МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**"Южно-Уральский государственный университет**

**(национальный исследовательский университет) "**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

**ОТЧЕТ**  
**о выполнении научно-исследовательской работы**бакалавра направления 02.03.02 «Фундаментальная информатика и информационные технологии»

|  |  |
| --- | --- |
|  | Автор работы:  студент группы КЭ-201  Циммерман Юлия Владимировна  Научный руководитель:  Доктор физико-математических наук, доцент кафедры  «Системное программирование»  М.Л. Цымблер  Оценка: \_\_\_\_\_\_\_\_\_\_\_  Дата: \_\_\_\_\_\_\_\_\_\_\_  Подпись: \_\_\_\_\_\_\_\_\_\_\_ |

Челябинск 2021МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**"Южно-Уральский государственный университет**

**(национальный исследовательский университет)"**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

УТВЕРЖДАЮ

Зав. кафедрой СП

\_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский

01.03.2021 г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

студенту группы КЭ-201

Циммерман Юлия Владимировна

обучающемуся по направлению 02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема НИР в семестре**Применение технологий искусственного интеллекта в виртуальных собеседниках на примере разработки простого чат-бота
2. **Срок сдачи студентом законченной работы**  31.05.2021 г.
3. **Перечень подлежащих разработке вопросов**

Обзор научных статей по тематике применения искусственных нейронных сетей для разработки чат-ботов

**Научный руководитель**

Доктор физико-математических наук, М.Л. Цымблер  
доцент кафедры   
«Системное программирование»

**Задание принял к исполнению** Ю.В. Циммерман

**Оглавление**

[Введение 4](#_Toc72279564)

[1. Программа чат-бот и история ее возникновения 5](#_Toc72279565)

[1.1 Понятие «чат-бот» 5](#_Toc72279566)

[1.2 История возникновения чат-ботов 5](#_Toc72279567)

[1.3 Классификация чат-ботов 6](#_Toc72279568)

[2. Реализация чат-бота 8](#_Toc72279569)

[2.1 Нейронные сети в обработке естественного языка 8](#_Toc72279570)

[2.2 Обучение чат-ботов 9](#_Toc72279571)

[2.3 Сложности при создании чат-бота 9](#_Toc72279572)

[2.4 Морфологический анализ данных 10](#_Toc72279573)

[2.4.1 Токенизация 10](#_Toc72279574)

[2.4.2 Нормализация 10](#_Toc72279575)

[2.4.3 Лемматизация 10](#_Toc72279576)

[2.4.4 Удаление стоп-слов 11](#_Toc72279577)

[2.5 Векторизация текста 11](#_Toc72279578)

[2.6 Работа с Telegram при разработке чат-бота 12](#_Toc72279579)

[3. Проектирование 13](#_Toc72279580)

[3.1 Архитектура программы 13](#_Toc72279581)

[3.2 Модульная структура программы 14](#_Toc72279582)

[3.2.1 Модуль «Морфологический анализ текста» 14](#_Toc72279583)

[3.2.2 Модуль «Векторизация текста» 15](#_Toc72279584)

[3.2.3 Модуль «Генерация ответа» 15](#_Toc72279585)

[4. Реализация 17](#_Toc72279586)

[4.1 Морфологический анализ текста 17](#_Toc72279587)

[4.2. Векторизация текста 18](#_Toc72279588)

[4.3. Генерация ответа 18](#_Toc72279589)

[5. Тестирование 19](#_Toc72279590)

[5.1. Морфологический анализ текста 19](#_Toc72279591)

[5.1.1. Токенизация 19](#_Toc72279592)

[5.1.2. Нормализация 19](#_Toc72279593)

[5.1.3. Лемматизация 20](#_Toc72279594)

[5.1.4. Удаление стоп-слов 20](#_Toc72279595)

[5.2. Векторизация текста 20](#_Toc72279596)

[5.3. Генерация ответа 21](#_Toc72279597)

[6. Работа итогового приложения 22](#_Toc72279598)

[Заключение 23](#_Toc72279599)

[Приложение 1. Реализация «Первичный морфологический анализ текста». 26](#_Toc72279600)

[Приложение 2. Реализация «Векторизация текста» 28](#_Toc72279601)

[Приложение 3. Модуль «Генерация ответа» 29](#_Toc72279602)

**Обзор научных работ по тематике разработки чат-ботов**

# Введение

Целью данной работы является обзор статей по тематике применения искусственных нейронных сетей для разработки чат-ботов.

Актуальность тематики по разработке чат-ботов обусловлена тем, что виртуальное общение становится основным видом коммуникации в современном мире. В настоящее время чат-боты представлены в широком спектре [19]. Так, в маркетинге их основными задачами являются поддержание интереса потребителя, ответы на вопросы и помощь в покупке какого-либо товара. Помимо этого, многие компании прибегают к использованию виртуальных помощников для проведения финансовых услуг и распространения рекламы. Также чат-боты оказали большое влияние на сферу образования за счет различных информационных рассылок и завлечения в образовательные курсы и вебинары. В таких сферах, как туризм и банки, чат-боты в основном выполняют функцию консультантов, таким образом, помогают потребителю подобрать подходящий тариф или путевку. Согласно исследованию Business Insider, рынок чат-ботов к 2024 году возрастет на 30%. Внедрение чат-ботов поможет сэкономить затраты на поддержку клиентов, а возможность коммуникации с компанией будет доступна в любое время [22].

**1. Программа чат-бот и история ее возникновения**

**1.1 Понятие «чат-бот»**

*Чат-бот* – это программа, специализированная на общении с пользователем при помощи текста или голоса, имитирующая человеческую речь [21]. Основными задачами, выполняемыми чат-ботами, являются понимание естественного языка, ведение диалога и синтез ответа пользователю [16].

**1.2 История возникновения чат-ботов**

История чат-ботов насчитывает около 70 лет [1]. Идея возникновения чат-ботов была придумана Аланом Тьюрингом в 1950 [10]. Именно он первым задался вопросом, можно ли создать программу, поддерживающую с пользователем такой диалог, что его было бы невозможно отличить от реального общения с человеком.

Однако первый чат-бот был разработан лишь в 1966 году Джозефом Вейценбаумом [12]. Программа под названием *«Элиза»* выполняла работу психотерапевта, реализуя технику активного слушания. Конечно, ее функции были ограничены, но именно она послужила дальнейшему развитию данного направления.

Так, следующим чат-ботом была программа под названием *«Parry»*, созданная в 1972 году американским психиатром Кеннетом Колби, имитирующая поведение больного шизофренией. Главным его отличием считается то, что данный чат-бот имеет «личность», соответственно по-особенному реагирует на сообщения пользователя [2]. Несмотря на это, Parry также являлся чат-ботом с низким уровнем возможностей.

Настоящим прорывом в данном направлении послужило создание *«A.L.I.C.E»* в 1995 году. Это чат-бот, способный поддерживать диалог на любом естественном языке и на любую тему. Конечно, он так же, как и другие, был далек от совершенства, и не мог генерировать человекоподобные ответы с выражением эмоций [11].

В 2001 году был сделан еще один шаг в развитии чат-ботов. Была разработана программа с развитием *SmarterChild*, доступная на таких платформах, как America Online и Microsoft, способная помогать людям решать бытовые задачи [7]. Всю информацию бот брал из различных источников, таких, как новости, погода, фильмы, спортивные результаты и др.

Первым персональным помощников в истории чат-ботов стала так называемая «*Siri»*, принадлежащая компании Apple. Siri дает рекомендации пользователю в виде аудио-, видео- и графических ответов, на основе различных Интернет-ресурсов [8]. Еще один персональный помощник по имени *«Cortana»*, разработанный в 2014 году, был способен анализировать и реагировать на голосовые команды пользователя. Так, он выполнял различные функции, как поддержка напоминаний, рассылка писем, общение с пользователем, игры, определение времени и др.

Новый этап в развитии чат-ботов случился в 2016 году, когда на платформах социальных сетей появилась возможность создавать ботов для улучшения сервиса собственных брендов [1]. Так, огромное количество ботов могло обмениваться сообщениями с клиентами абсолютно разных областей, отвечать на вопросы и решать их проблемы.

**1.3 Классификация чат-ботов**

В настоящее время существуют два вида классификации чат-ботов. Так, выделяются *бизнес-классификация*, разделяющая ботов исходя из выполняемых ими задач, и *техническая классификация*, основанная на принципе их работы [21].

*Бизнес-классификация* разделяет ботов на три категории. К ним относятся разговорные чат-боты, главной задачей которых является поддержание беседы с пользователем, персональные ассистенты, помогающие пользователю выполнять различные задачи и Q&A боты, дающие ответы на конкретные вопросы.

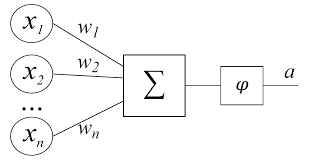
*Техническая классификация* также выделяет три типа чат-ботов. К ним относятся программы, основанные на бизнес-правилах. Диалог с ними ведется по определенному пути, где пользователь принимает решения, но не может получить ответ на вопрос на отвлеченную тему. Противоположностью данного типа являются чат-боты, основанные на искусственном интеллекте. Они не имеют определенного пути и постоянно учатся, используя предыдущие диалоги. Третьим типом являются *гибридные* боты, которые включают в себя элементы первого и второго типов. Таким образом, такие боты ведут диалог по определенному пути, но используют искусственный интеллект при разговоре с пользователем.

**2. Реализация чат-бота**

**2.1 Нейронные сети в обработке естественного языка**

В настоящее время нейронные сети являются лучшим способом распознавания речи и обработки естественного языка [20]. Чат-бот на основе нейронных сетей обучается в процессе работы, именно поэтому он является лучшим интеллектуальным виртуальным помощником на рынке чат-ботов.

*Искусственные нейронные сети* являются объединением искусственных нейронов, которые расположены на слоях внутри нейронной сети. Понятие искусственного нейрона возникло в 1943 году, в его основе лежал искусственный нейрон. Данная идея была представлена в работе Уоррена МакКаллока и Уолтера Питтса, согласно которой в *нейрон* (рис. 1) поступают сигналы, равные бинарным величинам (xi), которые внутри нейрона умножаются на вес (wi), соответствующий входу данного значения [5]. Затем функция активации (φ) преобразует суммарный импульс в выходное значение нейрона.



**Рис. 1.** Модель искусственного нейрона.

Обучение нейронных сетей реализуется за счет определения весов соединений между нейронами. Необходимо сделать это таким образом, чтобы сеть приближала функцию к заданной точности.

Существуют три способа обучения нейронных сетей [20]. Так, при первом методе, имеющем название *обучение с учителем*, на вход нейронной сети поступает набор сигналов с известным результатом, следовательно, веса будут меняться в соответствии с правильностью выходного сигнала. Отличием *обучения без учителя* является то, что нейронная сеть не знает ожидаемый верный результат. Третий подход – *обучение с подкреплением*, в данном случае нейронная сеть так же, как и в предыдущем методе, не получает результат заранее, а выводит его после контакта с внешней средой.

**2.2 Обучение чат-ботов**

Для того чтобы обучить чат-бота необходим большой набор текстовых данных, которые улучшат качество и логичность его ответов.

Существует большое количество источников, находящихся в свободном доступе, которые можно использовать в качестве данных для обучения чат-бота. К данным источникам относятся субтитры к фильмам и сериалам, тексты художественной литературы, сервис микроблогинга Twitter, публичные чаты мессенджеров, комментарии в социальных сетях, различные тематические веб-форумы и др. [16]. Затем полученные данные должны пройти несколько этапов обработки текста: первичный морфологический анализ и векторизация текста, в результате чего будет получена база данных – основа анализа запроса пользователя и получения ответа на него.

**2.3 Сложности при создании чат-бота**

Работа с текстом вызывает множество сложностей при разработке чат-бота. Одной из основных проблем является *мультиязычность*, часто встречающаяся в разговорном языке [3]. Так, в тексте может встречаться сленг и заимствованные из других языков слова или цитаты. Для решения данной проблемы необходима модель, способная обобщать разные языки в общем векторном пространстве [16].

Также важными деталями при разработке чат-бота являются неконсистентность ответов, то есть получение разных ответов на одинаковые по смыслу, но по-разному сформулированные вопросы, и ответы в виде общих фраз. Именно поэтому при создании чат-ботов чаще всего прибегают к выводу ответа из заранее заготовленного набора [16].

Однако важнейшей проблемой является *нечеловечность* чат-бота, которая не позволяет диалогу достичь *реалистичности*. Основным способом ее решения является использование различных источников, располагающих способами коммуникации между людьми. Так, на основе этих данных чат-бот получает различные варианты ответа на пользовательский запрос, которые будут соответствовать реальному общению между людьми [9].

**2.4 Морфологический анализ данных**

Для наиболее качественного получения признаков из исходного текста для его дальнейшей классификации необходимо проводить первичный морфологический анализ полученных на входе данных [14]. С помощью *классификации* текста чат-бот подбирает наиболее подходящие запросу ответы.

*Предварительная обработка текста* включает в себя несколько этапов, каждый из которых необходим для дальнейшего обучения программы.

**2.4.1 Токенизация**

*Токенизация* представляет собой разбиение текста на более мелкие части, такие как абзацы, предложения и слова, которые имеют название *токены*. Существует множество библиотек и программ-токенизаторов в особенности для таких популярных языков программирования, как Python и JavaScript. Для токенизации текста в Python чаще всего используются такие библиотеки, как «NLTK» и «re» [15].

**2.4.2 Нормализация**

Основной задачей *нормализации* является приведение запроса к стандартному виду, то есть преобразование текста к единому регистру, удалению знаков препинания и словесному написанию чисел [14]. Все эти действия можно осуществить благодаря доступным функциям работы со строками различных языков программирования.

**2.4.3 Лемматизация**

*Лемматизация* отвечает за преобразование слов к своей первоначальной форме. Так, глаголы приводятся в инфинитивную форму, а существительные и прилагательные к именительному падежу.

Функций, реализующих лемматизацию, как правило, нет в базовом наборе библиотек у какого-либо языка программирования. Однако существует множество доступных для установки программ и библиотек, в их числе такие популярные библиотеки, как «NLTK», «SpaCy» и PyMorphy2, однако лишь PyMorphy2 поддерживает возможность обработки текста на русском языке [14].

**2.4.4 Удаление стоп-слов**

Данный этап можно осуществить путем удаления всех шумовых слов, то есть за счет исключения всех слов, которые не удалось идентифицировать. Такие слова не должны влиять на смысловое значение полученного запроса, поэтому данный этап реализует функцию очистки текста от лишнего.

Удаление шумовых слов можно произвести с помощью встроенного списка стоп-слов в библиотеке «NLTK» на Python. Также можно создать собственный список слов, которые не несут смысловой нагрузки в текст, чтобы в дальнейшем программа могла их удалить, не изменив суть запроса.

**2.5 Векторизация текста**

После проведения всех этапов предобработки запроса пользователя полученный текст необходимо преобразовать в векторный формат. В процессе векторизации текстовые данные кодируются в числовой вид, которые на основе заданных алгоритмов будут формировать логичный ответ пользователю [17].

Существует большое количество *моделей векторизации текста*. Так, в модели «TF-IDF» [4] вычисляется произведение частотности слов в тексте и обратной частотности слова в базе данных. *Частотность* слов определяет вероятность появления слова в тексте, а *обратная частотность* его уникальность. Размерность векторов зависит от количества слов в запросе. Так, наибольший вес приобретают слова, часто встречающиеся в конкретном запросе, а не в изначальной базе данных.

Методы векторизации текста получили наибольшую популярность после возникновения нового подхода в 2013 году, который получил название «*Word2Vec*». В данном алгоритме векторное представление показывает контекстную близость слов, опираясь на модель непрерывного мешка слов [6].

Перечисленные выше методы называются *статическими*, так как в них не учитываются многозначность слов и их контекстно-зависимая природа [17]. Для решения данной проблемы были созданы *динамические* методы векторизации текста. Так, наиболее успешной оказалась модель под названием «*BERT*», повлекшая дальнейшее развитие данного направления. Данная модель может восстанавливать текст на основе контекста, что имеет название *эмбеддинг*.

Векторизация слов русского языка часто вызывает трудности, так как имеется не так много готовых моделей для данной работы. Однако, существуют русскоязычные проекты свободного доступа, подходящие для различных целей. Так, проект *RusVectores* учитывает часть речи слов при векторизации и хранит эту информацию непосредственно в самом слове [23]. Другой масштабный проект *Russian Distributional Thesaurus* предоставляет доступ к векторам слов (word embeddings), графам подобия слов (дистрибутивный тезаурус) и многим другим полезным данным для векторизации текстов [22]. Для создания данного проекта использовалась модель *skip-gram*, реализованная вышеупомянутым методом *word2vec*.

**2.6 Работа с Telegram при разработке чат-бота**

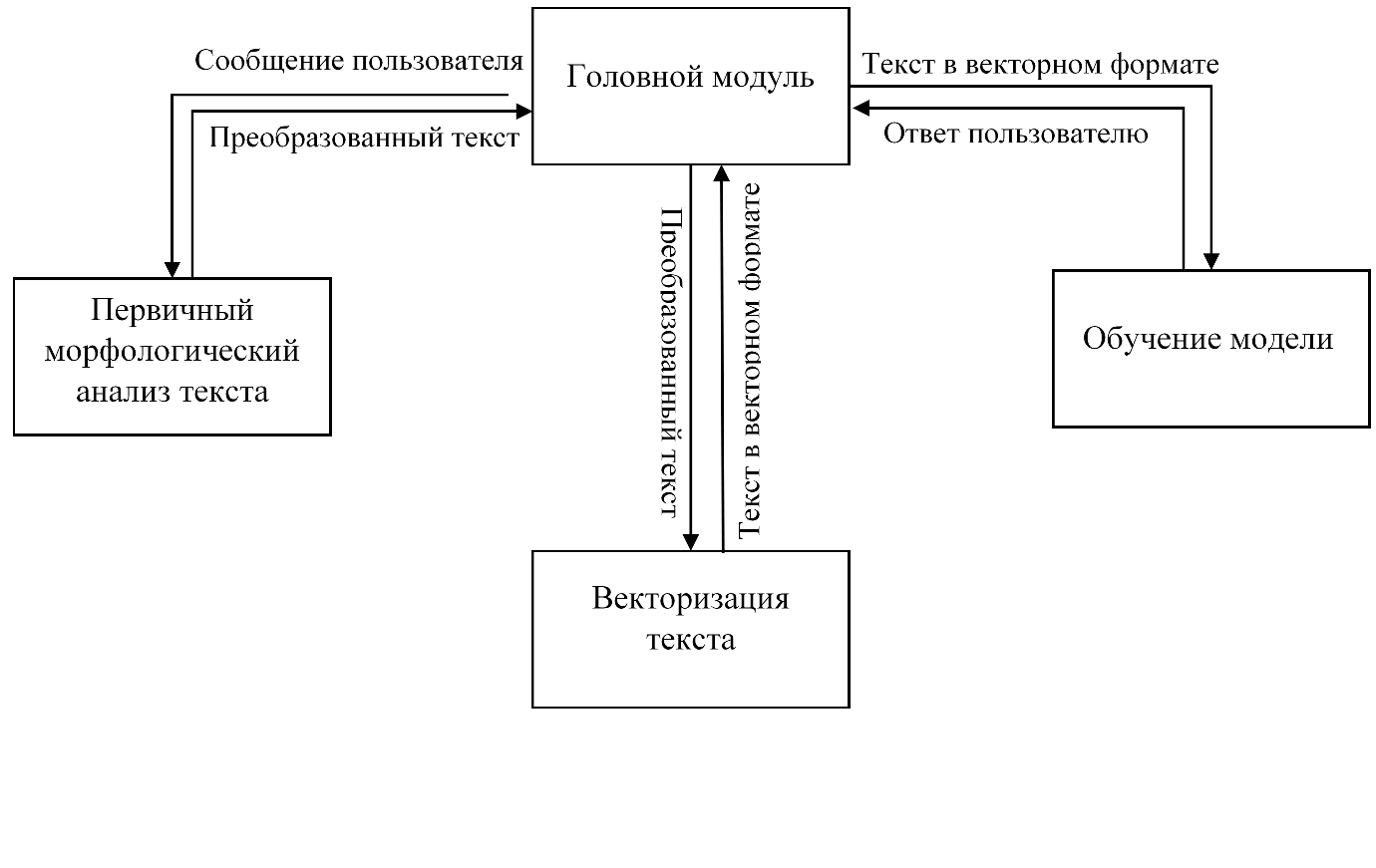
В настоящее время наиболее удобным мессенджером, поддерживающим подключение чат-бота, является Telegram. Начиная с 2015 года, Telegram добавил функцию создания и подключения собственного чат-бота, что повлекло за собой появление различных Интернет-помощников по реализации широкого спектра функций [18].

Для подключения собственного чат-бота к Telegram существует специальная библиотека «*pyTelegramBotAP*» для Python. Данный модуль включает в себя различные редакторы, такие как добавление команд, базовых сообщений и др. [13].

**3. Проектирование**

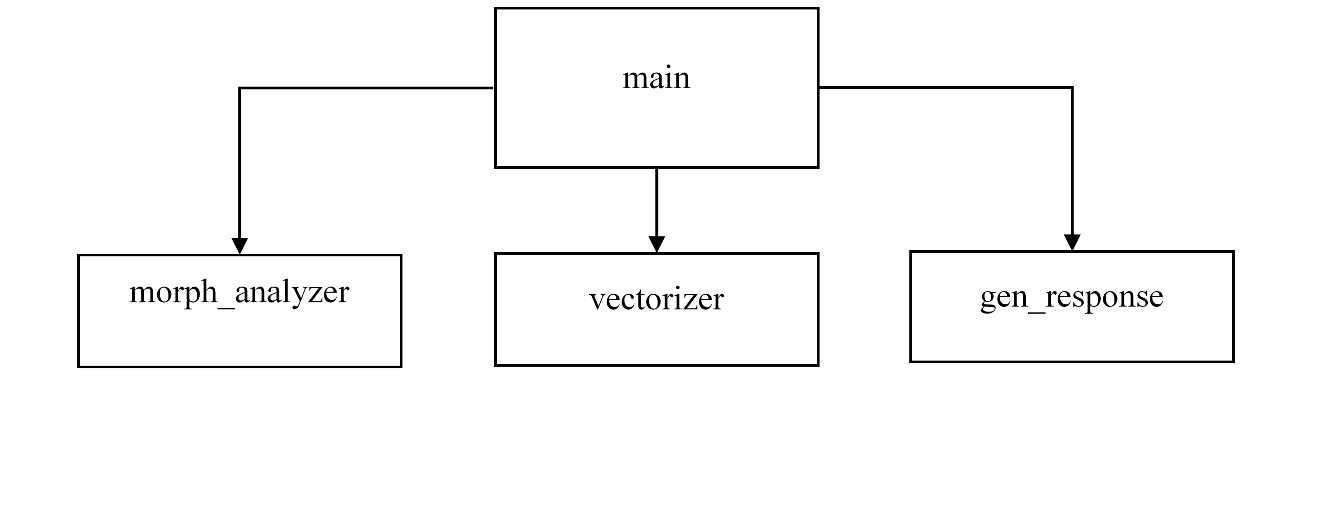
**3.1 Архитектура программы**

Архитектура программы представлена на рисунке 2. Чат-бот реализуется в виде трехэтапного процесса, состоящего из первичного морфологического анализа текста, векторизации текста и обучения модели. Так, перед обучением нейронной сети текст проходит этапы обработки и векторизации. Взаимодействие с чат-ботом реализовано как утилита командной строки.

**Рис. 2.** Архитектура программы.

**3.2 Модульная структура программы**

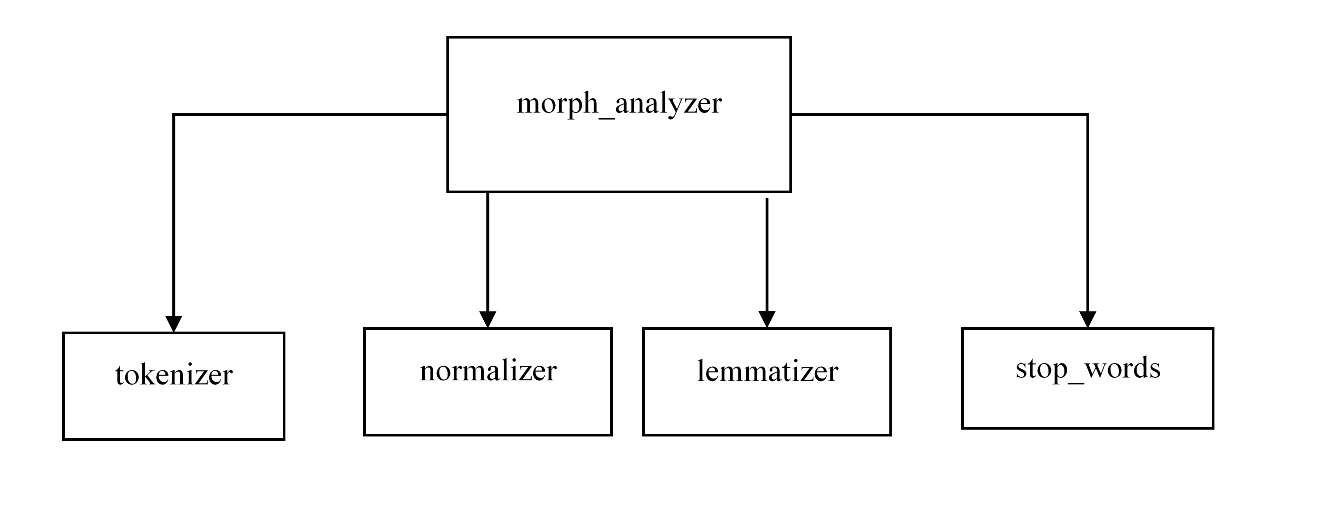
На рисунке 3 представлена модульная структура программы в общем виде. Головной модуль main управляет запуском программного продукта. Соответственно, он вызывает функции morph\_analyzer, vectorizer и gen\_response.



**Рис. 3.** Модульная структура программы.

### **3.2.1 Модуль «Морфологический анализ текста»**

Первичный морфологический анализ реализуется в модуле morph\_analyzer, который преобразовывает сообщение пользователя в формат, подходящий для дальнейшей с ним работы. Морфологический анализ включает в себя такие этапы, как токенизация, нормализация, лемматизация и удаление стоп-слов (рис. 4). Для удобства дальнейшей работы, входной текст сразу приводится к нижнему регистру.



**Рис. 4.** Модульная структура модуля "Морфологический анализ текста".

На этапе токенизации в подмодуле tokenizer запрос пользователя записывается в двумерный массив, состоящий из сообщения, разбитого на предложения, а предложения, в свою очередь, на слова.

Подмодуль normalizer отвечает за нормализацию текста: удаляет все знаки препинания в массиве. Для реализации данного процесса используются функции базового набора python.

В подмодуле lemmatizer, реализующем лемматизацию, элементы массива приводятся к своей начальной форме. Так, глаголы представляются в своей инфинитивной форме, существительные, местоимения и прилагательные приводятся к форме именительного падежа единственного числа мужского рода.

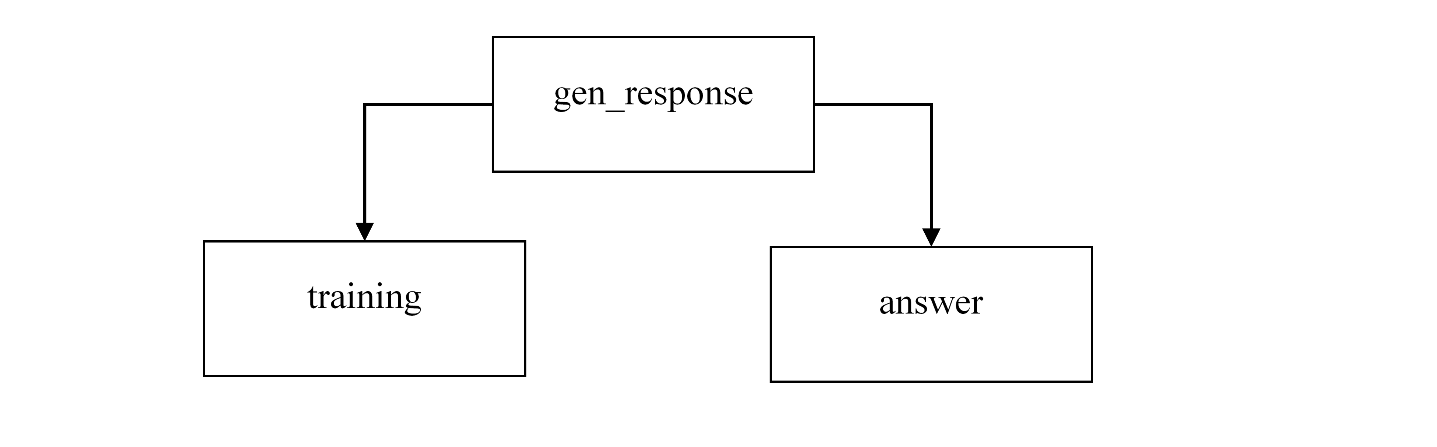
Последний этап морфологического анализа – удаление стоп-слов. Он осуществляется в подмодуле морфологического анализа stop\_words. Стоп-слова русского языка содержатся в отдельном файле.

### **3.2.2 Модуль «Векторизация текста»**

Модуль vectorization приводит сообщение пользователя, преобразованное морфологическим анализом, и тексты, требующиеся для обучения нейронной сети в векторный формат. Для реализации данного модуля используется готовая компактная модель*,* поддерживающая русский язык, от проекта *Russian Distributional Thesaurus.* Данный модуль приводит слова к векторному формату и определяет подобия слов.

### **3.2.3 Модуль «Генерация ответа»**

Главной задачей данного модуля является генерация логичного ответа пользователю на его запрос. Модуль gen\_response (рис. 5) реализуется с помощью готовой библиотеки chatterbot, которая предоставляет возможность работы с чат-ботом. Для начала необходимо обучить бота на основе готовых текстов, для чего потребуется отдельный подмодуль training. Обучение будет происходить на основе файла, содержащего основные возможные темы для разговора. Второй подмодуль answer отвечает за генерацию ответа на конкретный запрос пользователя.



**Рис. 5.** Модуль «Генератор ответа».

**4. Реализация**

Файловая структура программы представлена на рисунке 6.



Рис. 6. Файловая структура

## **4.1 Морфологический анализ текста**

Из файлов tokenizer.py, normalizer.py, lemmatizer.py и stop\_words.py экспортируются функции tokenizer, normalizer, lemmatizer и stop\_words соответственно в файл morph\_analyzer.py.

В файле tokenizer.py реализуется токенизация текста используются функции python-библиотеки nltk, в результате чего исходный текст (сообщение пользователя) делится на предложения и слова в каждом из них.

Файл normalizer.py содержит функцию, отвечающую за нормализацию текста, которая удаляет все встреченные в полученном параметре несоответствующие условиям символы: знаки препинания, специальные символы и т.д. Реализация данного действия выполняется за счет встроенных функций.

Функция лемматизации содержится в файле lemmatizer.py, выполняющая лемматизацию полученного параметра, в результате чего все токены приводятся к форме-инфинитиву. Данное действие осуществляется за счет использования python-библиотеки *pymorphy2*.

Удаление стоп-слов осуществляется в файле stop\_words.py. Пакет *nltk.corpus* содержит список *stopwords*, который хранит в себе стоп-слова различных языков, в данной работе используются слова только русского языка. Аналогично предыдущему пункту, каждый токен проверяется на вхождение в данный список стоп-слов и удаляется из входного параметра в случае вхождения.

Подробное описание реализации этапа «Морфологический анализ текста» представлено в приложении 1.

## **4.2. Векторизация текста**

Файл vectorizer.py приводит все полученные слова к векторному формату. Для построения векторных представлений слов используется стандартная имплементация *word2vec* проекта *Russian Distributional Thesaurus*. Готовые векторы хранятся в файле *all.norm-sz100-w10-cb0-it1-min100.w2v* – готовом компактном наборе векторов слов. Для использования векторов слов применяется пакет *gensim*. В векторизации текста применяются функции нахождения векторов и подобия слов.

Подробное описание реализации этапа «Векторизация текста» представлено в приложении 2.

## **4.3. Генерация ответа**

Файл gen\_response.py импортирует файлы answer.py и training.py. Ответ на запрос пользователя выдается с помощью базового чат-бота библиотеки chatterbot, который обучается в файле training.py на основе заранее подготовленных данных. Готовый чат-бот отвечает на запросы пользователя в функции файла answer.py.

Подробное описание реализации этапа «Генерация ответа» представлено в приложении 3.

# 5. Тестирование

Для данного раздела были разработаны автономные тесты, ниже можно увидеть выдаваемые результаты модулей.

В первом тесте введен простой текст с приветствием.

Во втором тесте введен стандартный вопрос для чат-бота. Символы входного текста записаны в разных регистрах.

В третьем тесте представлен вариант, когда введен непонятный боту текст, не имеющий смысла.

## **5.1. Морфологический анализ текста**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемые  выходные данные** | **Реальные  выходные данные** | **Совпадение (Да/Нет)** |
| 1 | Доброго дня! Я рад видеть тебя. Как твои дела? | [['добрый', 'день'], ['рад', 'видеть'], ['твой', 'дело']] | [['добрый', 'день'], ['рад', 'видеть'], ['твой', 'дело']] | Да |
| 2 | РаСсКаЖи мне, что ЗНАЧИТ ЧАТ-БОТ? | [['рассказать', 'значит', 'чат-бот']] | [['рассказать', 'значит', 'чат-бот']] | Да |
| 3 | абвгд!!! \*#%^ (}{) | [['абвгд'], []] | [['абвгд'], []] | Да |

### **5.1.1. Токенизация**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемые  выходные данные** | **Реальные  выходные данные** | **Совпадение (Да/Нет)** |
| 1 | Доброго дня! Я рад видеть тебя. Как твои дела? | [['доброго', 'дня', '!'], ['я', 'рад', 'видеть', 'тебя', '.'], ['как', 'твои', 'дела', '?']] | [['доброго', 'дня', '!'], ['я', 'рад', 'видеть', 'тебя', '.'], ['как', 'твои', 'дела', '?']] | Да |
| 2 | РаСсКаЖи мне, что ЗНАЧИТ ЧАТ-БОТ? | [['расскажи', 'мне', ',', 'что', 'значит', 'чат-бот', '?']] | [['расскажи', 'мне', ',', 'что', 'значит', 'чат-бот', '?']] | Да |
| 3 | абвгд!!! \*#%^ (}{) | [['абвгд', '!', '!', '!'], ['\*', '#', '%', '^', '(', '}', '{', ')']] | [['абвгд', '!', '!', '!'], ['\*', '#', '%', '^', '(', '}', '{', ')']] | Да |

### **5.1.2. Нормализация**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемые  выходные данные** | | | **Реальные  выходные данные** | **Совпадение (Да/Нет)** |
| 1 | [['доброго', 'дня', '!'], ['я', 'рад', 'видеть', 'тебя', '.'], ['как', 'твои', 'дела', '?']] | | [['доброго', 'дня'], ['я', 'рад', 'видеть', 'тебя'], ['как', 'твои', 'дела']] | [['доброго', 'дня'], ['я', 'рад', 'видеть', 'тебя'], ['как', 'твои', 'дела']] | | Да |
| 2 | [['расскажи', 'мне', ',', 'что', 'значит', 'чат-бот', '?']] | | [['расскажи', 'мне', 'что', 'значит', 'чат-бот']] | [['расскажи', 'мне', 'что', 'значит', 'чат-бот']] | | Да |
| 3 | [['абвгд', '!', '!', '!'], ['\*', '#', '%', '^', '(', '}', '{', ')']] | | [['абвгд'], []] | [['абвгд'], []] | | Да |

### **5.1.3. Лемматизация**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемые  выходные данные** | **Реальные  выходные данные** | **Совпадение (Да/Нет)** |
| 1 | [['доброго', 'дня'], ['я', 'рад', 'видеть', 'тебя'], ['как', 'твои', 'дела']] | [['добрый', 'день'], ['я', 'рад', 'видеть', 'ты'], ['как', 'твой', 'дело']] | [['добрый', 'день'], ['я', 'рад', 'видеть', 'ты'], ['как', 'твой', 'дело']] | Да |
| 2 | [['расскажи', 'мне', 'что', 'значит', 'чат-бот']] | [['рассказать', 'я', 'что', 'значит', 'чат-бот']] | [['рассказать', 'я', 'что', 'значит', 'чат-бот']] | Да |
| 3 | [['абвгд'], []] | [['абвгд'], []] | [['абвгд'], []] | Да |

### **5.1.4. Удаление стоп-слов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемые  выходные данные** | **Реальные  выходные данные** | **Совпадение (Да/Нет)** |
| 1 | [['добрый', 'день'], ['я', 'рад', 'видеть', 'ты'], ['как', 'твой', 'дело']] | [['добрый', 'день'], ['рад', 'видеть'], ['твой', 'дело']] | [['добрый', 'день'], ['рад', 'видеть'], ['твой', 'дело']] | Да |
| 2 | [['рассказать', 'я', 'что', 'значит', 'чат-бот']] | [['рассказать', 'значит', 'чат-бот']] | [['рассказать', 'значит', 'чат-бот']] | Да |
| 3 | [['абвгд'], []] | [['абвгд'], []] | [['абвгд'], []] | Да |

## **5.2. Векторизация текста**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемые  выходные данные** | **Реальные  выходные данные** | **Совпадение (Да/Нет)** |
| 1 | [['добрый', 'день'], ['рад', 'видеть'], ['твой', 'дело']] |  |  |  |
| 2 | [['рассказать', 'значит', 'чат-бот']] |  |  |  |
| 3 | [['абвгд'], []] |  |  |  |

## **5.3. Генерация ответа**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Входные данные** | **Ожидаемые  выходные данные** | **Реальные  выходные данные** | **Совпадение (Да/Нет)** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |

# 6. Работа итогового приложения

В данном разделе представлена работа итогового приложения, ниже можно увидеть диалоги с разработанным чат-ботом.

# Заключение

Чат-бот – это программа, формирующая корректные, логически обоснованные ответы, схожие с реальным разговором людей, на основе полученного от пользователя запроса.

В настоящее время чат-боты используются в широком спектре и затрагивают большое количество сфер. В основном они реализуют функции персонального помощника в коммерческих компаниях и функцию поддержания диалога в различных сервисах. Чат-боты способны решать различные задачи, такие как общение, развлечение, медицинские консультации, проведение финансовых услуг, заказа товаров и др. Для получения логических ответов на запрос пользователя важно провести его морфологический анализ, а затем представить данные в векторном формате. Чтобы добиться создания интеллектуального чат-бота, необходимо решить следующие проблемы: мультиязычность, неконсистентность и нечеловечность.

В ходе работы были исследованы различные способы реализации чат-ботов, включая процессы обработки текста, преобразования его в векторную форму и обучение нейронной сети. В результате исследований, были выбраны наиболее подходящие методы реализации и выполнено проектирование модульной структуры чат-бота. На основе модульной структуры была написана программа на языке программирована Python, реализующая простого чат-бота. Также было разработано и успешно пройдено тестирование.

**Список литературы**

1. Eleni Adamopoulou, Lefteris Moussiades. CHATBOT: Architecture, Design, & Development // Machine Learning with Applications. 2020.
2. Colby K.M., Weber S., Hilf F.D. Artificial paranoia. Artificial Intelligence. 1971. Vol. 2, No 1. P. 1–25.
3. Holger S., Douze M. Learning Joint Multilingual Sentence Representations with Neural Machine Translation // ACL workshop on Representation Learning for NLP. 2017. P. 157-167.
4. Manning C.D., Raghavan P., Schütze H. An Introduction to Information Retrieval // Cambridge University Press, 2009, p. 581.
5. McCulloch W.S., Pitts W. A Logical Calculus of the Ideas Immanent in Nervous Activity // The Bulletin of Mathematical Biophysics. 1943. Vol. 5, No. 4. P. 115–133.
6. Mikolov T. Distributed Representations of Words and Phrases and their Compositionality // Proceedings of Workshop at ICLR. 2013.
7. Molnar G., Zoltán S. The role of chatbots in formal education. International Symposium on Intelligent Systems and Informatics. 2018.
8. Nora Ni Loideain, R. Adams. From Alexa to Siri and the GDPR: The gendering of Virtual Personal Assistants and the role of Data Protection Impact Assessments // Computer Law & Security Review. 2020. Vol. 36.
9. Rui Yan. “Chitty-Chitty-Chat Bot”: Deep Learning for Conversational AI // Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. 2018.
10. Turing A.M. Computing machinery and intelligence. Mind. 1950. Vol. LIX, No. 236. P. 433–460.
11. Wallace R.S., R. Epstein, G. Roberts, G. Beber. The anatomy of A.L.I.C.E. Parsing the turing test: philosophical and methodological issues in the quest for the thinking computer. 2009. P. 181–210.
12. Weizenbaum, J. ELIZA—A computer program for the study of natural language communication between man and machine. Commun. 1966. Vol. 9, No. 1. P. 36–45.
13. Бийбосунов Б.И., Бийбосунова С.К., Жолочубеков Н.Ж. Описание концепции Telegram ботов и их разработка // Colloquium-journal. 2020. Vol. 7, No. 59. P. 3-6.
14. Бородин А.И., Вейнберг Р. Р., Литвишко О. В. Методы обработки текста при создании чат-ботов // Хуманитарни Балкански изследования. 2019. Т. 3, № 5. С. 108-111.
15. Гречанин В.А. К вопросу о токенизации текста // Международный научно-исследовательский журнал. 2016. Т. 6-4, № 48. С. 25-27.
16. 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. 50-66.
17. Жеребцова Ю.А., Чижик А.В. Сравнение моделей векторного представления текстов в задаче создания чат-бота // Вестник НГУ. Серия: Лингвистика и межкультурная коммуникация. 2020. Т. 18, № 3. С. 16-34.
18. Козлов А.А., Батищев А.В. Телеграм-бот как простой и удобный способ получения информации // Территория науки. 2017. № 5, С. 55-64.
19. Параскевов А.В., Каданцева А.А., Мороз С.И. Перспективы и особенности разработки чат-ботов // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. 2017. № 130. С. 395-404.
20. Созыкин А.В. Обзор методов обучения глубоких нейронных сетей // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 3. С. 28–59.
21. Ураев Д.А. Классификация и методы создания чат-бот приложений // International scientific review. 2019. № LXIV. С. 30-33.
22. Russian Distributional Thesaurus [Электронный ресурс]. URL: <https://nlpub.ru/Russian_Distributional_Thesaurus> (дата обращения: 26.12.2020).
23. RusVectōrēs [Электронный ресурс]. URL: <https://rusvectores.org/ru/> (дата обращения: 26.12.2020).
24. Будущее чат-ботов: 10 исследований и прогнозы экспертов [Электронный ресурс]. URL: <https://vc.ru/services/175276-budushchee-chat-botov-10-issledovaniy-i-prognozy-ekspertov> (дата обращения: 26.12.2020).

# 

# Приложение 1. Реализация «Первичный морфологический анализ текста».

**«Первичный морфологический анализ текста»**

В файле morph\_analyzer.py реализуется морфологический анализ текста. Morph\_analyzer.py содержит функцию def morph\_analyzer(raw), входным параметром которой является строка raw. Данная функция вызывает функции импортированных файлов tokenizer.py, normalizer.py, lemmatizer.py и stop\_words.py. Выходным параметром является обработанный массив данный fin\_text.

**«Токенизация»**

Файл tokenizer.py реализует разбиение исходного сообщения пользователя на предложения, которые, в свою очередь, разбиваются на слова. Реализует данные задачи функция def tokenizer(raw), входной параметр raw – строка. В данном подмодуле используется библиотека nltk, осуществляющая функции разделения текста на предложения sent\_tokenize и word\_tokenize соответственно. Входной параметр в начале разделяется на предложения с помощью функции *sent\_tokenize* библиотеки *nltk*, которые записываются в массив sent\_tokens. Данный массив делится на слова в цикле с помощью функции *word\_tokenize* библиотеки *nltk*, которые записываются в двумерный массив token\_text, являющийся выходным параметром.

**«Нормализация»**

Файл normalizer.py отвечает за нормализацию текста: удаление знаков препинания и специальных символов, что реализует функция def normalize(token\_text) с входным параметром token\_text – двумерным массивом слов. Функция сравнивает элементы массива с заданным заранее списком символов, не удовлетворяющих условиям, и добавляет подходящие элементы в новый массив norm\_text. Выходным параметром является двумерный массив norm\_text без знаков препинания.

**«Лемматизация»**

Файл lemmatizer.py осуществляет лемматизацию текста, то есть приведение слов к своей начальной форме. Эту задачу выполняет функция def lemmatizer(norm\_text), входным параметром является двумерный массив norm\_text. Каждый элемент массива приводится к начальной форме после использования функции *MorphAnalyzer* импортированной библиотеки *pymorphy2*. Выходным параметром является массив слов, в который были добавлены лемматизированные элементы входного массива – lem\_text.

**«Удаление стоп-слов»**

Файл stop\_words.py осуществляет удаление стоп-слов в функции def stop\_words(lem\_text) с входным параметром lem\_text – двумерный массив слов. Реализация происходит с помощью пакета *nltk.corpus* и её функции *stopwords*, хранящей в себе стоп-слова различных языков, в данной функции используется только русский язык. Элементы входного параметра в цикле проверяются на вхождение в список стоп-слов и добавляются в массив filtered\_text в случае не вхождения. Выходным параметром является двумерный массив filtered\_text.

# Приложение 2. Реализация «Векторизация текста»

**«Векторизация текста»**

Файл vectorizer.py осуществляет преобразование обработанного сообщения пользователя в массив чисел, что выполняет функция def vectorizer(filtered\_text), где входной параметр filtered\_text – обработанный морфологическим анализом массив слов. Вектора слов хранятся в переменной w2v\_fpath, содержащей данные модели *all.norm-sz100-w10-cb0-it1-min100.w2v*. Для работы с векторами используется пакет *gensim*. Выходным параметром является параметр vector\_text – массив, хранящий в себе информацию о векторах и подобии слов.

# Приложение 3. Модуль «Генерация ответа»

**«Генерация ответа»**

Файл gen\_response.py импортирует файлы answer.py и training.py. Входным параметром является массив text, выходным – bot\_response, строка-ответ пользователю.

**«Обучение нейронной сети»**

В файле training.py осуществляется обучение чат-бота с использование функций *ListTrainer* (обучение конкретным фразам) и *ChatterBotCorpusTrainer* (обучение готовому набору фраз корпуса русского языка *chatterbot.corpus.russian*) библиотеки *chatterbot.* Данный файл не содержит выходного параметра, так как отвечает только за подготовку чат-бота к дальнейшей работе.

**«Получение ответа»**

В файле answer.py содержится функция def answer(user\_response), которая на входе получает преобразованный массив слов user\_response (сообщение пользователя), выходным параметром является строка bot\_response, полученная от обученного чат-бота, которая будет являться ответом на запрос пользователя.