**НИТУ «МИСиС»**

Институт ИТКН

**Кафедра инженерной кибернетики**

Направление подготовки: 01.03.04 прикладная математика Квалификация (степень): бакалавр

**КУРСОВАЯ РАБОТА**

**учебная дисциплина**

**«Методы искусственного интеллекта»**

**VII семестр 2020 – 2021 у.г.**

**Учащийся: *Дементьева Ю. Д.***

**Группа: БПМ-17-1**

**Проверил: доц., к.т.н. А.С. Кожаринов**

**Оценка:**

**Дата защиты:**  23.11.2020

**Москва 2020**

Оглавление

[СПИСОК ИСПОЛЬЗУЕМЫХ ОСНОВНЫХ СОКРАЩЕНИЙ 3](#_Toc58272068)

[***1.*** ***ПОСТАНОВКА ЗАДАЧИ*** 5](#_Toc58272069)

[***1.1* Описание предметной области** 5](#_Toc58272070)

[***1.2* Актуальность задачи** 5](#_Toc58272071)

[***1.3* Цель работы** 5](#_Toc58272072)

[***1.4* Описание работы приложения** 6](#_Toc58272073)

[***2.*** ***ИСПОЛЬЗОВАННЫЕ СРЕДСТВА РАЗРАБОТКИ И СИСТЕМНЫЕ ТРЕБОВАНИЯ*** 7](#_Toc58272074)

[***2.1* Описание языковых и инструментальных средств ПО, использованных для создания приложения** 7](#_Toc58272075)

[***2.2* Системные требования для приложения** 7](#_Toc58272076)

[***3.*** ***СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ*** 8](#_Toc58272077)

[***3.1* Аспекты реализации чат-бота с помощью библиотеки TensorFlow** 8](#_Toc58272078)

[***4.*** ***ОПИСАНИЕ РАБОТЫ БОТА*** 11](#_Toc58272079)

[***5.*** ***СРАВНЕНИЕ РЕЗУЛЬТАТОВ РАБОТЫ ДВУХ ЧАТ-БОТОВ*** 12](#_Toc58272080)

[**5.1. Описание метрик для чат-ботов** 12](#_Toc58272081)

[**5.2 Сравнение бота DialogFlow и Tensorfow по выбранным метрикам** 13](#_Toc58272082)

[***6.*** ***ВЫВОДЫ*** 17](#_Toc58272083)

[***7.*** ***Распределение обязанностей*** 18](#_Toc58272084)

[***ПРИЛОЖЕНИЕ А. Примеры запросов к ботам*** 19](#_Toc58272085)

[***ПРИЛОЖЕНИЕ А. Скрипт для api*** 21](#_Toc58272086)

# СПИСОК ИСПОЛЬЗУЕМЫХ ОСНОВНЫХ СОКРАЩЕНИЙ

1) Api (англ. Application Programming Interface — программный интерфейс приложения) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

2) DialogFlow - это естественная платформа для понимания языка, которая позволяет легко проектировать и интегрировать разговорный пользовательский интерфейс в мобильное приложение, веб-приложение, устройство, бот, интерактивную систему голосового ответа и так далее.

3) Tensorflow - открытая программная библиотека для машинного обучения, разработанная компанией [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Google (компания)) для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия.

4) Flask -  [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для создания [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) на языке программирования [Python](https://ru.wikipedia.org/wiki/Python" \o "Python).

5) Heroku -  полноценная среда, поддерживающая ряд языков программирования, разработки и развертывания в облаке с ресурсами, которые позволяют предоставлять любые приложения, от простых облачных приложений до продвинутых облачных приложений.

6) PyTelegramBotApi - библиотека, контролирующая отправку и получение HTTP-запросов при взаимодействии чат-бота с пользователем.

7) Лемматизация - это процесс преобразования слова в его первоначальную неизменённую форму (лемму).

8) Токенизация - это процесс разделения предложений или слов из корпуса на небольшие единицы, то есть токены. Один из заключительных этапов языковой обработки, преобразующий неструктурированный текстовый материал в данные.

9) Dataset - обработанный набор очищенных данных, пригодных для обработки алгоритмами машинного обучения.

10) Оптимизаторы — это расширенный класс, который включает дополнительную информацию для обучения конкретной модели. Класс оптимизатора инициализируется с заданными параметрами, но важно помнить, что тензор не нужен. Оптимизаторы используются для повышения скорости и производительности при обучении конкретной модели.

# ***ПОСТАНОВКА ЗАДАЧИ***

## ***1.1* Описание предметной области**

Разрабатываемое приложение – это бот-помощник, задача которого подсказать пользователю правильное название функции Excel, соответствующей определенной статистической формуле.

Программа Excel, разрабатываемая и поддерживаемая компанией Microsoft, предоставляет множество формул, которые существенно облегчают статистические подсчеты. На данный момент насчитывается 104 статистические формулы. Однако, многие из них имеют похожие названия. В пример можно привести функции, связанные с распределением Стьюдента: «[СТЬЮДРАСП](https://support.microsoft.com/ru-ru/office/%D1%81%D1%82%D1%8C%D1%8E%D0%B4%D0%B5%D0%BD%D1%82-%D1%80%D0%B0%D1%81%D0%BF-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F-%D1%81%D1%82%D1%8C%D1%8E%D0%B4%D0%B5%D0%BD%D1%82-%D1%80%D0%B0%D1%81%D0%BF-4329459f-ae91-48c2-bba8-1ead1c6c21b2)», «[СТЬЮДЕНТ.РАСП.2Х](https://support.microsoft.com/ru-ru/office/%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F-%D1%81%D1%82%D1%8C%D1%8E%D0%B4%D0%B5%D0%BD%D1%82-%D1%80%D0%B0%D1%81%D0%BF-2%D1%85-198e9340-e360-4230-bd21-f52f22ff5c28)», «[СТЬЮДЕНТ.РАСП.ПХ](https://support.microsoft.com/ru-ru/office/%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F-%D1%81%D1%82%D1%8C%D1%8E%D0%B4%D0%B5%D0%BD%D1%82-%D1%80%D0%B0%D1%81%D0%BF-%D0%BF%D1%85-20a30020-86f9-4b35-af1f-7ef6ae683eda)», «[СТЬЮДЕНТ.ОБР](https://support.microsoft.com/ru-ru/office/%D1%81%D1%82%D1%8C%D1%8E%D0%B4%D0%B5%D0%BD%D1%82-%D0%BE%D0%B1%D1%80-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F-%D1%81%D1%82%D1%8C%D1%8E%D0%B4%D0%B5%D0%BD%D1%82-%D0%BE%D0%B1%D1%80-2908272b-4e61-4942-9df9-a25fec9b0e2e)», «[СТЬЮДЕНТ.ОБР.2Х](https://support.microsoft.com/ru-ru/office/%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F-%D1%81%D1%82%D1%8C%D1%8E%D0%B4%D0%B5%D0%BD%D1%82-%D0%BE%D0%B1%D1%80-2%D1%85-ce72ea19-ec6c-4be7-bed2-b9baf2264f17)», «[СТЬЮДЕНТ.ТЕСТ](https://support.microsoft.com/ru-ru/office/%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F-%D1%81%D1%82%D1%8C%D1%8E%D0%B4%D0%B5%D0%BD%D1%82-%D1%82%D0%B5%D1%81%D1%82-d4e08ec3-c545-485f-962e-276f7cbed055)». Сразу 6 формул имеют схожие названия несмотря на то, что рассчитывают разные параметры и величины. Человеку, плохо разбирающемуся в статистических формулах или имеющему небольшой опыт работы с программой Excel, придется потратить достаточно много времени на выбор нужной формулы. Также, не заглядывая в документацию Excel, бывает сложно разобраться в параметрах, передаваемых на вход функции.

## ***1.2* Актуальность задачи**

Согласно источнику «https://rb.ru/» растет востребованность профессии «Аналитик данных». Так, по результатам исследования в 2019 году, спрос на специалистов по анализу данных вырос на 226%. Одним из инструментов, которые используют аналитики в своей работе это программа Excel, особенно востребованным является блок со статистическими формулами. Поэтому задача реализации приложения, которое бы экономило бы время на поиске нужных функций excelактуальна.

## ***1.3* Цель работы**

Разработать чат - бота, подсказывающего формулы Excel.

## ***1.4* Описание работы приложения**

Пользователь пишет боту название величины, которую хочет посчитать. Если находится одно единственное совпадение, то бот сразу выводит пользователю название формулы с кратким описанием и ссылкой на официальную документацию Excel. Если же, алгоритм находит несколько подходящих формул, пользователь получает уточняющие вопросы. В процессе получения ответов на вопросы бот находит наиболее подходящую функцию и присылает информацию о ней.

# ***ИСПОЛЬЗОВАННЫЕ СРЕДСТВА РАЗРАБОТКИ И СИСТЕМНЫЕ ТРЕБОВАНИЯ***

## ***2.1* Описание языковых и инструментальных средств ПО, использованных для создания приложения**

В качестве платформы для запуска бота был выбран мессенджер Telegram. Для реализации логики работы бота были выбраны два пути. Первый – использование сервиса с открытым доступом «DialogFlow», разрабатываемый компанией Google. На этой платформе достаточно указать так называемые «намерения» - список возможных запросов пользователя и ответов бота. Затем, бот обучается. Процесс обучения модели можно контролировать, путем обозначения правильных и неправильных ответов. Второй путь – написание собственного алгоритма на языке pythonс использованием библиотек Tensorflow и PyTelegramBotApi. Для взаимодействия telegram-бота с моделью нейронной сети, предсказывающей формулы excel, используется api, написанное с помощью фреймворка Flask. Сама модель размещена на облачной платформе Heroku. Выбор двух разных реализаций программы был сделан с целью дальнейшего анализа результатов и определения наиболее выгодного решения.

## ***2.2* Системные требования для приложения**

Для запуска бота пользователю достаточно установить приложение Telegram, десктопную или мобильную версию, а также создать свой аккаунт в этой сети и в поиске найти бота по названию.

# ***СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ***

Для обоих способов реализации приложения (как посредством DialogFlow, так и с помощью библиотеки Tensorflow) будет использоваться клиент-серверная модель взаимодействия. В случае с первым ботом, когда обучение будет происходить с помощью алгоритмов DialogFlow, клиентское приложение будет посылать запросы серверам Googleи соответственно получать ответы. В случае второго бот, который написан с помощью фреймворка Flask, приложение так же будет обращаться с помощью POST-запросов к модели, которая размещена на сервере Heroku. Api-метод принимает только один параметр «message», который содержит сообщение пользователя. Диаграмма взаимодействия бота с сервером Heroku с помощью api представлена на рис. 1.

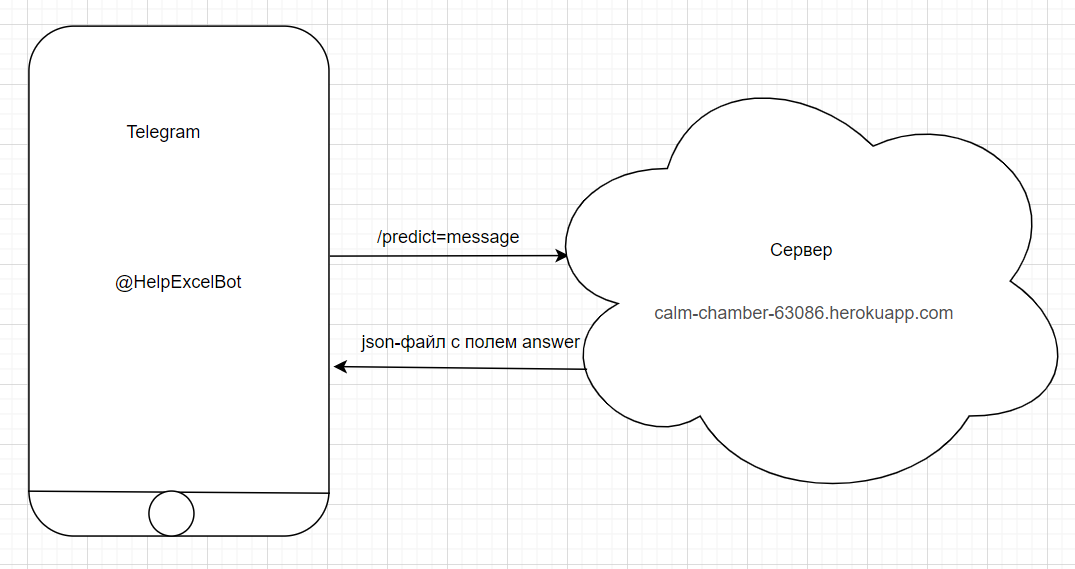


Рисунок 1 – Взаимодействие бота с сервером

## ***3.1* Аспекты реализации чат-бота с помощью библиотеки TensorFlow**

Модель на вход получает dataset в формате json, в котором содержатся возможные вопросы и ответы на них. Скриншот файла можно посмотреть на рис. 2.

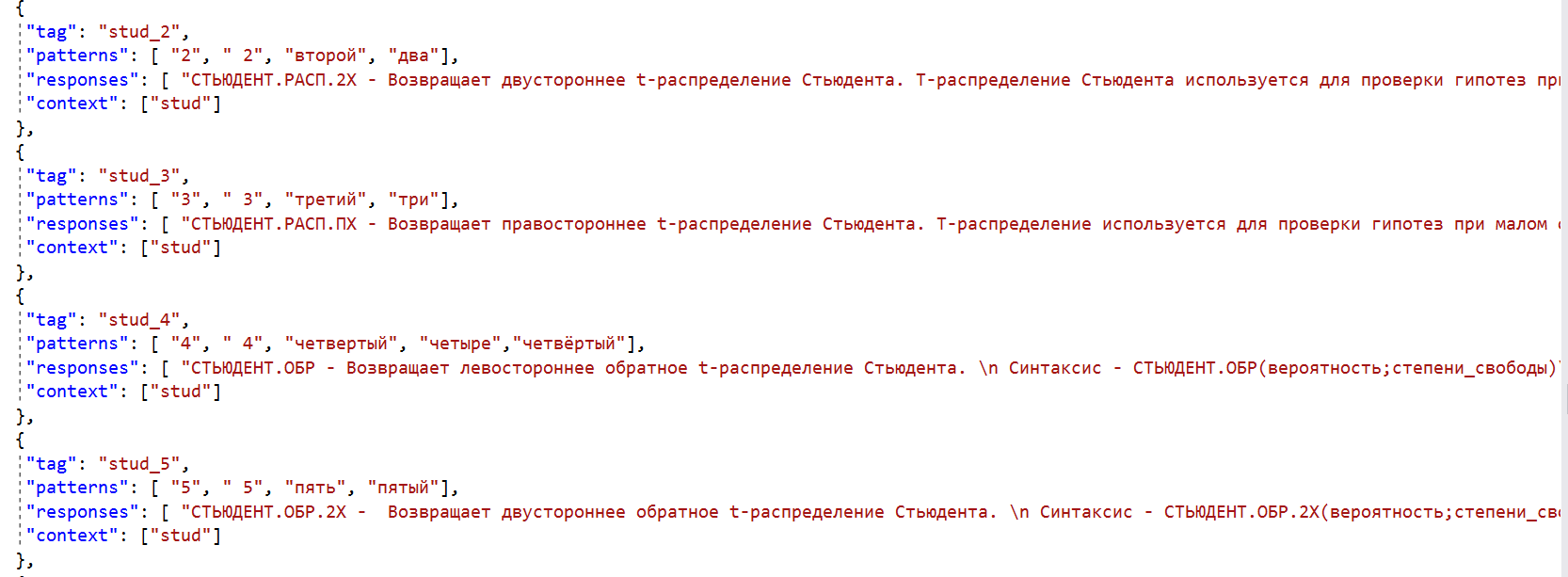


Рисунок 2 – Разметка файла Intents.json

Параметр «tag» используется для того, чтобы различать блоки вопросов и ответов, а также, чтобы можно было ссылаться на них, тем самым реализовывать возможность ответа на контекстные вопросы.

Затем считанные данные преобразуются в пригодный для дальнейшей работы вид с помощью техник лемматизации и токенизации, т.е. слова приводятся в начальную форму и сопоставляются конкретному числовому значению. После этого данные разбиваются на тестовый и тренировочный листы.

Следующим этапом создания чат-бота является обучение модели на подготовленных данных. В Tensorflow используются несколько оптимизаторов, одними из которых являются инерция, нестеров импульс, Adagrad, Adadelta и другие. Для обучения данной модели был взят оптимизатор "Стохастический градиентный спуск". При данном алгоритме значение градиента аппроксимируются градиентом функции стоимости, вычисленном только на одном элементе обучения. Затем параметры изменяются пропорционально приближенному градиенту. Таким образом параметры модели изменяются после каждого объекта обучения. Для больших массивов данных стохастический градиентный спуск может дать значительное преимущество в скорости по сравнению со стандартным градиентным спуском, что послужило весомым аргументов в выборе метода оптимизации. Блок-схема для данного метода представлена на рисунке 3.

Чтобы предотвратить недообученность модели, при принято увеличить количество эпох (epoch) до двухсот, тем самым, веса нейронной сети изменяются все большее количество раз, с каждым разом лучше подстраивая кривую градиентного спуска под данные, переходя последовательно из плохо обученного состояния в **оптимальное.** В качестве функции потерь используется перекрестная энтропия («categorical\_crossentropy»). После обучения, найденные веса и архитектура модели сохраняется в файл формата «h5».

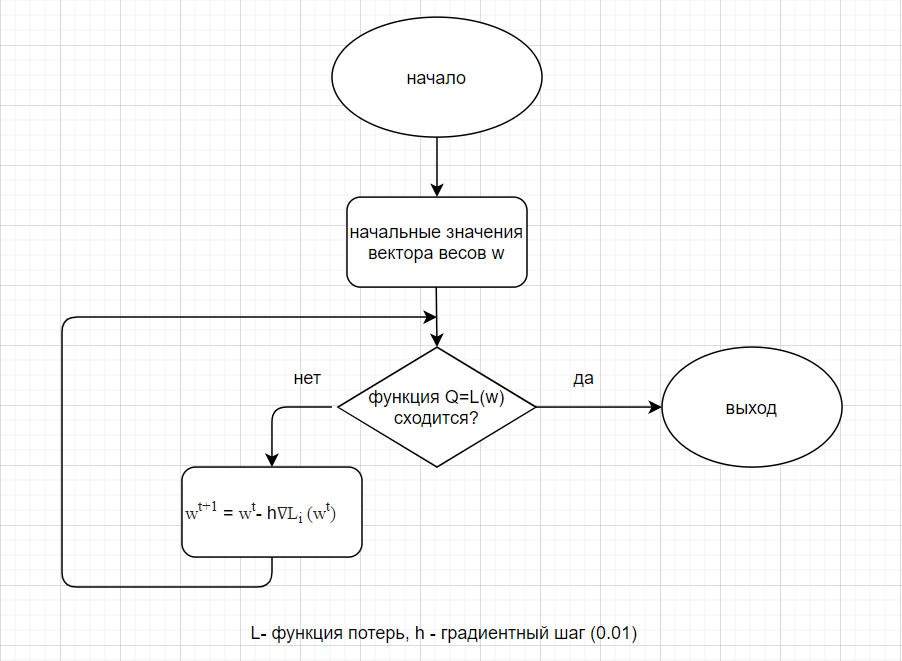


Рисунок 3 – алгоритм стохастического градиентного спуска

# ***ОПИСАНИЕ РАБОТЫ БОТА***

Для начала работы с ботом пользователю надо найти его в приложении Telegram. Версию бота, реализованную с помощью DialogFlow, можно найти по адресу «@AskExcelBot», версию с Tensorflow-моделью по адресу «@HelpExcelBot». После успешного нахождения виртуальных помощников, необходимо ввести стандартную команду «/start», которая активирует ботов. Затем, пользователь может писать любые интересующие его вопросы, и если же, в dataset, на котором учился бот, были соответствующие ответы, то он пришлет их в качестве ответного сообщения, если же, вопрос был не предусмотрен, то пользователь получит сообщение «простите, я Вас не понял». Как видно из рисунков 4-5, пользовательский интерфейс, а также логика работа с приложением не отличается в обоих ботах.

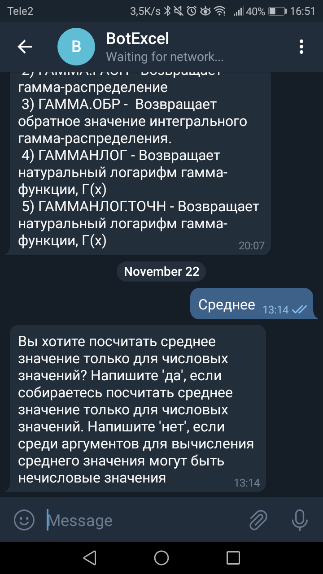


Рисунок 4 – Бот «BotExcel», ответ на запрос «Среднее»

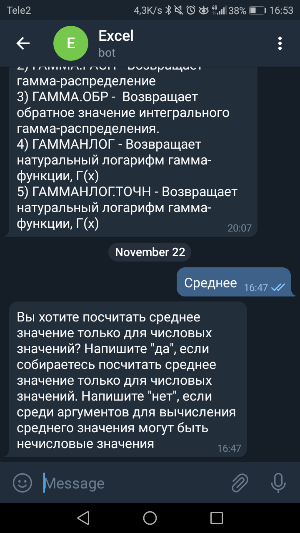


Рисунок 5 – Бот «Excel», ответ на запрос «Среднее»

# ***СРАВНЕНИЕ РЕЗУЛЬТАТОВ РАБОТЫ ДВУХ ЧАТ-БОТОВ***

## **5.1. Описание метрик для чат-ботов**

Чат-бот приложения с внедрением алгоритмов машинного обучения считается лишь только развивающимся направлением, в следствие этого международные стандарты для оценки качества работы такого рода приложений еще не выработаны. Однако, так как чат-бот приложения напрямую воспроизводят то, как могут работать реальные люди, то к чат-ботам применимы критерии, позволяющие контролировать работу сотрудников. Поэтому, для оценки эффективности чат-бот приложений, можно использовать KPI метрики (Key Performance Indicator), которые используются в бизнес-оценке как показатель деловой активности сотрудников организаций. И в то же время, чат-боты являются программным обеспечением, следовательно, для оценки качества также можно использовать стандартные методы оценки ПО. Таким образом, для оценки работы чат-ботов можно взять следующие метрики:

* Метрики для измерения бизнес-показателей:

1. Снижение загрузки колл-центра;
2. Конверсии в покупки;
3. Активация;
4. NPS(индекс потребительской лояльности).

* Метрики, оценивающие востребованность чат-бота:

1. Общее число вовлечённых пользователей;
2. Общее число активных пользователей;
3. Количество прочитанных сообщений.

* Метрики для оценки эффективности диалогов (технические показатели):

1. Срок жизни;
2. Общий процент ошибок;
3. Коэффициент правильности извлеченных данных;
4. Скорость реагирования бота на сообщение.

## **5.2 Сравнение бота DialogFlow и Tensorfow по выбранным метрикам**

Для сравнения рассматриваемых чат-ботов были взяты лишь «Метрики для оценки эффективности диалогов» из всех вышеперечисленных, в связи с тем, что на данном этапе разработка приложений не ставила перед собой задачу получение прибыли, а также интеграции в бизнес проекты. Кроме того, не были выбраны «Метрики, оценивающие востребованность чат-бота», поскольку работа внедрения аналитики в telegram-ботов не производилась.

После проведенного сравнения были сделаны следующие выводы по каждой метрике:

1. В чат-боте, с внедренным облачным сервисом распознавания естественного языка от Google, есть возможность настроить срок жизни, другими словами, можно указать количество высказываний пользователя, на протяжении которого контекст будет сохраняться. Однако, если поставить слишком большой срок жизни, то контекст не изменится в течение двадцати минут, что накладывает некоторые ограничения. В свою очередь, бот с Tensorflow сохраняет контекст до завершения запрограммированной ветки диалога. Таким образом, максимальный срок жизни контекста у бота BotExcel (реализован с помощью сервиса DialogFlow) равняется 20 минутам, у бота Excel (реализован с помощью библиотеки TensorFlow) этот показатель будем считать равным бесконечности, так как предела срока жизни контекста у бота не существует.
2. Процент ошибок у Tensorflow составил 33.3%, в то время как у DialogFlow не было выявлено в ходе тестирования ни одного не правильного ответа. Данный вывод был получен на результатах анализа сообщений ботов на пятнадцать одинаковых запросов. Пример одного из неверных ответов бота с Tensorflow представлен на рисунке 6.

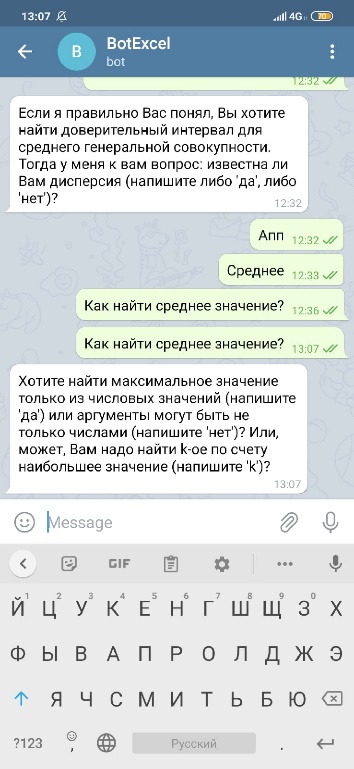


Рисунок 6 – Неправильное поведение «BotExcel»

1. Для оценки качества извлечения данных задача бота выделить ценные данные из пользовательских сообщений (например, имя пользователя, страну, город, дату, и т.п.). Из двух ботов такая возможность встроена лишь в DialogFlow. Он с точностью, равной 90% (согласно официальной информации компании-разработчика), записывает необходимые данные в указанные параметры для дальнейшего использования. Однако, для реализации поставленной задачи данной работы функционал извлечения данных не понадобился, следовательно, в заключительном выводе данный критерий будет пропущен.
2. Скорость реагирования ботов на сообщения пользователей сильно отличается, так на рисунке 7 можно увидеть, что бот с сервисом от Google отвечает мгновенно (средняя скорость получения ответа – 0,3 секунды), в свою очередь, сообщения от бота, скрипт которого размещен на Heroku, иногда приходится ждать значительно долго (см. Рис. 8). Среднее время отклика данного бота составляет 2 секунды.

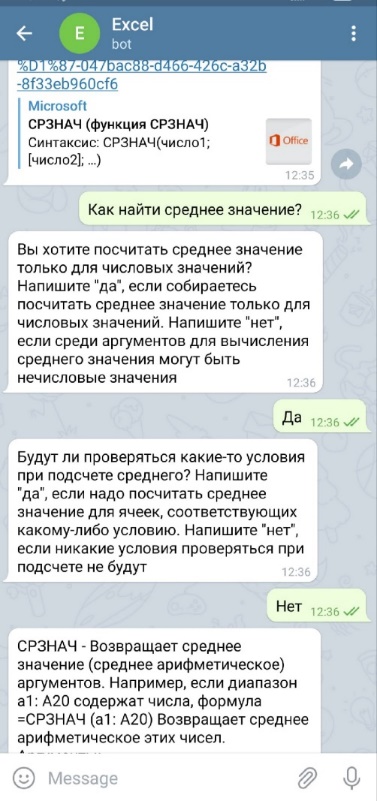


Рисунок 7 – Мгновенный ответ «Excel»

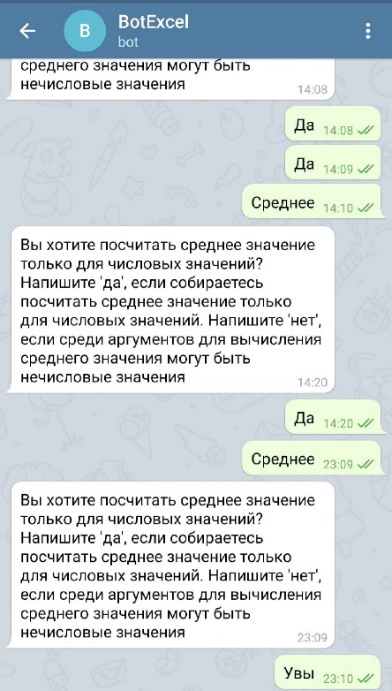


Рисунок 8 – Значительная задержка в «BotExcel»

Таким образом, используя Dialogflow при разработке чат-бота, можно свести неудовлетворенность пользователей к минимуму, так как согласно проведенному анализу работы обоих ботов, данный бот показал хорошие результаты по всем рассматриваемым метрикам.

# ***ВЫВОДЫ***

В результате курсовой работы было разработано два telegram-бота, подсказывающие пользователю статистические формулы Excel. Первый чат-бот был реализован с помощью сервиса DialogFlow, второй – с помощью библиотеки машинного обучения TensorFlow. В ходе оценивания качества обоих приложений были посчитаны основные характеристические метрики, на основании которых были сделаны следующие выводы. Чат-бот BotExcel (реализован с помощью сервиса DialogFlow) имеет преимущество над ботом Excel(реализован с помощью библиотеки TensorFlow) по быстроте реагирования на сообщение и точности нахождения правильного ответа. Однако, существенным недостатком этого решения является наличие лимита на количество принимаемых сообщений. Касательно бота Excel, никаких ограничений на количество запросов к сервису не накладывается. Проблему с долгим получением ответа можно решить путем развертывания модели бота на другом облачном сервисе, который имеет меньшую нагрузку.

Таким образом, если приложение разрабатывается для пользователей, которые уже имеют начальные знания в области статистики, и у них нет необходимости часто обращаться к сервису, то подходящим решением является использование сервиса DialogFlow. Так как предполагается, что такие пользователи будут задавать более сложные вопросы, на которые нужен точный ответ, который может предоставить бот BotExcel. Если же целевая аудитория бота — это люди, только начинающие свой путь в статистике, то им подойдет модель бота, созданная с помощью библиотеки TensorFlow и размещенная на облачной платформе с небольшой нагрузкой на сервера. Так как на начальном этапе знакомства со статистическими формулами Excel изучаются самые типовые функции, которые бот Excel может подсказать с высокой точностью и, что самое главное, может делать это неограниченное количество раз. И как уже говорилось ранее, для размещения бота нужна платформа с небольшой загрузкой, так как это позволит решить проблему с задержкой ответа.

# ***РАСПРЕДЕЛЕНИЕ ОБЯЗАННОСТЕЙ***

Терентьева Полина:

Создание модели бота ExcelBot, обучение модели, тестирование модели, развертывание модели на облачной платформе Heroku, подготовка отчета и презентации.

Дементьева Юлия:

Создание dataset, добавление обучающей выборки в сервис DialogFlow, обучение бота BotExcel на платформе DialogFlow, подготовка отчета и презентации.

# https://sun9-30.userapi.com/impg/qiUH7VXnTAvR6QGxzHxJnXHJhpa8iMV_ZswSnQ/Oqg4E_gPkmw.jpg?size=738x1600&quality=96&proxy=1&sign=450d138b1a8d76a4f282fbdf4a3741c9***ПРИЛОЖЕНИЕ А. Примеры запросов к ботам***

Рисунок 9 – Запрос об обычной корреляции и коэффициента корреляции Пирсона

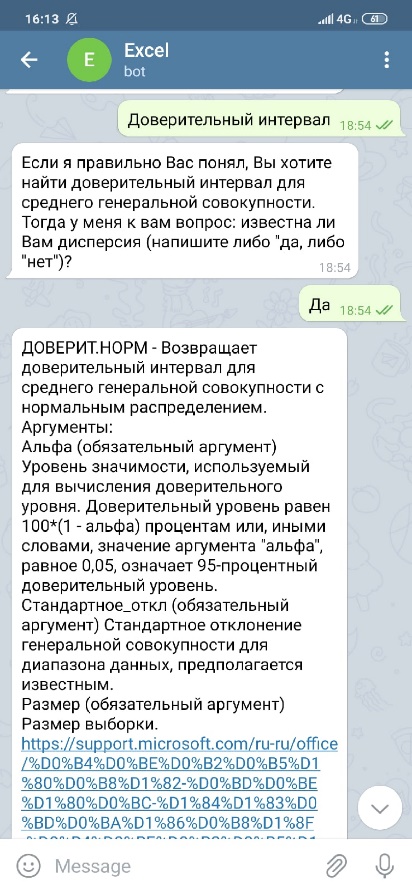


Рисунок 10 – Запрос о доверительном интервале

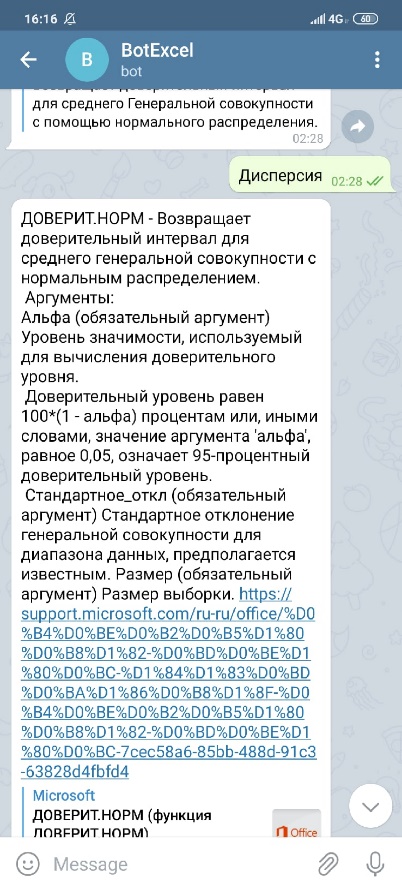


Рисунок 11 – Запрос о дисперсии

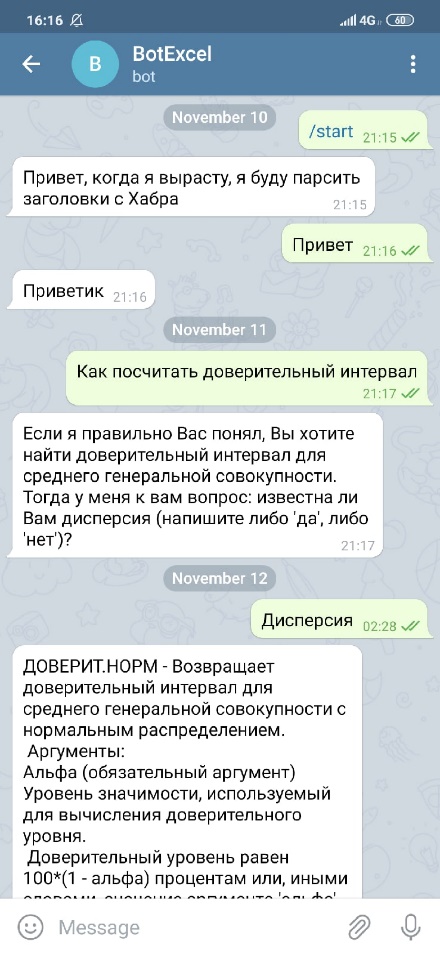


Рисунок 12 – Активация бота

# ***ПРИЛОЖЕНИЕ А. Скрипт для api***

import json

import pickle

import numpy as np

import keras

from keras.models import load\_model

import tensorflow as tf

from flask import Flask, request, abort, jsonify

import helper

flask\_app = Flask(\_\_name\_\_)

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import pickle

import numpy as np

from keras.models import load\_model

#path\_to\_model=tf.keras.utils.get\_file('chatbot\_model.h5', origin='https://drive.google.com/uc?id=1RJEfE3Cz4QmDg2epRq9dbW0QsWL0YMIW&export=download')

model = tf.keras.models.load\_model('chatbot\_model.h5')

import json

import random

#path\_to\_intents=tf.keras.utils.get\_file('intents.json', origin='https://drive.google.com/uc?id=1uo5RuJZRNMI87bZ8cSc7D5gIH4YGoqhA&export=download')

intents = json.loads(open('intents.json').read())

#path\_to\_words=tf.keras.utils.get\_file('words.pkl', origin='https://drive.google.com/uc?id=1YCk-M31yCEUf8gEtVwk\_jhn06Ub7EjfV&export=download')

words = pickle.load(open('words.pkl','rb'))

#path\_to\_class=tf.keras.utils.get\_file('classes.pkl', origin='https://drive.google.com/uc?id=1UXRdDwVka80Geo4Z6AKTOTzRt1RDgxQj&export=download')

classes = pickle.load(open('classes.pkl','rb'))

def clean\_up\_sentence(sentence):

# tokenize the pattern - splitting words into array

sentence\_words = nltk.word\_tokenize(sentence)

# stemming every word - reducing to base form

sentence\_words = [lemmatizer.lemmatize(word.lower()) for word in sentence\_words]

return sentence\_words

# return bag of words array: 0 or 1 for words that exist in sentence

def bag\_of\_words(sentence, words, show\_details=True):

# tokenizing patterns

sentence\_words = clean\_up\_sentence(sentence)

# bag of words - vocabulary matrix

bag = [0]\*len(words)

for s in sentence\_words:

for i,word in enumerate(words):

if word == s:

# assign 1 if current word is in the vocabulary position

bag[i] = 1

if show\_details:

print ("found in bag: %s" % word)

return(np.array(bag))

def predict\_class(sentence):

# filter below threshold predictions

p = bag\_of\_words(sentence, words,show\_details=False)

res = model.predict(np.array([p]))[0]

ERROR\_THRESHOLD = 0.25

results = [[i,r] for i,r in enumerate(res) if r>ERROR\_THRESHOLD]

# sorting strength probability

results.sort(key=lambda x: x[1], reverse=True)

return\_list = []

for r in results:

return\_list.append({"intent": classes[r[0]], "probability": str(r[1])})

return return\_list

def getResponse(ints, intents\_json):

tag = ints[0]['intent']

list\_of\_intents = intents\_json['intents']

for i in list\_of\_intents:

if(i['tag']== tag):

result = random.choice(i['responses'])

break

return result

def createAnswer(sentence):

return getResponse(predict\_class(sentence), intents)

import telebot

import parser

#main variables

TOKEN = "1424976571:AAGAYIBF5NJWq3b9otqIrGOcJzB2uh7NVRI"

bot = telebot.TeleBot(TOKEN)

@bot.message\_handler(commands=['start', 'go'])

def start\_handler(message):

bot.send\_message(message.chat.id, 'Привет, когда я вырасту, я буду парсить заголовки с Хабра')

@bot.message\_handler(content\_types=['text'])

def text\_handler(message):

text = message.text.lower()

chat\_id = message.chat.id

answer=createAnswer(text)

bot.send\_message(chat\_id, answer)

bot.polling(none\_stop = True)