Министерство науки и высшего образования Российской Федерации Санкт-Петербургский политехнический университет Петра Великого Институт компьютерных наук и технологий

Высшая школа программной инженерии

|  |  |
| --- | --- |
|  | Работа допущена к защите |
|  | Директор ВШПИ |
|  | П.Д.Дробинцев |
|  | « » 2023 г. |
| **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА** | |
| **Работа бакалавра** | |
| **ПРИМЕНЕНИЕ НЕЙТРОННЫХ СЕТЕЙ ДЛЯ ОЦЕНКИ ТРАФИКА МАГАЗИНА** | |
| по направлению подготовки (специальности) | |
| 09.03.04 Программная инженерия | |
| Направленность (профиль) |  |
| 09.03.04\_01 Технология разработки и сопровождения качественного программного продукта | |
| Выполнила студентка гр. | Ли Ицзя |
| 3540904/90102 |  |
| Руководитель | О.Г.Малеев |
| к.т.н., доцент ВШПИ ИКНТ |  |
| Консультант | Е.Г.Локшина |
| по нормоконтролю |  |
| Санкт-Петербург | |
|  | 2023 |

**САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО**

**Институт компьютерных наук и технологий**

**Высшая школа программной инженерии**

УТВЕРЖДАЮ

Директор высшей школы

П.Д. Дробинцев

«06» мая 2023 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы

студенту Ли Ицзя, группа 3530904/90102

1. Тема работы: Применение нейронных сетей для оценки трафика магазина.

2. Срок сдачи студентом законченной работы: 22.05.2023

3. Исходные данные по работе: Документация PyTorch,

4. Содержание работы (перечень подлежащих разработке вопросов):

1) Обзор литературы

2) Обучение модели для отследования людей в нескольких видеопотоках

3) Система оценки трафика магазина основа на нейтронных сетей

4) Анализ полученных результатов

5. Перечень графического материала (с указанием обязательных чертежей): отсутствует

-

6. Консультанты по работе:

-

7. Дата выдачи задания: 06.05.2023

|  |  |  |
| --- | --- | --- |
| Руководитель ВКР |  | О. Г. Малеев |
|  | (подпись) |  |

文本, 信件

描述已自动生成Задание принял к исполнению 06.05.2023

|  |  |  |
| --- | --- | --- |
| Студентка |  | Ли Ицзя |
|  | (подпись) |  |

## Реферат

### На xx с., xx рисунков, x таблицы

КЛЮЧЕВЫЕ СЛОВА: РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ, МАШИННОЕ ОБУЧЕНИЕ, РАСПОЗНАВАНИЕ РЕЧИ, ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА, ГОЛОСОВОЕ УПРАВЛЕНИЕ

Выпускная квалификационная работа на тему: «Применение нейронных сетей для оценки трафика магазина».

Данная статья посвящена реализации алгоритмов нейронных сетей для оценки трафика в магазине.

Задачи, решаемые в ВКР:

1. Для обнаружения пешеходов был использован алгоритм YOLOv5

2. Для установления связи между персонажами при работе с несколькими камерами был реализован алгоритм повторного распознавания (ReID).

3. Для отслеживания локальных зон был реализован алгоритм DeepSORT он основе повторного распознавания.

4. На основе методики, описанных в данной статье, была разработана система подсчета посетителей для супермаркета.

Для решения поставленных задач были использованы методы глубокого обучения и компьютерного зрения. В результате проведенных исследований была разработана система подсчета посетителей для супермаркета, которая демонстрирует точность \*\*\* и производительность.

#### Abstract

At 69 p., 58 figures, 2 tables

KEYWORDS: RECURRENT NEURAL NETWORKS, MACHINE LEARNING, SPEECH RECOGNITION, NATURAL LANGUAGE PROCESSING, VOICE CONTROL

Graduate qualification work on the topic: "Using algorithms for transcribing and accentuation of Russian text in a voice control system".

This work is concerned with the creation of a more accurate phonetic transcription, using for this purpose transcribing and accentuation algorithms, as well as a developed neural network. Creating an acoustic model based on this transcription and applying it in an application with implemented voice control, namely a Telegram bot that receives and stores links to Yandex.Maps with a specific address or a constructed route. Tasks to be solved in the EKR: Research of the subject area and open-source speech recognition libraries. Study Existing methods to improve the accentuation algorithm. Develop a neural network and train it to improve the accentuation algorithm. Obtain a transcription and train an acoustic model based on it. Create a test data collection algorithm and collect test data by topic. Conduct a Comparison of the acoustic model with the competitors. Develop a Telegram bot in java using modern libraries. Apply the acoustic model and transcriptions for voice control.

## Оглавление

[ВВЕДЕНИЕ 8](#_bookmark0)

[ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ОТКРЫТЫХ БИБЛИОТЕК](#_bookmark1) [РАСПОЗНОВАНИЯ РЕЧИ 11](#_bookmark1)

* 1. [Структура речи 11](#_bookmark2)
  2. [Распознавание речи 12](#_bookmark3)
  3. [Обзор Открытых библиотек распознавания речи 13](#_bookmark4)
     1. [HTK 13](#_bookmark5)
     2. [Kaldi 13](#_bookmark6)
     3. [Julius 14](#_bookmark7)
     4. [CMU Sphinx 14](#_bookmark8)
  4. [Архитектура Sphinx4 15](#_bookmark9)
  5. [Алгоритмы СMU Sphinx 17](#_bookmark10)
     1. [Получение МЧКК 17](#_bookmark11)
     2. [Скрытые Марковские Модели 21](#_bookmark12)
     3. [Алгоритм Витерби 23](#_bookmark13)
  6. [Исследования по транскрибированию и акцентуации 23](#_bookmark14)

[ГЛАВА 2. ПРИМЕНЕНИЕ АЛГОРИТМОВ ТРАНСКРИБИРОВАНИЯ И](#_bookmark15) [АКЦЕНТУАЦИИ 25](#_bookmark15)

* 1. [Транскрибирование 25](#_bookmark16)
     1. [Алгоритм транскрибирования 25](#_bookmark17)
  2. [Акцентуация 27](#_bookmark18)
     1. [Общие подходы к Нейронным сетям 28](#_bookmark19)
     2. [Рекуррентные нейронные сети 29](#_bookmark20)
     3. [LSTM 29](#_bookmark21)
     4. [GRU 32](#_bookmark22)
     5. [CBOW 33](#_bookmark23)
     6. [Разработанная нейронная сеть 36](#_bookmark24)

[2.2.6 Алгоритм создания словаря 38](#_bookmark25)

* 1. [Архитектура приложения 38](#_bookmark26)
  2. [Инструментарий и технологии разработки приложения 43](#_bookmark27)
     1. [Среда разработки 43](#_bookmark28)
     2. [система автоматизации сборки 44](#_bookmark29)
     3. [Фреймворк 45](#_bookmark30)
     4. [FFmpeg 49](#_bookmark31)
     5. [Telegram 50](#_bookmark32)
     6. [Selenium 52](#_bookmark33)
     7. [Sphinx4 53](#_bookmark34)
  3. [Создание тестовых данных для сравнения акустических систем 55](#_bookmark35)

[ГЛАВА 3. Анализ работы и результаты 59](#_bookmark36)

* 1. [Тестирование и сравнение с конкурентами 59](#_bookmark37)
  2. [Сравнение Акустических моделей и словарей на тестовой выборке. 59](#_bookmark38)
  3. [Результаты работы Telegram бота 61](#_bookmark39)

[Заключение 67](#_bookmark40)

[Список использованных источников 69](#_bookmark41)

#### ВВЕДЕНИЕ

В современном мире, с развитием технологического прогресса, набирают популярность различные приложения, имеющие голосовой интерфейс. Данные приложения могут иметь абсолютно различную направленность от мобильного приложения магазина доставки до использования внутри видеоигр. Широко распространены в системах “Умный дом” для решения различных задач и повышения комфорта проживания. Показателем популярности данной сферы может служить огромное количество различных НИИ и коммерческих компаний работающих над созданиями решений в данной области.

Встраиваемые системы распознавания речи имеют два основных подхода распространённых в коммерческой среде: первый это воспользоваться мощным сервисом от большой корпорации что отличается относительно высокой точностью распознавания и скорости обработки речи, но всегда имеет модель коммерческого распространения, а также может получать и записывать информацию, и имеет закрытый исходный код, что не позволяет использовать его в открытых проектах, и что повышает стоимость конечного коммерческого продукта; второй подход это создание своей системы распознавания речи на основе открытых библиотек, которые позволяют создавать свои узконаправленные решения.

У данных решений страдает универсальная точность распознавания речи. Поэтому возникает необходимость использовать дополнительные методы для улучшения распознавания речи и эти решения не всегда могут быть универсальными так как языки имеют свои уникальные особенности.

Одним из способов улучшения качества распознавания речи это улучшение качества фонетической транскрипции. И тут встает вопрос задачи транскрибирования – перевода обычной формы слова в фонетическую транскрипцию и чем лучше сделана транскрипция, тем лучше

8

система распознавания речи сможет распознавать слова. Транскрипция слова используется для обучения и создания акустической модели.

Особенностью русского языка является переменчивое ударение в словах и для того, чтобы провести качественное транскрибирование нужно учитывать этот фактор, что приводит нас к второй задачи акцентуации – расстановки ударений в словах. Одним из способов акцентуации это определения ударения слова по его морфологическим признакам, с чем прекрасно справляются нейронные сети.

Применение улучшенных словарей и акустической модели в прикладном приложении повышает удобство и качество использования подобных приложений. Например, применение систем распознавания речи для внедрения их различных видов ботов для различных мессенджеров таких как What’s Up, Viber, VK, Telegram. Telegram. Telegram является предпочтительным так показывает самый высокий рост пользователей и не имеет такую сильную привязку к определенным регионам какой есть например у Viber и What’s Up. Еще одним плюсом в сторону выбора именно Telegram ее удобное апи, что отражается на количестве ботов, разрабатываемых на данной платформе и их различных вариациях с абсолютно разным функционалом.

**Целью данной работы является** создание Telegram бота для получения маршрутов на картах Яндекс с включенным в процесс голосовым управлением на словаре и акустической модели, сформированной с помощью совмещенных методов транскрибирования и акцентуации.

Включение самого в транскрибирования в процесс работы приложения не считается целесообразным, потому что процесс создания и тренировки акустической модели является долгим процессом и не может происходить динамически.

Актуальность темы является высокой так как включает в себя применение и обучение нейронных сетей, изучение интеграции современных систем как Telegram, Selenium, углубление в проблематику

распознавания речи

Задачи выпускной квалификационной работы, которые нужно решить для реализации цели работы:

* + 1. Исследование предметной области и открытых библиотек распознавания речи.
    2. Изучение Существующих методов улучшения алгоритма акцентуации.
    3. Разработать нейронную сеть и провести ее обучение улучшив алгоритм акцентуации.
    4. Получить транскрипцию и обучить на ее основе акустическую модель.
    5. Провести Сравнение акустической модели с конкурентами.
    6. Разработать Telegram бота на java c применением современных библиотек.
    7. Применить акустическую модель и транскрипции для голосового управления.

# ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ОТКРЫТЫХ БИБЛИОТЕК РАСПОЗНОВАНИЯ РЕЧИ

### Структура речи

В распознавании речи под речью понимают поток аудио данных, поступающий в переменных состояниях, смена которых происходит регулярно. Состояния, обладающие наиболее схожими параметрами, объединяются в группы, называемые фонемами. Характеристики звуковой волны, связанные с определенной фонемой, зависят от многих условий таких как язык, особенности речи говорящего и т.д.

Фонемы формируют части слов называемых слогами. Ускоряя и замедляя речь такую группу фонем как слоги не изменить, что повышает их стойкость к такому эффекту как редукция. Слоги своими многочисленными вариациями формируют слова, а слова и различные шумы, произносимые человеком, формируют сообщения – отрезки аудиозаписи, несущие речевую информацию между тишиной.

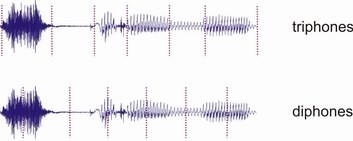
Довольно часто встречается рассмотрение фонем в контексте соседних фонем и их влияние на звучание конкретной фонемы и подобные фонемы называют трифоны. Фрейм звуковой волны трифона соответствует звуковой волне фонемы.

Для повышения такого показателя как быстродействие есть способ дробления трифонов на маленькие составные части, которые могут встречаться в и других трифонах что дает возможность пере использования. Возможность записать подобные частицы звука используют открытые библиотеки так как с помощью подобного набора можно получить высокий

процент

Другим подходом является рассмотрение стыков двух трифонов. Стык активных изменений между двумя фонемами может являться более полным

информации. Стыки выделяют отдельно и называют дифоны.



*Рис. 1.1 Фреймы различных типов*

#### Распознавание речи

Традиционно для распознавания речи используются отрезки тишины. Поток аудио данных нарезается по этим отрезкам. Разные системы могут использовать различные настройки минимальной громкости, которые выставляют в соответствии с используемыми данными, а также определяется время участков не превышающий этот минимум.

Оставшийся аудио ряд воспринимается системами как несущий речевую информацию и проходит процесс соотнесения с возможными комбинациями слов и выбирается конкретная комбинация в зависимости от алгоритма распознавания. Например, если использовать для распознавания шаблонов скрытую марковскую модель, то искомой комбинацией слов будет самая вероятная.

Для сравнения выделенный аудиосигнал разделяется на сегменты. Стандартным размером сегмента является период 10 миллисекунд. Для каждого сегмента происходит расчет вектора признаков. Обычно вектор признаков вычисляется из 13 мел-частотных кепстральных коэффициентов (МЧКК). После вычисления векторов в рамках 3 основных моделей, соответствующих структуре речи. Модели будут описаны в следующих разделах.

#### Обзор Открытых библиотек распознавания речи

В рамках данной работы рассматривается 4 известные библиотеки распознавания речи с открытым исходным кодом и похожим набором используемых в данных библиотеках алгоритмов.

#### HTK

HTK (Hidden Markov Model Toolkit) является инструментом для создания и модификации скрытых марковских моделей. Библиотека не ограниченна областью распознавания речи и может в других сферах, где скрытые марковские модели могут быть применены, например распознавания цепочек ДНК. Основной областью применения библиотеки является научная деятельность. Языком реализации является C. Основное применение библиотеки в качестве части автоматической системы распознавания речи. Хотя библиотека является открытой, коммерческое использование библиотеки запрещено лицензией, что сильно повлияло на популярность данной библиотеки.

#### Kaldi

Kaldi имеет самый широкий выбор используемых алгоритмов. Для извлечения вектора признаков сегмента, несущего речевую информацию, могут выступать как МЧКК, так и перцептивный (воспринимаемый) линейный прогноз (PLP). В акустической модели используются скрытые марковские модели, модели смеси гауссовских распределений (GMM) и нейронная сеть с временными задержками (TDNN) создание языковой модели происходить с использованием конечного автомата преобразователя (FST). Распознавание речи происходит с помощью алгоритма двухпроходной алгоритм прямого-обратного хода. Реализована на языке C++. Имеет реализацию на Python и применяется как в коммерческой, так и в научной деятельности. Имеет более высокий порог вхождения и рассчитан на

опытного в распознавании специалиста.

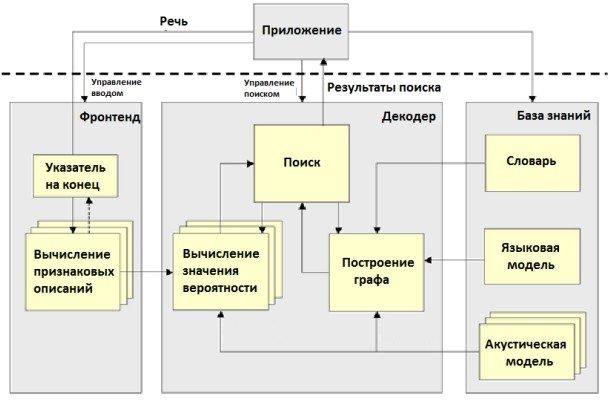
#### Julius

Библиотека Julius не имеет своих инструментов для создания акустической и языковой модели, что решает с помощью сторонних решений таких как модули HTK. Реализация представлена на языке С. Изначально разрабатывался для японского языка, но может применяться и к другим языкам. Имеет минимальное среди этих 4 библиотек потребление памяти, что позволяет использовать на более слабых машинах.

#### CMU Sphinx

CMU Sphinx – семейство библиотек распознавания речи, но рассмотрена в рамках данной работы будет только последняя 4 библиотека. Имеет две реализации на С и на Java. Pocketsphinx и Sphinx4 соответственно. Отдельно используются модули для создания Акустической и Языковой модели. Вектор признаков также вычисляется из МЧКК. И имеет вычисляемый размер для CMU Sphinx 39 признаков. Первые 13 — это сами коэффициенты, вторые 13 — это первые производные от коэффициентов. Третьи это вторая производная. Для моделирования языковой модели используются n-граммные модели. В основном это триграмма, и она рассчитывает вероятность последнего слова по 2 предыдущим словам. Для распознавания речи используется алгоритм Витерби. Имеет самое широкое количество готовых моделей для множества языков мира. Также как и Kaldi область применения является научная деятельность, но и прикладное коммерческое использование. Выбор данной библиотеки обусловлен более простой интеграцией с Java, и более выраженных результатов от применения алгоритмов транскрибирования и акцентуации.

* 1. **Архитектура Sphinx4**



*Рис. 1.2 Схема Sphinx4*

Sphinx4 имеет модульную структуру и управляется внешним приложением (на рис. 1.6 представлено схема CMUSphinx). Основные модули:

* Фронтенд (интерфейс библиотеки)
* Декодер
* 3 модели структуры речи:

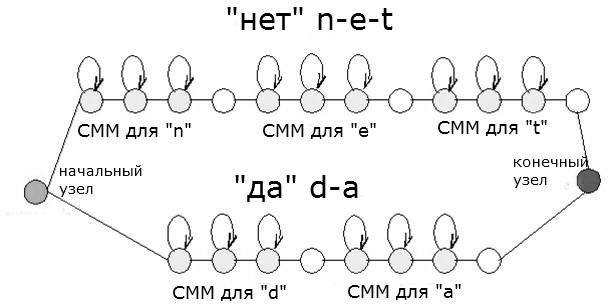
1. Акустическая модель - содержит список сенонов полученных после обучения модели и их параметров
2. Языковая модель – построенная в виде n-граммной модели определяет вероятности следующего слова на основании предыдущих. Является барьером для случайных и маловероятных слов.
3. Набор искомых слов – хранит пару слово и его фонетическая транскрипция. Является основным изменением внесенными алгоритмами. Так как при повышении точности словаря повышается точность

получаемой акустической модели, что в целом повышает

точность системы распознавания речи. Чем больше размер словаря, тем хуже система распознает речь, так что основное применение библиотеки в узкоспециализированных приложениях с ограниченным списком распознаванием слов. С помощью специальных файлов грамматики удается ограничить словарь до нужных в приложении слов и не испортить изначальный словарь, использованный для тренировки акустической модели.

Алгоритм работы библиотеки:

1. При старте работы библиотеки лингвист, находящийся в декодере, строит скрытые марковские модели на основании языковой модели, которую он до этого преобразовал в граф, акустической модели и набора искомых слов.
2. Интерфейс принимает аудиосигнал и очищает от тишины и различных шумов в зависимости от настроек. Традиционно это до 10 Дб.
3. Потом вычисляет параметры вектора каждого сегмента речи.
4. Передает вектора в декодер
5. Акустический счетчик [1] вычисляет из векторов вероятности состояний для каждого несущего речевую информацию сегмента.
6. Вычисленные вероятности передаются в вычисленный граф и возвращаются для пересчета следующей самой вероятной фонемы. (на рис. 1.3 представлен граф поиска слов из набора искомых слов)



*Рис. 1.3 Найденный граф для слов да и нет*

## Алгоритмы СMU Sphinx

#### Получение МЧКК

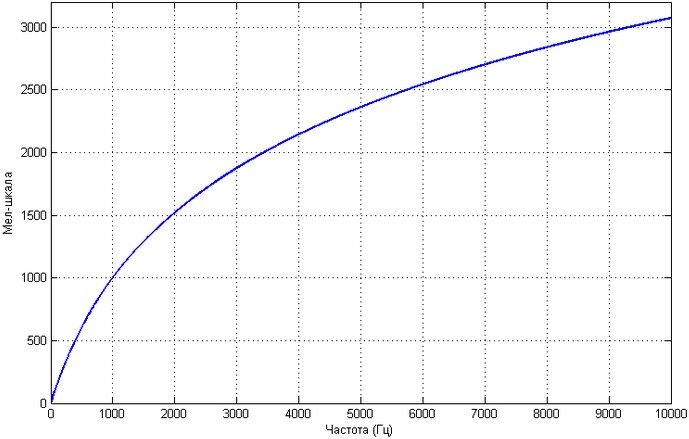
Мел – это не очень точная величина высоты звука, основанная на человеческом восприятии звука. Выражается с помощью формулы (на рис.

* 1. представлен график зависимости мела от частоты)

m = 1125ln (1 + 𝑓𝑓

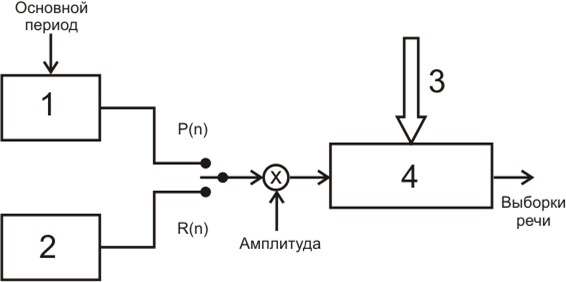
) *(1.1)*

700



*Рис. 1.4 Зависимость мела от частоты*

Если издаваемую человеком речь представить в виде Генераторов и фильтров сигнала, то можно формализовать в виде схемы (на рис. 1.9 представлена схема условного человеческого речевого аппарата) :



*Рис. 1.5 Схема формализации человеческой речи*

* + - Два генератора: первый тонов, второй шумов
    - Цифровой фильтр и его параметры

Выходной сигнал можно представить после конволюции как:

𝑓𝑓(𝑡) = 𝑠(𝑡) ⊗ ℎ(𝑡) *(1.2)*

s(t) – изначальный вид акустической волны

h(t) - характеристика фильтра Переведя формулу к частотам, получим:

𝐹(𝜔) = 𝑆(𝜔)𝐻(𝜔) *(1.3)*

Получаем сумму прологарифмировав выражение:

ln[𝑆2(𝜔) ∗ 𝐻2(𝜔)] = 𝑙𝑛𝑆2 (𝜔) + 𝑙𝑛𝐻2(𝜔) *(1.4)*

Выполнив обратное преобразование Фурье, мы получим кепстр[231]:

С(𝑞) = 1

2𝜋

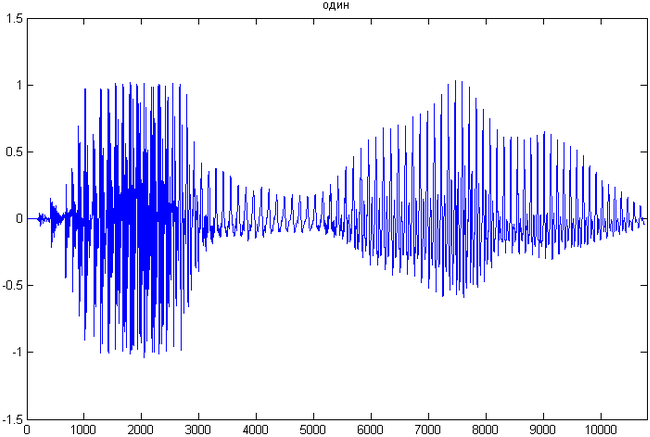
∞

∫−∞

ln [𝐹(𝜔)]2 𝑒𝑖𝑖𝜔𝑞𝑑𝜔 *(1.5)*

Далее рассмотрим получение МЧКК (на рис 1.10 представлена акустическая волна характерная слову один от определенного человека):

* + - Исходный сигнал: 𝑥𝑥[𝑛], 0 ≤ 𝑛 < 𝑁𝑁



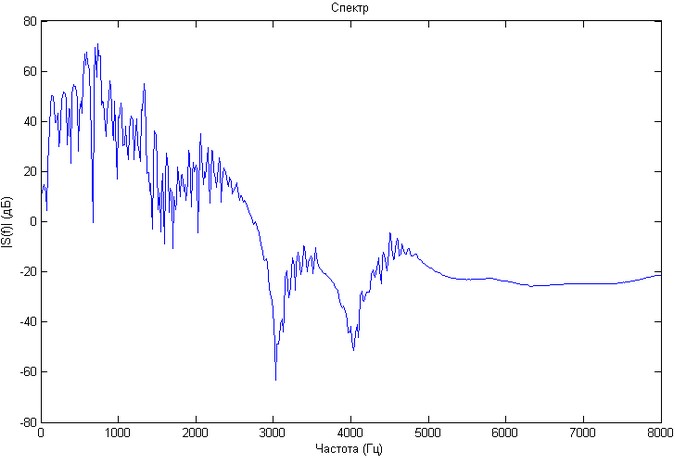
*Рис. 1.6 Пример исходного речевого сигнала*

* + - Использование преобразование Фурье для получения спектра (на рис 1.11 представлен график спектра исходного сигнала характерного для слова один):

2𝜋𝜋𝜋

𝑛=0

𝑋𝑋𝑎[𝑘] = ∑𝑁−1 𝑥[𝑛]𝑒− 𝑁 𝑘𝑛 , 0 ≤ 𝑘 < 𝑁 *(1.6)*



*Рис. 1.7 Спектр исходного сигнала*

* + - Расчет частот для оконной функции

𝑓𝑓[𝑚] = �𝑁� 𝐵−1(𝐵(𝑓𝑓 ) + 𝑚(𝐵(𝑓𝑓 ) − 𝐵(𝑓𝑓 )) *(1.7)*

𝐹 1 ℎ 1

* + - 𝐵(𝑏) − преобразование значения частоты в мел − шкалу

𝑏

𝐵−1(𝑏) = 700(𝑒1125 − 1) *(1.8)*

* + - Построение фильтров, используя оконную функцию (на рис.

1.12 и 1.13 представлены графики оконных фильтров на первом в единицах мел-шкалы, а во втором в единицах частоты)

0, 𝑘 < 𝑓𝑓[𝑚 − 1]

𝐻𝑚=

𝑓𝑓(𝑥) =

𝑘−𝑓𝑓[𝑚−1]

𝑓𝑓[𝑚]−𝑓𝑓[𝑚−1]

𝑓𝑓[𝑚+1]−𝑘

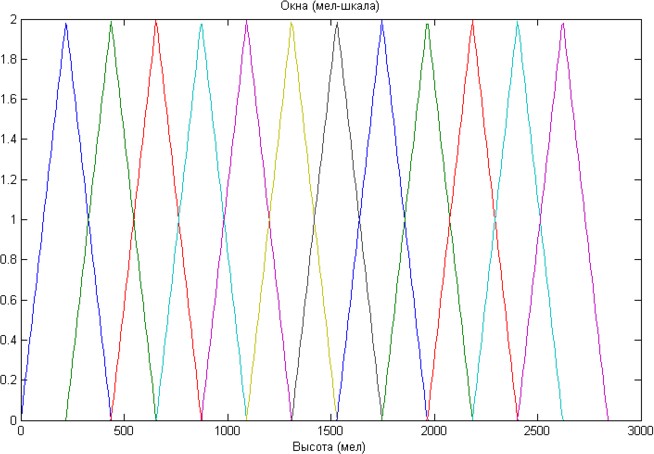
𝑓𝑓[𝑚+1]−𝑓𝑓[𝑚]

, 𝑓𝑓[𝑚 − 1] ≤ 𝑘 < 𝑓𝑓[𝑚]

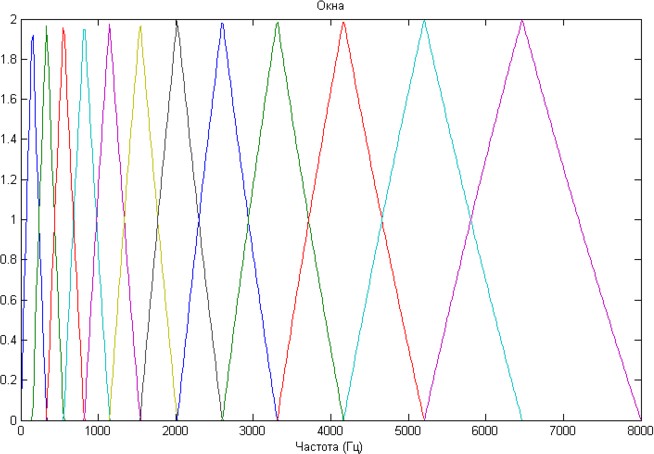
, 𝑓𝑓[𝑚] ≤ 𝑘 < 𝑓𝑓[𝑚 + 1]

*(1.9)*

𝗅 0, 𝑘 > 𝑓𝑓[𝑚 + 1]



*Рис. 1.8 Равномерно размещенные окна для расположения спектра на мел оси*



*Рис. 1.9 Частотный график окон*

* + - Расчет энергии окна:

𝑆[𝑚] = ln(∑𝑁−1 |𝑋𝑋𝑎[𝑘]|2 𝐻𝑚[𝑘]) , 0 ≤ 𝑚 < 𝑀 *(1.10)*

𝑚=0

* + - Применение дискретного косинусного преобразования (на рис.

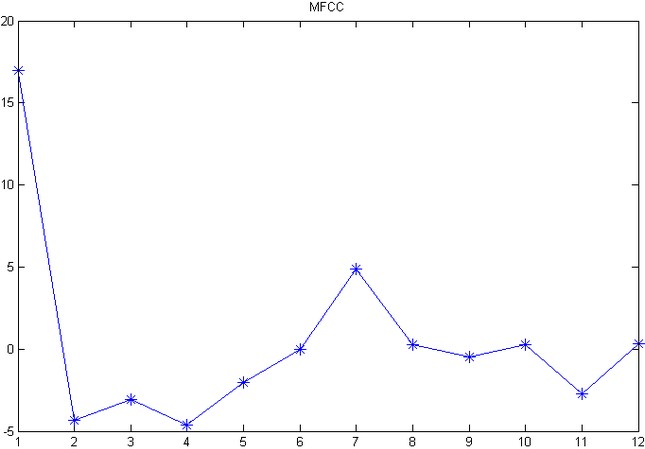
1.14 представлен график полученных МЧКК для слова один):

1

𝜋𝑛�𝑚+ �

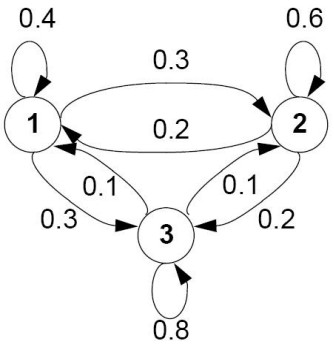
с[𝑛] = ∑𝑀−1 𝑆[𝑚]cos ( 2 ) , 0 ≤ 𝑛 < 𝑀 *(1.11)*

𝑚=0 𝑀



*Рис. 1.10 Пример полученных после расчёта МЧКК для исходного сигнала.*

#### Скрытые Марковские Модели



*Рис. 1.11 Визуальный пример СММ*

Скрытые Марковские Модели (СММ) – это широко применяемый статистический инструмент, например используемый в распознавании речи и конкретно в библиотеке CMU Sphinx. Представляет вычисленные МЧКК параметрами случайного процесса. СММ может представлять любой кусок распознаваемой речевой информации как последовательность наступающих событий. СММ предложения включает в себя все СММ самых мельчайших используемых в распознавании объектов речи, например, как часть трифона.

Если формулировать какая задача у распознавания речи то это нахождение самого вероятного варианта и считать что этот вариант будет

верным для сопоставление пары аудиосигналов несущих речевую информацию Y(y1,…yn) и слов S. Тогда формула будет выглядеть так:

𝑆 = 𝑎𝑟𝑔𝑆max P(𝑆|𝑌) *(1.12)*

По теореме Байеса, данного выражение будет выглядеть так:

𝑆 = 𝑎𝑟𝑔𝑆

max P(S)P�𝑆�𝑌�

𝑃(F)

*(1.13)*

Так как вероятности полученных наблюдений нельзя поменять то выражение начинает принимать вид:

𝑆 = 𝑎𝑟𝑔𝑆max P(S)P(𝑆|𝑌) *(1.14)*

Вероятность может выражаться тем, что P(S|Y) является акустической моделью а P(S) языковой моделью и может ли в данном языке существовать такая последовательность.

СММ наблюдения является не нулевой набор состояний zi и имеющий качества:

N – количество z состояний

πi – стартовое распределение вероятностей,

𝑁

∑

𝑖𝑖=1

π𝑖𝑖 = 1

*(1.15)*

A={ai,j}- вероятность изменения состояния от zi до zj

𝑁

∑

𝑗𝑗=1

𝑎𝑖𝑖,𝑗𝑗 = 1, 1 ≤ 𝑖𝑖 ≤ 𝑁

*(1.16)*

Bi (y) или Bi (m) – вероятность эмиссии:

∫ Bi(y)dv = 1 1 ≤ 𝑖𝑖 ≤ 𝑁, *(1.17)*

Интегрирование проходит во всех МЧКК измерениях,

𝑀

∑

𝑚=1

B𝑖𝑖 (m) = 1,

1 ≤ 𝑖𝑖 ≤ 𝑁, *(1.18)*

Для улучшения качества распознавания переходы из более поздних состояний в более ранние запрещаются так как это мешает распознаванию и состояние может совершать только 2 типа переходов в самого себя и в следующее состояние так как это ограничивает количество состояний до вектора. Если за период T, состояния в моменты времени можем принять за (k1…k T)

Проблему распознавания наблюдений Y(y1,…yT) моделью λ(A,B, π) и

выстраивания лучшей последовательности Решается с помощью алгоритма Витерби.

#### Алгоритм Витерби

Алгоритм Витерби это поиск самого точного пути и его сохранение. Вероятность такого пути:

δ𝑡(i) = max

𝑘1,𝑘2,…,𝑘𝑡−1

𝑃[𝑘1, 𝑘2, … , 𝑘𝑡−1 , 𝑘𝑡 = 𝑧𝑖𝑖 , 𝑦1, 𝑦2 … 𝑦𝑡|λ], *(1.19)*

δt(i) – вероятность самой вероятной последовательности.

Для запоминания данных о маршрутах переходов используется вектор Фt(i) который хранит позицию предыдущего состояния до состояния i в точке t на временной линии T. C обозначенными переменами рассматриваемый алгоритм начинает принимать вид:

Стартовые значения:

𝛿𝛿1(𝑖𝑖) = π𝑖𝑖B𝑖𝑖 (y1), Ф1(𝑖𝑖) = 0, 1 ≤ 𝑖𝑖 ≤ 𝑁. *(1.20)*

Индукция:

𝛿𝛿𝑡+1(𝑗𝑗) = max

1≤ 𝑖𝑖 ≤ 𝑁

[𝛿𝛿𝑡(𝑖𝑖)𝑎𝑖𝑖,𝑗𝑗 ]𝐵𝑗𝑗 (𝑦𝑡+1), 1 ≤ 𝑡 < 𝑇, 1 ≤ 𝑖𝑖 ≤ 𝑁. *(1.21)*

Ф𝑡(𝑗𝑗) = 𝑎𝑟𝑔 max

1≤ 𝑖𝑖 ≤ 𝑁

[𝛿𝛿𝑡(𝑖𝑖)𝑎𝑖𝑖,𝑗𝑗 ], 2 ≤ 𝑡 < 𝑇, 1 ≤ 𝑗𝑗 < 𝑁, *(1.22)*

Конец:

𝑃(𝑌|λ) = max

1≤ 𝑗𝑗≤ 𝑁

[𝛿𝛿𝑡(𝑗𝑗)] *(1.23)*

𝑘𝑇 = argmax[𝛿𝛿𝑇(𝑗𝑗)]. *(1.24)*

1≤ 𝑗𝑗 ≤ 𝑁

Воспроизведение маршрута:

𝐾𝐾𝑡 = Ф𝑡+1(𝐾𝐾𝑡+1), 𝑡 = 𝑇 − 1, 𝑇 − 2, … ,1. *(1.25)*

## 1.6 Исследования по транскрибированию и акцентуации

Связь между фонетической формой и буквенной формой слов исследуется не первое десятилетие и большой путь развития. Самыми фундаментальными работами, посвященными русской фонетике, можно признать труды и исследования Зализняка[8], Аванесова[9] и Воронцовой[10] и др. Большинство автоматизированных систем перевода из

графем, более общего понятия, включающего в себя буквы, в фонемы ссылаются на данные труды так как правила и примеры, выработанные в них, являются основой для данных систем.

Знаковой работой можно назвать труд Кривновой и команды, опубликованный в 1970 году. Это одна из первых работ, где было представлена система автоматической транскрипции русского языка, основанной на выработанных правилах. В 2001 году Кривнова [12] и др. представили основанную на правилах систему перевода из буквенной формы фонетическую, развивавшая идеи различных уровней абстракций и игнорирования аллофонов для определения различных типов звуков применительно для синтеза речи из текста (TTS). Еще одна основанная на правилах система транскрипции была реализована Кипятковой и Карповым [9]. Работа отличается разными транскрипциями для слов и фраз.

Помимо подходов, основанных на правилах, в настоящее время наблюдается научная тенденция к использованию статистического анализа и машинного обучения (ML) в задачах обработки естественного языка (NLP) и синтеза речи из текста. Эти методы предполагают предсказание ударений или фонем на основе графического представления слов [1].

# ГЛАВА 2. ПРИМЕНЕНИЕ АЛГОРИТМОВ ТРАНСКРИБИРОВАНИЯ И АКЦЕНТУАЦИИ

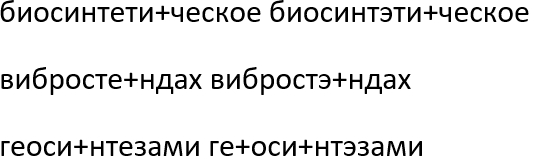
## Транскрибирование

Транскрибирование – это перевод стандартной формы слова в его фонетическую транскрипцию.

Правила транскрибирования, используемые в данной работе [19] были разработаны на трудах Ленинградской фонологической школы (ЛФШ) где учтены 6 гласных и 36 согласных звука. Также была добавлена фонема “ы” так как ЛФШ не рассматривает данную фонему, также в правилах учтены посторонние звуки, такие как шум, мычание, рычание и т.д. Учитываются ударные и не ударные гласные.

#### 2.1.1. Алгоритм транскрибирования

* Нахождение слова в списке исключений и выдача ему готовой правильной транскрипции. Слова должны быть в форме



*Рис. 2.1 примеры исключений*

* Предварительные правила транскрибирования

1. В комбинации букв “стн”, “стл”, “нтг” - буква т убирается
2. В комбинации букв “здн” “здц” “ндц” “рдц” “ндш” “гдт” – д убирается
3. В комбинации букв “лнц”– л убирается
4. Производятся замены вместо в комбинации букв - “сч” “жч” на

“щ”, “тс” “тьс” “тц” “дс” “дц” на “Ц”, “сш” на “ш”, “зж” на “ж”

1. Комбинации букв “ого”, “его”, “го” находящиеся в конце слов

заменяются на “ово”, “ево”, “во”, также есть список исключений:

P364#yIS1

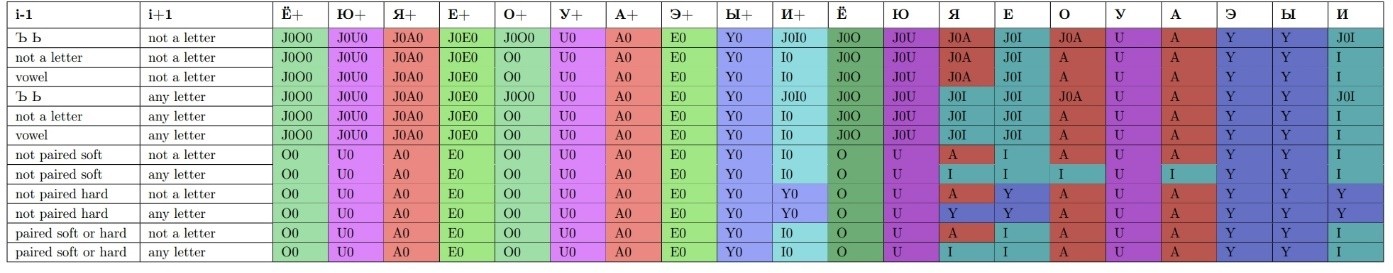
*Рис. 2.2 Примеры исключений для правила 5*

* Основные правила транскрибирования слова считываются справа налево и применяются к каждой букве

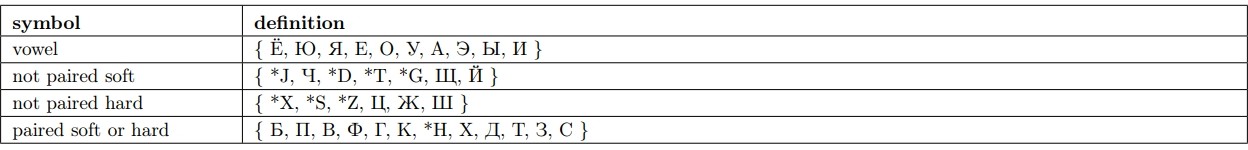
1. Если перед “Н” будет мягкие звуки [й] [ч] [ш] [дж] [ж] [д] [т] [з] [с] то “Н” транскрибируется твердым звуком
2. Если перед “C” или “З”:

a. [д`] [з`] то транскрибируется как мягкий и звонкий звук б. [т`] [с`] то транскрибируется как мягкий и глухой звук в. [н`] то транскрибируется как мягкий звук

1. Замена гласных в зависимости от соседей



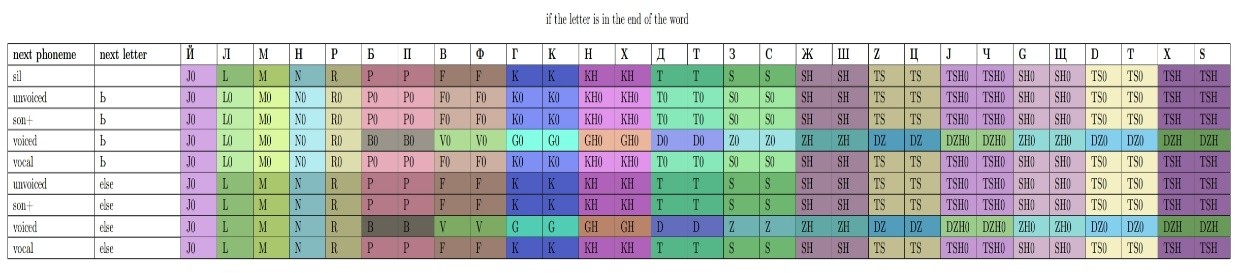
*Рис. 2.3 Таблица соответствия замен гласных звуков в зависимости от соседей*



*Рис. 2.4 Используемые группы фонем для замены гласных*

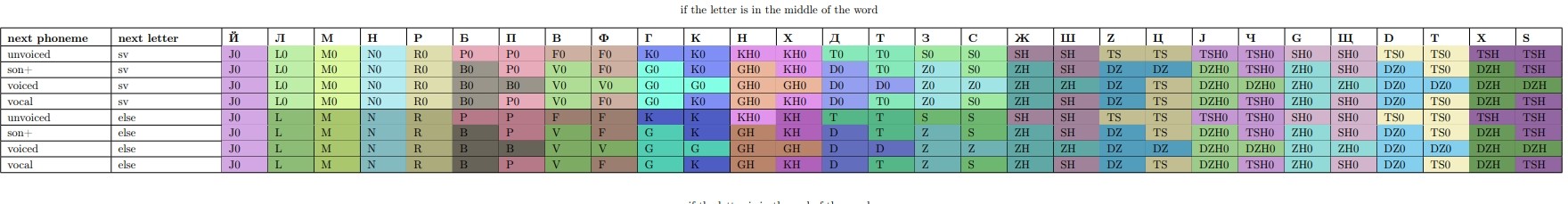
1. Замена согласных в зависимости от соседей

* Если согласная в конце слова

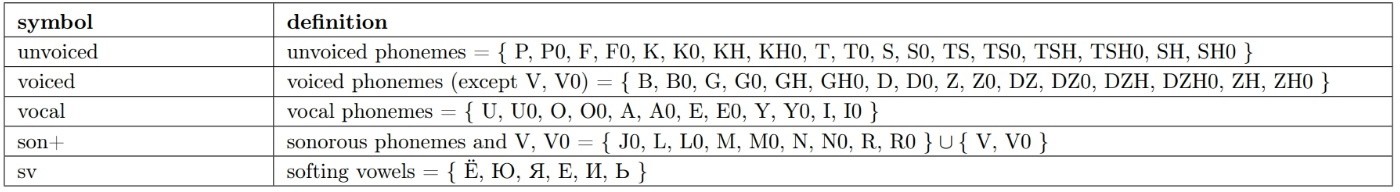


*Рис. 2.5 Замена согласной в конце слова по правилам*

* Если согласная в начале слова



*Рис. 2.6 Замена согласной в середине слова*



*Рис. 2.7 Список используемых для замены согласных групп фонем*

* Заключительные правила транскрибирования

1. если подряд идут одинаковые гласные звуки, то заменяются на долгий звук [а] [а] -> [аа]
2. если перед сложным звуком его под часть, то заменяются на долгий сложный звук [з] [ж(zh)] -> [ж]

## Акцентуация

Основная проблема автоматизации акцентуации для русского языка является не фиксированное ударение у слов, что одно слово может иметь несколько произношений, а следовательно, значительно усложняет процесс распознавания речи, пример: ́окна или окн́а. Такие множества слов называются омографы.

Список самых часто употребляемых русских слов был составлен в электронном виде на основе грамматического анализа словообразовательных средств данного электронного ресурса[15].

Автономный словарь для модуля акцентуации был построен с помощью

следующего ресурса. Он включает в себя простой список слов с ударением (которые имеют однозначный вариант ударения) и более сложную структуру данных для омографов. Омографы в словаре перечислены без ударения, за ними следует информация о части речи, числе, лице такие правила называются морфологическими.

соответствующие графические представления с ударением для перечисленных морфологических форм.

* 1. Поиск в списке не омонимов
  2. Поиск в списке омонимов выбирая точную форму, соответствующую слову и тегу.
  3. Использование нейронной сети для расстановки ударения по морфологическим признакам для омонимов отсутствовали в списке омонимов.
  4. Сохраняем в локальный список для ускорения работы алгоритма.

#### Общие подходы к Нейронным сетям

Компьютерные нейронные сети – упрощенная форма функционирования усложненных структур человеческого мозга. Всякий один отдельный нейрон является неким переключателем, который включается при наличии определенной информации и действий. В обычной нейронной сети информация проходит от входа к выходу, перемещаясь через нейроны в определенное направление. Если сигналы с входного слоя посылают прямо на выходной слой, то такие сети называют однослойными. Для того, чтобы повысить точность и приучить нейронную сеть к определенным свойствам между входом и выходом создаются скрытые

слои.

Эти сети состоят из набора отдельных слоев, и их обозначают как многослойные сети глубокого обучения, Такие сети используются для разных сильно отличающихся задач, и задача расстановки ударений по морфологическому признаку тоже решается с помощью подобных

многослойных сете глубокого обучения.

#### Рекуррентные нейронные сети

Модели нейронных сетей, распространяющиеся последовательно при добавлении разных внутренних слоев называемых скрытными, позволяют обучать сети решать задачи абсолютно разной применимости]. Особенность подобного типа моделей нейронных сетей наличия жесткого количества независимых от друг друга входных слоев, в этом случае соединения нейронов могут вернуть информацию обратно в себя. Что дает имитацию памяти и нейроны начинают хранить данные. Стандартные возвратные или по-другому рекуррентные сети не предназначены для задач, в которых данные нужно запоминать на длительный периоды, так как чем больше шагов или эпох совершает сеть, тем меньше промежуточное или по другое скрытое состояние влияет на последующие далее состояния. Тут приходят такие решения как долгая краткосрочная память (long short term memory) LSTM и спустя 17 лет ее более простая, но более эффективная версия фильтрующий или более прямо воротный рекуррентный юнит (Gated Recurrent Unit) GRU. Так возвратные или рекуррентные нейронные сети стали уметь запоминать значения предыдущих состояний на более долгое время, что дало еще более широкий список применений данного вида систем. Так как у данных решений нет активирующей функции.

#### LSTM

Одно из самых удачных решений популяризировавшее рекуррентные сети появилось как ответ на проблему взрывного градиента, с помощью размещения на каждом нейроне ячеек хранения данных, фильтрующих ворот или просто фильтров. Количество ворот на каждом нейроне зависит от модели используемой LSTM, в классической модели каждый нейрон несет на себе по 3 фильтра:

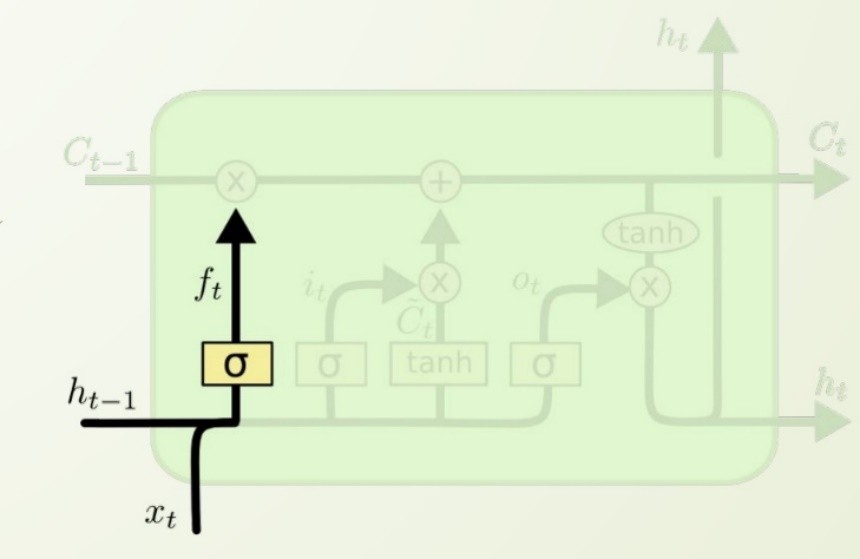
* Фильтр входа занимается управлением данными, получаемыми нейроном, и определяет сколько нужно хранить в памяти

данных полученных с предыдущего шага

* Фильтр выхода занимается менеджментом информации выходящей из фильтра
* Фильтр забывания определяется сигмоидальной функцией:

Принцип работы:

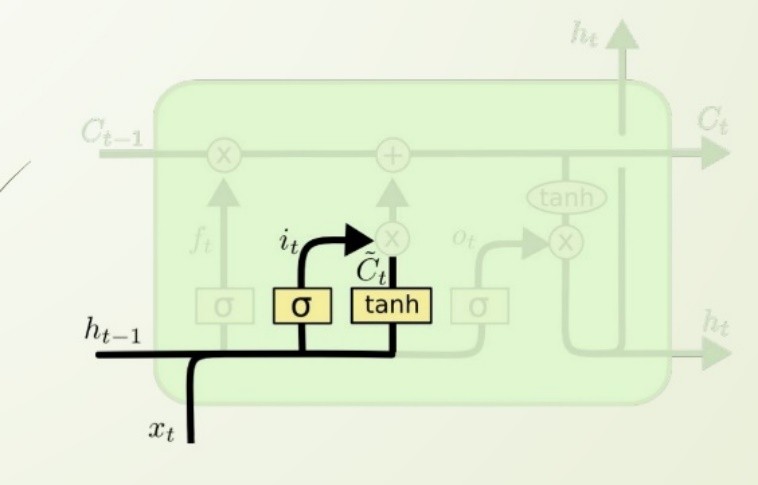
1. Сперва происходит процесс забывания определенной части информации и выражается он через формулу:

𝑓𝑓𝑡 = 𝜎(𝑊𝑊𝑓𝑓 ∗ [ℎ𝑡−1, 𝑥𝑡]) *(2.1)*

*Рис. 2.8 Первый шаг LSTM*

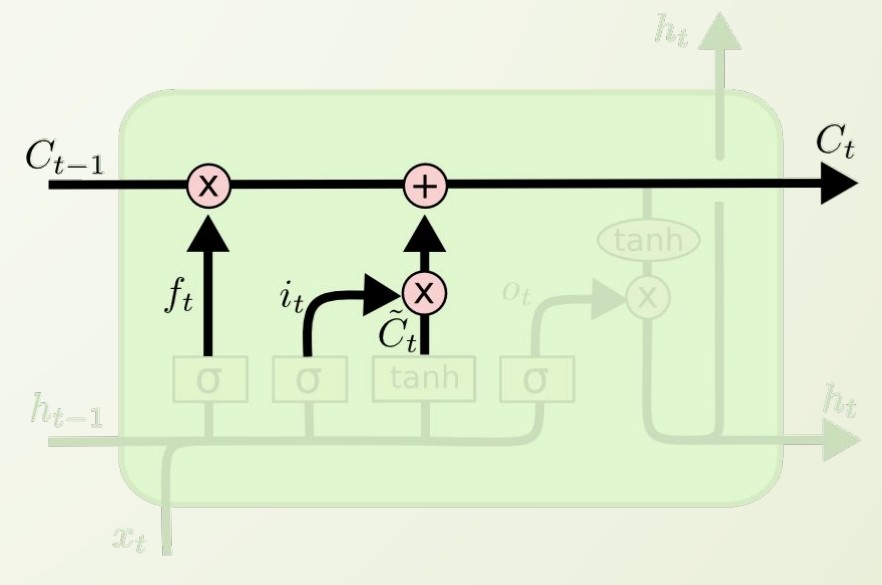
1. На втором шаге фильтр входа и tanh слой определяют какую информацию нужно сохранить:

𝑖𝑖𝑡 = 𝜎(𝑊𝑊𝑖𝑖 ∗ [ℎ𝑡−1, 𝑥𝑡] + 𝑏𝑖𝑖 ) *(2.2)*

𝐶̃𝑡 = 𝑡𝑎𝑛ℎ(𝑊𝑊𝐶 ∗ [ℎ𝑡−1, 𝑥𝑡] + 𝑏𝐶) *(2.3)*

*Рис. 2.9 Второй шаг LSTM*

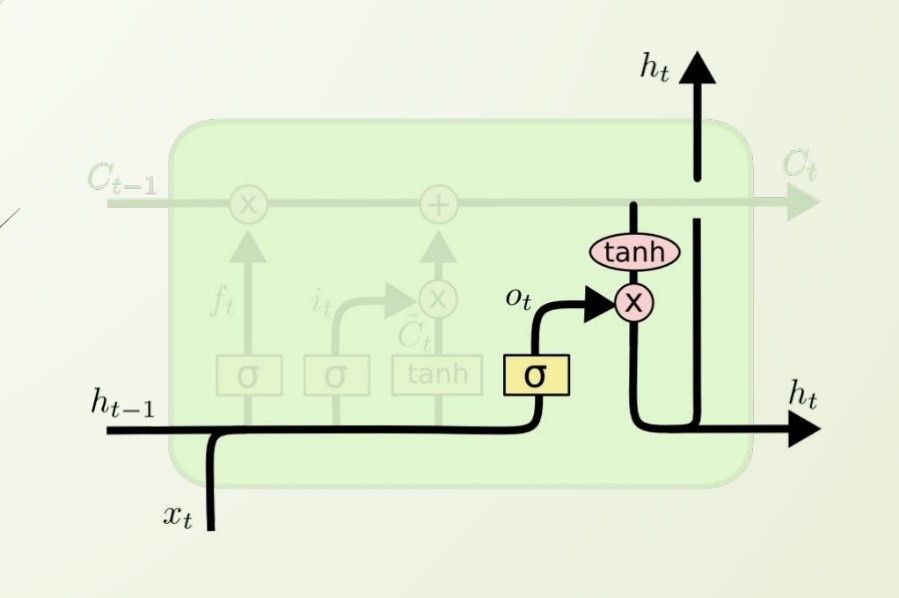
1. Третий шаг завершает процесс изменения состояния:

𝐶𝑡 = 𝑓𝑓𝑡 ∗ 𝐶𝑡−1 + 𝑖𝑖𝑡 ∗ 𝐶̃𝑡 *(2.4)*

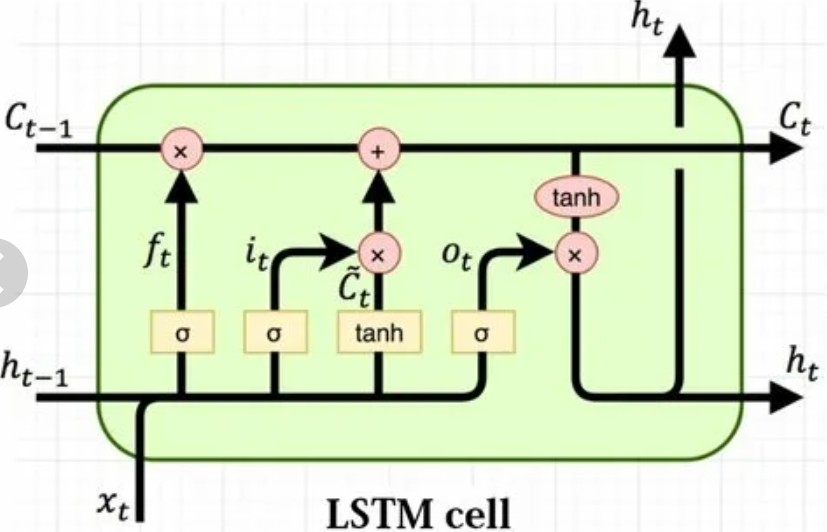
*Рис. 2.10 Третий шаг LSTM*

1. Четвертый шаг выносится информация из текущего состояния ячейки памяти:

𝑜𝑡 = 𝜎(𝑊𝑊𝑜 ∗ [ℎ𝑡−1, 𝑥𝑡] + 𝑏𝑜) *(2.5)*

ℎ𝑡 = 𝑜𝑡 ∗ tanh (𝐶𝑡) *(2.6)*

*Рис. 2.11Четвертый шаг*



*Рис. 2.12 Визуальное представление нейрона LSTM*

#### GRU

Одна из вариаций LSTM пошедшая по альтернативному пути упрощения нейронных сетей хотя тенденция всегда была в сторону усложнения. Обладает двумя фильтрами вместо трех как это было у LSTM, а в наличии у него фильтры:

* Фильтр перезагрузки или сброса текущего состояния имеет аналогичное применение, как и фильтр забывания у LSTM, но с измененным расположением
* Фильтр обновления похож на фильтр входа у LSTM и определяет какие данные хранить с предыдущего состояния и сколько данных нужно пропустить из слоя, предшествовавшего этому.

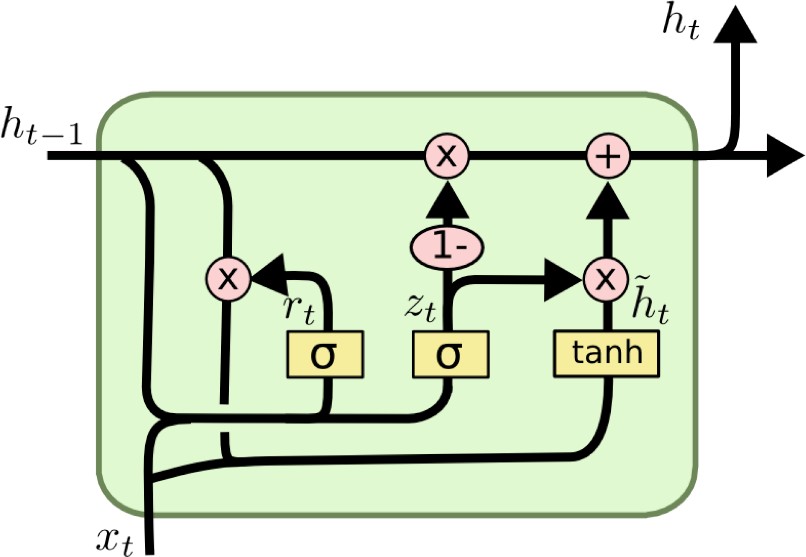
Выражается формулами:

𝑧𝑡 = 𝜎(𝑊𝑊𝑧 ∗ [ℎ𝑡−1, 𝑥𝑡]) *(2.7)*

𝑟𝑡 = 𝜎(𝑊𝑊𝑟 ∗ [ℎ𝑡−1, 𝑥𝑡]) *(2.8)*

ℎ�𝑡 = tanh(𝑊𝑊 ∗ [𝑟𝑡 ∗ ℎ𝑡−1, 𝑥𝑡]) *(2.9)*

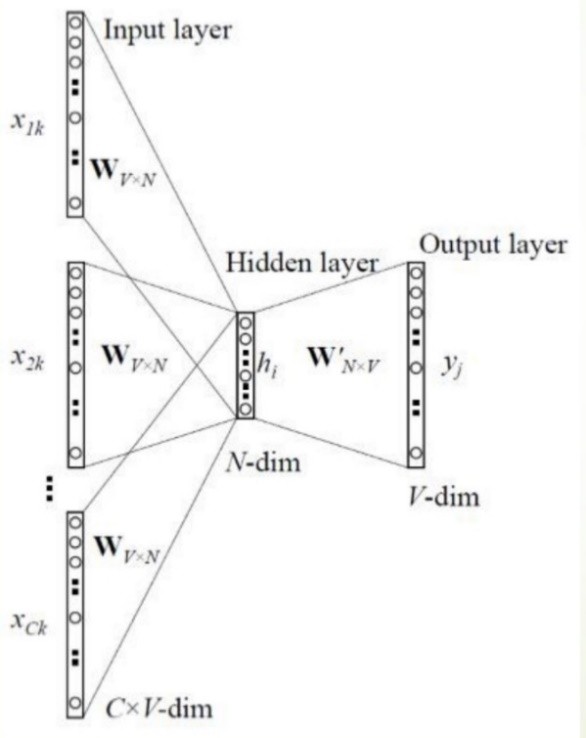
ℎ𝑡 = (1 − 𝑧𝑡) ∗ ℎ𝑡−1 + 𝑧𝑡 ∗ ℎ�𝑡 *(2.10)*



*Рис. 2.13 Схема модели нейрона GRU*

#### CBOW

Задача определения ударения по морфологическим признакам лежит в поле задач слово в вектор и одним из лучших решений в данной области является Continuous Bag-of-Words (CBOW) (на рис. представлена архитектура данной нейронной сети).



*Рис. 2.14 Архитектура нейронной сети СBOW*

Ее особенности она имеет 3 входных и один выходной слой и описывается формулой:

𝑝(𝑐 |𝑤, 𝖯

𝑊𝑊

, 𝖯

) = 𝑒

𝖯𝑇 ∗𝖯𝑐

*(2.11)*

𝑖𝑖

𝑤𝑤 𝑐

𝘍 𝖯𝑇 𝘍∗𝖯𝑐

* С – количество слов контекста
* V – размер словаря

∑ 𝑤𝑤 ∈𝐷𝑒

𝑊𝑊

* N – количество нейронов в скрытом слое\ One-hot векторы слов из контекста суммируются

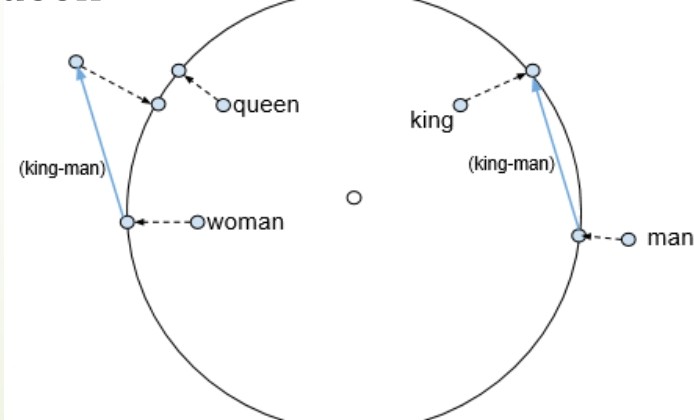
Для обучения модели есть 6 шагов:

1. Читаем корпус считая частоту слова w
2. Удаляем самые редко встречающиеся слова частота появления которых в тексте меньше, чем уровень фильтра шумовых данных
3. Проходим окном размер, которого определяет глубина левого и правого контекстов С по выделенному из корпуса предложению
4. Для слов в окне делаем один шаг алгоритма обратного распространения ошибки для CBOW-нейросети (вход – усреднённые one-hot вектора C слов справа и C слов слева от слова в середине окна, желаемый

выход – one-hot-вектор слова в середине)

1. Повторяем шаги 3-4, пока не прочитаем все субпредложения корпуса.
2. . Повторяем шаги 3-5, пока не выполним заданное число эпох обучения

(обычно небольшое число – две-три эпохи).



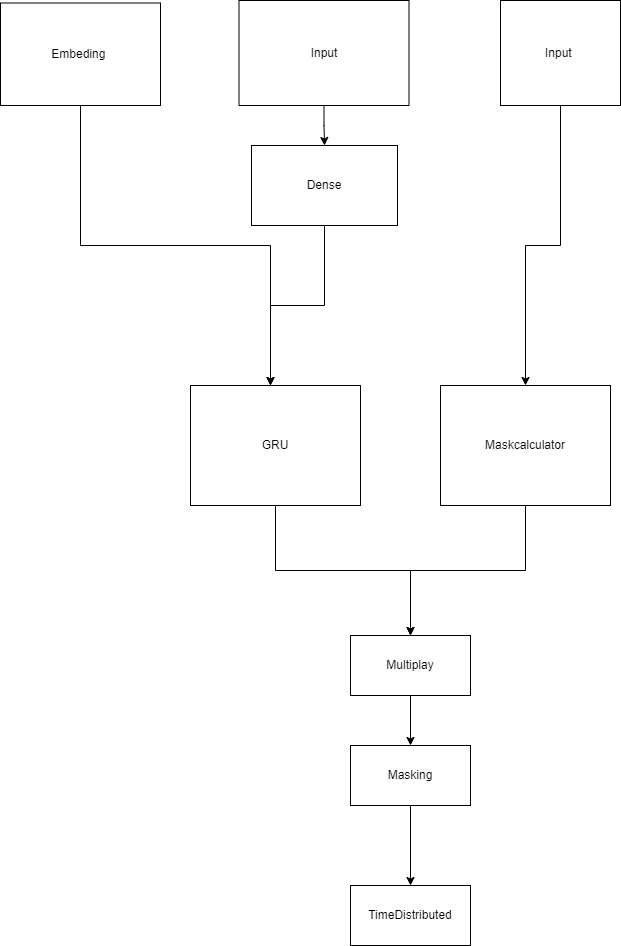
*Рис. 2.15 Визуализация близости слов по параметрам*

Плюсы данной модели нейронной сети:

* Эффективное сокращение размерности векторов слов
* Вектора слов, похожих по смыслу, тоже близки (имеют меньшее косинусное расстояние, чем вектора семантически далёких слов)

Модуль Транскрибирования и акцентуации основан на работе [2] И улучшен путем замены обращения к Викисловарю на нейронную сеть на подобии CBOW. для решения задачи расстановки ударений на основании части речи или по-другому названную акцентуацией.

#### Разработанная нейронная сеть



*Рисунок 2.16 Схема разработанной нейронной сети*

Параметры нейронной сети

* Embeding: Получает на вход векторы слов

размер словаря 34 и игнорирует маскированные параметры имеет размер слоя 256 нейронов.

Размер входа не фиксированный. Занимается маскированием матриц цепочек индексов символов.

* Input(1): Получает на вход 59 параметров с плавающей запятой векторы морфологической информации.
* Dense: обрабатывает первый инпут имеет размер в 256 нейронов
* Gru: Рекурентный слой типа GRU имеет размер в 256 нейронов, получает на вход результаты слоя Embeding, начальные значения получает из cлоя Dense.
* Input (2): Получает векторы согласных
* MaskCalcutor: простой самописный слой не тренирующийся слой помечает все вектора согласных нулем.
* Multiplay: перемножает получаемые вектора
* Masking: снимает маски
* TimeDistributed: слой Dense в оболочке c одним нейроном активатор сигмоид)

Формула сигмоидальной функции выражается так:

𝜎(𝑥) = 1 1+𝑒−𝑥

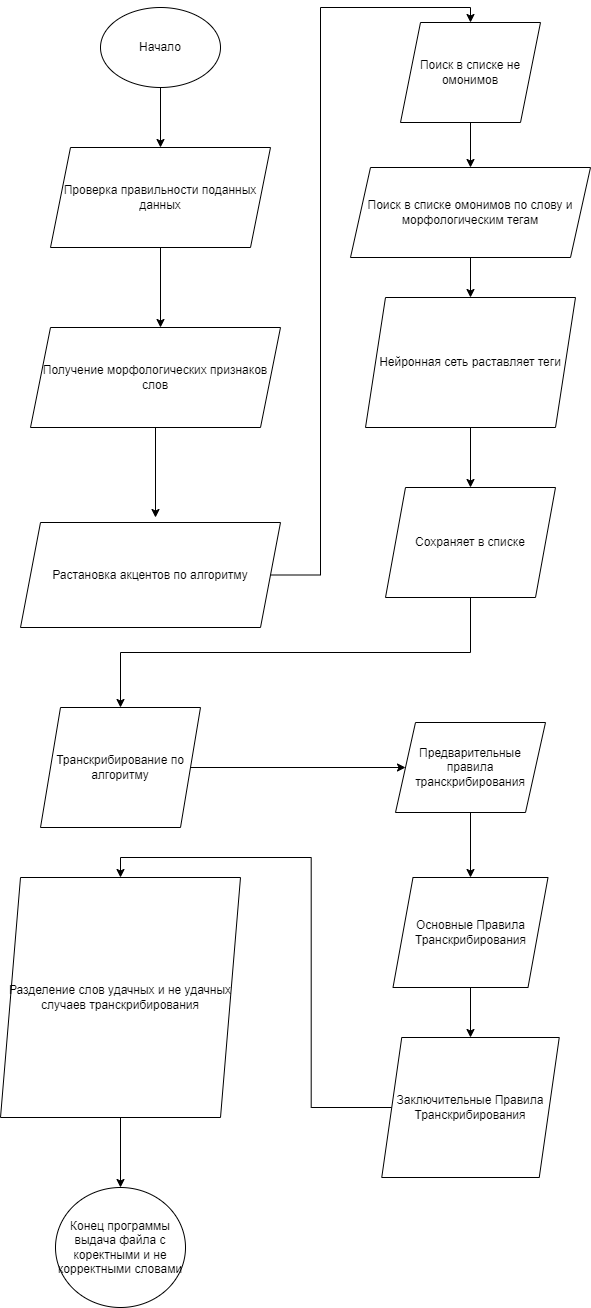
(2.12)

В качестве классификатора были выбраны фокальные потери Фокальные потери чрезвычайно полезны для классификации, когда у вас есть сильно несбалансированные классы. Она снижает вес хорошо классифицированных примеров и фокусируется на трудных примерах. Значение потерь гораздо выше для образца, неправильно классифицированного классификатором, по сравнению со значением потерь, соответствующим хорошо классифицированному примеру. Также были проведены эксперименты с заменой слоя GRU на LSTM, и количеством нейронов в каждом слое. и конфигурация GRU и 256 нейронов оказалось самой точной.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип реккурентного слоя | Количество нейронов | точность | Количество эпох | время обучения |
| GRU | 128 | 0,881 | 15 | 2 часа |
| LSTM | 128 | 0,876 | 15 | 2.5 часа |
| GRU | 256 | 0,946 | 15 | 4 часа |
| LSTM | 256 | 0,943 | 15 | 5 часов |
| GRU | 512 | 0,945 | 15 | 7 часов |
| LSTM | 512 | 0,942 | 15 | 9.5 часов |

Таблица 2.1 результаты экспериментов

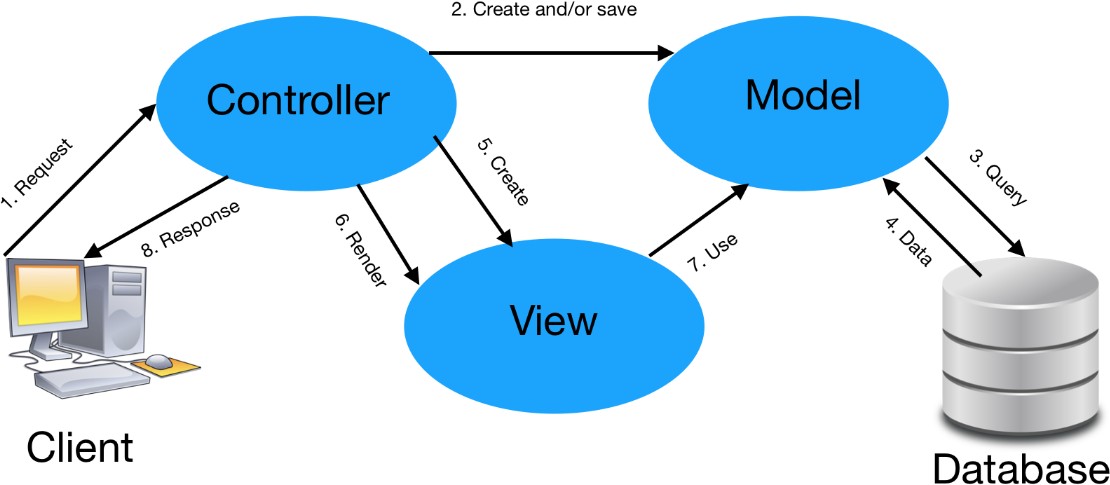
#### 2.2.6 Алгоритм создания словаря



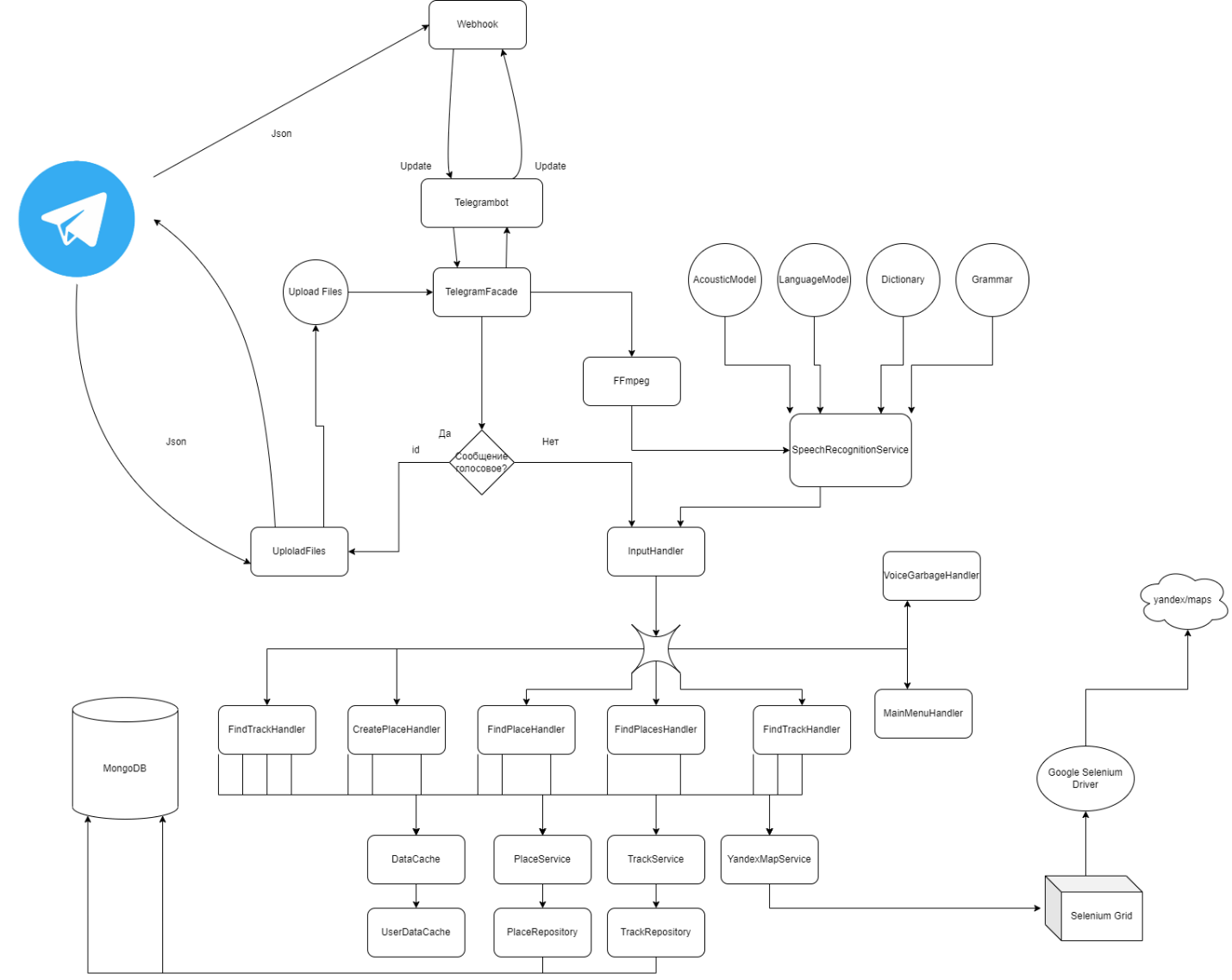
*Рис. 2.17 Алгоритм создания словаря*

## Архитектура приложения

### Основой для приложения была выбрана архитектура модель представление контроллер



*Рис. 2.18 Пример архитектуры MVC*

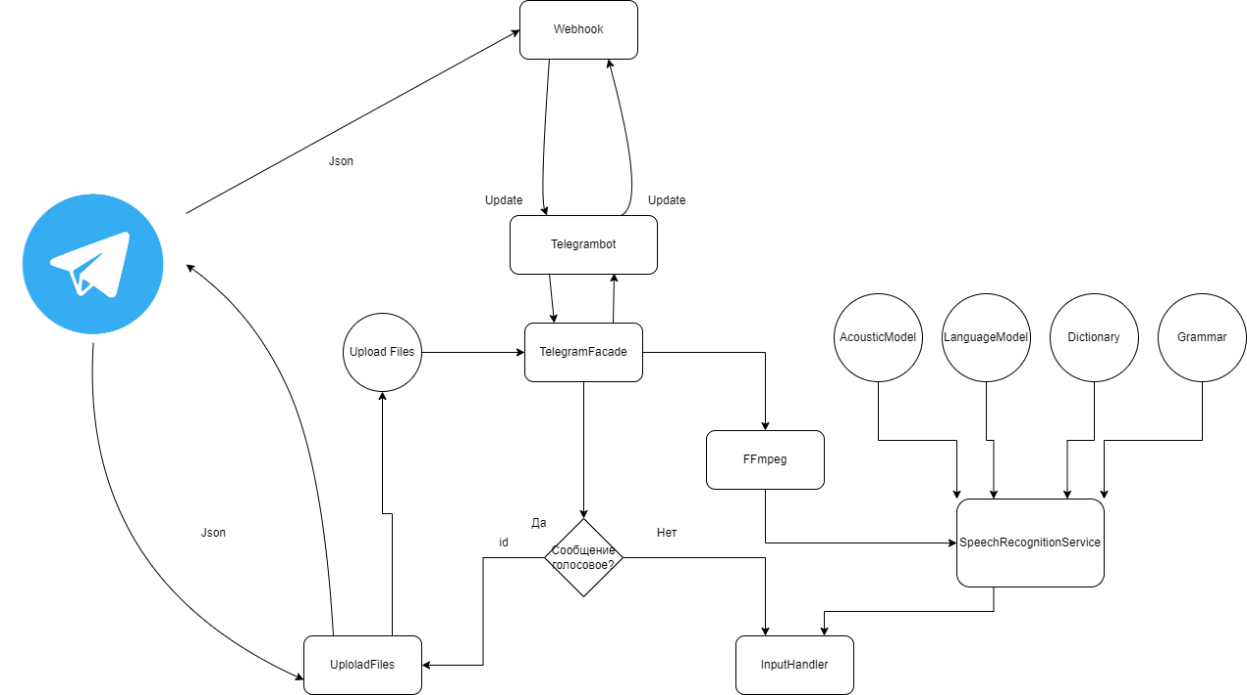


*Рис. 2.19 Схема Архитектуры разработанного приложения*

Схема показывает алгоритм работы Telegram бота имеющего поддержку голосового управления и вырабатывающий поиск определенных мест и маршрутов, а также сохраняет их в базу данных. Реализован на паттерне конечного автомата и сохраняет стадию пользователя.

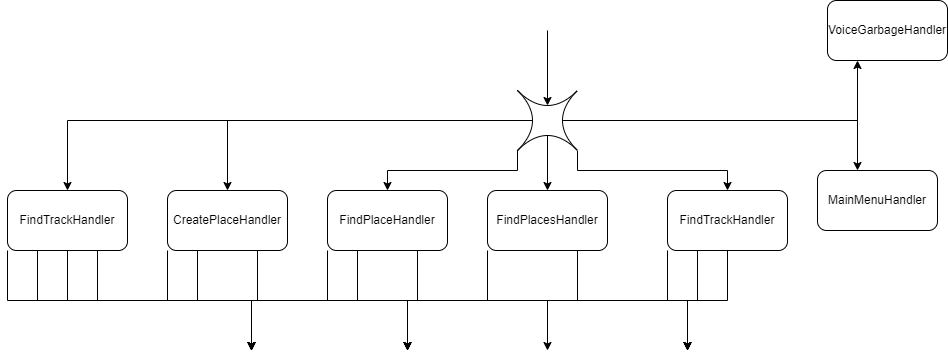
Рассмотрим 3 части приложения:

Контроллер и система распознавания речи:



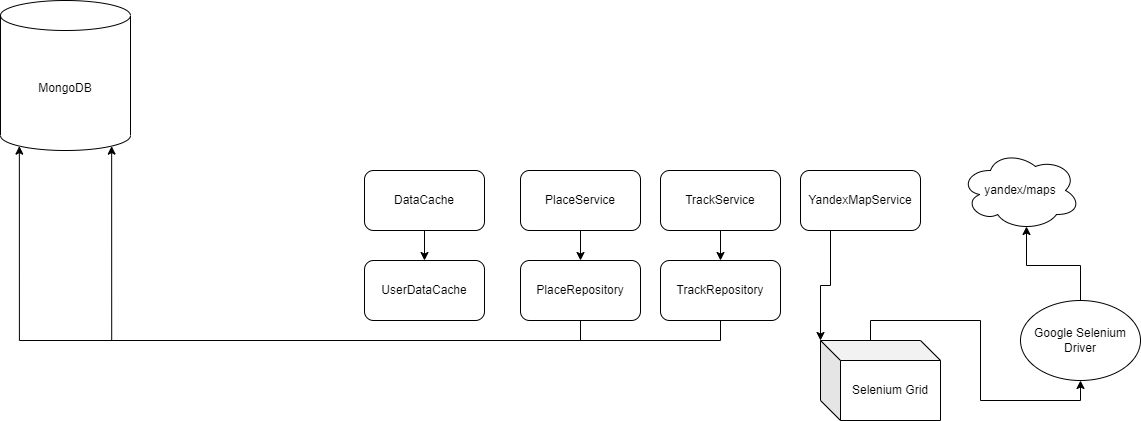
*Рис. 2.20 Слой получения и обработки данных*

Обработчики сообщений



*Рис. 2.21 Слой обработчиков данных*

Сервисы и репозитории



*Рис. 2.22 Сервисный слой, совмещённый со слоем модели и разных утилит*

Алгоритм работы приложения:

* 1. Бот получает сообщение, на первое сообщение бот выдает сообщение со всеми доступными командами бота
  2. Бот получает второе сообщение и если оно не соответствует ни одной команде бот снова выдает список команд бота
  3. Если команда правильная и пришла текстом, то команда сразу попадает в обработчик и меняет стадию бота для пользователя дальше зависит от конкретной команды
  4. Если команда пришла в виде голосового сообщения, то происходит ее получение из хранилища Telegram и с помощью библиотеки FFmpeg преобразования файла из формата oga в формат wav и настройка определенных параметров таких как битрейт, количество каналов, кодек и частоту дискретизации характерных для используемых для голосового распознавания речи. Дальше идет попытка распознать аудиофайл и получить текст, если текст распознан, но не соответствует ни одной команде то бот возвращает текст, в котором указывается какая фраза была распознана. Дальнейший ход зависит от конкретной команды:
     1. Задать и дальше конкретный набор слов, привязанных к месту такие слова как (место, дом, работу, ресторан). Дальше идет ответ от бота и переход в следующую стадию, где бот хочет получить адрес искомого места, голосовую поддержку адресов было не целесообразно так как количество географических названий не позволяет ограничить словарь достаточно хорошо, что достаточно сильно ухудшит распознавание слов и размер словаря расшириться многократно. Поэтому бот ждет текстовый адрес. После получения адреса бот пытается создать ссылку на искомый адрес через Selenium Web Driver. Если адрес удалось сохранить. Бот сохраняет в базу данных место. И выдает последним сообщением заданное название места и сформированную ссылку, а дальше возвращает бота в состояние главного меню;
     2. Все места команда выдает все сохранённые в базе данных места;
     3. Все маршруты получение всех сохранённых маршрутов в базе данных;
     4. Маршрут – бот переходит в следующую стадию и требует уже заданные в базу данных места, вызовы поддерживают голосовые запросы, если удалось найти в базе данных заданное место он переходит в следующую стадию и просить следующее заданное место если есть 2 место то бот пытается построить маршрут и последним сообщением перед возвращением в главное меню выдает маршрут место

-> место и прикрепляет ссылку на маршрут.

* + 1. Найти место получает данные из базы данных и возвращает ответ.

#### Инструментарий и технологии разработки приложения

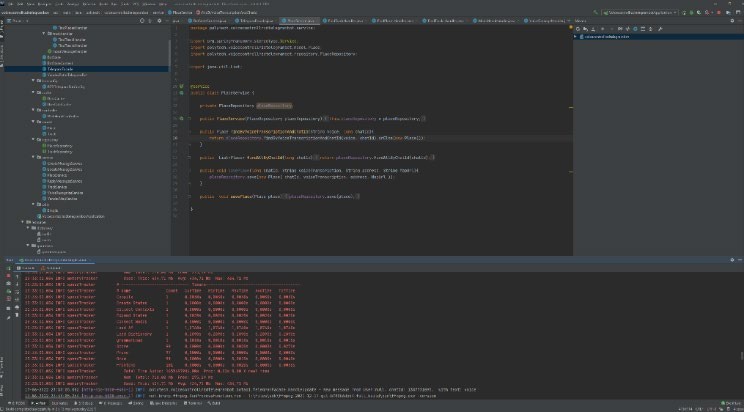
#### Среда разработки.

При выборе среды разработки для реализации подсистемы главными факторами были:

* + Поддержка языка Java 8+
  + удобство разработки;
  + скорость разработки (поддерживаемые автоматические функции рефакторинга и генерации кода);
  + наличие статического анализатора кода;
  + наличие встроенного отладчика;
  + Поддержка различных систем автоматизации сборки проекта;
  + Поддержка систем контроля версий (Git);
  + Расширенная поддержка дебага (пошаговое исполнение кода с интересующей нас точки останова)

В качестве такой среды, поддерживающей все вышеописанные функции, была выбрана среда Intellij Idea Ultimate edition (рис. 2.4.1).

Версия: 2020.1 (на рис. 2.1 показан интерфейс среды разработки во время работы программы)



*Рис. 2.23, Главное окно среды разработки*

Данная среда разработки упрощает жизнь разработчика и имеет обширные возможности для поддержки различных java фреймворков и систем сборки.

#### система автоматизации сборки

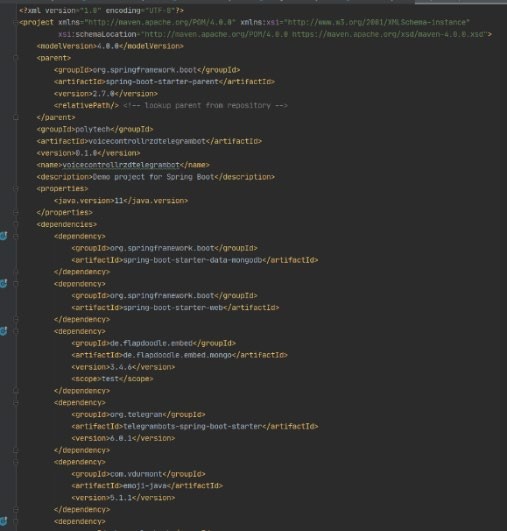
В java проектах не используются виртуальные окружения. Вместо этого используются системы автоматизации сборки (САС) проекта, которые напрямую управляют зависимостями проекта. Чтобы не хранить все используемые библиотеки в проекте они подтягиваются из репозиториев.

* Сначала производиться поиск всех подключенных к проекту библиотек в определенном порядке:

1. Поиск в локальном репозитории
2. Поиск в дополнительно подключенных репозиториях
3. Поиск в центральном репозитории

* Если библиотеки найдены не в локальном репозитории их загружает в локальный репозиторий и подключает к проекту.
* Локальный репозиторий может хранить разные версии одной библиотеки. Так как для подключения к проекту указывается версия библиотеки.

В работе использовалась САС Maven. Классическая САС написанная на java и управляющая зависимостями проекта с помощью специального XML файла POM (На рис. 2.2 представлена часть файла настроек Maven в виде XML).



*Рис. 2.24 pom.xml файл настроек Maven*

#### Фреймворк

Фреймворк — это каркас приложения, который закрывает собой низкоуровневые задачи, что уменьшает время разработки и обновления приложения.

В качестве серверной стороны проекта используется фреймворк Spring

boot – современный стандарт коммерческой разработки на java.

Имеет интеграцию с большинством написанных на java библиотек. На его основе можно строить различные архитектурные решения включает в себя:

* 3 веб сервера на выбор (по умолчанию Apache Tomcat);
* Систему управления потоков;
* Большую коллекцию различных библиотек подключаемых отдельно;
* Пакеты библиотек, подключающие все необходимые для работы основной библиотеки зависимости;
* Поддерживает большинство существующих базы данных

В частности, для работы с базой данных используется пакет Spring boot data JPA (Java Persistence API) содержащий в себе ORM (объектно- реляционное отображение) Hibernate позволяющий работать с таблицами и полями баз данных как с объектами java. Простейшие запросы можно сгенерировать названием метода, такие как CRUD (CREATE, READ, UPDATE, DELETE) операции. Поддерживает работу с транзакциями.

Также используется библиотека Lombok – библиотека добавляет возможность заменить стандартные методы и конструкторы на аннотации к классу. Позволяет уменьшить размер исходного, когда, что ускоряет разработку. Пример использования (рис. 2.26)

Spring реализует паттерн инверсия зависимости, что перекладывает создание объектов на фреймворк. Любой класс, помеченный аннотациями класса, создается фреймворком при запуске программы, чтобы Spring создал содержащиеся внутри помеченных классов классы их нужно пометить аннотацией Autowired напрямую или по умолчанию через конструктора или через сетеры.

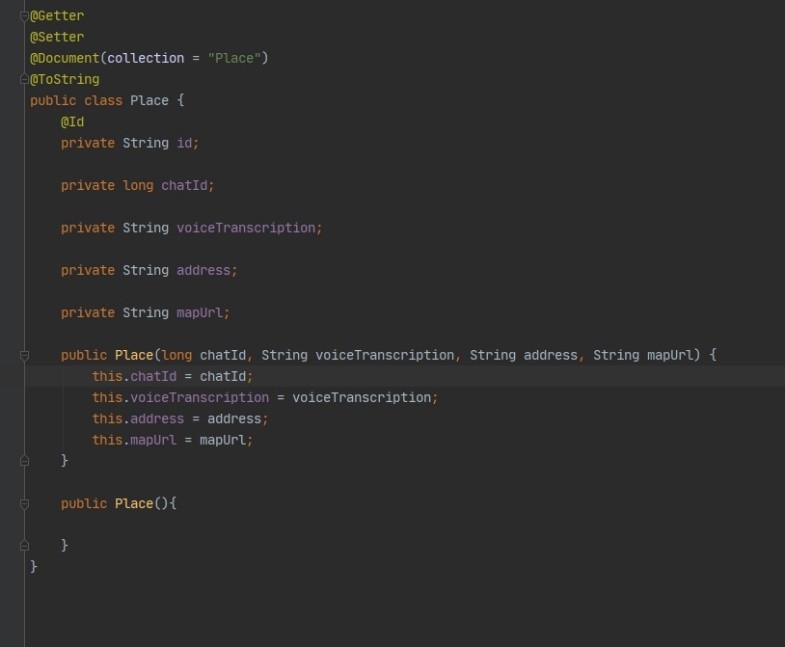
Spring boot на практике работает через аннотации 4 основные аннотации используемые для классов и аннотация старта приложения:

* Document
* Repository
* Service
* Controller
* SpringBootApplication

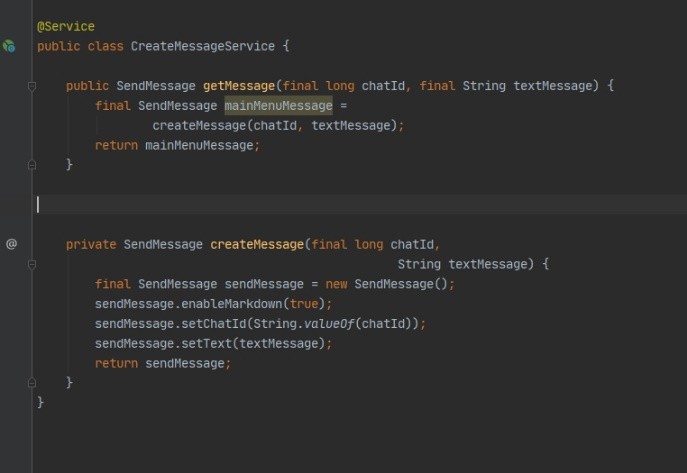
Первая аннотация зависит от типа используемой базы данных. Например, при использовании реляционных баз данных, например, как Postgress используется аннотация Entity (на рис. 2.25 2.26 2.27 2.28 2.29 было показано практическое применение аннотаций классов).



***Рис. 2.25 Аннотация старта приложения и запуск механизмов Spring***



***Рис. 2.26 Аннотация, определяющая документ в не реляционной базе данных***



***Рис. 2.27 Аннотация сервиса – основная аннотация классов.***



***Рис. 2.28 Основная аннотация Spring Data управляющая базой данных***

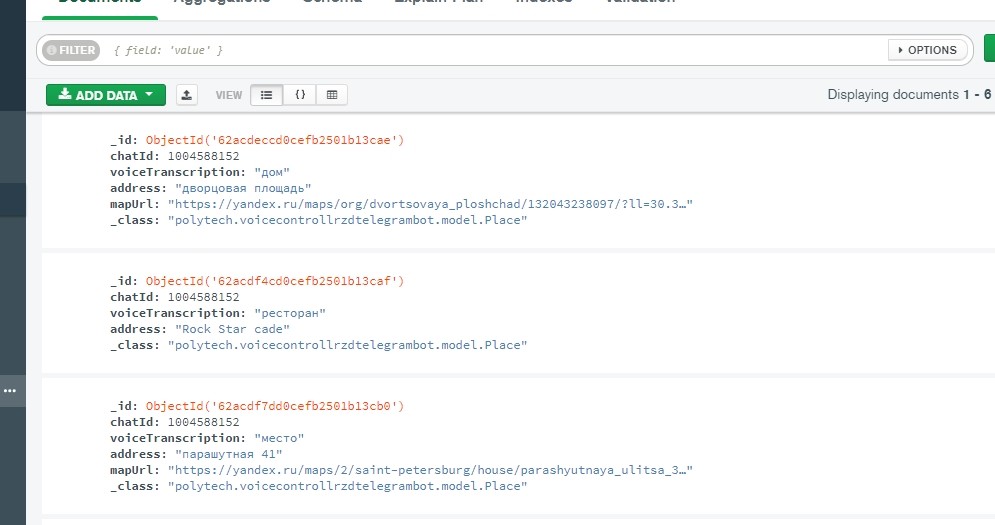
* + 1. Система управления базой данных (СУБД)

В качестве СУБД была использована не реляционная база данных

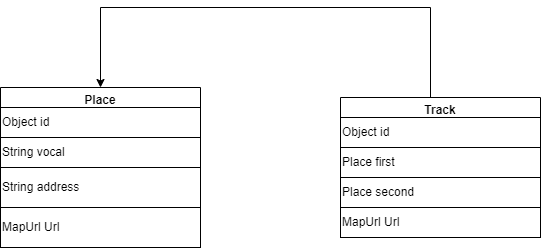
MongoDB. Ее основные плюсы:

* + Открытый исходный код
  + Относительная легковесность по сравнению с реляционными базами данных
  + Высокая скорость работы
  + Масштабируемость, что для данной работы не сильно актуально.
  + Гибкость присущая не реляционным CУБД.

Особенностью MongoDB является что bson используется в качестве аналога для таблицы и называется документом. Bson – аналог json с дополнительными типами полей. (на рис. представлено отображение существующих документов типа Track в интерфейсе СУБД MongoDB)



*Рис. 2.29 Интерфейс MongoDB Compass c готовыми документами*

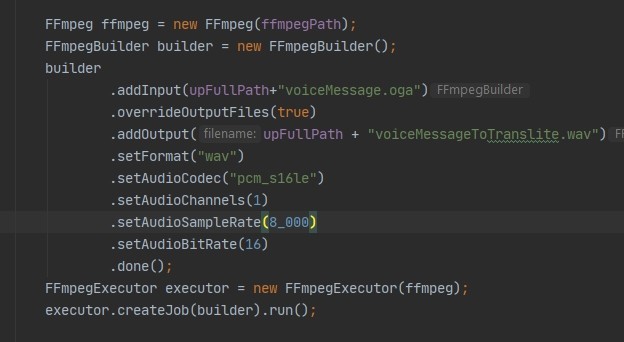


*Рис. 2.30 Схема базы данных*

#### FFmpeg

FFmpeg – известная и популярная открытая библиотека для обработки аудио и видео данных используется чтобы преобразовать аудио файл приходящий из Телеграмма в удобоваримый и хорошо воспринимаемый формат для Sphinx4. Привести к правильной частоте, битрейту, количеству каналов и формату аудиофайлов wav. Используется порт для Java. (на рис.

* 1. показано применение данной библиотеки в этой работе)



*Рис. 2.31 Применение библиотеки FFmpeg для преобразования аудиофайла в удобоваримую форму*

#### Telegram

В качестве фронтенда проекта был выбран Telegram бот. Так как это мощный инструмент и может использоваться в самых разных целях.

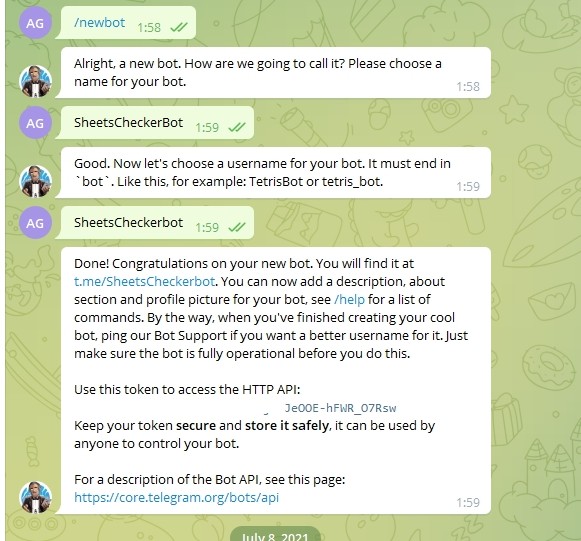
Плюсы:

* + - Поддержка голосовых сообщений
    - Возможность совмещения текстовой и голосовой информации
    - Удобный интерфейс для разработчика

Для того чтобы создать бота необходимо зарегистрировать нового бота у системного бота Telegram под названием BotFather.

Этот бот является интерфейсом для менеджмента ботов. Ограничение на одного пользователя составляет 20 ботов. Имя бота обязано заканчиваться на bot. у BotFather получаем уникальный токен бота. (на рис.

* 1. представлена регистрация в приложении Telegram)



*Рис. 2.32 Пример создания Telegram бота*

Telegram может двумя способами взаимодействовать с приложением:

1. Long Pooling - Приложение постоянно запрашивает данные у Telegram не появились ли новые сообщения и если появились проводить их обработку.
2. Web Hook – При появлении новых сообщений Telegram отправляет данные по определенному адресу URL. Адрес должен поддерживать https протокол и иметь SSL сертификат. Что существенно снижает нагрузку на приложение. Ответы из Telegram приходят в виде json

В данной работе был использован второй вариант. Настройка адреса URL происходит через запрос. <https://api.telegram.org/>bot Token

/setWebhook?url= адрес

Для получения любых файлов в Telegram, в частности звуковых файлов. Необходимо извлечь id файла из сообщения в котором этот файл был отправлен или записан. Получаем json по извлеченному id: <https://api.telegram.org/>bot Token/getFile?file\_id=file\_id

Выделяем из json путь до файла на сервере, где лежит интересующий нас файл. И последним шагом загружаем файл формате .oga:

<https://api.telegram.org/file/>bot Token/file\_path

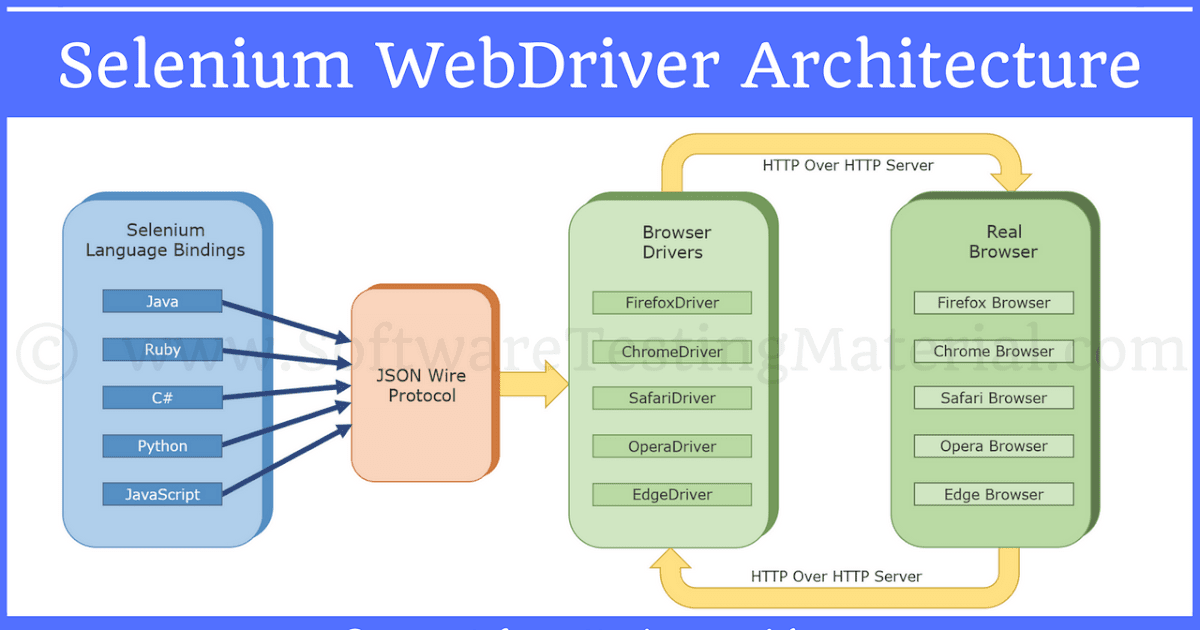
#### Selenium

Selenium это несколько модулей выполняющих разные функции. В данной работе использовались 2 модуля:

* + - Web Driver
    - Grid

Web Driver используется для автоматического управления браузером.

Для каждого браузера используется собственный драйвер. В основном модуль используется для автоматизированного тестирования Frontend части веб версии клиентского приложения. Например, для браузера Google Chrome драйвер поддерживает корпорация Google, что скорее является правилом, чем исключением (на рисунке 2.31 представлена архитектура Web Driver).



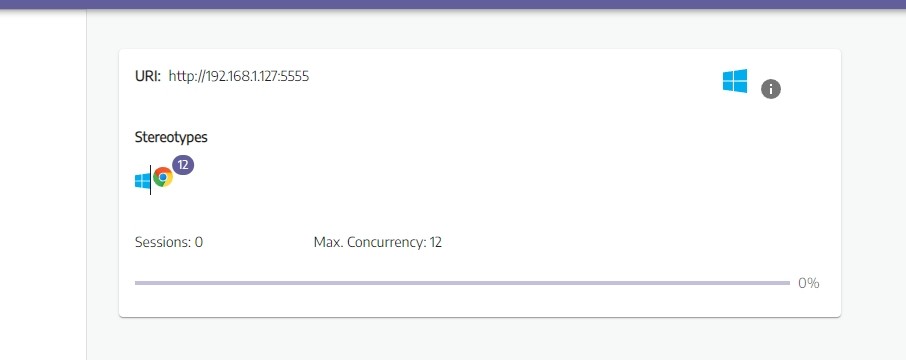
*Рис. 2.33 Архитектура Selenium Web Driver*

В данной работе используется для формирования ссылки на Яндекс.Карты с заданным адресом или маршрутом. При обычном использовании Web Driver запускает встроенный сервер занимается менеджментом очереди заявок на управление браузером. По умолчанию использует динамический порт.

Для более тонких настроек сервера и, например настройки статического порта используется модуль Grid, а также с отдельным

сервером можно связываться удаленно.

Grid – это модуль сервера который используется различными модулями и библиотеками Selenium в этом отношении. Может использоваться в нескольких вариантах, например, в качестве основного Сервера, где создаются ноды на подобии Docker. Так и в качестве этой ноды. В основном Grid рекомендуется использовать при высоких нагрузках и количестве пользователей, что дает задел на будущее масштабирования проекта.



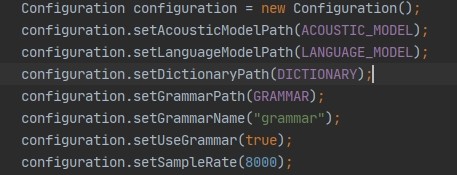
*Рис. 2.34 Пример ноды Selenium Grid на запущенном Selenium Grid хабе.*

#### Sphinx4

Библиотека уже была в основном рассмотрена в предыдущих частях работы. Современная версия библиотеки имеет версию 4-5prealpha.

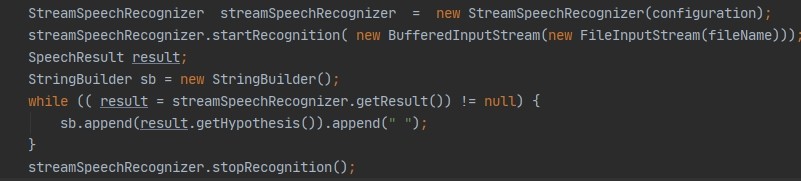
Основные места работы с библиотекой показаны на следующих скриншотах:

1. Конфигурация распознавателя речи, устанавливается акустическая модель, языковая модель, грамматика, частота дискретизации (рис 9);



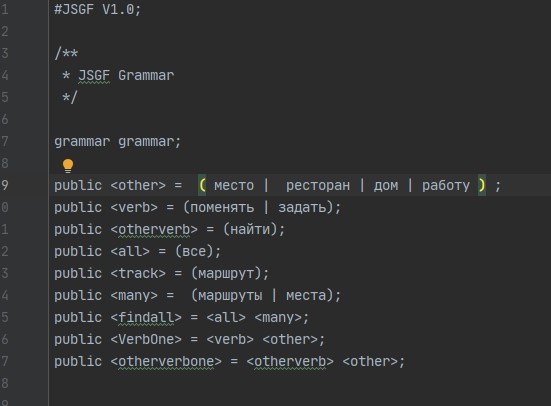
*Рис. 2.35 Настройка Конфигурации распознователя речи*

1. Использование распознавателя речи (рис. 10);



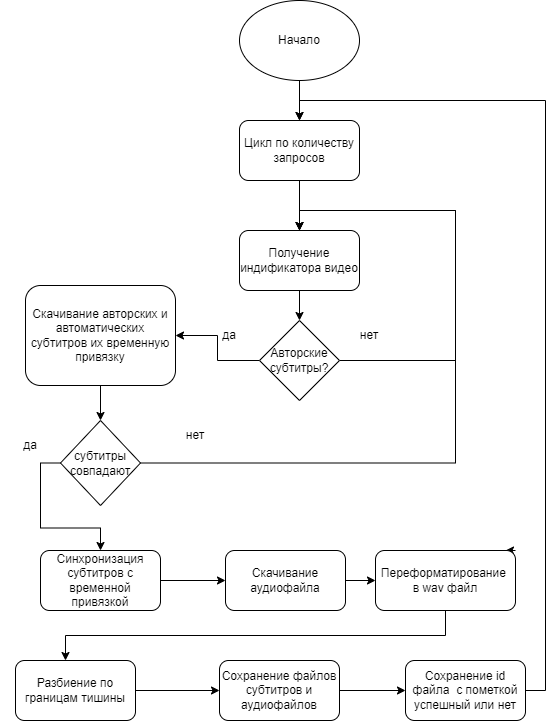
*Рис. 1.36 Использование распознователя речи*

1. Файл установленной грамматики (рис. 11);



*Рис. 2.37 Файл грамматики*

#### 2.5 Сбор тестовых данных для сравнения акустических систем



*Рис. 2.38 алгоритм поиска*

Начальное получение тестовых данных проходит в формате нахождения видео при использовании YouTube api где файлы имеют авторские субтитры так как в большинстве и за большим исключением, случаев автор не старается испортить субтитры под своим видео.

Создается файл с целым набором запросов по интересующий тематике. Этот пакет заполняется каждым вопросом вручную. Сервис WordStat – сервис российской компании Яндекс, позволил получить всевозможные вопросы касательно какой-либо темы и позволил

автоматизировать процесс составления запросов.

После этапа поиска тематических видео по автоматически составленным с помощью сервиса запросам, полученные в результате поиска ссылки на тематические видео, данные ссылки сохраняются в виде списка для следующего далее получение аудио и текстовой информации из видео.

Далее для каждого из видео из полученного списка производится попытка скачивания автоматически распознанных субтитров YouTube в формате характерном для Youtube. Для получения данных по собранным в результате поиска произошедшего ранее авторских текстовых сопровождений и аудио части видеоряда с YouTube было применено приложение youtube-dl. В автоматических создаваемых Youtube субтитрах отмечены временные параметры начала слов обнаруженных YouTube при их создании.

С помощью скрипта реализованного на языке программирования python была произведена синхронизация слов и их временных промежутков.

Следующим шагом мы выгружаем интересующие нас данные, а именно найденные на предыдущих этапах текстовые данные или по- другому называемые субтитрами. Далее производится точный поиск соответствия слов среди авторских субтитров и среди автоматически сгенерированных текстовыми данных.

Такой поиск будет иметь достаточно высокий процент отсева данных так как не все слова могли быть распознаны в результате генерации текстовой информации и автор видео, мог вставить субтитры не соответствующие аудио информации видео Если же соответствие найдено, то мы получаем более точные, чем в авторских субтитрах, отметки времени начала и конца речи, а также уверенность в том, что в аудио файле присутствует определенный текст. Данные о временных рамках начала и конца речи полученных в результате подобного поиска

будут на порядок точнее, также поиск провел дублирующее подтверждение наличия речи на полученном аудио файле.

Данные о времени начала и конца аудио сегмента несущего речевую информацию очень важно для корректного соотношения между речью и ее фонетической транскрипции так как при ошибочных данных могут обнаружится полуслова так как другая их половина не попала в временной промежуток или попадать фоновая речь которая не была отображена соотносящемуся к сегменту аудио тексту.

Если хоть одно соответствие было найдено, то производится загрузка аудиофайла и его разбиение. Когда мы получили хоть один положительный результат поиска производится скачивание аудиофайла в формате vvt и его дальнейшее дробление.

. Для проскочивших c некорректных отметок времени начала и конца производится смещения до ближайшего сегмента тишины в пределах полу секунды. Для нахождения в аудиофайлах сегментов тишины используется Voice Activity Detector.

Для каждого файла создается отдельная директория, содержащая отдельные директории с wav файлами, сконвертированными с помощью FFmpeg и файл с форматом transcriptions.

Файлы делятся на два типа успешно прошедшие обработку и собранные для тестирования, и завершившие обработку с ошибкой оба списка сохраняются для того, чтобы написанный скрипт не пытался обрабатывать данные видео повторно.

Данный способ имеет высокий показатель соответствия аудио и его текстовому представлению. Главным минусом является очень высокий процент отсева видео так как если авторские субтитры отличаются от автоматически сгенерированных Youtube данное видео отсеивается.

Скорость у данного подхода является очень низкой, что делает данный метод сбора данных для, например тренировки акустической модели не целесообразным. Сборка тестовых данных происходит со

скорость час реального времени к 3 часам речи, из которых 3% процента попадают в тестовую выборку. В итоге за 24 часа работы алгоритма было собрано для тестовой выборки 2 часа 11 минут аудио по 3 темам: каждая тема примерно составляет 43минуты Природа, Математика, Интервью.

# ГЛАВА 3. Анализ работы и результаты

#### Тестирование и сравнение с конкурентами

Обучение проводилось на наборе высказываний, доступных на сайте, посвященном исследованию распознавания речи [13]. Корпус русской речи состоял из набора аудиозаписей и набора текстовых аннотаций, описывающих произносимые фразы.

Текстовые аннотации были предварительно обработаны, вместо знаков препинания были вставлены квазифонемы шума и созданы фонетические словари, основанные на уникальной лексике.

На основе уникального словарного запаса обучающего набора.

Полученный корпус состоял из 2700 различных высказываний, включая 200 уникальных рассказчиков, в общей сложности 8 ч. Аудио было записано в условиях низкого уровня шума, с частотой дискретизации 8000 Гц, глубина 8 бит, запись велась в монофоническом режиме.

#### Сравнение Акустических моделей и словарей на тестовой выборке.

Для всех

В данной работе для определения эффективности системы распознавания речи и ее используется метрика Word Error Rate(WER)

𝑆 + 𝐷 + 𝐼

𝑊𝑊𝐸𝑅 =

𝑁

* S – количество заменённых слов
* D – количество удалённых слов
* I – количество вставленных слов

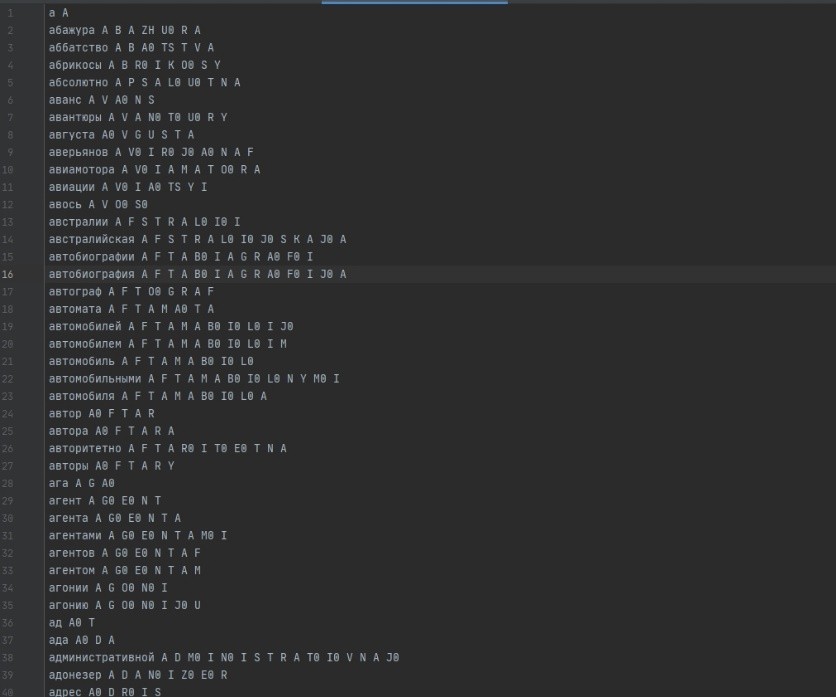
Данный показатель является оценкой качества распознавания речи и чем выше этот показатель, тем хуже произошло распознавание речи.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Мой вариант | russian\_g2p | text2dict | epictran |
| Природа | 49,37% | 49,42% | 48,95% | 50,57% |
| Математика | 36,35% | 36,25% | 38,24% | 37,52% |
| Интервью | 27,25% | 27,27% | 27,79% | 28,46% |

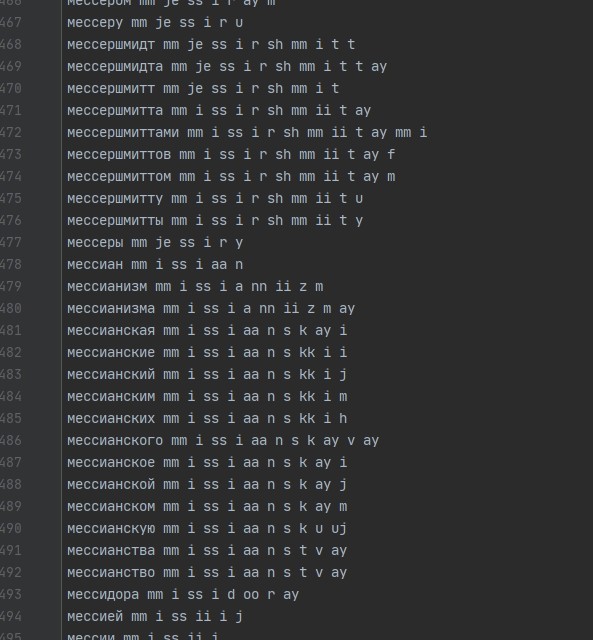
Таблица 3.1 Результаты распознавания разных Акустических моделей

и словарей

Ожидаемо более шумные или переполненные терминами темы показали более низкий результат распознавания. В тематике природы часто могут попадаться видео с открытого воздуха, переполненные посторонними звуками. Математика переполнена терминами что влияет на качество распознавания так как не все термины математики были учтены при создании словаря. Результат работы нейросети немного улучшил показатели изначального алгоритма russian\_g2p.



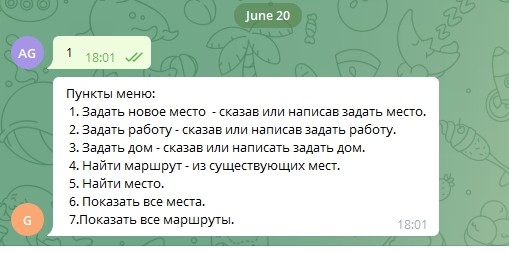
*Рисунок 3.1 Транскрипция, полученная новым методом.*



*Рис. 3.2 Словарь сгенерированный tex2dict*

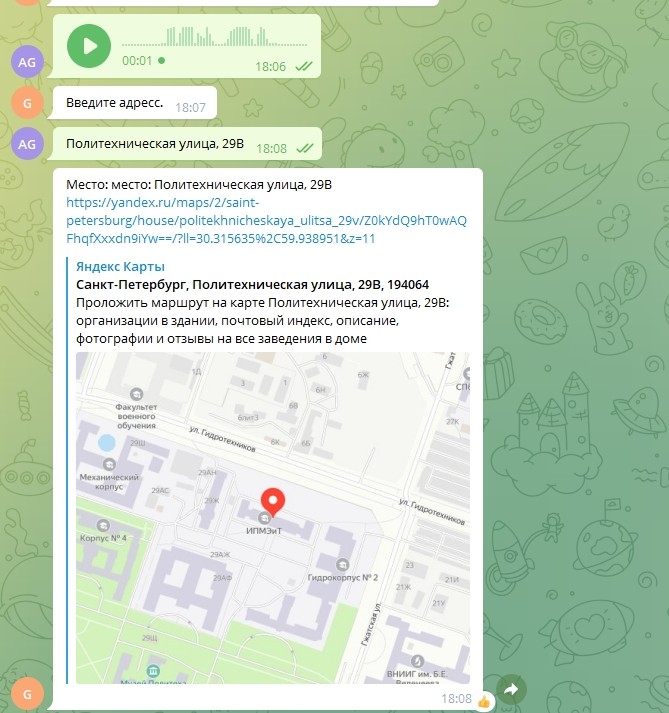
**3.2 Результаты работы Telegram бота**

Начальное сообщение

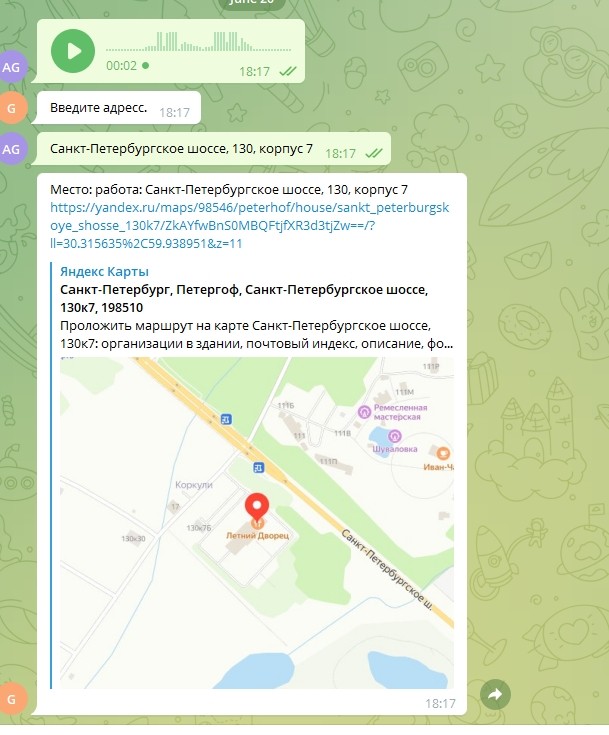


*Рис. 3.3 Первое сообщение и стадия главное меню*

Задать место

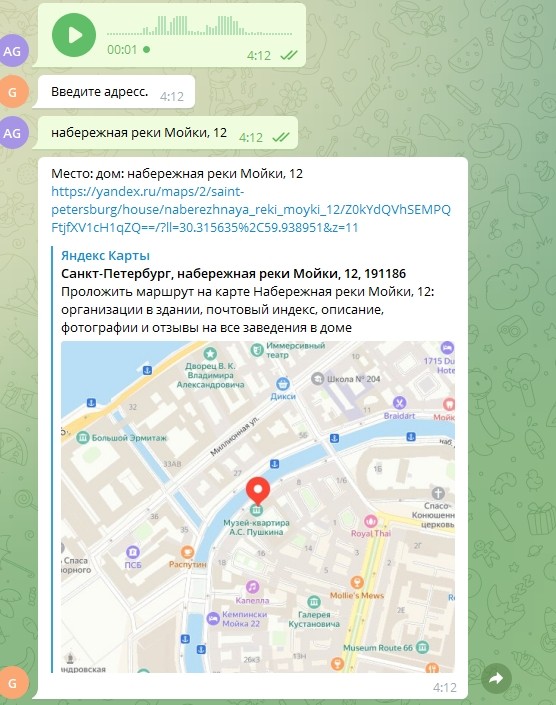


*Рис. 3.3 Результат работы голосового Запроса задать место*

Задать работу

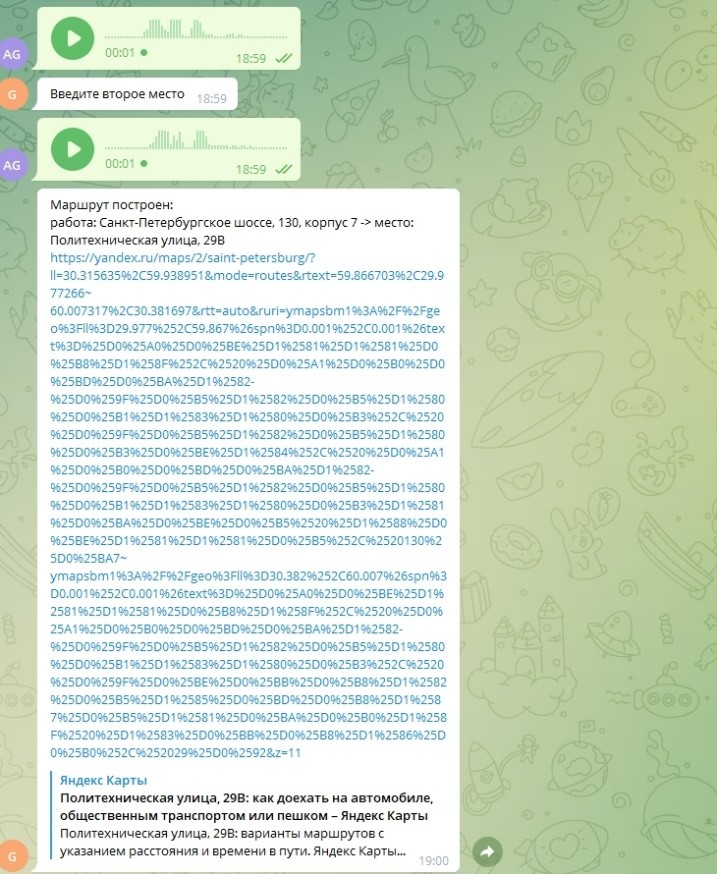
*Рис. 3.4 Заданная работа*

Задать дом



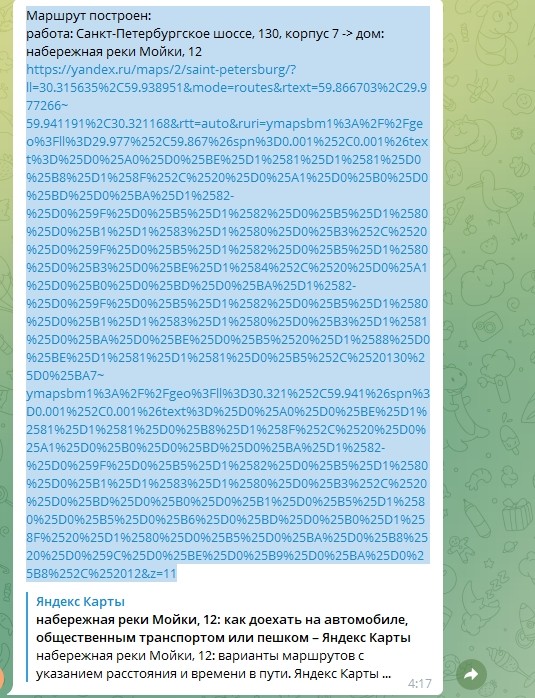
*Рис. 3.5 Заданный дом*

Маршрут место -> работа



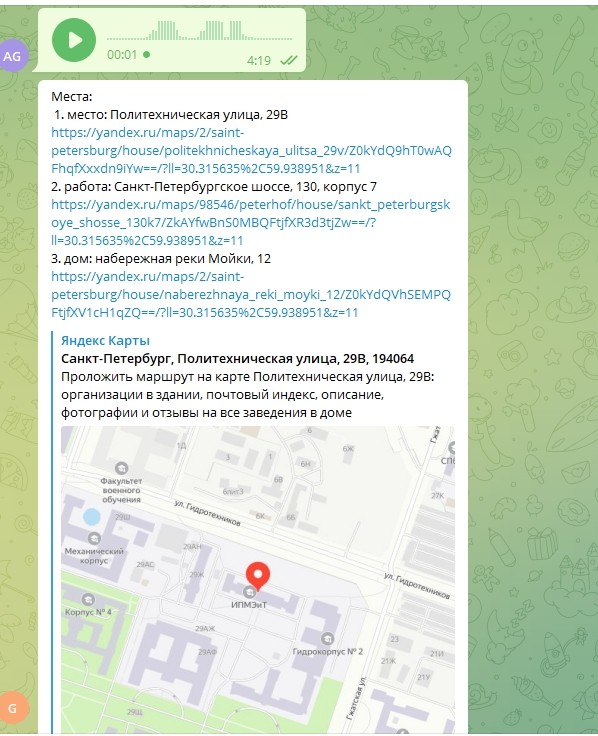
*Рис. 3.6 Маршрут место-> работа*

Маршрут работа -> дом



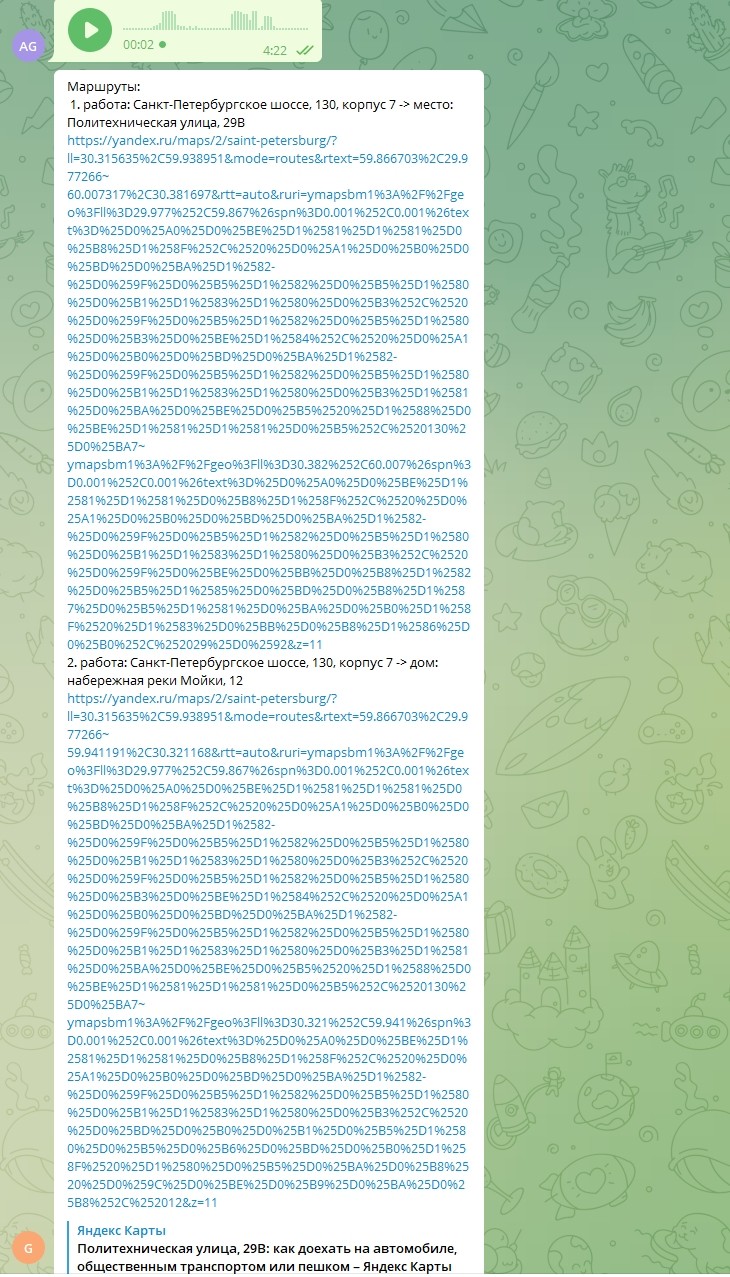
*Рис. 3.7 Маршрут работа -> дом*

Все места



*Рис. 3.8 Все места*

Все маршруты



*Рис. 3.9 Все маршруты*

# Заключение

Возможность подключения разработанного программного продукта к таким мессенджерам, как Telegram, что позволит в будущем Пере использовать разработанную систем для аналогичных систем, таких, как WhatsApp или Viber. Было произведено тестирование данной модели на омонимах, точно отсутствовавших в словаре и поиска слов.

Также был произведён сравнительный анализ с альтернативными акустическими моделями, построенными с помощью других инструментов или по-другому систем, основанных на других алгоритмах транскрибирования, что показало относительно хороший, но не выдающийся показатель точности распознавания речи. Дальнейшее улучшение алгоритма транскрибирования и акцентуации добавят относительно не высокий показатель улучшения качества распознавания речи. Одним из способов получить более высокие результаты распознавания

* это обучение акустической модели на более современных методах машинного обучения.

Разработанный бот для Telegram, разработанный на языке Java также показать хорошие результаты стабильности и скорости работы, что немало важно для современных стандартов. В ходе его разработки можно выделить лишь проблему с отсутствием хорошей документации, так как основная часть документации преимущественно пишется под API для таких языков программирования, как Python и Go. Однако Java в данном случае больше подходила для выполнения поставленной задачи, и опыт, полученный при разработке более, чем актуален на рынке.

Далее получившаяся на основе транскрипции, полученной улучшенным инструментом, акустическая модель была использована в практическом приложении и смогла выполнить поставленные для нее в этом приложении задачи.

Дальнейшие планы на развитие Telegram бота:

* + Добавление альтернативных карт и формирование адресов и маршрутов на выбор пользователя.
  + Развитие альтернативных функций использования голосового управления, например подключение к апи РЖД и добавлении возможности следить за поездами.

#### Список использованных источников

1. Mana F., Massimino P., Pacchiotti A. Using machine learning techniques for grapheme to phoneme transcription 2001.
2. Yakovenko O. [и др.]. Algorithms for Automatic Accentuation and Transcription of Russian Texts in Speech Recognition Systems 2018.
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. CoRR (2014)
4. Bisani, M., Ney, H.: Joint-sequence models for grapheme-to-phoneme conversion. SpeechCommun (2008)
5. Crammer, K., Kulesza, A., Dredze, M.: Adaptive regularization of weight vectors. Mach. Learn. (2013)
6. Deri, A., Knight, K.: Grapheme-to-phoneme models for almost any language. In: Proceedings of the ACL (2016)
7. Лобанов Б.М., Цирульник Л.И.: Компьютерный синтез и клонирование голоса. Белорусская Наука, Минск (2008)
8. Зализняк А. А. Грамматический словарь русского языка: Лексика. -

Русский язык, 1977.

1. Аванесов 1984 — Аванесов Р. И. Русское литературное произношение. 6-е изд. М., 1984.
2. Борунова С.Н., Воронцова В.Л., Еськова Н.А. Орфоэпический словарь русского языка: произношение, ударение, грамматические формы: около 65 000 слов // Русский язык. - 1989.
3. Кипяткова И. С., Карпов А. А. Модуль фонемной транскрипции для системы распознавания разговорной русской речи //Искусственный интеллект. - 2008. - С. 747-757.
4. Кривнова, О.Ф., Захаров, Л.М., Строкин, Г.С.: Автоматический транскрибатор русских текстов: проблемы, структура и применение. In: Труды Международной конференции-практикума "Речь и компьютер" (2001). 12.Voxforge — URL: [http://voxforge.org](http://voxforge.org/) (дата обращения 22.06.22) [Электронный ресурс].
5. CMUsphinx - URL: https://cmusphinx.github.io/ (дата обращения 22.06.22) [Электронный ресурс].
6. tex2dict — URL: https://github.com/zamiron/ru4sphinx/tree/master/text2dict (дата обращения 22.06.22) [Электронный ресурс].
7. epitran — URL: https://github.com/dmort27/epitran (дата обращения 22.06.22) [Электронный ресурс].
8. russian\_g2p — URL: https://github.com/nsu-ai/russian\_g2p (дата обращения 22.06.22) [Электронный ресурс].