух подсетей: генератора и дискриминатора. Такие нейросети называются генеративно-сопоставительными (GAN). Генератор создаёт новые данные, а дискриминатор пытается отличить их от реальных данных.

Основная идея GAN заключается в том, что генератор пытается обмануть дискриминатор, создавая данные, которые выглядят как реальные. Дискриминатор, в свою очередь, пытается отличить эти данные от настоящих. В процессе обучения обе подсети становятся всё более и более эффективными, и в конечном итоге генератор может создавать данные, неотличимые от реальных. Архитектура данной модели представлена на рисунке 8.

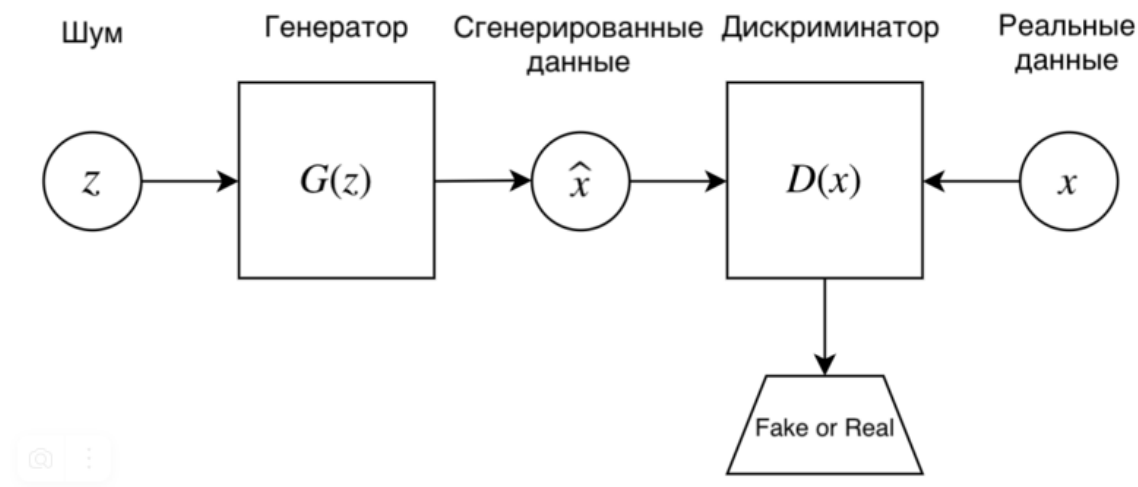


Рисунок 8 – Архитектура GAN

GAN могут создавать очень реалистичные данные, которые трудно отличить от настоящих и не требуют предварительного обучения на большом наборе данных, как это требуется для других методов.

Вместо этого они обучаются на небольшом наборе данных, который используется для настройки параметров генератора и дискриминатора.

Но есть и нюансы. Обучение может быть сложным и длительным процессом, а GAN могут быть подвержены неустойчивости, когда генератор начинает генерировать данные, которые дискриминатор не может отличить от реальных. Это может привести к тому, что модель будет генерировать повторяющиеся или бессмысленные данные.

GAN могут использоваться для создания изображений, видео, музыки и других видов данных.

2.1.6 Автоэнкодеры

Автоэнкодеры — это особый вид нейронных сетей, которые используются для сжатия данных и извлечения признаков из них. Они состоят из двух частей: кодировщика и декодировщика.

Кодировщик преобразует входные данные в более компактное представление, а декодировщик восстанавливает исходные данные из этого представления. Алгоритм работы представлен на рисунке 9.

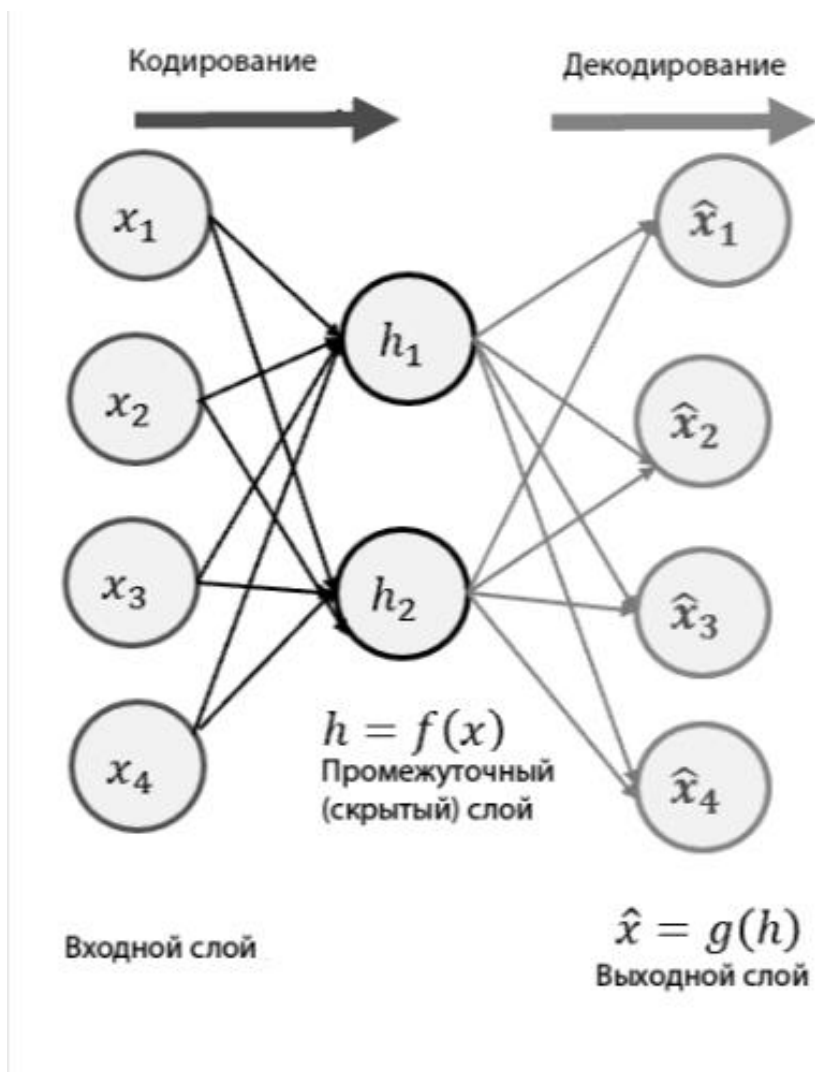


Рисунок 9 – Автоэнкодер

Автоэнкодеры обучают модель восстанавливать входные данные на выходе. Это достигается путём минимизации ошибки между входными данными и их восстановлением. В процессе обучения модель учится выделять наиболее важные признаки во входных данных и использовать их для восстановления.

Данный тип имеют несколько преимуществ перед другими методами сжатия данных. Во-первых, они могут автоматически извлекать признаки из

данных без необходимости ручного выбора признаков. Во-вторых, они могут работать с любыми типами данных, включая изображения, текст и аудио.

К недостаткам можно отнести сложное и длительное обучение, да и не всегда можно обеспечить хорошее восстановление данных, особенно если данные содержат шум или искажения.

Автоэнкодеры могут использоваться для уменьшения размерности данных перед их подачей в другие модели, обнаружения аномалий в данных и других целей.

2.1.7 Глубокие нейронные сети

Глубокие нейронные сети состоят из большого количества слоёв. Они могут достигать высокой точности в сложных задачах, таких как обработка изображений и речи. DNN используют несколько слоёв нейронов для обработки данных. Каждый слой выполняет свою функцию, например, выделение признаков, классификация или принятие решения. Это позволяет DNN решать более сложные задачи, чем однослойные нейронные сети.

DNN могут обрабатывать большие объёмы данных, что делает их подходящими для задач, требующих глубокого понимания данных. При этом, они могут обучаться на сложных данных, таких как изображения или аудиозаписи.

Недостатком является сложное и длительное обучение, поскольку каждый слой должен быть обучен отдельно, и это может потребовать больше времени и ресурсов. Ещё DNN могут быть подвержены переобучению, то есть они могут запоминать данные, на которых они были обучены, вместо того чтобы обобщать их. Чтобы избежать переобучения, необходимо использовать методы регуляризации, L1 или L2, так же как и для МН.

3 Обучение модели нейронной сети

3.1 Выбор датасета

Для того чтобы обучить нейронную сеть, необходимо предоставить ей данные, на которых она будет учиться. Эти данные должны быть правильно размечены, то есть должны содержать информацию о том, что именно представляет каждый элемент данных. Например, если мы хотим научить нейронную сеть распознавать текст, то нам нужно будет разметить каждое текстовое сообщение, указав его тему или содержание.

Разметка текста может быть выполнена вручную или автоматически. Ручная разметка требует больших затрат времени и усилий, но она позволяет получить более точные результаты. Автоматическая разметка выполняется с помощью алгоритмов, которые могут быть обучены на уже размеченных данных. Однако автоматическая разметка может привести к ошибкам, поэтому её результаты необходимо проверять вручную.

После того как данные были размечены, необходимо выбрать подходящий набор данных для «скармливания» его нейросети. Этот набор данных должен быть достаточно большим, чтобы нейронная сеть могла научиться обобщать полученные знания. Он также должен быть разнообразным, чтобы нейронная сеть не стала слишком специализированной для конкретных условий.

При выборе набора данных необходимо учитывать следующие факторы:

- размер набора данных. Чем больше набор данных, тем лучше нейронная сеть сможет научиться обобщать знания. Однако большие наборы данных требуют больше вычислительных ресурсов для обработки,
- разнообразие набора данных. Разнообразный набор данных позволит нейронной сети научиться работать с различными условиями. Это особенно важно для задач, требующих адаптации к изменяющимся условиям,
- качество набора данных. Качество набора данных влияет на точность

и надёжность работы нейронной сети. Плохо размеченные или искажённые данные могут привести к неправильному обучению нейронной сети.

Выбор подходящего набора данных является важным этапом в процессе обучения нейронных сетей. От него зависит, насколько хорошо нейронная сеть будет выполнять свою задачу. В ходе выполнения данной работы на paperswithcode.com был найден набор данных, подходящий для удовлетворения задач данной магистерской работы.

3.2 Описание датасета

Датасет, предоставленный на рисунке 10, содержит следующие классы:

- Joy (Радость),
- Sadness (Печаль),
- Fear (Страх),
- Anger (Гнев),
- Surprise (Удивление),
- Shame (Стыд),
- Disgust (Отвращение),
- Neutral (Нейтрально окрашенные сообщения).

Всего в датасете 2 колонки:

- emotion – описывает эмоциональную окраску сообщения,
- text – само сообщение со всем знаками препинания и эмодзи.

Затем необходимо будет добавить ещё один столбец без ненужных для обучения нейросети данных.

```
# Preview
df.head()
```

	Emotion	Text
0	neutral	Why ?
1	joy	Sage Act upgrade on my to do list for tommorow.
2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...
3	joy	Such an eye ! The true hazel eye-and so brill...
4	joy	@lluvmiasantos ugh babe.. hugggz for u .! b...

Рисунок 10 – набор данных для обучения нейросети

3.3 Процесс обучения

3.3.1 Анализ данных

Данный набор данных содержит 34792 записи и в нём отсутствуют пробелы, что представлено на рисунке 11. Такого количества записей должно быть достаточно для приемлемого обучения модели. Рассмотрим распределения классов в датасете.

```
# Shape
df.shape

(34792, 2)

# Check For Missing Values
df.isnull().sum()

Emotion    0
Text       0
dtype: int64
```

Рисунок 11 – Количество записей в датасете

На рисунке 12 представлено распределение по классом в датасете.

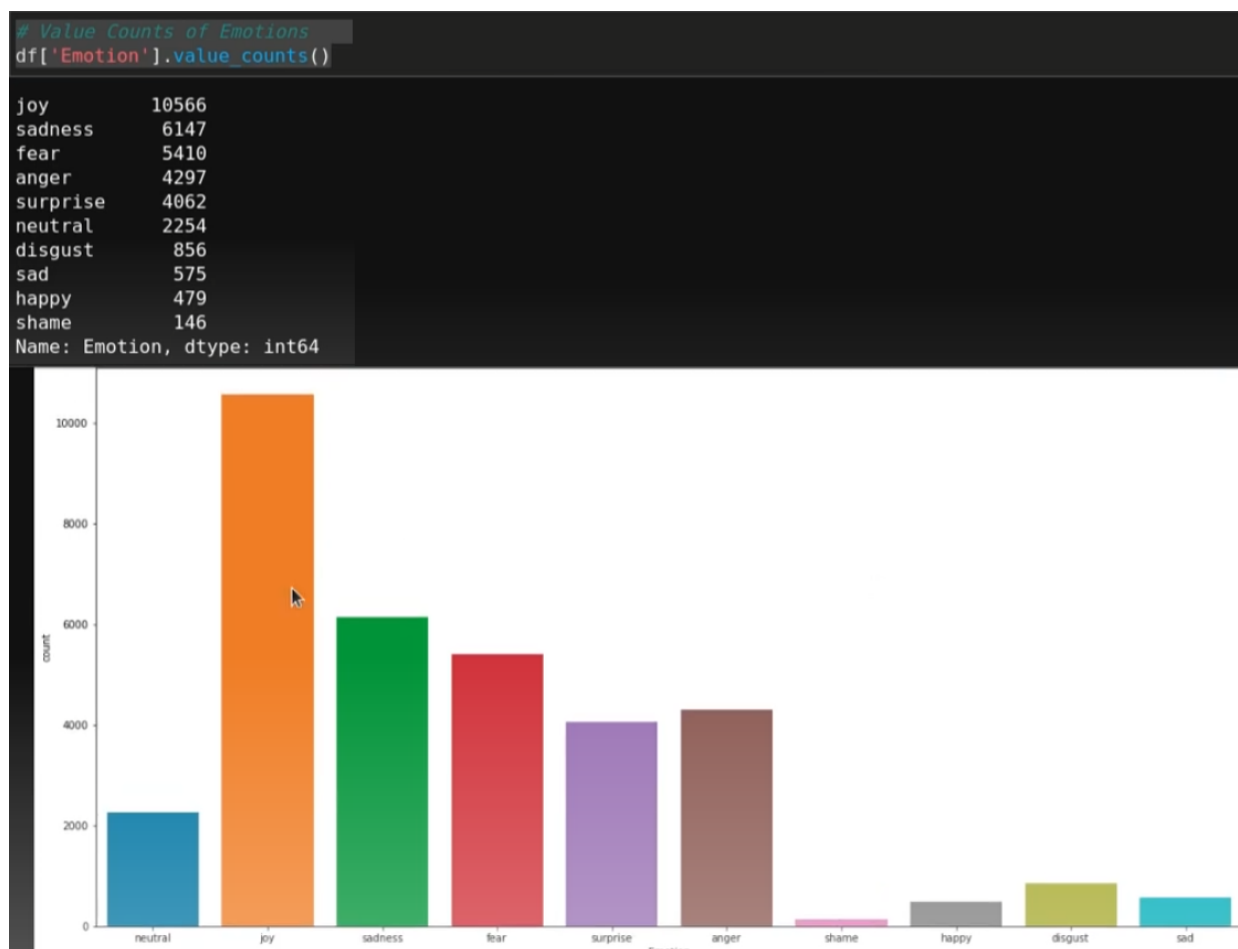


Рисунок 12 – Распределение классов в датасете

Из распределения видно, что классы распределены не равномерно, что скажется на точности нейросети.

3.3.2 Предобработка данных

Для увеличения точности работы модели необходимо предобработать данные, убрав бесполезную информацию из колонки Text. Для этого была использована библиотека `neattext`, которая содержит различные функции для обработки и очистки текстовых наборов данных. Код и результат предобработки представлены на рисунках 13, 14 соответственно.

```

# Data Cleaning
dir(nfx)

Показать скрытые выходные данные

[ ] # User handles
df['Clean_Text'] = df['Text'].apply(nfx.remove_userhandles)

[ ] # Stopwords
df['Clean_Text'] = df['Clean_Text'].apply(nfx.remove_stopwords)

```

Рисунок 13 – Очистка данных с помощью nfx

	Emotion	Text	Clean_Text
0	neutral	Why ?	?
1	joy	Sage Act upgrade on my to do list for tomorrow.	Sage Act upgrade list tomorrow.
2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...	WAY HOMEGIRL BABY FUNERAL!!! MAN HATE FUNERALS...
3	joy	Such an eye ! The true hazel eye-and so brill...	eye ! true hazel eye-and brilliant ! Regular f...
4	joy	@llovviasantos ugh babe.. hugggz for u .! b...	ugh babe.. hugggz u .! babe naamazd nga ako...
...	...	...	...
34787	surprise	@MichelGW have you gift! Hope you like it! It'...	gift! Hope like it! hand wear ! It'll warm! Lol
34788	joy	The world didnt give it to me..so the world MO...	world didnt me..so world DEFINITELY cnt away!!!
34789	anger	A man robbed me today .	man robbed today .
34790	fear	Youu call it JEALOUSY, I call it of #Losing YO...	Youu JEALOUSY, #Losing YOU...
34791	sadness	I think about you baby, and I dream about you ...	think baby, dream time

34792 rows x 3 columns

Рисунок 14 – Результат очистки данных

Теперь в датасете появился ещё один столбец, содержащий только необходимый для обучения нейросети текст. Наконец-то можно приступить к обучению модели.

3.3.3 Обучение модели

Для обучения модели разобьём данные на тестовые и обучающие с помощью функции `train_test_split` из `scikit-learn`. Параметр `test_size` установлен в 0,3, что означает, что 30% данных будут использованы для тестирования, а оставшиеся 70% - для обучения. Параметр `random_state` установлен в 42 для обеспечения воспроизводимости. Результат представлен на рисунке 15.

```
[ ] # Features & Labels
Xfeatures = df['Clean_Text']
ylabels = df['Emotion']

[ ] # Split Data
x_train,x_test,y_train,y_test = train_test_split(Xfeatures,ylabels,test_size=0.3,random_state=42)
```

Рисунок 15 – код разбиения данных на тестовые и обучающие

Затем мы создаём пайплайн и с помощью метода логистической регрессии обучаем нашу модель. Полученная точность модели представлена на рисунке 16

```
# Build Pipeline
from sklearn.pipeline import Pipeline

[ ] # LogisticRegression Pipeline
pipe_lr = Pipeline(steps=[('cv',CountVectorizer()),('lr',LogisticRegression())])

[ ] # Train and Fit Data
pipe_lr.fit(x_train,y_train)

Показать скрытые выходные данные

[ ] pipe_lr
Pipeline(steps=[('cv', CountVectorizer()), ('lr', LogisticRegression())])

# Check Accuracy
pipe_lr.score(x_test,y_test)

0.7400421536692853
```

Рисунок 16 – Результат обучения модели

По результатам проверки, точность модели составляет 74%, что весьма

неплохо, учитывая неравномерно представленные классы в наборе данных.

Проверяем работу модели и сохраняем её для дальнейшего развёртывания на сервере на рисунке 17.

```
[ ] # Make A Prediction
ex1 = "This book was so interesting it made me happy"

[ ] pipe_lr.predict([ex1])

array(['joy'], dtype=object)

[ ] # Save Model & Pipeline
import joblib
pipeline_file = open("emotion_classifier_pipe_lr_03_june_2021.pkl", "wb")
joblib.dump(pipe_lr, pipeline_file)
pipeline_file.close()
```

Рисунок 17 – проверка работы модели и сохранение её в файл

4 Результаты реализации приложения

4.1 Описание архитектуры приложения

Общая архитектура разработанного приложения представлена на рисунке 18. В данной работе рассматривается только нейросеть анализа настроения и веб-версия приложения.

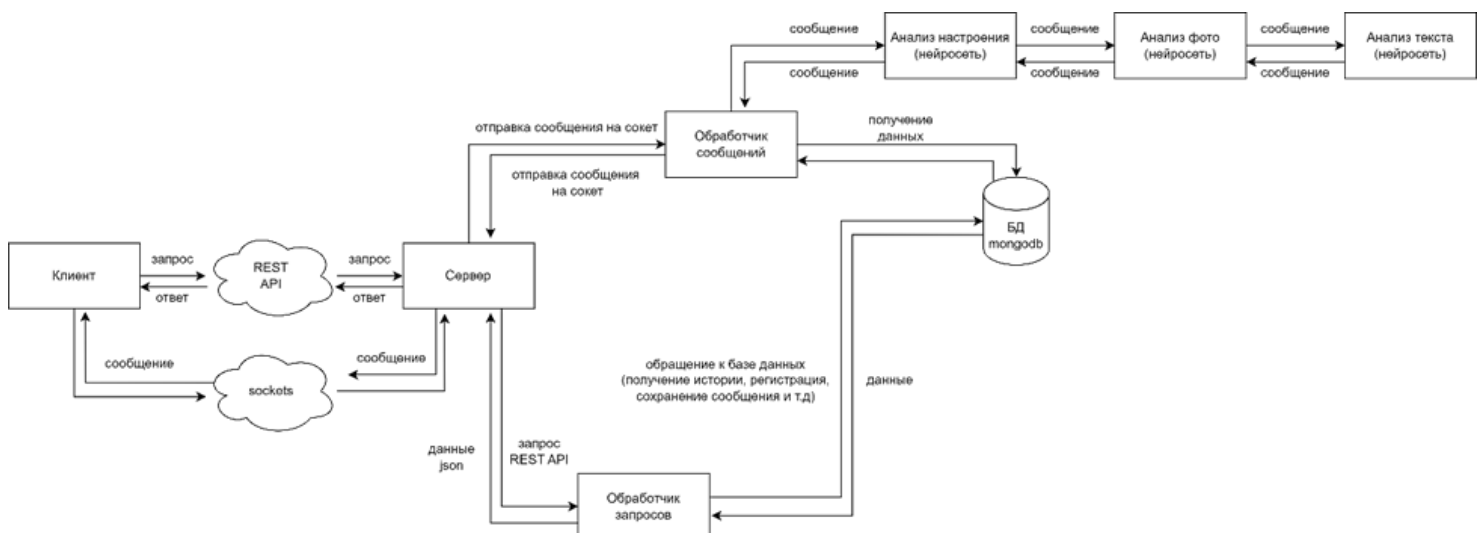


Рисунок 18 – UML диаграмма разработанного приложения

4.2 Реализация веб-версии приложения

4.2.1 Реализация авторизации

В ходе выполнения работ был реализован интерфейс авторизации в веб-приложении с использованием React и TypeScript, представленный на рисунке 19.

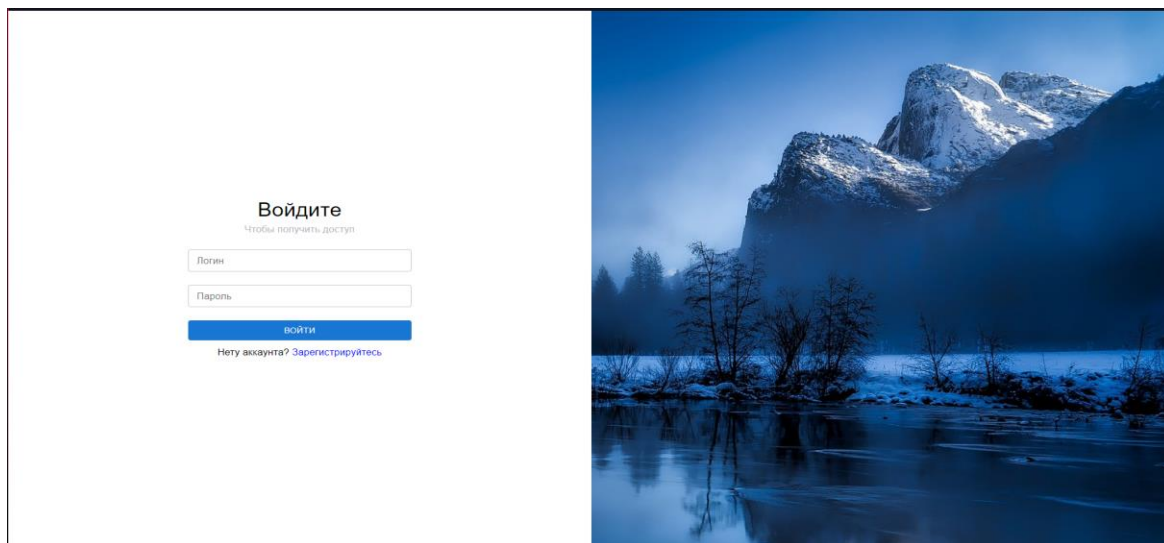


Рисунок 19 – интерфейс авторизации в вебе

Так же была реализована валидация, представленная на рисунке 20 и 21, для данных, который вводит пользователь.

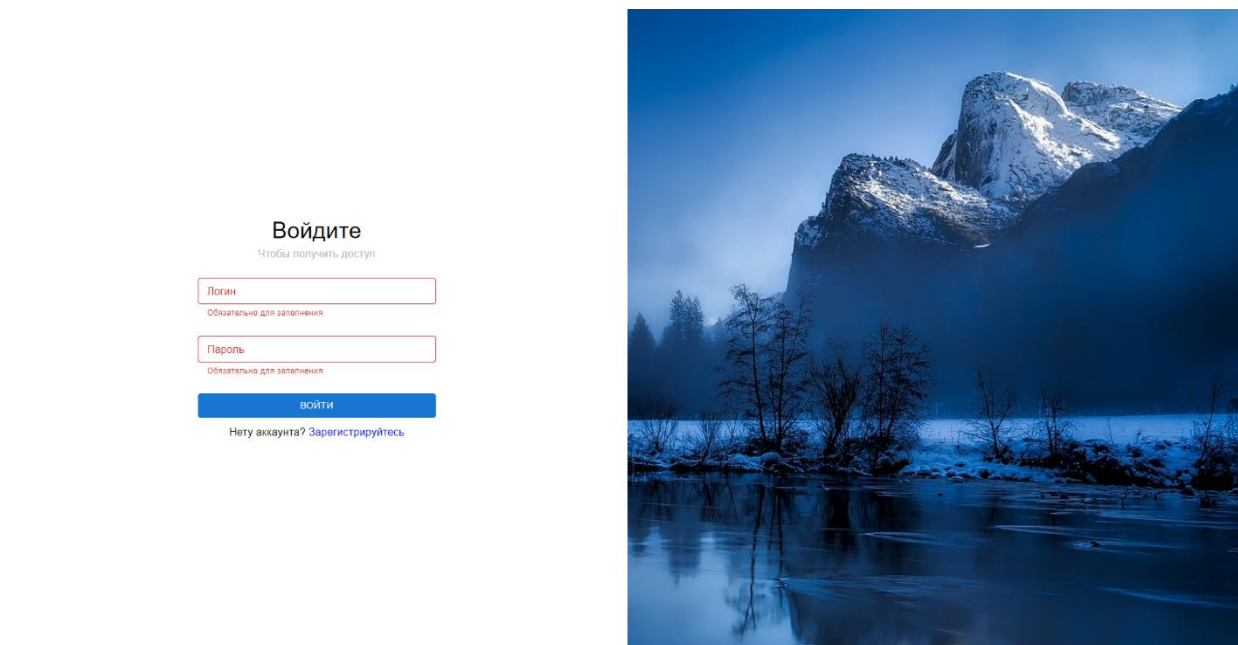


Рисунок 20 – интерфейс авторизации в вебе

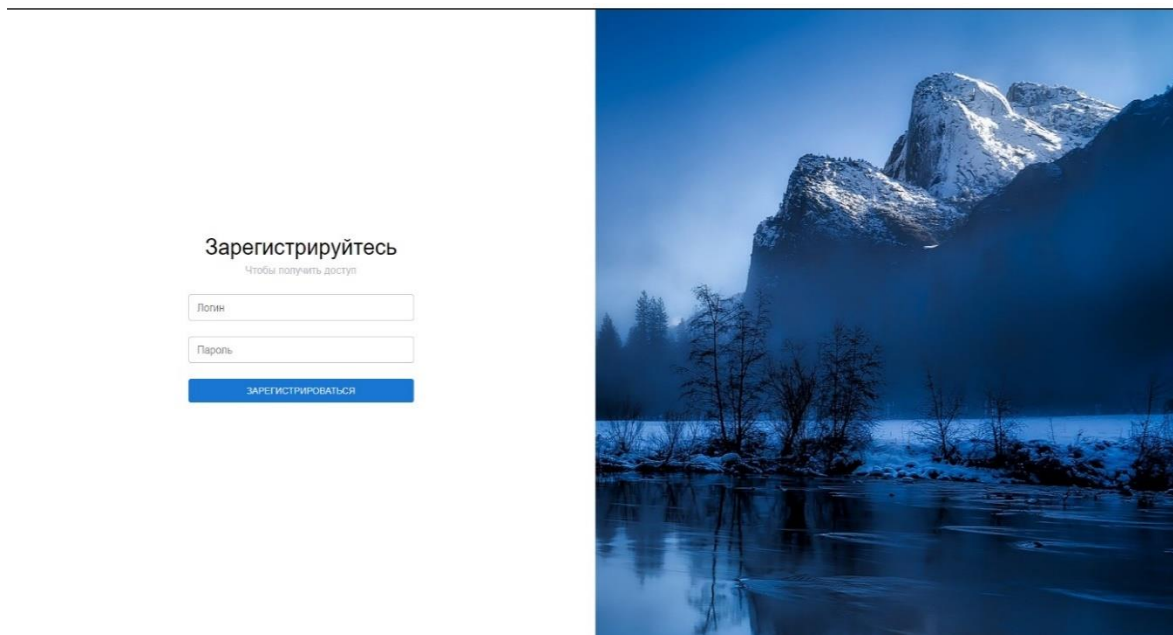


Рисунок 21 – интерфейс регистрации в вебе

```
const REQUIRED_FIELD = 'Обязательно для заполнения';

2 usages  Ilya Repin
export const loginValidation = {
  required: REQUIRED_FIELD,
  validate: (value: string) => {
    if(value.match(matcher: /[а-яА-Я]/)) {
      return 'Логин не может содержать русские буквы'
    }

    return true;
  }
};

2 usages  Ilya Repin
export const passwordValidation = {
  required: REQUIRED_FIELD,
  validate: (value: string) => {
    if(value.length < 6) {
      return 'Пароль должен длинее 6-ти символов'
    }

    return true;
  }
};
```

Рисунок 22 – реализация валидации в коде

Для того чтобы лучше понять, как работает процесс авторизации с использованием токенов, можно представить себя в роли сотрудника склада, который получает посылки из постамата.

Для получения посылок необходимо каждый раз предъявлять пропуск, подтверждающий статус сотрудника склада. Пропуски выдаются системой после регистрации пользователя. Система генерирует два пропуска: зелёный и красный. Запросы сваггера представлены на рисунках 23, 24.

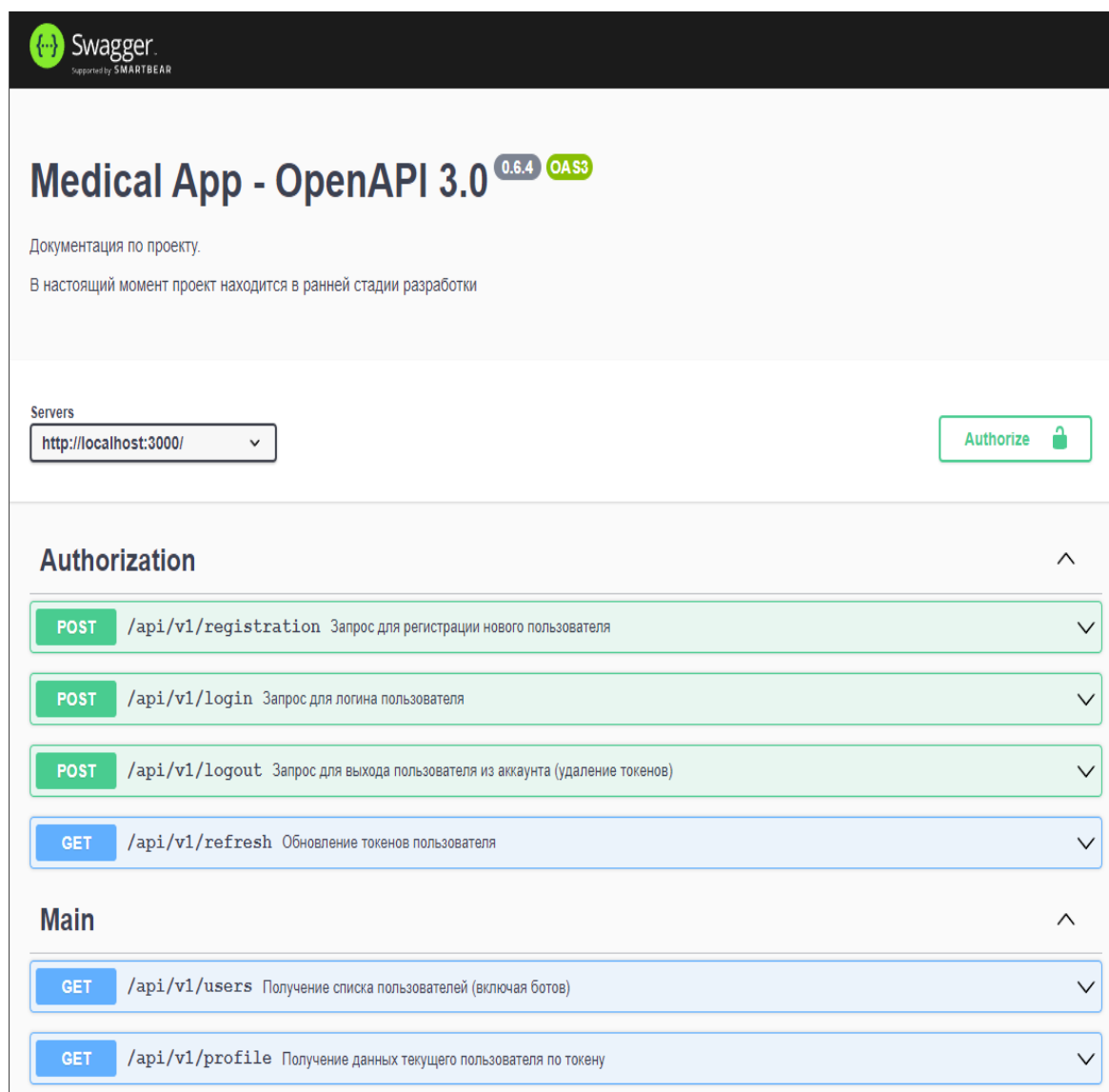


Рисунок 23 – запросы в swagger

Main			^
GET	/api/v1/users	Получение списка пользователей (включая ботов)	v
GET	/api/v1/profile	Получение данных текущего пользователя по токenu	v
GET	/api/v1/chat/:id/messages	Список сообщений в чате	v
Chat			^
GET	/api/v1/chats	Список чатов пользователя	v
POST	/api/v1/private-chat	Список чатов пользователя	v
POST	/api/v1/group-chat	Создать группу	v
GET	/api/v1/chat/:id	Детали чата по идентификатору	v
DELETE	/api/v1/chat/:id	Удаление чата по идентификатору (если пользователь имеет на это право)	v
Utility			^
GET	/api/v1/health	Запрос для проверки состояния сервера	v

Рисунок 24 – запросы в swagger

При получении посылки необходимо использовать зелёный пропуск. Срок действия зелёного пропуска ограничен, он короче, чем у красного пропуска. Когда срок действия зелёного пропуска истекает, получить посылку становится невозможно, так как система не может идентифицировать пользователя. В этом случае необходимо воспользоваться красным пропуском, который обновляет действие обоих пропусков.

В контексте IT-терминологии, зелёный пропуск представляет собой access token, а красный — refresh token. Постамат в данном случае выступает в роли нашего сервера, мы же являемся клиентом. Посылки представляют собой данные, предоставляемые нашим сервером.

Получение посылки означает выполнение запроса к серверу. Код приведён на рисунке 25, 26.

```
1 usage  a Ilya Repin
const onSubmit: SubmitHandler<ISignInForm> = async (data :ISignInForm ) => {
  try {
    const response = await axios.post<AuthResponse>({ url: [REDACTED]api/v1/login', data: {
      email: data.login,
      password: data.password
    }});

    if (response.status === 200) {
      const { accessToken, refreshToken, user } = response.data;
      // Здесь можно использовать полученные токены и данные пользователя для дальнейшей авторизации
      console.log('Access token:', accessToken);
      console.log('Refresh token:', refreshToken);
      console.log('User:', user);
    }
  } catch (error) {
    console.error('Error:', error);
  }
};
```

Рисунок 25 – реализация авторизации

```
Access token: auth-form.tsx:42
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2NDU3ZDhmOWExNGU5YjEyNWUyMmN1NSIsImVtYWlsIjoieWxlbmFAdGVzdC5jb20iLCJyb2x1cyI6WyJVVU0VSI10sIm1hdCI6MTcxNzQ2NTAyNCwiZXhwIjojoxNzIwMDU3MDI0fQ.n-oXa9BE2o3bUGBNzFFldwKeov9bNrpoMjE3C7AZW7Y

Refresh token: auth-form.tsx:43
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2NDU3ZDhmOWExNGU5YjEyNWUyMmN1NSIsImVtYWlsIjoieWxlbmFAdGVzdC5jb20iLCJyb2x1cyI6WyJVVU0VSI10sIm1hdCI6MTcxNzQ2NTAyNCwiZXhwIjojoxNzIwMDU3MDI0fQ.HUn0syM7F88I47JewPMmxsNt4MY139Vs-90cqbS6bZY

User: auth-form.tsx:44
▶ {id: '66457d8f9a14e9b125e21ce5', name: 'alena', color: '#0aceb4', role: 'USER'}
```

Рисунок 26 – Полученные токены с сервера

Охранники выполняют функцию переадресации пользователя на страницу регистрации или входа в систему. После успешной регистрации или авторизации, клиент получает access и refresh токены.

Эти токены должны быть сохранены на стороне клиента, после чего пользователь перенаправляется на главную страницу приложения с чатами, интерфейс которой представлен на рисунке 27.

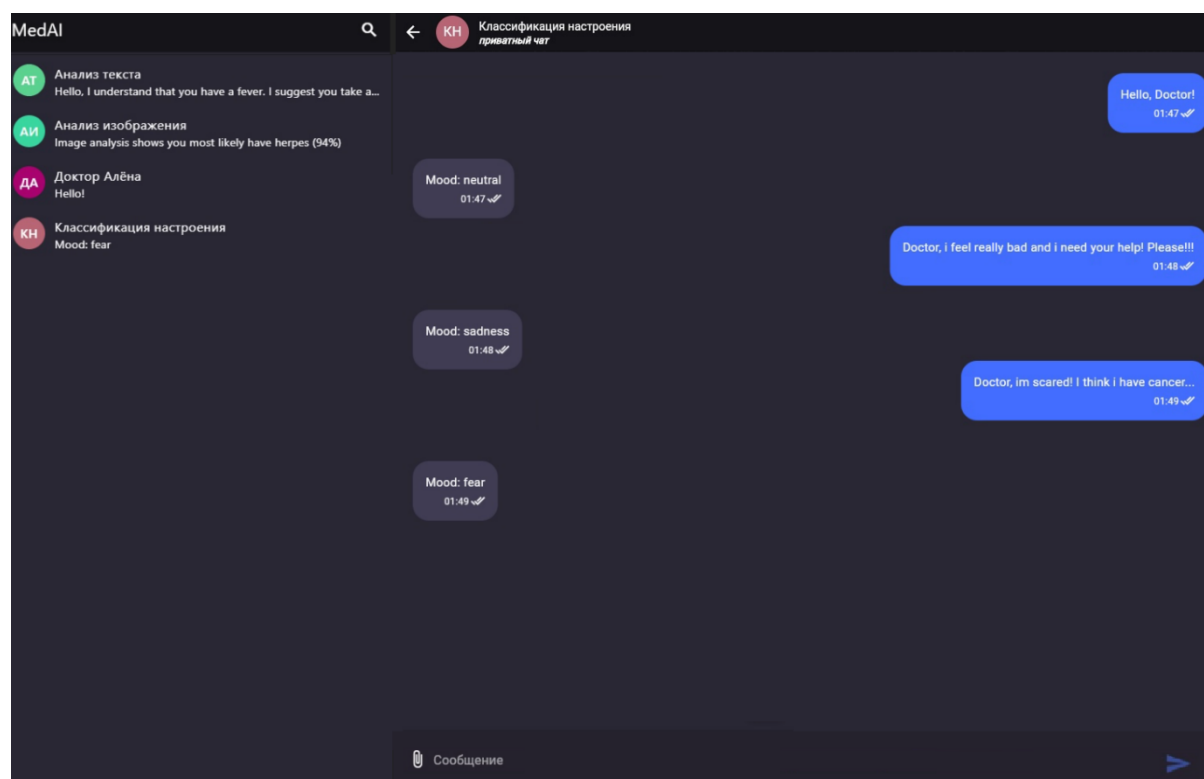


Рисунок 27 – Интерфейс чата веб приложения

Каждый раз при отправке запроса на сервер, необходимо включать access token в заголовки (headers) в виде строки «Bearer \$accessToken». Если срок действия access token истечёт, то при следующем запросе сервер вернёт ошибку 401 Unauthorized.

Необходимо обработать ошибку 401 Unauthorized на стороне клиента. В этом случае следует отправить запрос на обновление токена, используя сохранённый refresh token после регистрации/логина. Если refresh token всё ещё действителен, будет возвращена новая пара токенов, которые необходимо сохранить на клиенте.

Запрос, завершившийся с ошибкой 401, должен быть повторён с новым access token.

Однако, если и refresh token уже не действителен, то запрос на его обновление также завершится ошибкой. В таком случае необходимо удалить все сохранённые данные пользователя и перенаправить его на страницу авторизации/логина.

Кроме того, на основной странице приложения должна быть предусмотрена кнопка «Выйти из аккаунта». При её нажатии отправляется запрос /logout, после чего также удаляются все сохранённые пользовательские данные и происходит перенаправление на страницу регистрации/логина.

Для обновления токена необходим отдельный обработчик. В случае получения кода ошибки 401, следует отправить запрос /refresh на сервер. После этого, в случае успешного обновления, необходимо обновить токены. Если же обновление не удалось, то есть refresh token протух, нужно удалить все сохранённые данные клиента и перенаправить его на страницу регистрации/логина.

4.2.2 Реализация чата в приложении с использованием сокетов и Socket.IO

Сокеты — это технология, которая позволяет двум или более компьютерам обмениваться данными в режиме реального времени.

В контексте чата сокеты используются для обмена сообщениями между пользователями.

Для реализации чата с использованием сокетов необходимо выполнить следующие шаги:

- установить зависимости. Для работы с сокетами необходимо установить Node.js и npm (менеджер пакетов для Node.js). Затем нужно установить библиотеку Socket.IO с помощью команды `npm install socket.io`,
- создать сервер. Сервер будет обрабатывать запросы от клиентов и отправлять им сообщения. Для создания сервера необходимо создать файл `server.js` и импортировать библиотеку Socket.IO. Затем необходимо настроить сервер с помощью функции `io.listen()`,

– обработать события. Необходимо обработать события подключения клиента к серверу, отправки и получения сообщений. Для обработки событий можно использовать функции `io.on()` и `socket.on()`,

– отправить сообщения. При отправке сообщения от клиента на сервер необходимо использовать функцию `socket.emit()`,

– получить сообщения. При получении сообщения от сервера на клиент необходимо использовать функцию `socket.on()`, где `data` — это данные, отправленные сервером.

Код реализованного чата представлен на рисунках 27, 28.

```
import React, { useState, useEffect } from 'react';
import io from 'socket.io-client';

const ChatApp: React.FC = () => {
  const [socket, setSocket] = useState<SocketIOClient.Socket | null>{null};
  const [messages, setMessages] = useState<string[]>([]);
  const [inputValue, setInputValue] = useState('');

  useEffect(() => {
    const newSocket = io('http://localhost:3000');
    setSocket(newSocket);

    newSocket.on('message', (msg: string) => {
      setMessages((prevMessages) => [...prevMessages, msg]);
    });

    return () => {
      newSocket.disconnect();
    };
  }, []);

  const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setInputValue(e.target.value);
  };
};
```

Рисунок 27 – код чата

```

const handleSendMessage = () => {
  if (socket && inputValue.trim() !== '') {
    socket.emit('message', inputValue);
    setInputValue('');
  }
};

return (
  <div>
    <h1>Chat</h1>
    <ul>
      {messages.map((msg, index) => (
        <li key={index}>{msg}</li>
      ))}
    </ul>
    <input
      type="text"
      value={inputValue}
      onChange={handleInputChange}
      placeholder="Type a message..."
    />
    <button onClick={handleSendMessage}>Send</button>
  </div>
);
};

export default ChatApp;

```

Рисунок 28 – код чата

ЗАКЛЮЧЕНИЕ

Результатом данной магистерской диссертации является успешно реализованный модуль для распознавания настроения пользователя по его сообщениям.

В процессе выполнения данной работы были достигнуты следующие цели:

- была разработана авторизация в веб-приложении,
- был разработан чат для взаимодействия с пользователем,
- была обучена модель нейронной сети для распознавания настроения,
- обученная модель нейронной сети успешно определяет настроение пользователя по его сообщениям.

Перед началом реализации модуля были изучены:

- предметная область,
- существующие аналоги,
- инструменты разработки для реализации веб-приложения и обучения модели нейронной сети,
- методы реализации авторизации в приложении,
- существующие типы моделей нейронной сети и выбран наиболее подходящий под цели данной работы.

Благодаря этому была обучена и протестирована модель нейронной сети, успешно реализован модуль для определения настроения пользователя по его сообщениям.

В дальнейшем, данный модуль может быть полезен не только в медицине, но и в других сферах IT-бизнеса. Планируется дальше заниматься развитием проекта, добавить возможность автоматического заполнения ботом документов, возможность редактировать БД внутри бота, а также разработка собственной нейронной сети и собственного датасета по ряду тем, наиболее важных для студентов университетов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вандеркам Дэн В17 Эффективный TypeScript: 62 способа улучшить код. — СПб.: Питер, 2020. ил. — (Серия «Бестселлеры O'Reilly»). — С. 14-28.
2. Гэддис Т. Начинаем программировать на Python. — 4-е изд.: Пер. с нгл. СПб.: БХВ-Петербург, 2019 — С. 20-23.
3. Дэвис С. Р. С# 2005 для «чайников». — Издательство Диалектика, 2008 — С. 53-85.
4. Чурсин Г. С. Встраиваемый чат-бот в готовые корпоративные системы. — 2019 — С. 23-25.
5. Забутной А., Сокол М. Эволюция СУБД Oracle. — 2018 — С. 140-143.
6. Пономарев Д. А. Работа с базами данных при помощи языка программирования Python и СУБД SQLite3 в школьном курсе информатики//Информационные технологии в образовании. — 2020. — №. 3. — С. 187-191.
7. Тимошенко Л. И., Аракелов О. Г. Безопасность труда при работе с персональным компьютером //Культура и общество: история и современность. — 2014. — С. 114-117.

ПРИЛОЖЕНИЕ А

Листинг программы

```

import React from 'react';
import Typography from '@mui/material/Typography';
import TextField from '@mui/material/TextField';
import Button from '@mui/material/Button';
import { useForm, SubmitHandler, Controller, useFormState } from "react-hook-form";
import './auth-form.css';
import { loginValidation, passwordValidation } from './validation';
import axios from 'axios';

```

```

interface ISignInForm {
  login: string;
  password: string;
}

```

```

interface AuthResponse {
  accessToken: string;
  refreshToken: string;
  user: {
    id: string;
    name: string;
    color: string;
    role: string;
  };
}

```

```

export const AuthForm: React.FC = () => {
  const { handleSubmit, control } = useForm<ISignInForm>();
  const { errors } = useFormState({

```

```

        control
    });

    const onSubmit: SubmitHandler<ISignInForm> = async (data) => {
        try {
            const response = await
axios.post<AuthResponse>('http://213.171.10.91:3000/api/v1/login', {
                email: data.login,
                password: data.password
            });

            if (response.status === 200) {
                const { accessToken, refreshToken, user } = response.data;
                // Здесь можно использовать полученные токены и данные
пользователя для дальнейшей авторизации
                console.log('Access token:', accessToken);
                console.log('Refresh token:', refreshToken);
                console.log('User:', user);
            }
        } catch (error) {
            console.error('Error:', error);
        }
    };

    return (
        <div className="auth-form">
            <Typography variant="h4" component="div">
                Войдите
            </Typography>

```

```

    <Typography variant="subtitle1" gutterBottom component="div"
className="auth-form__subtitle">
      Чтобы получить доступ
    </Typography>
    <form className="auth-form__form"
onSubmit={handleSubmit(onSubmit)}>
      <Controller
        control={control}
        name="login"
        rules={loginValidation}
        render={({ field }) => (
          <TextField
            label="Логин"
            onChange={(e) => field.onChange(e)}
            value={field.value}
            fullWidth={true}
            size="small"
            margin="normal"
            className="auth-form__input"
            error={!!errors.login?.message}
            helperText={errors?.login?.message}
          />
        )}
      />
      <Controller
        control={control}
        name="password"
        rules={passwordValidation}
        render={({ field }) => (

```

```

    <TextField
      label="Пароль"
      onChange={(e) => field.onChange(e)}
      value={field.value}
      fullWidth={true}
      size="small"
      margin="normal"
      type="password"
      className="auth-form__input"
      error={!errors.password?.message}
      helperText={errors?.password?.message}
    />
  )}
/>
<Button
  type="submit"
  variant="contained"
  fullWidth={true}
  disableElevation={true}
  sx={{
    marginTop: 2
  }}
>
  Войти
</Button>
</form>

<div className="auth-form__footer">
  <Typography variant="subtitle1" component="span">

```

```
        Нету аккаунта?{" "}
      </Typography>
      <Typography variant="subtitle1" component="span" sx={{ color: 'blue'
    }}>
        Зарегистрируйтесь
      </Typography>
    </div>
  </div>
);
};
```

ПРИЛОЖЕНИЕ Б

Презентация

ТЕМА РАБОТЫ : ЧАТ-БОТ ДЛЯ АНАЛИЗА НАСТРОЕНИЯ ПОЛЬЗОВАТЕЛЯ И ОКАЗАНИЯ ЭМОЦИОНАЛЬНОЙ ПОДДЕРЖКИ

1

Научный руководитель: д.т.н. Коннов Н.Н.
Выполнил: студент гр. 22ВВиМ1 Репин Илья

ЦЕЛЬ И ОБЪЕКТ РАБОТЫ

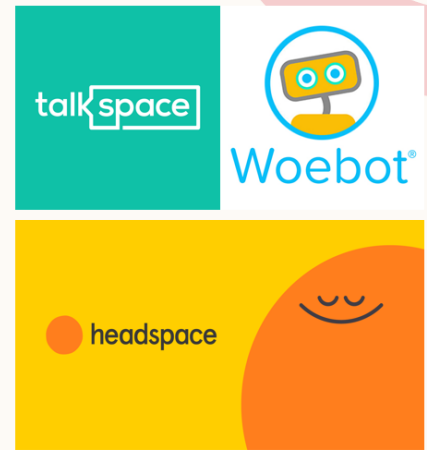
2

Разработать модуль приложения чат-бота для определения настроения пользователя по его сообщениям.

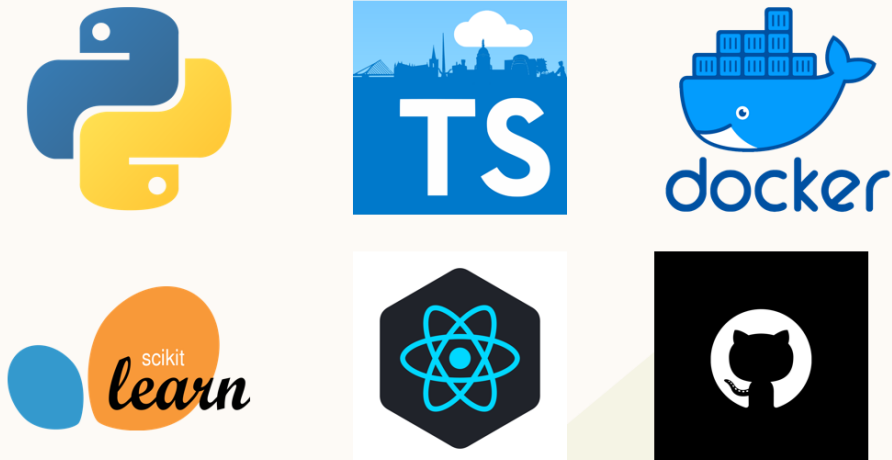
Приложение реализует сбор данных в процессе общения с пользователем, затем с помощью обученной нейросети определяет его настроение и в зависимости от настроения формирует ответ.

АНАЛИЗ АНАЛОГОВ

Достоинства	Недостатки
Использование ИИ для анализа поведения пользователя и предоставления персонализированных рекомендаций.	Ограниченность в выборе тем консультаций
Возможность отслеживания прогресса и оценки эффективности терапии.	Отсутствие возможности прямого общения с психологом.

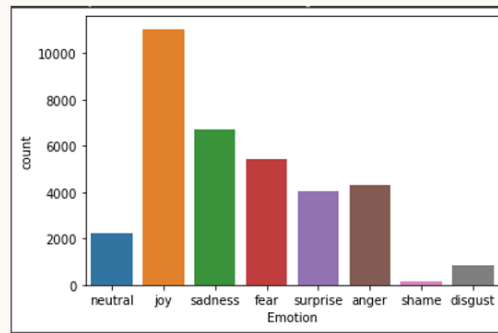


ИНСТРУМЕНТЫ РАЗРАБОТКИ



ОБУЧЕНИЕ НЕЙРОСЕТИ ДЛЯ РАСПОЗНАВАНИЯ НАСТРОЕНИЯ

5



```
[ ] # Make A Prediction  
ex1 = "This book was so interesting it made me happy"  
  
[ ] pipe_lr.predict([ex1])  
  
array(['joy'], dtype=object)
```

```
# Check Accuracy  
pipe_lr.score(x_test,y_test)  
  
0.7400421536692853
```

РЕАЛИЗАЦИЯ АВТОРИЗАЦИИ ВЕБ-ВЕРСИИ

6

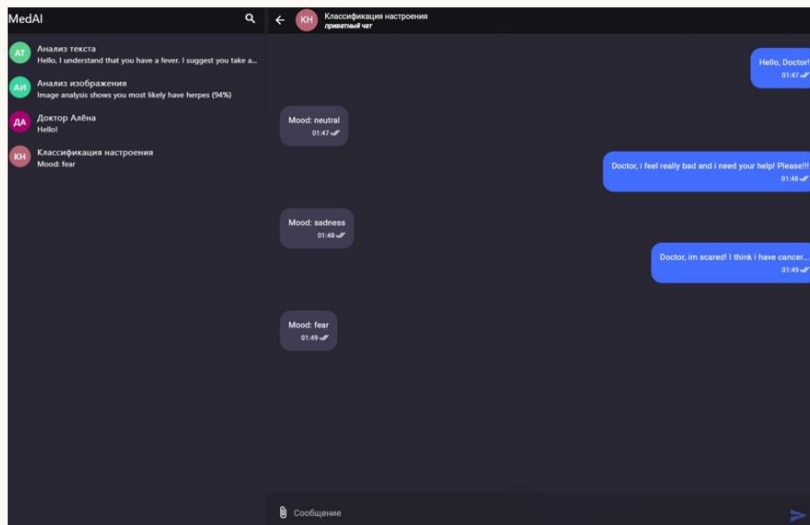
Войдите
Пользовательское имя или ID

[Нужно восстановить? Забыли пароль?](#)

Войдите
Пользовательское имя или ID

[Нужно восстановить? Забыли пароль?](#)

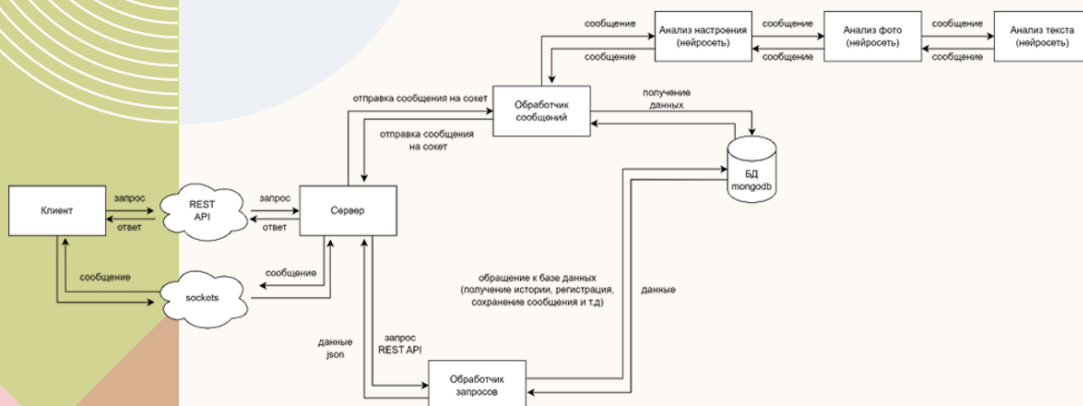
РЕАЛИЗАЦИЯ ЧАТА В ВЕБ-ВЕРСИИ



7

АРХИТЕКТУРА РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

8



ЗАКЛЮЧЕНИЕ

- Были проанализированы существующие аналоги и инструменты разработки с целью выпустить наилучший продукт
- На основе анализа был разработан модуль приложения чат-бота для определения настроения пользователя по его сообщениям.
- Приложение полностью соответствует поставленной цели

9

СПАСИБО ЗА ВНИМАНИЕ!

10