Санкт-Петербургский государственный университет

ЛИШИК Елена Викторовна

Выпускная квалификационная работа

Распознавание речи с использованием алгоритмов глубокого обучения

Уровень образования: Магистратура

Направление *03.04.01 «Прикладные математика и физика»*

Основная образовательная программа *ВМ.5521.2017 «Математические и информационные технологии»*

Научный руководитель:

Доцент кафедры ТСУЭФА,

кандидат физ.-мат. наук

Головкина А. Г.

Рецензент:

руководитель DevOps

подразделения

ООО "Ф-Лайн Софтвер",

кандидат физ.-мат. наук,

Райк А. В.

Санкт-Петербург

2020

Оглавление

Введение

Долгое время после изобретения первых ЭВМ [5] одним из наиболее актуальных вопросов их дальнейшего развития оставался процесс взаимодействия человека с машиной. Сначала это под силу было только людям, обладающим специальными знаниями – программистам. Такие времена продлились до возникновения диалогового интерфейса, когда каждый пользователь мог самостоятельно адресовать компьютеру с клавиатуры команду и получать ответ. Разработка и последующее внедрение графического интерфейса, при использовании которого человеку не требовалось специальное знание каких-либо команд, повлияло на широкое распространение персональных компьютеров.

Тем не менее, человечество всегда стремилось к более простому взаимодействию с техникой, потому что естественное, языковое общение представляется нам самым удобным и привычным. Главная задача речевого интерфейса заключается в понимании человеческой речи и правильном на неё реагировании. Таким образом, необходимо обучить машину понимать без посредника язык, на котором общаются люди, то есть создать алгоритм распознавания звуковых сигналов речи. Этим и должна заниматься технология распознавания речи.

В наши дни любой человек использует такой интерфейс в своей повседневной жизни. Обычным примером являются такие приложения как голосовое управление рабочим столом, разнообразные автомобильные устройства, даже «умные» дома. Также в нашем ежедневном обиходе используются речевые транскрипции, субтитры, переводы и изучение языка, голосовые поиск и помощники.

В данной работе сделан акцент не только на распознавании речи, в частности отдельных речевых команд, но и на их последующем преобразовании в текст, т.к. в современном мире человек все чаще использует для общения голосовые сообщения, что является более быстрой формой

коммуникации. Тем не менее, прослушать ответ в таком формате не всегда представляется возможным и удобным. Это делает разработку системы преобразования речи в текст, интегрированной в популярные мессенджеры **актуальной** и востребованной задачей.

Данная система состоит из трех основных компонент: (1) сервера для обработки и хранения данных [36, 37], (2) модуля, отвечающего непосредственно за распознавание речи [3, 9, 42], (3) удобного интерфейса для взаимодействия пользователя мессенджера (на примере Telegram) и блока распознавания.

Первый компонент (сервер 1) преобразует аудиодорожку в необходимый формат [31], сохраняет данные и отправляет преобразованный файл в следующий модуль (2) – сервис распознавания речи. На данном этапе происходит конвертирование голосового сообщения в текстовое. Последний компонент (3) является связующим элементом между пользовательским интерфейсом и созданным сервисом преобразования. Разделение приложения на 2 отдельных модуля является необходимым, поскольку позволяет интегрировать сервис распознавания в другие мессенджеры и социальные сети (как популярные, так и созданные в рамках одной компании) без временного отключения сервиса в уже существующих проектах.

Создание и настройка взаимодействия подсистем (1)-(3) между собой, а также выбор и реализация соответствующих технологий и алгоритмов, учитывающих особенности последующего использования данного сервиса, представляют собой основную цель данной работы.

В представленной работе исследуются известные на сегодняшний день методы и технологии распознавания речи [2, 5, 46, 50], а также рассматриваются существующие программные и технические решения. В частности, в разделе, посвященном обзору литературы, рассматриваются различные методы решения поставленных задач на каждом из этапов разработки и обосновывается их выбор. Первая глава содержит краткое описание истории развития технологии распознавания речи. Вторая глава

посвящена описанию принципов работы серверного приложения и его реализации (подсистемы (1)). В третьей главе проведен сравнительный анализ основных систем распознавания речи, и как следствие сделан выбор в пользу сервиса Sphinx [12, 15]: описан его математический аппарат [18, 24, 26-27] и поэтапно изложены шаги интеграции данного сервиса в разработанное приложение (подсистема (2)). В последней главе описано создание бота в качестве связующего элемента между пользователем и сервисом распознавания (подсистемы (3)).

Постановка задачи

Целью данной работы является разработка сервиса по распознаванию человеческой речи в форме аудиофайлов, и их преобразованию в текстовый формат.

В основе работы лежат группы систем распознавания речи CMUSphinx с их последними обновлениями и компонентами. Для удобства использования и проверки разработанного сервиса была создана и использовалась специальная программа, автоматически выполняющая по заданному алгоритму действия – бот, на платформе приложения Telegram.

Для достижения поставленной цели необходимо решить следующие *задачи*:

1) создать сервер для обработки и хранения данных;
2) настроить систему распознавания речи CMUSphinx:
    a. изучить область и способы применения программ и библиотек;
    b. выбрать метрики качества распознавания речи;
    c. обучить и подготовить к использованию сервером словарь;
    d. интегрировать систему в созданный сервер;
3) настроить VPN подключение между серверами Telegram и сервером обработки данных и распознавания речи для дополнительного шифрования и стабильной работы;
4) создать Telegram-бота, через которого будет происходить запись и отправка аудиофайлов.

Обзор литературы

На каждом из этапов разработки собственного интерфейса по распознаванию речи, а именно с момента изучения принципов работы таких технологий и создания серверного приложения, заканчивая автоматизацией процессов распознавания с помощью системы CMUSphinx, возврата конечного результата пользователю (подсистемы (1)-(3)) был подобран и изучен широкий спектр научной литературы – документации, статей, докладов, рецензий и т.п. [1, 17, 22-23, 27, 35-36, 52].

В основе любого речевого сервиса лежит система автоматического распознавания речи. Для распознавания речи чаще всего используются методы, принадлежащие одной из следующих групп:

1) методы распознавания речи, основанные на применении искусственных нейронных сетей [6, 18, 21, 24, 33, 56];

2) методы, основанные на применении скрытых марковских моделей [3, 18, 26, 45, 53, 55].

Например, в данной статье [18] рассматриваются основные теоретические сведения о скрытой марковской модели, обозначаются необходимые параметры, также описывается метод Баума–Велша, направленного на уточнение параметров модели, и метод Витерби для подбора наиболее вероятной последовательности состояний системы. Более подробно описание этих алгоритмов изложено в работах [19] и [23].

Достаточное количество исследований посвящено также описанию искусственных нейронных сетей и их применению для распознавания речевых команд [32, 34, 49].

Каждый из описанных выше методов имеет свои преимущества и недостатки. Например, скрытые марковские модели эффективно моделируют как временные, так и спектральные вариации речевого сигнала, располагают мощными алгоритмами, обеспечивающими эффективное распознавание отдельных слов и слитной речи. В то же время они являются моделью первого

порядка, т.е. их текущее состояние зависит только от предыдущего. В свою очередь, нейросетевые методы позволяют повысить скорость распознавания за счет распараллеливания вычислений и решить проблему увеличения количества связей между нейронами, затрат памяти, времени на обучение и функционирование сети при расширении словаря распознаваемых слов, несмотря на то что они не имеют механизмов, которые бы адекватно представляли временную вариативность и последовательную природу речевого сигнала.

Для распознавания речи существует ряд систем с открытым исходным кодом: CMUSphnix [12, 15], Mozilla [13], Julius [20], Kaldi [25]. Данные системы показывают достаточно высокий уровень качества и скорости распознавания речи, однако Sphinx имеет ряд преимуществ для его интеграции в реализованный сервис: поддержка русского языка, высокая скорость распознавания, наличие библиотек для Java. Кроме того, в основе данной системы лежит комбинация марковских моделей и нейросетевых методов, что позволяет использовать преимущества обоих подходов. Таким образом, в качестве основного инструментария для разработки собственного интерфейса распознавания речи была выбрана система Sphinx.

Эта система уже поставляется с несколькими высококачественными акустическими моделями с поддержкой разных языков. Помимо моделей, Sphinx также предоставляет как подходы к адаптации, которых достаточно в большинстве случаев, так и возможность обучения акустической модели. В зависимости от требований пользователя можно либо обучить акустическую модель с помощью Sphinxtrain, либо, как более быстрый и легкий вариант, сделать адаптацию уже обученной модели - расширить словарь с помощью инструмента g2p, основанного на нейронных сетях, реализованных в среде Tensorflow [47].

Несомненно, что речь – это непростое комплексное явление и во время записи звуковой дорожки многое может оказать влияние на ее последующее распознавание: акцент, темп произношения, фоновые звуки – шумы и т.п. Для

того, чтобы минимизировать ошибки в процессе распознавания в работе использовался набор программ и библиотек FFmpeg, который не только преобразовывает форматы аудиодорожки, но и может обеспечить эффективное шумоподавление. Официальная документация FFmpeg дает достаточно информации об имеющихся фильтрах и работе с ними [31].

Для упрощения работы с новым интерфейсом и обменом информацией было принято решение создать бота, который бы занимался отправкой полученных им аудиосообщений на сервер с последующим возвращением пользователю текстового варианта. Бот был создан на платформе популярного мессенджера Telegram. Для бесперебойной работы программы и дополнительного шифрования передачи данных используется защищенное интернет-подключение VPN [38, 40].

Для оптимизации работы интерфейса проводится тестирование с целью проверки промежуточных результатов. Для численной оценки точности распознавания используются следующие метрики качества – WER (Word Error Rate), отображающая точность распознавания речи, SER (Sentence Error Rate), показывающая точность системы распознавания, SF (Speed Factor), устанавливающая скорость распознавания системы [11]. Подробное описание оценки работы системы распознавания речи описаны в работе [10] и проанализированы в Главе 2.

Как отмечается в статье [1], в настоящее время существует ряд различных технических средств, которые могут воспринимать и распознавать произносимые человеком речевые обращения: компьютеры, медицинское электронное оборудование, автомобили и самое популярное - смартфоны. Однако не все устройства и программы могут преобразовывать полученную информацию в текстовый формат. Те программы, которые обладают такими свойствами, имеют как ряд преимуществ, так и недостатков. Как правило, в таких популярных приложениях как Yandex Speech [14], полностью отсутствует пользовательский контроль, т.е. качество распознавания речи напрямую зависит от используемой модели языка, о возможностях которой мы

не знаем (например, значение слов может зависеть от контекста), при большом объеме обрабатываемой информации сервис становится платным, а содержание сообщений может потерять приватность.

В настоящее время практически отсутствуют надежные бесплатные сервисы распознавания речи для коммерческого использования, интегрированные в популярные мессенджеры и предоставляющие пользователям возможность отправки и пересылки аудио сообщений для получения их текстового эквивалента. Этот факт подтверждает актуальность темы выпускной квалификационной работы.

Глава 1. История технологии распознавания речи

Человек в среднем произносит ~150 слов за 60 секунд, что как минимум в три раза больше того количества слов, которые он бы смог написать за то же время. В связи с этим, интерес к распознаванию речи существовал всегда. Однако данная технология стала неотъемлемой частью жизни современного человека совсем недавно, с публичным анонсированием открытий в этой сфере от ведущих гигантов мирового рынка [2, 12-15, 20, 25].

В настоящее время большинство современных устройств наделено функцией голосового управления, а также широкое распространение приобретают различного рода digital-помощники. Однако предпосылки для развития этих технологий были сформированы еще в XVIII веке в связи с открытием в 1788 году Вольфганга фон Кемпелена (австро-венгерский ученый-изобретатель), который создал «говорящее» устройство, состоящее из

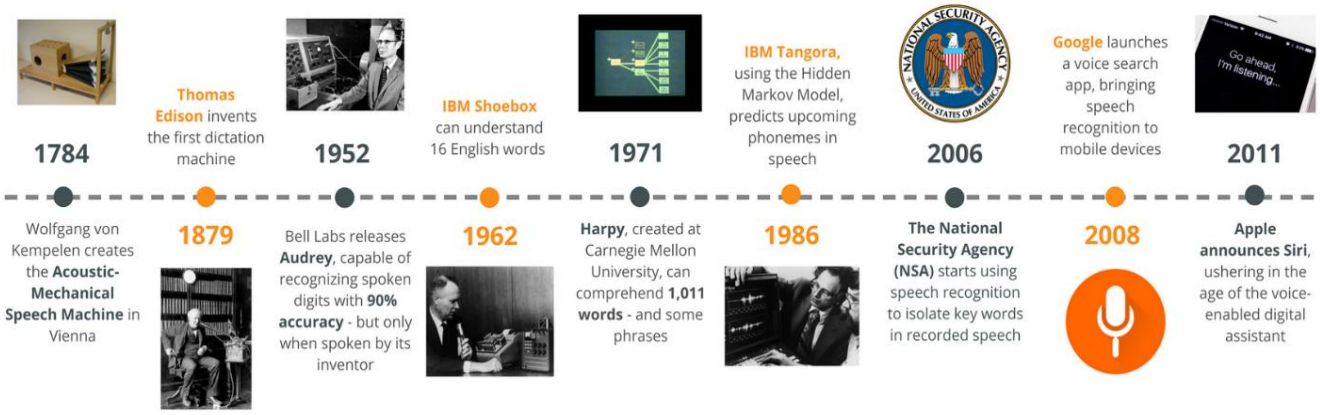


Рис. 1 Хронология развития систем распознавания речи

четырёхугольного деревянного ящика, размерами примерно 1 м на 0,5 м, снабженного мехами и довольно сложной системой клапанов, штифтиков и т. п. Эта машина воспроизводила голос маленького трех-четырех-летнего ребёнка. Позднее, в 1828 году это устройство было усовершенствовало механиком Пошем в Берлине.

В конце XVIIII века обретают популярность диктофоны, изобретателем которых считается американец Томас Эдисон. Эти устройства могли

записывать речь и использовались в основном секретарями и врачами, в обязанности которых входило делать большое количество записей ежедневно.

1952 год можно назвать годом рождения систем распознавания речи. Объяснить это можно прорывом американской компании Bell Laboratories, которая является крупным исследователем в сфере электронных и компьютерных систем. Bell Labs создает машину «Одри» *(от англ. «Audrey»)*, которая может распознать цифры от 0 до 9 с точностью до 90% [51]. Весьма интересным является тот факт, что такой высокий уровень точности регистрировали только в том случае, когда с машиной «говорил» ее изобретатель; процент точности распознавания варьировался от 70% до 80%, когда к «Одри» обращались и другие люди. Принимая во внимание тот факт, что распознавание естественного языка – это трудоёмкий процесс, при создании первых систем распознавания речи исследователи фокусировались на понимании и распознавании только цифр.

Спустя 10 лет, в 1962 году, компания IBM (*от англ. International Business Machines*) продемонстрировала систему "Shoebox", которая распознавала до 16 слов на английском языке. А IBM Tangora [2], выпущенная в середине 1980-х годов и носящая имя в честь Альберта Тангора, ставшая самой быстрой машинисткой в мире, могла приспосабливаться к голосу говорящего. Процесс распознавания требовал медленной, четкой и внятной речи, а также изоляцию от любого фонового шума, но при использовании скрытых моделей Маркова (основой такой модели является простое исследование данных и возможная степень ограничения возникающих отклонений) стало возможным повысить гибкость благодаря кластеризации данных и прогнозу следующих фонем на основе предыдущих паттернов. Несмотря на то, что для любого пользователя требовалось порядка двадцати минут обучения (в форме записанной речи), Tangora могла распознать около двадцати тысяч английских слов и даже некоторые полные предложения.

Александр Вайбель смог разработать устройство в Университете Карнеги-Меллона, которое могло понимать более тысячи слов, построенных по этому принципу.

В тот же самый период времени лаборатории в США, Японии, Англии и СССР смогли разработать еще несколько устройств, которые распознавали отдельные произнесенные звуки, улучшив технологию распознавания речи до системы поддержки четырех гласных и девяти согласных звуков. Система, конечно, была несовершенна, но именно эти первые попытки дали впечатляющий старт, особенно если не забывать, что компьютеры того времени были довольно примитивными.

При распознавании речи существует ряд сложностей, которые могут оказать существенное влияние на качество получаемого результата, например, различие голосов говорящих людей, непоследовательность разговорной речи. Кроме того, фонограмма одних и тех же слов может сильно варьироваться в зависимости от ряда факторов: скорости произношения, регионального диалекта языка, иностранного акцента, социального класса и даже пола человека. Таким образом, проблема масштабирования системы распознавания все время являлась значительным препятствием.

Только на пороге XXI века в 1997 году был анонсирован первый в мире «непрерывный распознаватель речи» (что означало отсутствие необходимости делать паузу между каждым произносимым словом) в виде разработанного программного обеспечения Dragon's Naturally Speaking [28]. Оно могло понимать около ста слов в минуту, и по сей день находит применение (уже в усовершенствованной форме), пользуясь спросом у докторов.

Положенное в те времена начало технологиям распознавания человеческой речи является одним из самых серьезных и важных этапов развития в этой сфере. До тех пор имела место убежденность, что распознавание речи можно достичь только лишь путем адаптации к уникальному способу общения каждого конкретного человека, но добиться такого результата было очень нелегко.

Глава 2. Серверное приложение

Первым основным компонентом реализованного комплекса распознавания речи является серверное приложение для распознавания (подсистема (1)), которое на входе получает аудиофайл, обрабатывает его и возвращает текстовый эквивалент.

Серверное приложение (далее - сервер) – это программа, принимающая и обрабатывающая HTTP запросы клиентов и дающая ответы на эти запросы (HTTP - ответы). Серверные приложения предназначены для того, чтобы разделить нагрузку (функционал) между клиентским и серверным приложением, таким образом обеспечив работу приложения на более «слабом железе» (Рис. 2).

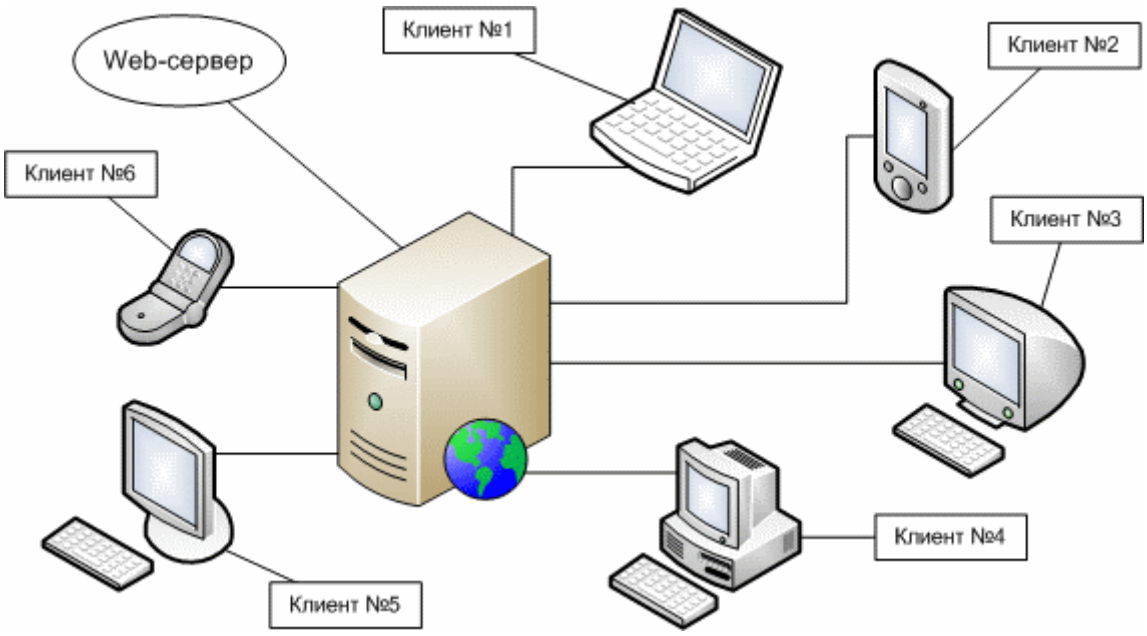


Рис. 2 Схема работы между сервером и клиентом

2.1. Основные типы запросов HTTP протокола

HTTP (*от англ. HyperText Transfer Protocol* (протокол передачи гипертекста) — это протокол прикладного уровня передачи произвольных данных [36].

Каждое сообщение включает в себя три части, передаваемые в указанном порядке:

1. Стартовая строка (Starting line) — необходима для определения типа сообщения;
2. Заголовки (Headers) — необходимы для характеризации тела сообщения, параметров передачи и др.;
3. Тело сообщения (Message Body) — содержит сами данные сообщения. (Тело сообщения может отсутствовать, но стартовая строка и заголовок являются обязательными элементами).

Рассмотрим эти части:

Стартовая строка различается для запроса и ответа. Для запроса необходимо указать метод, URL ((Uniform Resource Locator) путь к документу) и версия. Пример такого запроса: *GET http://developer.mozilla.org/en-US/docs/Web/HTTP/Message HTTP/1.1*.

Метод определяет какое действие необходимо выполнить для данного ресурса. Рассмотрим существующие методы и их определение [37]:

- Метод GET запрашивает представление ресурса. Запросы с таким методом используются только для извлечения данных.

- Метод POST используется для отправки сущностей к определённому ресурсу. В нашем случае так будут отправляться аудио дорожки.

- Метод HEAD в какой-то степени идентичен методу GET, он применяется, например, чтобы выяснить, существует ли в сети тот или иной URL и не произошло ли каких-нибудь изменений.

- Метод OPTIONS используется для получения информации о доступных параметрах соединения с ресурсом.

- Методы PUT и PATCH: PUT схож с GET и служит для передачи на сервер различного рода данных, но является, так сказать, менее продвинутым. В основном он используется для обновления информации, когда не

предполагается работа с большим объемом данных. В некоторых случаях может быть использован PATCH -метод с ещё более «урезанными» возможностями, который работает только с частью ресурса.

- Методы LINK и UNLINK помогают устанавливать и разрывать соединение между ресурсами в сети.
- Метод TRACE помогает отслеживать, что добавляется в запросы промежуточными серверами.
- Метод CONNECT используется для создания TCP/IP тоннелей для установки защищённого соединения (SSL).
- Метод DELETE удаляет указанный ресурс.

2.2. Реализация веб-сервера для распознавания аудио данных

В данной работе активно используются первые два метода – GET (запрос возвращает документацию для интегрирования новых сервисов), и POST («занимается» непосредственно преобразованием – получает аудиофайл в теле сообщения в нужном формате (см. далее Глава 3), возвращает текстовый эквивалент).

Рассмотрим сервер, представленный в данной работе. За основу был взят веб-сервер Undertow [57]. Выбор обусловлен его малым весом (менее 1 Мб), а также простотой запуска и настройки. Для приложений, состоящих из модулей, также удобно использовать мультипроектный подход Gradle [58], так как каждый модуль, расположенный в своей директории, может включать дочерние, которые будут расположены во вложенных каталогах. В корневой директории проекта есть *build.gradle*, который настраивает себя и дочерние модули. Описание мультипроектной структуры выполняется в *settings.gradle* - корневой директории проекта. Например, так в среде разработки IntelliJ IDEA с помощью системы автоматической сборки

Gradle подключаются необходимые зависимости (Рис. 3): *compile group: 'io.undertow', name: 'undertow-core', version: '2.0.22.Final'*.

```
dependencies {
    compile group: 'io.undertow', name: 'undertow-core', version: '2.0.22.Final'
    compile group: 'edu.cmu.sphinx', name: 'sphinx4-core', version:'5prealpha-SNAPSHOT'
    compile group: 'edu.cmu.sphinx', name: 'sphinx4-data', version:'5prealpha-SNAPSHOT'
    testCompile group: 'junit', name: 'junit', version: '4.12'
    compile group: 'org.slf4j', name:'slf4j-log4j12', version:'1.7.25'
    compile group: 'org.apache.commons', name: 'commons-lang3', version: '3.8.1'
```

Рис.3 Подключение зависимостей

Данный сервер может принимать и обрабатывать два вида запросов: POST и GET).

- GET (http://localhost:4274/read) - запрос который не поддерживает передачу каких-либо параметров, возвращает в качестве ответа строку (описание возможностей).

- POST (http://localhost:4274/audio?token=speech) – в качестве параметра здесь передается token (параметр, подобный авторизации) и в теле запроса передается аудиофайл (в необходимом для работы формате .wav), в качестве ответа отправляется строка (преобразованная аудиодорожка в текст).

Для тестирования данного компонента (API) используется POSTMAN [59] - приложение, предназначенное для построения, отправки и получения запроса любого вида. На Рис. 4 показан пример работы POST-запроса по распознаванию речи (в теле запроса находится запись слова «Привет» (1.wav). В качестве ответа возвращается текстовое сообщение «Привет».

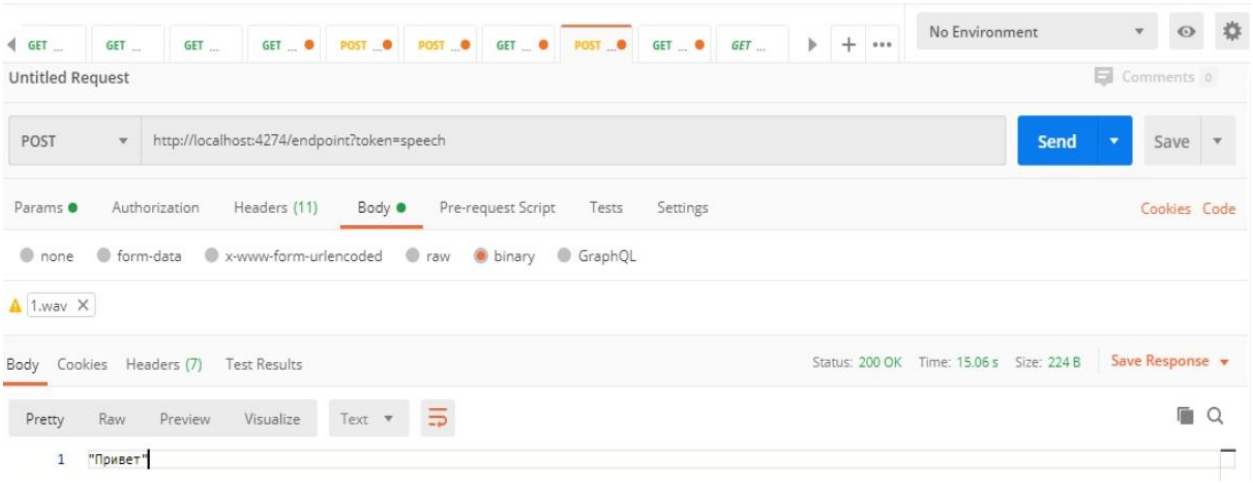


Рис. 4 Основной POST-запрос на сервер распознавания

На Рис. 5 изображена работа GET-запроса: возвращается алгоритм построения POST-запроса.

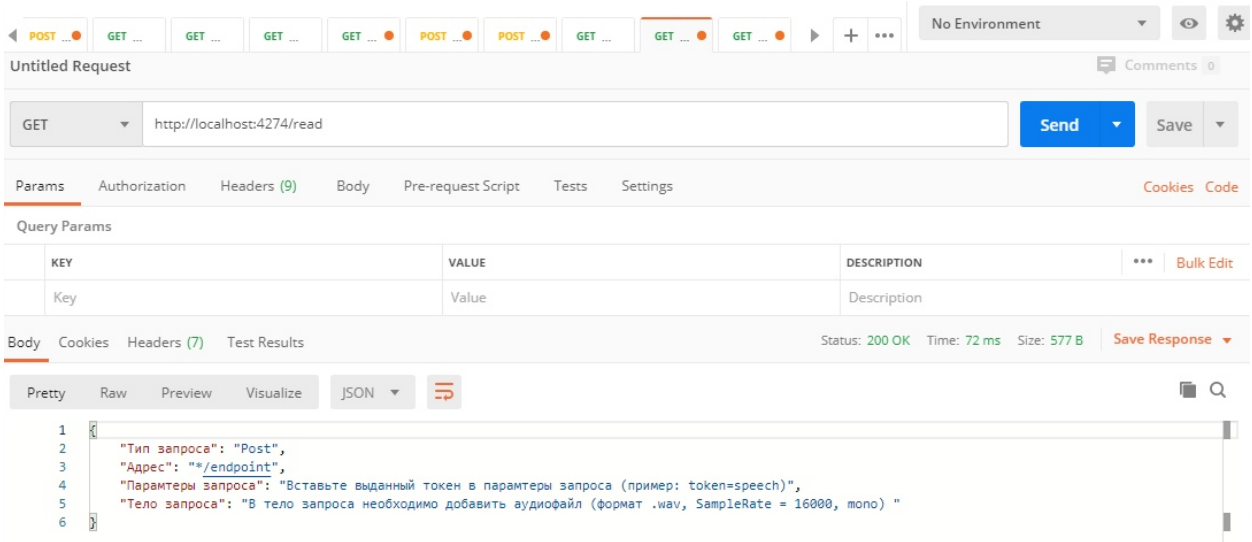


Рис. 5 GET-запрос, который возвращает способ формирования основного запроса

Таким образом, по результатам, полученным в POSTMAN, можно сделать вывод, что серверное приложение исправно функционирует и удовлетворяет необходимым для дальнейшей работы запросам. Дальнейший процесс распознавания речи подробно описан в Главе 3.

Глава 3. Модуль распознавания речи

3.1. Существующие системы распознавания речи

В настоящее время существует достаточное большое количество коммерческих систем распознавания речи, среди наиболее известных можно выделить: Google, Yandex. Siri.

Речевые сервисы, предоставляемые данными компаниями, можно бесплатно использовать в некоммерческих целях, поскольку они имеют открытое API. Качество распознавания речи у таких систем на достаточно высоком уровне, однако они не лишены и ряда недостатков. В первую очередь к ним относится передача данных через сеть интернет, таким образом, при отсутствии подключения система будет непригодна к использованию (на данный момент отсутствует возможность интеграции сервиса в мессенджеры, работающие без выхода в глобальную сеть интернет). Также плохим сигналом могут быть вызваны задержки при обмене данными. Во-вторых, полностью отсутствует пользовательский контроль: качество распознавания речи напрямую будет зависеть от используемой модели языка (например, в разном контексте слова могут принимать разные значения и т.п.). В-третьих, при работе с большим объемом аудиоданных сервисы являются платными. В-четвертых, не гарантируется приватность персональной информации пользователей.

Потеря приватности, обязательный выход в глобальную сеть, закрытый код ставит невозможность интеграции данных систем в закрытые социальные сети и мессенджеры. Таким образом, для решения выше обозначенных проблем разрабатывается сервис по распознаванию речи с использованием систем с открытым кодом. Среди наиболее известных из них следует отметить:

- Mozilla DeepSpeech [13] - архитектура распознавания речи с использованием платформы машинного обучения TensorFlow, в наборе которой предлагаются обученные модели и инструментарий для

распознавания, однако готовая модель поставляется только для английского языка;

- Kaldi [25] предоставляет наибольшее количество современных подходов, таких как использование нейронных сетей, моделей гауссовых смесей и использование конечных автоматов, однако ее последующая интеграция в сторонние приложения является крайне затруднительной;
- Julius [20] - система японского происхождения, в которой этапы акустического и языкового моделирования осуществляются с помощью утилит, входящих в состав CMM; поддерживает английский и японский языки, однако отсутствие хорошей акустической модели для английского языка означает гораздо более низкое качество распознавания и использование внешнего движка для обучения акустической модели подразумевает использование интерфейса только в личных целях;
- CMUSphinx [12, 15] - система реализована на языке программирования Java, что позволяет применять ее на множестве платформ, в том числе под управлением операционной системы Android и iOS, а также облегчает интеграцию в проекты, написанные на Java. Ввиду модульной структуры, появляется возможность быстрого внесения изменений и исправления ошибок.

3.2. Метрики качества распознавания речи

В данном разделе рассматриваются описанные выше системы распознавания речи и проводится их сравнительный анализ для оценки качества распознавания с использованием наиболее распространенных метрик [11]:

WER – это величина (расстояние Левенштейна) вычисляемая не на уровне фонемы, а на уровне слова, отображающая точность распознавания речи. Расстояние Левенштейна – метрика, определяемая как минимальное

количество односимвольных операций (а именно вставки/удаления/замены), необходимых для преобразования одной последовательности символов в другую.

WER = (S + D + I)/N = (S + D + I)/(S + D + C)

- S – количество операций по замене слов
- D – количество операций по удалению слов
- I – количество операций по вставки слов
- C – число распознанных (правильно) слов
- N – всего слов

SER – это метрика, которая показывает точность системы распознавания.

SER = Sв / S

- Sв – число безошибочно распознанных предложений
- S – всего предложений

SF – это метрика, которая показывает скорости распознавания системы.

SF = Tр / T

- Tр – время, которое ушло на распознавания сигнала
- T – длительность сигнала.

Ниже приведена таблица результатов сравнения рассматриваемых систем [10] с точки зрения введенных выше метрик:

	CMUSphinx	**Kaldi**	**DeepSpeech**	**Julius**
WER, %	22.7	6.5	29.9	23.1
SER, %	77.3	83.5	84.0	76.9

SF	1	0.6	2.1	1.3
Язык	C/Java	C++	Python	C
Языки	американский и британский английский, французский, мандаринский, немецкий, голландский и русский	Английский	Английский	Японский, Английский

Таблица 1 Результаты сравнения систем распознавания речи

Анализ показателей, приведенных в Таблице 1, позволяет сделать вывод о том, что наиболее подходящей системой для решения задач данной работы является Sphinx [15, 29]. Кроме того, Sphinx является простой и удобной в использовании, и на ее основе выполнено большое количество научных работ. Среди преимуществ Sphinx следует также отметить: поддержку русского языка, открытый программный код и простоту встраивания его в Java код [17].

3.3. CMUSphinx

CMUSphinx является одной из самых крупных разработок с открытым кодом в области распознавания речи человека. Комплекс включает в себя следующий набор программ и библиотек [15, 29]:

- Pocketsphinx — это программа, принимающая некие акустические модели, грамматику и словарь, а также звуковой поток/звуковой файл/поток с микрофона, и выдающая распознанный текстовый файл. Написана она на языке C, скорость работы достаточно высокая.
- Sphinxbase — одна из библиотек, которая необходимая для

бесперебойной работы Pocketsphinx.

- Sphinxtrain — программа, необходимая для обучения акустических моделей.
- Sphinx4 — библиотека, служащая для распознавания речи, написана на языке программирования Java.

Ниже перечислены основные определения, которыми оперирует Sphinx [15, 20, 29]:

- Словарь — это файл (часто текстовый документ), в котором прописываются и сопоставляются лексемы и фонемы (слово с его транскрипцией). В основном процессе преобразования речи он служит для преобразования фонем, полученных после распознавания акустической моделью, в слова (лексемы).
- Грамматика — это правила, описывающие механизм построения предложений. Полученные на предыдущем шаге лексемы, сопоставляются с грамматикой и выводится результат.
- Языковая модель — это статистическая модель, которая описывает вероятность вхождения слов, а также их комбинаций. Таким образом, распознавание лексем — это попытка максимально возможно получить грамматически и лексически правильную фразу.

Таким образом, можно сделать вывод: чем сложнее сам язык и больше правил слов, тем ниже точность распознавания. В связи с этим для минимизации ошибки и улучшения качества получаемого текста, зачастую имеет смысл использовать минимально достаточный набор инструментов под каждую конкретную задачу (например, не имеет смысла использовать несколько языков в том случае, если распознаваться будет только один).

3.4. Математический аппарат CMUSphinx

Речь – это составное комплексное явление. Не каждый человек до конца понимает, как она производится и расшифровывается. Неправильно полагать,

что речь построена из слов, каждое из которых состоит из фонем. Речь – это динамический процесс без четко разделяемых частей.

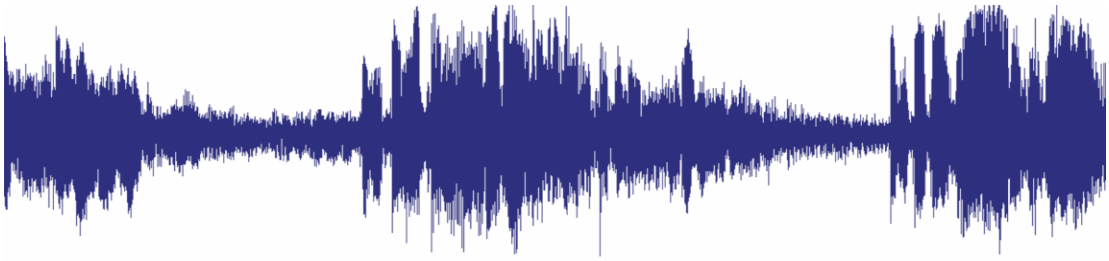


Рис. 6 Вид звуковой волны в редакторе

В современной практике речевая структура понимается как непрерывный аудио поток, в котором довольно стабильные состояния смешиваются с динамически измененными состояниями. В этой последовательности состояний можно определить более или менее похожие классы звуков или фонем. Фонема - это единица звукового конструктора языка [2, 3].

Акустические свойства формы сигнала, соответствующего фонеме, могут сильно различаться в зависимости от многих факторов - контекста, динамика, стиля речи и т.д. Часто можно найти три или более видов фонем различной природы.

Стандартный способ распознавания речи заключается в следующем: за основу исследования берется звуковая дорожка (звук - это не что иное, как волновой формат данных (Рис. 7)), в котором амплитуда изменяется во времени), а поскольку она может быть большой длины, появляется необходимость в ее разбиении на отдельные фрагменты (семплы), которые затем распознаются.

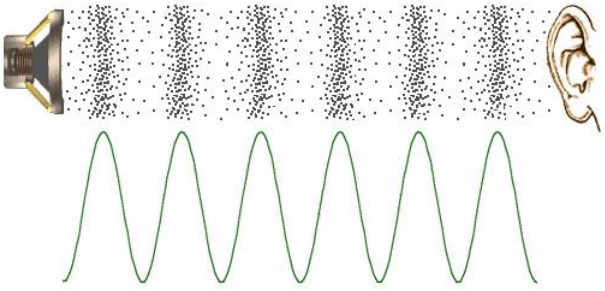


Рис. 7 Наглядный пример изображения звука

Таким образом аудио сигнал разбивается на семплы (как правило продолжительностью по ~ 10 мс), и затем для каждого из них вычисляется вектор признаков [53]. В используемой в данной работе системе CMUSphinx

такой вектор состоит всего из 39 чисел. Состав векторов напрямую зависит от способа извлечения признаков. Поиск наиболее эффективного способа вычисления этих чисел еще находится на стадии активного исследования, но, как правило, в большинстве случаев это производная от спектра.

На следующем этапе распознавания в соответствии со структурой речи полученные векторы анализируются в рамках трех моделей – языковой, акустической и фонетического словаря – для подбора наиболее подходящей комбинации слов.

В качестве математического аппарата, применяемого для распознавания речи в используемом в работе Sphinx, применяются скрытые марковские модели (*от англ. HMM, Hidden Markov model, далее - СММ*) [23, 25-26, 45], поэтому в этом случае «наиболее подходящей комбинацией» будет та, которая является наиболее вероятной. В науке СММ широко используются для усиления обучения и распознавания образов, таких как речь, почерк, жесты, частичные разряды и биоинформатика [27].

В основе данной модели лежит конечный автомат, состоящий из N-скрытых состояний. Переходы между состояниями в каждый дискретный момент времени t не являются определенными, поскольку происходят согласно вероятностному закону и характеризуются матрицей вероятностей переходов. Схематическое изображение диаграммы переходов между состояниями СММ приведено на Рис. 8.

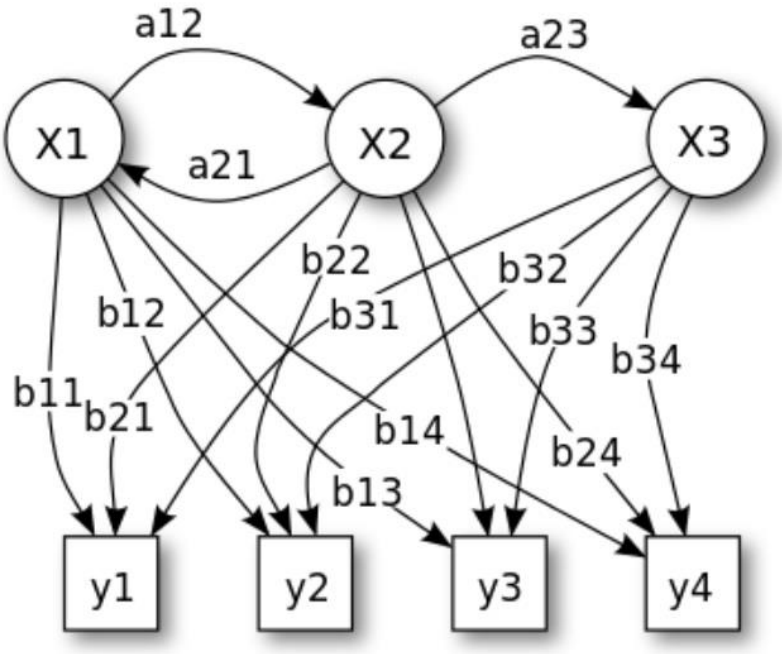


Рис. 8 Схема скрытой марковской модели, где X1, X2, X3 – состояния, y1, y2, y3, y4 – возможные наблюдения, a12, a22, a23 – вероятности перехода в состояние, b11,..,b34 – вероятности выхода.

Процесс работы со СММ, как и с любой другой адаптивной экспертной системой, происходит в несколько этапов:

1. Определение параметров модели с использованием алгоритма Баума-Велша [45].
2. Определение вероятности наблюдаемой последовательности векторов, сгенерированной этой моделью, – через алгоритм Витерби [55] (алгоритм максимума правдоподобия).
3. Обучение СММ.

3.4.1. Алгоритм Баума-Велша

Алгоритм Баума-Велша (прямого-обратного хода) основан на подходе динамического программирования и представляет собой частный случай алгоритма максимизации ожидания. Целью является «подгонка» параметров СММ: матрицы перехода состояний, матрицы излучения, матрицы распределения начальных состояний так, чтобы модель была наиболее близка к наблюдаемым данным.

Начальными данными алгоритма Баума–Велша являются СММ со случайными параметрами $\lambda = (A,B,\pi)$ а также заданная обучающая последовательность наблюдений $O = \{o_1,\dots,o_T\}$. Данный алгоритм необходим для увеличения вероятности $p(O \mid \lambda)$ посредством уточнения случайных параметров.

Алгоритм состоит из следующих шагов:

1. Прямая процедура вычисляет «прямые» переменные — вероятности получения начальной части (до момента времени t) заданной последовательности наблюдений $O = \{o_1,\dots,o_T\}$ с условием начала движения с момента времени 1 и до момента времени t в состоянии s_i:

$$\alpha_t(i) = p(o_1,\dots,o_t, q_t = s_i \mid \lambda).$$

Данная вероятность вычисляется в цикле, с использованием следующего выражения:

$$\alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij},\ 1 \le j \le N,\ 1 \le t \le T-1,$$

где

$$\alpha_1(j) = \pi_j b_j(o_1),\ 1 \le j \le N.$$

Находим:

$$\alpha_T(i),\ 1 \le i \le N.$$

По формуле X получаем значение вероятности:

$$p(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

2. Обратная процедура дополняет прямую процедуру, позволяя найти «обратные» переменные — вероятности получения второй части (с момента времени t+1 до T) заданной последовательности наблюдений $O = \{o_1,\dots,o_T\}$ с условием начала с момента времени t из исходного состояния s_i и окончанием движения в момент времени T:

$$\beta_t(i) = p(o_{t+1},\dots ,o_T \mid q_t = s_i ,\lambda).$$

Данная вероятность вычисляется аналогичным способом с использованием выражения:

$$\beta_t(i) = \sum_{j=l}^{N} \beta_{t+1} a_{ij}\, b_j\,(o_{t+1}),\ 1 \le i \le N,\ 1 \le t \le T - 1,$$

где

$$\beta_T\,(i) = 1,\ 1 \le i \le N.$$

Находим:

$$\beta_1(i),\ 1 \le i \le N.$$

Вычисляем вероятность через «обратные» переменные:

$$p(O \mid \lambda) = \sum_{i=l}^{N} \alpha_1(i)\beta_1(i) = \sum_{i=l}^{N} \pi_i b_i(o_1)\beta_1(i).$$

Используя формулу прямого метода, получим:

$$p(O,q_t = s_i \mid \lambda) = \alpha_t(i)\beta_t(i),\ 1 \le i \le N,\ 1 \le t \le T.$$

Формула вычисления вероятности через «прямые» и «обратные» переменные выглядит следующим образом:

$$p(O \mid \lambda) = \sum_{i=l}^{N} p(O,q_t = s_i \mid \lambda) = \sum_{i=l}^{N} \alpha_t(i)\beta_t(i),\ 1 \le t \le T.$$

3.4.2. Алгоритм Витерби

Алгоритм Витерби — это алгоритм для поиска наиболее подходящего списка состояний (пути Витерби — наиболее правдоподобная последовательность скрытых состояний), который в контексте моделей Маркова находит наиболее вероятную последовательность произошедших событий [23].

Во время распознавания звуковая дорожка воспринимается как последовательность событий, а строка текста является «скрытым смыслом»

акустического сигнала. Таким образом алгоритм Витерби вычисляет наиболее вероятную строку текста по данному звуковому сигналу.

Алгоритм «высказывает» следующие предположения:

- наблюдаемые и скрытые события должны быть последовательностью (последовательность упорядочена по времени);
- все последовательности должны быть выровнены: каждое «видимое» событие соответствует только одному скрытому событию;
- вычисление наиболее вероятной скрытой последовательности до момента t зависит только от наблюдаемого события в момент времени t, и наиболее вероятной последовательности до этого момента.

Рассмотрим основные составляющие алгоритма:

1. Вычисление вспомогательной переменной δ — максимальной вероятности того, что при заданной последовательности наблюдений $O = \{o_1,\dots,o_T\}$ модель в момент времени t (движение от момента времени 1 до t) будет находиться в состоянии s_i:

$$\delta_t(i) = \max_{q1,\dots,qt-1} p(q_1,\dots,q_{t-1},q_t = s_i ,o_1,\dots,o_{t-1} \mid \lambda).$$

Используя следующую формулу, вычисляем вероятности:

$$\delta_{t+1}(j) = b_j(o_{t+1}) \cdot \max_{1\le i\le N} (\delta_t(i) \cdot a_{ij}),\ 1 \le j \le N,\ 1 \le t \le T - 1,$$

где

$$\delta_1(j) = \pi_j\, b_j\,(o_1),\ 1 \le j \le N.$$

Стоит отметить состояние $\psi_t(j)$, при котором $\delta_t(j)$ достигает своего максимума:

$$\psi_{t+1}(j) = \arg\max_{1\le i\le N} (\delta_t(i) \cdot a_{ij}),\ 1 \le j \le N,\ 1 \le t \le T - 1.$$

Таким образом вычисляется:

$$\delta_T(i),\ 1 \le i \le N.$$

2. Вычисление наиболее вероятного конечного состояния системы q_t. После шага T − 1 вычисляется наиболее вероятное конечное состояние системы:

$$q_T = arg\ max_{1 \le i \le N} (\delta_T(i)),$$

где $\max_{1 \le i \le N} (\delta_T(i))$ — наибольшая из максимальных вероятностей нахождения системы в момент времени T в состоянии s_i.

3. Вычисление наиболее вероятной последовательности состояний системы $Q = \{q_1,\dots ,q_T\}$, соответствующей последовательности наблюдений $O = \{o_1,\dots ,o_T\}$ — происходит обратным обходом массива состояний ψ_t (когда вероятности $\delta_t(i)$ достигают максимума), начиная с наиболее вероятного конечного состояния q_T :

$$q_t = \psi_{t+1}(q_{t+1}),\ 1 \le t \le T - 1.$$

3.4.3. Нейронные сети

Искусственная нейронная сеть [18, 32-34] – это реализованная программно или аппаратно математическая модель, описывающая работу человеческого мозга. Базой этих сетей служат нейроны – элементы, которые получают некий импульс и передают выходной сигнал. Нейрон (Рис. 9) состоит из взвешенного сумматора и нелинейного элемента. Функционирование нейрона определяется следующими показателями:

$$NET = \sum_i w_1 x_1$$

$$OUT = F(NET - \theta)$$

где x_i — входные сигналы, совокупность всех входных сигналов нейрона образует вектор x;

w_i — весовые коэффициенты, совокупность весовых коэффициентов образует вектор весов w;

NET — взвешенная сумма входных сигналов, значение NET передается на нелинейный элемент;

θ — пороговый уровень данного нейрона;

F — нелинейная функция, называемая функцией активации.

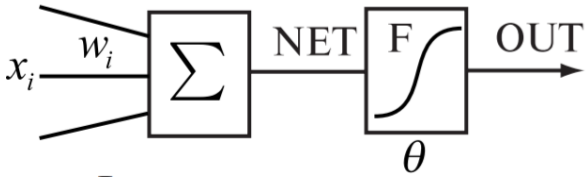


Рис. 9 Принципиальная схема искусственного нейрона

Нейрон имеет несколько входных сигналов x и один выходной сигнал OUT. Параметрами нейрона, определяющими его работу, являются: вектор весов w, пороговый уровень θ и вид функции активации F.

Ниже перечислены наиболее часто используемые функции активации [43, 56].

Логистическая функция (сигмоида, функция Ферми, Рис.10).

$$OUT = \frac{1}{1+e-NET}$$

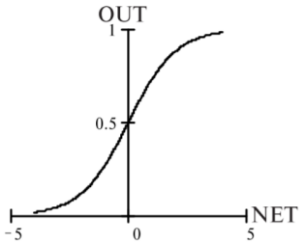


Рис. 10 График логистической функции активации

Применяется очень часто для многослойных персептронов и других сетей с непрерывными сигналами. Гладкость, непрерывность функции — важные положительные качества. Непрерывность первой производной позволяет обучать сеть градиентными методами (например, метод обратного распространения ошибки). Функция симметрична относительно точки (NET=0, OUT=1/2), это делает равноправными значения OUT=0 и OUT=1, что существенно в работе сети. Тем не менее, диапазон выходных значений от 0

до 1 несимметричен, из-за этого обучение значительно замедляется. Данная функция — сжимающая, т.е. для малых значений NET коэффициент передачи K=OUT/ NET велик, для больших значений он снижается. Поэтому диапазон сигналов, с которыми нейрон работает без насыщения, оказывается широким. Значение производной легко выражается через саму функцию. Быстрый расчет производной ускоряет обучение.

Гиперболический тангенс (Рис. 11): OUT = th(NET) $\frac{\mathrm{e\,NET-e-NET}}{\mathrm{eNET+e-NET}}$

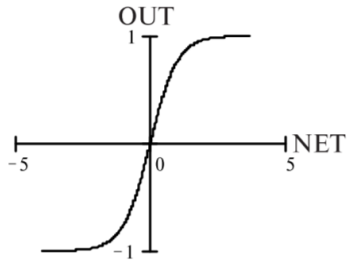


Рис. 11 График гиперболической функции активации

Тоже применяется часто для сетей с непрерывными сигналами. Функция симметрична относительно точки (0,0), это преимущество по сравнению с сигмоидой. Производная также непрерывна и выражается через саму функцию.

Формальные нейроны могут объединяться в сети различным образом. Например, самым распространенным видом сети стал многослойный персептрон, представленный на Рис. 12 [43, 56].

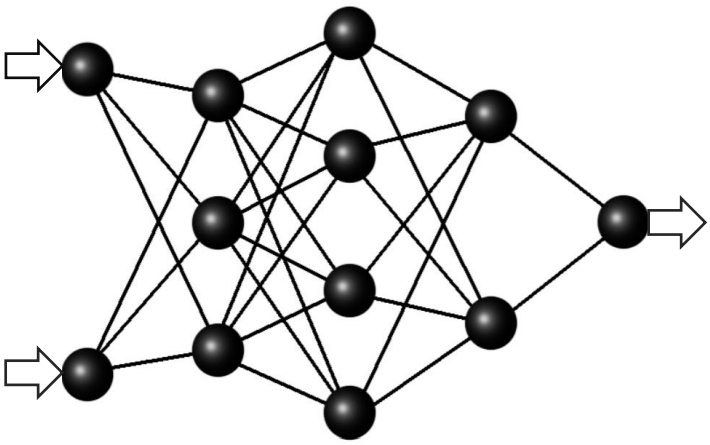


Рис. 12 Структура нейронной сети, где сферы изображают нейроны, прямые линии - взвешенные связи, стрелки показывают направление распространения сигнала

Такая сеть состоит из произвольного количества слоев нейронов. Нейроны каждого слоя соединяются с нейронами предыдущего и последующего слоев по принципу "каждый с каждым". Первый слой называется сенсорным или входным, внутренние слои называются скрытыми или ассоциативными, последний — выходным или результативным. Количество нейронов в слоях может быть произвольным. Обычно во всех скрытых слоях одинаковое количество нейронов.

Каждый слой рассчитывает нелинейное преобразование от линейной комбинации сигналов предыдущего слоя. Линейная функция активации может применяться только для тех моделей сетей, где не требуется последовательное соединение слоев нейронов друг за другом. Для многослойных сетей функция активации должна быть нелинейной, иначе можно построить эквивалентную однослойную сеть, и многослойность оказывается ненужной. Если применена линейная функция активации, то каждый слой будет давать на выходе линейную комбинацию входов. Следующий слой даст линейную комбинацию выходов предыдущего, а это эквивалентно одной линейной комбинации с другими коэффициентами, и может быть реализовано в виде одного слоя нейронов. В многослойном персептроне нет обратных связей. За счет поочередного расчета линейных комбинаций и нелинейных преобразований достигается аппроксимация произвольной многомерной функции при соответствующем выборе параметров сети.

Процесс распознавания речи с использованием нейронных сетей состоит из следующих этапов (Рис. 13):

1. Акустический препроцессор обрабатывает входной речевой сигнал, и определяет последовательность векторов признаков, для каждого отрезка времени (семпл) и состоят из спектральных или кепстральных коэффициентов, характеризующих отрезки речевого сигнала [22, 41, 43].
2. Далее полученные векторы подлежат сравнению с эталонными векторами, содержащимися в моделях слов. На этом же этапе вычисляются локальные метрики или меры соответствия (как правило

сравниваются речевые сегменты, описанные несколькими векторами признаков) [53].

3. С помощью метрик происходит временное выравнивание последовательностей векторов признаков с последовательностями эталонных векторов, образующими модели слов, а также вычисляется мера соответствия для компенсации изменений скорости произнесения. Далее находится максимально соответствующее слово. В случае если распознается непрерывная (слитная) речь, то полученные на втором этапе локальные метрики используются для временного выравнивания и определения мер соответствия для отдельных выражений.

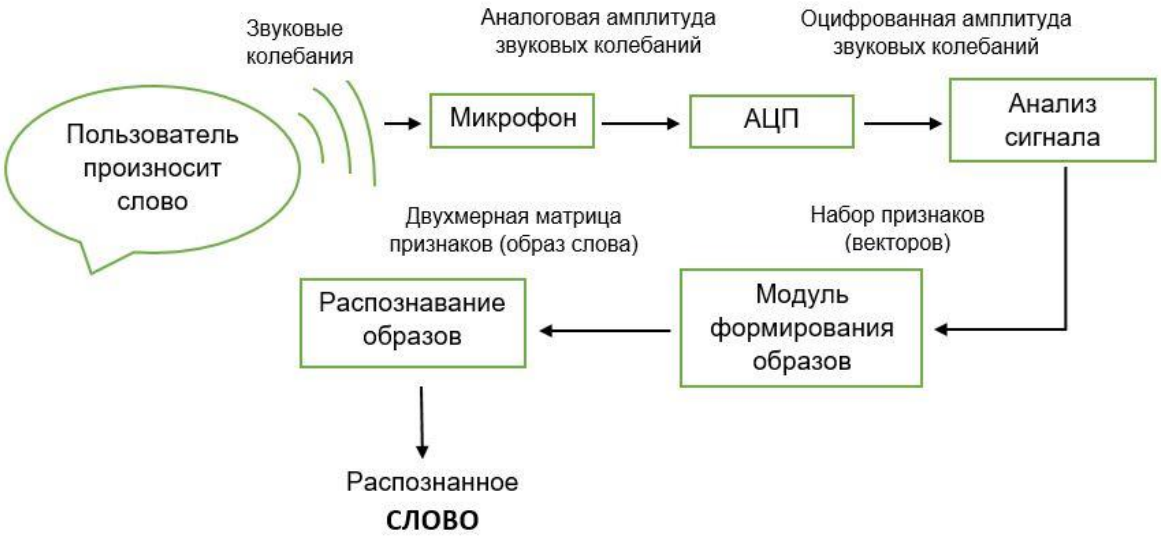


Рис. 13 Схема работы нейронной сети

Принимая во внимание все преимущества использования нейронных сетей для распознавания речи будет правильно отметить, что они не могут быть основным и единственным методом распознавания, ввиду отсутствия механизмов, адекватно представляющих временную вариативность и последовательную природу речевого сигнала; теоретических основ, используемых для вычисления или выбора параметров, определяющих динамику и топологию нейронных сетей (в настоящее время они выбираются на усмотрение разработчика); а также несмотря на постоянное совершенствование алгоритмов для упрощения и ускорения процедуры

обучения использование нейронных сетей до сих пор является довольно ресурсоемким процессом.

3.5. Настройка Sphinx

Проект Sphinx поставляется с несколькими высококачественными акустическими моделями [12, 21, 54]. Существуют русская, английская акустические модели для микрофона и прямой речи, а также модель для речи по телефону. Также в открытом доступе в случае необходимости можно найти французские или китайские модели, обученные на огромном количестве акустических данных. Данные модели были тщательно оптимизированы для достижения наилучшей производительности распознавания и хорошо работают практически во всех приложениях. Большинство командно-контрольных приложений и даже некоторые большие словарные приложения могут напрямую использовать модели по умолчанию.

Помимо моделей, Sphinx предоставляет некоторые подходы к адаптации акустических моделей, которых должно быть достаточно для большинства случаев, когда требуется большая точность распознавания.

Известно, что адаптация хорошо работает, когда используются различные условия записи (ближний или дальний микрофон или телефонный канал), или когда присутствует немного другой акцент или даже другой язык. Адаптация, например, отлично подходит, если необходимо быстро добавить поддержку какого-либо нового языка, просто сопоставив телефонный набор акустической модели с целевым телефонным набором с помощью словаря.

В данной работе произведена адаптация акустической модели и рассмотрен пример создания своей новой модели. Рассмотрим области применения обоих случаев.

Адаптация применяется, если:

- Необходимо повысить точность распознавания;

- Имеется недостаточно данных;
- Не хватает времени и/или опыта.

Создание новой акустической модели актуально, если:

- разрабатывается акустическая модель для нового языка или диалекта;
- есть необходимость в специализированной модели для небольшого словарного запаса;
- создатель располагает достаточным объемом данных для обучения (порядка 50 часов записи 200 ораторов для многих дикторов) и временем для обучения модели и оптимизации параметров (в срок от 1 месяца);
- создатель обладает знаниями о фонетической структуре языка.

3.5.1. Обучение акустической модели с помощью Sphnixtrain

Создание новой акустической модели происходит в несколько этапов. Рассмотрим их по порядку.

<u>I этап</u> – Подготовка данных

Для осуществления настройки обучения сначала необходимо создать базу данных обучения или загрузить существующую. База данных должна содержать записи с достаточным количеством говорящих, различными условиями записи, достаточным количеством акустических вариаций и всеми возможными лингвистическими предложениями. База данных должна иметь две части: обучающую и тестовую часть.

Структура файла для базы данных следующая:

```
├─ etc
│   ├─ my_db.dic              (Фонетический словарь)
│   ├─ my_db.lm.DMP           (Языковая модель)
│   ├─ my_db.filler           (Заполняющие фонемы)
```

```
│  ├─ my_db_train.fileids          (Список файлов для обучения)
│  ├─ my_db_train.transcription   (Транскрипции для обучения)
│  ├─ my_db_test.fileids          (Список файлов для тестирования)
│  └─ my_db_test.transcription    (Транскрипции для тестирования)
└─ wav
   ├─ speaker
   │  └─ file_1.wav               (Запись речевого высказывания)
```

Запись речи

Аудиозаписи должны содержать обучающее аудио, которое должно соответствовать аудио, которое будет необходимо распознать в конце. Файлы записи должны быть в формате .wav и с определенной частотой дискретизации - 16 кГц, 16 бит. Очень важно, чтобы аудио файлы имели определенный формат. Sphinxtrain поддерживает различные частоты дискретизации, но по умолчанию он настроен на обучение файлов именно в таком формате.

Для дальнейшей работы необходима установка следующих пакетов**:**

- sphinxbase-5prealpha;
- sphinxtrain-5prealpha;
- pocketsphinx-5prealpha.
- Perl (на практике в работе использовался ActivePerl для Windows);
- Python (использовался ActivePython для Windows).

II этап – Настройка скриптов обучения

Для начала обучения необходимо перейти в папку базы данных и выполнить следующую команду:

python ../sphinxtrain/scripts/sphinxtrain -t my_project setup

Команда осуществляет копирование необходимых файлов конфигурации в подпапку etc/ папки базы данных и подготавливает базу данных для обучения. Параллельно в процессе обучения создаются и другие папки с данными. После этой базовой настройки нужно отредактировать файлы конфигурации в папке *etc/:* файл *sphinx_train.cfg:*

- Настройка формата аудио базы данных;
- Настройка путей к файлам;
- Настройка типа модели и параметров модели;
- Настройка параметров функции звука;
- Настройка параллельных заданий для ускорения обучения;
- Настройка параметров декодирования;

III этап – Обучение

Запускается команда *python../sphinxtrain/scripts/sphinxtrain run* и проходит все необходимые этапы. Как правило, это занимает несколько минут (в данной работе время ожидание не превысило 10 минут), но на больших базах данных обучение может занять до месяца.

IV этап – Внутреннее обучение

В каталоге scripts *(./scripts_pl)* есть несколько каталогов, пронумерованных последовательно от 00 до 99. У каждого каталога либо есть каталог с именем *slave*.pl* либо файл с расширением *.pl*. Скрипт последовательно проходит по каталогам и выполняет либо один, *slave*.pl* либо один *.pl* файл.

V этап – Тестирование

Очень важно проверить качество обученной базы данных, чтобы выбрать лучшие параметры, понять, как работает приложение, и

оптимизировать производительность. Для этого необходим тестовый этап декодирования. Декодирование является последним этапом процесса обучения. Запускается процесс декодирования командой *sphinxtrain -s decode run*. Эта команда запускает процесс декодирования с использованием акустической модели, которую обучили, и языковой модели, которую настроили в *etc/sphinx_train.cfg* файле. Когда задание распознавания завершено, скрипт вычисляет частоту ошибок слова распознавания (WER) и частоту ошибок предложения (SER) [10]. Чем ниже эти показатели, тем лучше распознание. Для типичного 10-часового задания WER должен составлять около 10%. Для большой задачи это может быть и 30%.

В данной работе были получены следующие результаты:

SENTENCE ERROR: 83.3% (10/12) WORD ERROR RATE: 44.4% (30/54)

Это означает, что в 83.3% предложений сделана хотя бы одна ошибка. Если смотреть по словам (в данном случае их было 54), то из них правильно было распознано только 24.

Поскольку собственная база для обучения была слишком мала, было принято решение взять имеющуюся обученную модель и произвести ее адаптацию. Тем не менее в процессе обучения все голосовые сообщения сохранялись, что в дальнейшем может послужить для обучения собственной модели.

3.5.2. Адаптация акустической модели с помощью инструмента g2p

Существует ряд словарей, которые охватывают языки, которые поддерживаются Sphinx для английского, французского, немецкого, русского, голландского, итальянского, испанского и мандаринского языков. В случае необходимости другие словари можно найти в Интернете в свободном доступе.

В готовом словаре можно не удалять неиспользуемые слова, только если нет цели сэкономить память, т.к. они не влияют на точность распознавания.

Если же словарь не охватывает все необходимые слова, всегда можно расширить его с помощью инструмента g2p *(от англ. Grapheme-to-Phoneme)* [47]. Для новых языков сначала необходимо загрузить словарь, а затем использовать инструмент g2p для загрузки новых транскрипций и его расширения.

Существуют различные инструменты, которые помогут расширить существующий словарь для новых слов или создать новый словарь с нуля. Разработчики Sphinx рекомендуют использовать их новый инструмент g2p-seq2seq, основанный на нейронных сетях, реализованных в среде Tensorflow [47], который обеспечивает современную точность преобразования английского языка.

В данной работе был взят за основу готовый словарь русского языка (CMUDict) и его расширение было выполнено с помощью готового инструмента russian_g2p (разработанного специально для словаря русского языка). С помощью команды *python create_phonetic_dict.py —src new_words.txt —dst 'ru.dic —bad unknown_words.txt vs* были найдены слова, которые отсутствовали в словаре, например, появились некоторые новые сленг-слова или редкие имена. (Рис. 14), позже они были добавлены (включая их транскрипции) (Рис. 15).

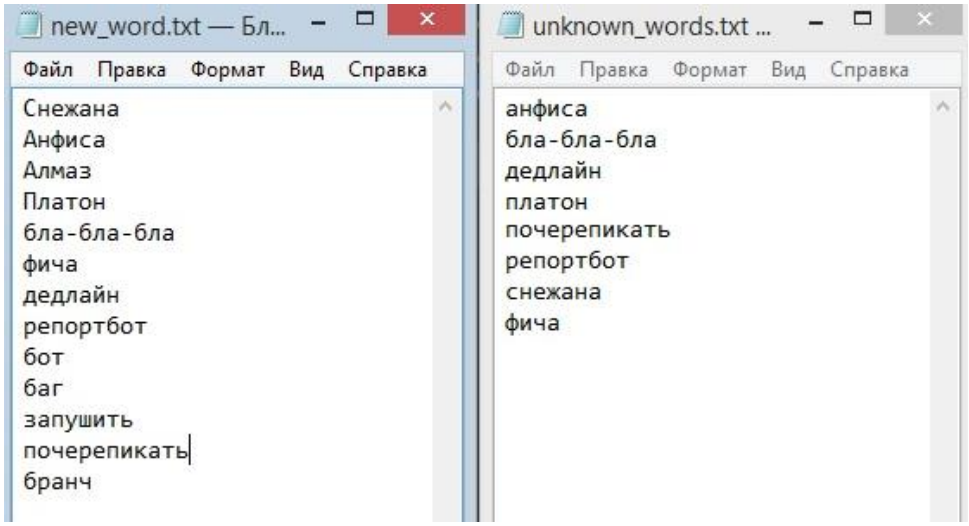


Рис. 14 Проверка словаря

```
D:\g2p\russian_g2p\russian_g2p\Grapheme2Phoneme.py:96: UserWarning: `бла-бла-бла`: the accent for this word is unknown!
  warnings.warn('`{0}`: the accent for this word is unknown!'.format(source_word))
['B', 'L', 'A', 'B', 'L', 'A', 'B', 'L', 'A']
>>> your_transcriptor.word_to_phonemes('Снежана')
D:\g2p\russian_g2p\russian_g2p\Grapheme2Phoneme.py:96: UserWarning: `Снежана`: the accent for this word is unknown!
  warnings.warn('`{0}`: the accent for this word is unknown!'.format(source_word))
['S0', 'N0', 'I', 'ZH', 'A', 'N', 'A']
```

Рис. 15 Транскрипция для расширения словаря

Полученные транскрипции добавляются в словарь для его расширения. Таким способом стало возможным распознавание новых слов (Рис. 16). Нельзя сказать, что точность равна 100%, т.к. некоторые слова были распознаны лишь частично, но в любом случае словарь был обновлен успешно.

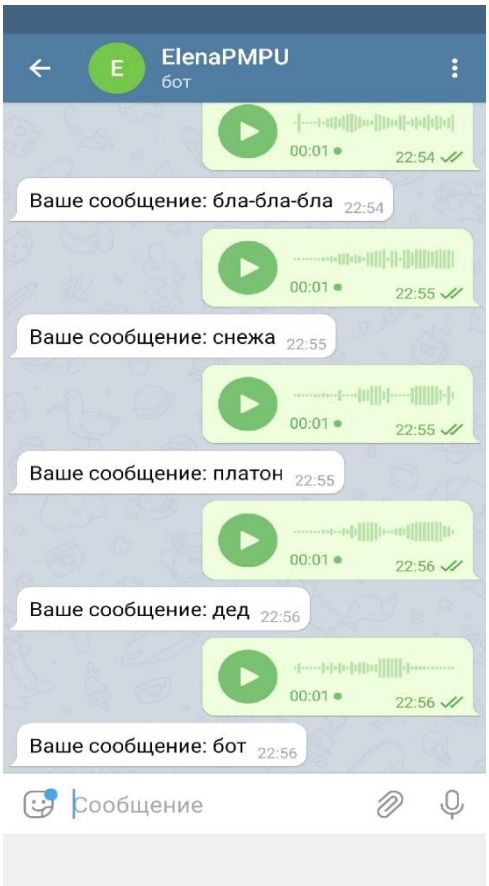


Рис. 16 Пример распознавания новых слов

3.6. Использование сети Sphinx

Одной из задач данной работы является интеграция Sphinx в разрабатываемое серверное приложение. Сервер поддерживает зависимость только для русского языка (Рис. 17).

```
package sphinx;

import edu.cmu.sphinx.api.Configuration;

public class Config {
    public static Configuration addCconfig() {
        Configuration configuration = new Configuration();
        configuration
                .setAcousticModelPath("resource:/zero_ru_cont_8k_v3/zero_ru.cd_cont_4000");

        configuration
                .setDictionaryPath("resource:/zero_ru_cont_8k_v3/ru.dic");
        configuration
                .setLanguageModelPath("resource:/zero_ru_cont_8k_v3/ru.lm");
        //configuration.setSampleRate(8000);
        return configuration;
    }
```

Рис. 17 Подключение зависимости русского языка

Текст распознается согласно следующему алгоритму:

1. Запуск сервера в среде разработки IntelliJ IDEA (Рис. 18).
2. Создание объекта для распознавания с текущими конфигурациями.
3. Получение запущенным сервером распознавания аудиодорожки в формате RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 16000 Hz.
4. Возврат результата запросившему ресурсу.

```java
public static void main(String[] args) throws IOException {
    BasicConfigurator.configure(); // needed for logger to work
    Configuration configuration = Config.addCconfig();

    Speech recognizer = new Speech(
            configuration);
    recognizer.start();
    final
    Undertow server = Undertow.builder()
            .addHttpListener( port: 4274,  host: "0.0.0.0")
```

Рис. 18 Функция запуска сервера

```
20:10:17.036 INFO speedTracker         # ---------------------------- Timers----------------------------------------
20:10:17.036 INFO speedTracker         # Name              Count   CurTime   MinTime   MaxTime   AvgTime   TotTime
20:10:17.060 INFO speedTracker         Load Dictionary     1       0,5810s   0,5810s   0,5810s   0,5810s   0,5810s
20:10:17.060 INFO speedTracker         Compile             1       6,3240s   6,3240s   6,3240s   6,3240s   6,3240s
20:10:17.061 INFO speedTracker         Load AM             1       2,8590s   2,8590s   2,8590s   2,8590s   2,8590s
0 [main] DEBUG org.jboss.logging  - Logging Provider: org.jboss.logging.Log4jLoggerProvider
396 [main] INFO Main  - Server start port 4274
525 [main] DEBUG io.undertow  - starting undertow server io.undertow.Undertow@27b5da6d
595 [main] INFO org.xnio  - XNIO version 3.3.8.Final
761 [main] INFO org.xnio.nio  - XNIO NIO Implementation Version 3.3.8.Final
1336 [XNIO-1 I/O-2] DEBUG org.xnio.nio  - Started channel thread 'XNIO-1 I/O-2', selector sun.nio.ch.WindowsSelectorImpl@92898b
1337 [XNIO-1 Accept] DEBUG org.xnio.nio  - Started channel thread 'XNIO-1 Accept', selector sun.nio.ch.WindowsSelectorImpl@1561bfdf
1336 [XNIO-1 I/O-1] DEBUG org.xnio.nio  - Started channel thread 'XNIO-1 I/O-1', selector sun.nio.ch.WindowsSelectorImpl@5bbd654e
1337 [XNIO-1 I/O-4] DEBUG org.xnio.nio  - Started channel thread 'XNIO-1 I/O-4', selector sun.nio.ch.WindowsSelectorImpl@e877e7e
1337 [XNIO-1 I/O-3] DEBUG org.xnio.nio  - Started channel thread 'XNIO-1 I/O-3', selector sun.nio.ch.WindowsSelectorImpl@2e5eb701
1420 [main] DEBUG io.undertow  - Configuring listener with protocol HTTP for interface 0.0.0.0 and port 4274
```

Рис. 19 Системная информация работы сервера

Глава 4. Интерфейс сервиса распознавания речи

4.1. Телеграм

С целью упрощения взаимодействия пользователя с сервисом распознавания речи был создан удобный интерфейс на базе популярного кроссплатформенного мессенджера Telegram, который позволяет обмениваться сообщениями и медиа файлами различных форматов.

Telegram-бот — это специальный аккаунт, который создается без привязки к какому-либо номеру телефона для автоматической обработки и отправки сообщений в беседе или групповом чате. Общение с такими аккаунтами организовано при помощи обычного HTTPS интерфейса с упрощёнными методами Telegram API. Разработчики называют его Bot API.

Рис. 20 Создание нового telegram-бота

Процесс создания бота (Рис. 20) заключается в обращении к пользователю @BotFather и выполнению его инструкций.

После получения собственного ключа авторизации (токена), происходит настройка аккаунта согласно документации [30].

Существует несколько видов таких аккаунтов с различными характеристиками. Некоторые из них используются для интеграции с другими сервисами, для одно- и многопользовательских игр, поисков собеседников или telegram-каналов, проведения опросов и др. Таким образом, любого бота можно запрограммировать в нужном ключе.

Основной ролью аккаунта является интерфейс к разработанному сервису, который работает на удалённом сервере. Общение с ботом происходит посредством команд,

которые должны начинаться с символа косой черты «/» и не могут быть длиннее 32 символов, выглядят они следующим образом: */команда [необязательный] [аргумент]*. Принимая во внимание тот факт, что работа аккаунта направлена на прием и распознавание голосовых сообщений, то в работе использовалась лишь команда */start*, которая начала общение с ботом.

Созданному telegram-боту можно отправить либо переслать голосовое сообщение и в ответ получить текстовое.

Алгоритм работы такого бота состоит из следующих этапов:

1. Изъятие аудиодорожки из полученного (пересланного) сообщения (Рис. 21). В случае если она отсутствует, бот автоматически уведомляет об этом соответствующим сообщением. Попытка сохранить сообщение (аудиодорожку) на сервер.

```
try {
    Voice voice = update.getMessage().getVoice();
    Message inMessage = update.getMessage();
    if (voice == null) {
        sendMessage( textwithletter: "Я тебя не понимаю. Пришли мне аудиосообщение", inMessage);
        return;
    }
    URL url = new URL( spec: url_t_bot + getBotToken() + get_file + voice.getFileId());
    JSONObject json = new JSONObject(IOUtils.toString(url));
    String key = json.getJSONObject("result").get("file_path").toString();
    URL urlRes = new URL( spec: url_bot_file +getBotToken()+"/" + key);
    URLConnection conn = urlRes.openConnection();
    InputStream is = conn.getInputStream();
    String name = UUID.randomUUID().toString();
    String filename = dir + name;
    OutputStream outstream = new FileOutputStream(new File( pathname: filename+".ogg"));
    byte[] buffer = new byte[Bytes];
    int len;
    while ((len = is.read(buffer)) > 0) {
        outstream.write(buffer,  off: 0, len);
    }
    outstream.close();
```

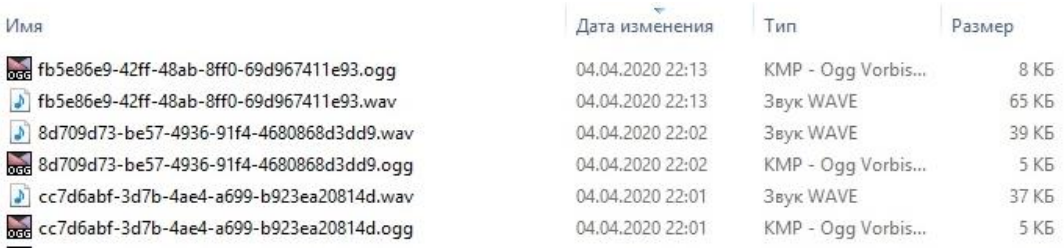
Рис. 21 Изъятие и сохранение аудиофайла.

2. Преобразование в нужный формат: из .ogg в .wav (с использованием FFmpeg)

FFmpeg [31] *(от англ. Fast Forward Mpeg)* – это бесплатный набор программ и библиотек по конвертированию аудио- и видеофайлов из одного формата в другой.

```
ffmpeg -i %1.ogg -ar 16000 -ac 1 %1.wav
```

Рис. 22 Команда для преобразования аудиофайла из формата .ogg в .wav



Имя	Дата изменения	Тип	Размер
fb5e86e9-42ff-48ab-8ff0-69d967411e93.ogg	04.04.2020 22:13	KMP - Ogg Vorbis...	8 КБ
fb5e86e9-42ff-48ab-8ff0-69d967411e93.wav	04.04.2020 22:13	Звук WAVE	65 КБ
8d709d73-be57-4936-91f4-4680868d3dd9.wav	04.04.2020 22:02	Звук WAVE	39 КБ
8d709d73-be57-4936-91f4-4680868d3dd9.ogg	04.04.2020 22:02	KMP - Ogg Vorbis...	5 КБ
cc7d6abf-3d7b-4ae4-a699-b923ea20814d.wav	04.04.2020 22:01	Звук WAVE	37 КБ
cc7d6abf-3d7b-4ae4-a699-b923ea20814d.ogg	04.04.2020 22:01	KMP - Ogg Vorbis...	5 КБ

Рис. 23 Сохраненные оригинал и преобразованные аудиофайлы

Помимо этого, можно склеить несколько файлов в один или разбить на отдельные, ускорить/замедлить, добавить субтитры в видео и звуковые дорожки, а также подавить сторонние шумы.

Шум — это совокупность апериодических звуков различной интенсивности и частоты. Шумы есть двух видов: постоянные (потрескивание, гудение) и непостоянными (звук двигателя автомобиля, падения, удара, мелодии).

Идея активного шумоподавления заключается в следующем: получение синхронного сигнала, который содержит исключительно шум (без вокала, голоса и прочего нужного звука). Затем этот «чистый» шум инвертируется в так называемую противофазу, с последующим наложением на «шумную» запись. Таким образом сам шум и «анти» шум подавляют друг друга, чтобы в итоге получилась нужная чистая запись [48].

Одним из способов шумоподавления может являться фильтрация «ненужных» звуковых диапазонов. Стандартным интервалом частот человеческого голоса является отрезок от 300 Гц до 3000 Гц. Таким образом, если ограничить диапазон частот рассматриваемой звукозаписи, то можно уменьшить фоновый шум. Тем не менее, такой способ не дает стопроцентного очищения и некоторые шумы все же присутствуют.

В начале ноября 2019 года вышла обновленная версия FFmpeg 4.1, которая используется в данной работе для преобразования аудиофайлов из одного формата в другой. Одним из существенных изменений стало

добавление фильтра *fftdnoiz filter*, который позволяет «очищать» от шума аудио- и видеодорожки.

Данный фильтр основывается на дискретном преобразовании Фурье [39], которое позволяет представить функцию или набор данных в виде комбинации синусов и косинусов, что позволяет выявить периодические компоненты в данных и оценить их вклад в структуру исходных данных или форму функции.

Формулы прямого и обратного преобразования Фурье:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} x_k e^{-\frac{2\pi i}{N} kn}$$

где N - количество значений сигнала, измеренных за период, а также количество компонент разложения; x_n - измеренные значения сигнала, которые являются входными данными для прямого преобразования и выходными для обратного; X_k - комплекс амплитуд синусоидальных сигналов, слагающих исходный сигнал; являются выходными данными для прямого преобразования и входными для обратного; поскольку амплитуды комплексные, то по ним можно вычислить одновременно и амплитуду, и фазу; k - индекс частоты.

Используемая команда для преобразования файла из одного формата в другой с подавлением шума представлена на Рис. 24:

```
ffmpeg -i %1.ogg -af afftdn -ar 16000 -ac 1 %1.wav
```

Рис. 24 Команда для преобразования аудиофайла из формата .ogg в .wav с шумоподавлением

На Рис. 25-26 представлен пример работы фильтра FFmpeg для шумоподавления (до/после):

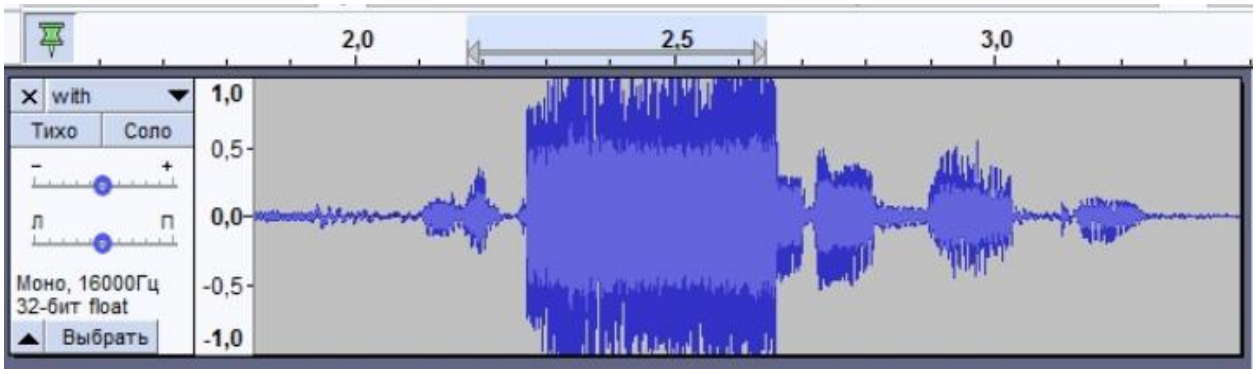


Рис. 25 Вид оригинальной звуковой волны в программе Audacity

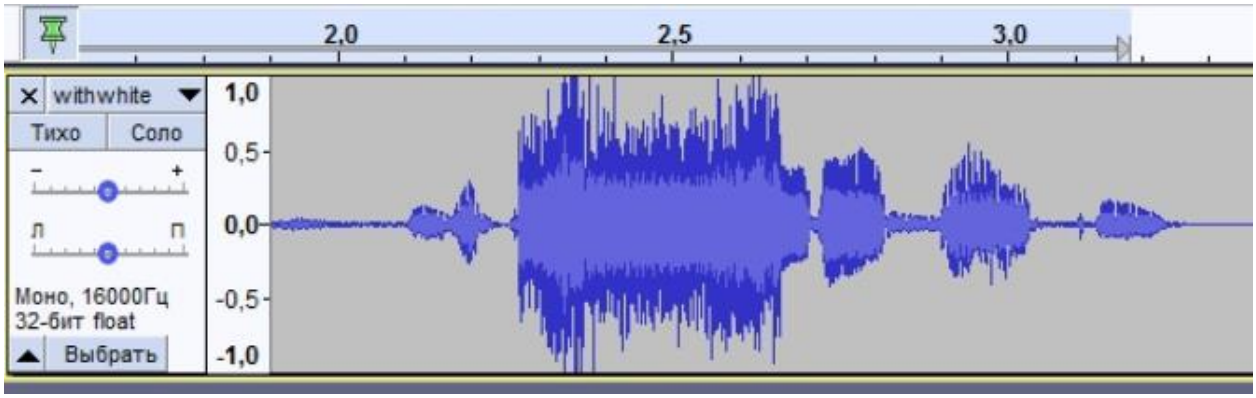


Рис. 26 Вид этой же звуковой волны в программе Audacity после применения фильтра

3. Отправка преобразованного файла на сервер распознавания и получение текстового эквивалента (Рис. 27).

```
BufferedReader in = new BufferedReader(new InputStreamReader(getTextFromVoice(res.toString()).getInputStream()));
String inputLine;
StringBuffer response_text = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response_text.append(inputLine);
}
in.close();
```

Рис. 27 Отправка на сервер аудиофайла в формате .wav и получение текстового фрагмента

4. Отправка полученного сообщения (Рис. 28).

```
sendMessage( textwithletter: "Ваше сообщение: " + response_text, inMessage);
```

Рис. 28 Отправка текстового сообщения пользователю

4.2. VPN

Для обеспечения более защищенного подключения и стабильной работы telegram-бота необходимо настроить VPN-подключение между серверами Telegram и сервером обработки данных и распознавания речи [38, 40].

VPN *(от англ. Virtual Private Network)* создается поверх сети Интернет для сохранения конфиденциальности пользователя. Основная цель — обеспечить безопасное и надежное частное соединение между компьютерными сетями через существующую общедоступную сеть, обычно Интернет.

Рис. 29 Схема работы подключения VPN

Виды и принципы работы

Существуют сети VPN двух видов:

1. Тип «сеть-сеть», который по-другому называется VPN между маршрутизаторами. Такой вид в большинстве случаев используется в корпоративной среде, как правило если у компании существуют филиалы с разным местоположением. Он применяется для формирования закрытой внутренней сети, где все филиалы смогли бы подключиться друг к другу. Такая технология по-другому называется интранет.

2. Шлюз защищенного удаленного доступа к сети VPN дает возможность подключаться как к интернету, так и к любой внутренней сети, по специальному приватному зашифрованному пути.

VPN сети работают по следующему алгоритму:

1. Прием данных, отправленных через Интернет;
2. Шифрование.

VPN принимает пакеты данных для отправки, покрывает их в дополнительный уровень безопасности и отправляет зашифрованные пакеты по пути через зашифрованный туннель.

3. Передача зашифрованных данных через собственный сервер.

В дополнение к тому, что зашифрованные данные становятся нечитаемыми, VPN передает эти данные через свой собственный сервер перед тем как отправить по назначению. Это создает впечатление, что данные поступили с этого сервера, и делает практически невозможным их подключение к пользователю, т.к. скрывается с уникальный IP-адрес пользователя.

4. Отправление данных по назначению.

Приложения

В настоящее время существует достаточное количество как платных VPN-приложений, так и приложений в свободном доступе. Они не могут сделать онлайн-соединения абсолютно анонимными, но как правило уровень конфиденциальности повышается, и пользователи всего мира могут сохранить приватность и безопасность своих данных в сети. Среди самых надежных и популярных бесплатных приложений можно назвать следующие:

- NordVPN;
- Hideman;
- TunnelBear;
- Windscribe;
- Hide.me.

В данной работе для решения задачи конфиденциальности данных пользователя использовалось приложение Windscribe.

Windscribe VPN – это безопасный VPN-сервис, который предназначен для приватного пользования сетью Интернет и избегания (беспроблемного преодоления) каких-либо территориальных ограничений и доступа к заблокированным сайтам.

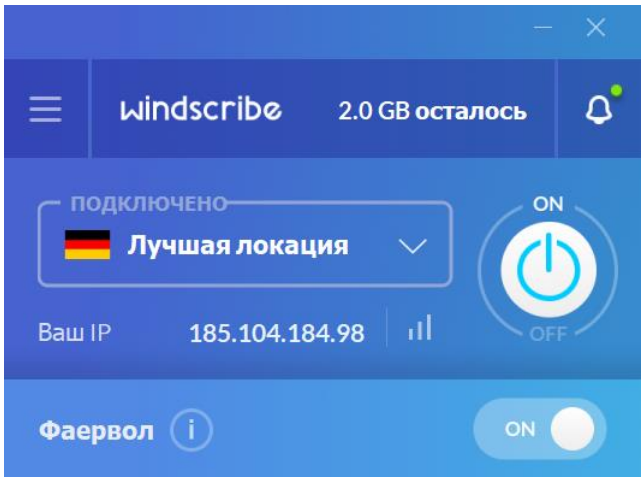


Рис. 30 Пример работы программы Windscribe

Такой выбор программы обусловлен широким выбором серверов, высокой скоростью соединения, гибкой настройкой тарифа под себя (для платной версии) и достаточным объемом трафика в бесплатной, отсутствием рекламы, достаточно оперативной англоязычной тех. поддержкой.

Выводы

В настоящей работе создана система автоматического распознавания речи на базе математического аппарата скрытых марковских моделей и нейронных сетей, а также разработан собственный интерфейс для преобразования аудио сообщений в текстовые.

В ходе выполнения работы были получены следующие результаты:

1) исследован математический аппарат скрытых марковских моделей и нейронных сетей;

2) реализован сервер для обработки и хранения данных;

3) на базе инструментальной системы CMUSphinx разработана и обучена программная система распознавания человеческой речи;

4) с помощью Sphinxtrain проведено обучение акустической модели, а с g2p-seq2seq – ее адаптация;

5) создан бот на основе платформы мессенджера Telegram для работы с аудиосообщениями;

6) отлажен процесс распознавания аудиофайлов и их конвертации в текст.

Заключение

В результате выполнения выпускной квалификационной работы был разработан сервис по распознаванию речи с возможностью его внедрения в будущем в другие приложения и социальные сети. Разработка сервиса производилось на ПК с процессором Intel® Core™ i5 3330 (3.0GHz, 4 ядра, 4 потока), видеокартой Radeon™ RX 480 (4 ГБ), ОЗУ 16 Гб.

В ходе работы был подобран и исследован широкий круг информационных источников, были обозначены и выделены основные методы для достижения поставленных в работе целей – комплекс программ CMUSphinx, на основе скрытых марковских моделей и нейронных сетей. Из анализа качества распознавания (см. Таблицу 1) следует, что используемые в Sphinx методы Баума-Велша и Витерби успешно применяются к решению задач распознавания раздельной речи.

Обученная через Sphinxtrain модель работает с точностью ~ 60%. Т.к. это показатель точности распознавания отдельно взятых слов, очевидно, что качество распознавания слитной речи, предложений будет еще ниже, т.к. отсутствует необходимая по величине база для обучения. В связи с этим, за основу был взят готовый словарь CMUDict и проведена его адаптация – обновление списка слов.

С помощью связки созданных telegram-бота и HTTP-сервера стало возможным и удобным отправка и пересылка аудиосообщений для их последующего распознавания. VPN-соединение в данном случае поддерживает бесперебойную работу бота и дополнительное шифрование сообщений.

В заключение следует отметить, что в течение всего времени разработки представленного сервиса велась запись каждого используемого аудиофайла, создавая собственную базу, которая может послужить для создания новой акустической модели и последующего развития данного проекта.

Список литературы

1. Ryzhikov I. S. About multiagent system applications for speech recognition problem // Сибирский журнал науки и технологий. 2012. №4 (44)
2. Фролов, А. В. Синтез и распознавание речи. Современные решения. / [Электронный ресурс] URL: http://www.frolov-lib.ru/books/hi/ch01.html
3. Мещеряков, Р. В. Структура систем синтеза и распознавания речи. // Известия Томского политехнического университета. Т.315, №5. – 2009. – С. 121-126.
4. Алимурадов, А. К. Обзор и классификация методов обработки речевых сигналов в системах распознавания речи. / А.К. Алимурадов и [др]. // Измерение. Мониторинг. Управление. Контроль. - 2015, №2. – С. 27-35
5. Титов Ю. Н. Современные технологии распознавания речи // Вестник российских университетов. Математика. 2006. №4.
6. Данков Н.И. Исследование возможностей нейросетевых технологий в области идентификации голоса // Экономика и качество систем связи. 2018. №3
7. Гапочкин, В. А. Нейронные сети в системах распознавания речи. / В.А.Гапочкин // "Science time". – 2014, № 1. – С. 29-36
8. Карпов Алексей Анатольевич, Кипяткова Ирина Сергеевна Методология оценивания работы систем автоматического распознавания речи // Приборостроение. 2012. №11
9. Гусев М.Н. Система распознавания речи: основные модели и алгоритмы / М.Н. Гусев, В.М. Дегтярев. - СПб.: Знак, 2013. - 128 с.
10. Беленко М.В., Балакшин П.В. Сравнительный анализ систем распознавания речи с открытым кодом // МНИЖ. 2017. №4-4 (58)
11. Алексеев И. В., Митрохин М. А., Кольчугина Е. А. Программное средство оценки эффективности технологий распознавания речи // Известия ВУЗов. Поволжский регион. Технические науки. 2018. №3 (47).
12. CMUSphinx Tutorial for Developers // GitHub. [Электронный ресурс] URL: https://cmusphinx.github. io/wiki/tutorialconcepts/
13. Speech and Machine Learning // Mozilla Research. [Электронный ресурс] URL:

https://research.mozilla.org/ machine-learning/

14. Распознавание речи. Документация SpeechKit Mobile SDK / Yandex. [Электронный ресурс] URL: https://tech.yandex.ru/speechkit/mobilesdk/doc/common/speechkit-common-asr-overview-technology-docpage/
15. Система распознавания речи CMU Sphinx. [Электронный ресурс] URL: http:// cmusphinx.sourceforge.net/
16. Запрягаев, В. А. Распознавания речевых сигналов. / С. А. Запрягаев, А. Ю. Коновалов //Вестник ВГУ. – 2009. №2. – С. 39 – 48
17. Блох, Джошуа Java: эффективное программирование, 3-е изд.: Пер. с англ. — СПб. : ООО “Диалектика”, 2019. — 464 с
18. Маковкин К. А. Гибридные модели: скрытые марковские модели и нейронные сети, их применение в системах распознавания речи // Модели, методы, алгоритмы и архитектуры систем распознавания речи: Изд-во «Вычислительный центр им. А. А. Дородницына РАН», – М.: 2006. – С.40-95
19. Гефке, Д. А. Применения скрытых марковских моделей для распознавания звуковых последовательностей. / Д.А. Гефке, П. М. Зацепин. // Известия Алтайского государственного университета. – 2012. – С. 72-76
20. Julius Tutorial [Электронный ресурс] URL: https://julius.osdn.jp/en_index.php
21. Меденников, И. П. Двухэтапный алгоритм инициализации обучения акустических моделей на основе глубоких нейронных сетей / И. П. Меденников // Научно-технический вестник информационных технологий, механики и оптики. — 2016. — Т. 16., № 2. — С. 379–381
22. Prudnikov, A. Improving Acoustic Models For Russian Spontaneous Speech Recognition / A. Prudnikov, I. Medennikov, V. Mendelev, M. Korenevsky, Y. Khokhlov // Speech and Computer, Lecture Notes in Computer Science. — 2015. — Vol. 9319. — P. 234–242
23. Рабинер Л. Р. Скрытые Марковские модели и их применение в избранных приложениях при распознавании речи / ТИИЭР. 1989. Т. 77. С. 86–120
24. Сорокоумова Д. А., Корелин О. Н., Сорокоумов А. В. Построение и обучение

нейронной сети для решения задачи распознавания речи / Труды НГТУ им. Р.Е. Алексеева. 2015. №3 (110)
25. Kaldi Tutorial [Электронный ресурс] URL: https://kaldi-asr.org/doc/
26. Авсентьев А.О., Лукьянов А.С. Применение скрытых марковских моделей для распознавания речи диктора. 2015. №2
27. Балакшин П. В. Повышение точности алгоритмов распознавания речи на основе скрытых марковских моделей // Научно-технический вестник информационных технологий, механики и оптики. 2008. №46
28. Dragon Naturally Speaking Solutions [Электронный ресурс]. URL: http://www.dragonsys.com
29. CMUSphinx Wiki Tutorial [Электронный ресурс] URL: http://cmusphinx.sourceforge.net/wiki/
30. Справочник по Bot API [Электронный ресурс]. URL: https://tlgrm.ru/docs/bots/api
31. FFmpeg Filters Tutorial [Электронный ресурс]. URL: https://www.ffmpeg.org/ffmpeg-filters.html#afftdn
32. Бовбель Е. И., Паршин В. В. Нейронные сети в системах автоматического распознавания речи - Зарубежная радиоэлектроника Успехи современной радиоэлектроники, 1998, №4, с 49-65.
33. Юрков П.Ю., Федоров 4. В.М., Бабенко Л.К. Распознавание гласных фонем с помощью нейронных сетей. // Тезисы доклада Всероссийского семинара «Нейроинформатика и ее приложения». - Красноярск, 1999.
34. Созыкин А. В. Обзор методов обучения глубоких нейронных сетей // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. №3
35. Савченко Л. В. Алгоритм пофонемного распознавания устной речи на основе метода нечеткого фонетического кодирования-декодирования слов // Информационно-управляющие системы. 2014. №1 (68).
36. Бондаренко Т.В., Федотов Е.А., Бондаренко А.В. Разработка http сервера // ИВД. 2018. №2
37. Дроздов С. А., Луканина В. Е. Особенности проектирования серверного и

клиентского программного обеспечения web-сайта с использованием rest-архитектуры // Вестник МГУП. 2016. №2

38. Николахин А.Ю Использование технологии vpn для обеспечения информационной безопасности // Экономика и качество систем связи. 2018. №3 (9)
39. Филиппенко И. Г., Пенкина О. Е., Филиппенко О. И. Многоуровневое дискретное преобразование Фурье // ВЕЖПТ. 2008. №3 (36).
40. Волохов В. В. Исследование принципов работы VPN, разработка политики безопасности VPN. Использование анонимайзеров // Наука, техника и образование. 2018. №5 (46)
41. Стас Тамби Тахсинович, Метод получения векторов акустических признаков для распознавания последовательности фразы в условиях шумовых помех // Известия ВУЗов. Поволжский регион. Технические науки. 2015. №2 (34).
42. Preeti Saini, Parneet Kaur Automatic Speech Recognition: A Review -International Journal of Engineering Trends and Technology - Volume4Issue2, 2013.
43. Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury Deep Neural Networks for Acoustic Modeling in Speech Recognition - IEEE, Signal Processing Magazine, 2012
44. Englund, C. Speech recognition in the JAS 39 Gripen aircraft -Adaptation to speech at dierent G-loads: Master Thesis in Speech Technology / C. Englund // Department of Speech, Music and Hearing, Royal Institute of Technology. - Stockholm. - 2004. -11th March. - P. 1
45. Dongsuk, Y. Robust speech recognition using neural networks and hidden markov models: Doctor of Philosophy / Y. Dongsuk. - New Jersey : Graduate school - New Brunswick Rutgers, The State University of New Jersey, 1999. - 18 p
46. Giampiero, S. Mining speech sounds, machine learning methods for automatic speech recognition and analysis: doctoral thesis / S. Giampiero. - Stockholm : KTH school of computer science and communication, 2006. - 25 p.
47. An Open Source Machine Learning Framework for Everyone // TensorFlow. -

[Электронный ресурс] URL: https://www.tensorflow.org/

48. Benesty J., Sondh M., Huang Y. (eds.) Springer Handbook of Speech Recognition. - N. Y.: Springer, 2008. - 1159 p.
49. Specht D. F. Probabilistic neural networks // Neural Networks. 1990. Vol. 3. P. 109-118.
50. Christopher M. Bishop. Pattern Recognition and Machine Learning // Information Science and Statistics. – 2006.
51. Roberto Pieraccini From AUDREY to Siri. Is speech recognition a solved problem? // Proc. of International Computer Science Institute at Berkeley. 2011. P. 1-20.
52. Schuster M. Speech Recognition for Mobile Devices at Google// LNCS. 2010. Vol. 6230. P. 8-10
53. Vyas G., Kumari B. Speaker Recognition System Based on MFCC and DCT. Intern. Journ. of Engineering and Advanced Technology (IJEAT), 2013, vol. 2, iss. 5
54. Walker W., Lamere P., Kwok P., Raj B., Singh R., Gouvea E., Wolf P., Woelfel J. Sphinx-4: A flexible open source framework for speech recognition. Technical Report, 2004
55. Nilsson M., Ejnarsson M. Speech recognition using hidden Markov model. Karlskrona: Kaserntryck-eriet AB, 2002.
56. Graves A., Mohamed A., Hinton G. Speech recognition with deep recurrent neural networks // Proceedings of International Conference on Acoustics, Speech and Signal Processing. Piscataway: IEEE, 2013. P. 6645-6649.
57. Undertow Tutorial [Электронный ресурс]. URL: http://undertow.io/
58. Gradle Tutorial [Электронный ресурс]. URL: https://gradle.org/
59. Postman Tutorial [Электронный ресурс]. URL: https://www.postman.com/

Приложение 1. Схема работы разработанного сервиса для распознавания речи