**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Образовательная программа «Прикладная математика и информатика»

**Отчет о программном проекте**

на тему **«Голосовой помощник “Алина”»**

**Выполнила**:

студентка группы БПМИ184 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Андреева Д. А ф

Подпись И.О. Фамилия

Ф 26.05.2021 ф

Дата

**Принял**:

руководитель проекта Симагин Денис Андреевич \_

Имя, Отчество, Фамилия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_разработчик\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Должность

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ООО «Яндекс.Технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Место работы

Дата \_\_\_\_\_\_\_\_\_2020 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка (по 10-тибалльной шкале) Подпись

**Москва 2021**

Аннотация

В этой работе рассматривается алгоритм создания голосового ассистента, реагирующего на собственное имя и выполняющего некоторые команды. Итогом проекта будет RaspberryPi с предустановленной программой на языке Python по рассмотренному алгоритму.

[Аннотация 1](#_Toc73146201)

[Введение 2](#_Toc73146202)

[Основные определения и термины 3](#_Toc73146203)

[Цели и задачи 3](#_Toc73146204)

[Обзор существующих решений 4](#_Toc73146205)

[Требования к проекту 5](#_Toc73146206)

[Функциональные требования 5](#_Toc73146207)

[Нефункциональные требования 5](#_Toc73146208)

[Выбранные алгоритмы и методы 6](#_Toc73146209)

[Детектор ключевого слова (keyword-spotting) 6](#_Toc73146210)

[Реализация 7](#_Toc73146211)

[Модель 7](#_Toc73146212)

[Драфт проекта 8](#_Toc73146213)

[Выводы и дальнейшее развитие 9](#_Toc73146214)

[Список используемой литературы 10](#_Toc73146215)

# **Введение**

Сейчас, вместе с эволюцией компьютерных технологий, запросов пользователей и их интересов, когда стало возможно не только воспроизводить 8 битов, а обрабатывать невероятно сложные запросы: будь то обработка человеческой речи, её распознавание и реакция на неё.

История голосовых помощников началась с 1939 года, когда советский физик Лев Мясников создал аппарат, способный различать некоторые гласные и согласные звуки. Разработка продолжилась и после войны: сотрудники лаборатории Bell представили миру аппарат, способный разобрать номер телефона, надиктованный цифрами от 0 до 9, а компания IBM представила Shoebox, распознававший не только цифры, но и некоторые арифметические команды и слова на английском [1]. Уже в 1987 в продажу вышла кукла Джули, которая не только распознавала речь, но и отвечала на реплики собеседника (например, на просьбу спеть песню, кукла пела “Twinkle, Twinkle, Little Star”). В 2001 году компания Microsoft добавила голосовой ввод в программное обеспечение Office, а следом за ней в 2002 году компания Google добавила голосовой поиск в свою поисковую систему.

Первым голосовым помощником в традиционном понимании стала Siri от компании Apple, выпущенная в 2007 году. Она не только искала информацию по запросу в интернете или на девайсе пользователя, но и была способна вести достаточно осмысленный диалог. С 2014 по нынешнее время вышли также Cortana от Microsoft (2014г.), Alexa от Amazon (2014г.), AliGenie от Alibaba (2017г.), Алиса от Яндекс (2018г.), Маруся от Mail.ru и Олег от Тинькофф (2019г.).

Голосовые помощники развиваются как сами по себе, в смысле точности распознавания речи, так и их дополнительные функции: учатся рассказывать анекдоты, поддерживать беседу, генерируя ответы на основе сообщений собеседника. Эта функция изначально полезна как развлечение и привлечение пользователей, однако многие совершенно справедливо увидели в этом возможность настроить автоматическую техническую поддержку.

И это не предел возможностей голосового помощника. Одно из главных преимуществ данного функционала – потенциал развития. Умный дом на основе голосового помощника – далеко не предел, а потому это увлекательная и перспективная сфера исследования и разработки.

Основные определения и термины

***Нейронная сеть*** — математическая модель, а также её программная или аппаратная реализация, построенная по принципу, приближающему организацию и функционирование биологических нейронных сетей — сетей нервных клеток живого организма.

***Рекуррентная нейронная сеть*** (***Recurrent neural network* , *RNN***) — вид нейронных сетей, где связи между элементами образуют направленную последовательность – сети с циклами, позволяющие обрабатывать последовательности (текста, звука, etc.).

***Сеть с долговременной и кратковременной памятью (Long short term memory, LSTM)*** — разновидность RNN, отличающаяся способностью к обучению долгосрочным зависимостям.

***Распознавание речи (Speech-to-Text)*** — задача перевода голосового сообщения в текстовое представление.

***Keyword spotting*** — задача распознавания ключевого слова (*wake-up word*) в потоке звука.

***Преобразование Фурье*** — математическое преобразование, которое раскладывает функции, зависящие от пространства или времени, на функции, зависящие от пространственной или временной частоты, например, выражение музыкального аккорда в терминах громкости и частоты составляющих его нот.

Цели и задачи

1. Изучить существующие реализации голосового ассистента.
2. Ознакомиться с возможностями RaspberryPi.
3. Реализовать распознавание речи на потоке.
4. Объединить блок распознавания с блоком перевода звука в текст.
5. Реализовать ответную реакцию на запросы пользователя.

Обзор существующих решений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Голосовой помощник** | **Олег** | **Alexa** | **Алиса** | **Jasper** |
| **Компания-производитель** | Tinkoff | Amazon | Яндекс | Группа людей на GitHub (crm416, Holzhaus, shbhrsaha) |
| **Активационная фраза** | «Привет, Олег», «Слушай, Олег» | «Alexa» | «Алиса» | «Jasper» |
| **Позиционирование** | Лайф-стайл услуги | Музыкальная колонка | Музыкальная колонка |  |
| **Язык программирования** | Python | AWS Lambda | C++ | Python |
| **Поддержка русского языка** | Да | Нет | Да | Нет |
| **Поддержка ОС** | iOS, Android | Колонка + Android, как замена Google Assistant | iOS, Android, Windows (как часть Яндекс.Браузера) | RaspberryPi |
| **Стоимость станций** | ---- | 50-150$ | 160$ (9900₽) | ---- |
| **Исходный код** | Закрытый | Закрытый | Закрытый | Открытый |

По итогу, основные недостатки *для нас* – закрытый код, так как для учебных и исследовательских целей было бы полезно изучить чужие реализации, их недостатки и достоинства, некоторые идеи.

Другой существенный минус – отсутствие поддержки русского языка некоторых помощников, как с открытым кодом, так и с закрытым.

Требования к проекту

Функциональные требования

1. Интерфейс программы – RaspberryPi.
2. Работа с аудиозаписями .wav и аудио на потоке.
3. Распознавание wake-up word.
4. Дальнейшая работа с голосовыми запросами пользователя:
   1. Озвучка курса валют.
   2. Озвучка погоды.
5. Обработка дальнейших поступающих запросов:
   1. Повторная реакция на wake-up word;
   2. Повторная обработка запросов пользователя.

Нефункциональные требования

1. Требования к надежности

1.1. Программа не должна завершаться некорректно. В случае, когда определение слова не удалось, сообщать об этом пользователю голосом.

1. Условия эксплуатации

2.1 Условия эксплуатации ограничены условиями эксплуатации настольных ПК и RaspberryPi.

1. Требуемая квалификация и уровень подготовки пользователя

3.1 Пользователь должен уметь работать с RaspberryPi.

1. Требования к информационной и программной совместимости

4.1. Требования к языкам программирования

Программа должна быть написана на языке Python не ниже версии 3.4.

Выбранные алгоритмы и методы

Детектор ключевого слова (keyword-spotting)

*[здесь будут теоретические выкладки из использованных статей и их объяснение моими словами]*

Реализация

Модель

Для обучения модели использовался Colab Pro – платная, но более мощная версия Google Colab. Это было сделано по той причине, что видеокарта Google Colab гораздо мощнее, чем видеокарта на персональном компьютере. Кроме того, среда не требует предустановки библиотек.

Для распознавания активационного слова используется разбиение записей на слова при помощи построения спектрограммы. Используя энергию, равную среднему квадратов амплитуд *Дискретного преобразования Фурье*, вводим порог, по которому разбиваем слова (разбиение происходит в месте, где энергия меньше него). Кроме того, в некоторых местах осуществлялась ручная разметка слов.

Данные, использованные для обучения модели:

1. 3 минуты данных положительного класса **без** шумов (запись, содержащая только слово «Алина», сказанное различными интонациями и разными людьми с отличающимся тембром голоса и качеством записи).
2. 3 минуты данных положительного класса **с** шумами (запись, содержащая только слово «Алина», сказанное различными интонациями и разными людьми с отличающимся тембром голоса и качеством записи).
3. 5 минут данных негативного класса, однако созвучных со словом «Алина» (например, «Клин», «Блин», «Калина», «Малина», «Аргентина» etc) **без** шумов и помех.
4. 5 минут данных негативного класса, однако созвучных со словом «Алина» **с** шумами и помехами.
5. 20 минут неопределенного шума (беседа с родными, шум телевизора, чтение рецептов и каталогов).

После предобработки записей, выборка была разделена в соотношении 1:9 на валидационную и обучающую.

Обучение происходило следующим образом: слова смешивались в случайном порядке и объединялись в группы по пять записей. Данное действие обеспечивает стабильность скрытого состояния при длительном применении сети. Данные наборы далее подавались на вход модели. При определения слова, как представителя отрицательного класса, от модели требовалось выдавать «0» на протяжении всего слова, в то время как для слов положительного класса требовалось определить «1» хотя бы в конце слова (это кажется очевидным: до того, как слово закончилось, мы не может сказать, то ли это слово или не то). В середине слова выход сети нам не важен, будет ли это «0» или «1» (ошибка не менялась, в зависимости от выхода сети).

Модель обучалась подобным образом на протяжении 23 эпох.

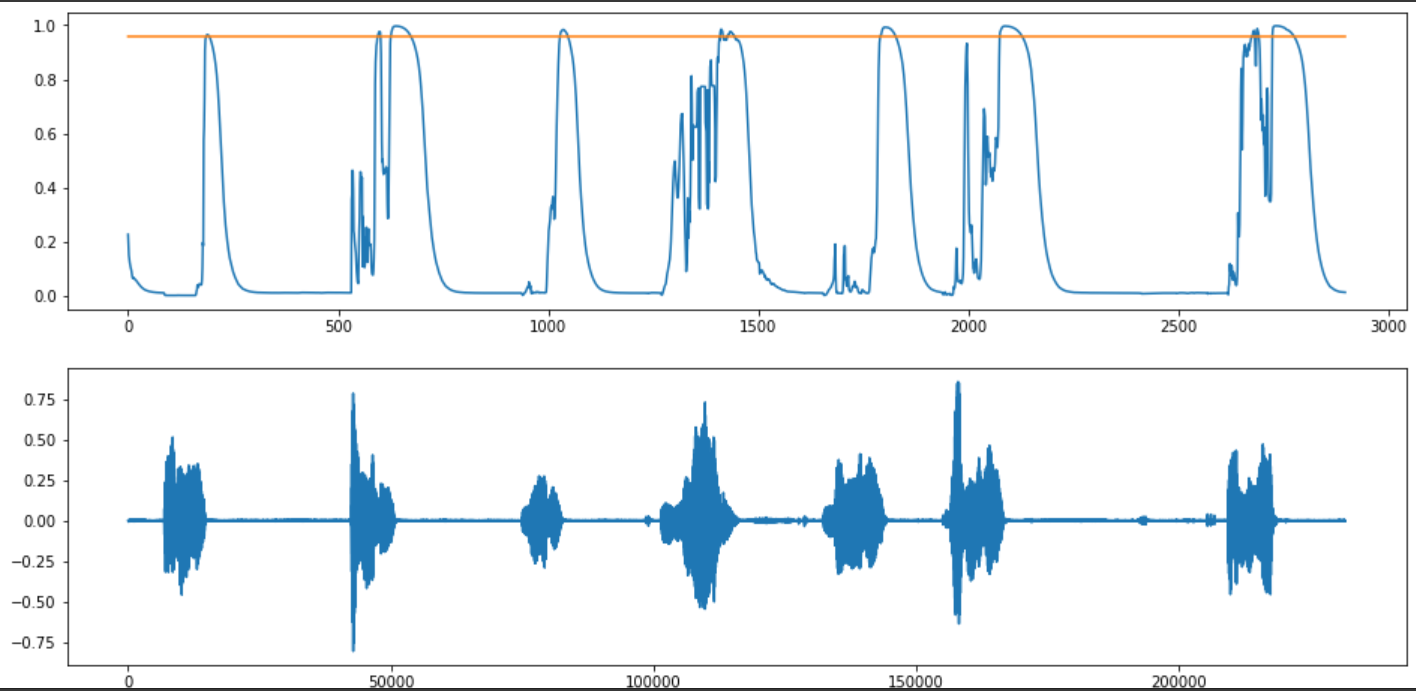
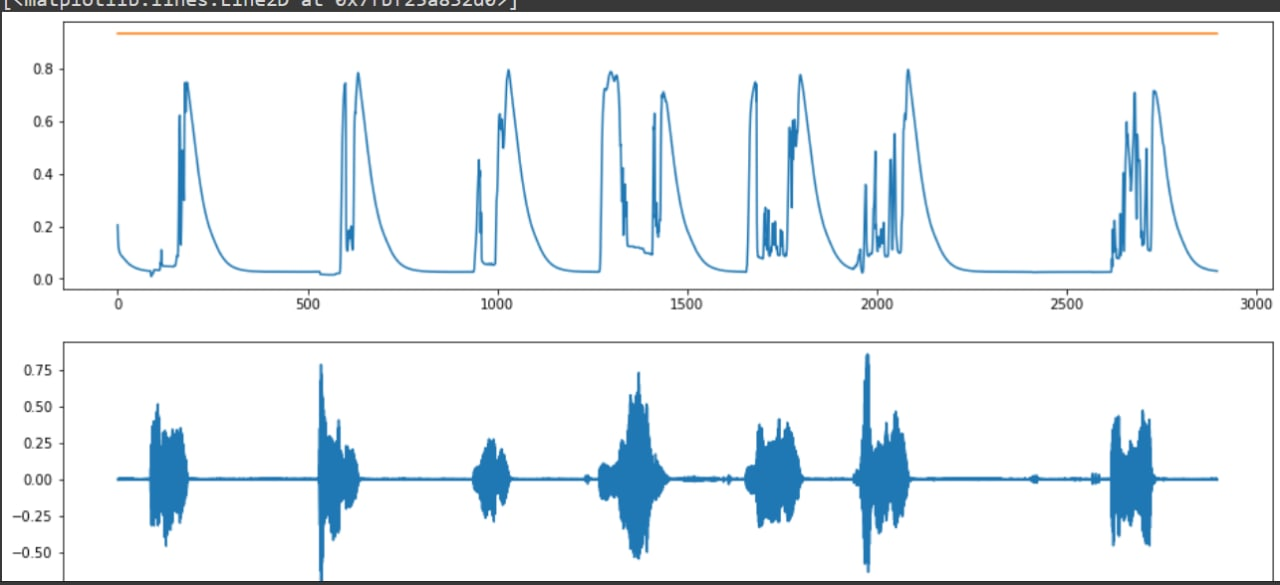


Рисунок 1. Распознавание "Алин"

Первое, с чем мы столкнулись, устанавливая порог для вероятностей, выше которого мы считаем, что слышали своё имя: модель либо распознаёт всё, что содержит звук «и» -- «блин», «клин», просто протяжное «иииии», либо не распознает вообще ничего (в зависимости от порога). Усугубляло это ещё и то, что в зависимости от качества записи микрофона, менялась точность распознавания слов. На рисунке выше только первые два слова являются «Алиной», в то время как остальные – просто достаточно созвучные слова. Причиной этого стало отсутствие негативов, созвучных с активационным словом.

Другая проблема, с которой мы столкнулись – плохое соотношение классов в выборке. Описанный набор записей приводил к соотношению классов 1:4, что уже является несбалансированной выборкой. В дальнейшем это решилось аугментацией записей, выравниванием классов с помощью увеличения весов меньшего (позитивного) класса, дозаписью аудиозаписей, а также встроенных методов SMOTE.

*[здесь дописать про страдания во время обучения модели]*

*[здесь прописать про страдания во время прикрутки распознавания речи]*

Для запуска проекта с предобученной моделью используется файл .py загруженный на RaspberryPi, запускающийся при включении оборудования. В этом файле реализована функция непрерывного считывания аудио с микрофона, распознавание wake-up word, реакция помощника на команды пользователя.

Драфт проекта

<https://github.com/xufana> - основной репозиторий проекта на GitHub.

Выводы и дальнейшее развитие

Итогом проекта стал голосовой помощник с открытым кодом, относительной дешевизной (цена RasberryPi около 3000 рублей), некоторым списком поддерживаемых команд и полной поддержкой русского языка (однако, с отсутствием поддержки английского языка).

В первую очередь, по итогам проекта был сделан главный вывод о предобработке тренировочных данных. Основной успех модели строился именно на них, а потому достаточно продолжительное время модель показывала не самые впечатляющие результаты. После тщательной работы с тестовыми и валидационными данными точность значительно улучшилась.

Также, по данным некоторых экспериментов было установлено, что обычная RNN модель работает хуже, чем GRU*. [дописать сюда ещё выкладок по модели]*

Кроме того, очевидно, что проекту есть, куда совершенствоваться: написать собственное распознавание речи, прописать больше команд для помощника и улучшить точность реакции на активационное слово.

Другое очевидное улучшение: научить помощника разговаривать. Добавить модель, реагирующую на сказанное пользователем, и способную разговор поддержать, например, в виде AI-собеседника. Ещё один вариант: реализация умного дома, которая также выглядит заманчивым и интересным улучшением существующего фундамента голосового помощника.

.

Список используемой литературы

1. **«1961 Shoebox IBM Archives»** <https://youtu.be/rQco1sa9AwU>