

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»

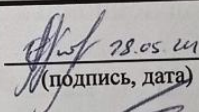
Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

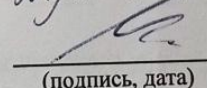
ЧАТ-БОТ ДЛЯ ИНФОРМАЦИОННОЙ
ПОДДЕРЖКИ ПРЕПОДАВАНИЯ
ДИСЦИПЛИНЫ

Студент


(подпись, дата)

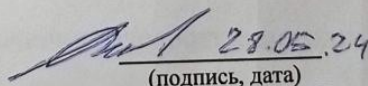
Кочеткова Ольга Анатольевна
(ФИО полностью)

Руководитель


(подпись, дата)

Митрохин М.А.
(фамилия, инициалы)

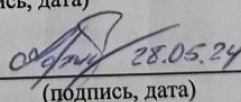
Нормоконтролёр


(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

Рецензент

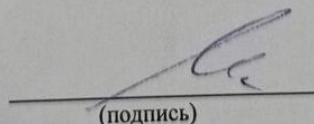
науч. сотрудник
ООО «ИТ-Групп», КриптоСофт»
(должность, место работы)


(подпись, дата)

Короткий А.А.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 № 11)

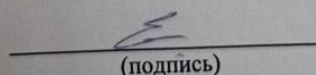
Заведующий кафедрой


(подпись)

Митрохин М.А.
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 6.06.24 № 1)

Секретарь ГЭК


(подпись)

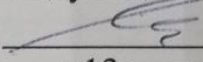
Гурин Е.И.
(фамилия, инициалы)

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 Кочеткова Ольга Анатольевна гр. 22ВВИм2 факультета ВТ
2. Тема работы «Чат-бот для информационной поддержки преподавания дисциплины»
Тема утверждена приказом ПГУ № 765/о от "13" сентября 2022 г.
3. Руководитель магистерской диссертации д.т.н., доцент, зав.кафедрой Митрохин М.А.
4. Задание на работу (назначение разработки, исходные данные и т.п.)

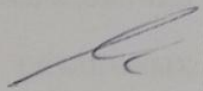
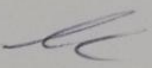
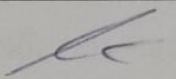
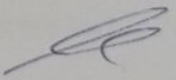
Назначение разработки: применение чат-бота в образовательном процессе высшего учебного заведения для информационной поддержки преподаваемой дисциплины.

Проектирование и разработка чат-бота для информационной поддержки дисциплины на основе технологий искусственного интеллекта.

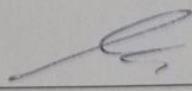
5. Перечень подлежащих разработке вопросов:

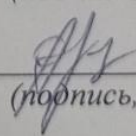
1. Теоретический анализ предметной области использования чат-ботов в образовательном процессе и определить функциональные требования к ним
2. Обзор и сравнительный анализ современных средств разработки чат-ботов
3. Разработка чат-бота на основе машинного обучения и его тестирование

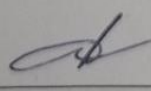
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Теоретический анализ предметной области	10	16.09.2022 -25.12.2022	
2.	Обзор и сравнительный анализ средств разработки чат-ботов	10	26.12.2022 -12.02.2023	
3.	Интеллектуальный анализ обработки естественного языка	20	13.02.2023 -14.05.2023	
4.	Реализация алгоритмов машинного обучения для автоматизированной обработки обращений пользователя и их анализ	25	15.05.2023- 10.12.2023	
5.	Разработка программного продукта	25	11.12.2023- 03.03.2024	
6.	Оформление выпускной работы	10	04.03.2024- 27.05.2024	

Дата выдачи задания "13" сентября 2022 г.

Руководитель магистерской диссертации  Митрохин М.А.
(подпись, дата)

Задание к исполнению принял студент  Кочеткова О.А.
(подпись, дата)

Работу к защите допустить
Декан факультета  30.05.2024 Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

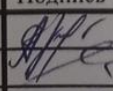
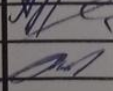
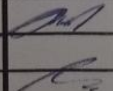
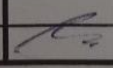
Работа содержит 102 л., 45 рис., 1 табл., 30 источн., 2 прил.

ЧАТ-БОТ, ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА, МОДЕЛИ
МАШИННОГО ОБУЧЕНИЯ, МЕТРИКИ В МАШИННОМ ОБУЧЕНИИ,
PYTHON, TELEGRAM

Целью ВКР является разработка приложения информационной поддержки дисциплины (на примере «Современные информационные технологии») на основе интеллектуального чат-бота. Объектом исследования является машинная обработка текста на основе использования различных методов и способов анализа фраз и предложений.

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 для информационной поддержки дисциплины на основе технологий искусственного интеллекта. Осуществлено функциональное тестирование разработанного программного продукта, которое показало корректность его работы.

					ПГУ 1.090401.34.001 ПЗ			
		№ докум.	Подпись	Дата				
Разраб.	Кочеткова О.А.				Чат-бот для информационной поддержки преподавания дисциплины	Лит.	Лист	Листов
Провер.	Митрохин М.А.						2	102
Н. Контр	Бычков А.С.					ФВТ, каф. ВТ гр. 22ВВИм2		
Утверд.	Митрохин М.А.							
					Пояснительная записка			

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	5
ВВЕДЕНИЕ	6
1. Теоретический анализ предметной области	9
1.1 Общие понятия чат-бота	9
1.2 Анализ эффективности использования чат-бота в образовательном процессе	14
1.3 Определение функциональных требований к работе образовательных чат-ботов	17
1.4 Выводы	19
2 Выбор программного обеспечения	20
2.1 Обзор популярных платформ для создания чат-ботов	20
2.2 Обзор и сравнительный анализ средств разработки	22
2.3 Анализ выбора базы данных для хранения сведений о пользователях .	31
2.4 Выводы	36
3 Разработка программного продукта	37
3.1 Регистрация чат-бота	38
3.2 Административная панель управления	42
3.3 Проектирование и реализация базы данных чат-бота	46
3.4 Модели машинного обучения для автоматизированной обработки обращений пользователя	48
3.5 Результаты разработки чат-бота	63
3.6 Выводы	70
ЗАКЛЮЧЕНИЕ	72
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	74
ПРИЛОЖЕНИЕ А Листинг исходного кода	78
ПРИЛОЖЕНИЕ Б Презентация	96

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ИИ – искусственный интеллект

ПИ – пользовательский интерфейс

ПО – программное обеспечение

СИТ – современные информационные технологии

AI (Artificial intelligence) – искусственный интеллект

API (Application Programming Interface) – интерфейс прикладного программирования

ID (Identifier) – идентификатор

JSON (JavaScript object notation) – обозначение объекта JavaScript

ML (Machine learning) – машинное обучение

SQL (Structured Query Language) – язык структурированных запросов

ВВЕДЕНИЕ

С развитием цифровой экономики, скорость получения информации с помощью современных средств интернет-коммуникаций быстро возросла. Цифровая трансформация образовательной среды формирует основы для качественных преобразований современных программных инструментов с целью повышения качества образования на разных уровнях обучения. Одним из средств улучшения качества взаимодействия с пользователями может выступать чат-бот. В настоящее время в качестве приоритетных направлений для научных исследований является создание разговорного искусственного интеллекта, способного поддерживать человеко-машинный разговор посредством текстового или голосового взаимодействия [1].

Чат-боты находят применение в самых разных областях, выполняя разнообразные задачи: они способствуют упрощению повторяющихся бизнес-процессов, помогают в изучении иностранных языков, занимаются набором персонала, обеспечивают техническую поддержку на мероприятиях и в компаниях, а также используются и в образовательной сфере. Во многих работах обосновывается большой потенциал использования чат-ботов для улучшения процесса обучения и результатов обучения [2, 3]. Чат-боты становятся все более популярными в образовательной сфере благодаря своей универсальности и многофункциональности. Они могут быть применены в качестве надежных помощников преподавателей не только для решения образовательных задач, но и для организации воспитательной работы [4].

Кроме того, чат-боты способны поддерживать изучении дисциплины в образовательной среде, напоминая обучающимся о важности выполнения заданий и соблюдения расписания. Они могут напоминать о предстоящих занятиях, контрольных работах и других мероприятиях, что способствует более эффективной организации учебного процесса.

В самом общем смысле чат-бот (от англ. chat – болтать, bot – робот) или виртуальный собеседник – программа-алгоритм, имитирующая общение с пользователем на естественном языке через цифровое устройство [5].

Таким образом, это программа, которая интерактивно имитирует человеческую речь и позволяет общаться с цифровыми устройствами как с людьми. Чат-боты могут быть простой программой, отвечающей на простой вопрос. Некоторые боты имеют достаточно сложную систему: обучающиеся и развивающиеся по мере накопления информации. При разработке современных чат-ботов активно применяется искусственный интеллект [6].

Таким образом, актуальность разработки чат-бота для преподавания учебной дисциплины обусловлено необходимостью оптимизировать работу преподавателей высшей школы, вынужденных затрачивать большое количество времени для ответов на частые вопросы студентов, что способствует более эффективной организации учебного процесса.

Целью проекта является разработка приложения информационной поддержки дисциплины (на примере «Современные информационные технологии») на основе интеллектуального чат-бота.

Для достижения поставленной цели необходимо решить следующие задачи:

- изучить различные подходы к понятию «чат-бот», выделить основные его функции, провести анализ эффективности использования ботов в образовательном процессе,
- провести обзор различных программных средств для разработки чат-бота и обосновать выбор для разработки программного продукта,
- разработать структуру бота для информационной поддержки выбранной учебной дисциплины,
- изучить модели машинного обучения для обработки запросов пользователей в чат-боте,
- продемонстрировать концепцию разработки и реализацию чат-бота для использования в образовательном процессе.

Научная новизна исследования заключается в том, что на основе изучения обширного количества программных инструментов для осуществления проекта, анализа эффективности использования чат-ботов в

сфере образования, разработан оригинальный чат-бот для оказания информационной поддержки студентов первого курса ПИ им. В.Г. Белинского ПГУ «Современные информационные технологии».

Практическая значимость заключается в том, что данный чат-бот может быть применен на занятиях очной и заочной формы обучения, при самостоятельной работе обучающихся, в дистанционном образовании.

Для практических исследований использована среда разработки PyCharm Community и язык Python 3.10.

Настоящая магистерская диссертация состоит из введения, трёх разделов, заключения, списка использованных источников и приложений.

Первый раздел содержит анализ предметной области и постановка задачи исследования.

Во втором разделе представлен анализ инструментов разработки чат-ботов.

Третий раздел посвящен описанию работы разработанного кода.

1 Теоретический анализ предметной области

1.1 Общие понятия чат-бота

На сегодняшний день одним из перспективных направлений в области информационных технологий является разработка разговорных агентов. Одним из таких методов является использование чат-ботов. Они нашли широкое применение в различных сферах деятельности благодаря своей универсальности и доступности. Взаимодействие с чат-ботами не требует от пользователя особых навыков или специальных знаний, оно интуитивно понятно и практически неотличимо от общения с живым человеком.

В различных источниках даются следующие определения чат-ботов [3, 7, 8]:

- разговорные агенты или диалоговые системы – это компьютерные системы, с которыми пользователь взаимодействует на естественном языке,
- чат-бот – компьютерная программа, которая имитирует реальный разговор с помощью слуховых или текстовых методов, помогающий решить задачи,
- под чат-ботами (в общем смысле) понимают специальные программы, реализующие интернет-общение, обычно с одним или несколькими пользователями, на основе искусственного интеллекта.

Таким образом, чат-боты представляют собой программу, получая информацию от пользователя, осуществляет вывод корректных и логически обоснованных ответов на основе технологии ИИ.

Первая теоретическая разработка ИИ была реализована еще в 40-ых годах XX-ого века, но переход на качественно новый уровень произошел после 2010 года, благодаря большому числу разработкам в таких областях ИИ, как нейронные сети и машинное обучение. Благодаря ИИ и при помощи современных суперкомпьютеров могут быть успешно решены алгоритмические, математические, сверхсложные программные задачи и оптимизация управления различными процессами.

В научной литературе имеется определение ИИ как способности автоматизированных вычислительных систем выполнять интеллектуальные функции, аналогичные тем, что выполняет человек [9, 10].

Искусственный интеллект (в узком смысле) представляет собой программу для компьютера, которая эмулирует поведение человека, изучая различные модели данных и процессы. Среди основных функций ПО ИИ можно выделить: машинное обучение, распознавание речи и голоса, виртуального помощника и т.д. [9].

В данной работе под ИИ будем понимать комплекс технологических решений для автоматизированных вычислительных машин, позволяющий имитировать когнитивные функции человека (включая самообучение и поиск решений без заранее заданного алгоритма) и получать достоверные, логически обоснованные результаты.

В настоящее понятие «искусственный интеллект» активно используется при разработке «умных» чат-ботов. Лидирующие позиции по использованию чат-ботов уверенно удерживают различные мессенджеры, в том числе и Telegram.

Далее рассмотрим общую классификацию чат-ботов:

- по алгоритму,
- виду,
- функциональности.

На рисунке 1 представлена общая классификация ботов.



Рисунок 1 – Классификация чат-ботов

Рассмотрим каждый из видов образовательных виртуальных помощников подробнее.

По алгоритму:

- ограниченный бот. Его функционирование основано на заранее разработанном алгоритме, содержащем ответы на самые часто задаваемые вопросы. При взаимодействии с ним пользователь получает возможность вести диалог, который кажется настолько естественным, будто он общается с обычным человеком,

- саморазвивающийся бот основан на использовании нейронной сети. При наличии достаточного количества обучающих примеров он способен имитировать человеческое общение. Путь разговора с ним формируется неявно, на основе анализа полученных данных, которые затем используются для обучения.

По внешнему виду:

- кнопочный (диалог строиться на основе выбора «кнопки» меню. Благодаря большому количеству кнопок, пользователь приходит к определенному пути),

– текстовый (распознает «ключевые слова» в запросе слова, используется искусственный интеллект для определения намерений пользователя, формируется запрос в базу данных и предлагает ответ пользователю).

По функционалу:

– коммуникационные боты используются для эффективного общения с пользователем, поскольку они позволяют предоставить четкие и понятные ответы на возникающие у него вопросы,

– функциональные боты предоставляют пользователю возможность взаимодействия с продуктом на основе удобного и интуитивно понятного интерфейса.

Так же выделяется бизнес-классификация (разговорные, ассистирующие и работающие по принципу «вопрос – ответ») и техническая классификация (реализованы на наборе бизнес-правил, на ИИ или объединяя оба вида) чат-ботов.

Таким образом, существует несколько способов коммуникации с ботом: назначенные диалоговые кнопки (например, «основное меню», «обратная связь», «частые вопросы» и т.д.), ключевые слова («/start», «/bye», «/help» и т.д.) и свободное общение на основе анализа произвольного текста на естественном языке. Чат-боты с технологией ИИ способны не только поддерживать связь с пользователем на примитивном уровне, но и разрабатывать новые алгоритмы, решать более сложные задачи.

Общая схема работы чат-бота приведена на рисунке 2.



Рисунок 2 – Принципы работы чат-бота

При выполнении запросов чат-бот обращается к внешним источникам информации, включая различные базы данных и сетевые репозитории.

Процесс взаимодействия может протекать по двум основным сценариям, которые важно учитывать:

1) в случае успешной обработки запроса пользователь получает ответ от бота, который использует специальный модуль для анализа естественного языка и понимания намерений пользователя. Если информации недостаточно, чат-бот может задавать уточняющие вопросы из своей базы данных, чтобы получить дополнительные данные. Весь контекст беседы и сообщения пользователя сохраняются для дальнейшего анализа и улучшения процесса взаимодействия,

2) в случае, если бот не может корректно обработать запрос пользователя, модуль управления диалогом сохраняет информацию в базе данных для последующего обучения и улучшения алгоритмов работы бота. Этот процесс позволяет боту постепенно улучшать свои навыки и повышать эффективность взаимодействия с пользователями.

Выделяется следующие подходы к реализации ПО для разработки чат-бота:

- однокомпонентная,
- многокомпонентная архитектура.

В первом случае, логика работы приложения не позволяет использовать взаимодействие между пользователями. При втором подходе, описываются отношения между клиентом (выполняет последовательность действий) и сервером (выполняет обработку, необходимую для получения результата и передает результат).

В современной российской практике влияние на формирование нового тренда, связанного с использованием чат-ботов, оказалось весьма значительным благодаря интеграции этих инновационных технологий в популярный мессенджер Telegram. Команда разработчиков внедрила специальные Bot API, предоставив возможность создания собственных чат-

ботов, а также запустила большое количество готовых ботов для удобства пользователей. Сейчас в Telegram доступно более тысячи разнообразных чат-ботов, способных удовлетворить разнообразные потребности пользователей и обеспечить им более удобное и интересное общение в мессенджере.

1.2 Анализ эффективности использования чат-бота в образовательном процессе

В сложившихся реалиях, активное внедрение цифровых образовательных ресурсов, в том числе и на основе искусственного интеллекта, становится все более актуальным и перспективным: появляются новые системы адаптивного обучения, увеличивается количество и качество визуальных контролирующих программ, автоматизированная проверка домашних работ, выявление и исправление ошибок, проведение онлайн-экзаменов и т.д. [1, 4].

На основе анализа различных источников [2, 5] выделим следующие сферы применения технологий на основе ИИ в сфере образования:

- автоматизированные учебные курсы,
- адаптированные и персонализированные системы обучения,
- консультирование и тестирование, корректировка лекционного и практического материала курса в зависимости предпочтений обучающегося и уровня усвоения его материала,
- фиксация сформированности учебных, профессиональных и цифровых компетенций на основе анализа данных,
- системы распознавания лиц в классах и аудиториях помогают понять, насколько обучающиеся погружены в образовательный процесс и т.д.

Использование чат-ботов в образовательном процессе предоставляет широкий спектр возможностей, которые могут значительно улучшить качество обучения и взаимодействия между преподавателями и студентами.

Первое, что стоит отметить, это возможность повышения мотивации обучающихся. Чат-боты способны стимулировать интерес к учебе, делая процесс обучения более увлекательным и привлекательным для студентов.

Кроме того, боты обеспечивают постоянную обратную связь, что позволяет студентам получать немедленные ответы на свои вопросы и сомнения, улучшая качество усвоения материала.

Одним из важных преимуществ использования чат-ботов является разгрузка преподавателей от работы по организации и проведению контрольно-оценочных мероприятий. Это позволяет преподавателям более эффективно использовать свое время на более значимые задачи.

Структурирование учебных и справочных материалов по изучаемой дисциплине также становится более удобным и доступным благодаря использованию чат-ботов. Студенты могут легко находить необходимую информацию и быстро ориентироваться в учебном процессе. Автоматизация процесса контроля знаний обучающихся позволяет эффективно отслеживать прогресс каждого студента и предоставлять персонализированную обратную связь для дальнейшего улучшения результатов.

Благодаря чат-ботам появляется возможность создания адаптивной и индивидуализированной траектории обучения, учитывающей особенности и потребности каждого студента. Различные виды консультаций, информационное сопровождение дисциплины, а также отправка актуальной информации, напоминаний и материалов для изучения становятся более удобными и доступными благодаря использованию чат-ботов.

Таким образом, педагогические работники образовательных организаций получают уникальные возможности для совершенствования своей работы. Они могут оперативно взаимодействовать со студентами, обмениваться важной информацией и эффективно управлять учебным процессом, что в конечном итоге способствует повышению качества образования.

Из выше сказанного можно сделать вывод, что оптимальной архитектурой взаимодействия пользователя с чат-ботом будет многокомпонентная:

- пользователь (клиент) задает вопрос или вводит фразу,

- бот должен вычленить какую-то фразу, чтобы понять намерения пользователя, т.е. сервер выполняет обработку запроса, необходимую для получения результата,
- сервер передает результат, бот выводит ответ.

Многокомпонентная архитектура чат-бота взаимодействия «Клиент – сервер» представлена на рисунке 3.

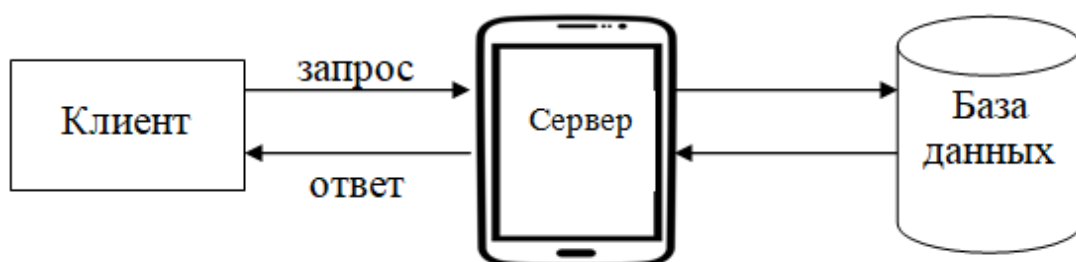


Рисунок 3 – Многокомпонентная архитектура взаимодействия «Клиент – сервер»

Среди приоритетных значений ИИ в образовании можно выделить:

- ИИ как предмет изучения;
- ИИ как инструмент обучения;
- ИИ как результат современной науки.

На рисунке 4 представлена концепция образовательного процесса, где интегрирована технология чат-ботов. С включением этого компонента в систему, количество учебных инструментов снижается заметно. Боты, выполняя роль как содержания обучающего материала, так и инструмента обучения, заменяют традиционные элементы, что способствует упрощению образовательного процесса и повышает его эффективность. Кроме того, стоит отметить, что хотя чат-боты пока не в состоянии взять на себя все образовательные обязанности, они способны выполнить их часть.

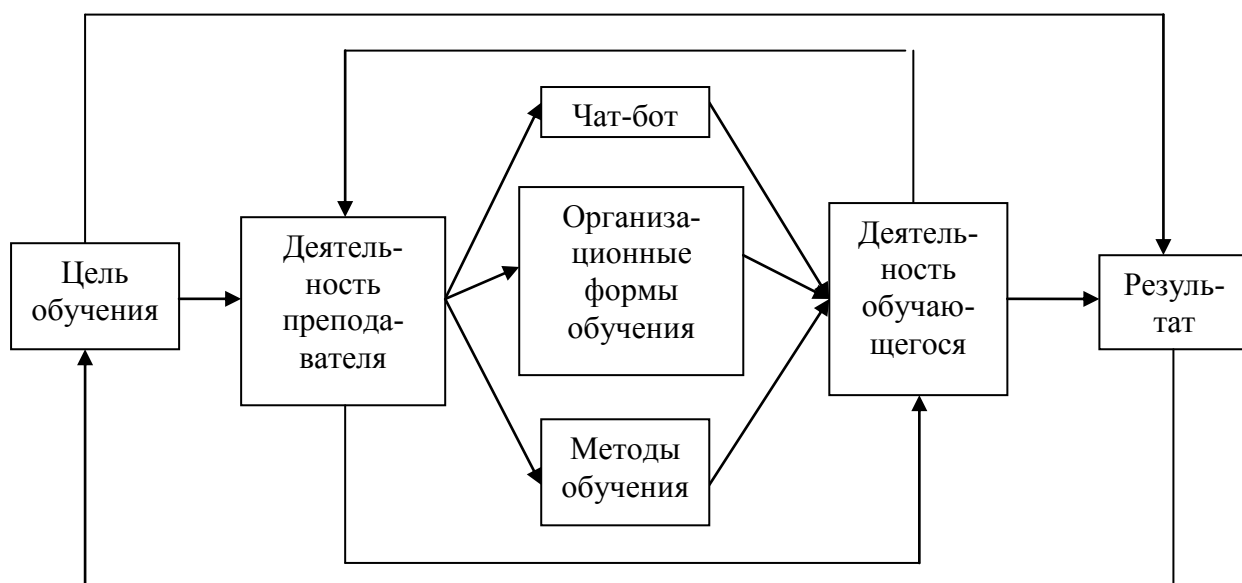


Рисунок 4 – Многокомпонентная архитектура взаимодействия

Таким образом, применение чат ботов не является новой технологией и активно применяется в различных сферах образования. Их база знаний, при активном использовании, будет только расти, а сами боты смогут потенциально учиться вместе с обучающимися уже через несколько лет.

1.3 Определение функциональных требований к работе образовательных чат-ботов

В настоящее время чат-боты часто находят применение в образовательной сфере. Их разрабатывают для упрощения процесса обучения, улучшения усвоения материала и контроля знаний. Для преподавателей учебных заведений открываются новые пути для оперативного общения со студентами и эффективного распространения учебных материалов. Несмотря на то, что не все аспекты обучения подходят для реализации через диалог с ботом, уже сейчас существует обширный ассортимент образовательных ботов, которые специализируются на предоставлении кратковременной и полезной информации:

- чат-бот «Гид по истории» (@RU_history_bot используется в Санкт-Петербургском государственном университете),
- чат-бот «Помощник студента» (https://vk.com/businesscollege_nsuem используется в Новосибирском государственном университете),

- бот ПГУ (<https://t.me/pnzgu> – чат-бот Пензенского государственного университета, предоставляет студентам информацию о проходных баллах, дате сессии и других академических вопросах),

- чат-бот «Блокнот философа» (<https://t.me/philosnote> публикуется информация о предстоящих мероприятиях, конференциях и новых исследованиях в области философии НИУ ВШЭ) и т.д.

Исходя из анализа множества научных работ [1, 11, 12], можно выделить универсальные критерии для создания чат-бота в образовательной сфере:

- при первом взаимодействии с пользователем чат-бот должен продемонстрировать приветствие и предоставить ключевую информацию о порядке работы приложения (к примеру, вступительное сообщение от бота, посвященного дисциплине «Современные информационные технологии», с предложением оказать помощь и предложением ознакомиться с наиболее частыми вопросами),

- интеллектуальный чат-бот должен быть оснащен уникальным алгоритмом, способным вести диалог на основе структурированной информации (например, классифицировать типы вопросов и предлагать корректные ответы).

При работе над приложениями с чат-ботами важно уделить внимание вопросу хранения информации (БД вопросов и ответов), необходимой для эффективного ведения диалога и обучения системы. Решение задачи понимания естественного языка можно разделить на следующие этапы:

- преобразование текста для анализа, включая предварительную обработку текста для определения намерений пользователя,

- организация запросов, их сопоставление с имеющимися данными в системе (удаление лишних символов, исправление опечаток, учет различных регистров слов и другие моменты).

Таким образом, взаимодействие пользователя с чат-ботом начинается с вывода приветственного сообщения (основные сведения приложения, вывод

инструкций по использованию и перечисление выполняемых программ). Далее необходимо отправить сообщение (задать вопрос) или выбрать нужный пункт меню. С помощью этих инструкций пользователь должен получить ответ на интересующий его вопрос. Разработанный программный продукт в ходе диалога может уточнять вопросы, после чего будет выдавать список ответов, подходящих полученным данным. По мере увеличения базы знаний чат-бота качество ответов будет повышаться. При этом пользователю будет проще понять, что именно ему требуется по параметрам меню.

1.4 Выводы

В рамках этого раздела были достигнуты следующие результаты: проанализированы общие понятия по теме исследования, рассмотрена общая схема работы чат-бота, его взаимодействие с пользователем, представлены результаты анализа эффективности использования чат-ботов в образовании. Отметим, что информационная поддержка дисциплины через чат-бот актуальна и в целом востребована среди преподавателей и студентов.

2 Выбор программного обеспечения

2.1 Обзор популярных платформ для создания чат-ботов

Создать чат-бот можно следующими способами:

- программировать вручную,
- использовать drag and drop конструктор.

Для создания чат-бота существует ряд бесплатных приложений, с помощью которых можно создать работоспособную автоматическую систему (например, для консультирования студентов):

- BotMother – удобный инструмент для создания чат-ботов в социальных сетях. Благодаря его простоте использования, нет необходимости в знании программирования. Главное преимущество – интуитивно понятный интерфейс на русском языке,

- PuzzleBot – узко направленный конструктор, который работает только с Телеграм. Для работы с системой не нужно дополнительных навыков в программировании – работа ведётся в личном кабинете. Сервис многофункциональный: рассылки, магазин, работа с группами и каналами, закреп, веб-приложения, платежи и многое другое,

- Botpress – то платформа для создания чат-ботов AI, которая предлагает простой и интуитивно понятный интерфейс для разработки и управления чат-ботами. Для использования некоторых продвинутых функций может потребоваться платная подписка,

- Rasa – это открытая платформа для создания чат-ботов AI, которая предлагает гибкость и контроль над процессом разработки. Rasa позволяет разработчикам полностью контролировать процесс обучения и развертывания чат-ботов. Для начинающих разработчиков Rasa может быть сложным из-за своей гибкости и необходимости управлять процессом обучения,

- DialogFlow – это популярная условно-бесплатная платформа для создания чат-ботов, которая предоставляет возможность создания виртуальных ассистентов пользователям, не имеющим навыков

программирования. Dialogflow использует машинное обучение для преобразования запросов клиентов в структурированные данные, которые можно использовать для генерации ответов. Обучая модель набору запросов клиентов, платформа может научиться распознавать шаблоны и извлекать ключевую информацию из запроса. Далее эта информация может быть использована для выработки ответа, адаптированного к потребностям заказчика. На рисунке 5 представлен скриншот основной страницы платформы DialogFlow.

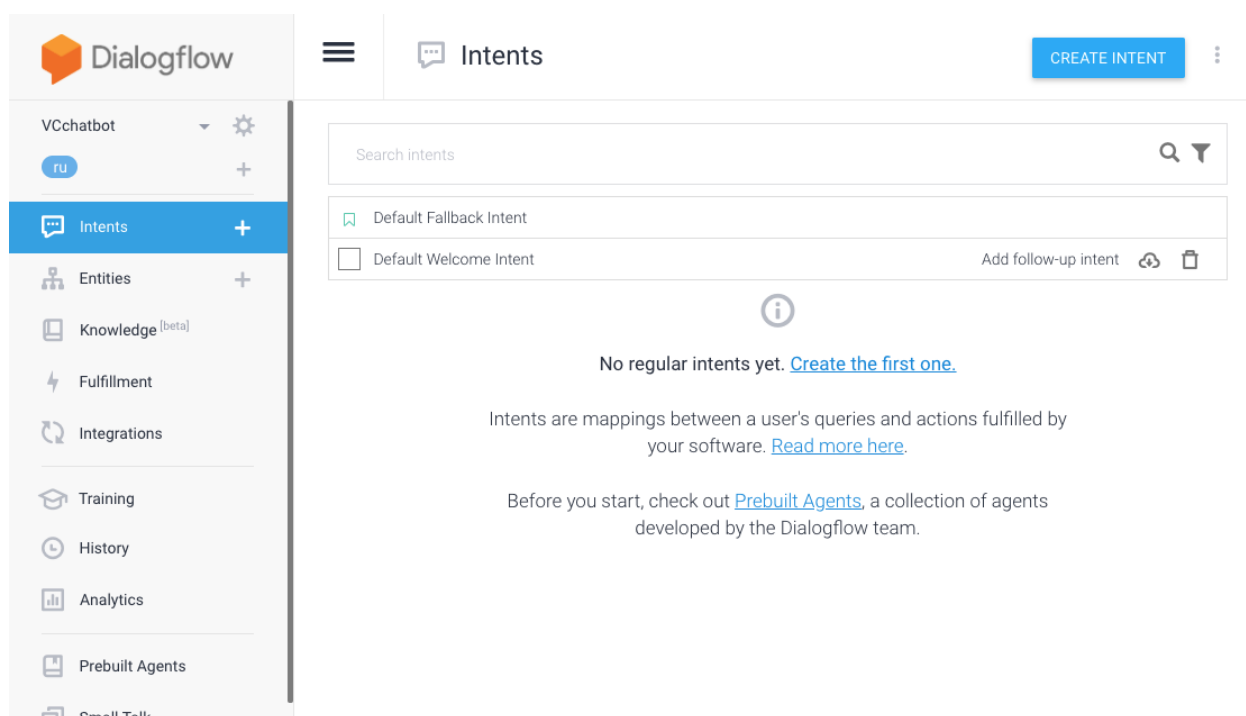


Рисунок 5 – Скриншот основной страницы DialogFlow

Таким образом, у конструкторов чат-ботов есть следующие недостатки:

- ограниченность функционала. Все же не все конструкторы поддерживают возможность прикреплять файлы и организовывать каталоги товаров, а также планировать рассылки. В качестве примера можно привести использование таких функций, как массовая рассылка, отслеживание статистики и анализ аудитории, ответы на комментарии и прием платежей, которые требуют дополнительных затрат,
- зависимость от поставщика услуги. С помощью платформы происходит загрузка пользовательских ботов на своих серверах. Если

произойдет непредвиденная ситуация в компании, то инструменты могут временно перестать выполнять свои функции,

- проблемы с конфиденциальностью. В виду того, что боты находятся на серверах платформы-разработчика, пользователям-клиентам приходится соглашаться с их правилами конфиденциальности.

2.2 Обзор и сравнительный анализ средств разработки

На сегодняшний день самыми востребованными языками программирования являются Java, JavaScript, Python и C#, именно их выберем для анализа и сравнения. Следует отметить, что у каждого из этих языков есть свои преимущества и недостатки (открытость разработки, функциональные возможности, синтаксис, встроенные библиотеки и модули, кроссплатформенность и т.д.).

Python занимает первое место по популярности языков программирования. Python – это универсальный язык программирования, который поддерживает принципы объектно-ориентированного программирования и используется для создания веб-приложений. Его простой и понятный синтаксис, множество полезных библиотек и хорошо продуманные механизмы модульности делают его легким в использовании. Бот, написанный на Python, будет отличаться относительно высокой скоростью работы, безопасностью и надежностью [6].

Выделим преимущества и недостатки создания интеллектуального чат-бота на Python:

- финансовая экономия;
- получение навыков программирования, в том числе в области решения задач ИИ;
- Python имеет много документации и имеет большой выбор встроенных модулей и библиотек;
- потребление значительного количества времени на изучение навыков программирования.

Python предоставляет множество функций и структур данных, включая списки, кортежи, словари и строки. Более того, он предоставляет эффективные инструменты для работы с объектами и их управления. Python также поддерживает создание модулей, классов и исключений, что позволяет создавать большие проекты.

Превосходство Python проявляется в обширном арсенале инструментов для работы с естественным языком, включая библиотеку `nltk`, которая является ключевым модулем данной программы и предлагает широкий спектр средств для анализа текстовых данных [14].

Основные функции `nltk`:

- разделение текста на токены: `nltk` предлагает способы разделения текста на токены с помощью различных правил и алгоритмов.
- исключение стоп-слов: `nltk` включает в себя список стоп-слов для разных языков. Стоп-слова могут быть использованы для фильтрации токенов и их удаления из текста.
- лемматизация и стемминг: `nltk` предоставляет инструменты для проведения лемматизации и стемминга слов.

Для работы с регулярными выражениями есть множество инструментов и онлайн-сервисов, в языке программирования Python существуют два ключевых модуля: `re` и `regex`.

Java занимает второе место после Python как один из лучших языков программирования для общего назначения и искусственного интеллекта. Java широко используется в разработке решений для машинного обучения, нейронных сетей, алгоритмов поиска, генетического программирования и мульти-робототехнических систем.

Преимущества Java включают в себя высокий уровень безопасности, обеспечиваемый байт-кодом, который проходит проверку перед исполнением, и надежную защиту от несанкционированного доступа к памяти. Кроме того, Java имеет крупное сообщество разработчиков и

собственную экосистему, где всегда можно получить помощь по вопросам программирования.

Среди недостатков выделяют следующие:

- использование памяти. В языке Java требуется больше оперативной памяти, чем в большинстве других объектно-ориентированных языках программирования. Это может привести к проблемам с запуском разработанных приложений на слабых устройствах. Относительная медлительность выполнения,

- производительность приложений, написанных на Java, особенно в случае крупных проектов, может оставлять желать лучшего. Хотя оптимизация и использование JVM позволяют ускорить обработку данных, этот прирост скорости не всегда значителен,

- сложность некоторых аспектов. Например, работа с многопоточностью и управление памятью могут быть сложными. Для новичков может быть трудно освоить соответствующие концепции сразу,

- из-за компиляции в байт-код приложения, разработанные на Java, могут запускаться дольше, чем при использовании других языков программирования.

JavaScript – это язык программирования, который поддерживает различные подходы: объектно-ориентированный, императивный и функциональный.

Преимущества JavaScript включают в себя:

- каждый современный браузер поддерживает JavaScript,
- полезные функциональные возможности,
- возможность взаимодействия с приложением даже через текстовые редакторы.

Выделим недостатки JavaScript:

- проблема работы с типами данных в языке JavaScript заключается в недостаточной строгости обращения к переменным. Возможность

выполнения операций, не соответствующих ожиданиям, и игнорирование явных несоответствий являются характерными чертами данного языка,

- пониженный уровень безопасности ввиду повсеместного и свободного доступа к исходным кодам популярных скриптов,

- JavaScript не является объектно-ориентированным языком, однако позволяет реализовать ООП-подобный подход, в частности, в функциональном стиле: роль классов выполняют функции-конструкторы.

Большинство библиотек машинного обучения (Brain, FlappyLearning, , Neurojs, DeepForge и т.д.) JavaScript являются достаточно новыми, и все еще находятся в разработке.

C# – это язык программирования, который объединяет в себе объектно-ориентированные и контекстно-ориентированные принципы. Он использует компонентно-ориентированный подход к программированию, который способствует уменьшению зависимости программного кода от архитектуры машины, увеличивает гибкость, переносимость и упрощает повторное использование фрагментов программы. C# – язык безопасных типов, т.е. компилятор гарантирует, что значение хранящееся в переменной будет всегда соответствующего типа. При объявлении переменной в C# необходимо сначала указать ее тип – строка, число или массив. Хотя разработка с использованием такого подхода может занять немного больше времени, написанный код становится более надежным, поскольку числа взаимодействуют только с числами, строки – со строками и так далее. Безопасность – один из основных аспектов при разработке приложений на языке программирования C#.

Несмотря на все преимущества, язык C# имеет некоторые недостатки, такие как: относительно низкая производительность (при открытии программы на C# код сначала адаптируется под конкретное аппаратное обеспечение, а уже потом исполняется), не кроссплатформенность.

По мнению специалистов в области разработки на С#, машинное обучение представляет собой сложную задачу, так как требует обширных специализированных знаний и ресурсов для создания и обучения моделей.

Язык программирования R широко применяется в сфере анализа и обработки данных с целью проведения статистических исследований. Кроме того, в его арсенале имеется целый ряд пакетов и библиотек, способных облегчить разработку проектов, связанных с машинным обучением. Например, Gmodels, Class, Tm, Tensorflow и RODBC предлагают различные инструменты для работы с данными и построения моделей машинного обучения.

В таблице 1 приведены сравнительные характеристики функциональных возможностей мессенджеров с точки зрения удобства использования и создания чат-бота: Telegram, Viber, ВКонтакте и WhatsApp.

Таблица 1 – Сравнительные характеристики мессенджеров

	Telegram	Viber	ВКонтакте	WhatsApp
Популярность на 2022 г. в России (%)	50,8	77	76,4	80
Возрастная целевая аудитория	Не старше 34 лет	25-44 года	Меньше 18, 18-24 года	26–35 лет
Открытый API	+	+	+	-
Предоставляемые возможности	Кнопки меню, отправка аудио- и видеофайлов, изображений, документов и т.д.	Нельзя отправ-лять аудиофайл	Устанавливает-ся для каждой группы, отправка видеофайлов, изображений, gif-анимации	Ограниченное количество кнопок меню, отправка аудио- и видеофайлов, изображений, документов gif-анимации и т.д.
Рассылка (платная или бесплатная)	Бесплатная	Бесплатная	Бесплатная	Платная

Функционал мессенджеров, включая Telegram, ориентирован прежде всего на активную мобильную аудиторию, которая активно использует смартфоны как для работы, так и для отдыха: пользователю не нужно заходить на веб-сайт, достаточно просто отправить сообщение боту. При выборе мессенджера для создания чат-бота важно оценить его возможности и уровень сложности запуска программы. Например, в Telegram имеется множество инструментов для разработки чат-ботов, что делает их создание проще и дешевле, в то время как в WhatsApp разработка ботов на данный момент запрещена [15, 16].

При работе с чат-ботами часто возникает проблема мультязычности из-за того, что пользователи используют заимствованные слова или цитаты. Современный подход к решению этой проблемы заключается в создании модели, способной объединять различные языки в одном векторном пространстве. Таким образом, предложения на разных языках будут находиться близко друг к другу независимо от языка их исходного ввода. Например, метод CountVectorizer из библиотеки Sklearn используется для преобразования набора текста документы в вектор количества терминов, т.е. создает векторное пространство текстовых данных.

Таким образом, в основе исследования по разработке чат-бота на основе искусственного интеллекта лежит процесс предварительной обработки естественного языка, в том числе и на основе алгоритмов машинного обучения.

R и Python – это два широко используемых языка программирования в области анализа данных, машинного обучения и искусственного интеллекта. Ни один язык не является идеальным для всех сценариев, а выбор конкретного языка может зависеть от требований проекта и предпочтений разработчика. В сравнении с такими языками программирования, как C#, C++, Java или JavaScript, Python выделяется своей простотой и функциональностью при разработке ботов для Telegram. Это, в сочетании с рассмотренными выше критериями выбора мессенджера, делает Python

самым привлекательным выбором для создания интеллектуальных чат-ботов для образовательных целей на основе использования библиотек для машинного обучения.

Рассмотрим наиболее популярные среды разработки, использующие Python.

Visual Studio Code – это редактор кода, отличающийся своей простотой и эффективностью. Он поддерживает операционные системы Windows, MacOS и Linux. Для удобства разработчиков, занимающихся созданием программного обеспечения, в VS Code предусмотрены функции отладки, инструменты для работы с Git-репозиториями, синтаксическая подсветка, возможности рефакторинга и IntelliSense – инструмент автодополнения, который предлагает команды по первым буквам.

Visual Studio Code является весьма гибким инструментом, который предоставляет пользователям уникальную возможность полностью настраивать его интерфейс и функции в соответствии с требованиями и задачами, которые они сталкиваются в процессе работы. Это достигается благодаря возможности не только изменять настройки самой программы, но и настраивать параметры различных плагинов, которые установлены в системе. Такой подход к настройкам позволяет каждому пользователю создать уникальную рабочую среду, оптимизированную для его индивидуальных потребностей и повышающую эффективность и удобство использования Visual Studio Code. Среда разработки поддерживает различные языки программирования, включая Python.

Выделим основные преимущества и недостатки Visual Studio Code.

Преимущества:

- простой и удобный редактор кода для создания веб-приложений на различных платформах,
- включает в себя отладчик, инструменты Git, подсветку синтаксиса, расширяемые плагины и возможности рефакторинга,

- программа предлагает широкие возможности для настройки: пользовательские темы, сочетания клавиш и файлы конфигурации,
- поддерживается на Windows, macOS и Linux.

Недостатки:

- редактор основан на Electron, что означает его работу как процесс JavaScript, а не как отдельное приложение,
- на старых компьютерах редактор может долго запускаться и работать медленнее.

Atom – это бесплатное приложение для редактирования текста на Windows с открытым исходным кодом, которое поддерживает плагины, написанные на JavaScript и встраиваемых по управлению Git. Он поддерживает большое количество различных расширений, благодаря которым его можно сравнить с настоящей средой разработки. Программа создана на основе фреймворка Electron (ранее известного как Atom Shell), который использует Chromium и io.js для кроссплатформенной разработки. Для работы с языком программирования Python необходимо установить соответствующее расширение и настроить основные функции.

Преимущества:

- легкий и быстрый в использовании текстовый редактор,
- поддержка широкого спектра языков программирования благодаря встроенным компиляторам или интерпретаторам для каждого языка, например Python или C++,
- у редактора имеется интуитивно понятный интерфейс, его установка и настройка проходят легко. После этого сразу можно приступить к использованию,
- наличие встроенного отладчика, инструментов для работы с Git, подсветки синтаксиса и возможность установки дополнительных плагинов.

Редактор работает как процесс JavaScript в Electron, а не как отдельное приложение, что может быть недостатком.

Для написания кода на Python будет использоваться редактор кода PyCharm. Данная интегрированная среда представляет собой одно из наилучших средств в настоящее время, являющееся полнофункциональной, специализированной и универсальной программой разработки, предназначенной для написания программного кода, связанного с функционалом чат-бота.

На рисунке 6 предоставлен интерфейс среды разработки PyCharm.

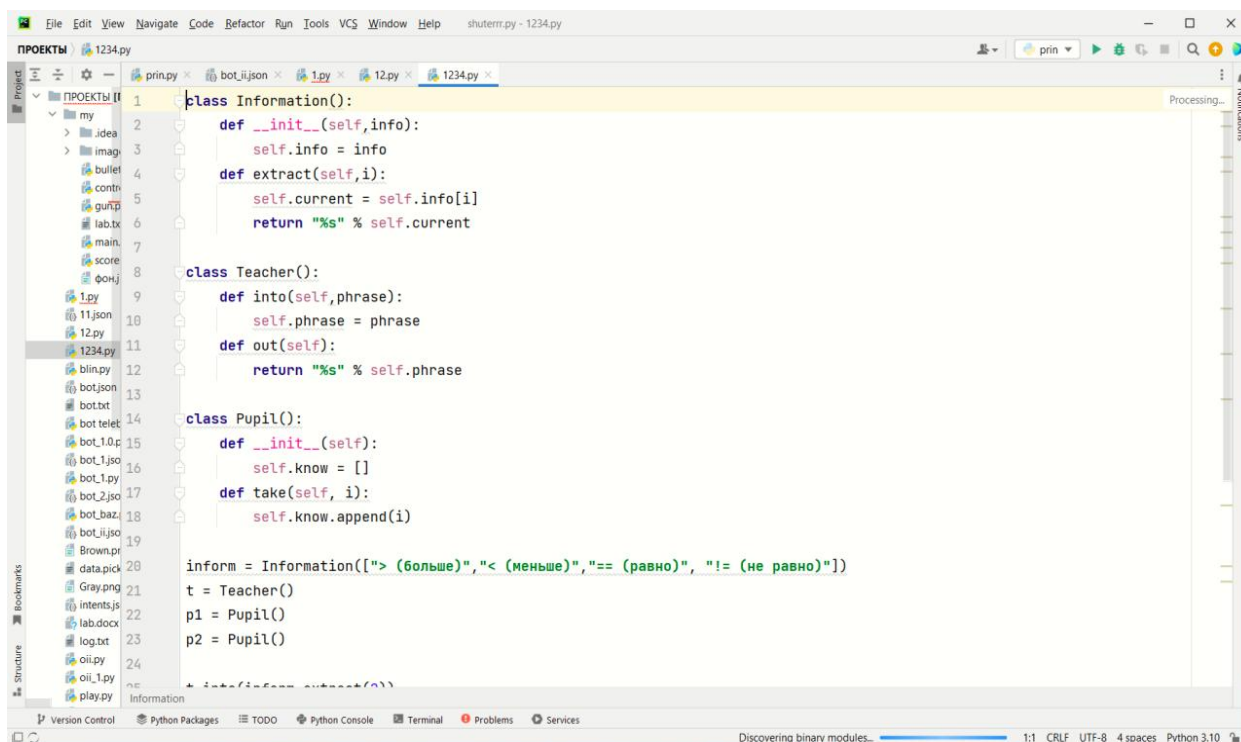


Рисунок 6 – Интерфейс PyCharm Community Edition 2022.2.1.

Далее в работе отмечены основные преимущества и недостатки IDE PyCharm.

Преимущества:

- все возможности интегрированных сред программирования (компилировать код, собирать проект, работать с системами контроля версий, создавать визуальный интерфейс и т.д.),
- простой запуск и написание кода, подсветка синтаксиса, помощь в написании кода с функцией автозаполнения, простая установка внешних библиотек и модулей (например, помощью pip) и т. д.,

- доступна свободно-распространяемая версия Community с открытым исходным кодом.

Недостатки:

- относительно медленная загрузка среды разработки,
- требует более детальной настройки под проект.

Одной из ключевой особенностью PyCharm является редактор кода, который поддерживает широкий спектр языков программирования, включая Python 3.x, JavaScript, CSS, HTML и многие другие. Эта интегрированная среда разработки предоставляет разработчикам инструменты для быстрой идентификации синтаксических элементов, проверки кода на ошибки и их последующего исправления. В PyCharm также присутствует интеллектуальный поиск, который позволяет осуществлять поиск не только в коде и событиях, но и в окне инструментов PyCharm, обеспечивая быстрый доступ к методам, тестированию и другим аспектам разработки ПО.

Факт, что PyCharm поддерживает популярные библиотеки, такие как Anaconda, Numpy, Matplotlib и Scikit-learn, и предоставляет пользователям возможность изучать код, создавать интерактивные графики, просматривать массивы и многое другое, играет важную роль при выборе этой среды разработки.

В работе по созданию чат-бота были использованы Python 3.10 и PyCharm Community Edition 2022.2.1.

2.3 Анализ выбора базы данных для хранения сведений о пользователях

Для хранения данных пользователей необходима база данных (БД). В работе она включает в себя список зарегистрированных (id, имя и фамилию в Telegram).

PostgreSQL – это самая распространенная бесплатная база данных для работы с веб-приложениями на Python. С помощью PostgreSQL можно создавать, хранить базы данных и работать с данными с помощью запросов на языке SQL. Система управления базами данных PostgreSQL обладает

различными вариантами пользовательского интерфейса, что обеспечивает удобство в настройке и персонализации в соответствии с требованиями проекта. Для осуществления соединения с базой данных PostgreSQL из скрипта на языке программирования Python требуется применение специализированной библиотеки-коннектора [17].

В Python можно выбрать одну из множества альтернатив, но библиотека Psycopg2 является наиболее известной и широко используемой. Чтобы использовать Psycopg2, нужно установить его через терминал или командную строку с помощью pip.

Одной из ключевых особенностей PostgreSQL является его масштабируемость. Система успешно применяется в крупных проектах обработки Big Data, благодаря своей стабильности и высокой производительности.

Преимущества:

- находится в свободном доступе, открытый исходный код,
- кроссплатформенность,
- поддерживает много разных типов и структур данных,
- размер базы данных не ограничен,
- поддерживает все функции, которые есть в современных БД,
- подходит для работы с большими данными и сложной аналитики.

Недостатки: многофункциональна, из-за чего работает медленней, чем MySQL, поэтому менее популярна;

SQLAlchemy представляет собой библиотеку, разработанную на языке программирования Python, предназначенную для взаимодействия с реляционными системами управления базами данных с использованием объектно-реляционного отображения (ORM).

ORM – это метод программирования, который устанавливает соответствие между базами данных и объектами в объектно-ориентированной парадигме, что позволяет создать виртуальную объектную модель базы данных.

Преимущества:

- ПО с открытым исходным кодом,
- кроссплатформенность,
- совместимость с SQL-запросами, написанными вручную,
- возможность формирования запросов через функции и выражения Python,
- модульность и достаточно высокий уровень расширяемости,
- гибкий дизайн позволяет легко писать сложные запросы,
- предусматривает предварительную и последующую обработку информации.

Недостатки:

- не обеспечивает базовые операции с БД, вместо этого это делается с помощью сторонних библиотек,
- тяжелый API, требующий долгого обучения.

MySQL является ведущей системой управления базами данных с открытым исходным кодом, на которой можно построить как веб-приложения, так и сложное ПО. Благодаря широкому кругу пользователей, работа с популярным ПО на MySQL упрощается, а баги быстро обнаруживаются и исправляются благодаря активной поддержке сообщества разработчиков. Установка MySQL обычно не вызывает проблем.

Для подключения к базе данных MySQL из Python требуется установить соответствующий SQL-драйвер. Она постоянно обновляется новыми плагинами и расширениями. Для создания БД в MySQL необходимо выполнить два шага: подключиться к серверу и выполнить запрос на создание базы данных SQL.

Преимущества:

- экономия памяти сервера,
- обеспечение кругу лиц доступа к дереву файлов БД без потерь для системы безопасности.

Недостатки:

- трудность в освоении,
- достаточно большие файлы БД для приложений,
- определенные финансовые затраты на обслуживание базы данных,

приобретение стороннего ПО, размещение на хостинге и т.д.

SQLite предоставляет чисто реляционную модель баз данных, где данные хранятся в таблицах и связаны между собой согласно реляционной модели [18]. Интерфейс программы SQLite представлен на рисунке 7.

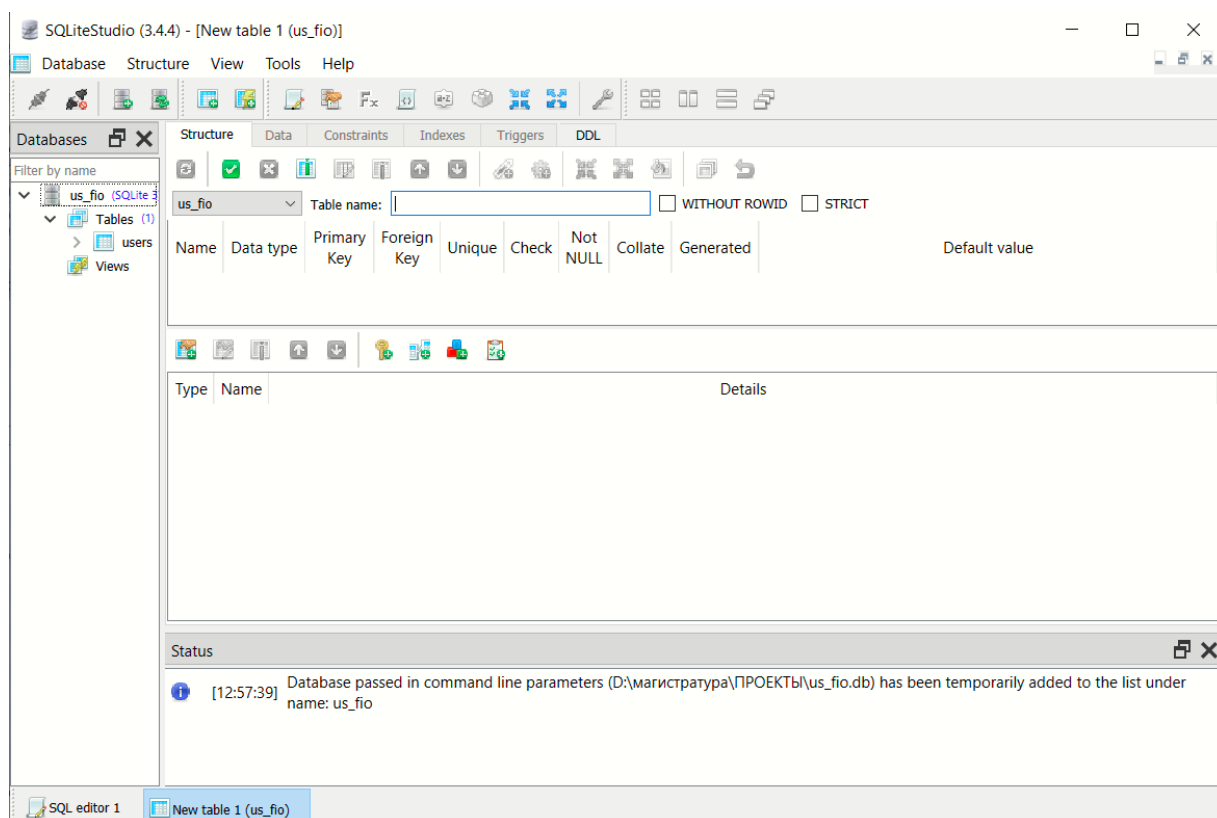
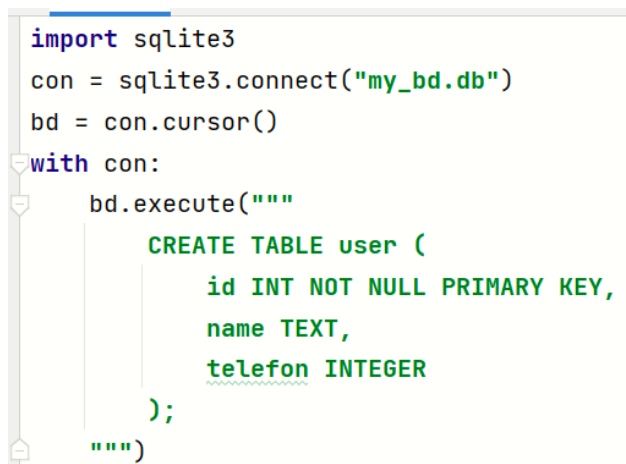


Рисунок 7 – Интерфейс SQLite

Она поддерживает различные типы данных, включая NULL, INTEGER, REAL, TEXT и BLOB, что позволяет нам гибко работать с разнообразными данными, включая целые числа, числа с плавающей запятой, текстовые строки и двоичные данные [19].

SQLite3 (или SQLite) – это файловая база данных SQL, которая является частью стандартного пакета Python 3. Для использования SQLite в Python достаточно импортировать соответствующий модуль.

Далее на рисунке 8 приведен пример кода для создания таблицы в SQLite в Python. БД будет содержать одну таблицу user с двумя полями id, name и telefon. Файл с созданной БД будет называться my_bd.db.



```
import sqlite3
con = sqlite3.connect("my_bd.db")
bd = con.cursor()
with con:
    bd.execute("""
        CREATE TABLE user (
            id INT NOT NULL PRIMARY KEY,
            name TEXT,
            telefon INTEGER
        );
    """)
```

Рисунок 8 – Создание таблицы в SQLite в Python

Для работы с БД SQLite в Python поддерживает меньше типов данных, чем некоторые другие SQL-реализации, но не требует наличия сервера, что обеспечивает возможность локального хранения базы данных на одном устройстве.

Преимущества:

- хранение информации в одном документе,
- надежная система защиты,
- не требуется установки,
- находится в свободном доступе,
- автономное функционирование.

Недостатки:

- ограничения в поддержке типов данных,
- ограниченность в применении,
- отсутствуют многопользовательские функции,
- бесплатной поддержки нет.

Для хранения данных пользователей Telegram-бота была выбрана база данных SQLite. Она идеально подходит для разработки и тестирования, поскольку она предоставляет достаточно функциональности и простоты

использования для работы с одним файлом и связанной библиотекой, что позволяет эффективно разрабатывать и проверять функциональность различных аспектов чат-бота, а также обеспечивает масштабируемое решение для будущего развития проекта.

2.4 Выводы

В рамках этого раздела были достигнуты следующие результаты: проведен обзор популярных платформ для создания чат-ботов, обоснован выбор программных средств для разработки программного продукта, по результатам которого, был выбран язык программирования Python, интегрированная среда разработки PyCharm Community Edition. Для хранения пользовательских данных разрабатываемого Telegram-бота была выбрана БД SQLite.

3 Разработка программного продукта

3.1 Регистрация чат-бота

В современном мире информационных технологий существует немало различных видов чат-ботов, каждый из которых обладает уникальными характеристиками и способами взаимодействия с пользователями. Одним из таких видов является кнопочный чат-бот. Этот вид ботов обычно используется в тех случаях, когда необходима односторонняя связь, где пользователь выбирает из предложенных кнопок уже готовые запросы. Такой подход к взаимодействию имеет свои особенности, поскольку он существенно ограничивает диалог, предоставляя лишь ограниченный набор фраз, которые были заранее запрограммированы.

Текстовый чат-бот, в свою очередь, представляет собой более продвинутый и функциональный вид ботов. Он позволяет пользователю вводить текст, который затем анализируется ботом, чтобы найти наиболее подходящий ответ в своей базе данных. Это открывает широкие возможности для диалога между ботом и пользователем, делая взаимодействие более гибким и динамичным.

Первым шагом разработки приложения в Telegram (имеется ввиду бот на основе BotApi telegram) необходимо перейти в официального бота @BotFather (<https://t.me/BotFather>), нажать «Запустить». Минус BotFather в том, что его интерфейс на английском, но вся лексика довольно проста. Далее нужно указать команду /newbot. Она необходима для создания бота, соответственно и для его дальнейшего использования. Один пользователь может создать до 20 ботов [20, 21].

Следующим шагом является определение имени для чат-бота, которое обязательно должно заканчиваться на «bot», например, «MyBot» или «my_bot». Когда все требования соблюдены, @BotFather выдает токен — уникальный код, необходимый для взаимодействия с HTTP API Telegram Bot. Этот токен содержит зашифрованную информацию, необходимую для

аутентификации и доступа к сервису. В случае утраты токена или его несанкционированного использования, всегда можно запросить новый.

Длина @username для бота: 5-32 символа.

Пример регистрации чат-бота представлен на рисунке 9.

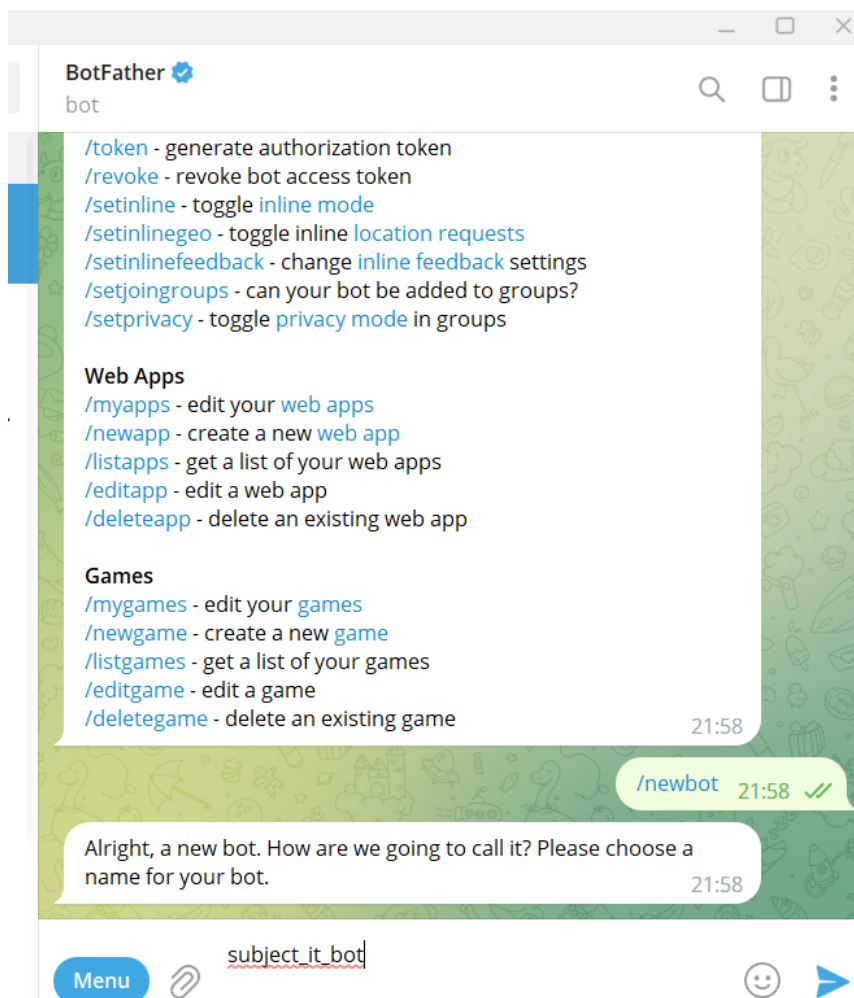


Рисунок 9 – Регистрация бота в Telegram

В @BotFather удобно управлять ботами своими командой /mybots. Интерфейс интуитивно понятный. Чтобы произвести пользовательские настройки, нужно перейти в раздел /mybot, далее Edit Bot.

На рисунке 10 представлен скриншот команды /mybot.

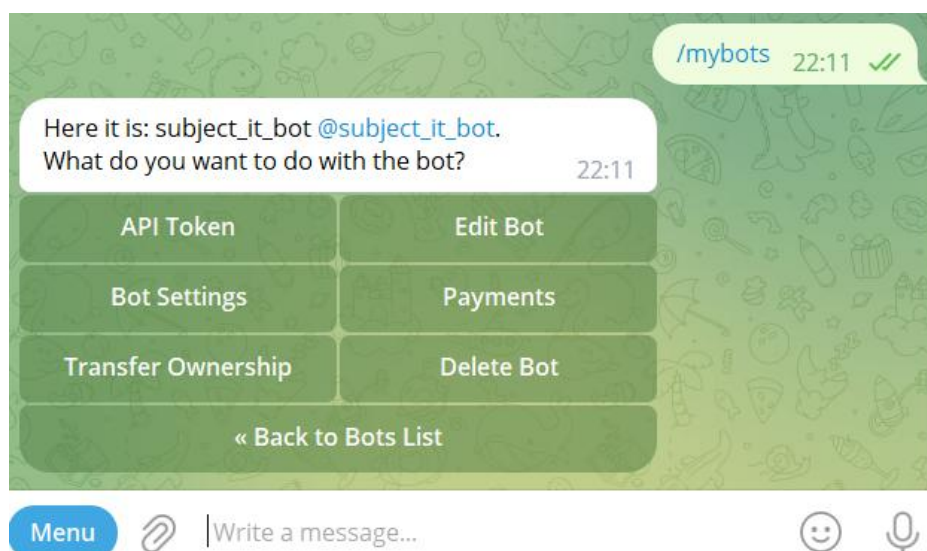


Рисунок 10– Настройка бота командой /mybot

В разделе настроек выделены следующие команды:

/setname – изменить имя бота

/setdescription – изменить описание бота, текст до 512 символов, описывающий бота

/setabouttext – добавить приветствие (информация о боте)

/setuserpic – установить аватар

/setcommands – установить список команд, которые будут доступны для использования

/deletebot – удаляет выбранного чат-бота

/stop – остановить работу бота

/start – запустить работу бота

Когда пользователь впервые открывает бота, он видит кнопку «Запустить» («/start»). Нажимая на эту кнопку, бот начинает свою работу. Если пользователь не запускал бота, он не активируется и не будет отправлять сообщения.

В Telegram можно делиться файлами до 2 ГБ, но в Bot API боты могут скачивать только файлы до 20 МБ и отправлять файлы до 50 МБ.

Если вы хотите позволить конкретному пользователю управлять созданным ботом и редактировать его содержимое, вам будет полезна функция передачи прав.

Таким образом, через BotFather можно зарегистрировать нового бота, придумать ему имя и логин, стартовое сообщение, добавить картинку, описание в профиле и дать команды. После того, как выполнены все необходимые настройки на стороне Telegram и успешно получен токен, наступает момент перехода к следующему этапу разработки – созданию программной части чат-бота. Этот этап играет ключевую роль в формировании функционала и уникальности бота, позволяя ему взаимодействовать с пользователями и выполнять поставленные задачи.

После всех настроек с помощью BotFather, разрабатываемый бот выглядит в Telegram как на рисунке 11.

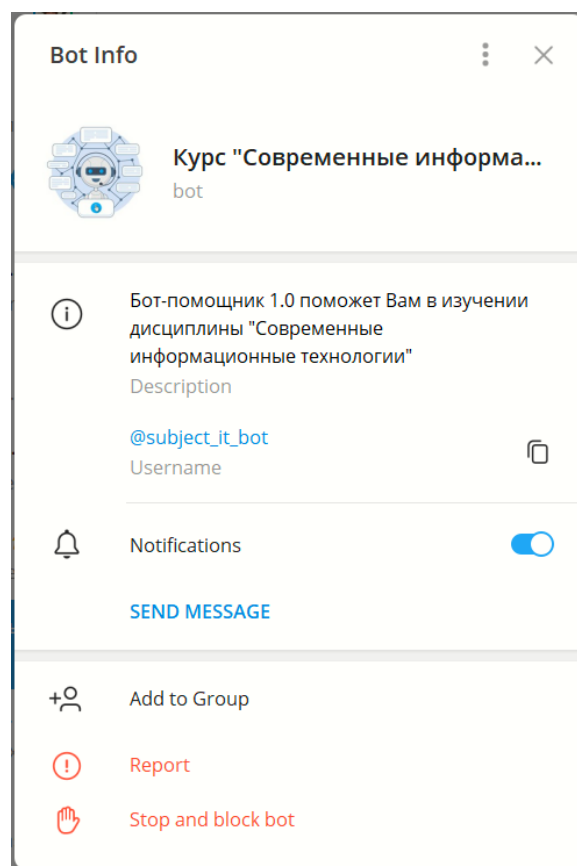


Рисунок 11 – Разрабатываемый чат-бот

«Курс «Современные информационные технологии»»

Стартовая страница бота представлена на рисунке 12.

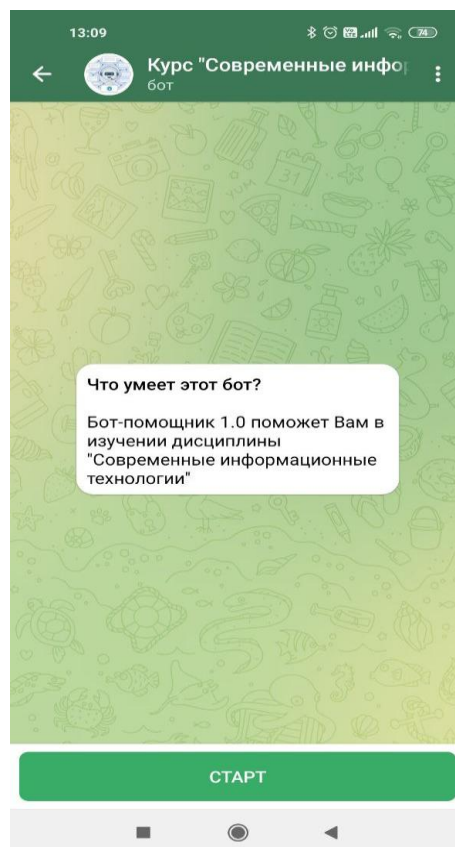


Рисунок 12 – Стартовая страница бота

Telegram Bot API является универсальным инструментом для создания чат-ботов на платформе Telegram. Он поддерживает множество языков программирования, но для удобства разработки был выбран язык Python, благодаря его гибкости и широкой поддержке сообщества.

Библиотека Telebot – это библиотека Python, которая предоставляет интерфейс для взаимодействия с Telegram Bot API. Она позволяет отправлять сообщения, создавать функциональные встроенные клавиатуры, обрабатывать обратные вызовы и выполнять множество других действий. Библиотека проста в использовании и может быть установлена с помощью pip.

У объекта bot есть множество методов, частью из них будет пользоваться при создании своего бота. Основной из них является функция send message, которая отправляет некоторое сообщение окну или нескольким окнам. Возврат из функции осуществляется только после обработки сообщения.

Чтобы заставить бота принимать сообщение, в конце файла нужно прописать: `bot.polling(none_stop=True)`. Параметр `none_stop` указывает, что принимать сообщения нужно будет постоянно.

3.2 Административная панель управления

Для создания телеграм-ботов на Python существует несколько десятков библиотек. Они различаются популярностью, размером комьюнити и функциональностью [22]. Рассмотрим самые популярные:

- `Aiogram` предлагает возможность выполнения кода асинхронно, что позволяет боту продолжать работу без ожидания ответа пользователя. Библиотека обладает подробной документацией и активным русскоязычным сообществом,

- `Python-telegram-bot`, с другой стороны, использует синхронный подход к выполнению кода, что останавливает его при ожидании ответа от пользователя,

- `TeleBot`, в свою очередь, предлагает возможность работы как с асинхронным, так и с синхронным подходом на выбор, что делает его удобным для небольших проектов.

Для реализации поставленных целей потребуется пять кнопок в меню:

- «Что я умею»,
- «Контакты»,
- «Аннотация к рабочей программе»,
- «Лабораторные работы»,
- «Заочная форма обучения».

Первое, что нужно сделать это импортировать библиотеку `telebot` и подключить токен бота:

```
import telebot
bot = telebot.TeleBot('НАШ ТОКЕН')
```

Далее рассмотрим основные команды для обработчика сообщений:

- `@bot.message_handler(commands=['start'])` – обработчик команды `"/start"`, выполняющийся при запуске бота,

– `@bot.message_handler(commands=['dist'])` – обработчик команды `"/dist"`, используется для рассылки,

– `@bot.message_handler(commands=['help'])` – обработчик команды `"/help"`, выводит основные команды бота,

– `@bot.message_handler(commands=['version'])` – обработчик команды `"/version"`, выводит сведения о версии бота.

Основными функциями-обработчика сообщений являются:

– `@bot.message_handler(content_types=["document"])` – обработчик получения аудио сообщений от пользователя,

– `@bot.message_handler(content_types=["audio"])` – обработчик получения документов от пользователя,

– `@bot.message_handler(content_types=['text', 'photo', 'sticker', 'animation', 'video'])` – обработчик всех остальных сообщений.

Функция `start` обрабатывает команду `/start` и регистрирует пользователя в базе данных: `id`, ник, фамилию и имя пользователя в Telegram. Так же создается клавиатура с кнопками `Reply Keyboard`. При нажатии на кнопку они будут выводить нам некий текст в чат или предлагать скачать файл. Пользовательский интерфейс бота представлен на рисунке 13.

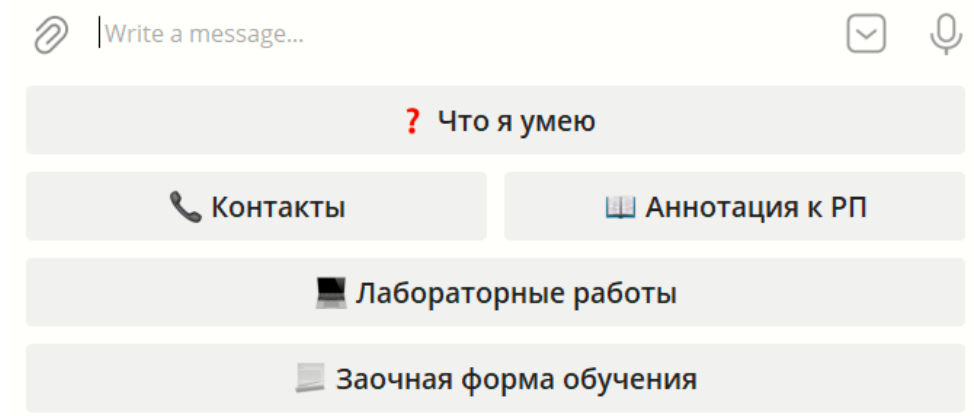


Рисунок 13 – Основное меню чат-бота

Для создания кнопок импортируем `from telebot import types`.

Рассмотрим функциональное назначение каждой строки кода.

`Message_handler` – это декоратор, который реагирует на входящие сообщения и содержит в себе функцию ответа.

Commands=['start'] – это команда, которая связана с декоратором и запускает вложенную в него функцию.

Объявление функции def handle_start(message), которая принимает один аргумент message. В данном случае, message представляет собой объект, содержащий информацию о входящем сообщении от пользователя.

Далее написан блок создания клавиатуры.

Создается объект клавиатуры ReplyKeyboardMarkup с шириной строки (row_width) равной 1. Это означает, что кнопка будет располагаться в одну колонку:

```
keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True,
row_width=1).
```

Создается кнопка с надписью '☐ Что я умею':

```
button1 = types.KeyboardButton('☐ Что я умею').
```

Отправка сообщения с клавиатурой обрабатывается следующим образом:

```
@bot.message_handler(content_types=['text', 'photo', 'sticker',
'animation', 'video'])
def handle_message(message):
    if message.text == "☐ Что я умею":
        bot.send_message(message.chat.id, "ТЕКСТ")
```

Пример реализации команды /start чат-бота изображен на рисунке 14.

```
@bot.message_handler(commands=['start'])
def handle_start(message):
    # Создание клавиатуры
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1)
    button1 = types.KeyboardButton('🔍 Что я умею')
    button2 = types.KeyboardButton('📁 Лабораторные работы')
    button3 = types.KeyboardButton('📞 Контакты')
    button4 = types.KeyboardButton('📄 Заочная форма обучения')
    button5 = types.KeyboardButton('📖 Аннотация к РП')
    keyboard.add(button1)
    keyboard.row(button3, button5)
    keyboard.add(button2, button4)

    bot.send_message(message.chat.id, 'Чем могу помочь?')
```

Рисунок 14 – Программный код команды /start

На рисунке 15 представлен скриншот программной обработки команды /version.

```
@bot.message_handler(commands=['version'])
def help_message(message):
    bot.send_message(message.chat.id, 'Бот-помощник 1.0 изучения '
                                     'дисциплины "Современные информационные '
                                     'технологии" студентами различных '
                                     'направлений ПИ им. В.Г. Белинского')
```

Рисунок 15 – Программный код команды /version

В разрабатываем боте предусмотрена рассылка от администратора по команде /dist всем пользователям по их id в БД. Код обработки команды представлен на рисунке 16.

```
@bot.message_handler(commands=['dist'])
def dist(message):
    sql.execute("SELECT chatid FROM users")
    results = sql.fetchall()
    for result in results:
        bot.send_message(result[0], "Рассылка: КТ2 будет с 6.05.24-12.05.24. "
                                   "\nНужно сдать лабораторные 4-6.")
    db.close()
```

Рисунок 16 – Программный код команды /dist

Разрабатываемый чат-бот может принимать документы (файлы в формате *.doc, *.pdf и т.д.), при этом в получаемом сообщении указывается id отправителя. Программный код и результат его выполнения представлен на рисунке 17.

```
@bot.message_handler(content_types=["audio"])
def handle_docs_document(message):
    bot.send_message(message.chat.id, "Файл получен")
    audio_id = message.audio.file_id
    file_info = bot.get_file(audio_id)
    bot.request.urlretrieve(f'http://api.telegram.org/file/bot{bot.token}/{file_info.file_id}')

print(BQACAgIAAxkBAAIY5mY3-Uf7RYEcYRxbYh8AAbuXm_9JFgAC7U4AAr9jwEm-8KZHYAFgLzUE
http://api.telegram.org/file/bot6111624795:AAGFB0NgaNvNrK09ZuUho0-l-dZ2zYJR1w4/documents/fi
BQACAgIAAxkBAAIY6GY3-wFuqn-Idb5PYU54idHg3L6vAALwTgACv2PASbWARY_egwABwzUE
http://api.telegram.org/file/bot6111624795:AAGFB0NgaNvNrK09ZuUho0-l-dZ2zYJR1w4/documents/fi
```

Рисунок 17 – Программный код команды /dist

Таким образом, была успешно разработана и запущена клиентская часть чат-бота, предназначенного для предоставления информационной поддержки студентам, изучающим определенный курс.

3.3 Проектирование и реализация базы данных чат-бота

Чтобы начать работу с SQLite в Python, нам потребуется библиотека `sqlite3`. Эта библиотека входит в стандартную библиотеку Python (начиная с версии 2.5).

Для начала работы с SQLite, давайте создадим новую базу данных и подключимся к ней:

```
import sqlite3

db = sqlite3.connect('users.db', check_same_thread = False)

sql = db.cursor().
```

БД чат-бота представляет собой таблицу SQLite3. Диаграмма структуры БД «users.db» представлена на рисунке 18.

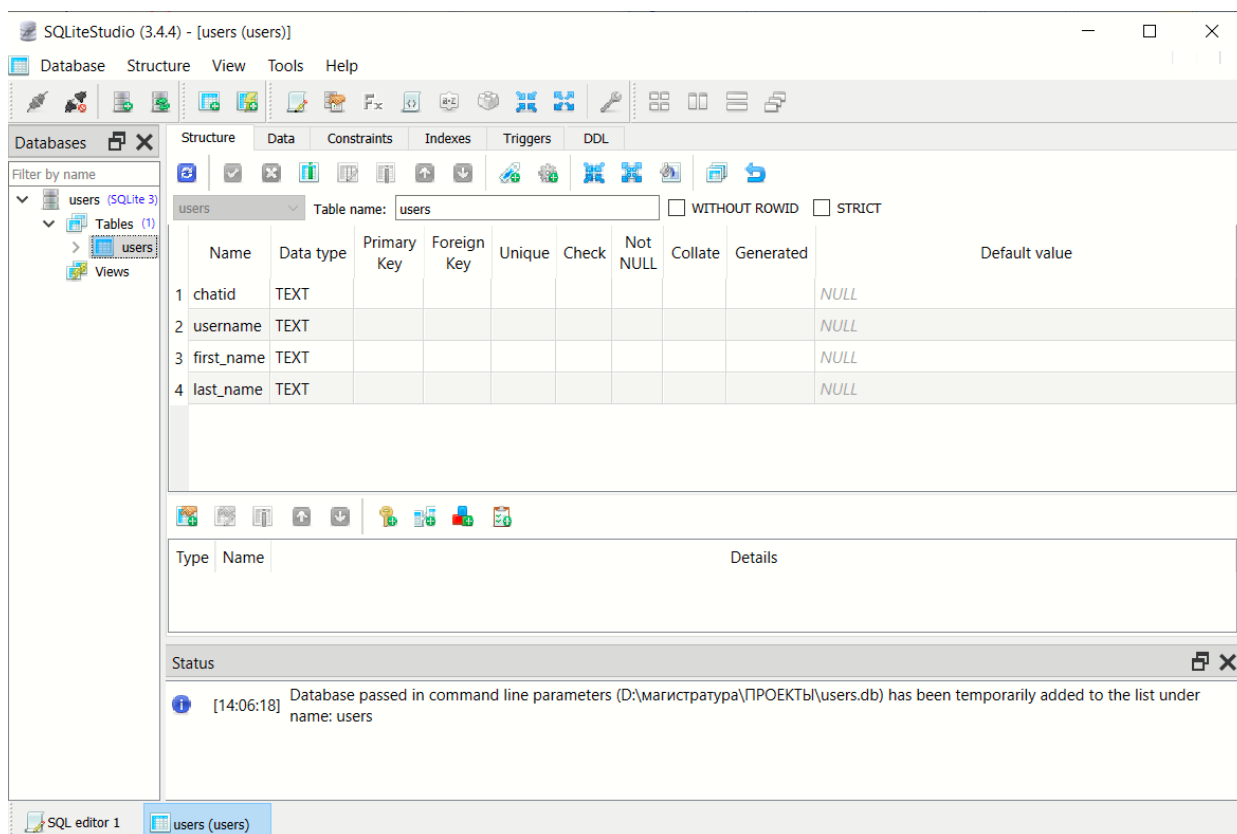


Рисунок 18 – Структура базы данных users.db

Алгоритм создания таблицы на Python, в которой будем хранить информацию о пользователях, он представлен на рисунке 19.

```
import sqlite3

db = sqlite3.connect('users.db', check_same_thread = False)
sql = db.cursor()

sql.execute("""CREATE TABLE IF NOT EXISTS users(
    chatid TEXT,
    username TEXT,
    first_name TEXT,
    last_name TEXT
)""")

db.commit()
```

Рисунок 19 – Создание БД на Python

Приведенная таблица выполняет следующие функции:

- user_id (уникальный id, который присваивается каждому пользователю, который запустил чат-бота),
- username (ник в мессенджере Telegram),
- first_name (имя пользователя в мессенджере Telegram),
- last_name (фамилия пользователя в мессенджере Telegram).

Содержимое БД «users.db» представлено на рисунке 20.

chatid	username	first_name	last_name
4	5759304410	NULL	Анастасия Евдокимова
5	961963610	Dimoonchikchel	DimitriP
6	5426624526	jare_jare_daze	Даниил Борискин
7	5571298373	NULL	Алина
8	5181719449	NULL	Ilya Blokhin
9	7130675057	NULL	Лиза К.
10	351844765	qkezz	кирил
11	2016249708	Peri190	Peri Turarbekova
12	1358673091	kateriness_14	Ekaterina
13	1120586406	sdobnickovaa	Арина
14	5325477229	NULL	Елена
15	977270409	Pyse4k_a	Алина Загартдинова

Рисунок 20 – Содержимое БД «users.db»

Таким образом, при нажатии пользователя команды на /start, его id вносится в БД, если его там не было ранее.

3.4 Модели машинного обучения для автоматизированной обработки обращений пользователя

Для эффективной работы серверной части чат-бота на платформе Telegram, необходимо решить следующие задачи:

- серверная часть должна принимать сообщения, отправленные пользователями, и анализировать их содержимое, включая текст, изображения, аудио или другие медиафайлы,
- полученная информация должна далее сравниваться с предопределенными шаблонами или правилами для определения подходящего ответа. Например, если вопрос связан с лабораторными работами, сервер должен распознать запрос и отправить информацию о лабораторных работах. Серверная часть должна также иметь доступ к базе данных с информацией о пользователях,
- на основе полученных данных сервер должен составить ответ и отправить его обратно пользователю в виде текста или другого формата. Ответ должен быть структурирован таким образом, чтобы удовлетворить запрос пользователя и предоставить необходимую информацию,

Для разработки интенгов пользователя и для дальнейшей их обработки, использовался модуль `json`. Программный код его обработки представлен на рисунке 21.

```
import json
with open("bot_ii.json", "r", encoding="utf-8") as config_file:
    data = json.load(config_file)
    INTENTS = data["intents"]
    print(INTENTS)
```

Рисунок 21 – Программный код обработка json-файла

Пример того, как может выглядеть файл json-файл представлен далее:

```
{
  "intents": {
    "lab": {
      "examples": ["нужно найти задание", "где взять задание"],

```

```

"responses": ["Перечень лабораторных работ и заданий
во вкладке 'Лабораторные работы'. Сдавать только преподавателю
на парах. Но можно отправить сюда файл на проверку сюда.",
"Посмотрите в меню 'Лабораторные работы'."]
},

```

...

В этом примере json-файл состоит из следующих разделов:

- название намерений «lab»,
- вопросы «examples»,
- ответы «responses».

Каждый раздел содержит массив возможных ответов. Раздел вопросы также содержит массив объектов, где каждый объект содержит вопрос и соответствующий ему ответ.

Далее рассмотрим как подготовить текстовые данные для подачи их в модель машинного обучения. На рисунке 22 представлены основные этапы предварительной обработки текстовых запросов пользователя в Python.



Рисунок 22 – Процесс обработки текста

Чаще всего применяются все эти шаги, но в разных задачах какие-то могут опускаться, поскольку приводят к потере информации [22].

Рассмотрим реализацию каждого шага в Python, при разработке чат-бота.

Шаг 1. Фильтрация символов текста:

- перевод всех символов к нижнему регистру, удаление пробелов и других символов, которые находятся в начале и конце строки, операции с

регулярными выражениями. Реализация данного вопроса представлена на рисунке 23,

– вычисление расстояния Левенштейна между двумя строками.

Программная реализация представлена на рисунке 24.

```
def filter_text(text):  
    text = text.lower()  
    text = text.strip()  
    pattern = r"^[^\w\s]"  
    text = re.sub(pattern, "", text)  
    return text
```

Рисунок 23 – Алгоритм реализации работы со строками

```
def text_match(user_text, example):  
    user_text = filter_text(user_text)  
    example = filter_text(example)  
    if user_text == example:  
        return True  
    if user_text.find(example) != -1:  
        return True  
    distance = nltk.edit_distance(user_text, example)  
    ratio = distance / len(example)  
    if ratio < 0.3:  
        return True  
    return False
```

Рисунок 24 – Алгоритм реализации вычисления расстояния между двумя строками

Шаг 2. Обработка естественного языка (удаление стоп-слов, лемматизация, стемминг, токенизация и векторизация).

Стоп-слова – это слова, которые относятся к категории наиболее часто встречающихся в языке, но при этом не несущих значительной смысловой нагрузки. К таким словам можно отнести частицы, предлоги и союзы, например, «и», «в», «на».

Лемматизация представляет собой процесс приведения словоформы к её нормализованной (словарной) форме, то есть лемме.

Стемминг представляет собой процесс поиска основы слова для заданного исходного слова.

Анализ текста – это основная область применения алгоритмов машинного обучения.

Токенизация представляет собой разбиение исходного текста на более мелкие единицы, которые могут быть как отдельными словами, так и предложениями.

Основная разница между ними кроется в том, что стемминг является более быстрым процессом, так как он осуществляется без необходимости учета контекста слов. Он просто режет слова, отбрасывая окончания, что позволяет ускорить обработку текста. В то же время, лемматизация, хотя и требует больше времени на выполнение, обладает более глубоким пониманием языка, так как она учитывает контекст слов, что позволяет более точно определить их основную форму, или лемму.

Класс `CountVectorizer` – инструмент, предоставляемый библиотекой `scikit-learn` на Python. Он используется для преобразования данного текста в вектор на основе частоты (количества) каждого слова, встречающегося во всем тексте. Это полезно, когда есть несколько таких текстов и необходимо преобразовать каждое слово в каждом тексте в векторы для использования в дальнейшем анализе текста. Этот класс также может отфильтровывать стоп-слова.

Программная реализация представлена на рисунке 25.

```

X = []
y = []
for intent in INTENTS:
    examples = INTENTS[intent]["examples"]
    for example in examples:
        example = filter_text(example)
        if len(example) < 2:
            continue
        X.append(example)
        y.append(intent)

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
vectorizer.fit(X)
vecX = vectorizer.transform(X)
print("vecX[459] = ",vecX[459])
print(" vecX=",vecX)

```

Рисунок 25 – Алгоритм векторизации текста

Фрагмент результата выполнения кода представлен на рисунке 26.

```

C:\Users\msic\AppData\Local\Programs\Python\
vecX[459] = (0, 245) 1
(0, 606) 1
vecX= (0, 619) 1
(1, 307) 1
(2, 617) 1
(3, 302) 1
(3, 344) 1
(4, 152) 1
(5, 668) 1
(6, 93) 1
(6, 561) 1
(7, 276) 1
(8, 338) 2

```

Рисунок 26 – Векторизация текста CountVectorizer

Шаг 3. Классификация текстов ML-моделью.

Классификация текстов является хорошо изученной задачей в области машинного обучения, однако большинство исследований проводится на

упрощенных условиях. В реальной жизни практически всегда встречается не менее 5 возможных намерений.

Эффективность такой модели зависит от различных факторов:

- оптимальным количеством намерений считается среднее количество от 5 до 10 для каждого конкретного случая. Меньшее число намерений приведет к упрощению задачи,

- объем и качество данных, как и в любой задаче машинного обучения, чем больше данных и чем более соответствующими они являются логическим запросам, тем лучше будут результаты,

- определить размер вводимых сообщений, так как пользователи не склонны кратко запрашивать бота, поэтому специализированные инструменты помогут приложению повысить точность.

В моделях ML обучения важную роль играют входные и выходные переменные. То, что подаётся на входы, принято называть признаками (англ. features). Эти признаки, как правило, являются аналогом переменных, используемых в научных исследованиях и экспериментах. Они описывают определенное явление и могут быть измерены количественно.

Когда эти признаки подаются на вход в модель ML, она старается выявить сходства и найти закономерности между ними.

На выходе от такого взаимодействия получаем определенный результат, который имеет своё уникальное название – метка. Метка может быть воспринята как некий «номерной знак», который присваивается выходам системы. Этот меткой представляет собой предположение или прогноз, который делает система на основе проведённого анализа признаков. Он показывает, в какую категорию, согласно результатам классификации, система решила отнести данный выход. Таким образом, процесс работы системы машинного обучения включает в себя не только анализ входных данных, но и генерацию выходных данных, которые в свою очередь могут быть использованы для дальнейшего анализа или принятия решений.

Классификация – самая распространенная задача машинного обучения на практике и относится к разделу обучения с учителем. Основная задача классификация заключается в том, чтобы эффективно и с высокой точностью обрабатывать значительные массивы информации, которые поступают на вход. Этот процесс предполагает не только сбор данных, но и их дальнейшее систематизированное распределения по различным категориям. Применение такой техники находит свое широкое применение в самых разнообразных областях. Например, в сфере распознавания изображений, где классификация помогает выделять и идентифицировать разнообразные объекты, такие как лица, различные виды животных, автомобильные номера. Кроме того, классификация находит свое применение и в работе с текстовыми данными. Благодаря классификации, можно обучать нейросети выполнять более сложные задачи.

Для ML на Python написано очень много библиотек. Далее рассмотрим одну из самых популярных – Scikit-Learn. Это библиотека для машинного обучения, которая предоставляет инструменты для создания моделей машинного обучения, их обучения и тестирования, которая включает в себя широкий набор алгоритмов для классификации, регрессии, кластеризации и других задач машинного обучения. Например, Logistic Regression, Random Forests, MLPClassifier и т.д. [23, 24, 25]

Scikit-learn предоставляет широкий спектр метрик для оценки качества моделей машинного обучения. Эти метрики позволяют оценить, насколько хорошо модель справляется с поставленной задачей. Например:

- precision – доля правильных ответов модели внутри конкретного класса, т.е. это отношение объектов, действительно принадлежащих данному классу, ко всем объектам, которые система отнесла к этому классу,

- recall – представляет долю истинно положительных классификаций, т.е. показатель показывает, насколько точно модель предсказывает объекты, которые действительно принадлежат к положительному классу,

- f1-score – представляет собой гармоническое среднее между Precision и Recall, обеспечивая единую оценку общей точности классификационной модели,

- accuracy – показывает процент правильных предсказаний, сделанных моделью,

- roc-auc является одной из наиболее значимых и информативных метрик в контексте решения задач бинарной классификации. ROC-кривая строится путем вычисления двух ключевых параметров для различных пороговых значений: True Positive Rate (TPR отражает долю правильно идентифицированных положительных случаев), и False Positive Rate (показывает долю неверно классифицированных отрицательных случаев). AUC, или площадь под ROC-кривой изменяется в диапазоне от 0 до 1, где значение 1 указывает на идеальную производительность модели, а значение 0.5 соответствует случайному угадыванию.

Важным этапом в процессе оценки модели ML является разделение исходных данных на обучающую и тестовую выборки. Этот шаг является ключевым, поскольку он позволяет модели учиться на одной части данных и проверять ее производительность на другой, независимой части. Обучающая выборка используется для обучения модели, в то время как тестовая выборка служит для оценки ее производительности. Обычно данные разделяются в пропорции 70-80% на обучение и 20-30% на тестирование.

В языке программирования Python для выполнения данной операции можем воспользоваться библиотекой `sklearn`, которая предоставляет функцию `train_test_split` [26, 27]. При использовании этой функции необходимо указать переменные `X` и `y`, которые представляют собой исходные данные и метки классов соответственно. Кроме того, параметр `random_state` гарантирует одинаковое разделение данных при каждом запуске программы. Программный код, реализующий разделение данных на обучающую и тестовую выборки, представлен на рисунке 27.

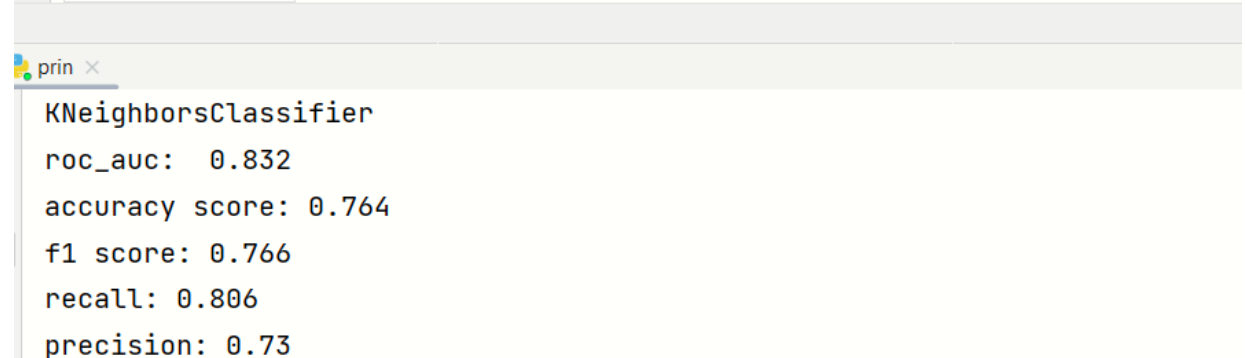
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(vecX, y, test_size=0.2, random_state=42)
```

Рисунок 27 – Пример разделения данных на обучающую
и тестовую выборки

Далее рассмотрим некоторые модели ML на примере обработки данных, подготовленных для разрабатываемого бота [28, 29, 30].

KNeighborsClassifier (метод k-ближайших соседей) – это популярный алгоритм классификации, который используется в разных типах задач машинного обучения. Данный подход основан на определении минимального расстояния между объектом, который необходимо классифицировать, и ближайшими классифицированными объектами из обучающего набора. Объект будет отнесен к классу, к которому принадлежит ближайший объект из набора. Реализация модели KNeighborsClassifier и вычисления метрик точности представлено на рисунке 28.

```
print("KNeighborsClassifier")
from sklearn.neighbors import KNeighborsClassifier
model_KN = KNeighborsClassifier()
y_scor_KN=model_KN.fit(X_train, y_train)
y_score_KN=y_scor_KN.predict(X_test)
y_pred_prob_KN = model_KN.predict_proba(X_test)[:, 1]
roc_auc_KN = roc_auc_score(y_test, y_pred_prob_KN)
print("roc_auc: ",round(roc_auc_KN,3))
print(f"accuracy score: {round(accuracy_score(y_test, y_score_KN),3)}")
print(f"f1 score: {round(f1_score(y_test, y_score_KN),3)}")
print(f"recall: {round(recall_score(y_test, y_score_KN),3)}")
print(f"precision: {round(precision_score(y_test, y_score_KN),3)}")
```



```
KNeighborsClassifier
roc_auc: 0.832
accuracy score: 0.764
f1 score: 0.766
recall: 0.806
precision: 0.73
```

Рисунок 28 – Реализация алгоритма KNeighborsClassifier

Данный алгоритм показал точность верного ответа равную 76,4%, и площадь под ROC-кривой 0,83. В результате использования алгоритма KNeighborsClassifier можно говорить о том, что данный метод не достаточно хорошо определяет классы

SVM (Support Vector Machines, метод опорных векторов) – это весьма эффективный алгоритм машинного обучения, применяемый для решения разнообразных задач классификации. Его основной принцип заключается в построении гиперплоскости в n-мерном пространстве с целью разделения объектов, принадлежащих двум или более классам. При этом гиперплоскость подбирается таким образом, чтобы максимизировать расстояние между ней и ближайшими объектами различных классов, что называется зазором. Те объекты, что находятся ближе всего к гиперплоскости, получили название опорных векторов. Одним из ключевых принципов работы SVM является преобразование данных в пространство более высокой размерности, где объекты могут быть легче разделены гиперплоскостью. Этот процесс осуществляется с помощью ядра, которое позволяет провести нелинейное преобразование данных, сохраняя при этом высокую вычислительную эффективность.

Обычно SVM формулируется как задача квадратичного программирования, где основной задачей является минимизация функции потерь и максимизация зазора между классами. Этот метод позволяет алгоритму работать эффективно и точно, обеспечивая качественную классификацию данных.

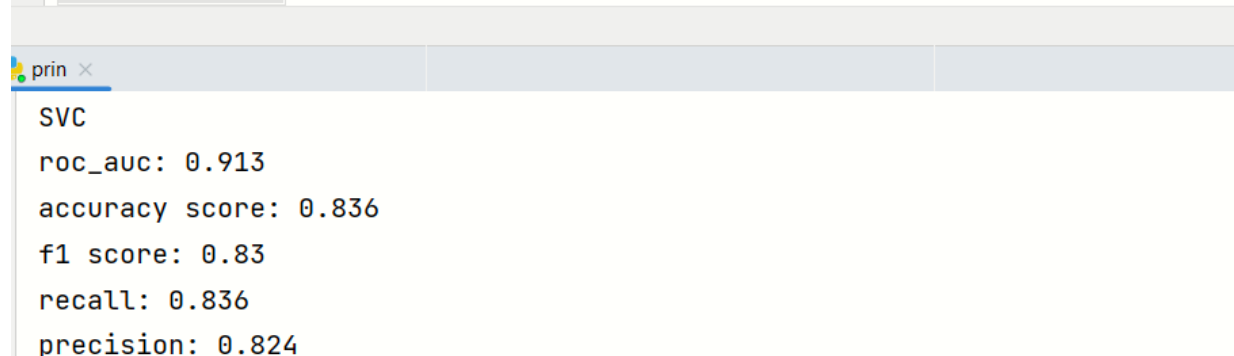
SVM может быть эффективен при обучении на небольшом объеме данных, что делает его востребованным инструментом в мире машинного обучения. После предварительной обработки данных модель SVM строится на основе обучающей выборки. Она может использоваться для прогнозирования значений зависимой переменной для новых наблюдений.

Одним из преимуществ метода опорных векторов является высокая точность прогнозирования, особенно если данные хорошо обработаны и

содержат значимые признаки. Кроме того, SVM может использоваться для решения задач классификации, регрессии и обнаружения выбросов. Однако, следует учитывать, что метод опорных векторов может быть громоздким и требовательным к вычислительным ресурсам, особенно при работе с большими объемами данных.

Реализация алгоритма SVC и вычисления метрик точности представлена на рисунке 29.

```
print("SVC")
from sklearn.svm import SVC
svc = SVC(probability=True)
y_svc = svc.fit(X_train, y_train)
y_score_svc=y_svc.predict(X_test)
y_pred_prob_svc = svc.predict_proba(X_test)[: , 1]
roc_auc_svc= roc_auc_score(y_test, y_pred_prob_svc)
print(f"roc_auc: {round(roc_auc_svc,3)} ")
print(f"accuracy score: {round(accuracy_score(y_test, y_score_svc),3)}")
print(f"f1 score: {round(f1_score(y_test, y_score_svc),3)}")
print(f"recall: {round(recall_score(y_test, y_score_svc),3)}")
print(f"precision: {round(precision_score(y_test, y_score_svc),3)}")
```



The screenshot shows a Jupyter Notebook interface. The top part displays the Python code for training and evaluating an SVC model. The bottom part shows the output of the code, which includes the model name 'SVC' and five performance metrics: roc_auc (0.913), accuracy score (0.836), f1 score (0.83), recall (0.836), and precision (0.824).

Рисунок 29 – Реализация алгоритма SVC

Данный алгоритм показал точность верного ответа равную 83,6%. Площадь под кривой составила 0,913, это говорит о достаточно высокой точности классификатора.

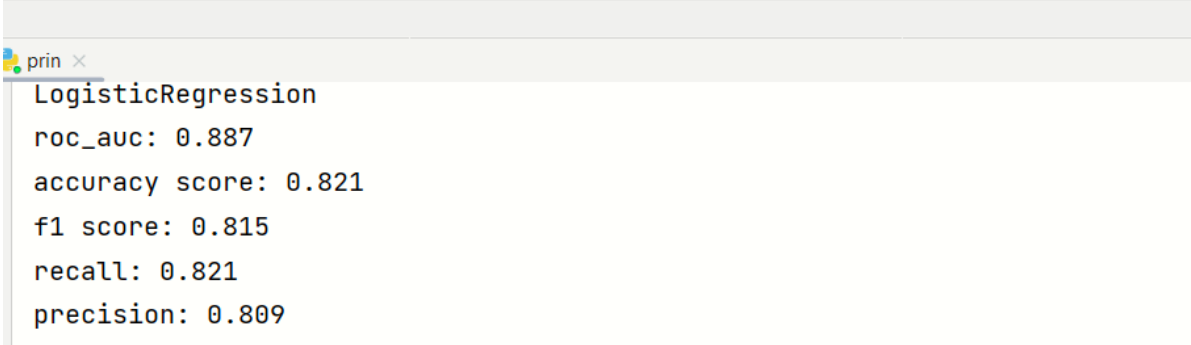
Logistic Regression (логистическая регрессия) – это способ определения зависимости между переменными, одна из которых категориально зависима, а другие независимы. Данную функцию можно использовать для задачи оценки вероятности принадлежности входного вектора к определенному

классу. После того, как набор данных был преобразован в числовые векторы признаков, необходимо приступить к обучению модели.

Для обучения логистической регрессии необходимо использовать библиотеку `sklearn.linear_model`. Далее необходимо посчитать точность (количество верных прогнозов, деленное на общее количество выборок, т.е. часть примеров, которые были правильно предсказаны моделью) на тестовой выборке.

Реализация модели Logistic Regression и вычисления метрик точности представлена на рисунке 30.

```
print("LogisticRegression")
from sklearn.linear_model import LogisticRegression
model_Log = LogisticRegression()
y_scor_Log=model_Log.fit(X_train, y_train)
y_score_Log=y_scor_Log.predict(X_test)
y_pred_prob = model_Log.predict_proba(X_test)[: , 1]
roc_auc_Log = roc_auc_score(y_test, y_pred_prob)
print(f"roc_auc: {round(roc_auc_Log,3)} ")
print(f"accuracy score: {round(accuracy_score(y_test, y_score_Log),3)}")
print(f"f1 score: {round(f1_score(y_test, y_score_Log),3)}")
print(f"recall: {round(recall_score(y_test, y_score_Log),3)}")
print(f"precision: {round(precision_score(y_test, y_score_Log),3)}")
```



The screenshot shows a Jupyter Notebook interface. The top part contains a code cell with Python code for training and evaluating a Logistic Regression model. The bottom part shows the output of the code, which includes the model name and various performance metrics. The output is displayed in a text area with a tab labeled 'prin'.

```
LogisticRegression
roc_auc: 0.887
accuracy score: 0.821
f1 score: 0.815
recall: 0.821
precision: 0.809
```

Рисунок 30 – Реализация модели LogisticRegression

Данный алгоритм показал точность диагностики равную 82,1%, и площадь под ROC-кривой 0,88. Данный результат является удовлетворительным для определения класса интента пользователя.

Random Forest (метод случайного леса) – это алгоритм машинного обучения, который используется для решения задач классификации и регрессии. В случае классификации каждый лист дерева определяет класс, к

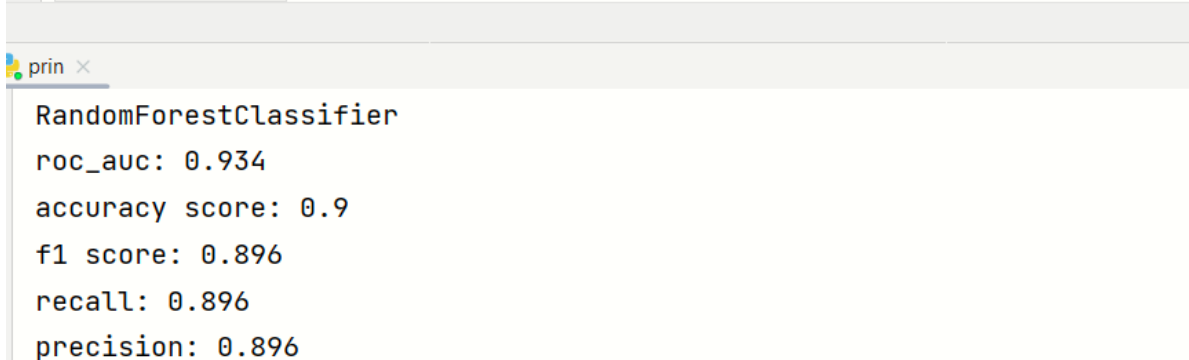
которому относится соответствующее наблюдение. Она использует все признаки и настроена с использованием значений, задаваемых по умолчанию.

Обучение модели случайного леса реализуется по следующему алгоритму:

- для построения модели случайного леса также сначала задаются входы и выходы и формируются тестовая и обучающая выборки,
- чтобы обучить модель случайного леса используется библиотека `sklearn.ensemble`,
- вычислить точность случайного леса.

Метод деревьев решений имеет множество преимуществ, в том числе простоту и удобство интерпретации полученных результатов, а также возможность обрабатывать наборы данных со множеством признаков. Реализация модели `RandomForestClassifier` и вычисления метрик точности представлена на рисунке 31.

```
print("RandomForestClassifier")
from sklearn.ensemble import RandomForestClassifier
model_rF = RandomForestClassifier()
y_scor=model_rF.fit(X_train, y_train)
y_score=y_scor.predict(X_test)
y_pred_prob_rF = model_rF.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, y_pred_prob_rF)
print(f"roc_auc: {round(roc_auc,3)} ")
print(f"accuracy score: {round(accuracy_score(y_test, y_score),3)}")
print(f"f1 score: {round(f1_score(y_test, y_score),3)}")
print(f"recall: {round(recall_score(y_test, y_score),3)}")
```



```
RandomForestClassifier
roc_auc: 0.934
accuracy score: 0.9
f1 score: 0.896
recall: 0.896
precision: 0.896
```

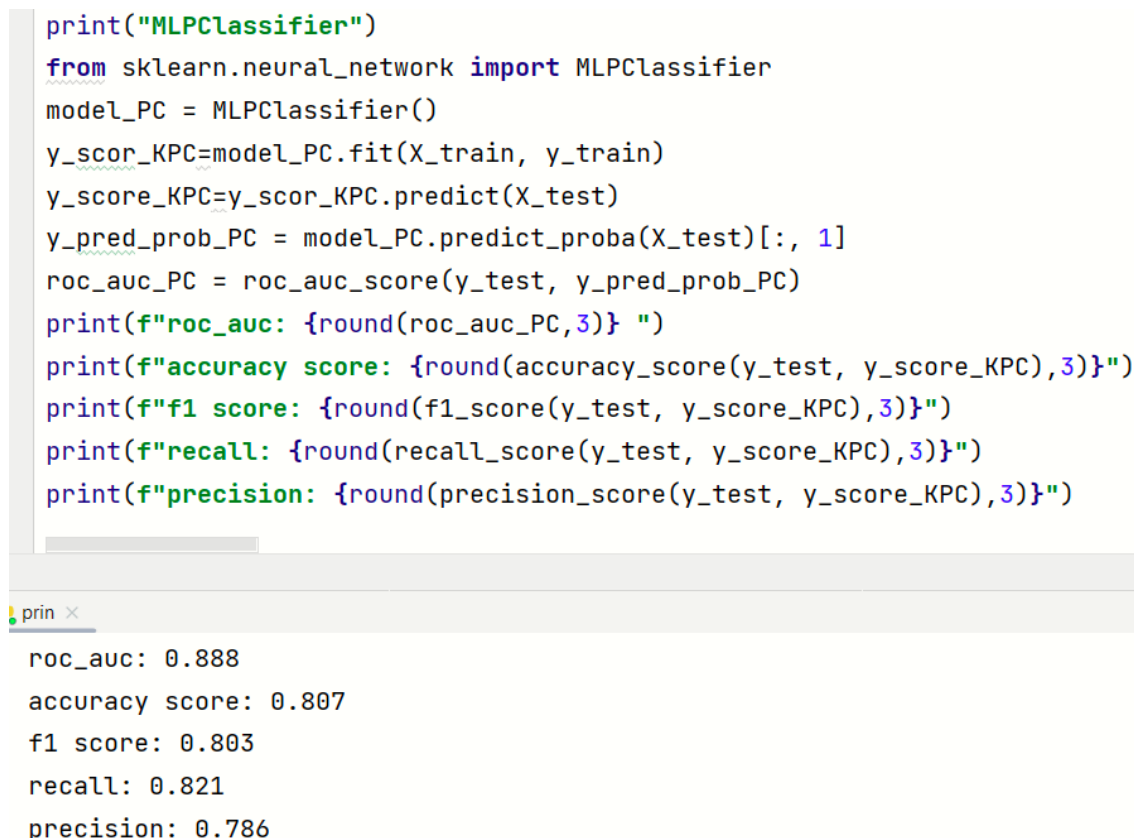
Рисунок 31 – Реализация модели `RandomForestClassifier`

Данный алгоритм показал точность диагностики равную 90,0%, и площадь под ROC-кривой 0,93. Данный результат является достаточно высоким.

MLPClassifier – это класс в библиотеке Scikit Learn для обучения многослойной нейронной сети персептрона (MLP) для задач классификации. Он используется для обучения нейронной сети с несколькими скрытыми слоями для решения задач классификации. Нейронные сети являются моделями машинного обучения, которые моделируют работу мозга и используются для обработки сложных и неструктурированных данных. MLPClassifier является одним из наиболее распространенных методов для решения задач машинного обучения, особенно в области работы с изображениями, текстом и другими типами данных.

MLP – это тип нейронной сети с прямой связью, которая состоит из нескольких слоев нейронов, каждый из которых соединен со следующим.

Реализация модели MLPClassifier и вычисления метрик точности представлена на рисунке 32.



```
print("MLPClassifier")
from sklearn.neural_network import MLPClassifier
model_PC = MLPClassifier()
y_scor_KPC=model_PC.fit(X_train, y_train)
y_score_KPC=y_scor_KPC.predict(X_test)
y_pred_prob_PC = model_PC.predict_proba(X_test)[: , 1]
roc_auc_PC = roc_auc_score(y_test, y_pred_prob_PC)
print(f"roc_auc: {round(roc_auc_PC,3)} ")
print(f"accuracy score: {round(accuracy_score(y_test, y_score_KPC),3)}")
print(f"f1 score: {round(f1_score(y_test, y_score_KPC),3)}")
print(f"recall: {round(recall_score(y_test, y_score_KPC),3)}")
print(f"precision: {round(precision_score(y_test, y_score_KPC),3)}")
```

roc_auc: 0.888
accuracy score: 0.807
f1 score: 0.803
recall: 0.821
precision: 0.786

Рисунок 32 – Реализация модели MLPClassifier

Данный алгоритм показал точность диагностики равную 80,7%, и площадь под ROC-кривой 0,89. Это говорит о том, что данный метод достаточно хорошо определяет классы.

Таким образом, модель RandomForestClassifier имеет более высокую точность по сравнению с остальными рассмотренными моделями ML и можно сделать вывод, что она делает правильные прогнозы в большинстве случаев.

Выведем classification report модели, которая будет содержать текстовые отчеты по основным показателям классификации RandomForestClassifier. Скриншот представлен на рисунке 33.

	precision	recall	f1-score	support
0	0.89	0.90	0.90	73
1	0.89	0.88	0.89	67
accuracy			0.89	140
macro avg	0.89	0.89	0.89	140
weighted avg	0.89	0.89	0.89	140

Рисунок 33 – Результат функции classification_report

Сообщаемые средние значения (reported averages include) включают:

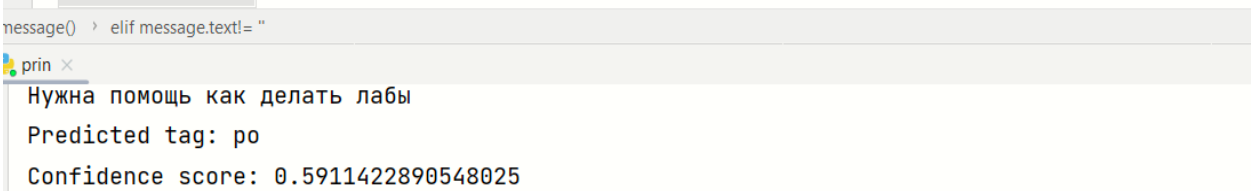
- macro average (самый простой из многочисленных методов усреднения, усреднение невзвешенного среднего по метке),
- weighted average (усреднение средневзвешенного значения по метке) и выборочное среднее значение (только для классификации по нескольким меткам),
- micro average (усреднение общего количества истинно положительных результатов, ложноотрицательных и ложноположительных результатов).

В основном коде программы так же используется метод predict() для прогнозирования категории для набора входных объектов на основе обученной модели. Он возвращает дискретное значение, которое может быть непосредственно присвоено каждому входному объекту.

С другой стороны, `predict_proba()` – метод возвращает предсказанные вероятности входных объектов, принадлежащих к каждой категории. Метод, вместо того, чтобы возвращать отдельный класс, возвращает вероятности, связанные с каждым классом.

Пример программы и результат ее выполнения представлены на рисунке 34.

```
print(message.text)
X_user = vectorizer.transform([message.text])
predicted_tag = model_1.predict(X_user)[0]
confidence_score = model_1.predict_proba(X_user)[0][model.classes_ == predicted_tag][0]
print("Predicted tag:", predicted_tag)
```



```
message() > elif message.text!= "
prin x
Нужна помощь как делать лабы
Predicted tag: po
Confidence score: 0.5911422890548025
```

Рисунок 34 – Применение метода `predict()`

В заключение отметим, что в ML широко применяются различные метрики для анализа и оценки работы моделей, а также для сравнения эффективности различных алгоритмов. Метрики играют важную роль в процессе обучения и тестирования моделей, помогая понять, насколько хорошо модель справляется с поставленными задачами и насколько она точно предсказывает результаты.

3.5 Результаты разработки чат-бота

Ниже приведены результаты разработки чат-бота для мессенджера Telegram и его функции.

С самого начала взаимодействия с ботом пользователю предлагается воспользоваться командой `/start`. При вызове данной команды пользователю автоматически отправляется приветственное сообщение.

Результат представлен на рисунке 35.

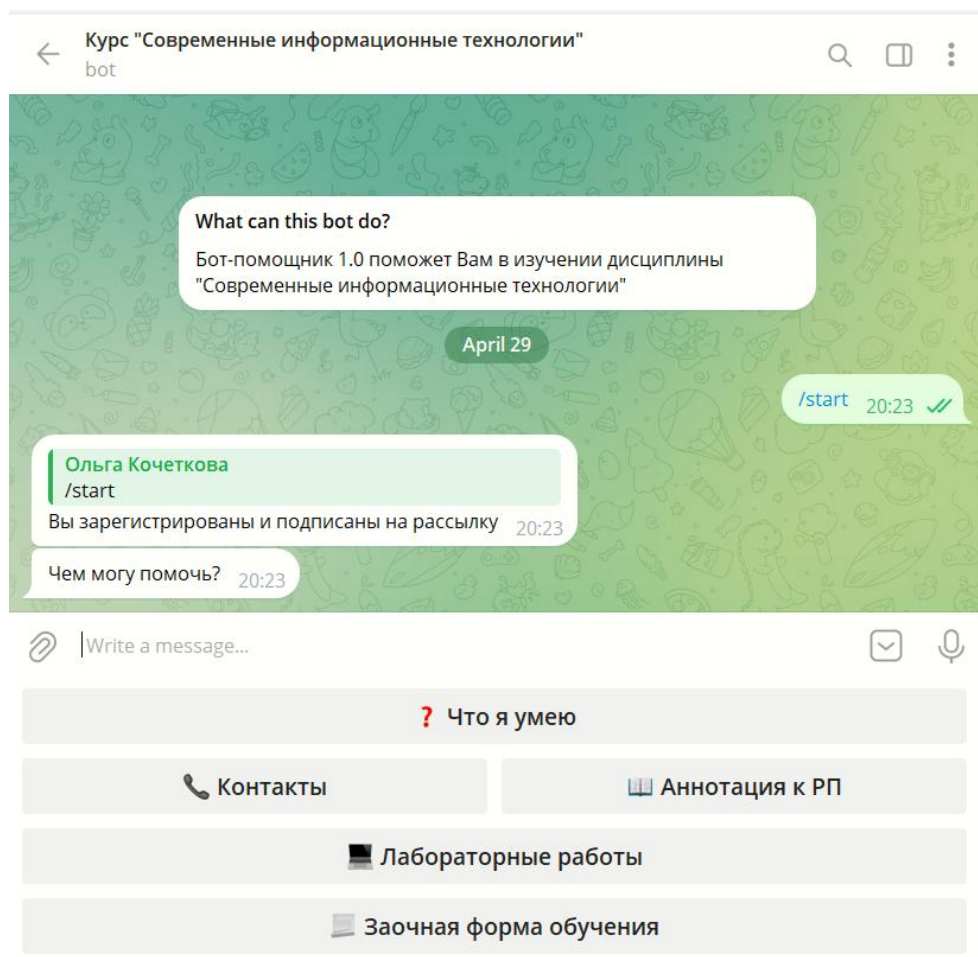


Рисунок 35 – Ответ на команду /start и главное меню чат-бота

Так же отображается главное меню, в котором на выбор есть 5 кнопок:

- «Что я умею»,
- «Контакты»,
- «Аннотация к рабочей программе»,
- «Лабораторные работы»,
- «Заочная форма обучения».

Есть возможность воспользоваться справкой, вызвав команду /help – команда, выводящая справку по боту и список команд. Результат представлен на рисунке 36.

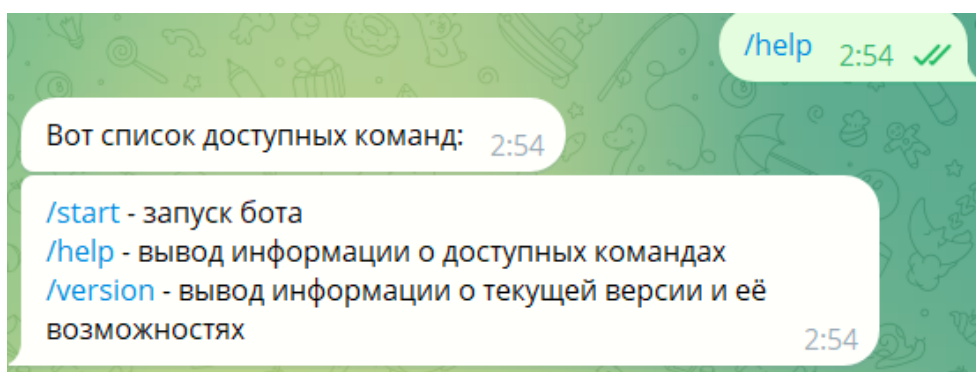


Рисунок 36 – Ответ на команду /help

Команда /version выводит сведения о чат-боте. Результат представлен на рисунке 37.

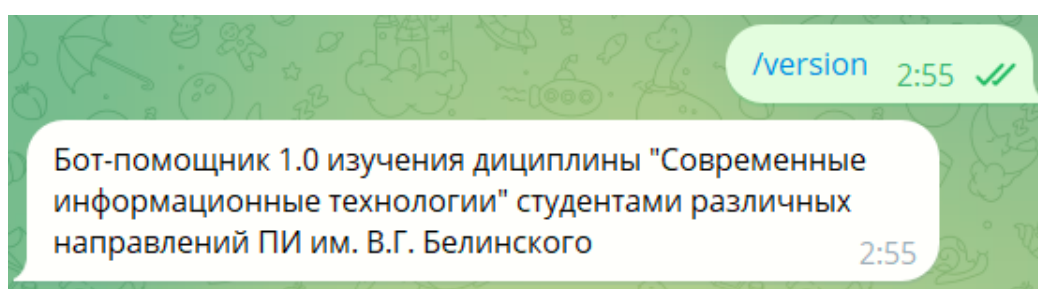


Рисунок 37 – Ответ на команду /version

Следующая функция «Что я умею» показывает основное назначение разработанного бота.

Далее на рисунке 38 представлен ответ на команду «Что я умею».

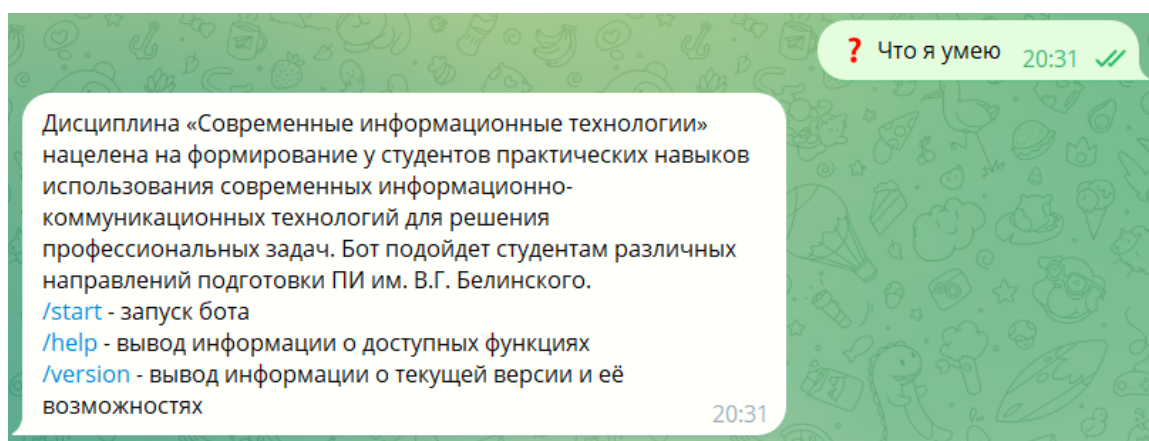


Рисунок 38 – Ответ на команду «Что я умею»

Меню «Контакты» выводит сведения о кафедре, преподаватели которой ведут соответствующую дисциплину.

Далее на рисунке 39 представлен ответ на команду «Контакты».

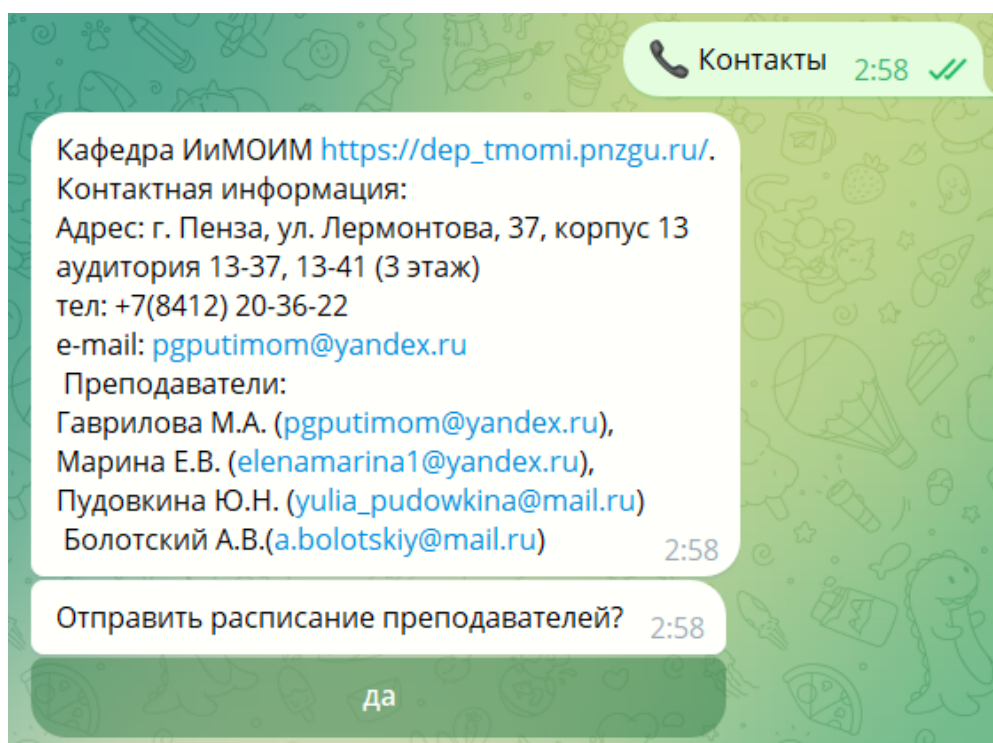


Рисунок 39 – Ответ на команду «Контакты»

Меню «Аннотация к РП» выводит сведения о дисциплине «Современные информационные технологии». Далее на рисунке 40 представлен ответ на команду «Аннотация к РП».

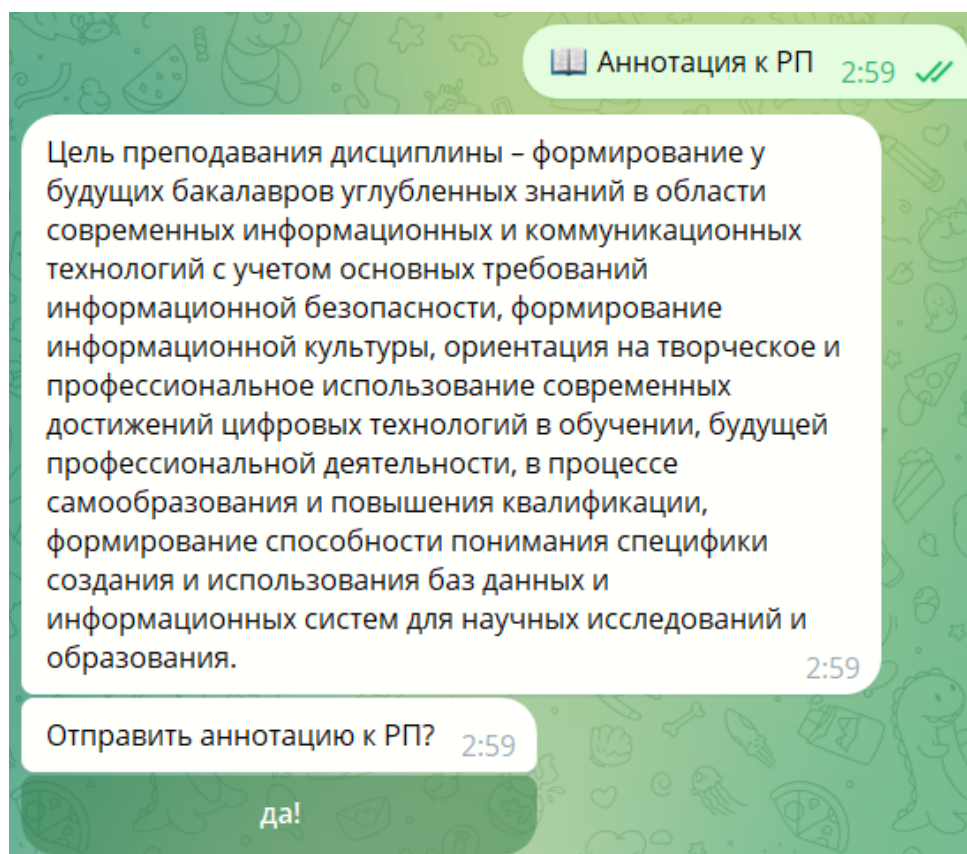


Рисунок 40 – Ответ на команду «Аннотация к РП»

Меню «Лабораторные работы» выводит задания, которые необходимо выполнить и сдать в течении семестра. Они охватывают различные темы и требуют от студентов не только теоретических знаний, но и практических навыков.

Далее на рисунке 41 представлен ответ на команду «Лабораторные работы».

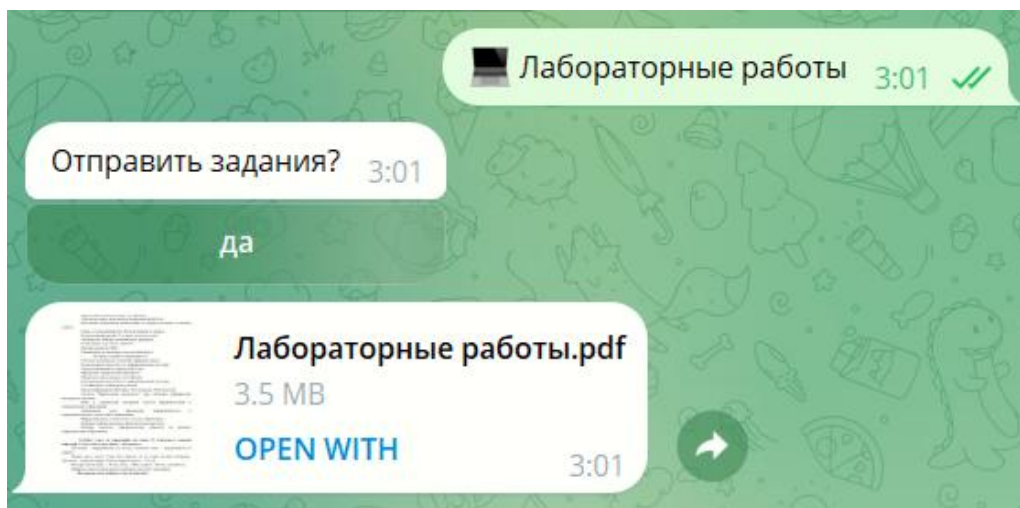


Рисунок 41 – Ответ на команду «Лабораторные работы»

Меню «Заочная форма обучения» предоставляет студентам доступ к заданиям на лабораторные работы, которые они должны выполнить в рамках подготовки к зачету. Кроме того, в этом меню можно найти список вопросов, которые могут быть заданы на зачете. Пользователь также может получить информацию о процедуре итогового тестирования, что поможет ему лучше подготовиться к этому этапу обучения

Далее на рисунке 42 представлен ответ на команду «Заочная форма обучения».

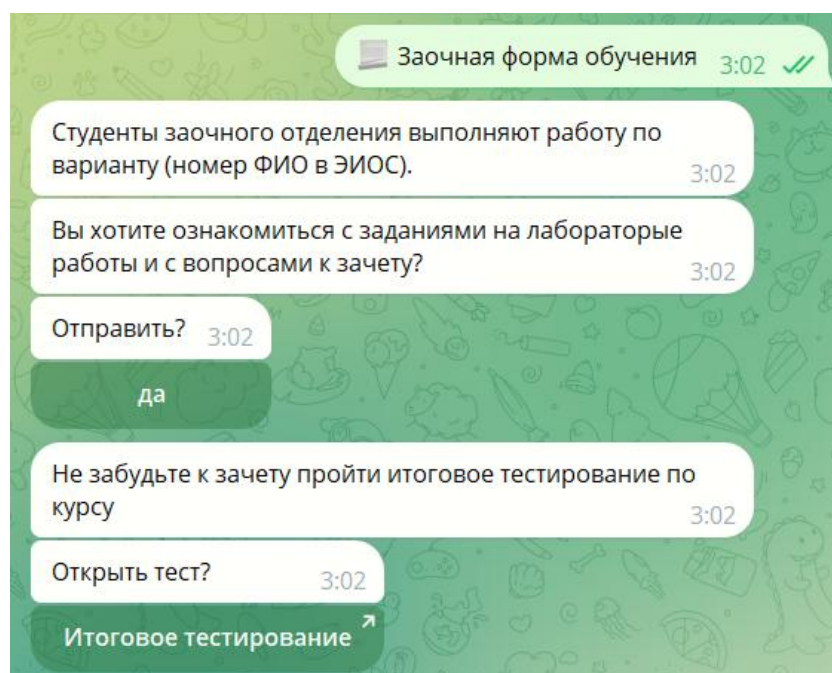


Рисунок 42– Ответ на команду «Заочная форма обучения»

На рисунке 43 представлена реакция бота, если пользователь отправил текстовый файл.



Рисунок 43– Ответ бота на полученный файл в формате *.doc

При этом администратору будет отправлено сообщение:

BQACAgIAAxkBAAIY5mY3-Uf7RYEcYRxbYh8AAbuXm_9JFgAC7U4
AAr9jwEm-8KZHYAFgLzUE http://api.telegram.org/file/bot6111624795:AAGFBONgANvNrK09ZuUhoO-l-dZ2zYJR1w4/documents/file_3.docx.

Далее на рисунках 44 и 45 представлен диалог между пользователем и разработанным чат ботом на основе ИИ.

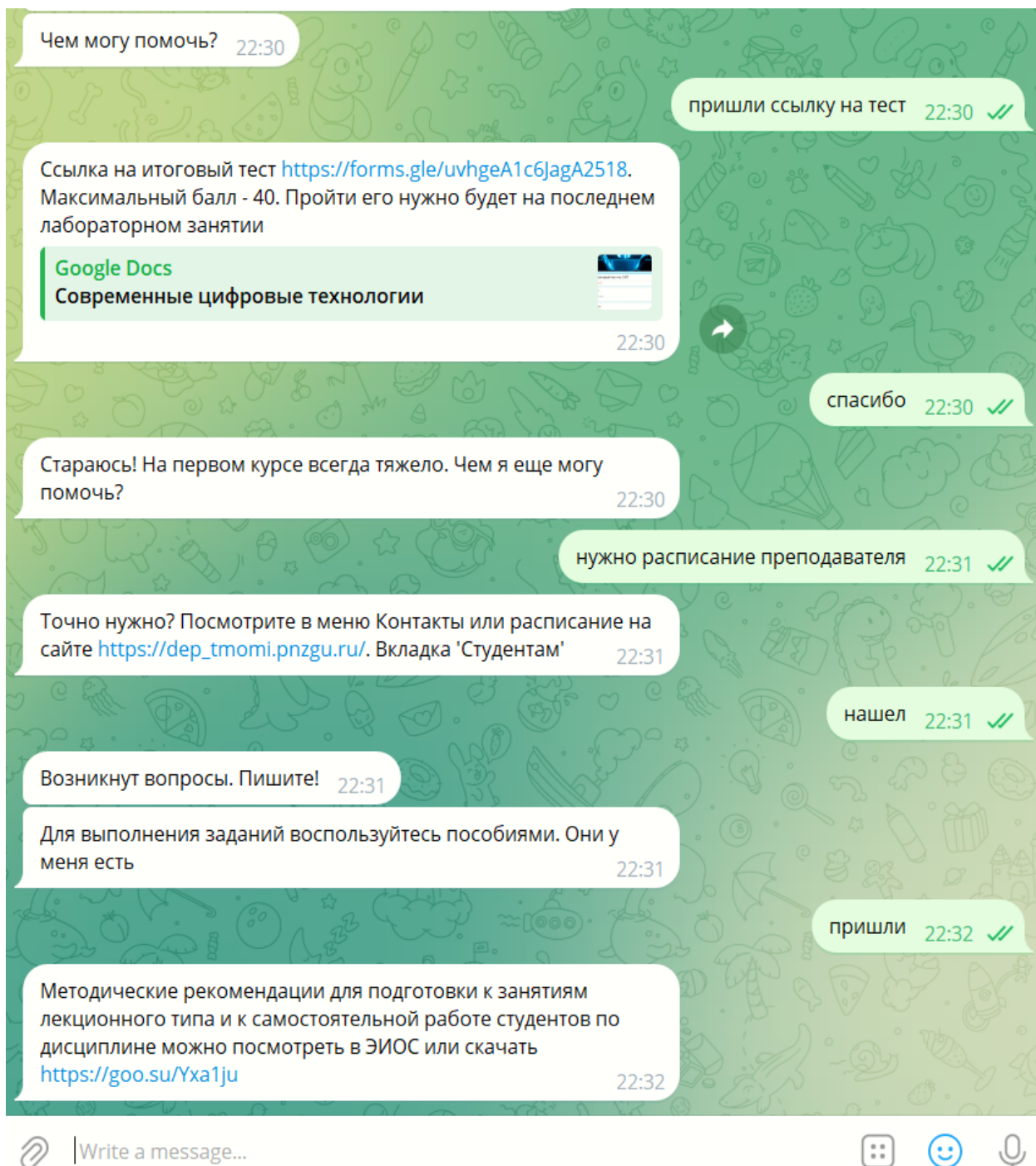


Рисунок 44 – Пример диалога между пользователем и ботом

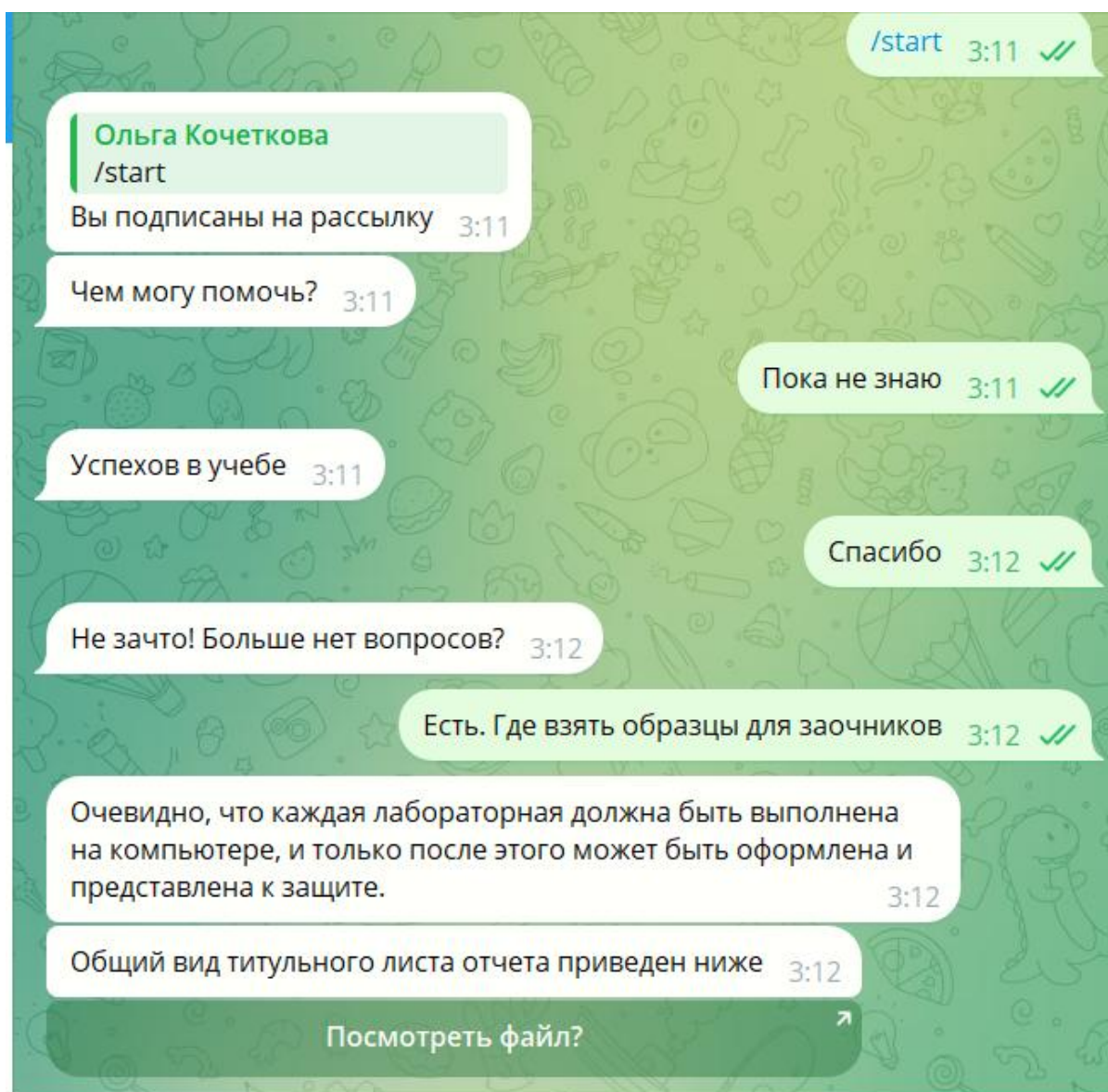


Рисунок 45 – Пример диалога между пользователем и ботом

Таким образом, в основе исследования по разработке чат-бота на основе искусственного интеллекта лежит процесс предварительной обработки естественного языка, в том числе и на основе алгоритмов машинного обучения.

3.6 Выводы

В рамках данного раздела был разработан и заложен функционал, который будет являться основой для дальнейшего создания продукта, представлено общее описание модели разрабатываемого проекта, проведен анализ моделей машинного обучения и дана оценка их точности. Кроме того, дано описание итогового приложения, которое реализует пользовательский

интерфейс для взаимодействия с ботом. Процесс разработки завершился итоговым функциональным тестированием, в ходе которого были проверены все ключевые функции и возможности программной системы.

Результат исследования – это разработанный бот, который, несомненно, станет чрезвычайно полезным и необходимым инструментом в структуре современного образовательного процесса. Этот бот не только обеспечивает эффективность и удобство в работе, но и является привлекательным и инновационным решением для пользователей разных уровней квалификации, в том числе для обучающихся и преподавателей.

ЗАКЛЮЧЕНИЕ

В современном мире одним из наиболее динамично развивающихся средств общения является чат-бот, который используется для оперативного обмена короткими сообщениями. Чат-боты находят применение в различных сферах деятельности. Они помогают автоматизировать рутинные задачи в бизнесе, упрощают процесс подбора персонала, предоставляют техническую поддержку на мероприятиях и в компаниях. Кроме того, чат-боты активно используются в образовательной сфере на различных уровнях обучения.

В сфере образования чат-боты выполняют множество полезных функций. Они могут стать надежными помощниками преподавателей, администрации учебных заведений и вспомогательного персонала. Чат-боты не только помогают в выполнении учебных задач, но и обеспечивают поддержку в организации воспитательной работы, создавая комфортное и продуктивное образовательное окружение. Взаимодействие с ними позволяет эффективно использовать современные технологии в образовательном процессе, повышая его доступность и качество.

Данная выпускная квалификационная работа посвящена разработке и реализации чат-бота на платформе Telegram для информационной поддержки студентов 1 курса очной и заочной формы обучения, изучающие дисциплину «Современные информационные технологии». В ходе исследования и разработки успешно достигнута поставленная цель и получен ценный опыт в области разработки чат-ботов на алгоритмах и методах ИИ.

Для создания чат-бота в данной выпускной квалификационной работе были задействованы современные технологии и инструменты: интегрированная среда разработки PyCharm и язык программирования Python, библиотеки для реализации алгоритмов ML и AI и для взаимодействия с Telegram Bot API.

Результаты магистерской диссертации были апробированы в докладах на конференциях и при публикации в научном журнале:

1. Чат-бот для сферы образования / Инжиниринг и технологии. 2023. Т. 8. № 2. С. 19-21.

2. Предварительная обработка текста для чат-бота / X Всероссийская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы» («ВНПК-2023») (ПГУ, 15-17 апреля 2023 г.)

3. Инструментальные средства разработки чат-ботов / «Актуальные проблемы науки и образования», посвященная 80-летию юбилею Пензенского государственного университета (19-25 октября 2023 г.)

4. Особенности разработки бота с искусственным интеллектом / VII Международная научно-практическая конференция студентов и молодых ученых «Модели, системы, сети в технике», посвященной 90-летию со дня рождения В. И. Пацаева (ПГУ, 24 марта 2024 г.).

5. Методические рекомендации по изучению темы искусственного интеллекта в школе / XX Всероссийской с международным участием научно-практической конференции «Артемовские чтения»: «Современное образование: научные подходы, опыт, проблемы, перспективы» (Педагогический институт им. В. Г. Белинского ПГУ, 17-18 апреля 2024 г.).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Волков С.В., Волков А.С. Использование чат-ботов в современном образовании. // Большие данные в образовании: сборник статей по итогам Международной конференции (Москва, 29–31 августа 2019 года). – М.: Экон-Информ, 2020. – С. 22–27.
2. Горячкин Б.С., Галичий Д.А., Цапий В.С., Бурашников В.В., Крутов Т.Ю. Эффективность использования чат-ботов в образовательном процессе // E-Scio, 2021. – № 4 (55). – С. 529-551.
3. Тараканова К.А. Использование чат-ботов для организации образовательного процесса // Материалы XIV Международной студенческой научной конференции «Студенческий научный форум» [Электронный ресурс]. – URL: <https://scienceforum.ru/2022/article/2018029490> (дата обращения: 10.03.2023).
4. Буяковская И.А., Дробахина А.Н. Чат-боты в организации воспитательной работы в вузе // Национальная конференция «Актуальные вопросы современной науки: теоретические и практические аспекты». – Новокузнецк: НФИ КемГУ, 2020 – С. 27-30.
5. Матвеева Н.Ю., Золотарюк А.В. Технологии создания и применения чат-ботов // Научные записки молодых исследователей. – 2018. – №1. – с. 28-30.
6. Лучшева Л.В. Социальные проблемы использования искусственного интеллекта в высшем образовании: задачи и перспективы // Научный Татарстан. – 2020. – № 4. – С. 84–89.
7. Жеребцова Ю.А., Чижик А.В. Создание чат-бота: обзор архитектур и векторных представлений текста // International Journal of Open Information Technologies. – 2020. – Т. 8, № 7. – С. 50-56.
8. Что такое бот в телеграмме и чем отличается от обычного пользователя? [Электронный ресурс]. – URL: <https://scienceforum.ru/2022/article/2018029490> (дата обращения: 20.01.2023).

9. Рассел С., Норвиг П. Искусственный интеллект: современный подход. – Москва: ДМК Пресс, 2019. – 1168 с.
10. Чулюков В.А., Астахова И.Ф., Потапов А.С., Каширина И.Л. Системы искусственного интеллекта. Практический курс: учебное пособие. – М.: Бином. Лаборатория знаний, 2017. – С. 292.
11. Фирсова, Е.А. Перспективы использования чат-ботов в высшем образовании // Информатизация науки и образования. – 2018 – №3 (35). – С. 157-166.
12. Митрохин М.А., Кочеткова О.А. Чат-бот для сферы образования / Инжиниринг и технологии. 2023. Т. 8. № 2. С. 19-21.
13. Основы программирования на примере языка Python [Электронный ресурс]. – URL: <https://dfedorov.spb.ru/python3/book.pdf> (дата обращения 12.12.2022).
14. Розов К.В. Технологии искусственного интеллекта на языке Python 3. – Новосибирск: Новосибирский государственный педагогический университет, 2021. – 211 с.
15. Могилевская Н.С., Самойленко Г.П. Чат-бот для тестирования знаний на платформе мессенджера Telegram. – Молодой исследователь Дона. – №3(36) 2022. [Электронный ресурс]. – URL: https://www.elibrary.ru/download/elibrary_48995351_91920602.pdf (дата обращения 23.03.2023).
16. Статистика популярности мессенджеров в России и мире [Электронный ресурс]. – URL: <https://mindbox.ru/journal/education/populyarnye-messendzheri/> (дата обращения: 10.03.2024).
17. Агальцов В.П. Базы данных: В 2-х книгах. Книга 2: Распределенные и удаленные базы данных. – М.: Форум, 2022. – 271 с.
18. Учебник по SQLite3 в Python [Электронный ресурс]. – URL: <https://digitology.tech/posts/uchebnik-po-sqlite3-v-python/> (дата обращения: 11.10.2023).

19. SQLite: Home Page [Электронный ресурс]. – URL: <https://www.sqlite.org/index.html> (дата обращения 18.12.2023).
20. Шмелев А.Н., Жилина Е.В. Программная реализация приложений по работе с Telegram-каналами // Проблемы проектирования, применения и безопасности информационных систем в условиях цифровой экономики: материалы XXI междунар. науч.-практ. конф., 29–30 ноября 2021 г. – Ростов н/Д: ИПК РГЭУ (РИНХ), 2021. – С. 214-220.
21. Инструкция по созданию и настройки бота в BotFather [Электронный ресурс]. – URL: <https://docs.radist.online/radist.online-docs/nashi-produkty/radist-web/podklyucheniya/telegram-bot/instrukciya-po-sozdaniyu-i-nastroiki-bota-v-botfather> (дата обращения 18.03.2023).
22. Архаков Д. PHP: Делаем кнопки в Telegram API (inline- keyboards) [Электронный ресурс]. – URL: <https://archakov.im/post/nodejs-make-buttons-on-telegramapi.html> (дата обращения 29.04.2023).
23. Горбаченко В.И., Савенков К.Е., Малахов М.А. Машинное обучение: учебное пособие. – М.: Ай Пи Ар Медиа, 2023. – 217 с.
24. Анализ функций потерь и метрик в машинном обучении [Электронный ресурс]. – URL: <https://www.yourtodo.ru/posts/analiz-funktsij-poter-i-metrik-v-mashinnom-obuchenii/> (дата обращения 11.01.2024).
25. Диагностика систем машинного обучения [Электронный ресурс]. – URL: <https://koroteev.site/pres/ml4/> (дата обращения 11.01.2024).
26. Учебник-справочник по sklearn [Электронный ресурс]. – URL: https://scikit-learn.ru/category/supervised_learning/ (дата обращения: 04.02.2024).
27. Кугаевских А.В., Муромцев Д.И., Кирсанова О.В. Классические методы машинного обучения. – СПб: Университет ИТМО, 2022. – 53 с.
28. Бринк Х., Ричардс Д., Феверолф М. Машинное обучение. – СПб.: Питер. – 2017. – 336 с.

29. Overview of Machine Learning Algorithms: Classification [Электронный ресурс]. – URL: <https://www.stratascratch.com/blog/overview-of-machine-learning-algorithms-classification/> (дата обращения: 04.12.2023).

30. Machine Learning with Python tutorial [Электронный ресурс]. – URL: <https://www.geeksforgeeks.org/machine-learning-with-python/> (дата обращения: 11.11.2023).

ПРИЛОЖЕНИЕ А

Листинг исходного кода

```
import telebot
from telebot import types
import re
import nltk
import random
import sqlite3
import seaborn as sns
from sklearn.metrics import confusion_matrix
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import auc
import numpy as np
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score,
f1_score
from sklearn.feature_extraction.text import CountVectorizer

bot = telebot.TeleBot('TOKEN')

# загрузка данных из JSON-файла
import json
with open("bot_ii.json", "r", encoding="utf-8") as config_file:
    data = json.load(config_file)

INTENTS = data["intents"]

#фильтрация текста
def filter_text(text):
    text = text.lower()
```

```

text = text.strip()
pattern = r"^[^\w\s]"
text = re.sub(pattern, "", text)
return text

# Сравнение текста (user_text пользовательский текст, example -
# контрольная фраза)
def text_match(user_text, example):
    user_text = filter_text(user_text)
    example = filter_text(example)
    if user_text == example:
        return True
    if user_text.find(example) != -1:
        return True
    distance = nltk.edit_distance(user_text, example)
    ratio = distance / len(example)
    if ratio < 0.3:
        return True
    return False

# Функция для нахождения категории вопроса пользователя по его
# тексту
def get_intent(user_text):
    for intent in INTENTS:
        examples = INTENTS[intent]["examples"]
        for example in examples:
            if len(filter_text(example)) <=2:
                continue
            if text_match(user_text, example):
                return intent
    return None

# Функция возвращает случайную фразу ответа по данному интену ( )
def get_random_response(intent):
    return random.choice(INTENTS[intent]["responses"])

```

```

# Создание обучающих выборок
X = []
y = []
for intent in INTENTS:
    examples = INTENTS[intent]["examples"]
    for example in examples:
        example = filter_text(example)
        if len(example) < 2:
            continue
        X.append(example)
        y.append(intent)
print(len(X))

# Векторизация текста
vectorizer = CountVectorizer()
vectorizer.fit(X)
vecX = vectorizer.transform(X)

np.random.seed(42)
vecX, y = make_classification(n_samples=700, n_classes=2,
                             random_state=42)
X_train, X_test, y_train, y_test = train_test_split(vecX, y,
                                                    test_size=0.2, random_state=42)

# Для оценки качества модели и сравнения различных алгоритмов
вычисляются метрики: Accuracy, Precision, Recall, F1-Score, ROC

# LogisticRegression
print("LogisticRegression")
from sklearn.linear_model import LogisticRegression
model_Log = LogisticRegression()
y_scor_Log=model_Log.fit(X_train, y_train)
y_score_Log=y_scor_Log.predict(X_test)
y_pred_prob = model_Log.predict_proba(X_test)[:, 1]
roc_auc_Log = roc_auc_score(y_test, y_pred_prob)

```

```

print(f"roc_auc: {round(roc_auc_Log,3)} ")
print(f"accuracy score: {round(accuracy_score(y_test,
y_score_Log),3)}")
print(f"f1 score: {round(f1_score(y_test, y_score_Log),3)}")
print(f"recall: {round(recall_score(y_test, y_score_Log),3)}")
print(f"precision: {round(precision_score(y_test,
y_score_Log),3)}")

# KNeighborsClassifier
print("KNeighborsClassifier")
from sklearn.neighbors import KNeighborsClassifier
model_KN = KNeighborsClassifier()
y_scor_KN=model_KN.fit(X_train, y_train)
y_score_KN=y_scor_KN.predict(X_test)
y_pred_prob_KN = model_KN.predict_proba(X_test)[:, 1]
roc_auc_KN = roc_auc_score(y_test, y_pred_prob_KN)
print("roc_auc: ",round(roc_auc_KN,3))
print(f"accuracy score: {round(accuracy_score(y_test,
y_score_KN),3)}")
print(f"f1 score: {round(f1_score(y_test, y_score_KN),3)}")
print(f"recall: {round(recall_score(y_test, y_score_KN),3)}")
print(f"precision: {round(precision_score(y_test,
y_score_KN),3)}")

# SVC
print("SVC")
from sklearn.svm import SVC
svc = SVC(probability=True)
y_svc = svc.fit(X_train, y_train)
y_score_svc=y_svc.predict(X_test)
y_pred_prob_svc = svc.predict_proba(X_test)[:, 1]
roc_auc_svc= roc_auc_score(y_test, y_pred_prob_svc)
print(f"roc_auc: {round(roc_auc_svc,3)} ")
print(f"accuracy score: {round(accuracy_score(y_test,
y_score_svc),3)}")
print(f"f1 score: {round(f1_score(y_test, y_score_svc),3)}")

```

```

print(f"recall: {round(recall_score(y_test, y_score_svc),3)}")
print(f"precision: {round(precision_score(y_test,
y_score_svc),3)}")

# MLPClassifier
print("MLPClassifier")
from sklearn.neural_network import MLPClassifier
model_PC = MLPClassifier()
y_scor_KPC=model_PC.fit(X_train, y_train)
y_score_KPC=y_scor_KPC.predict(X_test)
y_pred_prob_PC = model_PC.predict_proba(X_test)[:, 1]
roc_auc_PC = roc_auc_score(y_test, y_pred_prob_PC)
print(f"roc_auc: {round(roc_auc_PC,3)} ")
print(f"accuracy score: {round(accuracy_score(y_test,
y_score_KPC),3)}")
print(f"f1 score: {round(f1_score(y_test, y_score_KPC),3)}")
print(f"recall: {round(recall_score(y_test, y_score_KPC),3)}")
print(f"precision: {round(precision_score(y_test,
y_score_KPC),3)}")

# RandomForestClassifier
print("RandomForestClassifier")
from sklearn.ensemble import RandomForestClassifier
model_rF = RandomForestClassifier()
y_scor=model_rF.fit(X_train, y_train)
y_score=y_scor.predict(X_test)
y_pred_prob_rF = model_rF.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, y_pred_prob_rF)
print(f"roc_auc: {round(roc_auc,3)} ")
print(f"accuracy score: {round(accuracy_score(y_test,
y_score),3)}")
print(f"f1 score: {round(f1_score(y_test, y_score),3)}")
print(f"recall: {round(recall_score(y_test, y_score),3)}")
print(f"precision: {round(precision_score(y_test, y_score),3)}")

print('Неверно классифицированные образцы: %d' % (y_test !=

```

```

y_score).sum())

# Матрица неточности: вычисление и построение таблицы
cnf_matrix_log = confusion_matrix(y_test, y_score)
sns.heatmap(pd.DataFrame(cnf_matrix_log), annot=True, cmap="Reds"
, fmt='g')
plt.tight_layout()
plt.title('Матрица неточности', y=1.1)
plt.legend()
plt.show()

# График ROC-кривых
# ROC-кривая для LogisticRegression
fpr_1, tpr_1, _ = roc_curve(y_test, y_pred_prob)
roc_auc_1 = auc(fpr_1, tpr_1)

# ROC-кривая для KNeighborsClassifier
fpr_2, tpr_2, _ = roc_curve(y_test, y_pred_prob_KN)
roc_auc_2 = auc(fpr_2, tpr_2)

# ROC-кривая для SVC
fpr_3, tpr_3, _ = roc_curve(y_test, y_pred_prob_svc)
roc_auc_3 = auc(fpr_3, tpr_3)

# ROC-кривая для MLPClassifier
fpr_4, tpr_4, _ = roc_curve(y_test, y_pred_prob_PC)
roc_auc_4 = auc(fpr_4, tpr_4)

# ROC-кривая для RandomForestClassifier
fpr_5, tpr_5, _ = roc_curve(y_test, y_pred_prob_rF)
roc_auc_5 = auc(fpr_5, tpr_5)

plt.figure()
plt.plot(fpr_1, tpr_1, color='darkorange', lw=2, label='ROC
curve LogisticRegression (area = %0.2f)' % roc_auc_1)
plt.plot(fpr_2, tpr_2, color='green', lw=2, label='ROC curve

```

```

KNeighborsClassifier (area = %0.2f)' % roc_auc_2)
plt.plot(fpr_3, tpr_3, color='red', lw=2, label='ROC curve SVC
(area = %0.2f)' % roc_auc_3)
plt.plot(fpr_4, tpr_4, color='yellow', lw=2, label='ROC curve
MLPClassifier (area = %0.2f)' % roc_auc_4)
plt.plot(fpr_5, tpr_5, color='black', lw=2, label='ROC curve
RandomForestClassifier (area = %0.2f)' % roc_auc_5)
plt.plot([0, 1], [0, 1], color='blue', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Analysis')
plt.legend(loc="lower right")
plt.show()

from sklearn import metrics

# оценка точности модели
print(metrics.classification_report(y_test, y_score))

from sklearn.model_selection import cross_val_score
rf_scores = cross_val_score(RandomForestClassifier(), X_train,
y_train)
mlp_scores = cross_val_score(MLPClassifier(), X_train, y_train)
lr_scores = cross_val_score(LogisticRegression(), X_train,
y_train)
svc_scores = cross_val_score(SVC(), X_train, y_train)
knn_scores = cross_val_score(KNeighborsClassifier(), X_train,
y_train)
methods = ['RandomForest', 'MLP', 'Logistic Regression', 'SVC',
'KNeighbors']
scores = [rf_scores.mean(), mlp_scores.mean(), lr_scores.mean(),
svc_scores.mean(), knn_scores.mean()]
plt.figure(figsize=(8, 4))
plt.bar(methods, scores, color='skyblue')

```

```

plt.xlabel('Методы машинного обучения')
plt.ylabel('Точность')
plt.title('График оценки точности методов машинного обучения')
plt.ylim(0.6, 1)
plt.show()

# БД пользователей
db = sqlite3.connect('users.db', check_same_thread = False)
sql = db.cursor()
sql.execute("""CREATE TABLE IF NOT EXISTS users(
    chatid TEXT,
    username TEXT,
    first_name TEXT,
    last_name TEXT
) """)
db.commit()

import time
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(vecX, y)

# Команда /start
@bot.message_handler(commands=['start'])
def handle_start(message):
    # Создание клавиатуры
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True,
row_width=1)
    button1 = types.KeyboardButton('☐ Что я умею')
    button2 = types.KeyboardButton('☐ Лабораторные работы')
    button3 = types.KeyboardButton('☐ Контакты')
    button4 = types.KeyboardButton('☐ Заочная форма обучения')
    button5 = types.KeyboardButton('☐ Аннотация к РП')
    keyboard.row(button1)
    keyboard.row(button3, button5)
    keyboard.add(button2, button4)

```

```

        bot.reply_to(message, 'Вы подписаны на рассылку',
reply_markup=keyboard)
        sql.execute(f"SELECT chatid FROM users WHERE chatid =
'{message.chat.id}'")
        if sql.fetchone() is None:
            sql.execute(f"INSERT INTO users VALUES(?,?,?,?)",
(message.chat.id, message.from_user.username,
message.chat.first_name, message.chat.last_name))
            db.commit()
        else:
            for value in sql.execute("SELECT * FROM users"):
                print(value)
            bot.send_message(message.chat.id, 'Чем могу помочь?')

# Команда /dist
@bot.message_handler(commands=['dist'])
def dist(message):
    sql.execute("SELECT chatid FROM users")
    results = sql.fetchall()
    for result in results:
        bot.send_message(result[0], "Рассылка: KT2 будет с
6.05.24-12.05.24. "
                                "\nНужно сдать лабораторные
4-6.")
    db.close()

# Команда /help
@bot.message_handler(commands=['help'])
def help_message(message):
    bot.send_message(message.chat.id, 'Вот список доступных
команд:')
    bot.send_message(message.chat.id, '/start - запуск
бота\n/help - вывод информации о доступных командах \n/version -
вывод информации о текущей версии и её возможностях')

```

```

# Команда /version
@bot.message_handler(commands=['version'])
def help_message(message):
    bot.send_message(message.chat.id, 'Бот-помощник 1.0 изучения
    дисциплины "Современные информационные технологии" студентами
    различных направлений ПИ им. В.Г. Белинского')

# Реакция на document
@bot.message_handler(content_types=["document"])
def handle_docs_audio(message):
    bot.send_message(message.chat.id, "Файл получен")
    document_id = message.document.file_id
    file_info = bot.get_file(document_id)
    print(document_id) # Выводим file_id
print(f'http://api.telegram.org/file/bot{"API"}/{file_info.file_
path}')

# Реакция на audio
@bot.message_handler(content_types=["audio"])
def handle_docs_document(message):
    bot.send_message(message.chat.id, "Файл получен")
    audio_id = message.audio.file_id
    file_info = bot.get_file(audio_id)

bot.request.urlretrieve(f'http://api.telegram.org/file/bot{bot.t
oken}/{file_info.file_path}', file_info.file_path)

# Реакция на text, photo, sticker, animation, video
@bot.message_handler(content_types=['text', 'photo', 'sticker',
'animation', 'video'])
def handle_message(message):
    if message.text == "□ Что я умею":
        bot.send_message(message.chat.id, "Дисциплина
        «Современные информационные технологии» нацелена на формирование
        у студентов практических навыков использования современных
        информационно-коммуникационных технологий для решения

```

профессиональных задач. Бот подойдет студентам различных направлений подготовки ПИ им. В.Г. Белинского. \n/start - запуск бота\n/help - вывод информации о доступных функциях\n/version - вывод информации о текущей версии и её возможностях")

```
elif message.text == "☐ Контакты":
    bot.send_message(message.chat.id,
                      "Кафедра ИиМОИМ
https://dep_tmomi.pnzgu.ru/. "
                      "\nКонтактная информация:"
                      "\nАдрес: г. Пенза, ул. Лермонтова, 37,
корпус 13"
                      "\наудитория 13-37, 13-41 (3 этаж)"
                      "\ntел: +7(8412) 20-36-22"
                      "\ne-mail: pgputimom@yandex.ru"
                      "\n Преподаватели:      \nГаврилова М.А.
(pgputimom@yandex.ru), \nМарина Е.В. (elenamarinal@yandex.ru),
\nПудовкина Ю.Н. (yulia_pudowkina@mail.ru)\n Болотский
А.В. (a.bolotskiy@mail.ru)")
    otvet = types.InlineKeyboardMarkup(row_width=2)
    button11 = types.InlineKeyboardButton("да",
callback_data='да')
    otvet.add(button11)
    bot.send_message(message.chat.id, "Отправить расписание
преподавателей?", reply_markup=otvet)
elif message.text == "☐ Аннотация к РП":
    bot.send_message(message.chat.id,
                      "Цель преподавания дисциплины -
формирование у будущих бакалавров углубленных знаний в области
современных информационных и коммуникационных технологий с
учетом основных требований информационной безопасности,
формирование информационной культуры, ориентация на творческое и
профессиональное использование современных достижений цифровых
технологий в обучении, будущей профессиональной деятельности, в
процессе самообразования и повышения квалификации, формирование
способности понимания специфики создания и использования баз
данных и информационных систем для научных исследований и
```

```

образования. ")
    otvet = types.InlineKeyboardMarkup(row_width=2)
    button11 = types.InlineKeyboardButton(" да ",
callback_data=' да ')
    otvet.add(button11)
    time.sleep(2)
    bot.send_message(message.chat.id, "Отправить аннотацию к
ПП?", reply_markup=otvet)
    elif message.text == "□ Лабораторные работы":
        otvet = types.InlineKeyboardMarkup(row_width=2)
        button1 = types.InlineKeyboardButton("да",
callback_data='да')
        otvet.add(button1)
        bot.send_message(message.chat.id, "Отправить задания?",
reply_markup=otvet)
        elif message.text == "□ Заочная форма обучения":
            bot.send_message(message.chat.id, "Студенты заочного
отделения выполняют работу по варианту (номер ФИО в ЭИОС).")
            bot.send_message(message.chat.id,
                            "Вы хотите ознакомиться с заданиями
на лабораторные работы и с вопросами к зачету?")
            otvet = types.InlineKeyboardMarkup(row_width=2)
            button100 = types.InlineKeyboardButton(" да",
callback_data=' да ')
            otvet.add(button100)
            bot.send_message(message.chat.id, "Отправить?",
reply_markup=otvet)
            time.sleep(3)
            bot.send_message(message.chat.id, "Не забудьте к зачету
пройти итоговое тестирование по курсу")
            markup = types.InlineKeyboardMarkup()
            button3 = types.InlineKeyboardButton("Итоговое
тестирование", url='https://forms.gle/uvhgeA1c6JagA2518')
            markup.add(button3)
            bot.send_message(message.chat.id, "Открыть
тест?".format(message.from_user), reply_markup=markup)

```

```

        # Ответ на изображение
    elif message.photo:
        bot.send_message(message.chat.id, 'И зачем это
отправили?')
        time.sleep(2)
        bot_otv = random.choice(['Я, к сожалению, не могу
проанализировать картинку',
                                'Возможно посмотрю потом... Или
нет... Задавайте по делу вопросы',
                                'Ближе к делу',
                                'Это не относиться к изучаемому
курсу'])
        bot.send_message(message.chat.id, bot_otv)
        # Ответ на стикер
    elif message.sticker:
        bot_otv = random.choice([' Странно... Такое это имеет
отношение к учебе?',
                                'Возможно посмотрю потом... Или
нет... Какие у вас вопросы?',
                                'И зачем это отправили? Ближе к
делу... Чем помочь?'])
        bot.send_message(message.chat.id, bot_otv)
    elif message.animation:
        bot.send_message(message.chat.id, 'Не смешно...')
        time.sleep(2)
        bot_otv = random.choice(['Задавайте по делу вопросы',
                                'На отвлечённые темы не
разговариваю...'],
                                'Говорите лишнее! Ближе к делу!')
        bot.send_message(message.chat.id, bot_otv)
    elif message.video:
        bot.send_message(message.chat.id, 'Я не могу это
оценить...')
        time.sleep(2)

```

```

        bot_otv = random.choice(['Давайте поговорим об учебе?',
                                  'На отвлечённые темы не
разговариваю...'],
                                  'Давайте поговорим по
существу... Какие есть вопросы про обучение курса? '])
        bot.send_message(message.chat.id, bot_otv)
    elif message.text != '':
        print (message.text)
        X_user = vectorizer.transform([message.text])
        predicted_tag = model.predict(X_user)[0]
        confidence_score =
model.predict_proba(X_user)[0][model.classes_ ==
predicted_tag][0]
        print("Predicted tag:", predicted_tag)
        if predicted_tag=='lab':
            bot.send_message(message.chat.id,
get_random_response(predicted_tag))
            time.sleep(2)
            bot.send_message(message.chat.id, "...")
            predicted_tag = random.choice(['lab_otv','lab_sd'])
            print(predicted_tag)
            bot.send_message(message.chat.id,
get_random_response(predicted_tag))
        elif predicted_tag == 'book':
            bot.send_message(message.chat.id,
get_random_response(predicted_tag))
            time.sleep(2)
            predicted_tag = 'book_otv'
            bot.send_message(message.chat.id,
get_random_response(predicted_tag))
        elif predicted_tag == 'zaoch' or predicted_tag ==
'text' or predicted_tag == 'lab_non' or predicted_tag ==
'ball_dobor' or predicted_tag == 'bye' or predicted_tag ==
'praktika' or predicted_tag == 'praktika' or predicted_tag ==
'subject' or predicted_tag == 'zachet' or predicted_tag ==
'name_bot' or predicted_tag == 'speak' or predicted_tag == 'da'

```

```

or predicted_tag == 'net' or predicted_tag == 'ball' or
predicted_tag == 'propusk_uv' or predicted_tag == 'ball' or
predicted_tag == 'zachet_sd' or predicted_tag == 'pomosch' or
predicted_tag == 'lab_sd' or predicted_tag == 'zun' or
predicted_tag == 'spasibo':
    bot.send_message(message.chat.id,
get_random_response(predicted_tag))
    elif predicted_tag == 'teacher':
        bot.send_message(message.chat.id,
get_random_response(predicted_tag))
        if predicted_tag == 'itog':
            bot.send_message(message.chat.id,
get_random_response(predicted_tag))
        elif predicted_tag == 'lab_da':
            bot.send_message(message.chat.id,
get_random_response(predicted_tag))
            predicted_tag = 'lab_otv'
            bot.send_message(message.chat.id,
get_random_response(predicted_tag))
        elif predicted_tag == 'obraz':
            bot.send_message(message.chat.id,
get_random_response(predicted_tag))
            markup = types.InlineKeyboardMarkup()
            button3 = types.InlineKeyboardButton("Посмотреть
файл?", url='https://disk.yandex.ru/i/L79Q2QW4aX3gEA')
            markup.add(button3)
            bot.send_message(message.chat.id, "Общий вид
титульного листа отчета приведен
ниже".format(message.from_user), reply_markup=markup)
        elif predicted_tag == 'privet':
            bot.send_message(message.chat.id,
get_random_response(predicted_tag))
            time.sleep(2)
            predicted_tag = 'vopr'
            bot.send_message(message.chat.id,
get_random_response(predicted_tag))

```

```

elif predicted_tag == 'test':
    bot.send_message(message.chat.id,
get_random_response(predicted_tag))
    markup = types.InlineKeyboardMarkup()
    button3 = types.InlineKeyboardButton("Итоговое
тестирование", url='https://forms.gle/uvhgeA1c6JagA2518')
    markup.add(button3)
    bot.send_message(message.chat.id, "Открыть
тест?".format(message.from_user), reply_markup=markup)
elif predicted_tag == 'po':
    bot.send_message(message.chat.id,
get_random_response(predicted_tag))
    if predicted_tag == 'itog_1':
        bot.send_message(message.chat.id,
get_random_response(predicted_tag))
    elif predicted_tag == 'sit':
        bot.send_message(message.chat.id,
get_random_response(predicted_tag))
        markup = types.InlineKeyboardMarkup()
        button1 = types.InlineKeyboardButton("Глоссарий",
url='https://disk.yandex.ru/i/5dI7k4Jt4I8drg')
        markup.add(button1)
        bot.send_message(message.chat.id, "Открыть
глоссарий?".format(message.from_user), reply_markup=markup)
elif predicted_tag == 'par':
    bot.send_message(message.chat.id,
get_random_response(predicted_tag))
    time.sleep(2)
    bot.send_message(message.chat.id,
get_random_response(predicted_tag))
elif predicted_tag == 'zach_stud':
    bot.send_message(message.chat.id,
get_random_response(predicted_tag))
    time.sleep(2)
    predicted_tag = 'teacher'
    bot.send_message(message.chat.id,

```

```

get_random_response(predicted_tag))
    elif predicted_tag == 'grafik_lab':
        bot.send_message(message.chat.id,
get_random_response(predicted_tag))
        time.sleep(2)
        markup = types.InlineKeyboardMarkup()
        button1 = types.InlineKeyboardButton("Отправить
файл? ",url='https://disk.yandex.ru/i/2VUmA_P74K-PVw')
        markup.add(button1)
        time.sleep(2)
        bot.send_message(message.chat.id, "У меня есть
график лабораторных работ.".format(message.from_user),
reply_markup=markup)
        print("Confidence score:", confidence_score)
    else:
        bot.send_message(message.chat.id, 'Простите, я тебя не
понимаю')

# Обработка Callback-кнопка
@bot.callback_query_handler(func=lambda call: True)
def callback_inline(call):
    try:
        if call.message:
            if call.data == "да":
                doc = open('Лабораторные работы.pdf', 'rb')
                bot.send_document(call.message.chat.id, doc)
            elif call.data == "да":
                doc = open('Расписание преподавателей.pdf',
'rb')
                bot.send_document(call.message.chat.id, doc)
            elif call.data == " да":
                doc = open('Информация для заочного
отделения.pdf', 'rb')
                bot.send_document(call.message.chat.id, doc)
            elif call.data == " да ":
                doc = open('Аннотация.pdf', 'rb')

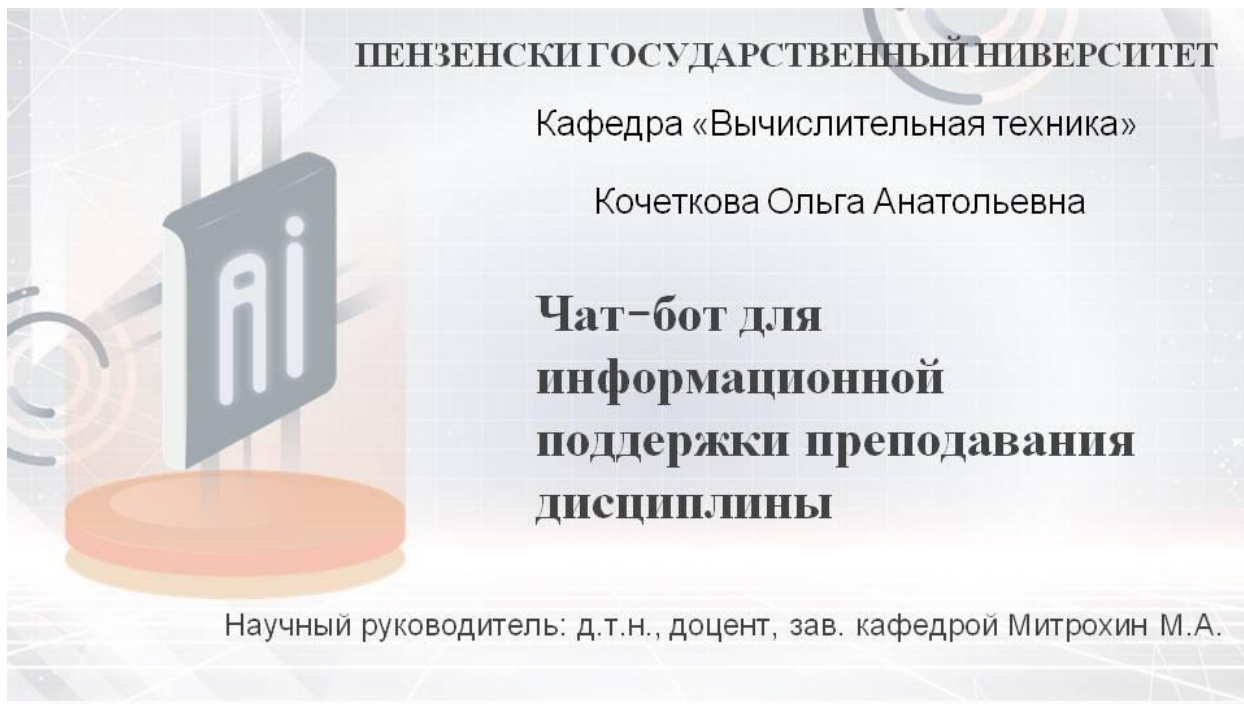
```

```
        bot.send_document(call.message.chat.id, doc)
    except Exception as e:
        print(repr(e))

bot.polling()
```

ПРИЛОЖЕНИЕ Б

Презентация



Цель и задачи исследования

Цель работы: разработка приложения информационной поддержки дисциплины (на примере «Современные информационные технологии») на основе интеллектуального чат-бота.

Задачи:

- изучить различные подходы к понятию «чат-бот», выделить основные его функции, провести анализ эффективности использования ботов в образовательном процессе,
- провести обзор различных ПО для разработки чат-бота и обосновать выбор,
- разработать структуру бота для информационной поддержки выбранной учебной дисциплины,
- изучить модели машинного обучения для обработки запросов пользователей в чат-боте,
- продемонстрировать концепцию разработки и реализацию чат-бота для использования в образовательном процессе.

Раздел 1. Теоретический анализ предметной области

Сферы применения технологий на основе ИИ в сфере образования:

- 1) автоматизированные учебные курсы,
- 2) адаптированные и персонализированные системы обучения,
- 3) консультирование и тестирование, корректировка лекционного и практического материала курса в зависимости предпочтений обучающегося и уровня усвоения его материала,
- 4) фиксация сформированности учебных, профессиональных и цифровых компетенций на основе анализа данных,
- 5) системы распознавания лиц в классах и аудиториях помогают понять, насколько обучающиеся погружены в образовательный процесс и т.д.

Входные данные: вопрос
или фраза собеседника

Анализ информации с
помощью логики
приложения

Воспроизведение ответа

Раздел 2. Выбор программного обеспечения для разработки чат-бота

Python



PyCharm
Community
Edition

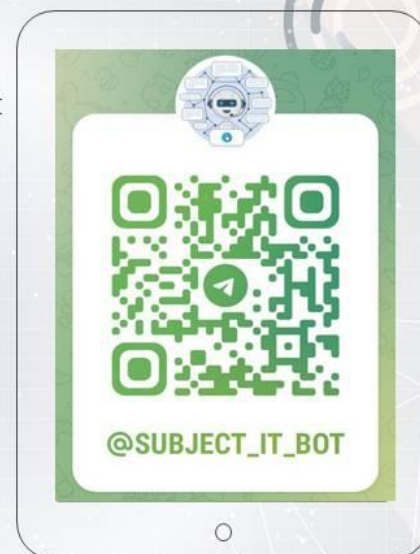


SQLite



Раздел 3. Разработка программного продукта

1. Регистрация в BotFather.
2. Административная панель управления (кнопки меню).
3. Проектирование и реализация базы данных чат-бота.
4. Анализ моделей машинного обучения для автоматизированной обработки обращений пользователя.
5. Разработки чат-бота для информационной поддержки преподавания дисциплины.



Подготовка текстовых данных для модели ML



1. Фильтрация символов текста

- перевод всех символов к нижнему регистру, удаление пробелов и других символов, которые находятся в начале/конце строки, операции с регулярными выражениями,
- вычисление расстояния между двумя строками.

2. Обработка естественного языка

- удаление стоп-слов, лемматизация, стемминг, токенизация и векторизация.

3. Классификация текстов ML-моделью.

Оценка результатов машинного обучения

Метрики качества – это способ оценки производительности модели машинного обучения на основе ее способности предсказывать правильные ответы для тестовой выборки.

1. Accuracy (точность)
2. Precision (точность)
3. Recall (полнота)
4. F1 Score
5. ROC-AUC
(площадь под ROC-кривой)



Модели ML

- KNeighborsClassifier (метод k-ближайших соседей)
- SVM (Support Vector Machines, метод опорных векторов)
- Logistic Regression (логистическая регрессия)
- MLPClassifier
- Random Forest (метод случайного леса)

Модель RandomForestClassifier

Работа алгоритма случайного леса:

- 1) Выполнить разбиение случайных выборок из заданного набора данных.

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(vecX, y, test_size=0.2, random_state=42)
```

- 2) Обучение модели.

```
model_rF = RandomForestClassifier()  
y_scor=model_rF.fit(X_train, y_train)
```

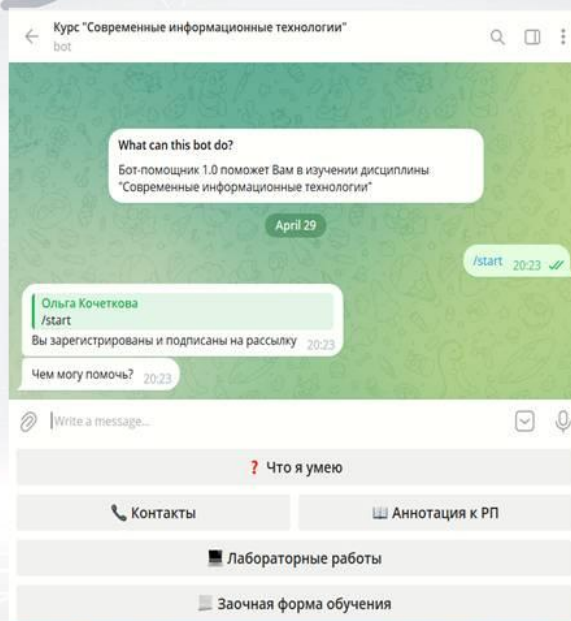
- 3) Получение прогнозов на тестовом наборе.

```
y_score=y_scor.predict(X_test)
```

- 4) Оценка точности модели.

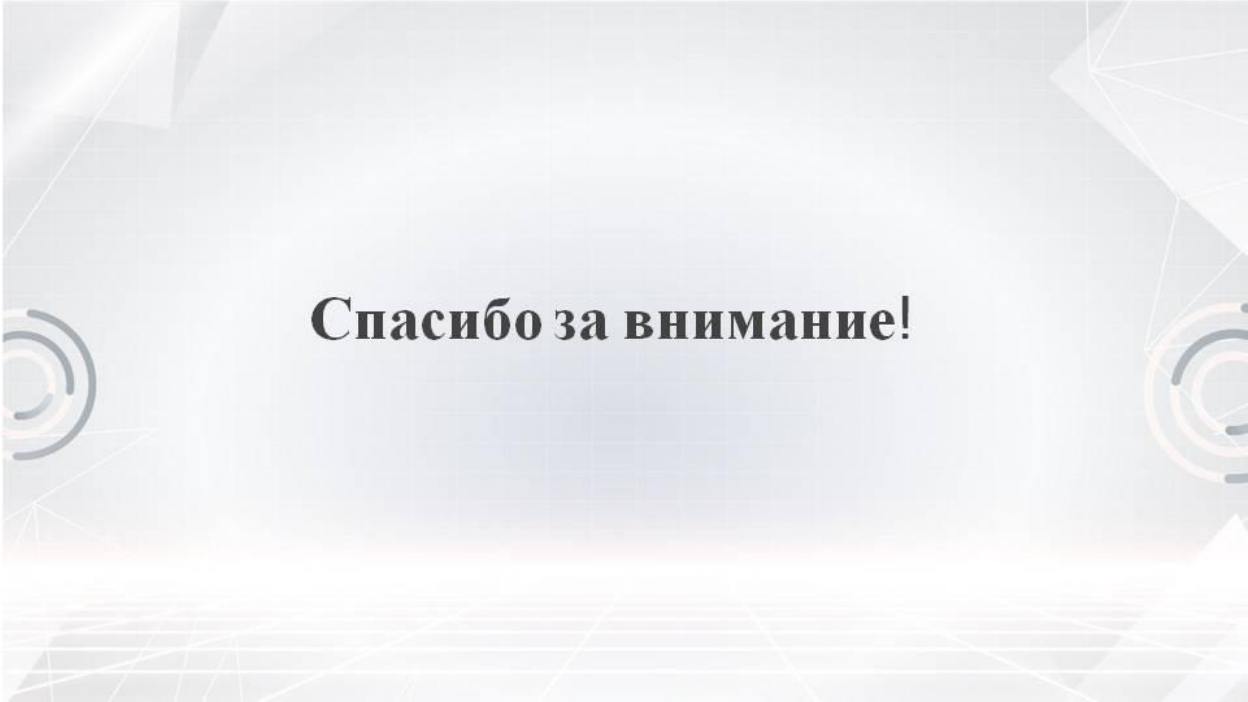
```
print(f"accuracy score: {round(accuracy_score(y_test, y_score),3)}")  
print(f"f1 score: {round(f1_score(y_test, y_score),3)}")  
print(f"recall: {round(recall_score(y_test, y_score),3)}")  
print(f"precision: {round(precision_score(y_test, y_score),3)}")
```

Результаты разработки чат-бота



Основные результаты и выводы

- разработка и реализации чат-бота на платформе Telegram для информационной поддержки студентов 1 курса очной и заочной формы обучения, изучающие дисциплину «Современные информационные технологии»,
- для создания чат-бота в выпускной квалификационной работе были задействованы современные технологии и инструменты: интегрированная среда разработки PyCharm и язык программирования Python, библиотеки для реализации алгоритмов ML и AI и для взаимодействия с Telegram Bot API.

The background of the slide is a light gray with a subtle grid pattern. On the left and right sides, there are decorative elements consisting of concentric circles and lines, resembling a stylized 'C' or a partial spiral. The text is centered in the upper half of the slide.

Спасибо за внимание!