МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет» (национальный исследовательский университет)

Высшая школа электроники и компьютерных наук

Кафедра «Информационно-измерительная техника»

РАБОТА ПРОВЕРЕНА

Рецензент, программист

_____/ Т.Д. Исупова /

«27» мая 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.т.н., проф.

_____/ А.Л. Шестаков /

«_____» _____2020 г.

РАЗРАБОТКА ЛОКАЛЬНОГО РАБОЧЕГО МЕСТА ИССЛЕДОВАТЕЛЯ ИНТОНАЦИОННЫХ КОНСТРУКЦИЙ РУССКОГО ЯЗЫКА

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ –12.04.01.2020.308-607.ВКР

Руководитель, к.т.н., доц. каф. ИнИТ

_____ / Д.А. Кацай/

«_____» _____2020 г.

Автор

студент группы КЭ – 225

_____/ А.А. Куликов/

«_____» _____2020 г.

Нормоконтролер, к.т.н., доц. каф. ИнИТ

_____/ А.С. Волосников/

«_____» _____2020 г.

Челябинск 2020

АННОТАЦИЯ

Куликов А. А. Разработка локального рабочего места исследователя интонационных конструкций русского языка – Челябинск: ЮУрГУ, КЭ-225, 2020, 152 с, 45 ил., библ. список – 116 наим., 1 прил.

Новые методики обучения с последующей возможностью на их основе разработки нейронных сетей для их реализации, подобных аналогов наборов данных для нейронных сетей на данный момент не существует. Вместо преобладающего в исследованиях акцента формантного анализа, предлагается использование мел-кепстральных коэффициентов и оценка распределения коэффициентов корреляции по частоте их появления между носителями русского и китайского языков. Отличие и новизна данной работы заключаются в определении и установлении дефектов речи.

Цель выпускной квалификационной работы – разработка модуля подбора оптимальных параметров для локального рабочего места исследователя интонационных конструкций русского языка.

Задачи выпускной квалификационной работы:

1. Обзор существующих аналогов систем по распознаванию речи;
2. Изучение структуры и реализованных концепций по распознаванию речи;
3. Изучение структуры и реализованных концепций по распознаванию интонационных конструкций;
4. Запуск программы, экспериментальное испытание и анализ.

По результатам работы произведен обзор аналогов, была предложена альтернатива существующим методам – использование мел-частотных кепстральных коэффициентов. Разработан алгоритм и написан код программы модуля подбора параметров для локального рабочего места исследователя интонационных конструкций русского языка.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 9

1 ОБЗОР СУЩЕСТВУЮЩИХ СИСТЕМ ПО РАСПОЗНАВАНИЮ РЕЧИ, ИХ ХАРАКТЕРИСТИК И МЕТОДОВ РАБОТЫ 11

1.1 Системы распознавания речи с закрытым исходным кодом 13

1.2 Системы распознавания речи с открытым исходным кодом 19

1.3 Распознавание речи 24

1.4 Цифровая обработка сигналов 27

1.5 Анализ метода мел-частотных кепстральных коэффициентов применительно к процедуре голосовой аутентификации 29

2 УСТАНОВЛЕНИЕ ОПТИМАЛЬНОГО РЕШЕНИЯ НА ОСНОВЕ СУЩЕСТВУЮЩИХ АЛГОРИТМОВ И МЕТОДИК РАСПОЗНАВАНИЯ РЕЧИ 37

2.1 Энергия 38

2.2 Форманты 39

2.3 Просодические особенности при классификации акцентов 41

2.4 Распознавание акцентированной речи на основе знания родного языка 43

2.5 Обработка речевого сигнала 47

2.6 Использование формант в речевом образце 49

2.7 Спектральные характеристики речевого сигнала 49

2.8 Анализ речевого сигнала с использованием корреляционной функции 50

2.9 Кепстральный анализ речевого сигнала 51

2.10 Анализ с использованием нейронных сетей 54

2.11 Анализ с использованием скрытых марковских моделей 56

2.12 Анализ с использованием динамического трансформирования времени 57

2.13 Многокомпонентная информация, заложенная в речевом образце 59

2.14 Алгоритмы пофонемного распознавания речи в амплитудно-временной области 62

2.14.1 Представление речевого сигнала во временной области 63

2.14.2 Классификация шипящих и пауз..... 64

2.15 Выбор языка и среды программирования 65

2.16 Мел-частотные кепстральные коэффициенты 65

3 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 67

3.1 Программная реализация алгоритмов..... 69

3.1.1 Алгоритм обнаружения речевой активности..... 69

3.1.2 Кадрирование входного сигнала во временной области 72

3.1.3 Алгоритм динамической трансформации временной шкалы 73

3.1.4 Разложение в ряд Фурье..... 75

3.1.5 Применение оконной функции Хэмминга 75

3.1.6 Расчет mel-фильтров..... 77

3.1.7 Применение фильтров и логарифмирование энергии спектра ... 79

3.1.8 Косинусное преобразование..... 79

3.1.9 Мел-частотные кепстральные коэффициенты..... 79

3.2 Формирование проектной библиотеки образцов произношения..... 81

3.2.1 Группы пользователей проекта 81

3.2.2 Группы звуковых образцов проекта 81

3.2.3 Основные требования к записи образцов 82

3.2.4 Основная информация записанных образцов 82

3.3 Разработка программного кода обработки речи 83

4 РАЗРАБОТКА МОДУЛЯ ПОДБОРА ОПТИМАЛЬНЫХ ПАРАМЕТРОВ ДЛЯ ЛОКАЛЬНОГО РАБОЧЕГО МЕСТА ИССЛЕДОВАТЕЛЯ ИНТОНАЦИОННЫХ КОНСТРУКЦИЙ. ЭКСПЕРИМЕНТАЛЬНАЯ ПРОВЕРКА И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 89

4.1 Создание библиотеки речевых образцов 90

4.2 Установление частотного диапазона входного сигнала 91

4.3 Применение алгоритма динамической трансформации временной шкалы 92

4.4 Применение алгоритма обнаружения речевой активности 93

4.5 Применение оконной функции Хэмминга 94

4.6 Установление размера ширины кадра 95

4.7 Установление размера кадрового перекрытия 99

4.8 Использование мел-частотных кепстральных коэффициентов в корреляции Пирсона 100

4.9 Результаты программной разработки локального рабочего места исследователя интонационных конструкций русского языка 103

ЗАКЛЮЧЕНИЕ 107

БИЛИОГРАФИЧЕСКИЙ СПИСОК 109

ПРИЛОЖЕНИЯ 122

ПРИЛОЖЕНИЕ А Листинг программы. 122

ВВЕДЕНИЕ

Разработка программного обеспечения для изучения различий в произношении носителей русского языка и дикторов, для которых он не является родным – актуальная задача на сегодняшний день. Это подтверждается распоряжением руководства от 3.09.2018 № 308/92 об организации проектного образования на тему «Интеллектуальная система анализа интонационных конструкций русского языка», грантовой поддержкой по постановлению правительства РФ № 1642 от 26.12.2017: Государственная программа РФ «Развитие образования». И связано с необходимостью популяризации русского языка за рубежом, примером тому можно считать открытие 8 центров по его изучению в Китае.

Требуется составить новые алгоритмы обработки речевых образцов с целью выявления интонационных признаков, сформировать и выполнить новые методики обучения с последующей возможностью разработки нейронных сетей для их реализации, подобных аналогов наборов данных для нейронных сетей на данный момент не существует. Вместо преобладающего в исследованиях акцента формантного анализа предлагается использование мел-кепстральных коэффициентов и оценка распределения коэффициентов корреляции по частоте их появления между носителями русского и китайского языков.

В настоящий момент существует большое множество образовательных ресурсов, у которых в наличии имеется обучение с родного языка на необходимый пользователю иностранный. Однако, как правило, практика верного произношения на них не в приоритете, либо совершенно отсутствует. В лучшем случае используется распознавание речи по типу google translate, и подход нацелен лишь на распознавание сказанного, вне зависимости было ли произношение правильным. Стоит заметить, что почти каждое используемое для этих целей ПО закрыто, т.е. каким образом оно работает, с помощью чего и на что опирается – остается неизвестным. Также в этом имеется минимум пользы при изучении вариации между

языками, однако некоторые платформы все же собирают речевые образцы, но что с ними происходит в дальнейшем – не разглашается.

С другой стороны, есть большое количество высокоспециализированного и многофункционального ПО (Praat, WaveSurfer, Speech Analyser и пр.), предназначенного для решения исследовательских задач в области обработки речи. Проблема состоит в том, что все они предоставлены на английском языке, требуют от филолога очень высокую подготовку по технической части, в том числе в умении написания кода, обладают очень неочевидным интерфейсом, недостаточно развернутой инструкцией. Также ставится под сомнение их эффективность, т.к. в статьях, в которых сравниваются эффективность и производительность формантного анализа по нескольким ПО, обнаруживается множество различий в точности.

Стоит упомянуть, что с интонациями работают лишь в узкоспециализированных областях, таких как, например, криминалистика и фонология в медицине, но всё же и они взаимодействуют лишь с языком, без особенностей межязыкового взаимодействия, потому что в паре языков «родной и изучаемый» сразу же возникает очень высокое количество отличий, которые необходимо фиксировать и впоследствии с ними работать.

На основе вышесказанного перед нами стоит цель создать подобный ресурс.

1 ОБЗОР СУЩЕСТВУЮЩИХ СИСТЕМ ПО РАСПОЗНАВАНИЮ РЕЧИ, ИХ ХАРАКТЕРИСТИК И МЕТОДОВ РАБОТЫ

Обладая значительной ролью в разработке, организации и модернизации самых современных поисковых систем и систем управления, системы распознавания речи и голосового поиска очень стремительно развиваются. Постепенно становясь неотъемлемой частью повседневной жизни, голосовое управление с каждым днем все более выбирается за пределы электронных гаджетов.

Разработка приложений, целиком подконтрольных голосовому управлению, совершенно новые инструменты к формированию и выводу результатов информации, автоматическое распознавание акцентов и диалектов – главные перспективы процесса развития в данной области. Следует отметить, что сегодня задача по распознаванию речи не реализована должным образом, т.к. система не может распознать сленг или акцент, оттого она до сих пор несовершенна.

Прежде чем любая машина сможет интерпретировать речь, микрофон должен перевести вибрации голоса человека в волновой электрический сигнал. Этот сигнал, в свою очередь, преобразуется аппаратными средствами системы, например звуковой картой компьютера, в цифровой сигнал. Именно цифровой сигнал анализирует программа распознавания речи, чтобы распознавать отдельные фонемы, основные строительные блоки речи. Фонемы затем рекомбинируются в слова. Однако многие слова звучат одинаково, и, чтобы выбрать соответствующее слово, программа должна полагаться на контекст. Тем не менее для исправления ошибок очень часто требуется вмешательство человека [1].

Системы по распознаванию речи продолжают модернизировать, например с помощью внедрения технологии «*Connectionist Temporal Classification*» (CTC, «нейросетевая темпоральная классификация») [2], которая повысила чистоту и надежность идентификации речевого сигнала в условиях постороннего шума, а вдобавок увеличена ее скорость распознавания.

Данная технология представляет из себя акустические модели, реализованные с помощью основанной на динамических изменениях взаимосвязи временных характеристик с каким-либо моментом времени. Данная технология была внедрена в свои внутренние системы компанией Google в третьем квартале 2018 года. Разработка системы распознавания речи – крайне ресурсозатратная и трудоемкая задача. В связи с этим разумно предположить, что для разработки будущего проекта необходимо интегрировать в него уже самую оптимальную существующую на данный момент версию системы.

В основном, системы распознавания речи делятся на две группы: с открытым исходным кодом и с закрытым исходным кодом. У каждой из групп имеются как свои преимущества, так и недостатки.

Открытый исходный код обращается к любой программной среде, он выполнен доступным и бесплатным для использования или модификации как для рядового пользователя, так и для других разработчиков, если они посчитают это целесообразным. Как только создатели программы с открытым исходным кодом запускают ее в сеть, она попадает в глобальный интернет, где каждый может ею воспользоваться или модернизировать. Сообщество программы совместными усилиями разработчиков улучшает код и совместно использует то, что у них получилось. Активное и хорошо осведомленное сообщество жизненно важно для здоровья и успеха программы с открытым исходным кодом. В этом преимущество открытого исходного кода. Ресурс с закрытым исходным кодом, о котором будет написано ниже, может прекратить поддержку программного обеспечения практически в любое время, оставив все, что у вас есть на данный момент без возможности дальнейших обновлений. В то же время для программного обеспечения с открытым исходным кодом, если сообщество достаточно велико и активно, можно ожидать новых обновлений, функций или исправлений в течение довольно длительного времени.

Однако в нашем случае, чтобы эффективно воспользоваться открытым исходным кодом, необходимо обязательно иметь готовую записанную исходную биб-

лиотеку с достаточной выборкой артикуляционных характеристик, языковых единиц и т.п.

Программное обеспечение с закрытым исходным кодом – это то, которое, как правило, разрабатывается и исправляется по усмотрению его автора, поэтому оно не содержит необходимую документацию, без которой невозможно внедрение дополнительных решений или модернизации уже имеющихся для оптимизации под поставленные задачи. Программное обеспечение с закрытым исходным кодом, как правило, имеет лишь столько гибкости и свободы, сколько было задумано разработчиком. Изменение этих данных может аннулировать гарантию или вызвать еще большие проблемы. Однако достоинством данного варианта является, что данные системы владеют высокой надежностью в идентификации речевого сигнала, но зачастую указанные системы – коммерческие и за использование разработок речевых технологий компании требуется платная подписка лицензионного соглашения, представленная компанией-разработчиком.

Благодаря купленной лицензии пользователю программного обеспечения с закрытым исходным кодом предоставляется перечень часто задаваемых вопросов и ответов на них, различные руководства и возможность связи с техподдержкой при возникновении проблем с программным обеспечением. Все это будет организовано и задокументировано. С другой стороны, для программного обеспечения с открытым исходным кодом не так много вариантов поддержки, только возможность обсуждения существующей проблемы через форумы, поиск и чтение статей или вовсе наем эксперта.

Так как у каждой из систем имеются свои достоинства и недостатки, было принято решение рассмотреть самых популярных представителей из обеих групп, и на основании оценок выбрать наиболее оптимальный вариант.

1.1 Системы распознавания речи с закрытым исходным кодом

Далее будут представлены наиболее распространенные и с самым высоким рейтингом системы по распознаванию речи с закрытым исходным кодом и сред-

ствами интеграции среди разработчиков [3]. Подобные системы разрабатываются на базе «*software development kit*» (SDK), представляющей из себя комплект программных инструментов для разработки и создания разнообразных приложений (в данном случае по идентификации и распознаванию речи), предназначенных для конкретных систем, и с заданными условиями системы и прочих платформ [4].

Мировое первенство в разработке и производстве ПО для распознавания речи принадлежит компании **NuanceCommunications** [5]. Организация работает со спецслужбами США, а также тесно сотрудничает с такими гигантами, как Apple.

Платформа **SpeechKit** [6] – мультимедийная система для использования логопедами для помощи в реабилитации пациентов с нарушениями функций опорно-двигательного аппарата. Зачастую пациенты – это жертвы инсульта, которым приходится заново развивать способность к эффективному общению и артикуляции. Процесс является трудоемким и обычно сводится к ситуации, когда взаимодействие между терапевтом и клиентом строится на визуальных и звуковых сигналах. Вместо обычных бумажных подсказок, используемых терапевтом, это мультимедийное решение предлагает экранные подсказки вместе с высококачественной оцифрованной речью.

Принцип работы данной платформы состоит в быстром подключении внутренних служб для идентификации и последующем синтезе речи в требуемые приложения и системы. Распознавание и синтез речи полностью осуществляется на серверах, которые несут ответственность за значительную часть операций в цикле обработки речи, помимо этого, благодаря ним относительно конфигурации разработчика производится аутентификация.

Поскольку для работы SpeechKit требуется удаленный доступ для использования разработанных библиотек и алгоритмов по распознаванию и синтезу речи, ему обязательно требуется выполнение следующих условий:

- распознавание и авторизация приложения;
- реализация соединения с сервером обработки речи.

Технология формирования и отправки моментальных запросов на сервер значительно улучшает степень надежности системы и конечный результат идентификации и распознавания речи.

Инструментарий **DragonMobile SDK** [7] фирмы Nuance предоставляет речевые услуги разработчикам, которые хотят усовершенствовать свои приложения с помощью функции распознавания речи и функции «текст в речь», он выполнен из компонентов клиента и сервера, состоит из множества примеров кода и образцов проектов, документации, а также простой структуры, обеспечивающей простой путь к интеграции речевых сервисов в любое приложение. Платформа подразделяется на две реализации: одна представляет собой приложение для клиента, другая – набор инструментов для внедрения на сервер, рассмотрим каждую подробнее.

Приложение **Dragon SDK Client** (DSC) [8] включает в себя инструменты, библиотеки и компоненты ActiveX, необходимые для добавления современной технологии распознавания речи для включения голоса в любом приложении на базе Windows. Встроен быстрый и простой конструктор приложений с поддержкой речи с нуля или добавление распознавания речи к существующим приложениям, таким как приложения для автоматического создания субтитров в реальном времени или документальной судебной отчетности. Имеется возможность конечным пользователям вводить, редактировать и исправлять текст, управляя приложением одним голосом. Реализована система для создания собственных команд или пользовательского словаря, адаптированного для каждого приложения. Платформа позволяет пользователям немедленно включить голосовую связь без дополнительного оборудования, используя лишь встроенный микрофон ПК.

Dragon SDK Server (DSS) [9] предоставляет набор инструментов и интерфейсов, которые помогают разработчикам программного обеспечения успешно создавать и интегрировать автоматическое распознавание речи в системы документооборота. Данная система была специально разработана для интеграции в решения, которые требуют распознавания речи в фоновом режиме, без какого-либо ин-

терактивного диктофона или саморедактирования со стороны пользователей. Бэкэнд-распознавание («backend — программно-аппаратная часть сервиса речи») [10] позволяет выполнять пакетную обработку записанной речи с целого ряда устройств, включая цифровые регистраторы, планшеты, смартфоны и системы телефонного диктофона. Dragon SDK Server не является самим процессом транскрибирования, что является перечнем правил и системы процессов, разработанной для записи речи, однако DSS может быть интегрирован в существующие системы для автоматизации транскрибирования, повышения производительности транскрипции и снижения на него затрат. Также в систему включена возможность создания пользовательских словарей для адаптации номенклатуры и слов к конкретным отраслям, компаниям, отделам или должностям.

Главный минус использования данными системами – лимитированный бесплатный функционал (не более 10.000 запросов в сутки, только оплачиваемая лицензия позволяет обойти это ограничение).

GoogleSpeech Recognition API [11] – разработка знаменитой корпорации Google, которая была внедрена во все линейки технических устройств со средством приема голосовой информации. Возможности платформы позволили применять голосовой поиск, базируясь на анализе и идентификации речи. Данный комплексный подход обработки речевых технологий был внедрен компанией в поисковую систему Google Chrome на персональные компьютеры с его поддержкой в середине 2011 года, а немногим позже аналогичной функцией были оснащены смартфоны с операционной системой Android.

После обязательного прохождения регистрации в Google Developers, компанией предоставляется возможность использования имеющихся системных баз данных по речевому распознаванию. Это стало возможным с конца первого квартала 2014 года, когда руководством Google было принято решение о предоставлении абсолютно легального доступа к ресурсам программного интерфейса приложения (API) для разработчиков со всего мира. В подавляющем большинстве самых посещаемых сайтов, таких как: YouTube, MS Office, Wikipedia, Windows Live, поис-

ковых ресурсов: Google, Yahoo, Yandex и остальных, изначально предустановлен блок голосового поиска. Для сервисов, которые функционируют на поисковых формах программного языка управления интернет-ресурсами «HyperText Markup Language» [12] пятой версии (HTML5) было создано расширение, добавляющее возможность для ввода информации с помощью голоса.

Выполнение операции речевого поиска происходит благодаря формированию POST-запросов по требуемому адресу с использованием акустических форматов типа «.spc» или «.flac». После, применяя наиболее подходящий язык программирования, осуществляется анализ и распознавание WAVE-файлов. Данная структура Google по распознаванию голосовых запросов в значительной степени схожа с системой DragonMobile SDK. Однако основное отличие и достоинство относительно нее в том, что у GoogleSpeech Recognition API отсутствует лимит по количеству запросов на сервер за один день.

Microsoft Speech API [13] – интерфейс программирования речевых приложений или SAPI – это API, разработанный Microsoft для обеспечения возможности использования распознавания речи и синтеза речи в приложениях Windows. На сегодняшний день выпущен ряд версий API, которые поставлялись либо в составе Speech SDK, либо в составе самой ОС Windows.

Существовало два основных "семейства" API Microsoft Speech. Версии SAPI 1-4 похожи друг на друга, их отличие в дополнительных функциях в каждой новой версии. SAPI 5, однако, был совершенно новым интерфейсом, выпущенным в 2000 году. С тех пор было выпущено несколько субверсий этого API.

Варианты применения Microsoft Speech API:

- Embedded – реализованные приемы, дающие возможность пользователю взять управление различными техническими устройствами с помощью голосового управления, например управление в операционной системе Windows Automotive;
- Services – создание голосовых утилит для их применения в реальном времени;

• Speech Platforms – внедрение платформы в разнообразные приложения, которые применяют различную форму программного обеспечения Microsoft, к примеру, включение в языковые пакеты;

• Windows Server – для приложения под управлением операционной системой включается речевая технология, в которой применяется либо исходный, либо подконтрольный ему код прямо из API, плюс применение голосового метода управления, предустановленного в операционную систему.

Самым распространенным выбором при употреблении русской речи из богатого множества SDK по результатам опроса среди российских разработчиков оказался **Yandex SpeechKit** [14], однако данная платформа обладает ограниченным количеством (не более десяти тысяч) запросов от адресата в сутки.

Производительность системы по распознаванию речи от Яндекс напрямую зависит от основополагающих требований, разработанных для пользователя приложения:

- корректность и разборчивость речи;
- скорость и темп произношения;
- качество исходного звука;
- кодирование и его помехоустойчивость;
- сложность произнесенной фразы;
- продолжительность записываемой речи.

Одновременно с отправлением информации на хост для последующей обработки, записываемая речь распознается в режиме онлайн, при этом система имеет задержку менее одной секунды. Для предоставления настолько приличной реакции с откликом, разработанная концепция осуществляет работу в режиме потоковой идентификации информации с интервальной оценкой, что подразумевает под собой разбиение речи говорящего на малые части с последовательной их обработкой прямо во время записи.

Выше представлен список из наиболее востребованных на сегодняшний день систем распознавания речи с закрытым исходным кодом. У каждой имеются достоинства и недостатки, например, отличная внутренняя документация и простой и интуитивно понятный код для подключения модуля по распознаванию речи у DragonMobile SDK. Поэтому он выглядит, как отличный выбор для начала разработки на его основе, однако сложная структура лицензирования и строгие нормы по контролю его использования указывают на нецелесообразность его использования.

Основываясь на приведенных характеристиках, таких как, самое высокое быстродействие относительно остальных, основанное на огромных вычислительных мощностях компании, а также удобная, почти универсальная программно-модульная встраиваемость и неограниченное количество запросов в день, наиболее перспективной выглядит GoogleSpeech Recognition API. Помимо этого, подтверждением данного выбора является тот факт, что корпорация Google постоянно модернизирует и развивает собственные сервисы по распознаванию речи, что является неоспоримым достоинством, т.к. возможности разработки неизменно расширяются, что значительно упрощает выполнение каких бы то ни было поставленных задач.

1.2 Системы распознавания речи с открытым исходным кодом

Ниже представлен перечень систем распознавания речи с открытым исходным кодом, выбор которых основывался на популярности у разработчиков программного обеспечения, частоте употребления в современных научно-исследовательских журналах и существующих разработок последних лет [15-23].

CMU Sphinx [24] – современная, независящая от диктора, непрерывная система распознавания речи, полностью написанная на языке программирования Java. Она была создана в рамках совместного сотрудничества между университетом Карнеги-Меллон, Калифорнийского университета в Санта-Крус (UCSC) и Массачусетского технологического института (MIT).

Принцип работы последней версии системы Sphinx-4 основана на закономерностях, возникших в результате разработок прошлых систем, а также новых требованиях, основанных на областях, которые в настоящее время исследуются. Для реализации данной платформы и предоставления разработчикам «готовой к исследованиям» системы, Sphinx-4 также включает в себя несколько реализаций как базовых, так и современных методов. Фреймворк и внедрение модулей доступны через открытый исходный код по программной лицензии университета Беркли (BSD) [25].

С помощью бета-версии 1.0 каждый может получить полное дерево исходного кода Sphinx-4 вместе с несколькими акустическими и языковыми моделями, способными управлять разнообразными задачами, начиная от простого распознавания цифр и заканчивая большим лексическим распознаванием N-граммы [26], элементы которого могут быть фонемами, слогами, буквами, словами в соответствии с примером, но на практике может представляют собой даже целый ряд слов или устойчивых словосочетаний. Поскольку он написан полностью на языке программирования Java, Sphinx-4 может работать на различных платформах, не требуя никакой специальной компиляции («перевод текста программы, написанного на языке программирования, в набор машинных кодов» [27]) или каких-либо изменений.

Julius [28] – программное обеспечение с большим встроенным словарем и подключенным декодером непрерывного распознавания речи, высокопроизводительная платформа с двухпроходной проверкой [29] по непрерывному распознаванию речи.

Он поддерживает большое лексическое распознавание N-граммы, синтаксический анализ на основе грамматики, детерминированный конечный распознаватель (Deterministic finite automaton, DFA) [30] и распознавание отдельных слов. В зависимости от контекста, фонемный анализ поддерживаются вплоть до трифона. Трифон в лингвистике – это последовательность из трех фонем, следующих друг за другом [31]. Помимо этого, Julius может выполнять мультимодальное декоди-

рование, т.е. имеет возможность восстанавливать исходную информацию из одного общего мультимедиа в исходные средства информации (модусы) до их совмещения [32, 33], а также поддерживает горячее подключение (hot plugging), т.е. подключение или замену произвольных модулей во время выполнения рабочих процессов [34].

Акустические и языковые модели являются подключаемыми и отключаемыми, что предоставляет возможность собрать и сформировать различные варианты систем по распознаванию речи, подобрав необходимые модели и модули, подходящих для требуемой задачи. Он также использует стандартные форматы для возможности использования сторонних инструментов. Основной механизм программного обеспечения реализован как встраиваемая библиотека для обеспечения возможности распознавания речи различными приложениями. В последней версии была добавлена поддержка модульности для программного расширения, чтобы пользователь мог настраивать систему под свои нужды.

RASR (Rheinisch Automatic Speech Recognition) [35] – программный пакет, содержащий декодер распознавания речи с возможностью обучения и декодирования акустической модели. Он также включает в себя подстраивание под конкретного диктора, самостоятельное обучение, библиотеку автоматики конечных состояний (finite state automata library) [36] и эффективный древовидный декодер поиска (efficient tree search decoder) [37]. RASR извлекает параметры аудиоданных и предварительно обрабатывает для извлечения нормализованных акустических признаков, получая кепстральные коэффициенты (MFCC). Модель словаря получается путем применения фонетической транслитерации к учебной лексике.

Версия RASR с открытым исходным кодом хорошо зарекомендовала себя широким инструментарием по распознавания речи. Ключевой особенностью является высокая гибкость относительно топологии сети, выбор критериев обучения и алгоритма оптимизации. Оценка рабочих характеристик выполняется с помощью обученной нейронной сети. Результаты показывают, что RASR достигает высоких

показателей по распознаванию, с возможностью полного встраивания и применения в крупномасштабных системах распознавания речи [38].

Kaldi – это набор инструментов с открытым исходным кодом для распознавания речи, написанный на языке C++. Инструменты компилируются на широко используемых Unix-подобных системах и в Microsoft Windows [39].

К важным особенностям Kaldi относятся:

- обширная поддержка линейной алгебры;
- расширяемый дизайн;
- открытая лицензия;
- тщательное тестирование;
- готовые решения для построения систем распознавания речи.

iATROS – это усовершенствованная версия предыдущей системы по распознаванию речи (ATROS), которая может использоваться как в распознавании речи, так и в распознавании рукописного текста [40]. iATROS предоставляет модульную структуру, которая может быть использована для построения различных систем с применением скрытых марковских моделей (Hidden Marks Model) [41]. Программный пакет предоставляет стандартные инструменты для автономного распознавания и интерактивного распознавания речи (на основе модулей ALSA [42]).

iATROS состоит из двух модулей предварительной обработки и выделения признаков (для изображений речевого сигнала и рукописного ввода) и модуля распознавания. Модули предварительной обработки и извлечения элементов предоставляют векторы элементов для модуля распознавания, который, используя модели HMM и языковые модели, выполняет поиск наилучшей гипотезы распознавания. Все модули реализованы в C.

Среди представленных систем с открытым исходным кодом был проведен сравнительный анализ по критериям точности и скорости распознавания. Первый критерий подразделяется на две составляющие: процент верно распознанных слов

(WRR – Word Recognation Rate) и процент неверно распознанных слов (WER – Word Error Rate) [43], которые определяются как:

$$WER = \frac{I + S + D}{T} \cdot 100\%; \quad (1)$$

$$WRR = 100\% - WER, \quad (2)$$

где I – количество процедур ведения слова, S – количество процедур замены слова, D – количество процедур по удалению слова, T – общее количество слов во фразе.

Количественная оценка временных характеристик распознавания систем с открытым исходным кодом была проведена с применением критерия показателя скорости распознавания (SF – Speed Factor) [43], который определяется как:

$$SF = \frac{T_{\text{p}}}{T}, \quad (3)$$

где T_p – время распознавания сигнала, T –продолжительность сигнала.

По результатам расчетов с использованием 170 звуковых образцов, была составлена таблица для наглядного представления выходных данных эксперимента (Таблица 1).

Таблица 1 – сравнительный анализ по критериям точности и скорости распознавания

Платформа	**WER, %**	**WRR, %**	**SF, с**
Kaldi	6,5	93,5	0,6
RASR	15,5	84,5	3,8
iAtros	16,1	83,9	2,1
CMU Sphinx	21,4	78,6	0,5
Julius	23,1	76,9	1,3

По результатам проведённого сравнительного анализа [43], представленного в таблице 1, наивысшим результатам в верном распознавании слов является платформа Julius. Помимо этого, данная система показывает один из лучших показателей в скорости обработки и распознавании речи. Алгоритм работы представленной системы может быть использован в разработке ПО для установления интонационных конструкций в данной работе. Для эффективной работы разрабатываемого приложения требуется составление большой библиотеки звуковых образцов, в обратном случае корректность результатов идентификации будет считаться невысокой в сравнении с системами с закрытым исходным кодом.

1.3 Распознавание речи

Распознавание речи – это способность устройства или программы идентифицировать записанные слова и фразы и конвертировать их в машиночитаемый формат. Модульное программное обеспечение распознавания речи имеет ограниченный словарь слов и фраз, и оно может идентифицировать их только в том случае, если они произнесены очень четко. Более сложное программное обеспечение имеет возможность принимать естественную речь.

Распознавание речи работает с использованием алгоритмов посредством акустического и языкового моделирования. Акустическое моделирование представляет взаимосвязь между лингвистическими единицами речевого и звукового сигналов [44].

Несмотря на то, что технология распознавания речи удобна, в ней по-прежнему существуют проблемы, над решением которых продолжается работа. Преимущество программного обеспечения распознавания речи в том, что оно является простым в использовании и легкодоступным. Программное обеспечение распознавания речи в настоящее время часто предустанавливается на компьютерах и мобильных устройствах, что обеспечивает легкий доступ.

На сегодняшний день нерешенными проблемами в распознавания речи являются: неспособность улавливать слова из-за вариаций произношения, отсутствие

поддержки большинства языков за пределами английского и его неспособность распознавания через фоновый шум. Эти факторы в большинстве приводят к ошибкам и неточностям распознавания.

Классификация систем распознавания речи:

- по размеру словаря;
- по зависимости от диктора;
- по типу речи;
- по назначению;
- по используемому алгоритму;
- по типу структурной единицы;
- по принципу выделения структурных единиц.

Методы и алгоритмы распознавания речи:

- классификация методов распознавания на основе сравнения с эталоном;
- динамическое программирование;
- контекстно-зависимая классификация.
- методы дискриминантного анализа Байесовской дискриминации [45];
- скрытые марковские модели [41];
- нейронные сети.

Структура систем распознавания:

- модуль очистки от шума и выделения полезного сигнала;
- акустическая модель оценки на предмет звукового сходства;
- языковая модель предсказания последовательности слов;
- декодеринг для программного преобразования сигнала.

Спектрально-временные признаки:

- среднее значение спектра анализируемого речевого сигнала;
- нормализованные средние значения спектра;
- относительное время пребывания сигнала в полосах спектра;

• нормализованное время пребывания сигнала в полосах спектра;

• медианное значение спектра речи в полосах;

Кепстральные признаки:

• мел-частотные кепстральные коэффициенты;

• коэффициенты линейного предсказания с коррекцией на неравномерность чувствительности человеческого уха;

• коэффициенты спектра линейного предсказания;

• коэффициенты кепстра линейного предсказания.

Амплитудно-частотные признаки:

• интенсивность, амплитуда;

• энергия;

• частота основного тона;

• формантные частоты.

Основные понятия:

• разборчивость речи – «количество правильно принятых элементов речи (звуков, слогов, слов, фраз), выраженное в процентах от общего числа переданных элементов»;

• качество речи – «параметр, характеризующий субъективную оценку звучания речи в испытуемой системе передачи речи»;

• нормальный темп речи – «произношение речи со скоростью, средняя длительность контрольной фразы равна 2,4 с»;

• ускоренный темп речи – «произношение речи со скоростью, средняя длительность контрольной фразы равна 1,5-1,6 с»;

• узнаваемость голоса говорящего – «возможность слушателей отождествлять звучание голоса, с лицом, известным слушателю ранее»;

• смысловая разборчивость – «показатель степени правильного воспроизведения информационного содержания речи».

1.4 Цифровая обработка сигналов

Цифровая обработка сигналов – «это методика, которая работает на базе численных методов и сопровождается применением цифровой вычислительной техники» [46].

Существует классификация, в соответствии с которой методы обработки сигналов можно разделить по следующим направлениям: временная и частотная области. Аналоговый сигнал подвергается дискретизации по квантованию и времени по оцифровке. Он выражается в цифровой форме. Можно сделать заключение, что эквивалентность частотно-временных преобразований устанавливается через преобразование Фурье [47].

Во временной области обработка сигналов активно применяется в цифровых осциллографах и электронной осциллографии. Чтобы сигналы можно было наглядно выразить для дальнейшего анализа и работы с ними, в частотной области применяются цифровые анализаторы спектра. С целью изучения математических особенностей обработки сигналов применяются расширения (в их виде представлены своеобразные пакеты, которые обозначаются Signal Processing) систем компьютерной математики Octave, MATLAB, Mathcad, Maple, Mathematica.

При этом устанавливаются задачи, которые необходимо решать в процессе работы. Среди них:

- линейная фильтрация – селекция (отбор) сигнала; синтез (организация наличия) фильтров, которые будут комбинироваться с сигналами; разделение каналов по частотам [48]; цифровые преобразователи Гильберта ($L^n(a, b)$), дифференциаторы; специальные корректоры свойств каналов;
- проведение анализа. Спектральное направление предполагает обработку звуковых, сейсмических, речевых, гидроакустических сигналов; выполняется распознавание образов.

• адаптивная фильтрация – процесс и алгоритм распознавания речи, графических объектов, определенных образов, устранение возникающих шумов, адаптивные антенные решётки;

• частотно-временной анализ. В рамках такового выполняется гидро- и радиолокация, сжатие графических объектов, выполняются другие задачи поиска сигнала;

• многоскоростная обработка. Предполагает интерполяцию (рост) и децимацию (снижение показателей) частоты дискретизации. Актуально для функционирования многоскоростных систем телекоммуникации, аудио;

• нелинейная обработка. Осуществляется расчет корреляций, медианная фильтрация; создание фазовых, амплитудных, частотных детекторов, анализ и подготовка речи, векторное кодирование;

• секционная свёртка;

• свёртка традиционных типов;

• поиск сигнала. На этом шаге необходимо найти сигнал, идентифицировав его среди помех и различных шумов;

• оценка качества и структуры сигнала. Необходимо для установления актуальных характеристик сигнала, среди которых такие критерии, как: частота, амплитуда, фаза);

• различение сигнала. Выполняется распознавание сигнала среди других, для которых свойственны похожие характеристики.

Основные преобразования:

1. Цифровая обработка сигнала в передатчике:

• форматирование;

• кодирование источника;

• шифрование;

• канальное шифрование;

• импульсная модуляция;

- полосовая модуляция;
- расширение спектра;
- передача сигналов.

2. Распространение сигналов по каналу связи.

3. Цифровая обработка сигнала в приёмнике:

- приём сигналов;
- сужение спектра;
- демодуляция и дискретизация;
- канальное декодирование;
- расшифрование;
- декодирование источника;
- форматирование.

1.5 Анализ метода мел-частотных кепстральных коэффициентов применительно к процедуре голосовой аутентификации

В настоящее время описаны и используются множество различных математических моделей обработки сигналов, многие из которых с успехом применяются для речевых сигналов. Для представления речевого сигнала и получения его амплитудного спектра в частотной области (и вытекающих из него характеристик) чаще других применяется преобразование Фурье [47]:

$$s(\omega) = \int_{-\infty}^{\infty} s(t) e^{-i\omega t}\, dt; \tag{4}$$

$$s(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} s(\omega) e^{i\omega t} d\omega, \tag{5}$$

где функция *s(t)* непрерывна и интегрируема, *s(ω)* – спектральная характеристика *s(t)*.

В дискретном виде [49]:

$$S(n) = \sum_{k=0}^{N-1} s(k) e^{-i\frac{2\pi}{N}nk}; \quad (6)$$

$$s(n) = \frac{1}{N} \sum_{k=0}^{N-1} S(k) e^{i\frac{2\pi}{N}nk}. \quad (7)$$

Для представления речевого сигнала в параметрическом виде используют кратковременный спектр, который можно рассматривать как текущий спектр с реальным временем [50-52]:

$$s(\omega, t) = \int_{-\infty}^{t} s(\tau) h(t-\tau) e^{-i\omega\tau} d\tau = a(\omega, t) - ib(\omega, t), \quad (8)$$

где $h(t)$ – временное окно, используемое для анализа сигнала $s(t)$ в текущий момент времени t через круговую частоту ω.

На практике, однако, чаще используют анализ амплитудного $|S(\omega,t)|$ или фазового $F(\omega,t)$ спектра вместо анализа кратковременного спектра $s(\omega,t)$:

$$|S(\omega, t)| = \sqrt{\left(\int_{-\infty}^{t} s(\tau) h(t-\tau) cos\omega\tau d\tau \right)^2 + \left(\int_{-\infty}^{t} s(\tau) h(t-\tau) sin\omega\tau d\tau \right)^2}; \quad (9)$$

$$F(\omega, t) = arctg \left(\frac{\int_{-\infty}^{t} s(\tau) h(t-\tau) sin\omega\tau d\tau}{\int_{-\infty}^{t} s(\tau) h(t-\tau) cos\omega\tau d\tau} \right). \quad (10)$$

Спектральный анализ, выполняемый на базе Фурье-представлений, характеризуется многими преимуществами, однако здесь тоже есть несколько минусов. Обозначим их в следующих тезисах:

- усложнено нахождение формант в условиях неверного выбора окна;
- есть некоторые ограничения на отражения фазовой структуры сигнала;

• наблюдается некорректное отображение в процессе передачи по каналу связи [53].

Использование методов линейного предсказания (ЛП) при анализе сигнала способствует более быстрому выполнению расчетов. Если представить речевой сигнал как линейную комбинацию конечных предшествующих отсчетов:

$$s(t) = \int_{-\infty}^{\infty} a(x)s(t-x)dx + G\delta(t) \quad (11)$$

или в дискретной форме:

$$s(n) = \sum_{k=-\infty}^{\infty} a_k\, s(n-k) + G\delta(n), \quad (12)$$

где $G\ \delta(t)$ – источник возбуждения с коэффициентом усиления G. Тогда линейный предсказатель и погрешность предсказания имеют вид соответственно:

$$S_\alpha(n) = \sum_{k=-\infty}^{\infty} \alpha_k s(n-k); \quad (13)$$

$$e(n) = s(n) - s_\alpha(n) = s(n) - \sum_{k=-\infty}^{\infty} \alpha_k s(n-k). \quad (14)$$

Однако использование моделей на основе методов линейного предсказания не лишено недостатков [54]:

• возникает затруднение при выборе порядка модели;

• результаты, полученные от линейного предсказания, не всегда удовлетворяют условию устойчивости;

• низкая надежность при использовании в системах с несколькими дикторами;

• модель не отражает влияние фазовых характеристик речи.

Для вычисления фазовых параметров удобно использовать преобразование Гильберта [55-57]. Если $A(t)$ – огибающая амплитуда, а $\varphi(t)$ – полная фаза, тогда речевой сигнал имеет вид:

$$s(t) = A(t)cos\varphi(t) \tag{15}$$

Если $s(t)$ представить в виде действительной части комплексного сигнала $z(t)$, то получим [54]:

$$z(t) = s(t) + is_c(t), \tag{16}$$

где $s_c(t)$ – функция, сопряженная с $s(t)$.

Согласно преобразованию Гильберта, прямая и сопряженная функции связаны соотношениями:

$$s_c(t) = -\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{s(x)}{x-t}ds; \tag{17}$$

$$s(t) = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{s_{c(x)}}{x-t}dx. \tag{18}$$

Подставляя формулы (12) и (14) в (13) получаем:

$$z(t) = A(t)cos\varphi(t) - \frac{i}{\pi}\int_{-\infty}^{\infty}\frac{A(t)cos\varphi(t)}{x-t}dx. \tag{19}$$

При использовании преобразования Гильберта не составляет труда найти амплитудные и фазовые спектры имея $s(t)$ восстановить $z(t)$ и $s_c(t)$. Однако, как правило, в математическом плане выражение $\varphi(t) = arctg\frac{s_c(t)}{s(t)}$ не является определенным в широком диапазоне времени и частоты, это затрудняет применение прямого подхода к вычислениям начальной фазы речевого сигнала [54].

Для построения систем распознавания речи предпочтение дается методам, точность идентификации которых не зависит от диктора:

- скрытые марковские модели (Hidden Markov model – HMM) [41];

• методы распознавания диктора, основанный на «машине опорных векторов» (Support Vector Machine– SVM) [58].

При использовании алгоритмов и методов НММ получают решение приведенные далее концепции:

• определение вероятности наблюдения $P(O|\lambda)$ определенной последовательности символов $O=O_1, O_2,\ldots,O_T$ (работает в данном случае модель;

• определение наиболее оптимальной последовательности состояний $Q=q_1, q_2,\ldots,q_T$ последовательности символов $O=O_1, O_2,\ldots,O_T$, (работает модель);

• определение параметров модели, при которой работает анализ вероятности $P(O|\lambda)$ (максимальный показатель).

При составлении НММ используется не одна методика. Предлагаем рассмотреть несколько актуальных способов работы над решением задач [54, 59-61]:

• сбор и создание модели фонем (при использовании минимальных языковых единиц);

• моделирование фонем тремя состояниями (начальное, среднее, конечное);

• моделирование разных типов фонем (монофоны и трифоны) на основании эффекта коартикуляции;

• для распознавания отдельно стоящих слов составляются НММ для каждого слова;

• для распознавания слитной речи составляют одну НММ, склеивая через промежуточные состояния.

Метод НММ имеет ограниченное применение и позволяет распознавать голос только в строго ограниченных условиях. Это объясняется тем, что в среде канала связи возникают различные помехи. Например, если нужно идентифицировать диктора и распознать различные языковые искажения.

При обработке речевых сигналов в естественно-языковых приложениях получили широкое распространение методы, основанные на алгоритме SVM. На прак-

тике эти методы демонстрируют высокую эффективность. Среди главных преимуществ алгоритма можно назвать такие, как:

- гарантия получения единственного решения, благодаря решению минимизации выпуклой функции. По сравнению с нейронными сетями – это серьезное преимущество. В нейронных сетях ответ может быть неопределенный или локальный минимум;
- алгоритм отличается повышенной устойчивостью к зашумленности исходящего сигнала и позволяет распознать речь;
- алгоритм эффективно работает с данными больших размерностей. Это важно для определения речи в тех случаях, когда вектор признаков достигает сотен или тысяч.

В современной науке в качестве альтернативы традиционным методам, все чаще используются методы, в основе которых используются мел-частотные кепстральные коэффициенты (MFCC). Среди преимуществ этого метода: способность во время идентификации субъекта учитывать индивидуальные характеристики его голосового тракта. Метод отличается малым размером и позволяет получать вектор независимо от длины образца [62].

При анализе речевого сигнала проводится подготовка, которая позволяет выделить характерные признаки. Для примера возьмем слово «привет», которое произнесено русскоговорящим диктором мужского пола. Используя этот образец, был произведен речевой анализ с использованием упомянутых выше методов (рисунок 1) [63]:

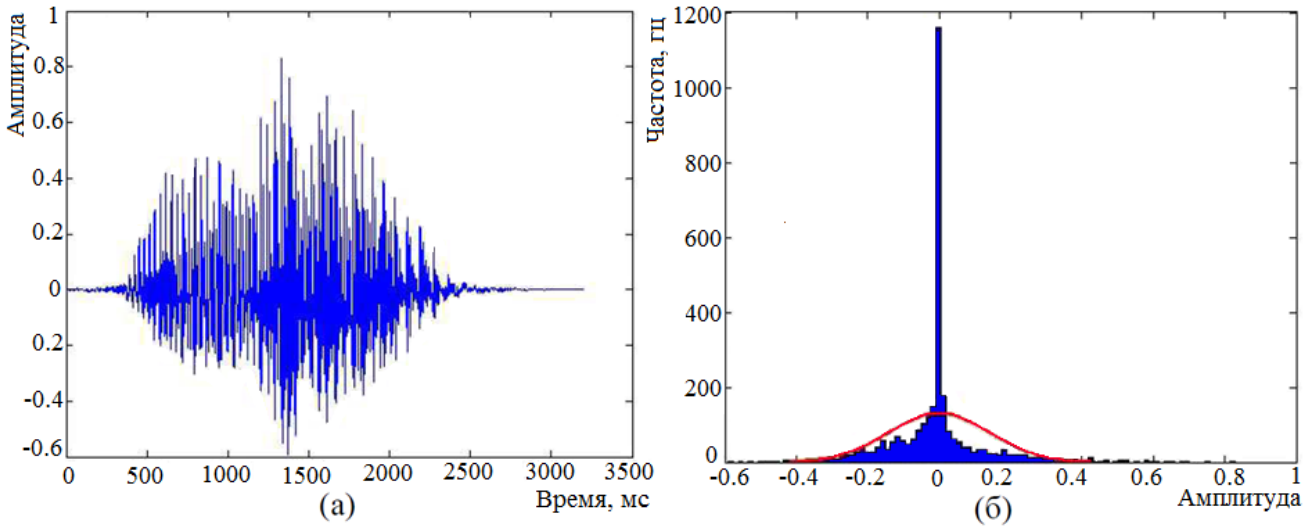


Рисунок 1 – Звуковой файл со словом «привет», произнесенный диктором мужчиной (а) и его гистограмма (б) [63]

Во время подачи сигнала на вход требуется устранение неинформативного шума и искажений в рамках предварительной обработки. Для данной корреляции используют фильтр высоких частот с передаточной функцией:

$$F_{\text{ВЧ}} = \frac{1 - z^{-1}}{1 - az^{-1}}, \tag{20}$$

где коэффициент $a \to 1$, на практике для a используют значение $a = \frac{127}{128}$;

Нормализованный сигнал имеет вид:

$$S'[n] = \frac{\Delta}{\max\limits_{m} |S[m]|} S[n], \tag{21}$$

где коэффициент Δ определяет ширину полосы пропускания. Ширина полосы пропускания должна быть симметрична относительно оси горизонтальной оси координат.

Если говорить об опытных системах, где применяется одно записывающее устройство, в которых будет произносить звуки человек, зачитывающий текст, нормализация сыграет небольшую роль. Инструменты, которые рекомендуется использовать для последующей обработки:

• удаление неинформативных фрагментов, не несущих смысловой и другой ценности (так называемых пауз). Это позволяет получить более качественный результат и упростить последующее решение задач по подготовке звука в целом;

• кадрирование. Непрерывный речевой сигнал не подходит под требуемые условия в программной среде. Актуальной будет разбивка речевого сигнала на участки установленной продолжительности для последующей обработки.

К кадрированному и оцифрованному сигналу применяется весовая функция с целью уменьшения искажения из-за конечности выборки.

Итоговыми значениями становятся последние коэффициенты, которые отвечают за интонацию речи конкретного человека. После описанных выше преобразований речевого сигнала из огромного количества сигналов получается небольшая выборка важных для анализа значений, ведь шумы и прочая ненужная информация отсеивается посредством указанных алгоритмов.

Выдающимся плюсом методики MFCC считается простота реализации при высоком качестве распознавания голосового потока. Данный метод держится на одном уровне с другими популярными методиками и алгоритмами.

Выводы по разделу 1. Были рассмотрены системы распознавания речи с закрытым и открытым системным кодом, их принципы работы, достоинства и недостатки. Разобраны термины, классификации, методы и алгоритмы, связанные с преобразованием речи. Рассмотрены достоинства и недостатки формантного анализа и предложена альтернатива существующим методам – использование мелчастотных кепстральных коэффициентов.

2 УСТАНОВЛЕНИЕ ОПТИМАЛЬНОГО РЕШЕНИЯ НА ОСНОВЕ СУЩЕСТВУЮЩИХ АЛГОРИТМОВ И МЕТОДИК РАСПОЗНАВАНИЯ РЕЧИ

Большинство методов классификации акцентов основаны на моделях, зависящих от акцентов, с использованием общего набора признаков [64] или различии на основе признаков [65, 66].

Для быстрой классификации акцентов с использованием небольшого количества данных в основном используются модели, основанные на фонемах для распознавания.

Набор фонем делится на шесть классов:

1) стопы (согласные, явно оканчивающие фонемы, /p, t, k / или / b, d, g /, длящиеся 15 или более 15 мс);

2) рикативы (гортанные звуки);

3) фрикативы (звук, образующийся от трения воздушной струи, выходящей из легких и встречающей на своем пути сильное сужение в тех или других органах речи);

4) назалы (носовые звуки);

5) полуслова и гласные;

6) гласные.

Исследуются следующие особенности и их первая и вторая производные для их влияния на акцент: основная частота (F0), энергия в среднеквадратическом значении (E0), первая частота формант(F1), частота второй форманты(F2), форманта третьей частоты (F3), и полосы пропускания F1, F2 и F3, B1, B2 и B3 соответственно. Формантами называются резонансные максимумы речевого тракта на определенных частотах.

Непрерывная речь сэмплируется на частоте 44,1 – 48 кГц, выполняется высокочастотная предварительная обработка искажений (удаление шумов, пустот),

обработка через окно Хэмминга. "Окна" – это инструмент, используемый для обработки дискретных данных и анализа спектра (частотной области).

Классификация основана на последовательности трех структур скрытых марковских моделей, имеющих одну гауссову плотность.

Базовая система построена с использованием всех 24 просодических характеристик. Просодические характеристики– это особенности языка, такие как ударение, высота тона, изменения высоты тона, интонация, стыки, паузы и т.д. Базовая производительность (эффективность, результативность) составляет 85,49% и 82,5% для закрытого и открытого теста соответственно. Маскируя одну особенность за другой, исследуется ее влияние на классификацию акцентов в обучающем наборе. Для классификатора (в нашем случае эталона) используется лучшая комбинация признаков.

В результате получаем характеристики в порядке важности, первостепенности, чтобы выделить различные классификации на: dd(E) d(E), E, d(F3), dd(F3), F3, B3, d(F0), F0, dd(F0), где E – энергия, F3 является третьей формант, B3 – полоса пропускания третьей форманты, d – это первые производные и dd – вторые производные.

Нулевая, самая низкая по частоте форманта F0 определяется частотой основного тона. Первая форманта F1 коррелирует с подъемом гласного: чем выше подъем, тем ниже частота первой форманты. Вторая форманта F2 соотносится с рядом гласного – чем более передним является гласный, тем она выше [67].

2.1 Энергия

Энергия – важная особенность, которая может показать различия в стиле и структуре общения на двух разных языках. Таблица 2 представляет среднюю энергию фоновых классов между двумя группами акцентов [68]. Средняя энергия выше для носителей английского языка во всех классах. Разница также намного выше для носителей английского языка во всех классах, кроме африкатов. Это говорит о том, что энергетический диапазон для носителя языка выше.

Таблица 2 – Средняя энергия различных фоновых классов [68]

Классификация фонем	**Английская**	**Кантонская**
Гласные	1035,1	506,65
Носовые согласные	601,7	252,28
Взрывные согласные	147,74	55,63
Аффрикаты	370,66	89,87
Фрикативные согласные	224,79	117,99
Полугласные	1282,49	522,93

Носовые (назальные) согласные звуки производятся блокированием воздуха во рту и изданием звука через нос.

Взрывные согласные – это звук, создаваемый путем полного блокирования потока воздуха и последующего его освобождения.

Аффрикаты – это согласные, начинающиеся с полного блока воздуха, переходящие во фрикативы (частичный блок).

Фрикативные согласные получаются путём выпуска воздуха через небольшой зазор между артикуляторами.

Полугласные подобны гласным, но произносимы в согласной позиции, характеризуются короткой длительностью и быстрым движением артикулятов.

2.2 Форманты

Вторым важным параметром является третий формант вместе с его производными. Форманта – это акустическая характеристика звука речи (главным образом гласного), связанная с уровнем частоты голосового тона и образующая тембр звука. Форманта представляет собой концентрацию акустической энергии вокруг конкретной частоты в речевой волне. Информация, которая требуется человеку для различия гласных звуков, может быть выражена количественно (в Гц).

Арслан и Хансен, написавшие учебник о классификации акцентов языка в Американском Английском [65], предполагают, что F2 и F3 чувствительны к ак-

центам, так как их положения смещаются в соответствии с движениями языка. Предполагается, что движения языка являются наиболее заметной разницей между носителями и не носителями языка. Однако в [65] источнике показывается, что использование F2 и F3 для классификации акцентов хорошо работает для европейских акцентов, но не для азиатских акцентов. В экспериментах было обнаружено, что только формантная позиция и полоса пропускания F3 важны для классификации между родным и гонконгским английским акцентами, а не F2. Рисунок 2.1 показывает, как особенности формант могут повлиять на результаты классификации [68].

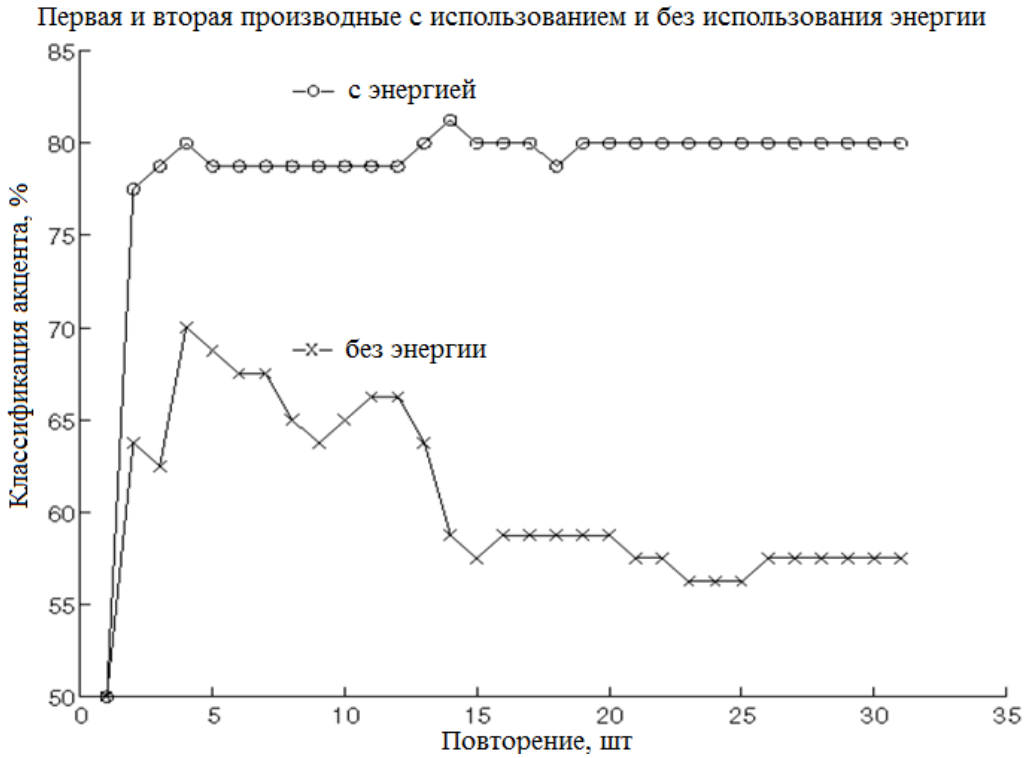


Рисунок 2.1 – Точность классификации акцентов с / без характеристики энергии [68]

Таблица 3 – Ошибка классификации, вызванная особенностями формант [68]

Особенность классификации	**Коэффициент ошибки, %**
Полный набор параметров	14,52
Без F1	14,14
Без d(F1)	14,34
Без dd(F1)	14,13
Без F2	14,21

Продолжение таблицы 3 [68]

Особенность классификации	Коэффициент ошибки, %
Без d(F2)	14,37
Без dd(F2)	14,22
Без F3	15,13
Без d(F3)	15,14
Без dd(F3)	15,14
Без B1	14,51
Без d(B1)	14,51
Без dd(B1)	14,36
Без B2	14,08
Без d(B2)	14,39
Без dd(B2)	14,33
Без B3	14,72
Без d(B3)	14,33
Без dd(B3)	14,46

Результаты, на которых основана данная таблица были получены в эксперименте, представленном в статье [68], который проводился в центре технологий изучения человеческого языка, факультета электротехники и электронной техники Научно-технического университета Гонконга.

2.3 Просодические особенности при классификации акцентов

Тесты человеческого восприятия показывают, что слушатели основывали свои решения о классификации акцентов частично на просодических особенностях, таких как тон, ритм и пауза [66]. Было определено, что контуры тона кантонских (один из диалектов упрощенного китайского) дикторов более прерывистые. Этот результат также может подтверждаться тем фактом, что среднее количество исчисляемых (счетных) озвучиваемых областей больше, а средняя продолжительность каждой озвучиваемой области меньше для кантонских говорящих. Кантонский язык односложен, так как слоги на кантонском языке состоят из начального

и конечного. На рисунке 2.2 изображено, как носители осуществляет свою первую попытку произношения на иностранном языке [68].

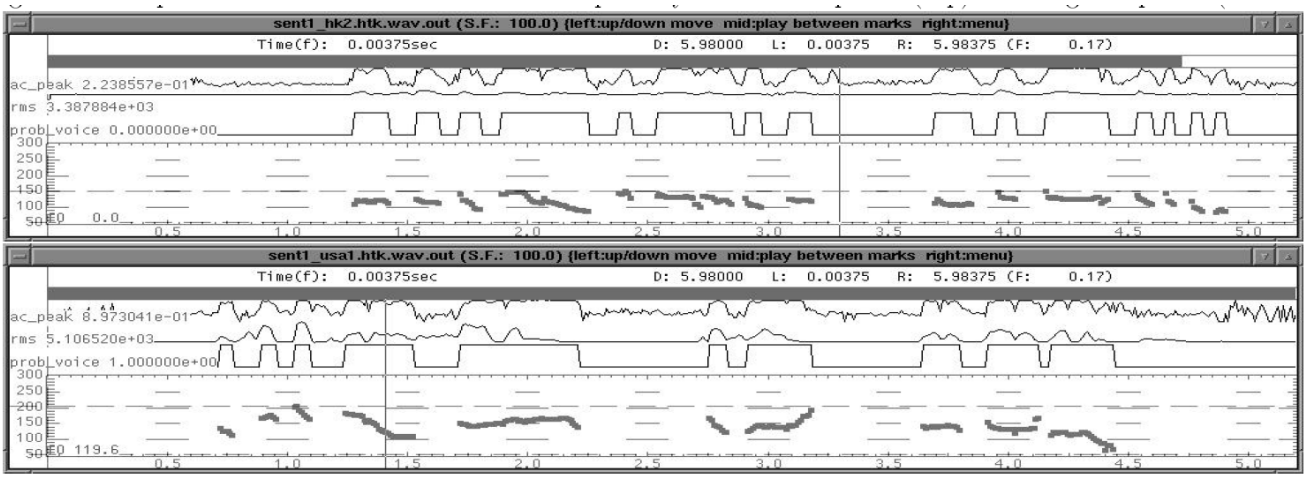


Рисунок 2.2 – Контур тона одного и того же высказывания, произнесенного кантонским говорящим (сверху) и английским говорящим (снизу) [68]

В таблице 4 представлены результаты влияния параметров F0 на коэффициент ошибки классификации.

Таблица 4 – Влияние параметров F0 на коэффициент ошибки классификации

Набор параметров	**Коэффициент ошибки, %**
Полный набор	14,52
Без dd(F0)	14,58
Без F0	14,65
Без d(F0)	14,67
Без информации F0	15,33

В эксперименте было обнаружено, что если игнорировать F0, его первые производные и его вторые производные маскируются, и тогда частота ошибок классификации акцентов увеличивается на 5,6%.

2.4 Распознавание акцентированной речи на основе знания родного языка

Анализ просодической (просодия, также просодика – наука об ударении, занимающаяся слогами с точки зрения их ударности и протяженности) информации показывает только акустические различия между различными акцентными группами. Эти функции, будучи эффективными для классификации акцентов, трудно использовать в обработке акцентов. Обратимся к еще одному важному различию в произношении между носителями и не носителями языка, рассмотренном на Инженерном факультете Кембриджского университета, в использовании специфичного для акцента моделирования произношения для улучшения непрерывного распознавания речи с большим словарным запасом [65].

Было установлено, что эффективно включать правила произношения с акцентом в словарь для распознавания. Фонема A в речи носителя языка может быть сопоставлена с фонемой B в речи неносителя языка.

Информацию о таких правилах отображения можно получить из трех источников.

Первый источник из положения фонем F1-F2. На рисунке 2.3 изображено значение разницы F1 и F2 частот гласных на родном американском английском и кантонском английском с акцентом. Этот метод может показать степень различий фонемы между двумя классами акцентов, и в каком направлении фонема движется к другому. Например, звук UW для носителей кантонского языка находится далеко от звука носителя американского языка и движется в направлении звука OW. Звук AA в обеих группах больше всех накладывается друг на друга. Такой же результат обнаруживается и на выходе распознавания фонем. Однако этот метод не может показать исключение фонемы и ее внесение, только ее замещение. Это может быть использовано в методах адаптации акцента на основе преобразования.

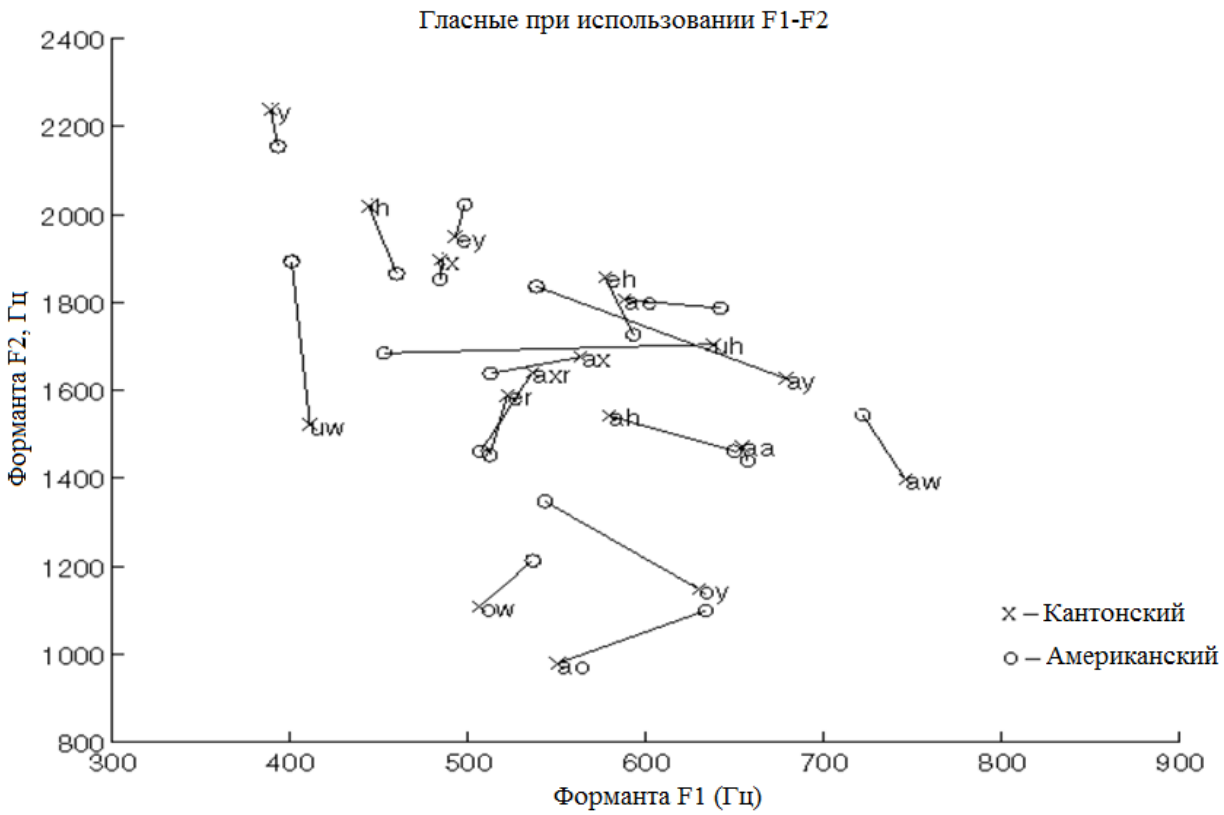


Рисунок 2.3 – Диаграмма первой форманты против второй форманты гласных [68]

Результатами таблицы, составленной в ходе эксперимента, проведенного в центре технологии лингвистики [68], подтверждается, что при разработке конечного алгоритма анализа речевых образцов не-носителей целевого языка требуется как учёт особенностей ошибок, допускаемых не-носителями, так и допустимых пределов произношения слова носителями языка. Также при разработке алгоритма требуется учесть, что он должен выполнять не только функцию непосредственной (таблица 5).

Таблица 5 – Правильность произношения групп обучаемых при использовании базового словаря английского языка и словаря, адаптированного к Кантонскому [68]

Диктор	Базовый словарь, %	Адаптированный словарь, %
Обучаемый 1	74,07	75,93
Обучаемый 2	64,81	69,44
Обучаемый 3	79,37	76,85

Продолжение таблицы 5 [68]

Диктор	Базовый словарь, %	Адаптированный словарь, %
Обучаемый 4	67,59	71,30
Среднее	69,21	73,38

Второй метод основан на анализе данных. Неродная акцентная речь передается распознавателю фонем родного акцента, и результатом может быть беспорядочная матрица, показывающая общее отображение фонем между двумя наборами. Наиболее приводящими к путанице звуками в Гонконгском английском языке являются /R/, /AXR/, /P/, /ER/, /K/ и /G/. Эти звуки либо отсутствуют, либо редко встречаются на кантонском языке.

Третий метод – метод получения правил отображения с использованием знаний лингвистов. Этот метод является самым быстрым и простым для создания специфичного для словаря, заточенного под акцент (определенного конкретно под него). Более того, такие знания хорошо изучены и в меньшей степени зависят от данных. Более того, лингвисты показали, что есть некоторые звуки, которые не встречаются в кантонском языке, такие как /AXR/, /AX/, /AE/, /IH/, /AH/ и /UH/. В нашей системе мы применяем 28 фонетических правил к электронному словарю (BEEP) (протокол обмена расширяемых блоков Blocks Extensible Exchange Protocol) [69], предназначенному для носителей английского языка. Размер словаря удваивается. Результаты распознавания акцентированной речи лучше с помощью адаптированного к акценту словаря, чем с помощью базового словаря, что дает в среднем снижение частоты ошибок более чем на 4% (см. Таблица 5 – Правильность произношения групп обучаемых при использовании базового словаря английского языка и словаря, адаптированного к Кантонскому).

Показано, что в целом энергия, форманты и основная частотная информация являются наиболее отличительными признаками для идентификации кантонского (и, возможно, других азиатских) акцентов. Также определено, что, в отличие от европейских акцентов, только F3 вместо F2 и F3 [65] указывает на кантонский (и,

возможно, другой азиатский) акцент. Также выявлена, возможность сократить частоту ошибок при помощи использования лингвистического словаря произношения.

При обучении целевому языку, люди ориентируются на фонетическую систему их родного языка, в которой могут отсутствовать особенности, присущие целевому языку. Так, например, в родном языке обучающегося может отсутствовать часть звуков целевого, твёрдость-мягкость или звонкость-глухость могут не иметь смыслоразличительной функции, а интонации – играть иную роль. Кроме того, у обучающихся уже сложилась артикуляционная база звуков родного языка, поэтому они пытаются приспособить свою артикуляцию для произнесения новых для них звуков. Таким образом, происходит интерференция – влияние родного языка на целевой, что приводит к ошибке.

На вход системы подаётся либо уже готовый файл с речевым образцом, либо запись с микрофона, минуя запись в файл и чтение из него. В первом случае нам необходимо извлечь данные из того формата, в котором они записаны (например, в .mp3 или .wav). Во втором случае мы собираем данные с микрофона, без декодирования из файла, а файл сохраняется отдельно для дальнейшего использования.

Определяющей информацией о сигнале являются: форма звуковой волны во временной области (speech waveform), частота дискретизации, глубина кодирования, количество каналов. При чтении из файла базы данных также должна прилагаться вспомогательная и служебная информация о файле и результатах его обработки человеком-специалистом или самой экспертной системой. В такую информацию могут входить: личная информация о дикторе (такая как пол, возраст, родной язык), оценка правильности произношения, составленный «профиль» слова и прочие маркеры. В случае поступления новой записи с микрофона информация может быть частично взята из профиля пользователя или вычислена и записана на дальнейших этапах.

2.5 Обработка речевого сигнала

Поступившие данные требуется обработать. Основным материалом является «сырая» волна (raw waveform) [70] – непосредственный результат извлечения звука из файла или с микрофона. В том случае, если запись не нормализована, её требуется нормализовать (см. «Нормализация сигнала»). В общем случае, в предварительную обработку входят такие процедуры, как фильтрация шумов, кадрирование, сегментация, удаление тишины и т.д. [71]

Чтобы работать с сигналом дальше, его требуется нормализовать по амплитуде. Нормализация звука по амплитуде – это выравнивание громкости сигнала. В данном случае используется пиковая нормализация, исключающая искажения сигнала.

В этом случае запись, содержащая условную «тишину» (на самом деле – слабый фоновый шум), также будет выровнена по своему максимальному значению, предлагая в дальнейшем анализировать этот шум. Этого можно избежать, если ввести безусловный коэффициент нормализации – некоторый уровень, ниже которого нормировочный коэффициент быть не может.

Например, если требуется предотвратить нормализацию сигнала, содержащего только шум. Предположим, что «тишиной» будет считаться 20% от максимально допустимой громкости. Тогда минимальный нормировочный коэффициент будет рассчитан как:

$$K_{\text{норм.мин}} = 2^I \cdot 20\%. \quad (22)$$

Тогда окончательный нормировочный коэффициент будет выбран как:

$$K_{\text{норм}} = \max\left(K_{\text{норм.мин}}, K_{\text{норм.выч}}\right). \quad (23)$$

После происходит кадрирование сигнала. При кадрировании сегмент речевого сигнала разбивается на кадры фиксированной длины – от 10 до 100 мс. На этом участке спектр сигнала считается неизменным. Кадрирование сигнала может производиться как с перекрытием, так и без него. Перекрытие помогает уменьшить

влияние от искажений на краях кадров, но увеличивает время на его обработку [72].

В дальнейшем производится оконное преобразование. К каждому выделенному кадру применяется *весовая функция*, также называемая *оконной функцией* – например, оконная функция Хэмминга.

Оконная функция используется для уменьшения влияния конечности выборки на последующее дискретное преобразование Фурье [47, 49].

$$x_h[n] = x[n] \cdot h[n], \tag{24}$$

где h[n] = 0,54 − 0,45 cos ($\frac{2\pi n}{N-1}$).

В различных задачах используются различные функции. Так, наиболее распространёнными являются окна Хэмминга (Hamming) [73]. Часто используется окно Ханна также известное как окно Хэннинга (Hanning – аллюзия на Hamming).

Часто используется окно Гаусса, в частности. Окно Гаусса даёт результат лучше, чем окна Хэмминга и Ханна, в том числе и потому, что лучше сглаживает боковые лепестки на спектрограммах [73].

Применение к окну дискретного преобразования Фурье:

$$x_k[k] = \sum_{n=0}^{N-1} x_h[n] \cdot e^{-\frac{2i\pi nk}{N}}, \tag{25}$$

где k=0,…,N−1.

Также требуется фильтрация шумов и посторонних звуков. Низкое соотношение сигнал/шум (ниже 12 дБ) вносит существенные ухудшения в процесс извлечения информации, в частности, о формантных траекториях сигнала [73,74].

Для уменьшения шума можно также использовать кепстральный метод [75].

Спектральный центр сигнала может быть использован в сегментации сигнала для шумоподавления и нахождение участков пауз [76].

2.6 Использование формант в речевом образце

Форманты – это резонансные частоты вокального тракта, которые появляются как чёткие пики в речевом спектре [77]. Такие пики являются признаком большой концентрации энергии внутри озвученной фонемы. Формантная структура, как правило, состоит из шести формантных частот.

Высшие формантные частоты содержат более высокую энергию, но менее разборчивы в высоких частотах.

Форманты могут быть расценены как характеристики частоты, а также как просодические характеристики, поскольку они моделируют эволюцию речевого сигнала во временной области.

Первая форманта F1 несёт наибольшую энергию и отражает стиль речи и структуру языка, что является важным для распознавания диктора [65]. Форманты F2, F3 и выше более важны для разборчивости речи.

Акцент затрагивает как свойства самого диктора, так и свойства его речи [65]. F2 и F3 очень чувствительны к изменениям обоим параметрам, которые отражают движения языка, в то время как F1 изменяет своё положение только в том случае, когда изменяется общая форма вокального тракта.

Форманты могут быть вычислены из спектральной плотности мощности (power spectral density). Тем не менее, основным метром считается метод разделения источника-фильтра (source-filter separation method), из которого выделяются коэффициенты LPC. Однако, вычисление формант из свойств спектральной формы вокального тракта нетривиальны, поскольку не все пики спектра являются результатом резонанса в вокальном тракте [65].

2.7 Спектральные характеристики речевого сигнала

Появление спектрографа пришлось на начало сороковых годов двадцатого века и предвосхитило дальнейшее бурное развитие фонетики. Звуковые спектрограммы предоставляют детальную информацию об акустических особенностях звуков в целом и о речевых звуках в частности. Они широко используются для

изучения акустических особенностей гласных и согласных, произносимых различными людьми. Например, спектрограммы показывают различия между качеством гласных более чётко, чем графики речевого сигнала. В то время как спектрограммы показывают, как акустические показатели меняются с течением времени, спектр предоставляет краткую характеристику этого явления, взятого в конкретный момент времени и необходимого для получения более точных измерений. С помощью спектрограмм обычно изучаются следующие основные характеристики речевых звуков:

1) полный звуковой пробел (overall gap);

2) постоянная энергия (continuous energy);

3) периодичность (periodicity);

4) гармоническая структура (harmonic structure);

5) форманты (formants);

6) антиформанты (anti-formants).

2.8 Анализ речевого сигнала с использованием корреляционной функции

Корреляционный анализ – процесс поиска связей между статичными величинами при выполнении работы с речевыми сигналами. Измерительная единица, которая характерна для математической области в сфере анализа корреляции, коэффициент корреляции. Он используется при работе с двумя величинами, взаимодействующими между собой в решении определенных задач. Актуальность используемой в данном контексте методики обусловлена тем, что использование в расчётах и связанных с ними процессов коэффициентов корреляции значительно упрощает задачу. Основными понятиями корреляционного анализа применительно к обработке речи выступают две функции - взаимнокорреляционная и автокорреляционная [78].

Автокорреляционная функция определяет статистическую взаимосвязь между величинами из одного речевого сигнала, разложенного в ряд, но взятых со сдвигом:

$$A = \int_{-\infty}^{+\infty} s(t)s(t-\tau)dt, \quad (26)$$

где $s(t)$ – речевой сигнал; τ – сдвиг во времени.

Взаимнокорреляционная функция определяет степень корреляции двух последовательностей значений речевых сигналов, разложенных в ряды, также взятых со сдвигом:

$$B = \int_{-\infty}^{+\infty} s_1(t)s_2(t-\tau)dt\,, \quad (27)$$

где $s_1(t)$, $s_2(t)$ – речевые сигналы.

2.9 Кепстральный анализ речевого сигнала

Кепстральный анализ популярен среди прочих методов обработки речевых сигналов, что объяснимо таким преимуществом, как компактное сжатие данных в процессе преобразования сигнала из временной в частотную область обработки. Как известно, в результате данного перехода улучшаются такие характеристики информации, как наглядность, компактность и подробность. Идея кепстрального анализа (замена осью времени оси частоты в спектре, т.е. представление спектра в качестве сигнала) проистекает из преимуществ спектрального метода представления данных. Исходя из этого, возможно еще более компактное представление спектральной информации, когда всякий отдельный гармонический ряд первоначального спектра представляется в кепстре только одной составляющей. Под кепстром подразумевается спектр логарифма спектра первоначального сигнала, а исходный спектр представляется в логарифмическом масштабе [79]:

$$C_s(q) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \ln\big(S(\omega)\big)^2 e^{j\omega}d\omega\,, \quad (28)$$

где $S(\omega)$ – амплитудный спектр сигнала $s(t)$.

Из формулы (25) видно, что $S^2(\omega)$ имеет смысл спектральной плотности энергии сигнала $s(t)$. Тогда $G_s(q)$ интерпретируется как энергетический спектр

функции $ln(S(\omega^2)$. Кепстральный анализ в задачах обработки речевых сигналов основан на выделении кепстральных коэффициентов на мел-шкале, называемых мел-частотно кепстральными коэффициентами. Метод получения МЧКК основан на модели функционирования органов слуха человека и использует частотную шкалу в мелах, которая моделирует частотную чувствительность человеческого уха [54].

Анализ с использованием линейного предсказания является одним из самых используемых методов в задачах обработки речевых сигналов? Структура основывается на предположении, что любой отсчет речевого сигнала $s(n)$ можно приближенно оценить линейной комбинацией некоторого числа p предшествующих ему отсчетов, что приводит к следующему соотношению:

$$s(n) = \sum_{i=1}^{p} a_i s(n-1) + Gu(n)\,, \tag{29}$$

где a_1, a_2, ..., a_p – коэффициенты предсказания; $u\ (n)$ – нормализованная последовательность возбуждения (ошибка предсказания); G – коэффициент усиления [80].

Ошибка предсказания u(n) определяется как разность между исходными и приближенно вычисленными (предсказанными) отсчетами:

$$u(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k). \tag{30}$$

Выполняется алгоритм линейного предсказания для решения ряда задач. Среди них можно обозначить основную – идентификация и установление набора коэффициентов предсказания, снижающих u(n).

На практике используется две главные методики поиска данных линейного предсказания. Названия способов – автокорреляционный и ковариационный. В рамках выполнения таковых необходимо применять представление сигнала в об-

ласти времени. В конкретном случае точно данные не устанавливаются в точке расположения, при этом результат будет зависеть от стационарности сигнала.

Коэффициенты предсказания позволяют установить частотную характеристику фильтра. Последний необходим для отражения состояния голосового тракта на заданный момент времени, определения усредненной оценки исследуемого участка сигнала.

С помощью MFCC можно редуцировать размерность параметров кадра. Получаемый набор признаков имеет меньшую размерность, чем частотный «портрет», и может применяться для выделения особенностей произношения не только отдельных звуков, но также слогов, слов, предложений [62].

Вместо шкалы герц используется шкала мел, рассчитываемая следующим образом:

$$m=1127.01048\ln(1+\frac{f}{700}), \quad (31)$$

где m – частота в шкале мел, f – частота в шкале герц.

Размерность вектора-портрета звука редуцируется до набора K коэффициентов. Для этого каждый кадр в частотной области разбивается на диапазоны с помощью гребёнки треугольных фильтров, границы которых рассчитывают в шкале мел.

В некоторых случаях необходимо найти точные данные о частотах индивидуальных обертонов в звуке в определенный момент времени, при этом изучение только лишь узкополосной спектрограммы является неинформативным. В подобных случаях нужно вычленить и подвергнуть анализу спектральный срез (также называемый FFT или спектр), который представляет собой результат быстрого преобразования Фурье, выполненного на небольшом участке звука. Спектральный срез позволяет судить о присутствующих в звуке частотах и об их относительных амплитудах, что важно при изучении колебаний частоты основного тона, назализации, спектрального наклона и других параметров.

Для K коэффициентов разлиновывается гребёнка фильтров. Вычисляются весовые коэффициенты. Пусть k – позиция во фрейме спектра, а m – порядковый номер коэффициента, тогда:

$$H_m[k]\begin{cases} 0, & k < f[m-1] \\ \dfrac{k - f[m-1]}{f[m] - f[m-1]} & f[m-1] \le k \le f[m] \\ \dfrac{f[m+1] - k}{f[m+1] - f[m]} & f[m] \le k \le f[m+1] \\ 0, & f[m+1] < k \end{cases} \quad (32)$$

где n – порядковый номер отсчета спектра сигнала, *f(m)*, *m*=1,2,...,M – частота в Гц m-го фильтра

Для каждого окна вычисляется энергия, после чего применяется дискретное косинусное преобразование:

$$S[m] = \ln\left(\sum_{k=0}^{N} |S[k\}|^2 \cdot H_m[k]\right); \quad (33)$$

$$c[n] = \sum_{m=0}^{K-1} S[m] \cdot \cos\left((m + 0{,}5)\frac{i\pi}{K}\right), \quad (34)$$

где $i = 1,...,$ *K*.

Поскольку нулевой коэффициент c_0 представляет собой энергию сигнала и не несёт информации о говорящем, он отбрасывается [81]. Оставшиеся *K*-1 коэффициентов формируют «портрет» кадра.

2.10 Анализ с использованием нейронных сетей

С целью распознавания сигналов речи будет рационально применять метод, задействующий нейронную сеть, чья структура формируется нейронами и установленными между ними связями. Каждый нейрон в рамках нейронной сети следует рассматривать как отдельную ячейку. Состояния нейрона меняются так же, как и у клеток головного мозга: он может быть либо заторможен, либо возбужден. Связи, которые формируются между нейронами, делятся на два типа. Входные

связи, направленные только в одну сторону, называют синапсами, а выходные связи — аксонами. Именно от аксонов последующие нейроны получают на свои синапсы сигналы торможения либо возбуждения. То, как выглядит нейрон, представлено на рисунке 2.4.

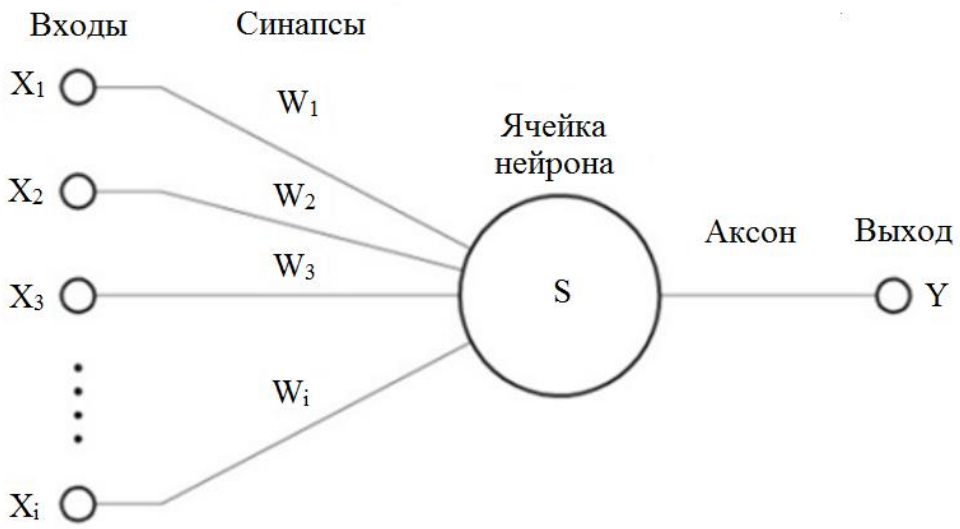


Рисунок 2.4 – Общий вид нейрона

Для любой однонаправленной связи можно вычислить вес w_i (так обозначают величину синаптической связи). Эквивалентом этого веса в физическом смысле является электрическая проводимость. Если синапс заторможен, w_i примет отрицательное значение, если синапс возбужден – положительное [82]:

$$s = \sum_{i=1}^{n} x_i w_i. \tag{35}$$

Выход нейрона есть функция его состояния:

$$y = f(\mathrm{s}). \tag{36}$$

Чтобы задействовать нейронную сеть для распознавания речевых сигналов, следует создать сеть, чьи параметры удовлетворяли бы решению конкретной поставленной задачи. Затем нейронную сеть необходимо научить распознавать множественные речевые сигналы. Чтобы свести вероятности ошибки до минимума, нейронная сеть должна научиться подбору весовых синаптических коэффициентов.

2.11 Анализ с использованием скрытых марковских моделей

Метод, задействующий СММ, доказал свою результативность при распознавании и обработке сигналов речи. СММ представляет собой статистическую модель, которая имитирует принцип работы процесса, аналогичного марковскому процессу с неизвестными параметрами. Первостепенная цель применения СММ – определить неизвестные параметры, исходя из тех величин, что поддаются наблюдению. Полученными параметрами допустимо пользоваться для дальнейшего анализа.

Для распознавания сигналов речи СММ задействуют, отталкиваясь от следующих допущений [83]:

- сигнал речи можно фрагментировать (разделить на отдельные состояния). В рамках одного состояния сигнал допустимо считать стационарным. Чтобы перейти от одного фрагмента к другому, сигналу требуются доли секунды;
- то, с какой вероятностью появятся порождаемые моделью символы, определяется исключительно текущим состоянием модели. Никакие порожденные ранее символы на этой вероятности не сказываются.

СММ представлены в нескольких разновидностях, каждой из которых присуща уникальная топология. Детальный разбор топологий СММ дан в [84].

На рисунке 2.5 в качестве наглядной иллюстрации можно увидеть топологию СММ, предусматривающей три состояния. СММ является конечным автоматом. Состояние данного автомата меняется в каждый n (буквенное обозначение для дискретного момента времени). Величина a_{ij} отражает вероятность перехода автомата от состояния Si к состоянию S_j. Величина $b_j(O_n)$ отражает вероятность порождения моделью вектора наблюдений O_n для каждого дискретного момента времени.

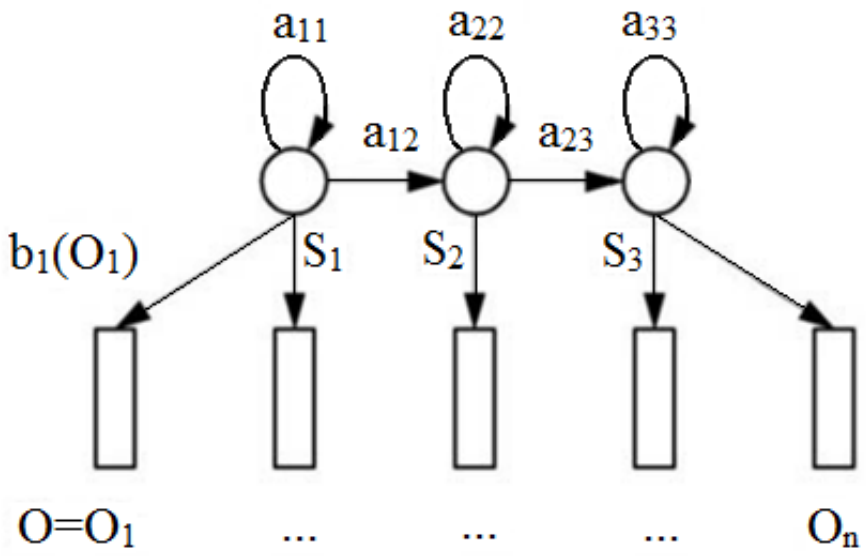


Рисунок 2.5 – Топология СММ с тремя состояниями

2.12 Анализ с использованием динамического трансформирования времени

Характеристики сигнала речи способны меняться с высокой скоростью. Сигналы речи допустимо представить как векторы. С целью оценки расхождения двух векторов следует задействовать динамическое трансформирование времени (ДТВ), которое позволит осуществить выравнивание по времени [85].

ДТВ ориентировано на работу с фрагментами. Чтобы анализировать признаки, необходимо через регулярные интервалы обрабатывать вектор этих признаков. В вектор может входить много фрагментов – поэтому, чтобы определить локальную оценку расстояния, задействуют специальные средства расчета.

Оценка расстояния между двумя векторами признаков рассчитывается с помощью Евклидового расстояния:

$$d(x,y) = \sqrt{\sum_i (x_i - y_j)^2}, \quad (37)$$

где x_i, y_i – сравниваемые фрагменты; i – номер фрагмента.

Один из возможных вариантов – вычислить Евклидово расстояние. По отношению к другим возможным операциям этот вариант может быть не самым выгодным, зато он отличается высокой результативностью.

В шаблоне хранятся варианты, с которыми необходимо сопоставить входящий сигнал. В итоге должен получиться шаблон, в котором характеристики одного из уже имевшихся вариантов как можно более идентичны характеристикам входящего сигнала. Глобальную оценку расхождений в рамках данного маршрута следует оценивать как результат сложения всех локальных расстояний между вариантами из шаблона и элементами фрагментированного сигнала. Все способы обработки сигналов речи поддаются классификации в зависимости от того, какой тип анализа в них задействован.

Задачи обработки:

- фильтрация и подавление шума;
- сегментация на информативные участки;
- определение информативных параметров;
- распознавание.

Область обработки:

- частотная область обработки;
- частотно-временная область обработки;
- временная область обработки.

Методы обработки:

- анализ с использование преобразования Фурье;
- анализ с использованием вейвлет-преобразования;
- анализ с использованием линейного предсказания;
- анализ с использованием нейронных сетей;
- анализ с использованием скрытых марковских моделей;
- анализ с использованием кепстра;
- анализ с использованием преобразования Гилберта-Хуанга;
- анализ с использованием корреляционной функции;
- анализ с использованием динамического трансформирования времени.

Все перечисленные способы активно применяются в системах распознавания речевых сигналов. Эта классификация не является исчерпывающей, однако она помогает составить представление о том, насколько разнообразны аналитические методики. С ее помощью можно оценить сильные и слабые стороны различных методик и понять, насколько они пригодны для решения конкретной задачи.

2.13 Многокомпонентная информация, заложенная в речевом образце

Сигнал речи может наделяться специфическими характеристиками за счет особенностей центральной и вегетативной нервной системы говорящего. Эта система контролирует речевую реализацию индивидуальных аспектов голосообразования (фонации) [86] в психическом и нейрофизиологическом плане.

Выделяют два типа вариативности речевых сигналов:

а) интраиндивидульную. Она формируется под влиянием комплекса факторов, не имеющих отношения к семантике высказываний. На выражение такой вариативности влияет множество не поддающихся контролю факторов работы компонентов речевого аппарата. В силу этого речь и голос могут самопроизвольно варьироваться даже в том случае, если речевой отрезок сам по себе неизменен;

б) интериндивидуальную. Этот тип вариативности выступает фундаментом для информационного поля признаков, обладающих индивидуальной значимостью. Он зависит от социальных, психологических и анатомо-физиологических особенностей индивида.

Важнейшей стороной человеческой речи является операционная, также известная как исполнительная. В качестве ее первого звена следует выделить звуковую (физическую) материю. Анализируя эту материю, можно на примере отдельного языка выявить, как реализация речи индивида соотносится с инвариантом и вариантами модели интонации и звука.

Второе звено операционной стороны – это то, как речевой материал организован в лексико-семантическом аспекте. Существующие в языке понятия и образы получают словесное воплощение, за счет чего в речи реализуется лексико-

морфологический языковой код. Из этого можно вывести концептуальную основу для эффективной идентификации говорящего в соответствии с поставленными целями.

Речь индивида строится на основе артикуляторных и фонационных жестов. Эти жесты напрямую связаны с социально отработанным фонологическим представлением высказывания и спецификой его лексико-семантических аспектов. При идентификационном анализе говорящего рекомендуется принимать во внимание следующие факторы построения акустико-лингвистического алгоритма:

- как именно речевой сигнал декодируется по уровням в глобальном лингвистическом плане;
- как сигнал декодируется в интеллектуально-содержательном смысле;
- каковы особенности декодирования сигнала с учетом его социо-психологической ориентации;
- каковы особенности декодирования сигнала с учетом его анатомо-физиологической ориентации;
- как сигнал обрабатывается с аппаратно-программной (акустической) точки зрения.

Весь спектр задач можно объединить общей целью: сформировать персонализированный «портрет» говорящего с учетом следующих коррелятов его речи: супрасегментных (просодических), сегментных (артикуляторных), голосовых (фонационных). Базой для анализа акустико-лингвистических аспектов выступает итеративная обработка речевой волны. Акустико-лингвистические признаки сигнала принято разделять на вторичные и первичные.

Вторичные признаки принято также называть просодическими. Они реализуются на базе первичных и потому условно носят характер надстройки. За счет супрасегментной (надсегментной) реализации вторичных признаков в речи формируются структурно-организованные фигуры и их конкатенации, характер которых сугубо индивидуален.

Первичные включают в себя следующие признаки:

- артикуляторные. Во внимание принимается тип артикуляторного порождения сигнала (в частности, напряжена ли артикуляция или нет) и соотношение данного сигнала к одному из видов артикуляции;
- фонационные. Учитывается тип голосовой мимики (в частности, щадящее ли голосообразование или форсированное) и соотношение данного сигнала к одному из типов фонации.

Эти признаки во многом определяются анатомическими и физиологическими особенностями индивида.

В речи и синкразии (индивидуальном складе) человека содержится индивидуальная и коммуникативная информация.

Языковым сигналом в пределах речи является коммуникативное содержание сообщения, а внеязыковым — сведения социально-демографического характера об индивиде.

Информацию в речевом сигнале можно классифицировать следующим образом:

А.

- интеллектуальная;
- социальная (статус, происхождение);
- психологическая (личностная характеристика);
- биологическая (состояние здоровья, возраст, пол, габариты тела);

Б.

- аффективная (как меняется состояние индивида);
- идиосинкразическая (личностные нюансы);
- групповая информация (в каком регионе проживает говорящий, какова его профессия).

Создавая речевой портрет индивида, выделяют три разновидности нормы: синкразическую [87], групповую, универсальную. Голосовая информация в этом процессе имеет ключевое значение.

2.14 Алгоритмы пофонемного распознавания речи в амплитудно-временной области

Речь важно распознавать для того, чтобы как минимум в рамках небольшого спектра команд можно было достичь точности 100%. Вторая цель – распознавать поток непрерывной произвольной речи с приемлемой точностью независимо от того, кто именно говорит. Специалисты пытаются решить эти проблемы уже свыше пятидесяти лет, но безрезультатно.

Сигналы речи меняются слишком часто и по слишком многим параметрам. Они могут быть разными по продолжительности, скорости, тембру голоса, помехам, колебаниям эмоционального состояния говорящего, значительной разницы в голосах двух дикторов. Один и тот же фрагмент речи в исполнении одного и того же диктора может звучать совершенно по-иному, если записать его в разное время. Следует выявлять такие характеристики сигнала речи, которые давали бы его исчерпывающее описание (на основе которого можно было различать звуки речи между собой) и в то же время выявляли общие параметры между различными речевыми вариациями. Потом эти характеристики надо сопоставлять с образцами. Целью сопоставления будет выявить не любое совпадение, а максимальное соответствие. Из-за этого желаемую форму расстояния приходится искать в ограниченном параметрическом пространстве.

Чтобы распознать речевой сигнал, следует отталкиваться от релевантной системы характеристик и задействовать разумные алгоритмы. Ниже изложен подход пофонемного распознавания речи, при котором речевой сигнал рассматривается преимущественно не в частотном, а во временном представлении.

Если сигнал распознается по фонемам, это значит, что конечным результатом распознавания станут не морфологические и не лексические единицы, а именно фонемы. Этим термином обозначают элементы фонетического строя языка, то есть звуки речи.

2.14.1 Представление речевого сигнала во временной области

После того, как распознаваемые речевые единицы определены, следует обратиться к системе тех характеристик, что достаточно быстро и точно отличают между собой классы фонем.

Речевые и, в целом, звуковые сигналы после оцифровки на звукозаписывающем устройстве превращаются в массив семплов (отсчетов). Фрагмент речи допустимо расценивать как дискретную функцию амплитуды сигнала от временного значения. При этом следует не принимать во внимание параметры звуковой карты и микрофона, которые неизбежно влияют на цифровой сигнал, и не учитывать погрешность квантования.

Ряд суждений касательно произносимых звуков можно вынести исходя из того, как выглядит график данной функции. Рисунок 2.6 демонстрирует: на тех участках, где присутствует голос (при произнесении звонких согласных либо гласных), кривая выглядит не так, как при произнесении глухих согласных без участия голоса.

Рисунок 2.6 – Визуализация слова «каша» [88]

Поведение функции отличается на участках с разными фонемами. Следовательно, можно выявить определенные изменяемые параметры ее изменения. С помощью этих параметров классы фонем можно отличать между собой, сопоставляя их с пороговыми значениями.

Временное представление сигнала речи вариативно и нестабильно. Тем не менее, если распознавать сигналы по фонемам, можно разрабатывать эффективные алгоритмы распознавания (например, для сегментации фонем).

Поведение функции отличается на участках с разными фонемами. Следовательно, можно выявить определенные изменяемые параметры ее изменения.

С помощью этих параметров классы фонем можно отличать между собой, сопоставляя их с пороговыми значениями.

Величина $V = \sum_{i=1}^{n} |x_{i+1} - x_i|$ – является прямым случаем данной характеристики, которая принимает значение численного аналог целой вариации функции для дискретного случая. Здесь n – число отсчетов на участке сигнала, x_i –значение i-го отсчета.

Временное представление сигнала речи вариативно и нестабильно. Тем не менее, если распознавать сигналы по фонемам, можно разрабатывать эффективные алгоритмы распознавания (например, для сегментации фонем).

2.14.2 Классификация шипящих и пауз

На выделенном участке с глухими согласными, для отделения шипящих звуков от паузообразных определяется функция V – вариация с переменным верхним пределом: $V(1)=0, V(n) = \sum_{i=1}^{n} |x_{i+1} - x_i|$. Чтобы отсортировать шипящие звуки от паузообразных, на выделенном сегменте с глухими согласными задействуют функцию V – вариацию, чей верхний предел носит переменное значение. Для паузообразных звуков темп возрастания данной функции ниже, чем для шипящих. Чтобы пользоваться этим фактом для классификации звуков, задействуют также W – вспомогательную функцию, которая возрастает параллельно с V, но обнуляется при достижении заданной фиксированной величины A.

Положим $W(n) = V(n)$ при $1 \le n \le N_1$,

где N_1 – максимальное число такое, что $W(N_1) \le A$.

Положим $W(N_1+1) = 0$,

$$W(n) = \sum_{i=N_1+1}^{n} |x_{i+1} - x_i| \text{ при } N_1 + 1 \le n \le N_2.$$

Продолжим ряд чисел N_1, N_2, …, пока не закончится выделенный участок с глухими согласными. При этом график функции W примет характерный вид (рисунок 2.7).

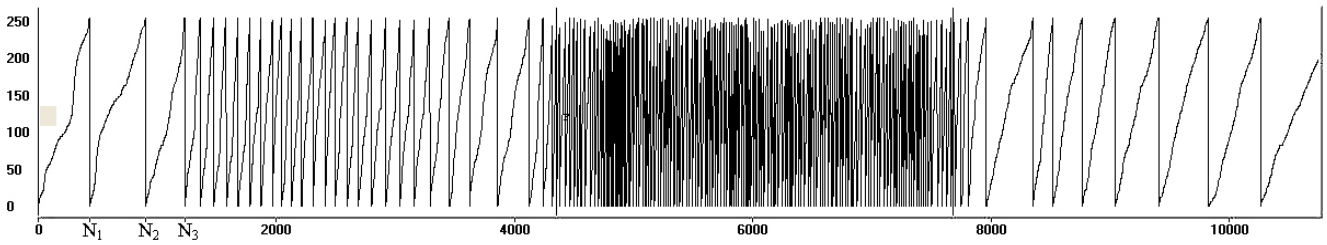


Рисунок 2.7 – График функции *W* для слова «каша» [88]

Паузы и шипящие можно разделять на основе оценки расстояния между точками Nk. Это расстояние будет длинным для паузообразных звуков и коротким для шипящих. Элементы массива расстояний N1, N2 − N1, N3 − N2, … иногда будут превышать определенный порог на участке с глухими согласными. В случае превышения звук следует классифицировать как паузообразный, при отсутствии превышения — как шипящий (рисунок 2.8) [89].

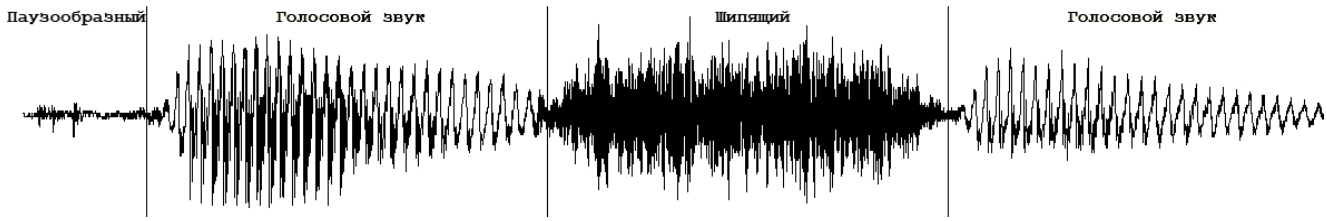


Рисунок 2.8 – Классификация шипящих и пауз в слове «каша» [88]

Для вышеприведенных алгоритмов характерна высокая точность и стабильность при сегментировании фонем и выявлении пороговых значений. Если исследовать с помощью такого подхода временные представления речевых сигналов, это откроет возможности для разработки и оптимизации прочих алгоритмов. Тем самым удастся решить многие другие задачи, связанные с распознаванием речевых сигналов.

2.15 Выбор языка и среды программирования

Для разработки локального рабочего места в качестве языка программирования был выбран C#, среда программирования Visual Studio 2019.

2.16 Мел-частотные кепстральные коэффициенты

Для полученного набора фреймов, которые соответствуют конкретному слову как числовой характеристике фрейма, допустимо задействовать функцию Root

Mean Square (то есть средний квадрат всей совокупности его значений). Доля сведений, пригодных для дальнейшего анализа, в данной метрике крайне мала.

В связи с этим удобнее пользоваться мел-частотными кепстральными коэффициентами (Mel-frequency cepstral coefficients). Аббревиатурой MFCC обозначают представление энергии спектра сигнала. Преимущества его применения таковы:

- количество коэффициентов, которые предстоит вычислить, можно ограничить любым значением. Фрейм сожмется, а вместе с ним и количество информации для обработки;
- проекция спектра на mel-шкале позволит выявить те частоты, что имеют наибольшую значимость для человеческого восприятия;
- с помощью спектра сигнала (разложения по базису ортогональных (ко)синусоидальных функций) можно принимать во внимание волновую природу сигналов.

В рамках данного этапа каждому фрейму соответствует набор из M мел-частотных кепстральных-коэффициентов. Их можно задействовать для дальнейшего анализа.

Вывод по разделу 2. На основе аналитического обзора была проведена классификация существующих методов обработки речевых сигналов, применяемых в системах распознавания речи. Представленные классификации систем, характеристик и методов обработки позволяют на их основании дать оценку возможности применения новых математических аппаратов в задачах обработки речевых сигналов в системах распознавания речи. Следующим этапом следует разработка программного комплекса анализа речи.

3 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Во время изучения иностранного языка человек ориентируется на систему фонетики родного языка, которая может разниться с особенностями изучаемого, например, отсутствие ряда целевых звуков, различие в произношении и смысловой нагрузке интонаций.

Требуется на основе изучения уже реализованных принципов работы и исследований существующих методов обработки речевых сигналов, применяемых в системах распознавания речи, составить алгоритм обработки и сопоставления речевых образцов с последующей разработкой программного обеспечения. На его основе произвести анализ интонации входного образца посредством сравнения с загружаемыми эталонами из базы образцов произношения, сформированной на основе определенных требований к образцам-эталонам носителей целевого языка.

Обобщенная схема системы распознавания включает в себя этапы, схематично представленные на рисунке 3.1.

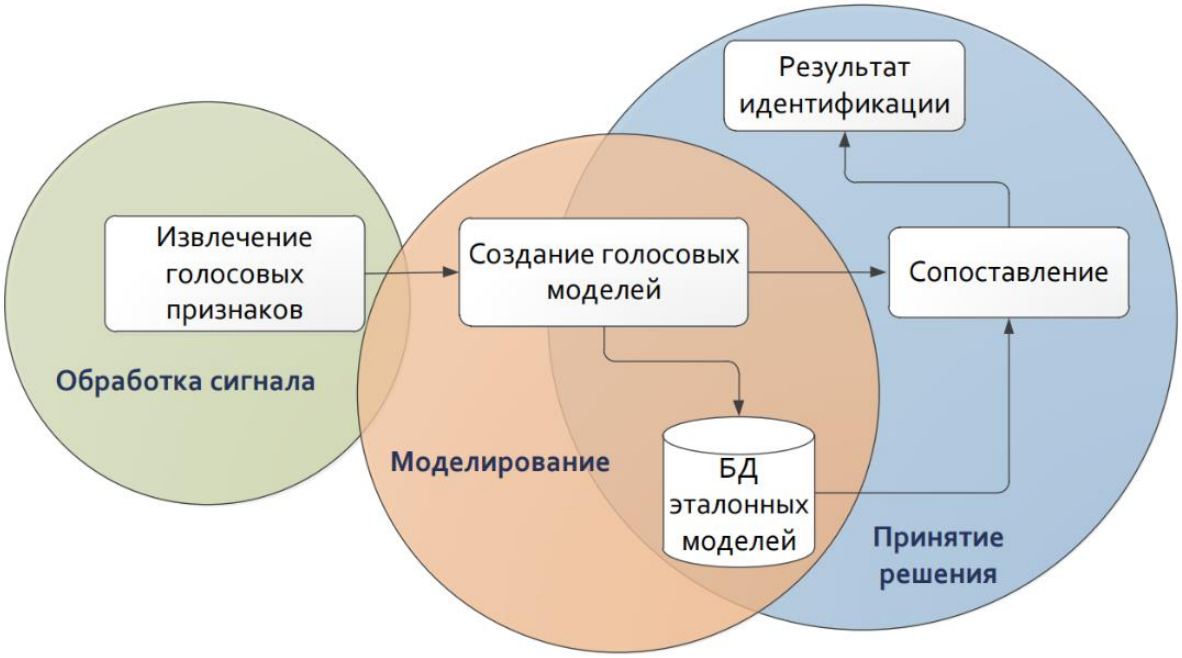


Рисунок 3.1 – Обобщенная схема системы распознавания диктора

Входными данными для построения модели голоса выступает набор акустических признаков речевого сигнала, извлеченный из него на этапе его обработки. Так как речевой сигнал анализируется в покадровом режиме, акустические при-

знаки, как правило, представляют собой совокупность векторов признаков, по одному вектору на кадр. Суть классификации голосовых признаков состоит в математической модели голоса в целях обеспечения возможности вычисления меры подобия голосов различных дикторов.

Разработанный алгоритм по установлению интонационных конструкций представлен ниже на рисунке 3.2:

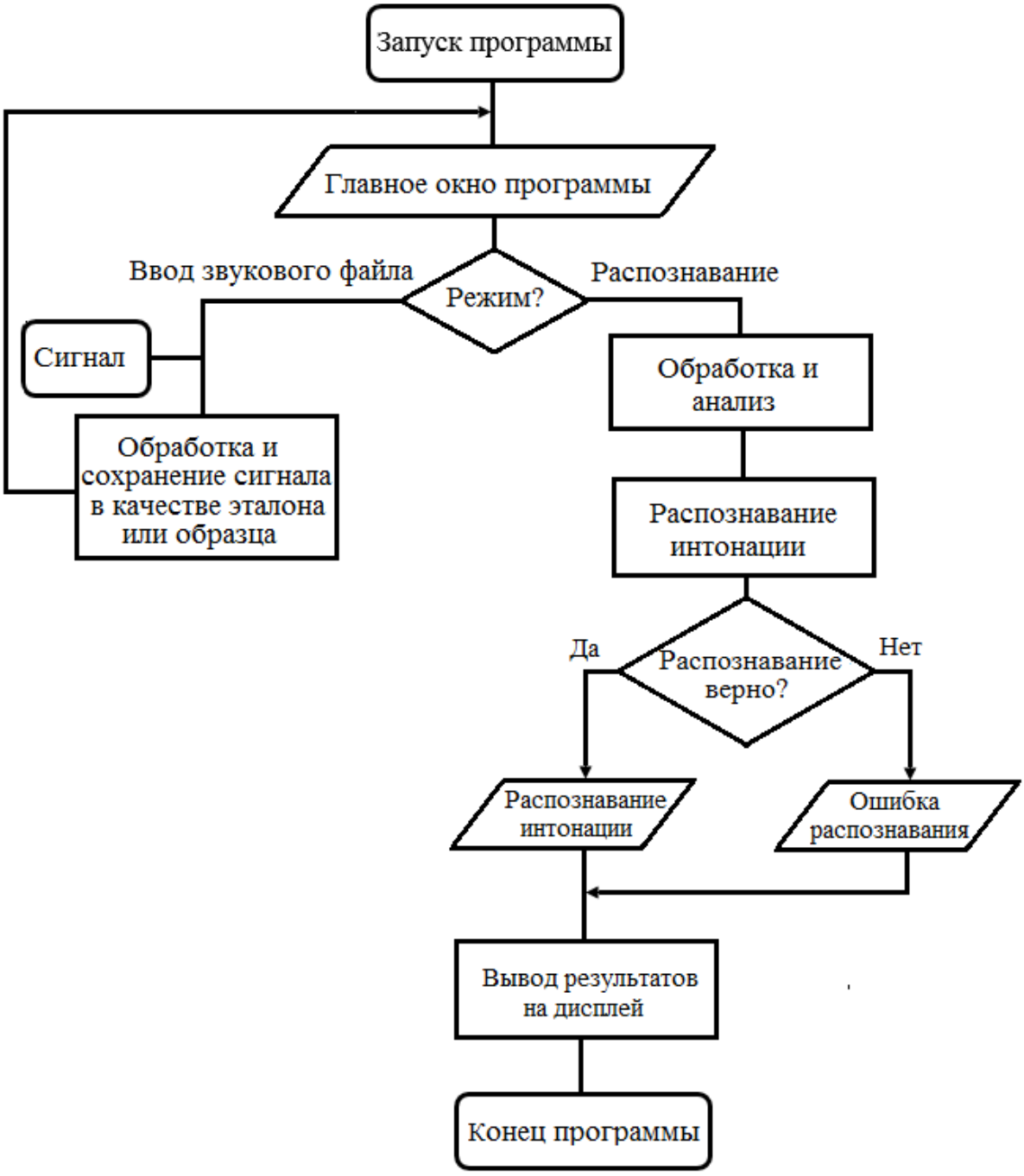


Рисунок 3.2 – Алгоритм работы программы

3.1 Программная реализация алгоритмов

На основе изучения уже имеющегося программного кода [90-99] от локального рабочего места исследователя был составлен алгоритм для разрабатываемого модуля подбора параметров. Реализация алгоритма идентификации была выполнена в среде разработки Visual Studio 2019, языком программирования был выбрал C#, поскольку исходный код локального рабочего места представлен на нем Инструментарий среды программирования Visual Studio позволяет анализировать большой объем данных и разрабатывать трудоемкие алгоритмы благодаря встроенным математическим функциям и возможности подключения новых. Исходный код представлен в приложении А. Для моделирования системы по распознаванию и определению интонационной составляющей была подготовлена база звуковых файлов, состоящая из 3650 записей четырех русскоязычных дикторов и восьми иностранных пользователей обоих полов. Алгоритм моделирования системы распознавания диктора включает в себя 2 этапа: обработка и сравнение. И на первом и на втором этапе выполняется выделение уникальных характеристик диктора – MFCC-коэффициентов. Данный процесс сопровождается предварительной обработкой речевого сигнала, схематично представленной на рисунке 3.3.

3.1.1 Алгоритм обнаружения речевой активности

Во время предварительной обработки сигналов будет использоваться алгоритм обнаружения речевой активности (voice activity detection, VAD), принцип работы которого визуально отображен ниже (рисунки 3.4, 3.5, 3.6). Данный инструмент представляет собой метод, используемый при обработке звукового сигнала, для обнаружения участков человеческой речи. Он облегчает обработку сигнала, а также может быть использован для отключения некоторых процессов во время аудио-трансляции избеганием ненужного кодирования/передачи пакетов тишины в приложениях, экономя на вычислениях и на полосе пропускания сети. Метод считается целесообразным для применения в реальном времени, поскольку он

проверен на записях с разным уровнем шума [100]. Эксперименты показали удовлетворительные результаты на каждом из них.

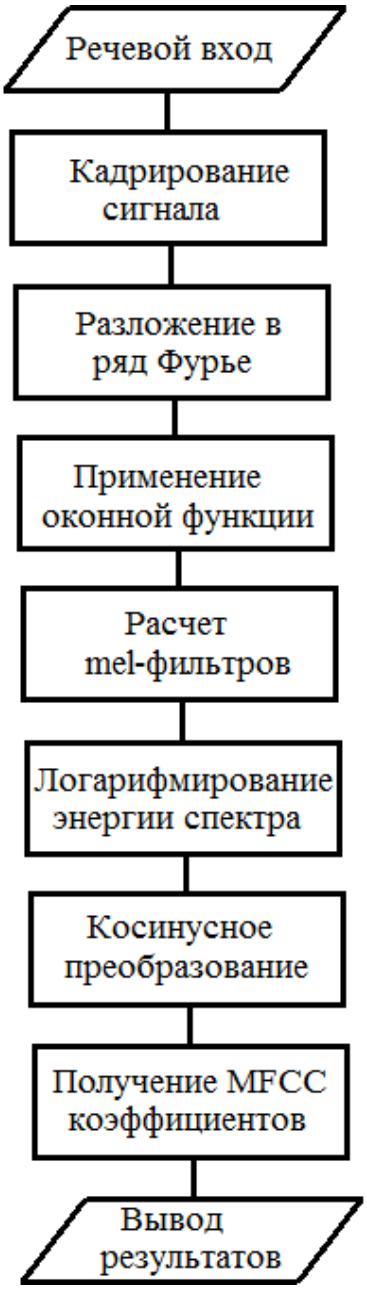


Рисунок 3.3 – Схематичное представление выявления мел-частотных кепстральных коэффициентов

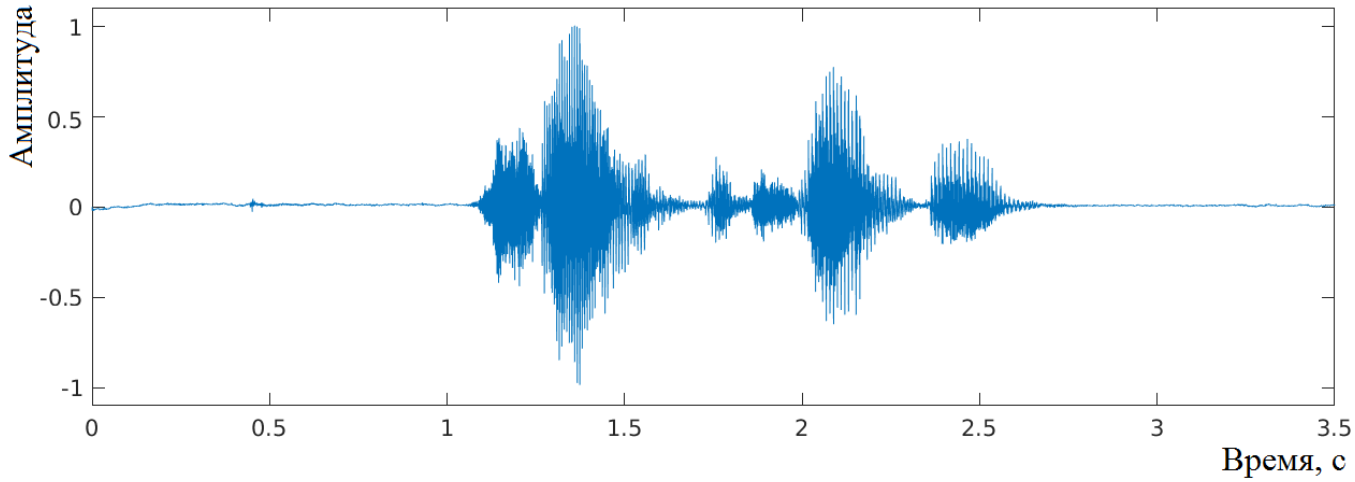


Рисунок 3.4 – Время сигнала

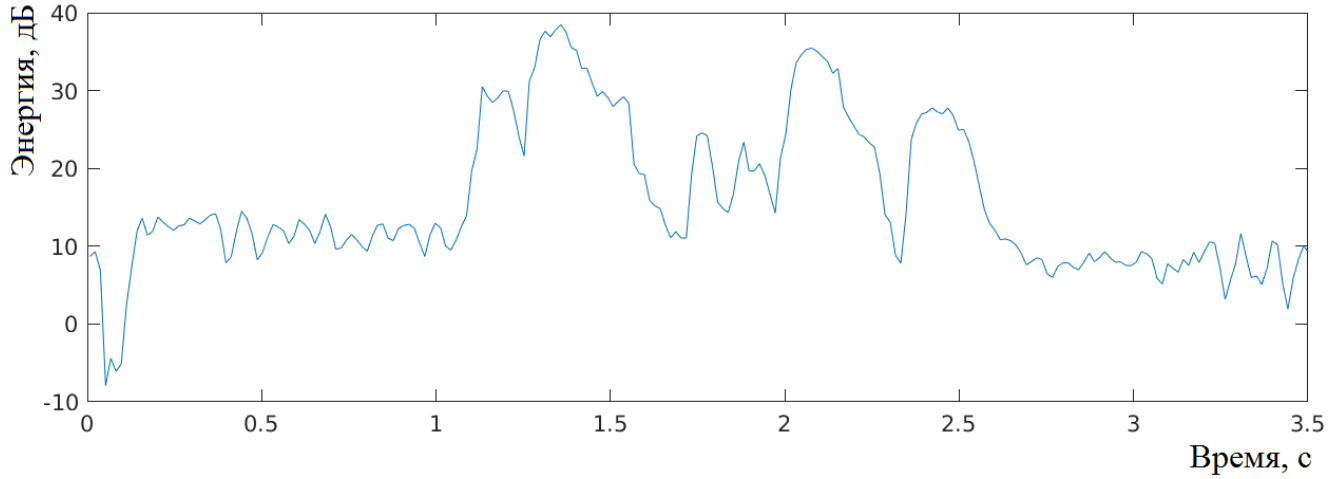


Рисунок 3.5 – Энергия сигнала

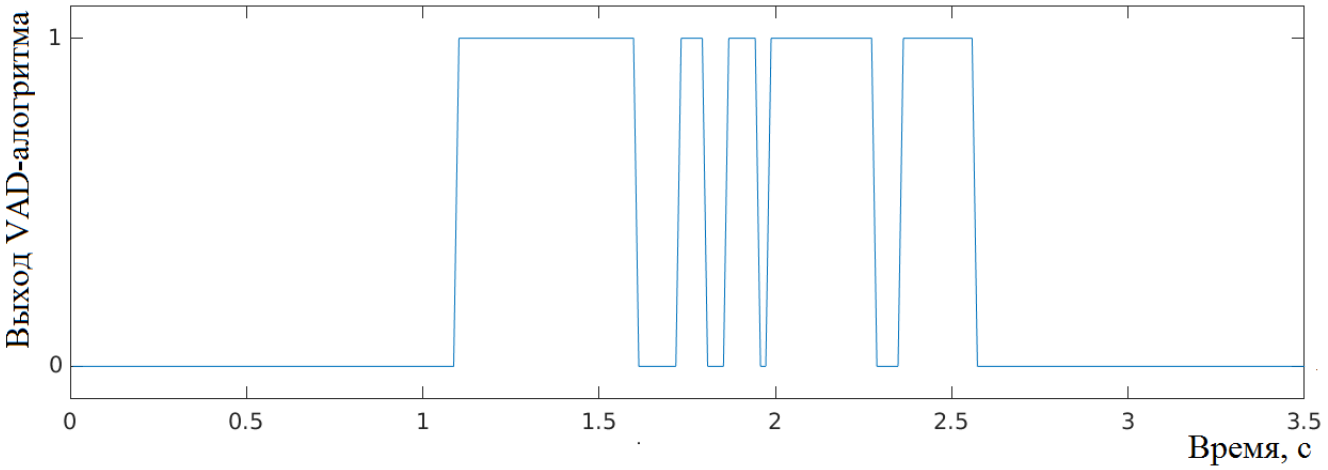


Рисунок 3.6 – Извлечение звука при помощи VAD-алгоритма,
по вертикальной оси 1 – голос, 0 – тишина.

3.1.2 Кадрирование входного сигнала во временной области

Первым делом требуется разграничить подающийся на вход звуковой сигнал на небольшие сегменты во временной области, называемыми кадрами. Однако кадры идут не вплотную один за другим, а накладываются друг на друга. Данное правило служит для снижения искажений по краям кадров. Пример такой операции визуально отражен на рисунке 3.7. На рисунке представлен входной сигнал, разбитый на четыре кадра с наложением в 50%, в результате обработки которых получается его восстановленный вид.

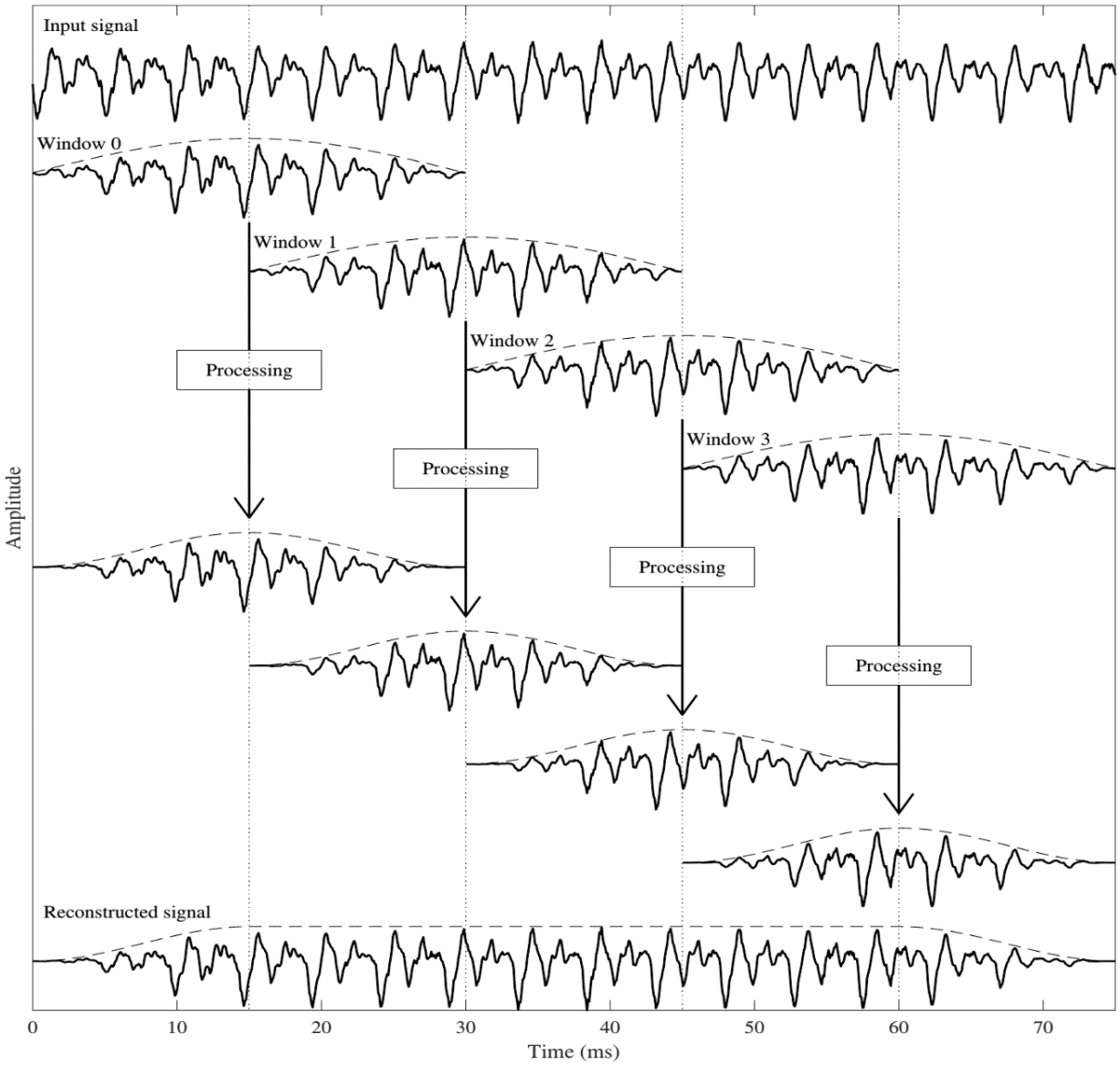


Рисунок 3.7 – Визуальное представление перекрытия кадров [101]

При кадрировании сегмент речевого сигнала разбивается на кадры фиксированной длины – от 10 до 100 мс (рисунок 3.8). На этом участке спектр сигнала считается неизменным. Кадрирование сигнала может производиться как с перекрытием, так и без него. Перекрытие помогает уменьшить влияние от искажений на краях кадров, но увеличивает время на его обработку.

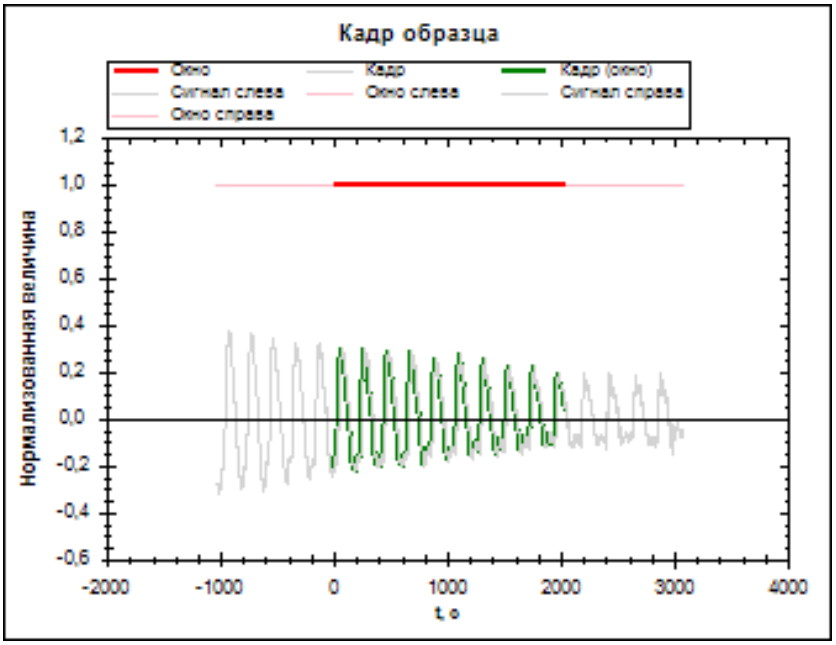


Рисунок 3.8 – Кадрирование и оконные функции преобразования кадра сигнала

Вместо конкретных значений сигнала более удобной единицей для анализа данных являются кадры, полученные сегментированием звука во временной области. Достоинство данного метода заключается в том, что анализ полученных волн на некоторых промежутках намного практичнее. Кадровое перекрытие позволяет сгладить результаты анализа.

3.1.3 Алгоритм динамической трансформации временной шкалы

Алгоритм динамической трансформации временной шкалы (dynamic time warping) – нелинейный алгоритм, являющийся чрезвычайно эффективным в качестве установления меры подобия временных рядов. Его принцип заключается в минимизации эффектов сдвига и искажений во времени, предоставляя «эластичное» преобразование временных рядов для обнаружения сходных форм с различными фазами. Другое его название – «путь наименьшей стоимости». Использует-

ся в распознавании речи для определения того, как два различных речевых сигнала представляют одну и ту же исходную произнесённую фразу [102].

Представим два записанных файла с одинаковым словом, имеющие различную продолжительность. Они нелинейно сопоставляются по кадрам, т.е. либо наименьшее растягивается соразмерно наибольшему, либо наибольшее уменьшается. На выходе для совпадения количество кадров, условно один кадр первого файла со словом может быть равным, например, трем кадрам второго файла (рисунок 3.9). Это происходит, когда за одну и ту же единицу времени было передано разное количество информации.

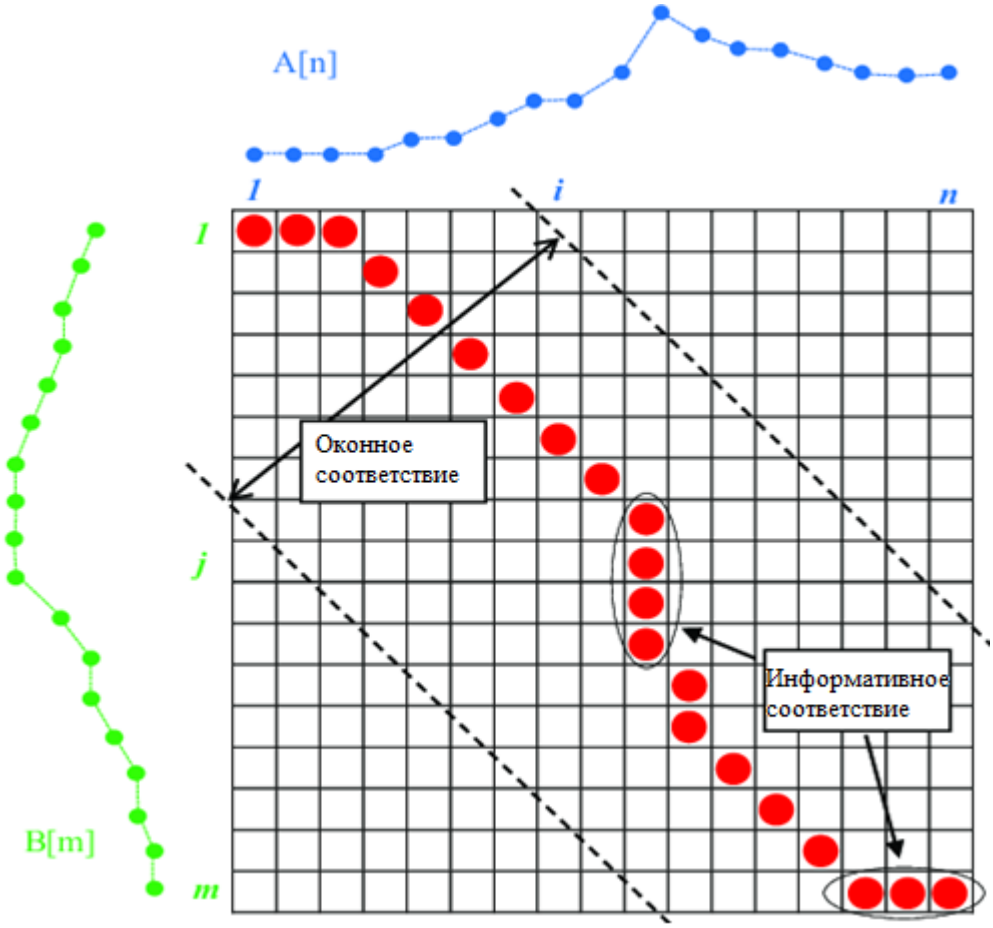


Рисунок 3.9 – Графическое изображение сопоставления двух аудиофайлов с применением DTW-алгоритма [102]

На рисунке визуально отражается принцип работы DTW-алгоритма в нелинейном сопоставлении кадров выборок из векторов звуковых сигналов A[i,n] и B[j,m].

3.1.4 Разложение в ряд Фурье

Рассчитывается спектр сигнала с помощью дискретного преобразования Фурье:

$$X_k = \sum_{n=0}^{N-1} x_n * e^{-2*\pi*i*k*\frac{n}{N}}, 0 \le k \le N, \quad (38)$$

где X_k – N комплексных амплитуд синусоидальных сигналов, представляющих исходный сигнал; N – количество значений сигнала; x_n – измеренные значения сигнала; k – индекс частоты сигнала, которая равна $\frac{k}{T}$, где T – период времени, за который принимались входные данные.

3.1.5 Применение оконной функции Хэмминга

На настоящий момент существует большое множество разнообразных оконных функций, в таблице 6 представлены самые употребляемые из них:

Таблица 6 – Окна и их дискретные выражения [103]

Наименование окна	**Выражение в дискретном виде**
Прямоугольное окно (rectangle window)	$w(k) = 1$
Синус-окно	$w(k) = \sin\left(\frac{\pi \cdot k}{N-1}\right)$
Окно Ланцоша (Lanczos window), или sinc - окно	$w(k) = \mathrm{sinc}\left(\frac{2 \cdot k}{N-1}\right), sinc(x) = \frac{\sin(\pi \cdot x)}{\pi \cdot x}$
Окно Барлетта (Bartlett window), или треугольное окно	$w(k) = 1 - \left\lvert\frac{k}{A} - 1\right\rvert, A = \frac{N-1}{2}$
Окно Ханна (Hann window)	$w(k) = 0{,}5 - 0{,}5 \cdot \cos\left(\frac{2 \cdot \pi \cdot k}{N-1}\right)$
Окно Барлетта — Ханна (Bartlett–Hann window)	$w(k) = a_0 - a_1\left\lvert\frac{k}{N-1} - 0{,}5\right\rvert - a_2 \cdot \cos\left(\frac{2 \cdot \pi \cdot k}{N-1}\right)$, $a_0 = 0.62, a_1 = 0.48, a_2 = 0.38$

Продолжение таблицы 6 [103]

Наименование окна	**Выражение в дискретном виде**
Окно Хемминга (Hamming window)	$w(k) = 0{,}54 - 0{,}46 \cdot \cos\left(\frac{2 \cdot \pi \cdot k}{N-1}\right)$
Окно Блэкмана (Blackman window)	$w(k) = a_0 - a_1\left(\frac{2 \cdot \pi \cdot k}{N-1}\right) + a_2 \cdot \cos\left(\frac{4 \cdot \pi \cdot k}{N-1}\right),$ $a_0 = 0.42, a_1 = 0.5, a_2 = 0.08$
Окно Блэкмана — Харриса (Blackman–Harris window)	$w(k) = a_0 - a_1\left(\frac{2 \cdot \pi \cdot k}{N-1}\right) + a_2 \cdot \cos\left(\frac{4 \cdot \pi \cdot k}{N-1}\right) - a_3 \cdot \cos\left(\frac{6 \cdot \pi \cdot k}{N-1}\right),$ $a_0 = 0.35875, a_1 = 0.48829, a_2 = 0.14128, a_3 = 0.01168$
Окно Наталла (Nuttall window)	$w(k) = a_0 - a_1\left(\frac{2 \cdot \pi \cdot k}{N-1}\right) + a_2 \cdot \cos\left(\frac{4 \cdot \pi \cdot k}{N-1}\right) - a_3 \cdot \cos\left(\frac{6 \cdot \pi \cdot k}{N-1}\right),$ $a_0 = 0.355768, a_1 = 0.487396, a_2 = 0.144232, a_3 = 0.012604$
Окно Блэкмана — Наталла (Blackman–Nuttall window)	$w(k) = a_0 - a_1\left(\frac{2 \cdot \pi \cdot k}{N-1}\right) + a_2 \cdot \cos\left(\frac{4 \cdot \pi \cdot k}{N-1}\right) - a_3 \cdot \cos\left(\frac{6 \cdot \pi \cdot k}{N-1}\right),$ $a_0 = 0.3635819, a_1 = 0.4891775, a_2 = 0.1365995, a_3 = 0.010641$
Окно с плоской вершиной (Flat top window)	$w(k) = a_0 - a_1\left(\frac{2 \cdot \pi \cdot k}{N-1}\right) + a_2 \cdot \cos\left(\frac{4 \cdot \pi \cdot k}{N-1}\right) -$ $- a_3 \cdot \cos\left(\frac{6 \cdot \pi \cdot k}{N-1}\right) + a_4 \cdot \cos\left(\frac{8 \cdot \pi \cdot k}{N-1}\right)$ $a_0 = 1.0, a_1 = 1.93, a_2 = 1.29, a_3 = 0.388, a_4 = 0.032$
Окно Гаусса (Gaussian window)	$w(k) = exp\left(-\frac{1}{2} \cdot \left(\frac{\pi \cdot k}{N-1}\right)^2\right), A = \frac{N-1}{2}.$

Из представленных оконных функций к полученным значениям, полученным после разложения в ряд Фурье, применяется оконная функция Хэмминга (Рисунок 3.10), для «сглаживания» значений на границах фреймов [103]:

$$H(k) = 0{,}54 - 0{,}46 \cdot \cos\left(\frac{2\pi k}{N-1}\right). \quad (39)$$

То есть результатом будет вектор следующего вида:

$$X(k) = X(k) \cdot H(k), 0 \le k \le N. \quad (40)$$

Применение фильтра заключается в покадровом перемножении его значений со значениями спектра. Результатом этой операции является mel-коэффициент. Сколько будет использовано фильтров, столько же будет получено коэффициентов.

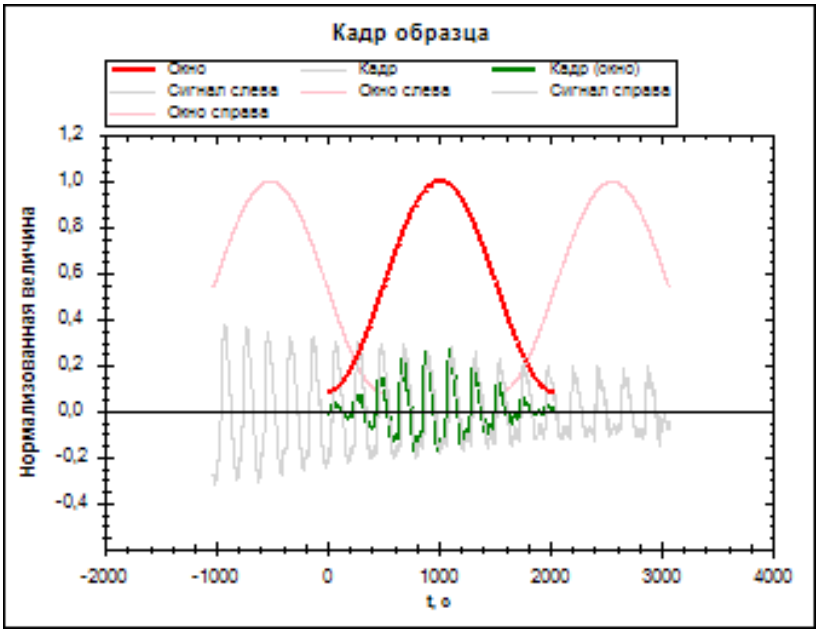


Рисунок 3.10 – Использование «окна Хэмминга» и перекрытия 1/4

3.1.6 Расчет mel-фильтров

Mel – это «психофизическая единица высоты звука», основанная на среднестатистическом субъективном восприятии человека [104, с.105-107]. Зависит в первую очередь от частоты звука (а также от громкости и тембра). Другими словами, эта величина, показывающая, насколько звук определённой частоты «значим» для человеческого слуха.

Преобразовать частоту в мел можно по следующей формуле:

$$M = 1127 \cdot \log\left(1 + \frac{F}{700}\right). \quad (41)$$

Обратное преобразование:

$$F = 700 \cdot \left(e^{\frac{M}{1127}} - 1\right). \quad (42)$$

На рисунке 3.11 представлена нелинейная зависимость mel-единиц от частоты сигнала:

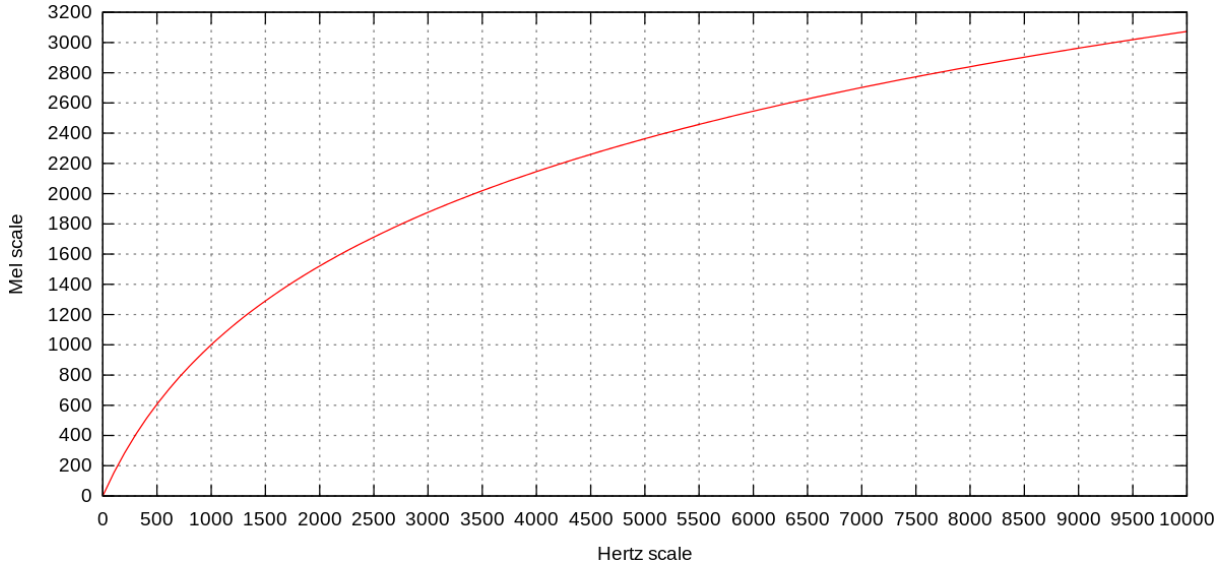


Рисунок 3.11– График зависимости mel / частота [104]

Следующий этап обработки сигнала заключается в разложении полученного выше спектра по mel-шкале. Каждый mel-фильтр – это треугольная оконная функция, позволяющая просуммировать количество энергии на определённом диапазоне частот, чтобы получить mel-коэффициент. Узнав количество мел-коэффициентов и анализируемый диапазон частот строится набор фильтров (рисунок 3.12).

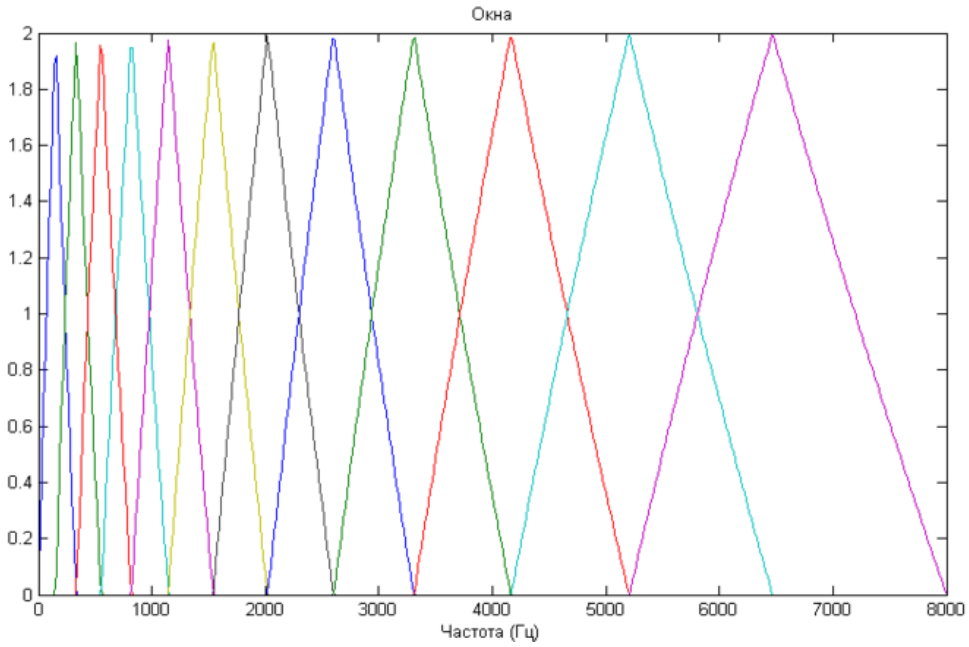


Рисунок 3.12 – Набор mel-фильтров [104]

При увеличении порядкового номера мел-коэффициента увеличивается ширина основания фильтра, поскольку разбиение диапазона частот на обрабатываемые фильтрами диапазоны происходит на шкале mel.

3.1.7 Применение фильтров и логарифмирование энергии спектра

Применение фильтра заключается в попарном перемножении его значений со значениями спектра. Результатом этой операции является mel-коэффициент [104]. Количество коэффициентов равно количеству фильтров, M:

$$S[m] = \log\left(\sum_{k=0}^{N-1} |X[k]|^2 \cdot H_m[k]\right), 0 \le m \le M. \quad (43)$$

3.1.8 Косинусное преобразование

Дискретное косинусное преобразование (DCT) [105] используется для того, чтобы получить кепстральные коэффициенты. Данный алгоритм, широко используется с целью сжатия данных. Подобно быстрому преобразованию Фурье, он преобразует данные в наборы частот. Первые частоты в наборе являются наиболее значимыми, чем последние. Для сжатия данных наименее значимые частоты удаляются на основании допустимых потерь разрешения.

$$C[l] = \sum_{m=0}^{M-1} S[m] \cdot \cos\left(\pi \cdot l \cdot \frac{m + 0{,}5}{M}\right), 0 \le l \le M. \quad (44)$$

В результате, для каждого окна получены выборки [1, M] мел-частотных кепстральных коэффициентов, необходимых в дальнейшем анализе.

3.1.9 Мел-частотные кепстральные коэффициенты

В результате, из полученной выборки окон голосового файла требуется определить численную характеристику, например, среднеквадратическое, но полученные благодаря этому расчету данные не удовлетворяют предстоящему анализу. Вместо этого было решено использовать мел-частотные кепстральные коэффици-

енты, представленные на рисунках 3.13 и 3.14. При обработке звука, кепстр представляет собой кратковременный спектр мощности звука, основанный на линейном косинусном преобразовании логарифмического спектра мощности на нелинейной шкале мелов.

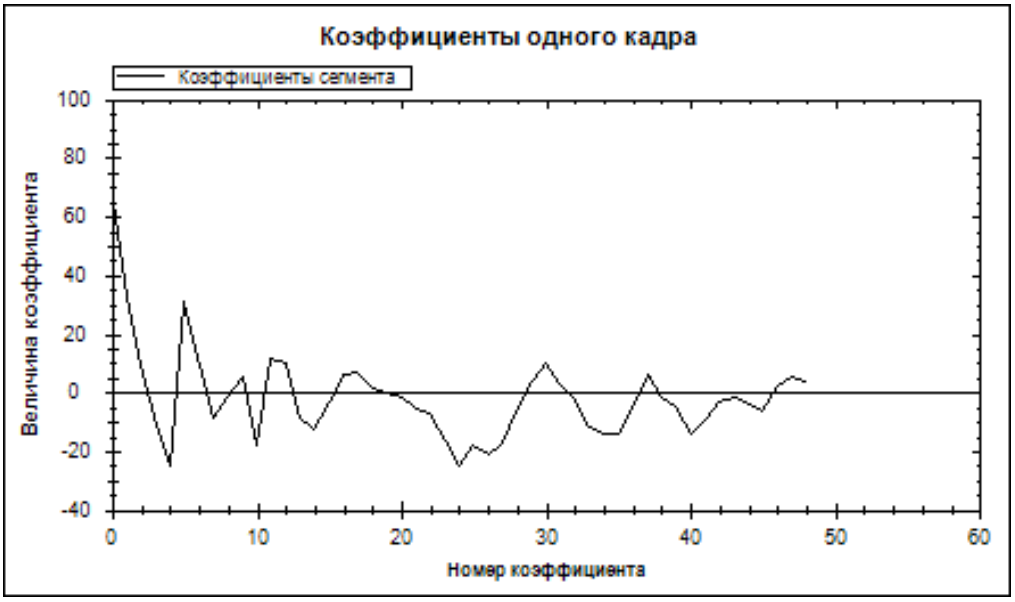


Рисунок 3.13 – Полученные коэффициенты в одном кадре (центральном)

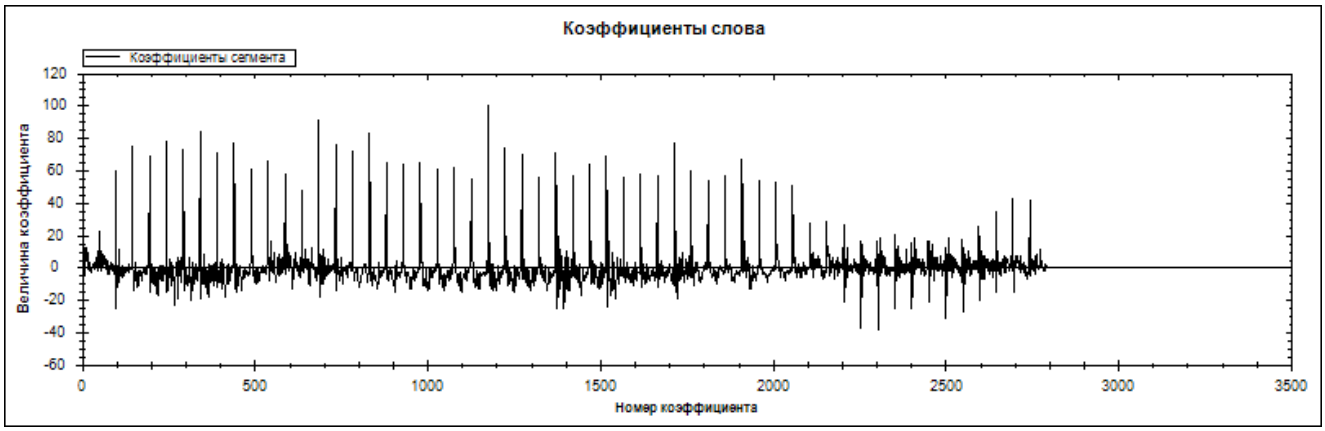


Рисунок 3.14 – Полученные коэффициенты во всем слове целиком

Достоинства применения мел-частотных кепстральных коэффициентов:

• благодаря проекции сигнала на mel-шкалу, получаются наиболее значимые частоты, воспринимаемые человеческим слухом;

• благодаря установлению требуемого количества получаемых коэффициентов уменьшается время обработки и размер обрабатываемого анализа;

• появляется возможность учитывать «волновую природу» сигнала в дальнейшем анализе благодаря использованию спектра сигнала.

3.2 Формирование проектной библиотеки образцов произношения

3.2.1 Группы пользователей проекта

Все пользователи проекта были структурированы по критерию владения русским языком.

• **Экспертные участники.** Участники, обладающие профессиональной компетенцией в оценке речевых образцов (например, филологи).

• **Русскоговорящие участники.** Звуковые файлы, записанные китайскими пользователями, которые прошли проверку экспертными участниками проекта и составили основную базу эталонных образцов.

• **Иностранные участники**. Звуковые файлы, записанные данными пользователями, прошли проверку экспертными участниками проекта и составили основную базу образцов для исследования интонационных конструкций.

3.2.2 Группы звуковых образцов проекта

Записанные образцы были отсортированы в зависимости от принадлежности к той или иной группе пользователей.

• **Эталонный набор.** Состоит из отобранных образцов, русскоговорящих участников проекта. Данная выборка была записана при использовании профессионального оборудования в «студийных условиях» под экспертным контролем филологов. Сформированные образцы принимаются стандартом, с которым впоследствии будет проводится сопоставление образцов, записанных иностранными участниками.

• **Набор образцов иностранных участников (прошедшие контроль).** Состоит из образцов, записанных китайскими пользователями. Сформированные образцы прошли экспертную оценку филологов и впоследствии будут использо-

ваться для установления интонационных конструкций при сопоставлении с образцами из эталонного набора.

• **Набор образцов иностранных участников (не прошедшие контроль).** Состоит из образцов, записанных китайскими пользователями. Полученные образцы не прошли экспертную оценку филологов и впоследствии могут использоваться для формирования профилей основных ошибок в произношении. Данный профиль необходим для разработки системы рекомендаций по устранению неверного произношения.

3.2.3 Основные требования к записи образцов

Требования к файлу:

• частота дискретизации от 44.1 кГц, глубина записи 16 бит;

• расширение файла .wav, .mp3.

Требования к условиям записи:

• минимальный уровень шума;

• изолированная помещение без эха;

• четкое произношение.

3.2.4 Основная информация записанных образцов

Информация записанного образца должна содержать следующие характеристики пользователя проекта:

• родной язык;

• пол участника;

• интонация;

• произносимое слово.

3.2.5 Разработка программного кода обработки речи

В программу необходимо ввести переменные, которым будут присваиваться данные, с поступающих на вход звуковых сигналов (рисунок 3.15).

```
10
11      namespace LingREC
12      {
13          public partial class QuickIntonate : Form
14          {
15              private Signal signalQ = new Signal();
16              private Signal signalN = new Signal();
17              private Signal signalE = new Signal();
18
19              private Markup markupQ;
20              private Markup markupN;
21              private Markup markupE;
22
23              private MFCCSettings settings = new MFCCSettings();
24
```

Рисунок 3.15 – Объявление переменных для эталонов и образца пользователя

На выходе программы необходимо построение корреляционных кривых, результирующих сопоставление с интонационными эталонами для визуального отображения сходства или различия на каждом из участков слова (рисунок 3.16).

```
86                  ComparisionCurve curveQ = new ComparisionCurve(profileE, profileQ);
87                  ComparisionCurve curveN = new ComparisionCurve(profileE, profileN);
88
89
90                  double max = Math.Max(curveQ.total, curveN.total);
```

Рисунок 3.16 – Построение корреляционных кривых сравнения образцов и нахождение максимального из двух тотальных сравнений

Если максимальное сходство ниже установленной планки в 75%, программа делает вывод, что слово произнесено ошибочно (рисунок 3.17), если более, то выводится результат об установленной интонации.

```
93              string solution;
94              if (max < 0.75)
95              {
96                  solution = "Неверное слово";
97              }
98              else
99              {
100                 if (max == curveN.total) solution = "Повествование";
101                 else solution = "Вопрос";
102             }
```

Рисунок 3.17 – Задание результирующих уведомлений идентификации интонационных признаков на основе сравнения образцов на вывод программы

По окончанию обработки и анализа звуковых файлов, требуется вывести на экран полученные результаты, включающие в себя установленную интонацию и корреляционные кривые сопоставления образцов (рисунок 3.18).

```
output.Text = $"Вопрос: {curveQ.total.ToString("F4")}; Повествование {curveN.total.ToString("F4")}; Вывод? {solution}";


Draw.DrawCorrelationCurve(WCurveQ, wordE, signalE, SignalChannel.Left, curveQ);
Draw.DrawCorrelationCurve(WCurveN, wordE, signalE, SignalChannel.Left, curveN);
```

Рисунок 3.18 – Вывод текста и корреляционных кривых

Полный листинг кода программы представлен в приложении А.

Параллельно с разработкой кода проектировалось рабочее окно диалоговой среды для использования программного обеспечения (рисунок 3.19) Окно разделено на 4 сектора для визуального отображения результатов обработки и анализа образца.

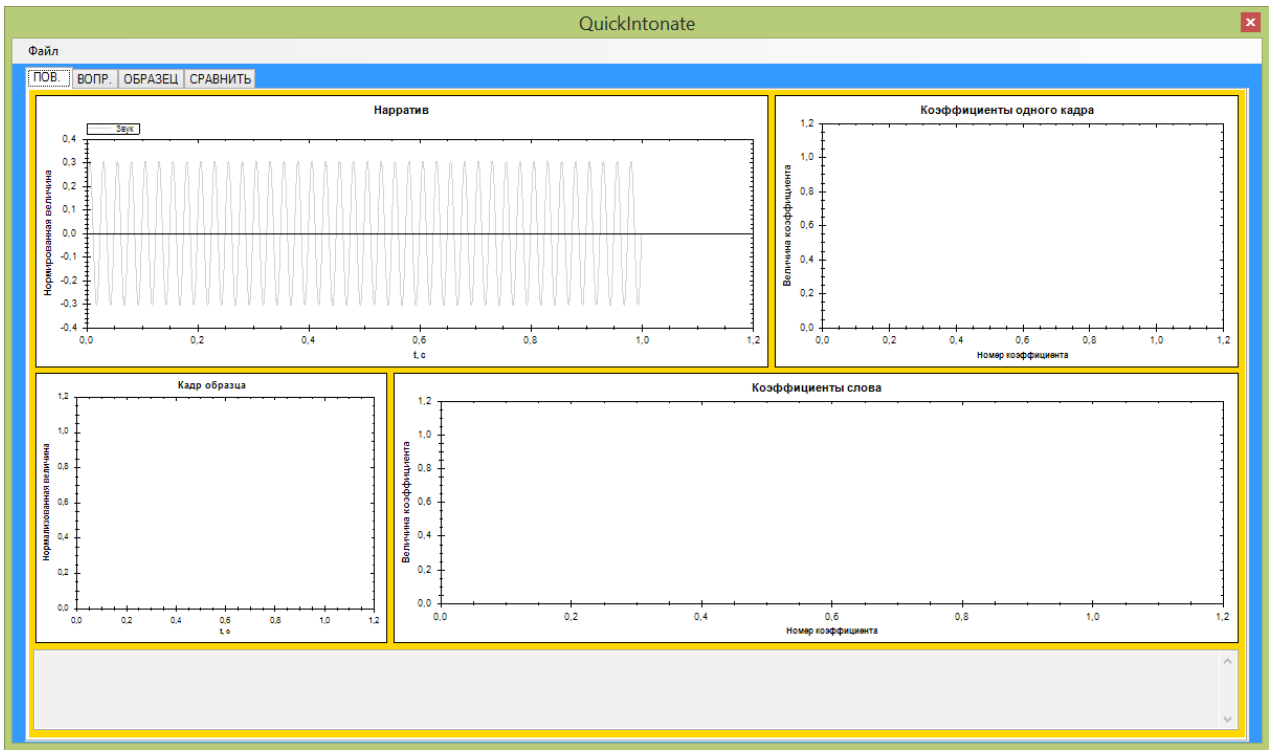


Рисунок 3.19 – Главное рабочее окно программы. Вкладка повествовательного эталона

В рабочее окно были подключены 5 кнопок. При нажатии кнопки «Файл» предлагается выбор загружаемый образец в систему (рисунок 3.20).

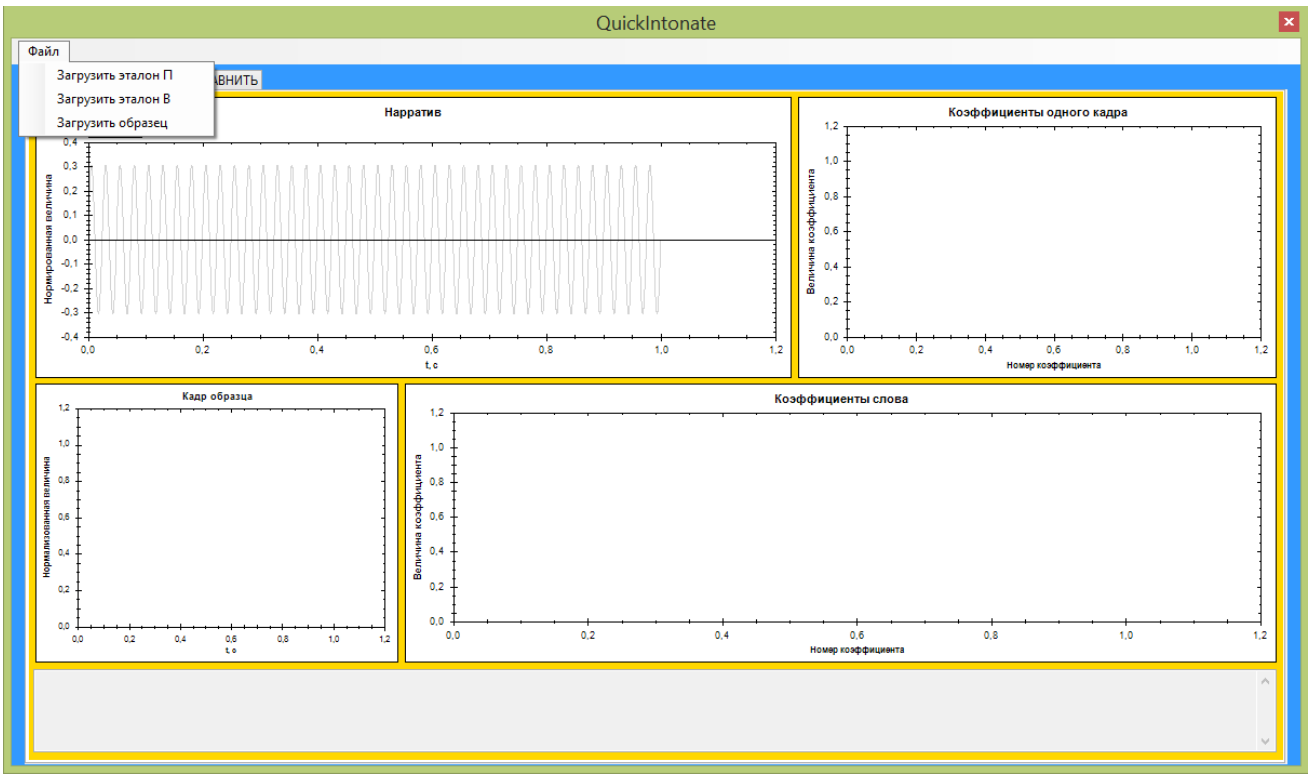


Рисунок 3.20 – Выбор загружаемого образца

После выбора в представленной вкладке строки требуемого образца открывается проводник системы, с помощью которого на вход программы поступает необходимый образец формата .wav или .mp3 (рисунок 3.21).

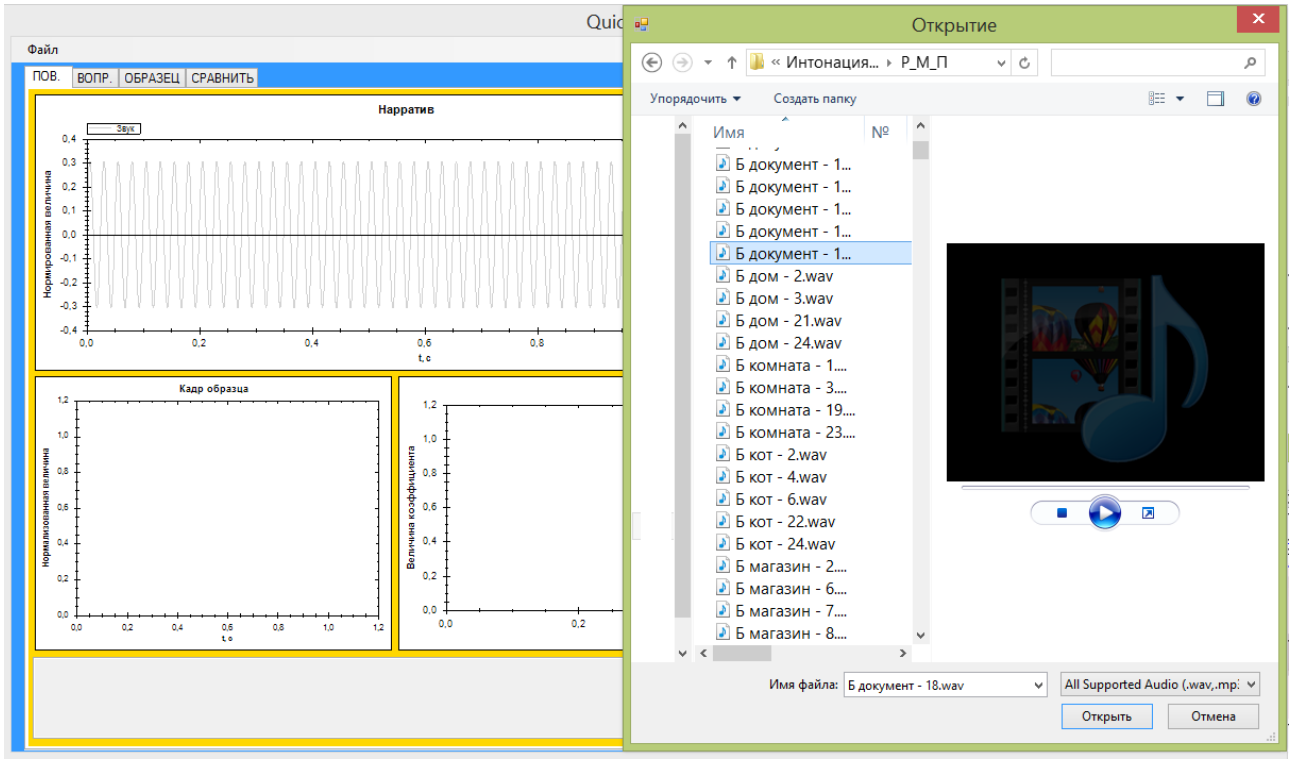


Рисунок 3.21 – Загрузка речевого образца в программу

После автоматической обработки загруженного речевого сигнала в, до этого пустых, сегментах рабочего окна визуально отображаются: звуковой файл в частотной области, центральный кадр с применяемой к нему оконной функцией, мел-частотный рисунок центрального кадра сигнала, центр и мел-частотный портрет целого слова (рисунок 3.22). В нижней области представлена общая информация о загружаемом файле: его название, адрес файла, глубина и частота записи образца, количество используемых каналов.

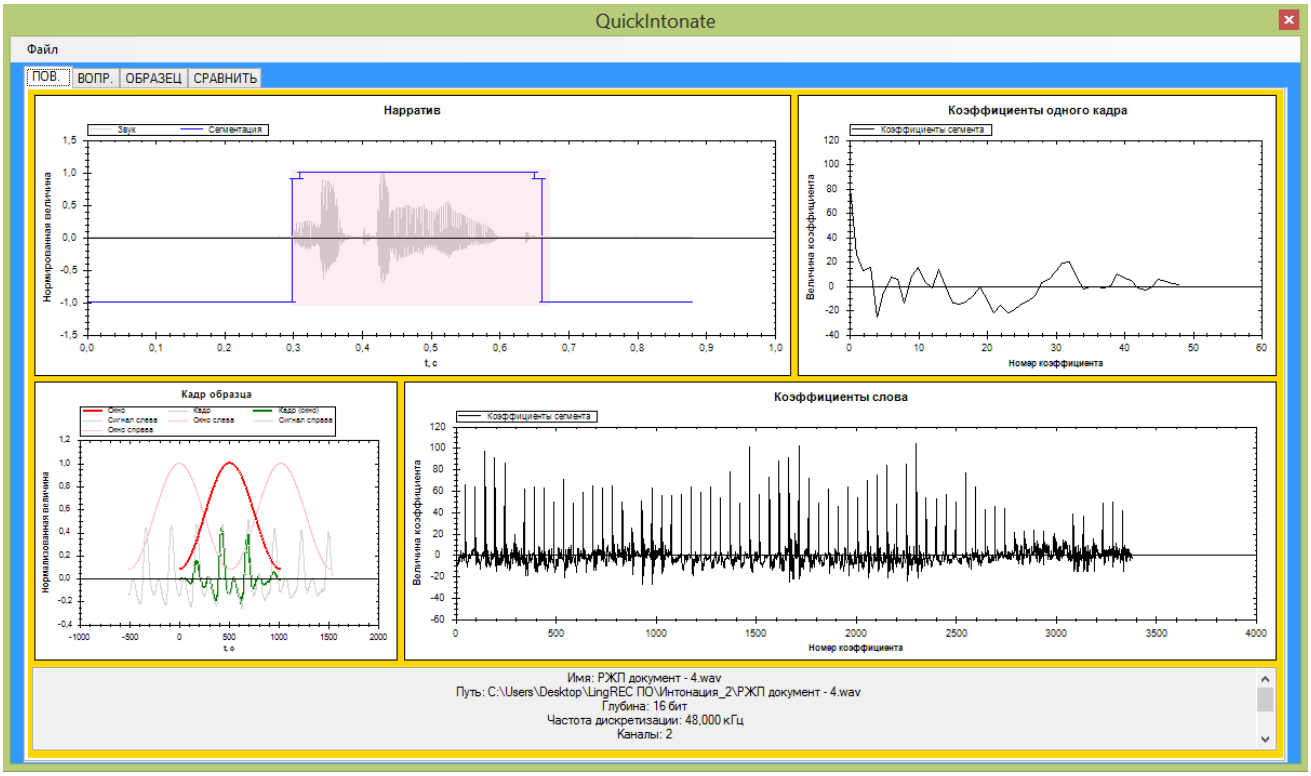


Рисунок 3.22 – Рабочее окно программы после загрузки образца

Результирующим окном программы является вкладка «Сравнить» (рисунок 3.23).

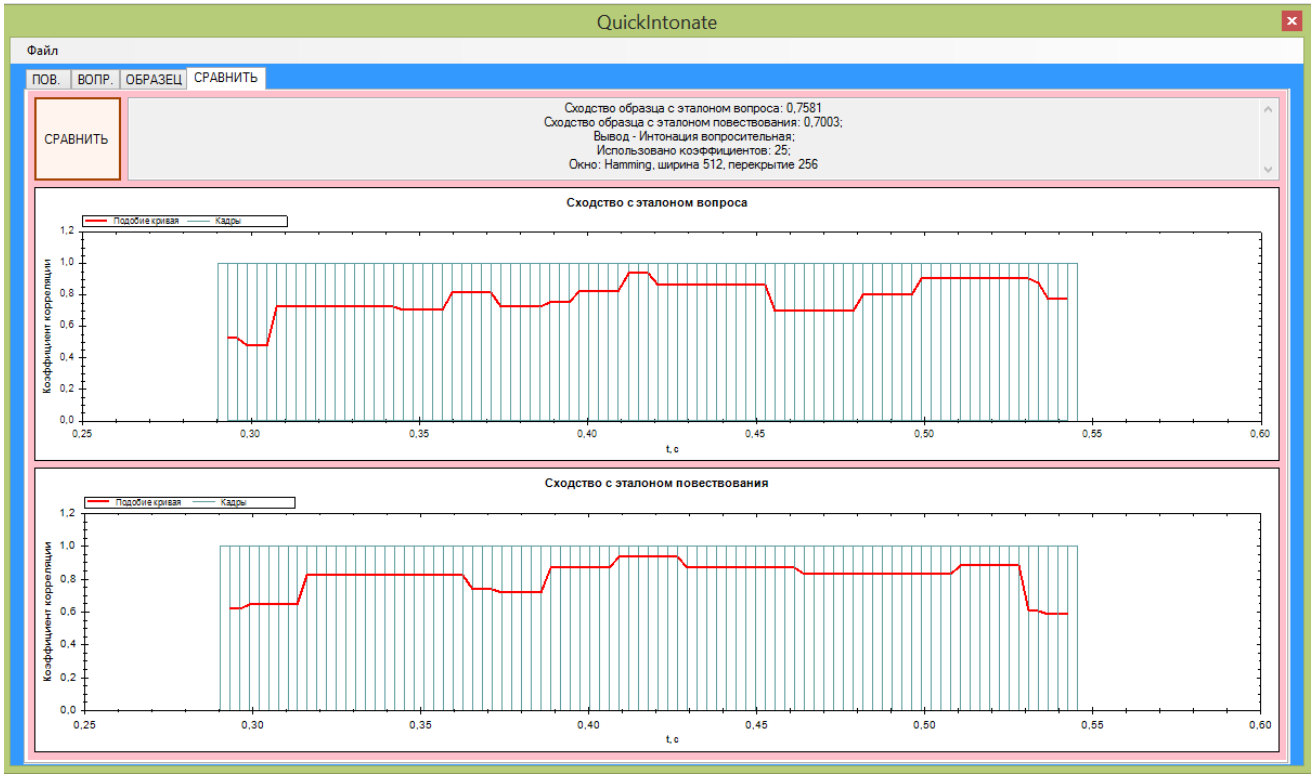


Рисунок 3.23 – Результирующее окно программы

В данном окне визуально представлены две корреляционные кривые, строящиеся на основании полученных мел-частотных кепстральных коэффициентов. Первая кривая строится на основании сопоставления загружаемого образца с эталоном вопросительной интонации, вторая – повествовательной. В верхней части вкладки выводится информация о том, насколько произношение иностранного участника сходится с эталонными образцами проекта. Данный результат берется из среднего значения корреляции. Также выводятся сведения о количестве извлекаемых из кадра мел-частотных кепстральных коэффициентов, применяемую оконную функцию, размер ширина кадра и его перекрытия.

Вывод по разделу 3. На основе изучения реализованных принципов работы и исследований существующих методов обработки речевых сигналов, применяемых в системах распознавания речи, был составлен алгоритм обработки с последующей разработкой программного обеспечения на его основе, и определены требования к образцам-эталонам носителей целевого языка. ПО позволяет объективно произвести анализ интонации входного образца посредством сравнения с загружаемыми эталонами и на основе анализа с разными параметрами подобрать оптимальные значения этих параметров. Следующим этапом ВКР следует экспериментальная проверка, последующие доработка и модернизация программного комплекса анализа речи.

4 РАЗРАБОТКА МОДУЛЯ ПОДБОРА ОПТИМАЛЬНЫХ ПАРАМЕТРОВ ДЛЯ ЛОКАЛЬНОГО РАБОЧЕГО МЕСТА ИССЛЕДОВАТЕЛЯ ИНТОНАЦИОННЫХ КОНСТРУКЦИЙ. ЭКСПЕРИМЕНТАЛЬНАЯ ПРОВЕРКА И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

По завершении разработки программного обеспечения по составленному алгоритму и подготовки библиотеки образцов произношения по сформированным к ним требованиям необходимо экспериментально проверить и подтвердить работоспособность и правильное выполнения программного кода для получения предполагаемого результата.

Данная разработка может послужить дополнением к образовательному мультимедийному тренажеру по обучению фонетике русского языка, основанный сотрудниками и студентами Южно-Уральского государственного университета [98, с. 492].

В данной работе рассматриваются структура и основные функции программы для исследования интонационных признаков речевых образцов. Программа анализа речевого образца основана на алгоритме вычисления мел-кепстральных коэффициентов, которые используются для формирования значений парной корреляции с эталоном речевого образца. Программа обеспечивает выполнение следующих функций:

- вычисление мел-кепстральных частотных характеристик речевого образца;
- визуальное отображение кепстрального портрета сегмента звука;
- численное и графическое сопоставление речевого образца с эталоном,
- поиск в речевом образце участков максимального сходства с эталоном.

В ходе эксперимента было проведен подбор оптимальных параметров входных характеристик сигнала:

- звуковой диапазон;
- ширина кадра;
- перекрытия кадра;
- количество MFCC;
- применяемая оконная функция.

4.1 Создание библиотеки речевых образцов

В поставленном эксперименте в записи звуковых образцов участвовало 18 дикторов в возрасте от 20 до 35 лет, из них восемь носителей русского языка (4 мужского и 4 женского пола) и десять носителей китайского языка (5 мужского и 5 женского пола). Результаты представлены в таблице 7.

Таблица 7 – Выборки речевых образцов, отобранных для участия в эксперименте, распределенные по интонации и типу носителя языка

Носители языка	**Интонация**	**Количество речевых образцов**	
		По интонационным характеристикам	**По типу носителей**
Русского	Вопросительная	600	1200
	Повествовательная	600	
Китайского	Вопросительная	1225	2450
	Повествовательная	1225	

Данная библиотека состоит из речевых образцов, которые были отобраны как наиболее качественные записи для участия в эксперименте в равных количествах.

4.2 Установление частотного диапазона входного сигнала

Человеческая речь находится в среднечастотном звуковом диапазоне, в пределах от 300 до 3400 Гц, так как, в основном, форманты, определяющие ее разборчивость, находятся именно в этой полосе частот (рисунок 4.1) [106, с. 20].

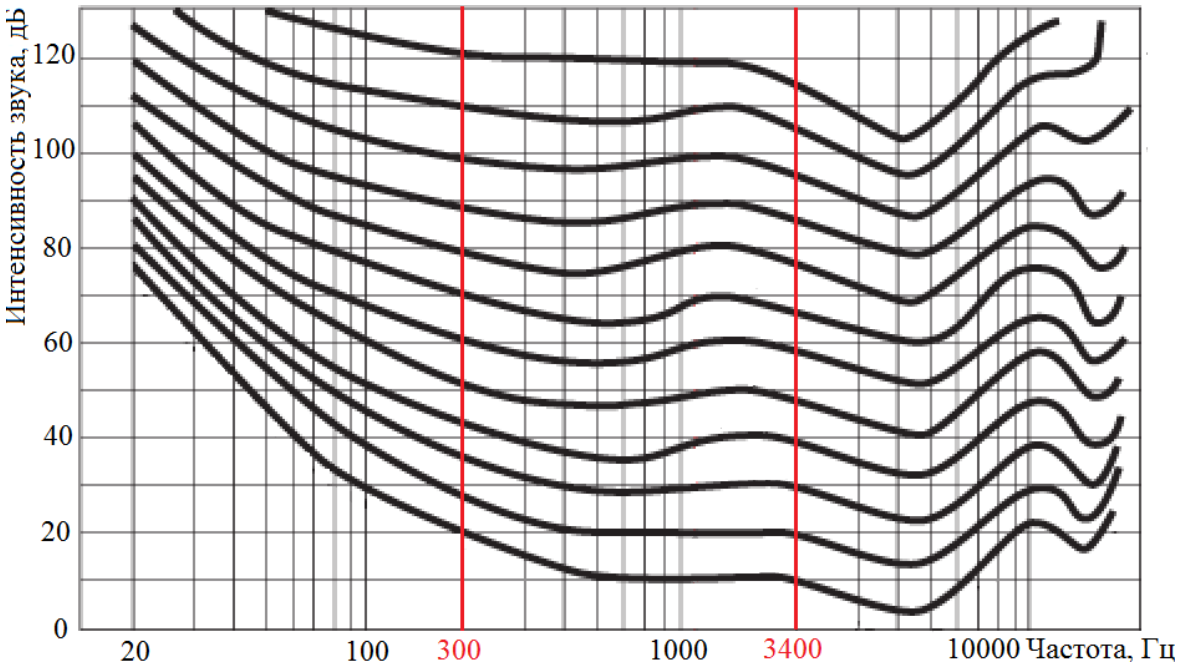


Рисунок 4.1 – Диапазон человеческой речи и громкость в фонах [106]

Опираясь на данную информацию, в программу были установлены параметры использования заданных частот, с целью избежать обработки лишнего неинформативного шума, не относящегося к речи (рисунок 4.2).

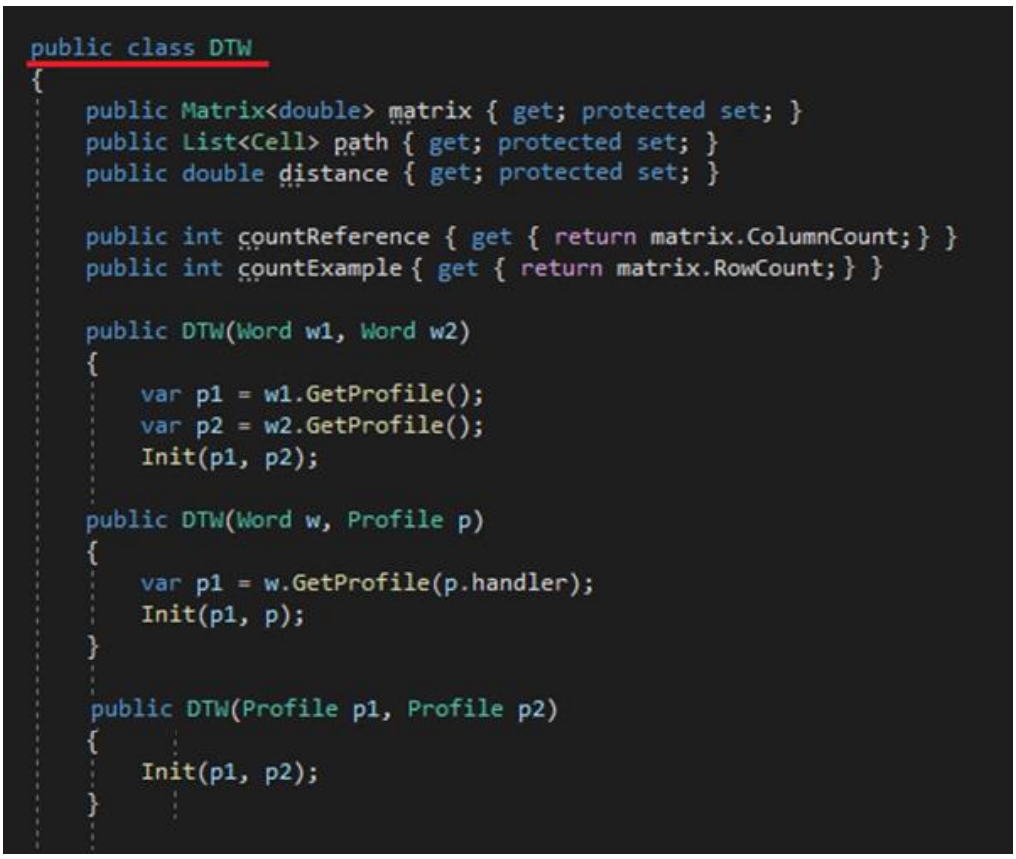
```
public MFCCSettings(int coefficientNumber = 50,
                    double minimalFrequency = 300,
                    double maximalFrequency = 3400,
                    bool cutZero = true)
{
    this.coefficientNumber = coefficientNumber;
    this.minimalFrequency  = minimalFrequency;
    this.maximalFrequency  = maximalFrequency;
    this.cutZero           = cutZero;
}
```

Рисунок 4.2 – Участок программного кода параметризации частоты обработки входного звукового файла

4.3 Применение алгоритма динамической трансформации временной шкалы

Перед сравнением записанные образцы следует привести к единой метрике с эталонными образцами произношения, записанными филологами. Эталонные звуковые файлы записаны преподавателями в рекомендованных оптимальных условиях с использованием профессионального оборудования. Образцы, записанные иностранными студентами, следует нелинейно привести к единой метрике по временной шкале с образцами, записанными филологами. Также данное решение связано с тем, что в планах данная разработка сможет применяться пользователями на личных ПК, и у них может не оказаться полноценной возможности в использовании профессионального микрофона с шумоподавлением, а также существует вероятность озвучки требуемого слова быстрее или медленнее от рекомендуемой нормы и др., поэтому их образец при использовании кадрирования будет нелинейно подстраиваться под эталонные, записанные в условиях, максимально приближенных к идеальным.

Это требуется для того, чтобы в тех случаях, когда звуковой файл эталонного образца имеет, например, длительность 2 секунды, а входной образец иностранного пользователя – 4 секунды, предполагая, что это одно и то же слово, программа могла нелинейно сопоставить кадры слов, в данном случае уменьшив количество кадров в образце, ориентируясь на поступивший в нее у эталон. В этом состоит принцип алгоритма динамического трансформирования времени (dynamic time warping, DTW) [107, с. 69-84], встроенного в разработанную программу (рисунок 4.3).



```
public class DTW
{
    public Matrix<double> matrix { get; protected set; }
    public List<Cell> path { get; protected set; }
    public double distance { get; protected set; }

    public int countReference { get { return matrix.ColumnCount; } }
    public int countExample { get { return matrix.RowCount; } }

    public DTW(Word w1, Word w2)
    {
        var p1 = w1.GetProfile();
        var p2 = w2.GetProfile();
        Init(p1, p2);

    public DTW(Word w, Profile p)
    {
        var p1 = w.GetProfile(p.handler);
        Init(p1, p);
    }

    public DTW(Profile p1, Profile p2)
    {
        Init(p1, p2);
    }
```

Рисунок 4.3 – Объявление класса DTW в коде программы

Данный алгоритм предназначен для измерения подобия между двумя временными последовательностями разной длины по заданным ограничениям и правилам. Например, это используется для установления соответствий между двумя речевыми образцами разной длины, но, предположительно, представляющими одно и то же слово, произнесённое с разной скоростью. Данные последовательности нелинейно «деформируются» по временной области для установления подобия между рядами.

4.4 Применение алгоритма обнаружения речевой активности

После включения в программное обеспечение модуля по нелинейному сопоставлению кадров требовалось внедрить в разработку алгоритм по определению участков речи во входных звуковых файлах для программного удаления тишины и неинформативных участков на записи. Данному условию отвечает VAD-алгоритм (принцип работы описан в подразделе 3.1.1), поэтому было принято ре-

шение о включении механизма его работы в разрабатываемое приложение (рисунок 4.4) [100, с. 102].

```
public static class VAD
{

    public const int cR    = 0; public const string nameR    = "R (рейтинг кадра)";
    public const int cEn   = 1; public const string nameEn   = "En (мгновенная энергия)";
    public const int cLSFM = 2; public const string nameLSFM = "LSFM (лог. мера спектральной плотности)";
    public const int cMDFC = 3; public const string nameMDFC = "MDFC (доминирующая частота)";
    public const int cZn   = 4; public const string nameZn   = "Zn (переходы через нуль)";

    public const int variablesTresh = 3;

    public const int maxVariables = 4;

    public static int maxVariablesAndRating { get { return maxVariables + 1; } }

     public static void FillCBXList(CheckedListBox list)
    {
        list.Items.Clear();
        list.Items.Add(nameR,    false);
        list.Items.Add(nameEn,   true);
        list.Items.Add(nameLSFM,  true);
        list.Items.Add(nameMDFC, false);
        list.Items.Add(nameZn,   false);
    }
```

Рисунок 4.4 – Объявление статического класса VAD в коде программы

На данном скриншоте представлен отрывок кода, в котором объявляется переменная, отвечающая за алгоритм предварительной обработки входного звукового сигнала.

4.5 Применение оконной функции Хэмминга

В программу внедрен инструментарий по выбору и применению оконной функции, структурированный по алфавитному порядку (от окна Бартлетта до треугольного окна) (рисунок 4.5).

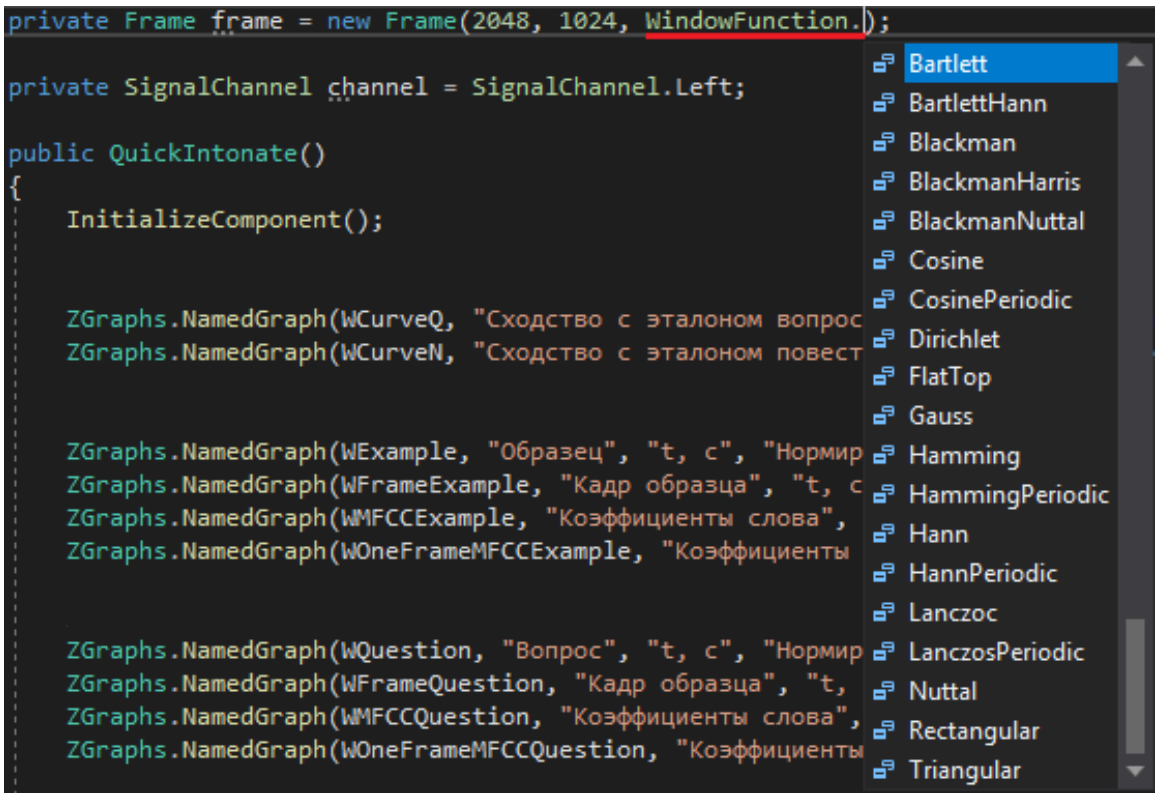


Рисунок 4.5 – Выбор требуемой оконной функции в окне программы

В эксперименте участвовала оконная функция Хэмминга [108, с. 1-7], так как она выигрывает за счет универсальности и простоты относительно других. При использовании «окна Хэмминга» происходит сглаживание краевых эффектов по границам кадра, вследствие чего теряется незначительное количество информации из-за «заглушения» части амплитуд и частот. Другие оконные функции наоборот этого не делают, в результате чего появляются краевые эффекты [73] и, так называемые, утечки спектра на границах кадра [109], что является значимым фактором для обработки и анализа в данном эксперименте. В данном случае критична потеря части спектра, в то время, когда амплитудно-частотная информативность исходного сигнала менее важна.

4.6 Установление размера ширины кадра

При разработке программного обеспечения возникла задача подбора оптимальной ширины кадра, применяемого ко входным образцам звука. Здесь требуется подобрать баланс между разрешением по времени и разрешением по частоте: меньший кадр предоставляет большее разрешение по времени, но меньшее по ча-

стоте – и наоборот. Слишком большое дробление на кадры приводит к неоправданным вычислительным затратам, уменьшает информативность по частоте и, как следствие, сокращает количество доступных коэффициентов для анализа в дальнейшем. Наоборот, слишком широкие кадры могут охватывать сигнал, не являющийся условно-стационарным на выделенном промежутке времени, и не учитывать динамику сигнала.

Например, при использовании N-ной ширины кадра, мы бы подразделяли входной сигнал на количество «A» фреймов, чтобы получить из них те же «A» средних значений корреляционных коэффициентов, которые впоследствии сравнивались бы между собой, данный случай представлен на рисунке 4.6 с использованием ширины кадра в 2048 семплов.

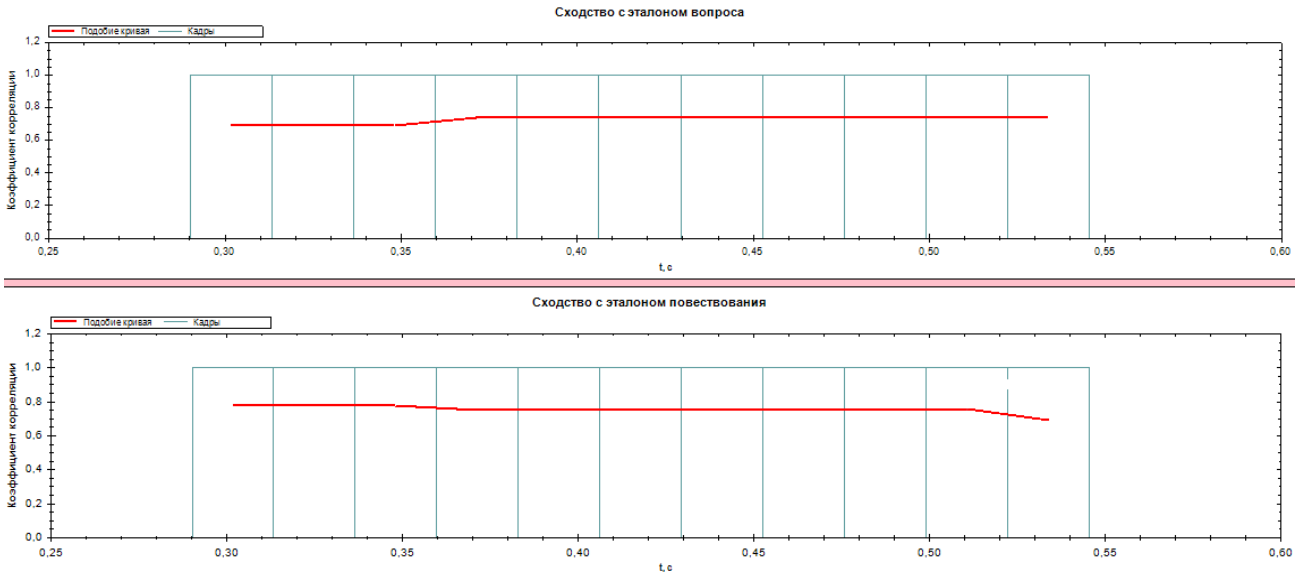


Рисунок 4.6 – Результат работы программы с заданной шириной кадра 2048

Однако, если уменьшить заданную размерность, например, в два раза, установив ширину N/2, в результате количество фреймов соответственно увеличится в два раза. Получившаяся размерность «B» (рисунок 4.7) представляет результат информативнее и точнее.

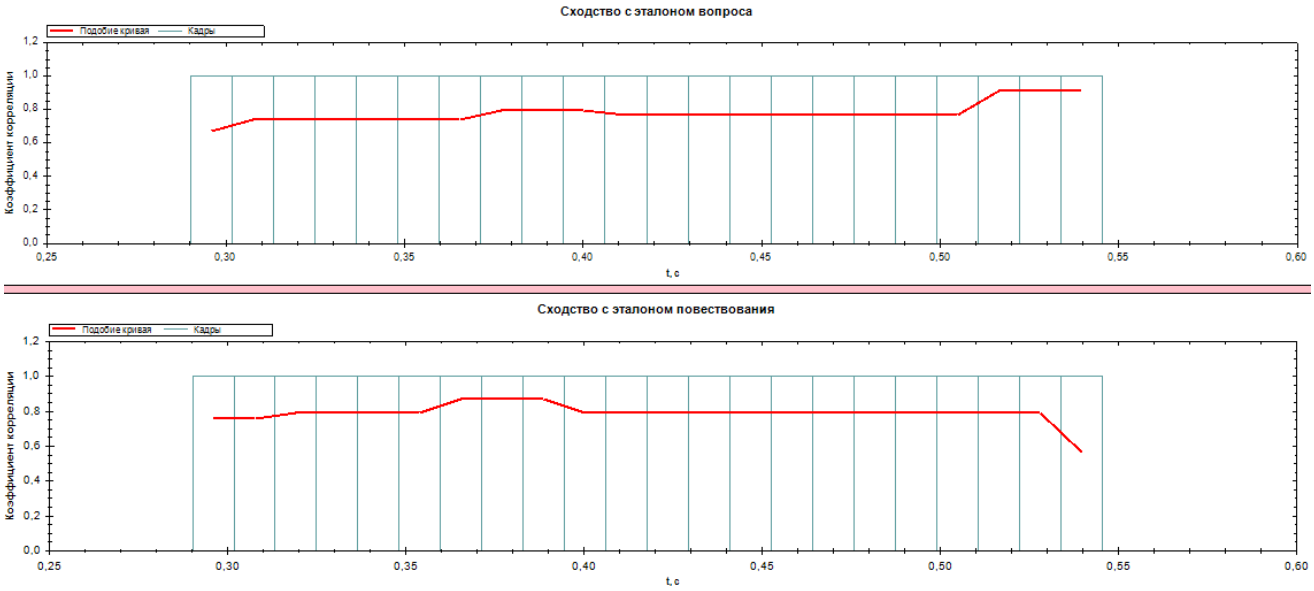


Рисунок 4.7 – Результат работы программы с заданной шириной кадра 1024

Исходя из поставленных задач, было проведено экспериментальное тестирование по определению оптимальных заданных параметров для обработки входных звуковых сигналов. По результатам тестирования, представленного в таблице 8, было установлено, что при увеличении количества кадров путем уменьшения их размера, с определенного момента перестает расти информативность, и даже наоборот, начинает увеличиваться размер шума и искажений, особенно при сравнении образцов.

Таблица 8 – Результаты программы при изменении ширины кадра без использования перекрытия

Ширина кадра, семплов	**Максимальное сходство с эталоном, %**	**Минимальное сходство с эталоном, %**	**Среднее сходство с эталоном, %**	**Правильное определение интонации, %**
8192	93,24	78,32	85,67	62,34
4096	90,12	75,43	85,12	67,12
2048	87,51	68,24	85,33	78,32
1024	89,42	73,14	84,56	74,21
512	79,54	54,21	77,18	73,82
256	75,74	52,18	73,24	71,94

Представленная таблица составлена на основании экспериментального сравнения звуковых файлов по следующему принципу: на вход программы подаются два звуковых эталонных образца, с записанным филологами конкретным словом с повествовательной и вопросительной интонацией, после чего на вход программы поступает файл, записанный иностранным студентом с тем же словом и заданной интонацией. Цель – сравнить данную запись с эталонными образцами. Заранее известна интонация, с которой было произнесено слово иностранным студентом, и если программа выявляет сходство с эталонной записью с данной интонацией более 75%, то результат считается успешным при соблюдении дополнительных условий, таких как:

- результат сравнения с другой интонацией не превышает 75%;
- результат сравнения с такой же интонацией больше результата сравнения с противоположной интонацией минимум на 10%.

Во время тестирования программы участвовало 3650 отобранных звуковых файлов. На его основании, отталкиваясь от наивысшего результата верного определении интонации, был установлен оптимальный размер ширины кадра в 2048 семплов для частот сигнала 44.1 и 48 кГц.

Заданную частоту дискретизации на сегодняшний день может позволить каждая современная звуковая карта на персональном компьютере. Частота дискретизации представляет собой заданное количество семплов в секунду, чем выше ее размер, тем большее количество семплов содержится в единице времени [110]. Семпл, в свою очередь, является минимальным сегментом аудио-сигнала, несущим в себе амплитудное значение звуковой волны [111]. То есть, чем выше частота дискретизации, тем точнее и качественнее результат записи звука.

Определим связь между размером ширины кадра и частоты дискретизации. Установлено, что при обработке звукового сигнала важно, чтобы ширина кадра попадала в диапазон 25-40 мс, так как в данных единицах речевой тракт не изменяет свою конфигурацию, поэтому речевой сигнал считается стационарным, по-

тому что его спектральные характеристики не меняются [112]. При превышении данного предела теряется разрешение в частотной области, а при меньших единицах – во временной [112, 113]. Поэтому при частоте дискретизации от 44100 Гц до 48000 Гц и ширине кадрирования в 2048 семплов, размер кадра во временной области попадает в данный диапазон, принимая значения 35-40мс.

4.7 Установление размера кадрового перекрытия

Следующим этапом следовало экспериментальное определение процентного отношения перекрытия кадров (таблица 9), что служит уменьшением стороннего влияния искажения на краях кадров с помощью сглаживания [72, с 267-278], что приводит к повышению результативности верной идентификации, однако, при этом увеличивая время на обработку звукового сигнала из-за увеличения количества кадров.

Таблица 9 – Результаты программы при параметризации размера перекрытия

Ширина перекрытия, %	**Максимальное сходство с эталоном, %**	**Минимальное сходство с эталоном, %**	**Среднее сходство с эталоном, %**	**Правильное определение интонации,%**	**Время сравнения, с**
90	92,64	78,98	90,23	86,96	16,4
75	92,32	78,87	89,70	86,65	4,1
50	92,51	79,18	90,43	87,20	2,6
25	90,21	78,40	88,16	85,45	2,3
12,5	88,15	76,87	83,66	84,32	1,9

По результатам сравнительного анализа в установлении наилучшего функционального использования оконного перекрытия было определено, что наивысший процент верной идентификации интонационных признаков в совокупности с оптимальным временем распознавания, относительно остальных, добивается при использовании перекрытия в половину кадра.

4.8 Использование мел-частотных кепстральных коэффициентов в корреляции Пирсона

Для получения мел-частотных коэффициентов применялись треугольные оконные фильтры [114, с. 117-121]. Разрешение по частоте является вдвое меньшим, чем размер кадра. При установленном размере кадра в 2048 семплов частотный спектр будет составлять 1024 значения, и, опираясь на ранее установленную методологию расчетов [115, с. 1-4], по заданным параметрам максимальным количеством получаемых характеристик является 50 мел-частотных коэффициентов кадра. Установленная максимальная единица и участвовала в эксперименте.

После получения выборок MFCC-коэффициентов на каждом кадре в количестве 50 штук у загружаемых образцов (эталон с повествовательной интонацией, эталон с вопросительной интонацией и образец иностранного пользователя) происходит их покадровое сравнение с использованием установления коэффициентов корреляции Пирсона [116].

Пусть даны две выборки коэффициентов двух соответствующих друг другу кадров из сравниваемых между собой сигналов a_m и b_m, где m – это количество полученных мел-частотных коэффициентов кадра, в данном случае 50. Тогда коэффициент корреляции Пирсона P_{ab} для двух данных кадров равен:

$$P_{ab} = \frac{\sum_{i=1}^{m}(a_i - \overline{a}) \cdot (b_i - \overline{b})}{\sqrt{\sum_{i=1}^{m}(a_i - \overline{a})^2 \cdot \sum_{i=1}^{m}(b_i - \overline{b})^2}}. \quad (45)$$

Коэффициент корреляции Пирсона между наборами MFCC-коэффициентов, полученных из соответствующих кадров двух сравниваемых сигналов, рассчитывается следующим методом. В числитель заносится разность между конкретным мел-частотным кепстральным коэффициентом кадра эталона (a_i) и средним мел-частотным значением кадра эталона ($\overline{a}$) того же кадра, которая умножается на соответствующую порядковую $i[1, m]$ разность мел-частотного кепстрального коэффициента кадра образца (b_i) и его среднего значения ($\overline{b}$). После чего происхо-

дит суммирование полученных произведений, в количестве штук, соответствующем количеству мел-частотных кепстральных коэффициентов в кадре (50).

Затем, полученное значение числителя, делится на среднеквадратичное отклонение дисперсий соответствующих кадров сигналов. То есть в знаменатель заносится сумма квадратов разности мел-частотного кепстрального коэффициента кадра эталона (a_i) со средним мел-частотным значением кадра эталона ($\overline{a}$). После чего полученное значение умножается на сумму квадратов разности мел-частотного кепстрального коэффициента кадра образца (b_i) и его среднего значения ($\overline{b}$). В знаменатель заносится значение квадратного корня из произведения сумм сопоставляемых выборок мел-частотных кепстральных коэффициентов из соответствующих кадров сигналов эталона и образца.

Результатом решения данной дроби будет являться коэффициент корреляции Пирсона для конкретного кадра, принимающий значение в диапазоне от 0 до 1, где 0 – полное несоответствие, 1 – «идеальное» совпадение мел-частотных кепстральных коэффициентов, извлекаемых из кадра.

Данная операция совершается покадрово, сопоставляя сигналы на каждом сегментированном участке. Допустим в эталонном сигнале N-кадров, в сигнале образца также N-кадров, благодаря применению алгоритма динамической трансформации временной шкалы (DTW) [107, с. 69-84], соответственно будет получено N-значений коэффициентов корреляции Пирсона. Данные значения корреляции заносятся точками на временной шкале и соединяются между собой, в результате на выводе программы строится корреляционная кривая сопоставления входных сигналов эталона и образца (см. рисунки 4.6, 4.7). На корреляционной кривой визуально отображено, насколько два слова похожи друг на друга на каждом из участков. Помимо этого, берется среднее из значений корреляции Пирсона в качестве результирующей, отображающей насколько произношение иностранного участника схоже с эталонными образцами.

На представленном ниже примере изображено сравнение с эталонными образцами входного сигнала произнесенного слова «комната» китайского пользователя с вопросительной интонацией (рисунок 4.8).

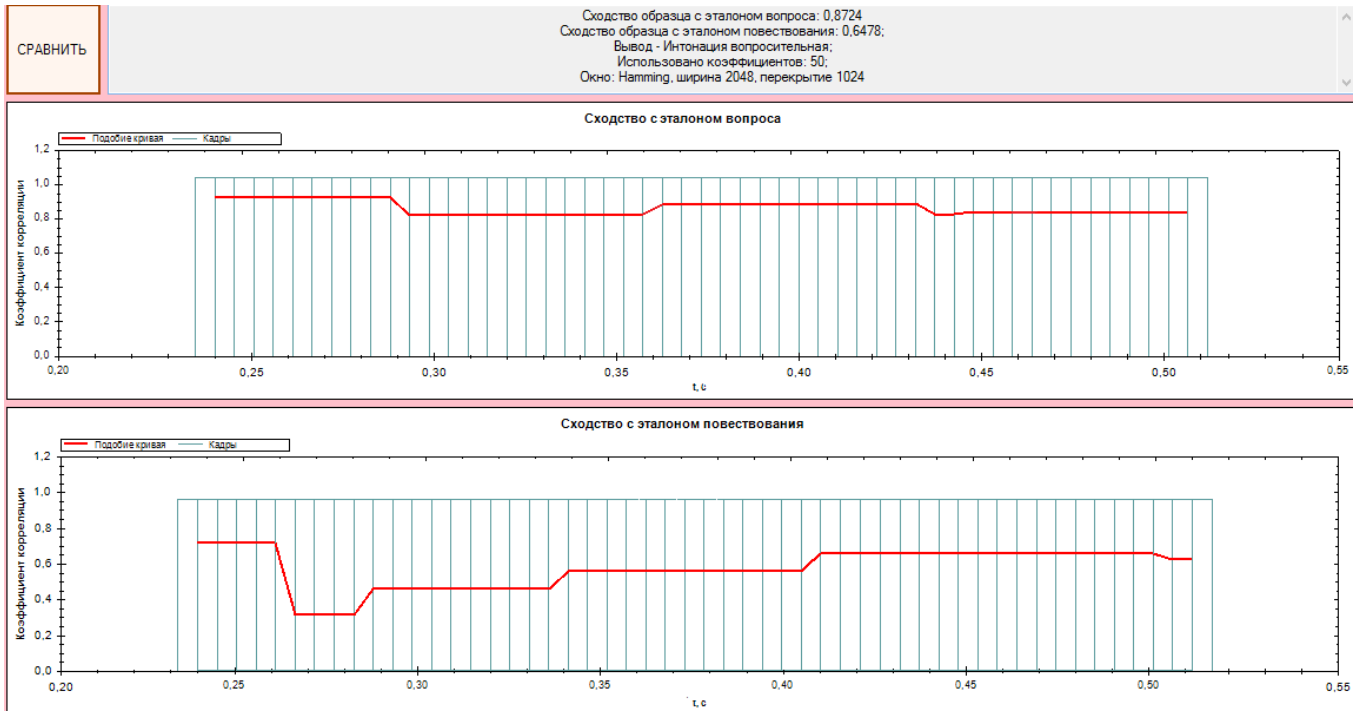


Рисунок 4.8 – Результаты сравнения слова «комната», произнесенного иностранным пользователем

В результате на визуальной составляющей сравнения с повествовательным эталоном, представляющей собой корреляционную кривую, произошла «просадка» в начале слова, являющимся ударным слогом. Данная ситуация могла бы сигнализировать о том, данное слово было произнесено неверно, если бы подобная ситуация отображалась на обоих корреляционных кривых, однако при сравнении с эталоном вопросительной интонации полученные коэффициенты корреляции Пирсона в первых кадрах сигнала близки единице («идеальному» соответствию). Из чего следует вывод, что данное слово было произнесено с вопросительной интонацией, конструкция которой пришлась на первый ударный слог слова. В результате программы получен правильный результат.

Однако существует вероятность, что пользователь может допустить неправильное произношение слова (рисунок 4.9).

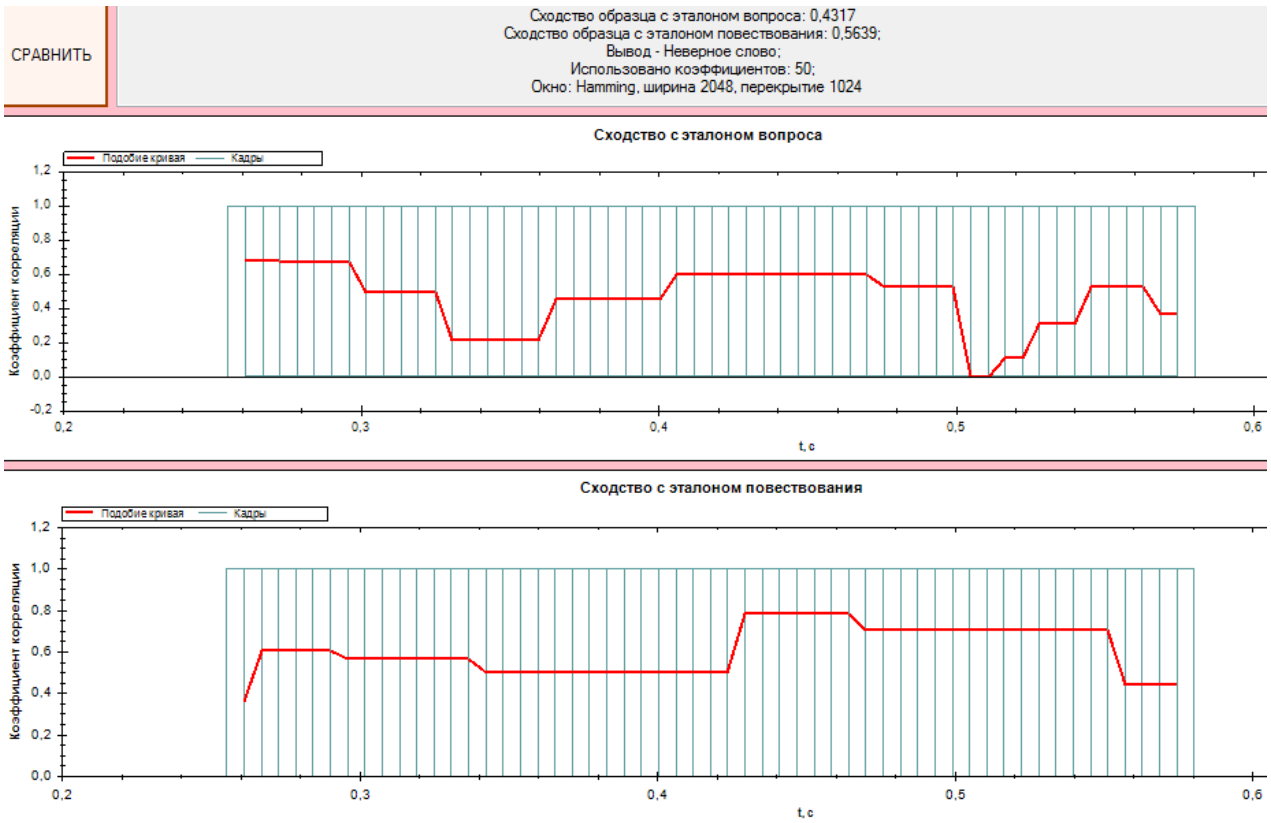


Рисунок 4.9 – Результаты вывода программы при неправильном произношении слова

Благодаря корреляционной кривой в выводе программы, исследователь или пользователь на самостоятельной основе может получить информацию, в каком месте он допустил неверное произношение слова, используя результирующую визуальную составляющую.

4.9 Результаты программной разработки локального рабочего места исследователя интонационных конструкций русского языка

Разработанная программа даёт приемлемый результат в распознавании интонационных признаков в произношении слов. В будущем алгоритм программы можно связать с обучением нейронной сети по выборкам из звуковых фалов для повышения результата тестирования. Результаты работы программы можно увидеть на примере, представленном на рисунках 4.10 – 4.13.

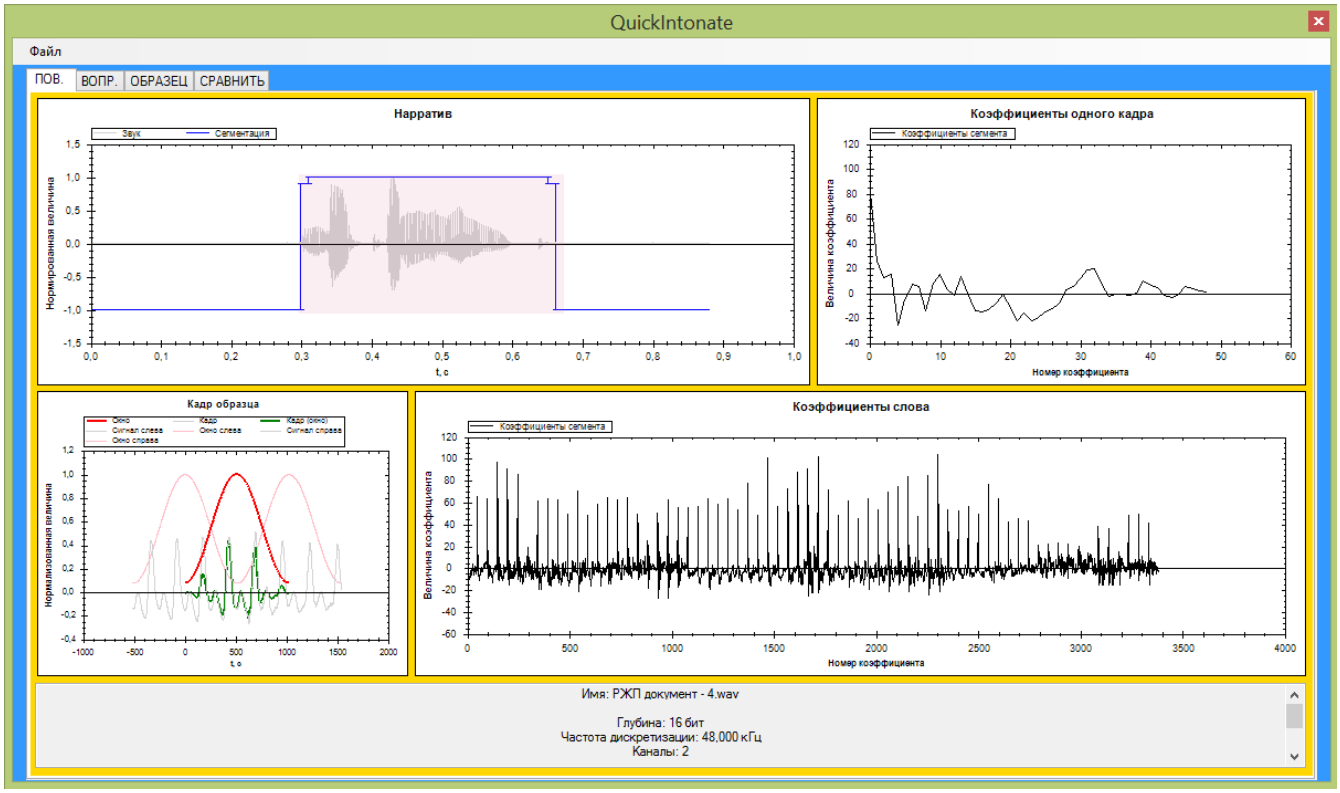


Рисунок 4.10 – Загрузка образца русскоязычного диктора женского пола повествовательной интонации (РЖП) слова «документ» в программу

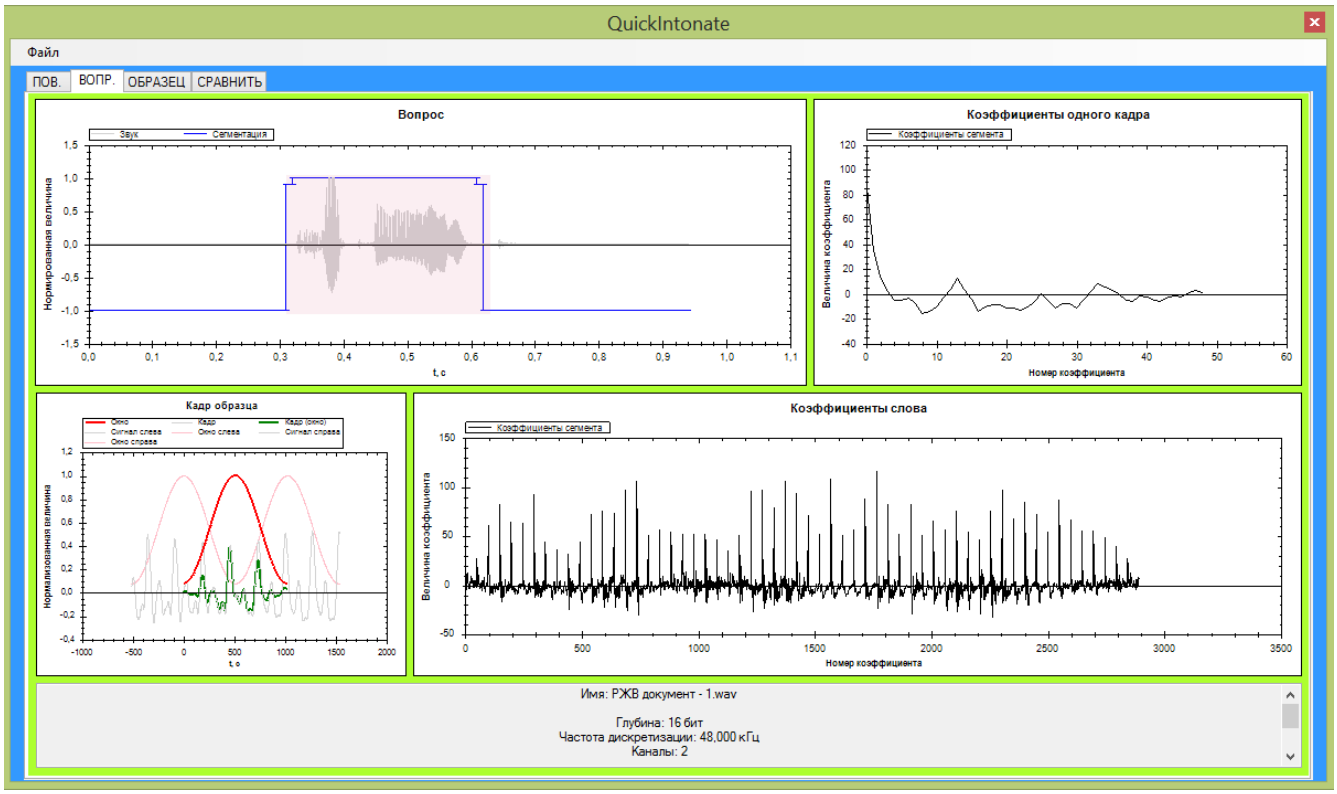


Рисунок 4.11 – Загрузка образца русскоязычного диктора женского пола вопросительной интонации (РЖВ) слова «документ» в программу

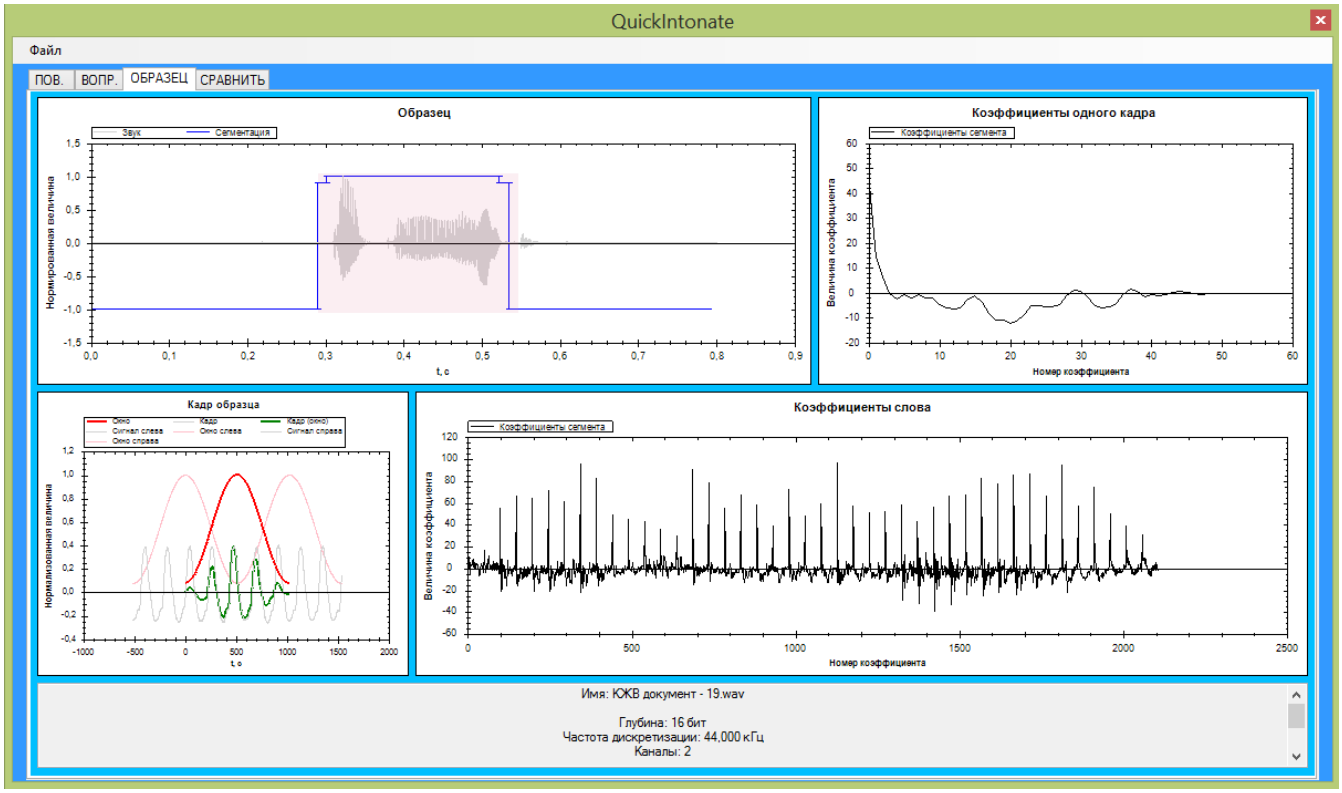


Рисунок 4.12 – Загрузка образца китайского диктора женского пола вопросительной интонации (РЖВ) слова «документ» в программу

Рисунок 4.13 – Результат программы по оценке распределения коэффициентов корреляции при использовании мел-кепстральных коэффициентов

Данное ПО можно модульно подключить к мультимедийному тренажеру по обучению фонетике русского языка на базе Южно-Уральского государственного университета. Подобный сервис может служить инструментом как для преподавателей, занимающихся обучением иностранных студентов, так и для иностранных студентов, изучающих русский как неродной.

Выводы по разделу 4. Для участия в эксперименте по тестированию программного обеспечения была сформирована библиотека речевых образцов. В ходе эксперимента был проведен подбор оптимальных параметров входных характеристик сигнала, таких как: звуковой диапазон, ширина кадра, перекрытие кадра, количество мел-частотных кепстральных коэффициентов и применяемая оконная функция. На выходе программы полученные мел-частотные кепстральные коэффициенты были использованы в корреляции Пирсона для сопоставления входных образцов и визуального представления результата. Разработанное программное обеспечение позволяет экспериментальным образом подобрать оптимальные параметры, которые впоследствии будут использованы в автоматическом режиме работы основной системы.

ЗАКЛЮЧЕНИЕ

В ходе данной работы были рассмотрены системы распознавания речи с закрытым и открытым системным кодом, их принципы работы, выявлены достоинства и недостатки каждой из них. Разобраны термины, классификации, методы и алгоритмы, связанные с преобразованием речи. Рассмотрены достоинства и недостатки формантного анализа и предложена альтернатива существующим методам – использование мел-частотных кепстральных коэффициентов.

Была проведена классификация существующих методов обработки речевых сигналов, применяемых в системах распознавания речи на основе аналитического обзора. Представленные классификации систем, характеристик и методов обработки позволяют на их основании дать оценку возможности применения новых математических аппаратов в задачах обработки речевых сигналов в системах распознавания речи.

На основе изучения реализованных принципов работы и исследований существующих методов обработки речевых сигналов, применяемых в системах распознавания речи, был сформирован алгоритм обработки речи и разработано программное обеспечение на его основе. Определены требования к образцам-эталонам носителей целевого языка. Проведена экспериментальная проверка, последующие доработка и модернизация программного комплекса анализа речи.

Для участия в эксперименте по тестированию программного обеспечения сформирована библиотека речевых образцов. В ходе эксперимента проведен подбор оптимальных параметров входных характеристик сигнала, таких как: звуковой диапазон, ширина кадра, перекрытие кадра, количество мел-частотных кепстральных коэффициентов и применяемая оконная функция. На выходе программы полученные мел-частотные кепстральные коэффициенты использованы в корреляции Пирсона для сопоставления входных образцов и визуального представления результата. Разработанное программное обеспечение позволяет экспериментальным образом подобрать оптимальные параметры,

которые впоследствии будут использованы в автоматическом режиме работы основной системы.

По данной разработке были выполнены две научные публикации в рамках 63ей международной научной конференции Евразийского Научного Объединения.

БИЛИОГРАФИЧЕСКИЙ СПИСОК

1. Zwass, V. Speech recognition technology / V. Zwass // Encyclopedia Britannica. – 2016. –V.8, № 2. – P. 112-113.

2 Liwicki, M. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks / M. Liwicki, A. Graves, J. Schmidhuber // Pattern Recognition Research. –2006. – V.1, №1 – P. 54-87.

3 Роженцева, М.А. Алгоритмы распознавания речи в мобильных приложениях для OS Android / М.А. Роженцева, Э.М. Вихтенко // Информативные технологии и высокопроизводительные вычисления. – 2019. – Вып. 2. – №5 (14). – С. 271-277.

4 Abenova, Z.S. Research into the methods of software product developing and maintaining / Z.S. Abenova, M.N. Petrov // Siberian Journal of Science and Technology. – 2017. – V. 18, №. 4. – P. 706–710.

5. Baciu, A. System and method for performing an action on a phone in response to a user initiating an outbound call to one or more select phone numbers / A Baciu, S. Bhojwani, J. Billa, C. Berner. – Burlington, MA: Nuance Communications Inc., 2013. – 16 p.

6. Calder, D. SpeechKit: a multimedia speech tool / D. Calder // Proc. of the 10th International Conference on Information Integration and Web-based Applications & Services. – 2008. – V. 10. – P. 647–650.

7. Rai, R. Fragmentary shape recognition: A BCI study / Computer-Aided Design. – 2016. – V. 71. – P. 51–64.

8. Kumarage, C.J. Voice Driven Email Client / C.J. Kumarage. – LAP Lambert Academic Publishing, 2019 –176 p.

9. Jeon Y. A multimodal ubiquitous interface system using smart phone for human-robot interaction / Y. Jeon, H. Ahn // Proc. of the 2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). – 2011. – V. 21. – P. 183–192.

10. Kretschmer, F. Persistent data backend for OPC UA namespaces in IT infrastructures / F. Kretschmer, C. Arnim, A. Lechler, A. Verl //Procedia CIRP. – 2018. – V.72. – P. 174–178.

11. Lange, P. Tuning Sphinx to outperform Google's speech recognition API / P. Lange, D. Suendermann // Proc. of the Conference on Electronic Speech Signal Processing (ESSV), 2014. – 2014. – V.42. – P. 32–41.

12. Gutierrez, R.T. Understanding the role of digital commons in the web. The making of HTML5 / R.T. Gutierrez // Telematics and Informatics. – 2018. – V. 35, №5. – P. 1438–1449.

13. Windrum, P. Leveraging technological externalities in complex technologies: Microsoft's exploitation of standards in the browser wars / P. Windrum // Research Policy. – 2004. –V. 33, №3. – P. 385–394.

14. Matarneh, R. Speech recognition systems: a comparative review / R. Matarneh, S. Maksymova, V. Lyashenko, N. Belova // Journal of Computer Engineering (IOSR-JCE). – 2017. – V. 19, №5. – P. 71–79.

15 Gaida, C. Comparing open-source speech recognition toolkits / C. Gaida // Technical report of the Project Oasis. – 2017. – V. 65, №2. – 117–123.

16 El Moubtahij, H. Using features of local densities, statistics and HMM toolkit (HTK) for offline Arabic handwriting text recognition / H. El Moubtahij, A. Halli, K. Satori // Journal of Electrical Systems and Information Technology – 2016. – V. 3. №3. – P. 99-110.

17 Lujan-Mares, M. iATROS: A speech and handwriting recognition system / M. Lujan-Mares, V. Tamarit, V. Alabau // V Journadas en Technologia del Habla – 2008. – P. 75-58.

18 El Amrania, M.Y. Building CMU Sphinx language model for the Holy Quran using simplified Arabic phonemes / M.Y. El Amrania, M.M. Hafizur Rahmanb, M.R. Wahiddinb, A. Shahb // Egyptian Informatics Journal – 2016. – V. 17. №3. – P. 305–314.

19 Ogata, K. Analysis of articulatory timing based on a superposition model for VCV sequences / K. Ogata, K. Nakashima // Proceedings of IEEE International Conference on Systems, Man and Cybernetics – 2014. – January ed. – P. 3720-3725.

20 Алимурадов, А.К. Адаптивный метод повышения эффективности голосового управления / А.К. Алимурадов, П.П. Чураков // Труды международной научно-технической конференции «Перспективные информационные технологии» – 2016. – С. 196-200.

21. Бакаленко В.С. Интеллектуализация ввода-вывода кода программы с помощью речевых технологий: дис. … магистра техники и технологии. – ДонНТУ, Донецк, 2016.

22 Балакшин П.В. Функция плотности длительности состояний СММ. Преимущества и недостатки / П.В. Балакшин // Современные проблемы науки и образования. – 2011. – № 1. – С. 36-39.

23 Беленко М.В. Сравнительный анализ систем распознавания речи с открытым кодом / М.В. Беленко // Сборник трудов V Всероссийского конгресса молодых ученых. Т. 2. – СПб.: Университет ИТМО, 2016. – С. 45-49.

24. Telmem, M. Estimation of the Optimal HMM Parameters for Amazigh Speech Recognition System Using CMU-Sphinx / M. Telmem, Y. Ghanou // Procedia Computer Science. – 2018. – V. 127. – Pages 92-101.

25. Thompson N.C. University licensing and the flow of scientific knowledge / N.C. Thompson, A.A. Ziedonis, D.C. Mowery // Research Policy. – 2018. – V. 47, №6. – P. 1060–1069.

26. Li, X. A fast and memory-efficient N-gram language model lookup method for large vocabulary continuous speech recognition / X. Li, Y. Zhao // Computer Speech & Language. – 2007. – V. 21, №1. – P. 1–25.

27. Polig, R. A hardware compilation framework for text analytics queries / R. Polig, K. Atasu, H. Giefers, C. Hagleitner // Journal of Parallel and Distributed Computing. – 2018. – V. 111. – P. 260-272.

28. Lee, A. Julius – An Open Source Real-Time Large Vocabulary Recognition Engine / A. Lee, T. Kawahara, K. Shikano // European Conference on Speech Communication and Technology. – 2001. – V. 7. – P. 167–170.

29. Lee, A. An Efficient Two-Pass Search Algorithm using Word Trellis Index / A. Lee, T. Kawahara, S. Doshita. // In Proc. International Conference on Speech and Language Processing. – 1998. – P. 1831–1834.

30. Fernau, H. A multi-parameter analysis of hard problems on deterministic finite automata / H. Fernau, P. Heggernes, Y. Villanger // Journal of Computer and System Sciences. – 2015. – V. 81, №4. – P. 747–765.

31. Моисеева, Е.В. Реализация гласных после мягких согласных на стыках слов в современном русском языке: дис. ... кан. филол. наук / Е.В. Моисеева. – М.: Изд. МГУ им. М. В. Ломоносова, 2015. – 217 с.

32. Mattheyses, W. Audiovisual speech synthesis: An overview of the state-of-the-art / W. Mattheyses, W. Verhelst // Speech Communication. – 2015. – V. 66 – P. 182–217.

33. Mas, M. Modus ponens and modus tollens in discrete implications / M. Mas, M. Monserrat, J. Torrens // International Journal of Approximate Reasoning. – 2008. – V. 49, № 2. – P. 422–435.

34. Abbott, D. Hot plug and hot swap / D. Abbott // PCI Bus Demystified. – 2004. – – V. 41 – P. 153–169.

35. Beyerlein, P. Large vocabulary continuous speech recognition of Broadcast News – The Philips/RWTH approach / P. Beyerlein, X. Aubert, R. Haeb-Umbach, M. Harris // Speech Communication. – 2002. – V. 37, №1 – P. 109–131.

36. Swetapadma, A. An innovative finite state automata-based approach for fault direction estimation in transmission lines / A. Swetapadma, A. Yadav // Measurement. – 2017. – V. 99. –P. 13–22.

37. Huang, C. A geometry based efficient decoder for underdetermined MIMO systems / C. Huang, C. Lee, W. Chung, T. Lee // Digital Signal Processing. – 2015. – V. 41 – P. 60–69.

38. Wiesler, S. RASR/NN: The RWTH neural network toolkit for speech recognition / S. Wiesler, A. Richard, P. Golik, R. Schlüter // 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). – 2014. – P. 1–5.

39. Povey, D. The Kaldi Speech Recognition Toolkit / D. Povey, A. Ghoshal, G. Boulianne // Computer Speech & Language. – 2014. – V. 28, № 4. – P. 888–902.

40. Bergner, R. The IATROS study — ibandronate for treatment of renal osteodystrophy / R. Bergner, M. Hoffmann, M. Uppenkamp // Bone. – 2008. – V. 42. – P. 45–46.

41. Li, J. Multivariate time series anomaly detection: A framework of Hidden Markov Models / J. Li, W. Pedrycz, I. Jamal // Applied Soft Computing. – 2017. – V. 60. – P. 229–240.

42. Alaylioglu, H. ALSA: A Hybrid Stress Finite Element Program / H. Alaylioglu // Structural Analysis Systems. – 1986 – P. 1–10.

43. Беленко, М.В. Сравнительный анализ систем распознавания речи с открытым кодом / М.В. Беленко, П.В. Балакшин // Международный научно-исследовательский журнал. –2017. – Вып. 4, №58. – С. 13–18.

44. Hu, G. Mid-to-high frequency piecewise modelling of an acoustic system with varying coupling strength / G. Hu, L. Tang, X. Cui // Mechanical Systems and Signal Processing. – 2019. – V. 1321 – P. 595–608.

45. Duineveld, K. Hierarchical Bayesian analysis of true discrimination rates in replicated triangle tests / K. Duineveld, M. Meyners // Food Quality and Preference. – 2008. – V. 19, №3. –P. 292–305.

46. Оппенгейм, А. Цифровая обработка сигналов / А. Оппенгейм, Р. Шафер. – М.: Техносфера, 2006. – 856 с.

47. Сергиенко, А.Б. Цифровая обработка сигналов / А.Б. Сергиенко. – СПб.: Питер, 2002. – 608 с.

48. Nower, N. Restoration scheme of instantaneous amplitude and phase using Kalman filter with efficient linear prediction for speech enhancement / N. Nower, Y. Liu, M. Unoki // Speech Communication/ – 2015. – V. 70. – P. 13–27.

49. Афанасьев, А.А. Аутентификация. Теория и практика обеспечения безопасного доступа к информационным ресурсам: учебное пособие для вузов / А.А. Афанасьев – М.: Горячая линия - Телеком, 2009. – 564 с.

50. Высоцкий, Г.Я. Алгоритм выделения основного тона спектральными методами для ЭВМ среднего класса / Г.А. Высоцкий, Н.В. Сомин // Дискретная обработка речевых сигналов. – М.: ВЦ АН СССР, 1979. – С. 36–66.

51. Нуссбаумер, Г. Быстрое преобразование Фурье и алгоритмы вычисления сверток / Г. Нуссбаумер: – М.: Радио и связь, 1985. – 248с.

52. Питерсон, Дж. Теория Петри и моделирование систем / Дж. Питерсон – М.: Мир, 1984. – 264 с.

53. Харкевич, А. А. Спектры и анализ / А.А. Харкевич – М.: Государственное издательство технико-теоретической литературы, 1953. – 213 с.

54. Huang, X. Spoken Language Processing. Guide to Algorithms and System Development / X. Huang, A. Acero, H.W. Hon. – Prentice Hall, 2001. – 980 p.

55. Гоноровский, И.С. Радиотехнические цепи и сигналы / И.С. Гоноровский, М.П, Демин – М.: Радио и связь, 1994. – 480с.

56. Дворянкин, С. В. Взаимосвязь цифры и графики, звука и изображения / С.В. Дворянкин // Открытые системы. – 2000. – Вып.1. – №3(14). – С. 34–40.

57. Дворянкин, С.В. Голосовые признаки индивидуальной информативности в речевых системах управления доступом / С.В. Дворянкин, В.В. Кханг // Тезисы докладов научно-технической конференции профессорско-преподавательского состава. – М.: МТУСИ. – 2001. – С. 11–12.

58. Chandaka, S. Support vector machines employing cross-correlation for emotional speech recognition / S. Chandaka, A. Chatterjee, S. Munshi // Measurement. – 2009. – V. 42, № 4. – P. 611–618.

59. Becchetti, C. Speech Recognition. Theory and C++ Implementation / C. Becchetti, L.P. Ricoili – John Wiley and Sons Ltd, 1999. – 428 p.

60. Rabiner L. Fundamentals of speech recognition / L. Rabiner, B.H. Juang – Prentice-Hall Inc., 1993 – 507 p.

61. Rabiner, L. A tutorial on hidden Markov models and selected applications in speech recognition / L. Rabiner // Institute of electrical engineering and radio electronics. – 1989. – V. 2. – P. 86–120.

62. Koolagudi, S.G. Identification of Language using Mel-Frequency Cepstral Coefficients (MFCC) / S.G. Koolagudia, D. Rastogi, K.S. Rao // Procedia Engineering. – 2012. – V. 38. – P. 3391–3398.

63. Иванов, И.И. Анализ метода мел-частотных кепстральных коэффициентов применительно к процедуре голосовой аутентификации / И.И. Иванов // Актуальные проблемы гуманитарных и естественных наук. – 2015. – Вып. 10, №1. – С. 106–114.

64. Trancoso, I. Accent identification / I. Trancoso, C. Teixeira, A. Serralheiro // Institute of electrical engineering and radio electronics. – 1996. – P. 1784–1787.

65. Arslan, L.M. Frequency characteristics of foreign accented speech / L.M. Arslan, H.L. Hansen // In Proc. 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing. – Institute of electrical engineering and radio electronics. – 1997. – P. 1123–1126.

66. Kumpf, K. Foreign speaker accent classification using phoneme-dependent accent discrimination models and comparisons with human perception benchmarks / K. Kumpf, R.W. King // In Proc. European Conference on Speech Communication and Technology 1997. – 1997 – P. 2323–2326.

67. Кодзасов, С.В. Общая фонетика. / С.В. Кодзасов, О.Ф. Кривнова. – М.: РГГУ, 2001. – 592 с.

68. Fung, P. Fast accent identification and accented speech recognition / P. Fung, W.K. Liu // Proceedings. 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. – 1997. – V. 1. – P. 93–97.

69. Herring, P. BEEP: A Python library for Battery Evaluation and Early Prediction / P. Herring, C. B. Gopal, M. Aykol // SoftwareX. – 2018. – V. 11. – P. 81–86.

70. Johansson, S. Spectral analysis of time domain induced polarization waveforms / S. Johansson, P. Hedblom, T. Dahlin // Journal of Applied Geophysics. – 2018. – V. 159. – P. 446–452.

71. Wu, B. Audio signal separation via a combination procedure of time-reversal and deconvolution process / B. Wu, G. Too, S. Lee // Mechanical Systems and Signal Processing. – 2010. – V. 24, № 5. – P. 1431–1443.

72. Trethewey M.W. Window and overlap processing effects on power estimates from spectra / M.W. Trethewey // Mechanical Systems and Signal Processing. – 2000. – V. 12, № 2. – P. 267–278.

73. Archila-Suerte, P. Central processing of speech sounds and non-speech sounds with similar spectral distribution: An auditory evoked potential study / P. Archila-Suerte, J. Zevin, A.E. Hernandez // Brain and Language. – 2015 – V. 141. – P. 35–49

74. Герман, Д. Я. Конспект лекций по курсу «Цифровая обработка сигналов» / Д. Я. Герман. – М.: МГТУ им. Н. Э. Баумана, 2009. – 174 с.

75. Peeters, C. A comparison of cepstral editing methods as signal pre-processing techniques for vibration-based bearing fault detection / C. Peeters, P. Guillaume, J. Helsen // Mechanical Systems and Signal Processing. – 2017. – V 91. – P. 354–381.

76. Lantz, E. Subpixel signal centering and shift measurement using a recursive spectral phase algorithm / E. Lantz // Signal Processing. – 1989 – V. 17, № 4. – P. 365–372.

77. Miller, D.G. Comparison of vocal tract formants in singing and nonperiodic phonation / D.G. Miller, A.M. Sulter, H.K. Schutte, R.F. Wolf // Journal of Voice. – V. 11, №1. – P. 1–11.

78. Баскаков, С. И. Радиотехнические цепи и сигналы / С. И. Баскаков. – М.: Высшая школа, 2001. – 214 с.

79. Козлов, А.С. Кепстральный анализ в задачах слепой оценки скорости передачи цифровых данных / А.С. Козлов, В.Н. Малышев // Радиотехника. – 2012. – № 7. – С. 67–71.

80. Любимов, А.Ю. Линейное предсказание речи – это просто / А. Любимов, М.М. Евсиков // Монитор. – 1995. – № 4. – С. 30–35.

81. Huang, X. Spoken Language Processing: A Guide to Theory, Algorithm, and System Development / X. Huang, A. Acero, H. Hon. – Prentice Hall, 2001. – 980 p.

82. Чураков П.П. Современные методы обработки речевых сигналов / П.П. Чураков, А.Ю. Тычков, А.К. Алимурадов. – Пенза: Изд-во ПГУ, 2014. – 72 с.

83 Огнев, И. В. Распознавание речи методами скрытых марковских моделей в ассоциативной осцилляторной среде / И. В. Огнев, П. А. Парамонов // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2013. – № 3 (27). – С. 115–126.

84 Моттль, В. Скрытые марковские модели в структурном анализе сигналов / В. Моттль, И. Мучник. – М.: Физматлит, 1999. – 352 с.

85 Goldenstein, S. Time warping of audio signals / S. Goldenstein, J. Gomes // Computer Graphics International, 1999. Proceedings. – 1999. – P. 52–57.

86. Sundberg J. Objective Characterization of Phonation Type Using Amplitude of Flow Glottogram Pulse and of Voice Source Fundamental / J. Sundberg // Journal of Voice. – 2020. – P. 16–24.

87. Потапова, Р.К Основы многоаспектного исследования «электронной личности» по голосу и речи в информационно-коммуникационной среде Интернета / Р.К. Потапова, В.В. Потапов // Человек: Образ и сущность. Гуманитарные аспекты. – 2017. – С. 87–111.

88. Карабалаева, М.Х. Алгоритмы пофонемного распознавания речи в амплитудно-временном пространстве / М.Х. Карабалаева., А.А. Шарипбаев // Знания-Онтологии-Теории. – 2009. – Т. 1, №4. – 245–252.

89. Шелепов, В.Ю. Амплитудная сегментация речевого сигнала, использующая фильтрацию и известный фонетический состав / В.Ю. Шелепов, А.В. Ниценко А. // Искусственный интеллект. – 2003. – № 3. – С. 421–426.

90. Свидетельство № 2018612612. Программа анализа речевых образцов: свидетельство о государственной регистрации программы для ЭВМ № 201861261

Рос. Федерация / Кацай Д.А., Исупова Т.Д.; заявитель и правообладатель ФГАОУ ВО «ЮУрГУ» (НИУ), – № 2017663963; заявл. 28.12.2017; зарегистрировано в Реестре программ для ЭВМ 20.02.2018. – 4 с.

91. Свидетельство № 2017663709. Тренажёр речевых образцов для лингафонного кабинета: свидетельство об офиц. регистрации программы для ЭВМ № 2017663709 Рос. Федерация / Филиппов Д.С., Бирюков А.Д., Кацай Д.А., Исупова Т.Д., Анисимов Я.О.; заявитель и правообладатель ФГАОУ ВО «ЮУрГУ» (НИУ), – № 2017660272; заявл. 12.10.2017; зарегистрировано в реестре программ для ЭВМ 11.12.2017. – 4 с.

92. Свидетельство № 2017663710. Сервер для тренажёра речевых образцов лингафонного кабинета: свидетельство об офиц. регистрации программы для ЭВМ № 2017663710 Рос. Федерация / Анисимов Я.О., Кацай Д.А., Исупова Т.Д., Бирюков А.Д., Филиппов Д.С.; заявитель и правообладатель ФГАОУ ВО «ЮУрГУ» (НИУ), – № 2017660277; заявл. 12.10.2017; зарегистрировано в реестре программ для ЭВМ 11.12.2017. – 4 с.

93. Свидетельство № 2018620324. Речевые образцы представителей разных национальностей для программы изучения русского языка: свидетельство о государственной регистрации базы данных № 2018620324 Рос. Федерация / Кацай Д.А., Исупова Т.Д., Филиппов Д.С., Шестакова Л.И., Харченко Е.В., Березовская Я.Л., Казакова Ю.В., Доронина Е.Г., Валеева Д.М., Панова Р.С.; заявитель и правообладатель ФГАОУ ВО «ЮУрГУ» (НИУ), – № 2017621601; заявл. 28.12.2017; зарегистрировано в Реестре баз данных 11.12.2017. – 8 с.

94. D. A. Katsay. Analysis of Speech Samples as a Means of Intensification of Teaching Phonetics of a Foreign Lanuage / D.A. Katsay, E.G. Doronina, Yu.V. Kazakova, E.V. Kharchenko, T.D. Isupova, // ICERI2017 Proceedings. – 2017. – P. 2771–2778.

95. Y. L. Beresovskaya. An Intelligent System of Analysis of Intonation structures: Application in Teaching the Russian Language to the Chinse Language Native Speakers / Y.L. Beresovskaya, T.D. Isupova, D.A. Katsay, O.I. Sharafutdinova, L.I. Shestakova,

O.B. Elagina, // Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. – 2018. – V. 7. – № 4. – P. 105–121.

96. Т. Д. Исупова. Программа идентификации речевых образцов с использованием метода кепстрального анализа / Т.Д. Исупова, Д.А. Кацай // Материалы XX конференции молодых ученых с международным участием. Под общей редакцией В.Г. Пешехонова. – СПб., 2018. – С. 142–143.

97. Д. А. Кацай. Алгоритм анализа речевых образцов иностранных студентов, изучающих русский язык / Д.А. Кацай, Т.Д. Исупова, Е.В. Харченко // Международная научно-практическая интернет-конференция «Актуальные вопросы описания и преподавания русского языка как иностранного/неродного»: Сборник материалов под общ. ред. Н.В. Кулибиной. – М., 2018. – С. 43–48.

98. Д. А. Кацай. Структура тренажёра для совершенствования речи иностранных студентов, изучающих русский язык / Д.А. Кацай, Т.Д. Исупова, Е.В. Харченко, А.Д. Бирюков, Д.С. Филиппов // Международная научно-практическая интернет-конференция «Актуальные вопросы описания и преподавания русского языка как иностранного/неродного»: Сборник материалов под общ. ред. Н.В. Кулибиной. – М., 2018. – С. 492–495.

99. Я. О. Анисимов. Алгоритмический подход к определению интонационной характеристики звукового образца / Я.О. Анисимов, Д.А. Кацай, Т.Д. Исупова, // Информационные технологии в управлении. – СПб.: ГНЦ РФ ОАО ЦНИИ Электроприбор, 2018. – С. 614–622.

100. Ramachandran, R. Modern Methods of Speech Processing / R. Ramachandran, R. Mammone. – Springer Science & Business Media, 2012. – 102 p.

101. Backstrom, T. Introduction to Speech Processing / T. Backstrom – Espoo: Aalto University Inc, 2020. – 613 p.

102. Senin, P. Dynamic Time Warping Algorithm Review / P. Senin // Computer Science. Information and Computer Science Department University. – 2008. – P. 97–120.

103. Chithra, P. Role of Windowing Techniques in Speech Signal Processing for Enhanced Signal Cryptography / P. Chithra, R. Aparna // The International Journal of Advanced Manufacturing Technology. – 2018. – P. 43–51.

104. Радзишевский, А.Ю. Основы аналогового и цифрового звука / А.Ю. Радзишевский. – М.: Вильямс, 2006. – 288 с.

105. Laska, B. Discrete cosine transform particle filter speech enhancement / B. Laska, M. Bolic, R. Goubran // Speech Communication. – 2010. – V. 52, №9. – P. 762–775.

106. ITU-T P.310–2000. International telecommunication union. Telecommunication standardization sector of ITU. – Geneva: ITU, 2001 – 20 p.

107. Muller M. Dynamic time warping. / M. Muller // In Information Retrieval for Music and Motion. – 2007. – Chap. 4. – P. 69–84.

108. Podder, P. Comparative Performance Analysis of Hamming, Hanning and Blackman Window. / P. Podder, T. Zaman, M. Haque // International Journal of Computer Applications. – 2014. – V. 65, №18. – P. 1–7.

109. Fulop, S.A. Speech Spectrum Analysis / S.A. Fulop. – Berlin: Springer, 2011. – 214 p.

110.. Gomes, D. The effectiveness of different sampling rates in vegetation high-impedance fault classification / D. Gomes, C. Ozansoy, A. Ulhaq // Electric Power Systems Research. – 2019. – V. 174. – P. 74–82.

111. Adcock, B. Computing reconstructions from nonuniform Fourier samples: Universality of stability barriers and stable sampling rates / Adcock, M. Gataric, J. L. Romero // Applied and Computational Harmonic Analysis. – V. 46, №2. – 2019 – P. 226-249.

112. Gunnar, F. Speech Acoustics and Phonetics: Selected Writings / F. Gunnar. – Kluwer Academic Publishers. – 2004. – 334 p.

113. Огнев, И. В. Предварительная обработка речевого сигнала для построения базы произношений одиночных слов / И.В. Огнев, П.А. Парамонов // Информаци-

онные средства и технологии: труды Международной научно-технической конференции (20 – 22 октября 2012 г.) в 3 т. – М.:МЭИ, 2012. – 1 т. – С. 53–58.

114. Sirko M. Computing mel-frequency cepstral coefficients on the power spectrum / M. Sirco, M. Pitz // Lehrstuhl fur Informatik VI, Computer Science Department. – 2002. – V. 65, №2. – P. 117–121.

115. Bhadragiri J.M. Speech recognition using MFCC and DTW / J.M. Bhadragiri, B.N. Ramesh // 2014 International Conference on Advances in Electrical Engineering (ICAEE). – 2014. – P. 1–4.

116. Zhou. H. A new sampling method in particle filter based on Pearson correlation coefficient / H. Zhou, Z. Deng, Y. Xia, M. Fu // Neurocomputing. – V. 216. – 2016 – P. 208-215.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Листинг программы.

Класс QuickIntonate.

```
using MathNet.Numerics.LinearAlgebra;
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using ZedGraph;
namespace LingREC
{
   public partial class QuickIntonate : Form
   {
      private Signal signalQ = new Signal();
      private Signal signalN = new Signal();
      private Signal signalE = new Signal();
      private Markup markupQ;
      private Markup markupN;
      private Markup markupE;
      private MFCCSettings settings = new MFCCSettings(50, 300, 3400, true);
      private Frame frame = new Frame(1024, 512, WindowFunction.Hamming);
      private SignalChannel channel = SignalChannel.Left;
      public QuickIntonate()
      {
         InitializeComponent();
         ZGraphs.NamedGraph(WCurveQ, "Сходство с эталоном вопроса", "t, c", "Коэффициент корреляции");
         ZGraphs.NamedGraph(WCurveN, "Сходство с эталоном повествования", "t, c", "Коэффициент корреляции");
         ZGraphs.NamedGraph(WExample, "Образец", "t, c", "Нормированная величина");
         ZGraphs.NamedGraph(WFrameExample, "Кадр образца", "t, c", "Нормализованная величина");
         ZGraphs.NamedGraph(WMFCCExample, "Коэффициенты слова", "Номер коэффициента", "Величина коэффициента");
         ZGraphs.NamedGraph(WOneFrameMFCCExample, "Коэффициенты одного кадра", "Номер коэффициента", "Величина коэффициента");
         ZGraphs.NamedGraph(WQuestion, "Вопрос", "t, c", "Нормированная величина");
```

```
            ZGraphs.NamedGraph(WFrameQuestion, "Кадр образца", "t, c", "Нормализованная величина");
            ZGraphs.NamedGraph(WMFCCQuestion, "Коэффициенты слова", "Номер коэффициента", "Величина коэффициента");
            ZGraphs.NamedGraph(WOneFrameMFCCQuestion, "Коэффициенты одного кадра", "Номер коэффициента", "Величина коэффициента");
            ZGraphs.NamedGraph(WNarrative, "Нарратив", "t, c", "Нормированная величина");
            ZGraphs.NamedGraph(WFrameNarrative, "Кадр образца", "t, c", "Нормализованная величина");
            ZGraphs.NamedGraph(WMFCCNarrative, "Коэффициенты слова", "Номер коэффициента", "Величина коэффициента");
            ZGraphs.NamedGraph(WOneFrameMFCCNarrative, "Коэффициенты одного кадра", "Номер коэффициента", "Величина коэффициента");
            UpdateGraphs();
        }
        private void UpdateGraphs()
        {
            ZGraphs.Clear(WExample);
            ZGraphs.Clear(WQuestion);
            ZGraphs.Clear(WNarrative);
            Draw.DrawSignal(WExample, signalE, channel, true, "Звук");
            Draw.DrawSignal(WQuestion, signalQ, channel, true, "Звук");
            Draw.DrawSignal(WNarrative, signalN, channel, true, "Звук");
            if (markupE != null) Draw.DrawMarkup(WExample, markupE, signalE, channel);
            if (markupQ != null) Draw.DrawMarkup(WQuestion, markupQ, signalQ, channel);
            if (markupN != null) Draw.DrawMarkup(WNarrative, markupN, signalN, channel);
        }
        private void ApplySettings()
        {
            SettingsMFCC.settings = settings;
            SettingsMFCC.framingMode = SettingsMFCC.FramingMode.Classic;
            SettingsClassicFrame.frame = frame;
        }
        private void Check()
        {
            try
            {
                ApplySettings();
                if (markupE == null) { Debug.Error("Загрузите образец"); return; }
```

```
        if (markupQ == null) { Debug.Error("Загрузите эталон вопросительной интонации"); return; }
        if (markupN == null) { Debug.Error("Загрузите эталон повествовательной интонации"); return; }
        if (markupE.wordCount == 0) { Debug.Error("В файле образца не найдено слов"); return; }
        if (markupQ.wordCount == 0) { Debug.Error("В файле эталона вопросительной интонации не найдено слов"); return; }
        if (markupN.wordCount == 0) { Debug.Error("В файле эталона повествовательной интонации не найдено слов"); return; }
        Word wordE = markupE.words[0];
        Word wordQ = markupQ.words[0];
        Word wordN = markupN.words[0];
        Profile profileE = new Profile(wordE);
        Profile profileQ = new Profile(wordQ);
        Profile profileN = new Profile(wordN);
        ComparisionCurve curveQ = new ComparisionCurve(profileE, profileQ);
        ComparisionCurve curveN = new ComparisionCurve(profileE, profileN);
        double max = Math.Max(curveQ.total, curveN.total);
        string solution;
        if (max < 0.75)
        {
           solution = "Неверное слово";
        }
        else
        {
           if (max == curveN.total) solution = "Интонация повествовательная";
           else solution = "Интонация вопросительная";
        }
        double corrQuestion  = curveQ.total;
        double corrNarrative = curveN.total;
        List<string> log = new List<string>();
        log.Add($"Сходство образца с эталоном вопроса: {corrQuestion.ToString("F4")}");
        log.Add($"Сходство образца с эталоном повествования: {corrNarrative.ToString("F4")};");
        log.Add($"Вывод - {solution};");
        log.Add($"Использовано коэффициентов: {profileE.handler.settings.coefficientNumber};");
        log.Add($"Окно: {profileE.handler.frame.function}, ширина {profileE.handler.frame.width}, перекрытие {profileE.handler.frame.overlapping}");
        output.Lines = log.ToArray();
        Draw.DrawCorrelationCurve(WCurveQ, wordE, signalE, channel, curveQ);
```

```
            Draw.DrawCorrelationCurve(WCurveN, wordE, signalE, channel, curveN);
        }
        catch (Exception e)
        {
            Debug.Log(e);
        }
    }
    private enum SignalType { Question, Narrative, Example }
    private void DrawWindowGraphs(Word word, Signal signal, ZedGraphControl
GFrame, ZedGraphControl GFrameMFCC, ZedGraphControl GFullMFCC)
    {
        ZGraphs.Clear(GFrame);
        ZGraphs.Clear(GFrameMFCC);
        ZGraphs.Clear(GFullMFCC);
        Selection selection = word.wordSelection;
        Draw.DrawFrameInContext(GFrame, signal, channel, selection, frame);
        Profile profile = word.GetProfile();
        int middleFrame = profile.count / 2;
        Vector<double> fullMfcc = profile.ReturnFullMFCC();
        Vector<double> frameMFCC = profile.ReturnPartitialMFCC(middleFrame,
middleFrame);
        Draw.DrawMFCC(GFullMFCC, fullMfcc);
        Draw.DrawMFCC(GFrameMFCC, frameMFCC);
    }
    private void Open(Signal signal, TextBox output, SignalType signalType)
    {
        ApplySettings();
        try
        {
            if (!signal.OpenFile()) return;
            output.Lines = signal.InfoList().ToArray();
            switch (signalType)
            {
                case SignalType.Question:
                    markupQ = new Markup(signalQ, channel);
                    ZGraphs.Clear(WQuestion);
                    Draw.DrawSignal(WQuestion, signalQ, channel, true, "Звук");
                    if (markupQ != null)
                    {
                        Draw.DrawMarkup(WQuestion, markupQ, signalQ, channel);
                        if (markupQ.wordCount != 0)
                        {
                            Word word = markupQ.words[0];
```

```
                    DrawWindowGraphs(word, signalQ, WFrameQuestion,
WOneFrameMFCCQuestion, WMFCCQuestion);
                }
            }
            break;
        case SignalType.Narrative:
            markupN = new Markup(signalN, channel);
            ZGraphs.Clear(WNarrative);
            Draw.DrawSignal(WNarrative, signalN, channel, true, "Звук");
            if (markupN != null)
            {
                Draw.DrawMarkup(WNarrative, markupN, signalN, channel);
                if (markupN.wordCount != 0)
                {
                    Word word = markupN.words[0];
                    DrawWindowGraphs(word, signalN, WFrameNarrative,
WOneFrameMFCCNarrative, WMFCCNarrative);
                }
            }
            break;
        case SignalType.Example:
            markupE = new Markup(signalE, channel);
            ZGraphs.Clear(WExample);
            Draw.DrawSignal(WExample, signalE, channel, true, "Звук");
            if (markupE != null)
            {
                Draw.DrawMarkup(WExample, markupE, signalE, channel);
                if (markupE.wordCount != 0)
                {
                    Word word = markupE.words[0];
                    DrawWindowGraphs(word, signalE, WFrameExample,
WOneFrameMFCCExample, WMFCCExample);
                }
            }
            break;
        default: throw new NotImplementedException("Неизвестный тип, изме-
нилось перечисление?");
      }
    }
    catch (Exception e)
    {
      Debug.Log(e);
    }
```

```
        }
        private void загрузитьЭталонПToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Open(signalN, fileInfoNarrative, SignalType.Narrative);
        }
        private void загрузитьЭталонВToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Open(signalQ, fileInfoQuestion, SignalType.Question);
        }
        private void загрузитьОбразецToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Open(signalE, fileInfoExample, SignalType.Example);
        }
        private void btnCheck_Click(object sender, EventArgs e)
        {
            Check();
        }
        private void TabPage1_Click(object sender, EventArgs e)
        {
        }
        private void TabPage2_Click(object sender, EventArgs e)
        {
        }
    }
}
```

Класс MFCC.

```
using System;
using System.Numerics;
using MathNet.Numerics.LinearAlgebra;
namespace LingREC
{
    public class MFCCSettings
    {
        #region Settings Data
        private int m_coefficientNumber = 50;
        public int coefficientNumber
        {
            get { return m_coefficientNumber; }
```

```
        protected set
        {
            if (value < 2) value = 2;
            m_coefficientNumber = value;
        }
    }
    private double m_minimalFrequency = 300;
    public double minimalFrequency
    {
        get { return m_minimalFrequency; }
        protected set
        {
            if (value < 1) value = 1;
            m_minimalFrequency = value;
        }
    }
    private double m_maximalFrequency = 3400;
    public double maximalFrequency
    {
        get { return m_maximalFrequency; }
        protected set
        {
            if (value < m_minimalFrequency) value = m_minimalFrequency + 1;
            m_maximalFrequency = value;
        }
    }

    public bool cutZero { get; protected set; } = false;
    #endregion
    public MFCCSettings(int coefficientNumber = 50,
                        double minimalFrequency = 300,
                        double maximalFrequency = 3400,
                        bool cutZero = true)
    {
        this.coefficientNumber = coefficientNumber;
        this.minimalFrequency  = minimalFrequency;
        this.maximalFrequency  = maximalFrequency;
        this.cutZero           = cutZero;
    }
    public MFCCSettings Copy()
    {
        return new MFCCSettings(coefficientNumber, minimalFrequency, maximalFre-
quency, cutZero);
```

```
    }
    public bool SameAs(MFCCSettings other)
    {
        if (cutZero != other.cutZero) return false;
        if (coefficientNumber != other.coefficientNumber) return false;
        if (minimalFrequency  != other.minimalFrequency)  return false;
        if (maximalFrequency  != other.maximalFrequency)  return false;
        return true;
    }

}

public class MFCC
{
    public double Fs { get; private set; }
    public int N { get; private set; }
    public int[] f { get; private set; }
    public MFCCSettings settings { get; private set; } = null;
    public int M { get { return mel.Count - 2; } }
    public Vector<double> mel { get; private set; }
    public Vector<double> herz { get; private set; }
    public Vector<double> setSm { get; private set; }
    public Vector<double> MCC { get; private set; }
    public MFCC(double samplingRate, MFCCSettings settings, Frame frame)
    {
        this.settings = settings;
        mel = CreateVector.DenseOfArray(new double[settings.coefficientNumber +
2]);
        herz = CreateVector.DenseOfArray(new double[settings.coefficientNumber +
2]);
        double minMel = settings.minimalFrequency.FrequencyToMel();
        double maxMel = settings.maximalFrequency.FrequencyToMel();
        double melStep = (maxMel - minMel) / (settings.coefficientNumber + 1);
        for (int i = 0; i < settings.coefficientNumber + 2; i++)
        {
            mel[i]  = minMel + melStep * i;
            herz[i] = mel[i].MelToFrequency();
        }
        SetupForFrame(samplingRate, frame.width);
    }
    public Vector<double> EvaluateFrame(double[] frame)
    {
        Complex[] XSpectrum = FourierTransform.FFT(frame);
```

```
    return EvaluateFrameSpectrum(XSpectrum);
}
public Vector<double> EvaluateFrame(Vector<double> frame)
{
    Complex[] XSpectrum = FourierTransform.FFT(frame);
    return EvaluateFrameSpectrum(XSpectrum);
}
public Vector<double> EvaluateFrameSpectrum(Complex[] XSpectrum)
{
    Vector<double> output = GetCepstralCoefs(XSpectrum);
    if (settings.cutZero) { output = output.SubVector(1, output.Count - 1); }
    return output;
}
private void SetupForFrame(double samplingRate, int frameSize)
{
    N = frameSize;
    Fs = samplingRate;
    f = new int[M + 2];
    for (int i = 0; i < M + 2; i++)
    {
        f[i] = (int)Math.Floor((frameSize + 1) * herz[i] / samplingRate);
    }
}
private double Hm(int k, int m)
{
    if (k < f[m - 1]) return 0;
    if ((k >= f[m - 1]) && (k < f[m]))
        return ((double)(k - f[m - 1])) / (f[m] - f[m - 1]);
    if ((k >= f[m]) && (k < f[m + 1]))
        return ((double)(f[m + 1] - k) / (f[m + 1] - f[m]));
    return 0;
}
private Vector<double> GetCepstralCoefs(Complex[] XSpectum)
{
    int length = XSpectum.Length;
    Vector<double> S = CreateVector.DenseOfArray(new double[M]);
    for (int m = 1; m < M; m++)
    {
        double summ = 0;
        for (int k = 0; k < length; k++)
        {
            summ += Math.Pow(XSpectum[k].Magnitude, 2) * Hm(k, m);
        }
```

```
            S[m] = Math.Log(summ);
        }
        setSm = S;
        Vector<double> c = CreateVector.DenseOfArray(new double[M]);
        for (int n = 0; n < M; n++)
        {
            double summ = 0;
            for (int m = 0; m < M; m++)
            {
                summ += S[m] * Math.Cos(Math.PI * n * (m + 0.5) / M);
            }
            c[n] = summ;
        }
        MCC = c.Clone();
        return c;
    }
  }
}
```

Класс DTW.

```
using System;
using System.Collections.Generic;
using MathNet.Numerics.LinearAlgebra;
using static System.Math;
namespace LingREC
{
  public struct Cell
  {
    public int row;
    public int column;
    public double value;
  }
  public class DTW
  {
    public Matrix<double> matrix { get; protected set; }
    public List<Cell> path   { get; protected set; }
    public double distance       { get; protected set; }
    public int countReference { get { return matrix.ColumnCount; } }
    public int countExample   { get { return matrix.RowCount;    } }
    public DTW(Word w1, Word w2)
    {
      var p1 = w1.GetProfile();
```

```
            var p2 = w2.GetProfile();
            Init(p1, p2);
        }
        public DTW(Word w, Profile p)
        {
            var p1 = w.GetProfile(p.handler);
            Init(p1, p);
        }
        public DTW(Profile p1, Profile p2)
        {
            Init(p1, p2);
        }
        private void Init(Profile p1, Profile p2)
        {
            int c1 = p1.handler.settings.coefficientNumber;
            int c2 = p2.handler.settings.coefficientNumber;
            if (c1 != c2) throw new Exception("Количество коэффициентов профилей должно совпадать!");
            matrix = GetMatrix(p1, p2);
            path = GetPath(matrix);
            distance = 0;
            for (int i = 0; i < path.Count; i++) { distance += path[i].value; }
        }
        public List<int> ComparePathByExample(int row)
        {
            List<int> answer = new List<int>();
            for (int i = 0; i < path.Count; i++)
            {
                if (path[i].row != row) continue;
                answer.Add(path[i].column);
            }
            return answer;
        }
        public List<int> ComparePathByReference(int column)
        {
            List<int> answer = new List<int>();
            for (int i = 0; i < path.Count; i++)
            {
                if (path[i].column != column) continue;
                answer.Add(path[i].row);
            }
            return answer;
        }
```

```
private static Matrix<double> GetMatrix(Profile v1, Profile v2)
{
   int R = v1.count, T = v2.count;
   int w = Abs(R - T); //int w = Max(w, Abs(R - T));
   Matrix<double> D = CreateMatrix.Dense<double>(R, T);
   for (int r = 0; r < R; r++)
   {
      for (int t = 0; t < T; t++)
      {
         D[r, t] = (v1[r].mfcc - v2[t].mfcc).L2Norm();
      }
   }
   Matrix<double> DTW = CreateMatrix.Dense<double>(R, T);
   DTW[0, 0] = D[0, 0];
   for (int r = 1; r < R; r++) DTW[r, 0] = D[r, 0] + DTW[r - 1, 0];
   for (int t = 1; t < T; t++) DTW[0, t] = D[0, t] + DTW[0, t - 1];
   for (int r = 1; r < R; r++)
   {
      for (int t = 1; t < T; t++)
      {
         Vector<double> vs = CreateVector.Dense<double>(3);
         vs[0] = DTW[r - 1, t];
         vs[1] = DTW[r, t - 1];
         vs[2] = DTW[r - 1, t - 1];
         DTW[r, t] = D[r, t] + vs.Minimum();
      }
   }
   return DTW;
}
private static List<Cell> GetPath(Matrix<double> DTW)
{
   int cols = DTW.ColumnCount;
   int rows = DTW.RowCount;
   List<Cell> path = new List<Cell>();
   int row = rows - 1;
   int col = cols - 1;
   void AddCell()
   {
      Cell cell = new Cell();
      cell.row = row;
      cell.column = col;
      cell.value = DTW[row, col];
      path.Add(cell);
```

```
        }
        AddCell();
        while ((row > 0) && (col > 0))
        {
            Vector<double> vs = CreateVector.Dense<double>(3);
            vs[0] = DTW[row - 1, col];
            vs[1] = DTW[row, col - 1];
            vs[2] = DTW[row - 1, col - 1];
            if (vs[2] <= vs[0])
            {
                if (vs[2] <= vs[1]) { row--; col--; }
                else { col--; }
            }
            else
            {
                if (vs[0] <= vs[1]) { row--; }
                else { col--; }
            }
            AddCell();
        }
        if (row != 0)
        {
            while (row != 0)
            {
                row--;
                AddCell();
            }
        }
        if (col != 0)
        {
            while (col != 0)
            {
                col--;
                AddCell();
            }
        }
        return path;
    }
}
 }
```

Класс VAD.

```
using System;
using System.Collections.Generic;
using System.Linq;
using static System.Math;
using MathNet.Numerics.LinearAlgebra;
using System.Drawing;
using System.Windows.Forms;
namespace LingREC
{
   public class VADData
   {
      public Matrix<double> matrix { get; protected set; } = null;
      public Matrix<double> criterial { get; protected set; } = null;
      public Vector<double> silence { get; protected set; } = null;
      public Matrix<double> ratings { get; protected set; } = null;
      public FrameIndexation indexation { get; protected set; } = null;
      private List<bool> m_boolMap = null;
      public List<bool> boolMap
      {
         get { return new List<bool>(m_boolMap); }
         protected set { m_boolMap = value ?? throw new ArgumentException("Cannot
be null"); }
      }
      public Frame frame { get { return indexation.frame; } }
      public double tps { get; protected set; } = 0;
      public VADData(Signal signal, SignalChannel channel, Frame frame)
      {
         var samples = CreateVector.DenseOfArray(signal.GetChannel(channel, true));
         indexation = new FrameIndexation(samples, frame, signal.samplingRate);
         var VADAnswer = VAD.GetMatrix(indexation);
         matrix  = VADAnswer.matrix;
         silence = VADAnswer.silence;
         ratings = VADAnswer.ratings;
         criterial = CreateMatrix.Dense<double>(matrix.RowCount, ma-
trix.ColumnCount);
         for (int r = 0; r < criterial.RowCount; r++)
         {
            Vector<double> criteriaRow = VAD.GetCriteria(matrix.Row(r), si-
lence).criteria;
            criterial.SetRow(r, criteriaRow);
         }
```

```
            boolMap = VAD.RawMarkup(matrix);
            tps = signal.timePerSampleChannel;
        }
    }
    public static class VAD
    {
        public const int cR    = 0; public const string nameR    = "R (рейтинг кадра)";
        public const int cEn   = 1; public const string nameEn   = "En (мгновенная энергия)";
        public const int cLSFM = 2; public const string nameLSFM = "LSFM (лог. мера спектральной плотности)";
        public const int cMDFC = 3; public const string nameMDFC = "MDFC (доминирующая частота)";
        public const int cZn   = 4; public const string nameZn   = "Zn (переходы через нуль)";
        public const int variablesTresh = 3;
        public const int maxVariables = 4;
        public static int maxVariablesAndRating { get { return maxVariables + 1; } }
        public static void FillCBXList(CheckedListBox list)
        {
            list.Items.Clear();
            list.Items.Add(nameR,    false);
            list.Items.Add(nameEn,   true);
            list.Items.Add(nameLSFM,  true);
            list.Items.Add(nameMDFC, false);
            list.Items.Add(nameZn,   false);
        }
        public static event Settings.OnVoidChange onVADChange;
        public static double treshZnLow { get; } = 30;
        private static double m_treshZn = 300;
        public static double treshZn
        {
            get { return m_treshZn; }
            set
            {
                if (value < treshZnLow) value = treshZnLow;
                if (value == m_treshZn) return;
                m_treshZn = value;
                onVADChange?.Invoke();
            }
        }
        public static double treshIndentEnLow { get; } = 0;
        private static double m_treshIndentEn = 0.004;
```

```
public static double treshIndentEn
{
    get { return m_treshIndentEn; }
    set
    {
        if (value < treshIndentEnLow) value = treshIndentEnLow;
        if (value == m_treshIndentEn) return;
        m_treshIndentEn = value;
        onVADChange?.Invoke();
    }
}
public static double treshIndentLSFMHigh { get; } = 0;
private static double m_treshIndentLSFM = -1.5;
public static double treshIndentLSFM
{
    get { return m_treshIndentLSFM; }
    set
    {
        if (value > treshIndentLSFMHigh) value = treshIndentLSFMHigh;
        if (value == m_treshIndentLSFM) return;
        m_treshIndentLSFM = value;
        onVADChange?.Invoke();
    }
}
public static double treshIndentMDFCLow { get; } = 0;
private static double m_treshIndentMDFC = 50; // Гц
public static double treshIndentMDFC
{
    get { return m_treshIndentMDFC; }
    set
    {
        if (value < treshIndentMDFCLow) value = treshIndentMDFCLow;
        if (value == m_treshIndentMDFC) return;
        m_treshIndentMDFC = value;
        onVADChange?.Invoke();
    }
}
public static int silenceFrameOffsetLow { get; } = 3;
private static int m_silenceFrameOffset = 5;
public static int silenceOffset
{
    get { return m_silenceFrameOffset; }
    set
```

```
        {
            if (value < silenceFrameOffsetLow) value = silenceFrameOffsetLow;
            if (value == m_silenceFrameOffset) return;
            m_silenceFrameOffset = value;
            onVADChange?.Invoke();
        }
    }
    public static int pauseUpdateAfterLow { get; } = 1;
    private static int m_pauseUpdateAfter = 5;
    public static int pauseUpdateAfter
    {
        get { return m_pauseUpdateAfter; }
        set
        {
            if (value < pauseUpdateAfterLow) value = pauseUpdateAfterLow;
            if (value == m_pauseUpdateAfter) return;
            m_pauseUpdateAfter = value;
            onVADChange?.Invoke();
        }
    }
    public static event Settings.OnVoidChange onVADFrameChange;
    public static bool useAutomaticFrame { get; private set; } = true;
    public static double automaticSecondsPerFrame { get; private set; } = 0.02;
    private static Frame m_frame = new Frame(1024, 0, WindowFunc-
tion.Rectangular, 0.3);
    public static Frame frame
    {
        get { return m_frame; }
        set
        {
            if (m_frame != null) if (m_frame.SameFrame(value)) return;
            if (value == null) throw new ArgumentException("Cannot be null!");
            if (useAutomaticFrame)
            {
                value = new Frame(m_frame.width, m_frame.overlapping, value.function,
value.parameter);
            }
            if (m_frame.SameFrame(value)) return;
            m_frame = value;
            onVADFrameChange?.Invoke();
        }
    }
    public static void SetAutomaticVAD(Signal signal)
```

```
    {
        useAutomaticFrame = true;
        double expectedSamples = automaticSecondsPerFrame / sig-
nal.timePerSampleChannel;
        int bin = expectedSamples.FindNearest2Bin();
        Frame newframe = new Frame(bin, bin / 2, frame.function, frame.parameter);
        if (!m_frame.SameFrame(newframe))
        {
            m_frame = newframe;
            onVADFrameChange?.Invoke();
        }
    }
    public static void SetManualVAD()
    {
        useAutomaticFrame = false;
    }
    public static VADData GetData(Signal signal, SignalChannel channel)
    {
        if (useAutomaticFrame) SetAutomaticVAD(signal);
        return new VADData(signal, channel, frame);
    }
    public static (Vector<double> criteria, Vector<double> intRatings)
            GetCriteria(Vector<double> data, Vector<double> silence)
    {
        if (data.Count != silence.Count) throw new ArgumentException("Не могут
быть разной размерности");
        double R  = 0;
        double En  = data[cEn] - silence[cEn];
        double LSFM = data[cLSFM] - silence[cLSFM];
        double MDFC = data[cMDFC] - silence[cMDFC];
        double Zn  = data[cZn];
        Vector<double> criteria = CreateVector.Dense<double>(data.Count);
        criteria[cEn]  = En;
        criteria[cLSFM] = LSFM;
        criteria[cMDFC] = MDFC;
        criteria[cZn]  = Zn;
        Vector<double> ratings = CreateVector.Dense<double>(data.Count);
        if (criteria[cEn]  > treshIndentEn)  { R += 1; ratings[cEn]  = 1; }
        if (criteria[cLSFM] < treshIndentLSFM) { R += 1; ratings[cLSFM] = 1; }
        if (criteria[cMDFC] > treshIndentMDFC) { R += 1; ratings[cMDFC] = 1; }
        if (criteria[cZn]  < treshZn)       { R += 1; ratings[cZn]  = 1; }
        ratings[cR] = criteria[cR] = R;
        return (criteria, ratings);
```

```
    }
    public static Vector<double> GetFrameVector(ProfileFrame pf)
    {
        Vector<double> input = CreateVector.Dense<double> (maxVariablesAndRat-
ing);
        input[cR]    = 0;
        input[cEn]   = pf.En;
        input[cLSFM] = pf.LSFM;
        input[cMDFC] = pf.MDFC;
        input[cZn]   = pf.Zn;
        return input;
    }
    /// <param name="silence">Средние по тишине</param>
    /// <param name="pf">Профиль кадра</param>
    /// <returns></returns>
    public static Vector<double> GetIntRating(Vector<double> silence, ProfileFrame
pf)
    {
        Vector<double> input = GetFrameVector(pf);
        var answer = GetCriteria(input, silence);
        return answer.intRatings;
    }
    public static (Matrix<double> matrix,
                Vector<double> silence,
                Matrix<double> ratings)
                GetMatrix(FrameIndexation indexation)
    {
        var samples = indexation.samples;
        var frame   = indexation.frame;
        var count   = indexation.count;
        Matrix<double> answer = CreateMatrix.Dense<double>(count, maxVaria-
blesAndRating);
        Matrix<double> ratingMatrix = CreateMatrix.Dense<double>(count, maxVaria-
blesAndRating);
        List<Vector<double>> silenceVector = new List<Vector<double>>();
        Matrix<double> silenceMatrix    = CreateMatrix.Dense<double>(1, maxVaria-
blesAndRating);
        Vector<double>                 silenceAverages                 =
CreateVec-
tor.Dense<double>(maxVariablesAndRating);
        int pause_counter = 0;
        for (int i = 0; i < count; i++)
        {
```

```
            var win = Frame.SelectWindowed(samples, frame, indexation[i].startPosition);
            ProfileFrame pf = new ProfileFrame(win, indexation.samplingRate);
            Vector<double> newRow = GetFrameVector(pf);
            answer.SetRow(i, newRow);
            if (i <= silenceOffset)
            {
               silenceVector.Add(answer.Row(i));
               if (i == silenceOffset - 1)
               {
                  silenceMatrix = CreateMatrix.DenseOfRowVectors(silenceVector);
                  double mask = 0;
                  silenceAverages[cR]    = mask; // маска
                  silenceAverages[cEn]   = silenceMatrix.Column(cEn).Average();
                  silenceAverages[cLSFM] = silenceMatrix.Column(cLSFM).Average();
                  silenceAverages[cMDFC] = silenceMatrix.Column(cMDFC).Average();
                  silenceAverages[cZn]   = silenceMatrix.Column(cZn).Average();
                  for (int k = 0; k < silenceMatrix.ColumnCount; k++)
                  {
                     if (silenceAverages[k] == mask) continue;
                     var average = silenceMatrix.Column(k).Average();
                     if ((double.IsNaN(average)) || (double.IsInfinity(average)))
                        silenceAverages[k] = mask;
                  }
               }
            }
            else
            {
               Vector<double> rating = GetIntRating(silenceAverages, pf);
               ratingMatrix.SetRow(i, rating);
               answer[i, cR] = rating[cR];
               bool sound = (answer[i, cR] >= variablesTresh);
               if (sound)
               {
                  pause_counter = 0;
               }
               else
               {
                  pause_counter++;
                  if (pause_counter > pauseUpdateAfter)
                  {
                     silenceMatrix.InsertRow(silenceMatrix.RowCount - 1, answer.Row(i));
```

```
                    silenceAverages[cEn]   = silenceMatrix.Column(cEn).Average();
                    silenceAverages[cLSFM]                =                silenceMa-
trix.Column(cLSFM).Average();
                    silenceAverages[cMDFC]                =                silenceMa-
trix.Column(cMDFC).Average();
                    silenceAverages[cZn]   = silenceMatrix.Column(cZn).Average();
                }
            }
        }
    }

    return (answer, silenceAverages, ratingMatrix);
}
public static List<bool> RawMarkup(Matrix<double> matrix)
{
    int rows = matrix.RowCount;
    List<bool> output = new List<bool>();
    for (int i = 0; i < rows; i++)
    {
        output.Add(matrix[i, 0] >= variablesTresh);
    }
    return output;
}
public static void GridVADAverages(DataGridView grid, VADData data)
{
    List<string> captions = new List<string>();
    Vector<double> silence = data.silence.SubVector(1, data.silence.Count - 1);
    captions.Add("En");
    captions.Add("SFM [дБ]");
    captions.Add("MDFC [Гц]");
    captions.Add("Zn [..]");
    Grid.UploadVector(grid, silence, 0, false, "F3", captions);
}
public static void GridVAD(DataGridView grid, VADData data, bool averageCor-
rection)
{
    Matrix<double> matrix = (averageCorrection) ? data.criterial : data.matrix;
    List<string> captions = new List<string>();
    captions.Add("SIL.RATE");
    captions.Add("En");
    captions.Add("SFM [дБ]");
    captions.Add("MDFC [Гц]");
    captions.Add("Zn [..]");
```

```
Color good = Color.LightGreen;
Color bad  = Color.GhostWhite;
Color aprsilence = Color.LightSteelBlue;
Color breaked = Color.LightSalmon;
int rows = matrix.RowCount; int cols = matrix.ColumnCount;
Color[,] colors = new Color[rows, cols];
for (int r = 0; r < rows; r++)
{
   if (r < silenceOffset)
   {
      for (int c = 0; c < cols; c++) colors[r, c] = aprsilence;
      continue;
   }
   var row = matrix.Row(r);
   bool isNanOrInf = false;
   for (int i = 0; i < row.Count; i++)
   {
      if (double.IsNaN(row[i]) || double.IsInfinity(row[i]))
      {
         isNanOrInf = true;
         for (int c = 0; c < cols; c++) colors[r, c] = breaked;
         break;
      }
   }
   if (isNanOrInf) continue;
   for (int c = 0; c < cols; c++)
   {
      double value = matrix[r, c];
      if (c == cR)
      {
         colors[r, c] = (data.ratings[r, c] >= variablesTresh) ? good : bad;
      }
      else
      {
         colors[r, c] = (data.ratings[r, c] > 0) ? good : bad;
      }
   }
}
Grid.UploadMatrix(grid, matrix, 0, false, "F4", captions);
Grid.ColorMap(grid, colors);
List<string> frameIDXs = new List<string>();
for (int i = 0; i < matrix.RowCount; i++)
{
```

```
            double timeStamp1 = data.indexation[i].startPosition;
            double timeStamp2 = data.indexation[i].startPosition + da-
ta.indexation.frame.width;
            timeStamp1 *= data.tps * 1000;
            timeStamp2 *= data.tps * 1000;
            string idx = $"{i} = {timeStamp1.ToString("F0")}-
{timeStamp2.ToString("F0")} мс.";
            frameIDXs.Add(idx);
         }
         Grid.AddColumn(grid, 0, "Номер кадра", null, frameIDXs);
      }
   }
}
```

Класс MathExtencion.

```
using System;
using System.Numerics;
using MathNet.Numerics.LinearAlgebra;
using MathNet.Numerics.Statistics;
using static System.Math;
namespace LingREC
{
   public static class MathExtension
   {
      public static double GetFramesInMS(this int frames, Frame frame, double tps)
      {
         return frames * frame.width * tps;
      }
      public static Vector<double> NormalizeByOne(this Vector<double> vs)
      {
         return vs / vs.AbsoluteMaximum();
      }
      public static double CrossCorrelation(Vector<double> one, Vector<double> two)
      {
         return Correlation.Pearson(one, two);
      }
      public static int FindNearest2Bin(this double value)
      {
         if (value < 2) value = 2;
         int scale = 1;
         double pBin = 2, nBin = 2;
         while (true) // считаем вверх
```

```
            {
                scale++;
                nBin = Pow(2, scale);
                if ((value >= pBin) && (value <= nBin)) break;
                pBin = nBin;
            }
            double a = Abs(value - pBin);
            double b = Abs(value - nBin);
            value = (a <= b) ? pBin : nBin;
            return (int)value;
        }
        public static T Clamp<T>(this T val, T min, T max) where T : IComparable<T>
        {
            if (val.CompareTo(min) < 0)      return min;
            else if (val.CompareTo(max) > 0) return max;
            else return val;
        }
        public static T ClampMin<T>(this T val, T min) where T : IComparable<T>
        {
            if (val.CompareTo(min) < 0) return min; else return val;
        }
        public static T ClampMax<T>(this T val, T max) where T : IComparable<T>
        {
            if (val.CompareTo(max) > 0) return max; else return val;
        }
        public static double Heaviside(this double x)
        {
            if (x < 0) return 0; else return 1;
        }
        public static double ToDecibel(this double amplitude, double reference = 1)
        {
            return (10 * Math.Log10(amplitude / reference));
        }
        public static double FrequencyToMel(this double frequency)
        {
            return 1127 * Math.Log(1 + frequency / 700);
        }
        public static double MelToFrequency(this double mel)
        {
            return 700 * (Math.Exp(mel / 1127) - 1);
        }
    }
}
```

Класс Averages.

```
using MathNet.Numerics.LinearAlgebra;
using MathNet.Numerics.Statistics;
using System;
using System.Numerics;
using System.Collections.Generic;
namespace LingREC
{
   public class VectorAverage
   {
      public Vector<double> average  { get; protected set; }
      public Vector<double> sd { get; protected set; }
      public Vector<double> variance { get; protected set; }
      public VectorAverage(Matrix<double> samples)
      {
         int colCount = samples.ColumnCount;
         for (int col = 0; col < colCount; col++)
         {
            var column = samples.Column(col);
            average[col]  = column.Mean();
            sd[col]       = column.StandardDeviation();
            variance[col] = column.Variance();
         }
      }
   }
   public class ValueAverage
   {
      public double average  { get; protected set; }
      public double sd       { get; protected set; }
      public double variance { get; protected set; }
      public ValueAverage(Vector<double> samples)
      {
         average  = samples.Mean();
         sd       = samples.StandardDeviation();
         variance = samples.Variance();
      }
   }
}
```

Класс CorrelationCurve.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace LingREC
{
   public class CorrelationCurve
   {
      public static double error = 0.78;
      public List<double> curve { get; private set; } = null;
      public double total { get; private set; } = 0;
      public DTW dtw { get; private set; } = null;
      public Profile pExample   { get; private set; }
      public Profile pReference { get; private set; }
      public CorrelationCurve(Profile profile, Word word)
      {
         MFCCHandler handler = new MFCCHandler(word.samplingRate,
                                    profile.handler.frame,
                                    profile.handler.settings);
         pExample   = word.GetProfile(handler);
         pReference = profile;
         dtw = new DTW(pExample, pReference);
         Init();
      }
      private void Init()
      {
         int count = dtw.countExample;
         curve = new List<double>();
         for (int i = 0; i < count; i++)
         {
            var p1 = pExample[i].mfcc;

            List<int> pairs = dtw.ComparePathByExample(i);
            List<double> subcurve = new List<double>();
            for (int j = 0; j < pairs.Count; j++)
            {
               int pair = pairs[j];
               var p2 = pReference[pair].mfcc;
               double correlation = MathExtension.CrossCorrelation(p1, p2);
               subcurve.Add(correlation);
```

```
            }
            if (subcurve.Count == 1) curve.Add(subcurve[0]);
            else
            {
               curve.Add(subcurve.Average());
            }
         }
         total = curve.Average();
      }
   }
}
```

Класс SettingsClassicFrame.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LingREC
{
   public static class SettingsClassicFrame
   {
      public enum CopyVADMode { None, OnlyWindow, WindowAndOverlapping, All
}
      public enum CopyVADDivider { Equal, Half, Quarta, Octave }

      static SettingsClassicFrame()
      {
         copyVAD = CopyVADMode.None;
         SyncMitVAD();
      }
      public static Settings.OnVoidChange onSettingsChange;
      private static CopyVADMode m_copyVAD = CopyVADMode.None;
      public static CopyVADMode copyVAD
      {
         get { return m_copyVAD; }
         set
         {
            if (m_copyVAD == value) return;
            if (value == CopyVADMode.None)
            {
```

```
            onSettingsChange?.Invoke();
        }
        else
        {
            m_copyVAD = value;
            SyncMitVAD(); // автоматом вызовет onSettingsChange.Invoke
        }
    }
}
private static CopyVADDivider m_copyVADDivider = CopyVADDivider.Half;
public static CopyVADDivider copyVADDivider
{
    get { return m_copyVADDivider; }
    set
    {
        if (m_copyVADDivider == value) return;
        m_copyVADDivider = value;
        if (copyVAD == CopyVADMode.None)
        {
            onSettingsChange?.Invoke();
        }
        else
        {
            SyncMitVAD();
        }
    }
}
public static Frame CopyVAD(Frame input,
CopyVADMode mode = CopyVADMode.WindowAndOverlapping,
CopyVADDivider divider = CopyVADDivider.Equal)
{
    Frame expected;
    int denominator;
    switch (divider)
    {
        default: denominator = 1; break;
        case CopyVADDivider.Equal: denominator = 1; break;
        case CopyVADDivider.Half: denominator = 2; break;
        case CopyVADDivider.Quarta: denominator = 4; break;
        case CopyVADDivider.Octave: denominator = 8; break;
    }
    switch (mode)
    {
```

```
            default: return input;
            case CopyVADMode.All:
               expected = new Frame(VAD.frame.width / denominator,
                           VAD.frame.overlapping / denominator,
                           VAD.frame.function,
                           VAD.frame.parameter);
               break;
            case CopyVADMode.WindowAndOverlapping:
               expected = new Frame(VAD.frame.width / denominator,
                           VAD.frame.overlapping / denominator,
                           input.function,
                           input.parameter);
               break;
            case CopyVADMode.OnlyWindow:
               expected = new Frame(VAD.frame.width / denominator,
                           input.overlapping,
                           input.function,
                           input.parameter);
               break;
         }
         return expected;
      }
      private static void SyncMitVAD()
      {
         VAD.onVADFrameChange -= SyncMitVAD;
         Frame expected = CopyVAD(m_frame, copyVAD, copyVADDivider);
         if (!m_frame.SameFrame(expected))
         {
            m_frame = expected;
            onSettingsChange?.Invoke();
         }
         VAD.onVADFrameChange += SyncMitVAD;
      }
      private static Frame m_frame = new Frame(1024, 0, WindowFunction.Hamming,
0.3);
      public static Frame frame
      {
         get { return m_frame; }
         set
         {
            value = CopyVAD(value, copyVAD, copyVADDivider);
            if (m_frame.SameFrame(value)) return;
            m_frame = value;
```

```
            onSettingsChange?.Invoke();
        }
    }
    private static int m_frameCount = 5;
    public static int frameCount
    {
        get { return m_frameCount; }
        set
        {
            if (m_frameCount == value) return;
            if (value < 1) value = 1;
            m_frameCount = value;
            onSettingsChange?.Invoke();
        }
    }
  }
}
```

Класс LinearMath.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MathNet.Numerics.LinearAlgebra;
namespace LingREC
{
  public static class LinearMath
  {
    public static Vector<double> MatrixAsVector(Matrix<double> segment)
    {
      int size = segment.ColumnCount * segment.RowCount;
      Vector<double> output = CreateVector.Dense<double>(size);
      int position = 0;
      for (int frame = 0; frame < segment.ColumnCount; frame++)
      {
        for (int coef = 0; coef < segment.RowCount; coef++)
        {
          output[position] = segment[coef, frame];
          position++;
        }
      }
```

```
            return output;
        }
        public static Vector<double> ConcatVectors(List<Vector<double>> list)
        {
            List<double> copies = new List<double>();
            for (int vector = 0; vector < list.Count; vector++)
            {
                for (int idx = 0; idx < list[vector].Count; idx++)
                {
                    copies.Add(list[vector][idx]);
                }
            }
            Vector<double> output = CreateVector.DenseOfEnumerable(copies);
            return output;
        }
        public static Matrix<double> ColumnVectorsAsMatrix(List<Vector<double>>
list)
        {
            return CreateMatrix.DenseOfColumnVectors(list);
        }
    }
}
```