МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Информационных технологий и систем безопасности

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(дипломная работа)

На тему Разработка алгоритмического обеспечения идентификации личности по голосу для санкционирования доступа к информации

Исполнитель Иванова Арина Юрьевна
(фамилия,имя,отчество)

Руководитель кандидат технических наук
(ученая степень, ученое звание)

Чернецова Елена Анатольевна
(фамилия, имя, отчество)

«К защите допускаю»

Заведующий кафедрой _____
(подпись)

профессор, доктор технических наук
(ученаястепень,ученое звание)

Бурлов Вячеслав Георгиевич
(фамилия,имя,отчество)

«17» февраля 2017 г.

Санкт–Петербург

2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Информационных технологий и систем безопасности

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(дипломная работа)

На тему Разработка алгоритмического обеспечения идентификации личности по голосу для санкционирования доступа к информации

Исполнитель Иванова Арина Юрьевна
(фамилия,имя,отчество)

Руководитель кандидат технических наук
(ученая степень, ученое звание)

Чернецова Елена Анатольевна
(фамилия, имя, отчество)

«К защите допускаю»

Заведующий кафедрой_____

(подпись)

профессор, доктор технических наук
(ученаястепень,ученое звание)

Бурлов Вячеслав Георгиевич
(фамилия,имя,отчество)

«__»_____20__ г.

Санкт–Петербург

2017

РЕФЕРАТ

Отчет 73 с., 4 г., 34 рис., 30 источников, 2 прил.

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ, ИДЕНТИФИКАЦИЯ ЛИЧНОСТИ, ГОЛОСОВАЯ БИОМЕТРИЯ, ОБРАБОТКА СИГНАЛА, ГАУССОВЫ СМЕСИ.

Объектом исследования является модель гауссовых смесей, которая применяется для идентификации личности по голосу.

Цель работы — разработка программного средства идентификации личности по голосу.

В процессе работы реализовывался алгоритм, решающий задачу текстонезависимой идентификации.

Задачи, которые необходимо решить в работе для достижения цели :

— Исследование существующих методов решения задачи идентификации личности по голосу, способы оценки их качества, а также существующие проблемы и ограничения;

— Реализация одного из лучших алгоритмов идентификации личности по голосу;

— Разработка обучающей и тестовой выборок, которые позволяют наиболее полно оценить все преимущества и недостатки реализованного алгоритма.

В работе был проведен полный обзор предметной области и реализован алгоритм, основанный на применении модели гауссовых смесей.

Было установлено, что использование алгоритма K-means++ приводит к увеличению скорости обучения модели гауссовых смесей. В результате тестирования было определено оптимальное число компонент модели гауссовых смесей, при котором система идентификации показывает максимальную точность распознавания.

Оглавление

Введение

Речь – неотъемлемый элемент человеческой деятельности, который позволяет человеку познавать окружающий мир, передавать свои знания и опыт другим людям. Устная речь – это высказывания в звуковой форме, которые становятся возможными благодаря голосовому аппарату человека.

Любой человек имеет свои особенные голосовые характеристики, определяющиеся индивидуальным строением его голосовых органов. В процессе общения человек способен на подсознательном уровне различать голоса любых людей, но тем не менее, эта задача для вычислительной техники является нетривиальной.

Известно, что задача распознавания личности по голосу была поставлена более 40 лет назад, но до сих пор продолжаются исследования в данной области. За последние годы качество распознавания речевой информации значительно возросло, однако главная проблема автоматического распознавания диктора в любой среде до сих пор далека от идеального решения. Поэтому актуальны не только исследования уже существующих алгоритмов, но и поиск новых решений в данной области.

Задачей распознавания личности по голосу является выделение, классифицирование и реагирование на речь человека из входного звукового потока. При этом обычно выделяется две подзадачи: идентификация и верификация.

Кроме этого, система распознавания может быть разделена на текстозависимую и текстонезависимую. При текстозависимом распознавании используются как фиксированные фразы, так и фразы, которые были сгенерированы системой и были предложены пользователю. Текстонезависимые системы нужны для обработки произвольной речи.

В данной работе рассматривается задача автоматической идентификации диктора и реализуется алгоритм, который решает задачу текстонезависимой идентификации.

Распознавание личности по голосу может найти применение во многих сферах: криминалистика и судебная экспертиза, безопасность, банковские технологии, электронная коммерция.

Речь представляет собой сложный сигнал, который образуется в результате преобразований, происходящих на разных уровнях: семантическом, языковом, артикуляционном (уровень голосового аппарата человека) и акустическом (уровень физических свойств звука). Различия в этих преобразованиях приводят к различиям в свойствах речевого сигнала. При решении задачи распознавания диктора по голосу данные отличия могут использоваться с целью выделения индивидуальных характеристик голоса человека.

Целью данной работы является разработка программного средства идентификации личности по голосу. Для достижения указанной цели в работе решаются следующие задачи:

— Исследуются существующие методы решения задачи идентификации личности по голосу, способы оценки их качества, а также существующие проблемы и ограничения;

— Реализовывается один из лучших алгоритмов идентификации личности по голосу;

— Разрабатывается обучающая и тестовая выборки, позволяющие наиболее полно оценить все преимущества и недостатки реализованного алгоритма.

Глава 1 Обзор существующих моделей авто - распознавания голоса

1.1 Структура систем распознавания дикторов

Речевой сигнал является средством передачи информации, он используется для создания естественных интерфейсов, которые помогают связаться с машиной, что значительно упрощает решение многих задач. Главной задачей систем обработки речевого сигнала является распознавание произнесенной фразы.

Первой процедурой при обработки речевого сигнала является акустическая обработка. Ее смысл заключается в сопоставлении некого набора признаков каждому фрагменту сигнала [1]. Эти признаки содержат закодированную фонетическую информацию, которая находится в данном отрывке сигнала. Большинство современных систем обработки данных речи построены по модульному принципу. Поэтому информация, которая получается с помощью процедуры акустического кодирования, в том или ином виде используется всеми остальными процедурами. Ошибки, которые были допущены на первом этапе – этапе обработки, могут привести к снижению эффективности работы целой системы. Результатом работы методов акустического кодирования является качество работы всей системы целиком, поэтому следует с особым вниманием заниматься разработкой методов, которые весьма чувствительны к любым изменениям в фонетической структуре, но которые в то же время устойчивы к шумовым искажениям сигнала. Устойчивостью к шумовым искажениям можно назвать изменение уровня помех, которые не приведут к существенному изменению вычисляемых признаков [2]. В случае, если в структуре помехи будет наблюдаться некая закономерность, возможно, довольно нетрудно подобрать метод компенсации этой помехи в сигнале и отфильтровать ее, находясь еще на этапе предобработки сигнала. В случае отсутствия данных о помехе и затруднительного подбора компенсирующего алгоритма, фильтрация сведется к отбрасыванию всех данных, за исключением соответствующих полезному сигналу. Также задачей акустической обработки является фильтрация помех.

Необходимость представления фонетической информации в речевом сигнале его вокализованной частью является известным фактом. Таким образом, появляется еще одна важная задача алгоритмов акустической обработки- задача кодирования фонетической информации для вокализованных фрагментов речевого сигнала, которое будет устойчиво к наличию в нем вариаций произношения и различных помех.

Существующие методы обработки вокализованных сигналов подразумевают, что нужно использовать линейные параметрические модели, которые позволяют достаточно эффективно моделировать гармоническую структуру сигнала [3]. В этом случае акустическими признаками рассматриваются параметры модели. У параметрических моделях возникает задача определения порядка модели. При неверных значениях порядка происходит потеря полезной информации, либо внесение паразитной информации в оценки параметров, а это негативно сказывается и на оценках полученных признаков, и на качестве работы.

Методы определения порядка, которые существуют в настоящее время, к сожалению, дают невысокие значения и обеспечивают не в достаточной степени точность решения, поэтому в большинство систем пользуется усредненными значениями порядков, которые устанавливаются экспериментально.

При работе систем распознавания выделяют два основных этапа [4]:

— Регистрацию пользователей в системе;

— Сам процесс распознавания (попытка идентификации или верификации).

Пользователи предварительно проходят регистрацию в системе, записывают свои голоса. Образцы голосов каждого диктора обрабатываются для того, чтобы извлечь признаки, которые будут использоваться для распознавания. На основе этих полученных признаков строятся модели (шаблоны) пользователей. Модель представляется как некоторая структура,

позволяющая при данных признаках оценивать степень подобия, или сразу же принимать решение.

При верификации пользователь пытается войти в систему, предъявив идентификатор и образец голоса. Признаки, которые извлекались из предоставленного образца начинают сравниваться с соответствующей моделью, которая сохранена в базе, и, возможно, что и с референтной моделью, которая представляет собой фиксированное множество некоторых пользователей, которые будут ближе всего к данному голосу [5]. Результат будет сравнен с заданным порогом, и мы получим решение о допуске - положительное или отрицательное.

Также во время идентификации происходит извлечение признаков из предоставленного образца, которые позже подвергнутся сравнению с моделями всех пользователей зарегистрированных в системе, либо только тех, которые были предварительно отобраны.

Исходя из этого, можно выделить основные этапы (уровни) общей схемы системы распознавания, с помощью которых она реализуется [6]:

— Уровень обработки сигналов. На этом уровне сигнал подвергается обработке, для того, чтобы выделить существенные для задачи распознавания признаки. Речевой сигнал представляется с помощью последовательности векторов признаков.

— Уровень моделей. При регистрации пользователя уровень использует последовательность векторов признаков для построения модели, которая была получена от уровня обработки сигналов. Суть моделирования заключается как в обычном копировании векторов признаков, так и в построении вероятностных моделей, либо каких-нибудь других структур. После этого будет возможно вычисление степени подобия между признаками и сохранённой моделью, при данных признаках.

— Уровень принятия решений. Функции принятия решений обычно выделяют в специальный уровень, хоть он и может выполнить тривиальные

функции или вообще отсутствовать, если на уровне моделей вычисляют конечные решения. Для принятия решений пользуются степенями подобия, которые были вычислены на уровне моделей, и, если нужно, заданные пороги.

Отличием распознавания по голосу от многих биометрических систем является то, что в данном случае предметом распознавания является именно процесс, а не обычное изображение, например, как при распознавании отпечатков пальцев, лица или радужной оболочки глаза. Из-за этого обычно образец голоса представляется в виде последовательности векторов признаков, каждый из которых описывает характеристики небольшого участка речевого сигнала единого вектора признаков, а не в виде последовательности векторов признаков. Последовательность векторов, которая получается после этапа «обработка сигнала», используется для построения модели диктора или для того, чтобы осуществить сравнение с шаблонами, построенными ранее. Для задач верификации и идентификации можно определить способ вычисления степени подобия представленного образца с одним, двумя и более шаблонами. Степень подобия вычисляется и на основе определённой метрики или на основе оценки вероятности [7].

На данный момент существуют несколько способов для классификации моделей при задаче распознавания. Часто в литературе ссылаются на дискриминативные и генеративные модели. Сутью ганеративных моделей, является моделирование данных, которые были получены для обучения, к примеру, при помощи оценки функции плотности вероятности (модель гауссовых смесей). Основа дискриминативных моделей лежит на построении границы между классами, как это реализовывается в методе опорных векторов [8].

Рассмотрим процесс идентификации. Образец речевого сигнала подвергается обработке и представляется с помощью последовательности векторов v_1, ..., v_L. От каждого вектора v_i вычисляются самые короткие расстояния до шаблонов каждого из дикторов. Для этого пользуемся

вычислением евклидового расстояния, которое усреднено по количеству элементов. Обозначаем через d_{ij} - расстояние от вектора i до шаблона j. Обычный подход к классификации на основе векторного квантования (метод ближайшего соседа) будет заключаться в том, чтобы вычислить среднее по векторам расстояние до шаблонов.

Несколько расстояний до необходимого шаблона, которые получены от векторов, соответствующих разным звуковым отрезкам, могут значительно отличаться. Если объединить вектора признаков из разных признаковых пространств, то становится необходимо произвести процедуру нормализации расстояний. Поэтому нужно осуществить процедуру нормализации расстояний. Нормализацию на уровне принятия решений часто сравнивают с нормализацией, которая осуществляется в задаче верификации, где вычислялись отношения степеней подобия представленного образца и заявленной идентичности к степени подобия с неким множеством референтных пользователей, которые называются когортой. В задаче идентификации нет необходимости раздельно хранить когортные модели, для фиксированного расстояния d_{ij} в качестве референтных выступают расстояния d_{ik}, $k \neq j$ [9].

Степень подобия становится выше, чем становится короче расстояние. Каждый из способов вычисления степеней подобия можно рассмотреть как вектор-функцию, при данных значениях расстояний ($d_{i,1}$, ..., $d_{i,N}$), которая вычисляет степени подобия ($s_{i,1}$, ..., $s_{i,N}$). Полученные для исходных векторов степени подобия потом суммируются для принятия итогового решения:

$$C = \operatorname{argmax}_{1 \le j \le N} \sum_{i=1}^{L} s_{i,j} \quad (1)$$

Пусть найдены кратчайшие расстояния ($d_{i,1}$, ..., $d_{i,N}$) от вектора v_i до хранимых шаблонов. Для того, чтобы дальше их использовать, нужно упорядочить расстояния по возрастанию: $(d'_{i,1},..,d'_{j,N}), d'_{i,j} \le d'_{i,j+1}$ [10].

Задаем функцию вычисления степеней подобия:

$$s_{i,j} = \begin{cases} \frac{d_c - d_{i,j}}{d_c - d'_{i,1}} & , d_{i,j} < d_c \\ 0, d_{i,j} \geq d_c \end{cases} \quad (2)$$

Здесь $d_c - d'_{i,1}$ - некое расстояние, которое используется для нормирования.

1.2 Dynamic Time Warping

Dynamic Time Warping (DTW) – алгоритм динамической трансформации временной шкалы, метод динамического программирования, который позволяет найти близость между парой последовательностей измерений за некий промежуток времени. Обычно такие последовательности бывают разной длины, а измерения производятся с разной скоростью.

Существуют условия, которые налагаются на DTW алгоритм для того, чтобы обеспечить быструю конвергенцию, а именно:

— Монотонность. Путь никогда не возвращается, это значит, что индексы i и j, использующиеся в последовательности, не уменьшаются;

— Непрерывность. Продвижение последовательность проходит постепенно, а именно, за один шаг, оба индекса i и j увеличиваются не более чем на единицу;

— Предельность. Начало последовательности- левый нижний угол, а конец находится в правом верхнем углу [11].

В качестве сохраняемой модели в методе DTW выступает последовательность из векторов признаков входного сигнала речи из обучающей выборки $Q=\{q_1,\ldots q_n\}$.

Пусть $C=\{c_1,\ldots,c_m\}$ – последовательность векторов признаков входного речевого сигнала из тестовой выборки. Необходимо ввести понятия матрицы выравнивания двух последовательностей M_{mxn}, позиции (i, j), в которой

находится значение выравнивания между двумя элементами - c_i и q_j последовательностей C и Q соответственно, и набора индексов смежных элементов этой матрицы $W=\{w_1,\ldots,w_T\}$, который определяет, соответствуют ли элементы сопоставляемых последовательностей друг другу. Так же элементы набора W обязаны удовлетворять условиям [12]:

— $w_1=(1,1)$, $w_T=(m,n)$

— Если $w_{t-1}=(a',b')$, то $w_t=(a,b)$, где $a - a' \leq 1$, $b - b' \leq 1$

Цель алгоритма Dynamic Time Warping заключается в том, чтобы найти набор W , который будет удовлетворят условиям 1 и 2, и при котором суммарное искажение последовательности C относительно последовательности Q будет минимальным, а именно:

$$DTW(Q,C)= \min \{ \frac{1}{T}\sqrt{\sum_{t=1}^{T} M(w_t)}\} \quad (3)$$

Значение этого выражения определяет меру близости последовательностей Q и C . Для того, чтобы найти значения DTW (Q,C) нужно применить метод динамического программирования, в нем на каждом шаге вычисляется значение M(i,j):

$$M(i,j)=d(i,j)+\min\{M(i-1, j-1), M(i-1,j), M(i,j-1)\} \quad (4)$$

При этом M(0,0)=0, а все оставшиеся элементы столбца и строки с нулевым индексом инициализируются бесконечным значением. d(i,j) позволяет определить Евклидово расстояние между элементами c_i и q_j. Значение DTW(Q,C)=M(m,n) является результатом работы алгоритма.

Для того, чтобы определить конкретного диктора необходимо вычислить значение $DTW(Q_i,C)$, для каждого сохраненного шаблона. Значение i, при котором достигается минимум, определит номер диктора, образец голоса которого наиболее похож на образец входного речевого сигнала.

Хотя алгоритмом успешно пользуются во многих областях, иногда он выдает неверные результаты. Алгоритм пытается объяснить непостоянство

оси *x* при помощи трансформации оси *y*, что может привести к выравниванию, при котором одной точке начальной последовательности немаленькая подгруппа точек другой последовательности [13].

Еще одна проблема - алгоритму сложно найти видимое выравнивание двух рядов из-за того, что какая-то точка (пик, впадина, плато, точка перегиба) одного ряда будет располагаться чуть выше или чуть ниже соответствующей ей некой точки в другом ряду.

Не смотря на это, алгоритм Dynamic Time Warping все еще открыт для улучшений и хорошо подходит для приложений, которым нужно несложное распознавание слов (телефонам, автомобильным компьютерам, системам безопасности и т.д.) [14]. Главным преимуществом алгоритма является простота реализации. Однако алгоритм DTW невозможно применять в решении задачи текстонезависимой идентификации диктора.

1.3 Hidden Markov Model

Hidden Markov Model (HMM) – скрытая марковская модель. Это статистическая модель, использующаяся в решении задачи классификации скрытых параметров на основе наблюдаемых параметров. HMM является конечным автоматом, где переходы между состояниями производятся с некой вероятностью, и где задано начальное состояние, с которого стартует весь процесс.

Первые заметки о HMM опубликовал Баум в 1960-х, а уже в 70-х они впервые применились для распознавания речи. С середины 1980-х марковские модели применимы при анализе биологических последовательностей, в особенности ДНК [15].

Главное применение CMM получили в области распознавания речи, письма, движений, криптоанализе, машинном переводе, биоинформатике.

В области распознавания о сигналах принято думать как о результате умножения, которое действует статистически. Целью анализа таких сигналов является точнейшее моделирование статических свойств источников сигналов. Основой таких моделей является простые исследования данных и возможной степени ограничения появляющихся отклонений. Но определяющаяся модель обязана не только повторять выработку неких данных с высочайшей точностью, но и доставлять полезную информацию о некоторых важных единицах для сегментации сигналов. ССМ могут обрабатывать оба этих аспекта моделирования. В двухэтапном стохастическом процессе информацию для сегментации можно получить из внутренних состояний модели, тогда как сама генерация сигнала данных происходит на следующем этапе.

Популярность эта технология моделирования получила благодаря успешному применению и дальнейшему развитию в областях авто-распознавания речи. Исследование скрытых моделей Маркова превзошло все известные подходы, и является главной парадигмой обработки. Способность моделей описывать процессы и сигналы с положительным результатом изучается в течение долгого времени. Причиной этому послужило редкое применение для распознавания речи (и других подобных проблем сегментации) технологии построения искусственных нейронных сетей. Не смотря на это, существует ряд гибридных систем, которые состоят из взаимодействия скрытых моделей Маркова и искусственных нейронных сетей, которые пользуются преимуществами обоих методов моделирования [16].

Через дискретные моменты времени возможно осуществление перехода в новые состояния. При этом каждому скрытому состоянию с заданной вероятностью соответствует наблюдаемое состояние. Текущее состояние автомата зависит только от конечного числа предыдущих автоматов, при этом закон смены состояний во времени не меняется.

Формально СММ определяются следующими параметрами [17]:

— Множеством скрытых состояний $Q=\{q_0,\ldots,q_N\}$, где q_0 – начальное состояние, q_{end} – конечное состояние;

— Множеством наблюдений $O=\{o_1,\ldots,o_M\}$;

— Исходным распределением состояний $\pi=\{\pi_i\}$, $1 \le I \le N$, которое определяет вероятность начала работы в состоянии i ;

— Матрицей вероятностей переходов между скрытыми состояниями A_{NxN}: $A(I,j)= a_{ij} = P(q_i,q_j)$, $1 \le I,j \le N$;

— Матрицей вероятностей наблюдений B_{NxM} :$B(I,j) = b_{ij} =P(o_j|q_i)$, $1 \le i \le N$, $1 \le j \le M$

Диаграмма, представленная на рисунке 1.1, показывает общую структуру Hidden Markov Model. Овалы обозначают переменные со случайным значением.

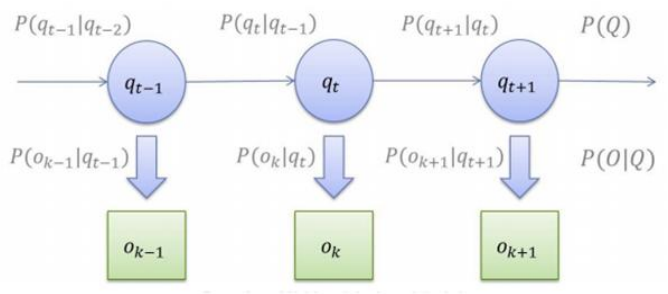

Рисунок 1.1 Общая структура Hidden Markov Model

Выделяется три основных задачи, которые решает HMM [18]:

— Вычисление вероятности последовательности наблюдений. Нужно определить вероятность возникновения заданной последовательности наблюдений $O=\{o_1,\ldots,o_R\}$. Для этого можно воспользоваться следующим алгоритмом:

$\alpha_0(i) = \alpha_{0i}$, $1 \le i \le N$

$\alpha_j(r) = \sum_{i=1}^{N} \alpha_i(r-1)\alpha_{ij}b_{jt}, 1 \le j \le N, 1 \le r \le R,$

$\alpha_j(r)$- вероятность того, что на r-ом шаге модель окажется в состоянии j.

Искомая вероятность определяем с помощью формулы:

$$P(O|A, B)= \sum_{i=1}^{N} \alpha_i(R) \quad (5)$$

— Нахождение самой правдоподобной последовательности скрытых состояний для последовательности, которую наблюдаем. Нужно найти самую правдоподобную последовательность скрытых состояний $Q=\{q_1,\ldots,q_R\}$ для заданной последовательности наблюдений $O=\{o_1,\ldots,o_r\}$, при которой достигается $P(O|Q)$. Данную задачу можно решить при помощи алгоритма, похожего на алгоритм из предыдущего пункта, но с одни отличием - на каждом шаге запоминается состояние i, в котором $\alpha_i(r)$ принимает самое большое значение. В итоге выбирается последовательность состояний, для которой $\alpha_i(R)$ принимает самое большое значение. Алгоритм получил название алгоритм Витерби;

— Обучение параметров модели с помощью заданной последовательности наблюдений и множеством скрытых состояний. Необходимо вычислить матрицы A и B для заданной модели. Чтобы решить эту задачу применяется алгоритм Баума-Велша. Суть его заключается в том, что нам дана одна или несколько выходных последовательностей с дискретными значениями, но они нуждаются в тренировке системы на данном выходе.

Для современного применения данной модели для задач анализа сигнала используются только непрерывные CMM, хоть и необходимость моделирования непрерывного распределения довольно хорошо увеличивает сложность анализа.

Модель Hidden Markov Model достаточно проста в понимании, имеет довольно высокую точность распознавания, но, наряду с DTW, обычно применяется для задач текстозависимой идентификации диктора.

1.4 Vector Quantization

В этом разделе рассматривается одновременное квантование блока символьных отсчётов или блока сигнальных параметров. Данный вид квантования

имеет название блоковое или векторное квантование. Оно широко применяется в кодировании речи в цифровых сотовых системах связи.

Результат теории искажения заключается в том, что лучшую характеристику можно достигнуть векторным, а не скалярным квантованием, даже если у непрерывного источника нет памяти. Кроме того, если отсчёты сигнала или параметры сигнала статистически зависят друг от друга, то можно пользоваться зависимостью путем совместного квантования блоков отсчётов или параметров. Этот способ помогает достигнуть наибольшей эффективности, чем с той, которой можно достигнуть со скалярным квантованием.

Задачей векторного квантования с кодовыми векторами $W= \{w_1,w_2,\ldots,w_n\}$ для последовательности входных векторов $C= \{c_1,c_2,\ldots,c_m\}$ является задача минимизации искажения при замене каждого вектора из Q соответствующим кодовым вектором [19].

Моделью диктора в методе векторного квантования будет являться множество кодовых векторов, которое можно получить из входной последовательности векторов признаков речевого сигнала. Для того, чтобы построить это множество, исходная последовательность векторов признаков должна быть разбита на L кластера, а в качестве кодовых векторов должны взяться их центры.

Процесс определения диктора по входному речевому сигналу протекает следующим образом. Для каждого тестового вектора c_i нужно определить k ближайшие кодовые векторoа. Пусть k_{ij} – количество векторов, которые принадлежат диктору S_j, среди найденных самых ближних. Тогда вероятность того, что вектор c_i принадлежит диктору S_j , можно определить с помощью формулы:

$$P(S_j|c_i) = \frac{k_{ij}}{k} \quad (6)$$

Из этого следует, что последовательность тестовых векторов может быть классифицирована по правилу:

$$S = \text{argmax}_{1 \le j \le N} \prod_{i=1}^{L} P(S_j | c_i) \quad (7)$$

Для сглаживания погрешности измерения, которая связана с близкими к нулю вероятностями, с учетом постоянности числа k, можно использовать правило:

$$S = \text{argmax}_{1 \le j \le N} \sum_{i=1}^{L} P(S_j | c_i) = \text{argmax}_{1 \le j \le N} \sum_{i=1}^{L} k_{ij} \quad (8)$$

Метод векторного квантования прост в реализации, его можно применять к задаче текстонезависимой идентификации диктора, но он не гарантирует высокой точности распознавания.

1.5 Support vector machine

Support vector machine (метод опорных векторов) – бинарный классификатор, который строит в пространстве признаков разделяющую функцию, задающую гиперплоскость, вида:

$$f(x) = w \cdot x + b \quad (9)$$

Основная задача машинного обучения – это классификация данных, именно в данном направлении применимы методы оптимизации и аналитической геометрии.

Представим, что некие объекты принадлежат к одному из двух классов.

Задачей является определение того, к какому классу будут относиться новые наблюдения. В случае метода Supported Vector Machine точку в пространстве рассматривают как вектор размерности *p*. Нужно узнать, возможно ли разделение данных точки гиперплоскостью размерности (*p*-1). Такая плоскость называется плоскость классификатора. Данные можно разделить при помощи разных гиперплоскостей. Самая подходящая гиперплоскость такая, при построении которой будет максимальное разделение и разница между двумя классами.

В качестве примера рассматриваются данные на плоскости, гиперплоскостью в этом случае будет обычная прямая, представленная на рисунке:

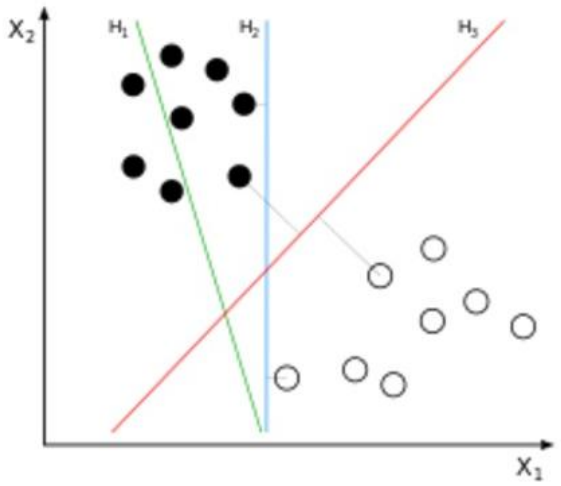

Рисунок 1.2 H1,H2,H3- гиперплоскости, H3- гиперплоскость максимальной разности.

Пусть дана последовательность точек пространства признаков X = $\{x_1,x_2,\ldots,x_n\}$ с метками Y = $\{y_1,y_2,\ldots y_n\}$, $y_i \in \{-1,1\}$, $1 \leq i \leq n$, которые соответствуют двум классам. В случае линейной разделимости данных условия для нахождения функции f(x) будут записаны в виде [20]:

$$\begin{cases} w \cdot x_i + b \geq 1, y_i = 1 \\ w \cdot x_i + b \leq -1, y_i = -1 \end{cases} \Leftrightarrow y_i (w \cdot x_i + b) - 1 \geq 0, 1 \leq i \leq n \qquad (10)$$

Для того, чтобы надежно разделить классы нужно делать как можно большее расстояние между разделяющими гиперплоскостями. Расстояние можно вычислить как $\frac{2}{||w||}$,из этого следует, что задача поиска разделяющей гиперплоскости сводится к минимизации $||w||^2$ при данных условиях. Данную задачу можно решить при помощи метода множителей Лагранжа.

В случае линейно-неразделимых множеств, необходимо ввести функцию ядра. Идея будет заключаться в следующем: отображение исходного пространства в пространство с более высокой размерностью, где множества могут быть линейно разделены. При этом, из-за того, что в данном алгоритме используются раздельные признаки, а не признаки в виде скалярных

произведений, то нет никакой нужды для построения данного преобразования в явном виде. Будет достаточно задания функции ядра, которая определит скалярное произведение в новом пространстве:

$$K(x_i,x_j) = \varphi(x_i) \cdot \varphi(x_j) \qquad (11)$$

После вычисления решающей функции f(x) , принадлежность вектора x´ соответствующему классу будет определяться как f(x´).

Для того, чтобы применить метод опорных векторов в задаче многоклассового распознавания нужно использовать стратегию «один против других». Для этого требуется построить *q*-классификаторов, каждый из которых будет обучаться тому, чтобы отличить один определенный класс от других. При распознавании объект будет приписываться именно к тому классу, чей классификатор выдает самое большое значение разделяющей функции f(x).

Метод опорных векторов может создать гиперплоскость или же набор гиперплоскостей в многомерном или бесконечномерном пространстве, которые можно использовать при решении задач классификации, регрессии и иных задач, подобного рода.

Достаточно хорошего разделения можно достигнуть при помощи гиперплоскости, имеющей наибольшее расстояние до самой близкой точки обучающей выборки любого класса, потому что обычно ошибка классификатора тем меньше, чем больше расстояние.

Во время того, как исходная задача формулируется в конечномерном пространстве, может случиться следующее - наблюдения для дискриминации не будут являться линейно делимыми в данном пространстве.

Можно предложить, что исходное конечномерное пространство отображается в пространство с большей размерностью, а это упрощает разделение.

Для сохранности разумной вычислительной нагрузки, отображения, которые используются в алгоритмах метода опорных векторов, должны

обеспечить простое вычисление точек в переменных в исходном пространстве, в терминах функции ядра K(x,y).

Гиперплоскостью в многомерном пространстве можно назвать набор точек, для которых константой в данном пространстве будет являться скалярное произведение с вектором. Векторы, которые определяют гиперплоскости, можно выбрать как линейную комбинацию с параметрами a_i-изображения функции векторов, которые встретятся в базе данных [21].

При данном выборе гиперплоскости, точки x в пространстве функций отобразятся в гиперплоскость и определятся по формуле:

$$\sum_i^n \propto_i K(x_i x) = const \quad (12)$$

Сумму ядер можно использовать для того, чтобы измерить относительную близость каждой контрольной точки до точек из данных, которые принадлежат одной категории при дискриминации. Множество точек, которые отображаются в любой гиперплоскости, может быть сложным. В результате нужна более сложная дискриминация между наборами, не являющимися выпуклыми в целом в исходном пространстве.

Метод опорных векторов предполагает высокую точность классификации, он обладает теоретическим обоснованием, позволяет применять разные подходы к классификации в соответствии с выбором функции ядра. Среди недостатков стоит обратить внимание на проблему выбора ядра, а также на медленное обучение при задаче многоклассового распознавания.

1.6 Gaussian Mixture Model

В задачах распознавания личности каждый диктор представляется как модель гауссовых смесей, он соответствует своей модели λ . Модель гауссовой смеси имеет различные формы в зависимости от того, какой вид имеет ковариационная матрица. Модель может иметь одну ковариационную матрицу

для каждого компонента модели, одну ковариационную матрицу для всех гауссовых компонент в модели или одну ковариационную матрицу, которая будет использоваться всеми дикторами во всех моделях. Ковариационная матрица бывает полной или диагональной. В экспериментах обычно используют диагональную ковариационную матрицу для каждого отдельного компонента модели [22].

Модель гауссовых смесей представляет собой взвешенную сумму M-компонент и описываться выражением:

$$P(\overline{x}|\lambda)= \sum_{i=1}^{M} p_i\, b_i(\bar{x}), \quad (13)$$

где $\bar{x}$ – D-мерный вектор случайных величин, p_i, $1 \le i \le M$ – веса компонентов модели, $b_i(\bar{x})$, $1 \le i \le M$ – функции плотности распределения составляющих модели:

$$b_i(\bar{x})=\frac{1}{(2\pi)^{\frac{D}{2}}\,|\sum_i|1/2}\, exp\left\{-\frac{1}{2}(\overline{x}-\overline{\mu_l})^T \sum_i^{-1} \bar{x}-\overline{\mu_l}\right\}, \quad (14)$$

где $\overline{\mu_l}$ - вектор математического ожидания и $\sum_i$ – ковариационная матрица. Веса смеси должны удовлетворять условию:

$$\sum_{i=1}^{M} p_i = 1 \quad (15)$$

Полностью модель гауссовой смеси можно определить вектором математического ожидания, ковариационными матрицами и весами смесей каждых компонентов модели:

$$\lambda = \{p_i, \overline{\mu_l}, \sum i\}, i = 1, \dots M \quad (16)$$

В данном методе каждый из дикторов представляется как модель гауссовых смесей λ.

Для того, построить модель диктора нужно оценить её параметры, наилучшим образом соответствующие распределению векторов признаков обучающего высказывания. Шире всех прочих используется метод оценки

максимального правдоподобия. Цель оценки максимального правдоподобия можно определит как нахождение параметров модели, максимизирующих правдоподобие данной модели, при заданных обучающих данных [23].

Для последовательности обучающих векторов $X = \{\overline{x_1}, \dots, \overline{x_T}\}$ правдоподобие модели гауссовых смесей записывается в следующем виде:

$$P(X|\lambda) = \prod_{t=1}^{T} P(\overline{x_t}|\lambda) \qquad (17)$$

Данное выражение представляет собой нелинейную функцию от параметров λ, но вычислить ее невозможно, поэтому для того, чтобы оценить параметры, обычно применяют EM-алгоритм.

Пусть $S=\{S_1,S_2,\dots,S_N\}$ – группа дикторов, представленные набором моделей гауссовых смесей $\lambda_1, \lambda_2, \dots, \lambda_N$.

При идентификации диктора нужно определить модель, имеющую самое большое значение апостериорной вероятности для заданного высказывания:

$$S = argmax_{1 \le k \le N} \sum_{t=1}^{T} \log P(\overline{x_t}|\lambda_k) \qquad (18)$$

Существует две причины для того, чтобы использовать модели гауссовых смесей для идентификации личности по голосу. Первой причиной является интуитивное предположение того, что некоторые компоненты модели могут смоделировать некое множество акустических признаков или событий. Существует предположение, что акустические пространства голосов дикторов могут быть характеризованы множеством классов, которые представляют собой некие фонетические события (звуки) как гласные, фрикативные и т.п. Данные акустические классы отражают с одной стороны общие, но с другой стороны особые конфигурации голосового тракта для каждого диктора, из-за этого они являются эффективными для того, чтобы идентифицировать пользователя.

Спектр акустического класса можно представить как вектор математического ожидания, а изменение среднего спектра можно представить как ковариационную матрицу. Благодаря тому, что звуки речи, которые

используются для обучения и распознавания, не имеют отметок, то фонетические события шифруются в классах акустического пространства. Если сделать предположение, что векторы признаков являются зависимыми друг от друга, то плотность наблюдения векторов, которые образуют эти классы, можно будет считать смесью гауссовых распределений.

Еще одной причиной для использования моделей гауссовых смесей для идентификации диктора является то, что линейная комбинация гауссовых распределений может представить немаленькое число классов акустических признаков. Одной из сильных сторон смеси гауссовых моделей можно считать тот факт, что данные модели могут отлично аппроксимировать произвольные распределения. Основную модель представления диктора при помощи одного гауссова распределения можно описать при помощи позиции распределения (вектор математического ожидания) и формой распределения (ковариационная матрица). Модель векторного квантования будет представлять диктора с помощью дискретного множества кластеров кодовой книги. Модель гауссовых распределений можно представить как гибрид между моделями векторного квантования и гауссова распределения, потому что она пользуется дискретным множеством гауссовых функций. Каждая из функций обладает своей величиной вектора математических ожиданий и ковариационной матрицы [24].

Модели гауссовых смесей являются эффективным алгоритмом с высочайшей точностью распознавания. Но наряду с этим может возникнуть ряд проблем, которые связаны с выбором числа компонентов модели и с инициализацией её начальных параметров.

Так как гауссовы смеси вместе моделируют функцию плотности вероятности, то не существует необходимости для использования полных ковариационных матриц, даже если параметры вектора не зависят друг от друга полностью. Линейная комбинация диагональных ковариационных матриц может смоделировать корреляцию между элементами вектора наблюдений. Эффекта использования множества M ковариационных матриц можно

достигнуть с помощью увеличения числа гауссовых компонент, которые используют диагональные ковариационные матрицы.

1.7 Выводы

Осуществление развития систем распознавания дикторов происходит по нескольким направлениям. Методы создания моделей дикторов развивались от обычного усреднения векторов признаков до усложненных генеративных и дискриминативных моделей.

Методы создания моделей, преобладающие в наше время:

— Метод для текстозависимых систем — динамическое искажение времени (Dynamic Time Warping; DTW) и скрытые марковские модели (Hidden Markov Model; HMM);

— Метод для текстонезависимых систем — векторное квантование (Vector Quantization; VQ), модели гауссовых смесей (Gaussian Mixture Model; GMM) и метод опорных векторов (Support Vector Machine; SVM).

Принятие решений происходит благодаря использованию единого метода принятия решений (классификатора) и объединению решающих правил.

Каждый из методов принятия решений обладает как преимуществами, так и недостатками, они по-разному реагируют на разницу в условиях обучения и распознавания, а также на особенность каждого из голосов дикторов. Необходимо использовать решения разных классификаторов таким образом, чтобы минимизировать возможную ошибку распознавания. Главный прием состоит в следующем - нужно учитывать качество каждого из классификаторов и принимать решение как взвешенную сумму решений по данным оценкам. Исходя из доказательств, которые Демпстера-Шэйфера применял в своей теории, при агрегировании необходимо, чтобы участвовали решения как минимум трех классификаторов. В распознавании дикторов каждый из классификаторов должен характеризоваться следующими факторами:

— Матрицей ошибок для каждого диктора;

— Списком дикторов с плохим решением;

— Списком дикторов с правильным решением;

— Списком дикторов для каждого произнесения.

Верный прогноз или верное решение приводят к наименьшим потерям, чем неверное. При традиционном статистическом подходе производится оценка потерей прогноза, если сравнивать с некой идеальной моделью принятия верных решений, основанной обычно на некоторой статистической модели, которая описывает наблюдаемые данные. При этом сначала производится оценка параметров статистической модели на основе наблюдений, а позже производится прогноз, который основан на этой модели при оцененных ранее параметрах.

На настоящее время существует несколько методов, которые позволяют решить задачу текстонезависимой идентификации диктора по голосу, при этом каждый из этих методов имеет свои преимущества и недостатки.

Самым распространенным методом является Gaussian Mixture Model. Этот метод взят за основу программной реализации в данной работе.

Модели гауссовых смесей отлично зарекомендовали себя в качестве случайной модели для построения систем распознавания. Данные модели будут удобны как для моделирования характеристик голоса диктора, так и для канала звукозаписи, окружающей среды. Некоторые компоненты модели могут смоделировать некое множество акустических признаков. Каждая из компонент модели отображает как некоторые общие, так и сугубо индивидуальные для каждого человека особенности голоса.

Исходя из вышесказанного, можно сделать вывод об успешном применении данного подхода для решения задачи текстонезависимой идентификации диктора по голосу.

Глава 2 Компьютерное моделирование модели идентификации личности по голосу

2.1 Предварительная обработка

Этапы работы систем распознавания:

- Регистрация нового пользователя;
- Идентификация зарегистрированного пользователя.

Каждый пользователь записывает свой голосовой образец, это необходимо для регистрации в системе. Распознавание происходит благодаря признакам, полученным из образца. Далее строятся "шаблоны" пользователей, которые основаны на полученных признаках. Шаблон - это структура, которая

устанавливает, при этих признаках, степень подобия. Признаки, полученные из записанного голоса, подвергаются сравнению с другими признаками, которые берутся из базы данных. После этого происходит либо отказ в доступе, либо идентификация[25].

Речевой сигнал, полученный в результате записи, при идентификации личности по голосу является обрабатываемым образцом. При кодировании аналоговый сигнал представляется как последовательность мгновенных измерений значений амплитуд. Для записи и обработки речевого сигнала, используют частоту дискретизации, равной 8 или 16 кГц.

Во избежание некачественного распознавания, следует избегать некоторых факторов, например:

- Плохой акустики в помещении;
- Разного расстояния от произносящего до микрофона;
- Несовпадения канала и др.

Например, при распознавании голоса, передающегося по телефону, нельзя быть уверенным в использовании одинаковых микрофонов при регистрации и идентификации пользователя. Необходимо учитывать влияние сторонних помех.

Для более качественной записи необходимо, чтобы канал представлял собой аналого-цифровой преобразователь, кабель и микрофон.

Предварительная обработка сигнала чрезвычайно важна. Суть ее заключается в следующем - обработка фильтром определенных частот, а так же удаление участков, которые не содержат речевой сигнал.

Для определения крайних точек первого слова на практике, необходимо осуществить предобработку.

Например, допустим, что в интервале 300мс от начала записи у нас есть только помехи и посторонние шумы. Разделяем весь входной сигнал на 256 сегментов. Речь представляем как:

$$S = F_p(p = 0,1), \qquad (19)$$

$$F_p = S_p(t), \qquad (20)$$

где S - последовательность отчетов входного сигнала, t = 0,1...255.

Для первых 10 первых сегментов пользуемся быстрым преобразованием Фурье (БПФ):

$$X_p(i) = \sum_{i=0}^{255} S_p(t) e^{-j\frac{2\pi}{n}ni}, \qquad (21)$$

где i = 0,1,...,255, а p = 0,1,...,9.

Далее подсчет среднего арифметического значения:

$$m_i^* = \frac{\sum_{p=0}^{9} |X_p(i)|}{10}, \qquad (22)$$

где i = 0,1,...,127 (так как наблюдается симметрия).

Среднее квадратичное отклонение считается по формуле:

$$\sigma_i^* = \sqrt{\frac{1}{9}\left(\sum_{p=0}^{9}(X_p(i) - m_i^*)^2\right)}, \qquad (23)$$

Расчет порога шумов:

$$\Pi_i = m_i^* + k(\alpha)\sigma_i^*, \qquad (24)$$

где $\alpha = 0{,}95$, $k(\alpha)$=2,33.

В итоге получаем 128 значений порогов шума. Далее проверяется каждый сегмент. Если в каком-то сегменте есть превышение 15 порогов, то можно с уверенностью сказать, что начало слова находится именно здесь.

Но точность нахождения нужного сегмента, в котором находится начало слова, равняется 23 секундам. Для определения более точного расположения начала слова, необходимо разбиение его на 8 отрезков, в каждом из которых будет находиться по 32 отсчета. Таким образом, каждый отрезок будет

равняться 3 мс. Все количество начальных расчетов шума нужно разделить на 80 блоков, для того, чтобы вычислить модуль средней амплитуды шума:

$$S^*_{N_i} = \frac{1}{32}\left(\sum_{j=0}^{31}\left|S_{(32xi)+j}\right|\right), \quad (25)$$

$$m^*_N = \frac{1}{80}\left(\sum_{i=0}^{79} S^*_{N_i}\right), \quad (26)$$

$$\sigma^*_N = \sqrt{\frac{1}{79}\left(\sum_{i=0}^{79}(S^*_{N_i} - m^*_N)^2\right)}, \quad (27)$$

$$\Pi_N = m^*_N + k(\alpha)\sigma^*_N, \quad (28)$$

где $\alpha = 0{,}95$, $k(\alpha)$=2,33.

Заключительный этап - сравнение среднего значения модуля каждого их блоков S^*_N в сегменте, в котором находится слово с порогом Π_N. Если S^*_N идущих двух подряд блоков выше порога Π_N, то можно сделать вывод, что именно в этом блоке находится начало слова, произнесенного пользователем при регистрации.

Ниже представлены блок-схемы алгоритма определения начала слова (Рисунок 2.1) и алгоритма уточнения данного интервала (Рисунок 2.2).

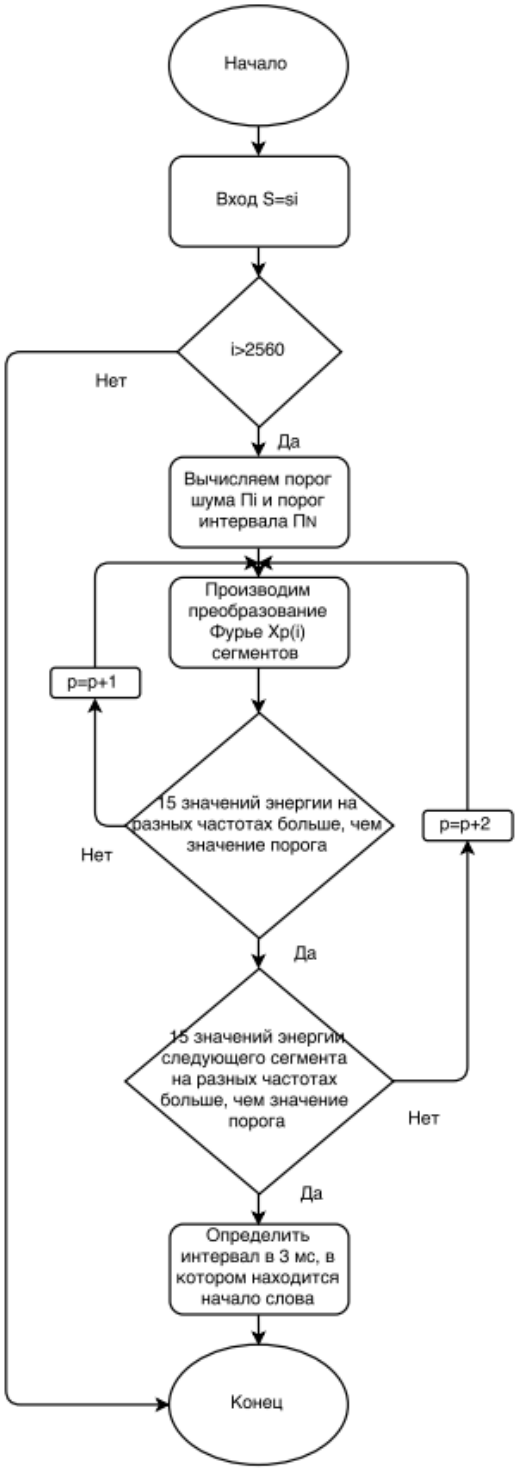

Рисунок 2.1 Блок-схема алгоритма определения начала слова

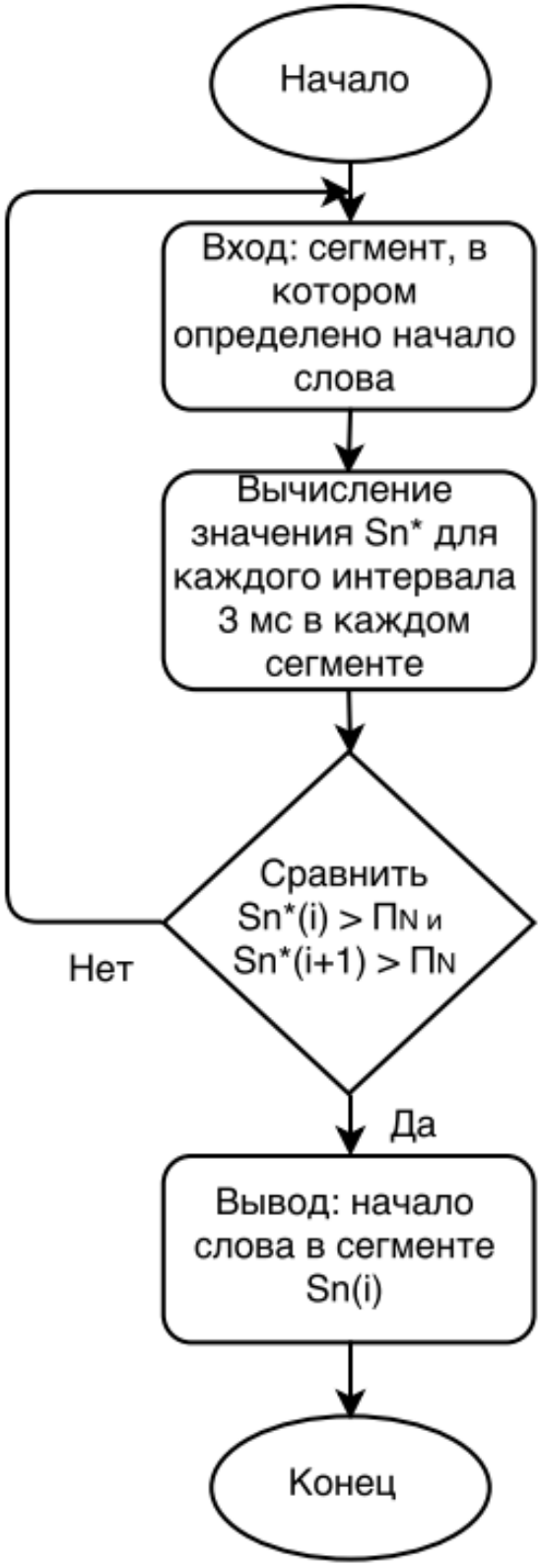

Рисунок 2.2 Блок-схема алгоритма уточнения определения начала слова

2.2 Признаки речевого сигнала

Обработка сигнала нужна для того, чтобы выделить в речевом сигнале релевантную для распознавания личности по голосу информацию. Такая информация представляет собой уникальные для голоса каждого человека признаки [26]. Эти признаки выделяются для того, чтобы сформировать шаблон или для их сравнения с зарегистрированными шаблонами. Определение более подходящих признаков для распознавания невозможно. Для этого требуется пробная оценка с начальным перебором всех вероятных признаков.

Признаки можно разбить на 2 вида:

— Низкоуровневые (анатомическое строение речевого аппарата);
— Высокоуровневые (манера произношения).

Для обработки речевого сигнала, требуется использование кратковременного анализа. Сигнал нуждается в разбиении на временные окна фиксированного размера. Предполагается, что в этих окнах параметры сигнала неизменчивы. При работе с речевым сигналом, размер окна составляет 10-30 мс. Перекрытия, равные половине длины окна, способствуют наибольшей точности. Для извлечения признаков из каждого окна применяются определенные алгоритмы. Рассмотрим основные методы извлечения признаков из речевого сигнала.

Мел-частотные кепстральные коэффициенты. В древнегреческом языке "мэлос"- это звук. На практике мел - это психофизическая единица высоты звука, в основании которой лежит восприятие этого звука человеческими слуховыми анализаторами [27].

Амплитудно-частотные характеристики человеческого органа слуха и близко не похожи на прямую, а амплитуда не является точной мерой измерения громкости (Рисунок 2.3). В связи с этим и были введены эмпирические единицы громкости звука.

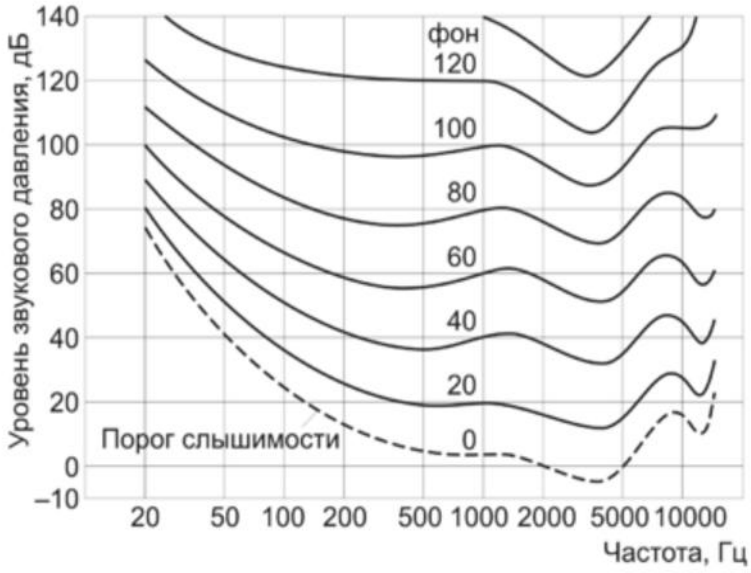

Рисунок 2.3 АЧХ человеческого органа слуха

Высота звука, воспринимаемая человеческими органами слуха, не является линейно зависимой от его частоты:

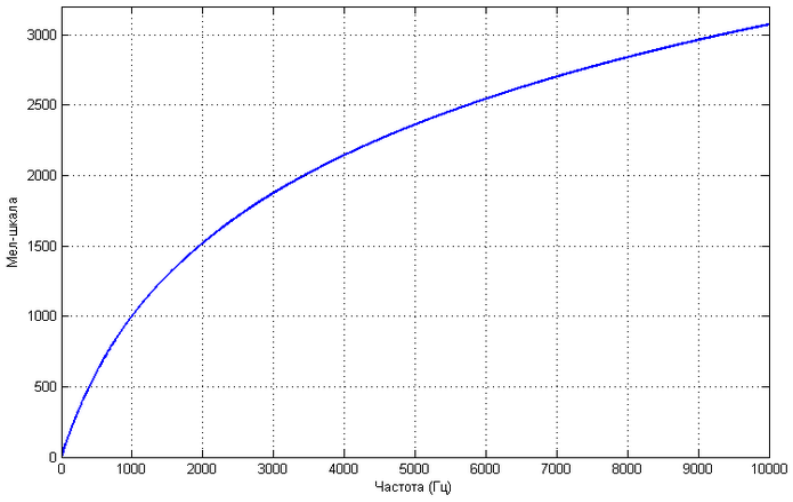

Рисунок 2.4 Зависимость высоты звука от его частоты

Единицы измерения мел частот используются в системах распознавания, так как они отлично помогают исследовать устройство человеческого восприятия.

Кепстр- это эмпирически измеряемая величина, результат взятия преобразования Фурье логарифма спектра сигнала [28].

Виды кепстра:

— Энергетический;

— Комплексный;

— Реальный;

— Фазовый.

Способы определения энергетического кепстра:

— Устно: энергетический кепстр сигнала - это величина Фурье - спектра логарифма квадратичной величины Фурье-спектра сигнала;

— С помощью алгоритма:

$$signal \rightarrow FT \rightarrow abs() \rightarrow square \rightarrow log \rightarrow FT \rightarrow abs() \rightarrow square \rightarrow powercepstrum$$

Комплексный кепстр предложил Оппенгейм, он работал над теорией гомоморфных систем. Алгоритмическое представление комплексного кепстра:

$$signal \rightarrow FT \rightarrow abs( ) \rightarrow log \rightarrow phaseu\ nwrapping \rightarrow FT \rightarrow cepstrum$$

Реальный кепстр (РК) использует логарифм функции, определенной для реальных значений. Этот кепстр имеет взаимосвязь с энергетическим кепстром (ЭК):

$$ЭК = (4 * РК)^2 \quad (29)$$

А так же с комплексным спектром (КК):

$$РК = 0{,}5 * (КК + КК^*), \quad (30)$$

где $КК^*$ - обращенный по времени комплексный кепстр.

В основе комплексного кепстра лежит комплексный логарифм функции, которая определена для комплексных значений.

Взаимосвязь комплексного кепстра и фазового:

$$ФК = (КК - КК^{*})^{2} \quad (31)$$

В реальном кепстре содержится информация об амплитуде спектра, тогда как на раду с этим, комплексный кепстр содержит так же и данные о фазе исходного спектра. Это дает возможность реконструировать сигнал [29].

В целом кепстре рассматривается информация о скорости изменения в различных диапазонах спектра. Изначально его использовали для того, чтобы измерять сейсмические отголоски после сильнейших взрывов и землетрясений.

Этап извлечения признаков представляет собой преобразование в последовательность векторов признаков из входного речевого сигнала.

Первый шаг – преобразование аналогового сигнала в цифровой сигнал. Этот процесс проходит в два этапа: дискретизация и квантование. На рисунке представлен пример входного сигнала:

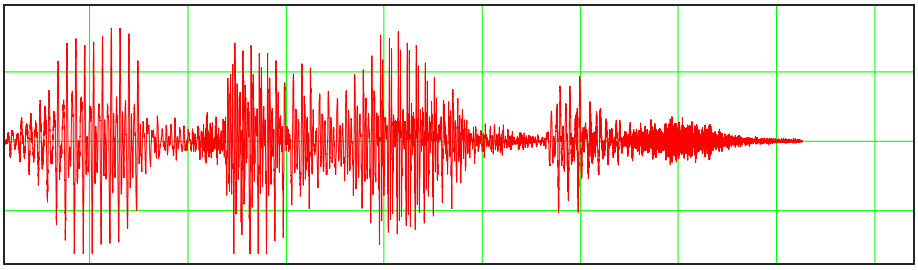

Рисунок 2.5 Входной сигнал

При дискретизации происходит разбиение сигнала на отдельные участки измерения амплитуды через определенные временные интервалы. Частота дискретизации - число измерений в секунду. При частоте дискретизации, равной 8000Hz производится 8000 измерений в секунду, из этого следует, что умение эффективного хранения значений амплитуды в памяти- крайне важно. Для этого нужно использовать 16-битные целые числа. Квантование-процесс представления целыми значениями вещественных значений. Далее оцифрованный сигнал будет обозначаться как x[t], где t- некоторый момент времени.

Весь записанный сигнал нужно разбить на перекрывающиеся окна конкретной длительности, в данном случае длина каждого из окон выбирается 23 мс, для того, чтобы упростить расчеты. Отдельная область показана на рисунке:

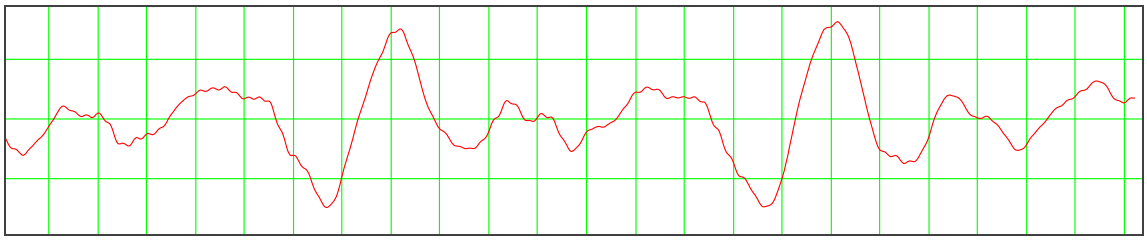

Рисунок 2.6 Область сигнала длительностью 23мс

Затем разбиваем оцифрованный сигнал на маленькие фрагменты, или фреймы, которые характеризуют определенные элементы речевого сигнала, для которых можно допустить, что сигнал - стационарный. Разделение на фрагменты делается следующим образом: выбираем оконную функцию, принимающую ненулевое значение внутри какого-то временного отрезка, а за его пределами – нулевое. Далее извлекается информация, находящаяся внутри фрагмента, путем последовательного применения оконной функции к сигналу. Извлечение сигнала производится по формуле:

$$y[t]=w[t]x[t] \quad (32)$$

Оконная функция определяется тремя величинами: шириной (в мс), смещением (число мс между границами последовательных окон) и формой. В данной работе применялось окно Хэмминга с шириной L = 25мс и смещением 10мс:

$$w[t] = 0.54 - 0.46cos\left(\frac{2\pi t}{L}\right), 0 \leq t \leq L - 1 \quad (33)$$

На рисунке представлен отфильтрованных от шумов и помех сигнал:

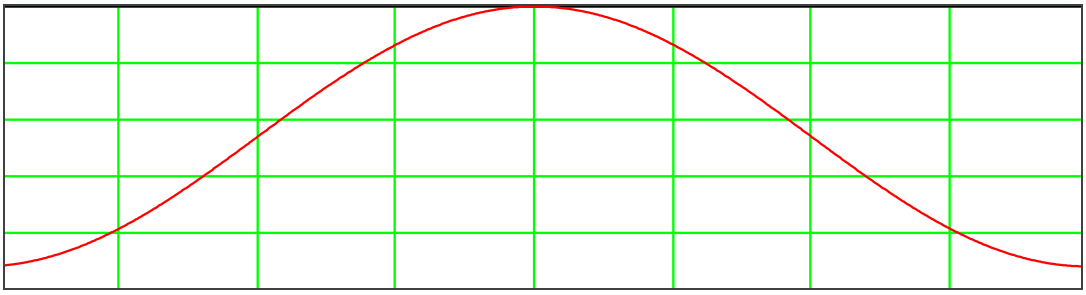

Рисунок 2.7 Фильтрация сегмента

Далее следует извлечь спектральную информацию для сигнала, который получили на предшествующих этапах. На этом этапе нужно выяснить, какое количество энергии находится в каждом частотном диапазоне. Для извлечения спектральной информации применяем дискретное преобразование Фурье.

На вход подаем разделенный на фреймы сигнал, а на выход для каждого из T частотных диапазонов – комплексное число X[k] , которое представляет собой амплитуду и фазу исходного сигнала:

$$X_k = \sum_{n=0}^{N-1} x_n \omega_n e^{\frac{-2\pi i}{N} kn}, k = 0, \dots, N-1. \quad (34)$$

где k соответствует частотам:

$$f_k = \frac{F_s}{N} k, k = 0, \dots, N/2, \quad (35)$$

где F_s является частотой дискретизация.

На рисунке представлен полученный спектр сегмента сигнала:

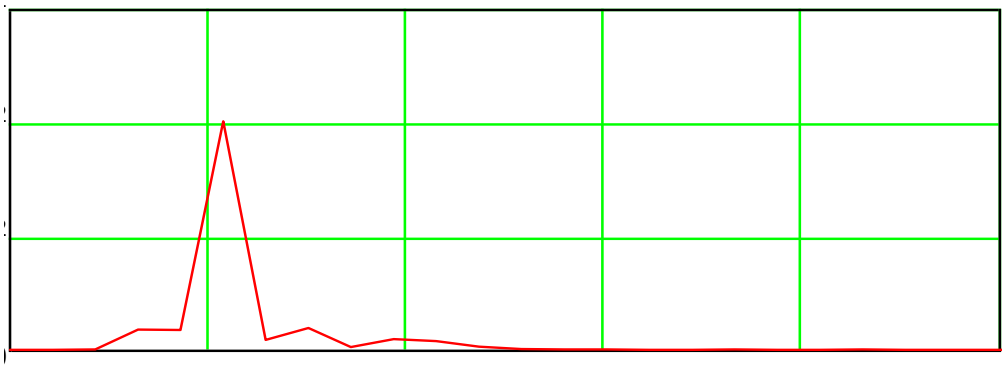

Рисунок 2.8 Спектр сегмента сигнала

Так же возможно использование быстрое преобразование Фурье:

$$X_p(i) = \sum_{i=0}^{255} S_p(t) e^{-j\frac{2\pi}{n}ni}, \quad (36)$$

Главная идея быстрого преобразования Фурье заключается в том, чтобы каждую вторую выборку можно было бы использовать для получения половинного спектра. Это означает, что формулу дискретного преобразования Фурье можно представить в виде двух сумм.

Далее, при помощи треугольных фильтров происходит разбитие на диапазоны. Шкала мел используется для расчета границ этих фильтров. Мел - единица высоты звука, основанная на восприятии этого самого звука ушами. Формула для перевода в мел-частотную область:

$$B(f) = 1127 \times \ln(1 + \frac{f}{700}) \quad (37)$$

Формула обратного преобразования:

$$B^{-1}(b) = 700(e^{b/1127} - 1) \quad (38)$$

Это необходимо для того, чтобы смоделировать тот факт, что человеческий слух обладает различную чувствительность для разных частотных диапазонов.

Для накопления значения энергии для каждого из частотных диапазонов (10 фильтров распределены линейно ниже 1000Hz, а оставшиеся - логарифмически выше 1000Hz) формируются треугольные фильтры, и берется логарифм каждого мел - значения. Использование логарифма делает оценку признаков не такими чувствительными к отличиям в способах подачи входного сигнала.

Последний этап данного метода- дискретное косинусное преобразование. На данной стадии вычисляют мел-частотные кепстральные коэффициенты (MFCC):

$$c_i = \sum_{m=0}^{N_{FB}-1} e_m \cos\left(\frac{\pi i(m+0,5)}{N_{FB}}\right), i = 1, \ldots, N_{MFCC}. \quad (39)$$

Коэффициент c_0 не используется, так как это энергия сигнала. На практике, количество мел-частотных кепстральных коэффициентов равняется 12.

Полученные окна, которые равномерно расположены на мел- оси:

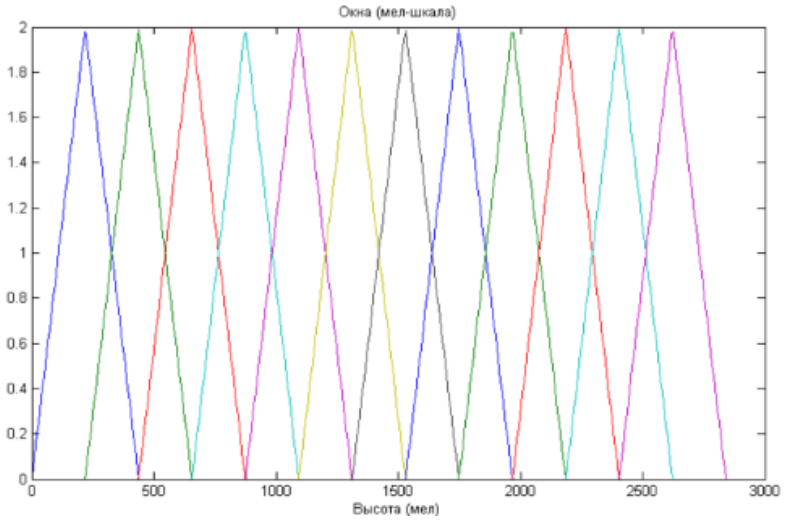

Рисунок 2.9 Окна на мел-оси

Далее переводим этот график в шкалу с частотами:

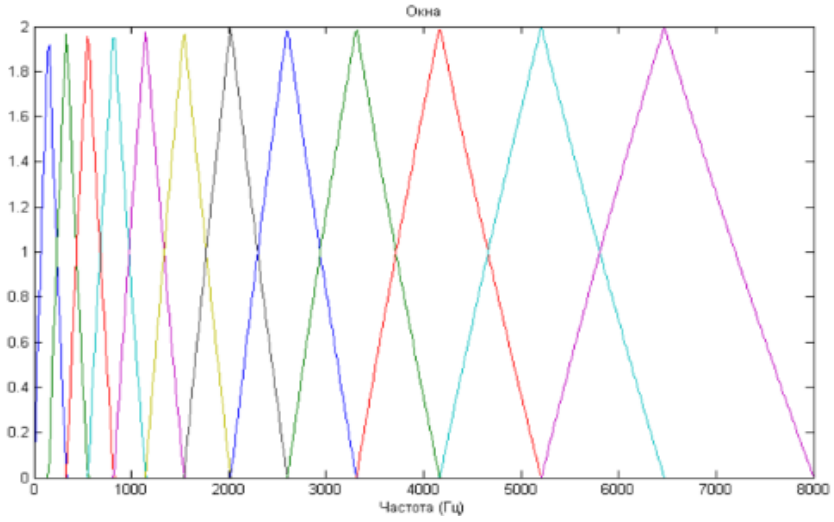

Рисунок 2.10 Частотная шкала

В данном методе так же участвуют кепстральные коэффициенты. Кепстр помогает отделить источник звуковой волны от фильтра, свойства которого могут проявиться при прохождении исходной волны по голосовому каналу,

когда образовывается звук. При этом в фильтре содержится большая часть полезной информации. Кепстр определяют следующим образом:

$$c[t] = \sum_{t=0}^{T-1} log\left(\left|\sum_{t=0}^{T-1} x[t]e^{-i\frac{2\pi}{T}kt}\right|\right) e^{i\frac{2\pi}{T}kt} \quad (40)$$

В качестве результата берем первые 12 значений кепстра. Кроме этого, для каждого фрейма добавляется новый признак – энергия сигнала:

$$E = \sum_{t=t_1}^{t_2} x^2[t] \quad (41)$$

Заключительным этапом является расчет дельта значений и двойных дельта значений для полученных на предшествующих этапах признаков, которые нужно добавить в результирующий вектор признаков. Дельта значения для кепстральных коэффициентов можно вычислить по формуле:

$$d(t) = \frac{c(t+1)-c(t-1)}{2} \quad (42)$$

Двойные дельта значения рассчитываются так же, только вместо кепстральных коэффициентов используют вычисленные дельта значения.

После формирования вектора признаков, к нему применяют фильтры Cepstral mean normalization и Cepstral variance normalization. Первый фильтр реализуется как вычитание из каждой компоненты вектора среднего значения по соответствующим компонентам всех полученных векторов и применяется для того чтобы снизить влияние канала на качество распознавания диктора. Фильтр Cepstral variance normalization применяется для повышения качества распознавания диктора в условиях шума. При этом преобразование выполняется путем умножения i-й компоненты вектора на значение.

$$CVN_i = \sqrt{\frac{N}{\sum_{j=1}^{N} c_{ij}^2}}, 1 \le i \le 39 \quad (43)$$

где N – общее число векторов, c_{ij} – i-я компонента вектора с номером j.

Вычисление мел-частотных кепстральных коэффициентов для каждого сегмента. На каждый сегмент сигнала приходится по 12 мел-частотных кепстральных коэффициентов. Для их нахождения пользуемся формулой:

$$c(n) = \sum_{m=0}^{M-1} S(n) \cos\left(\frac{\pi n\left(m+\frac{1}{2}\right)}{M}\right), \quad (44)$$

где $0 \leq n < M$.

Ниже представлена таблица с полученными коэффициентами сигнала. В каждом сегменте сигнала находится 12 мел-частотных кепстральных коэффициентов:

	0	1	2	3	4	5	6	7	8
0	67.688	71.731	78.922	78.037	84.704	88.818	82.423	89.211	88.177
1	19.817	21.717	20.388	16.238	22.921	16.518	13.339	16.984	14.582
2	56.622	59.533	61.038	61.363	68.06	71.571	61.344	61.619	56.916
3	0.669	2.446	-2.202	-0.59	0.468	1.826	5.556	5.503	5.365
4	54.491	57.629	61.945	53.216	60.183	57.545	51.55	52.413	50.969
5	3.352	4.81	9.271	6.086	8.033	9.512	4.043	6.386	4.513
6	37.058	34.728	32.224	34.371	26.661	24.566	36.415	40.002	38.96
7	10.714	8.806	9.739	6.155	5.121	3.149	0.443	2.618	-0.121
8	29.775	30.238	27.608	32.268	29.543	27.725	38.942	36.28	34.986
9	5.709	4.375	5.803	1.91	2.817	0.539	-0.564	-0.779	-3.4
10	32.062	37.022	38.777	40.568	41.377	36.97	38.557	40.166	37.865
11	-0.974	-0.22	2.833	-0.285	4.581	10.239	0.475	5.779	4.374

Рисунок 2.11 Таблица мел- частотных кепстральных коэффициентов

Для примера построен график, на котором изображены мел-частотные кепстральные коэффициенты двух первых сегментов записи речи двух разных людей, произносящих одну и ту же фразу:

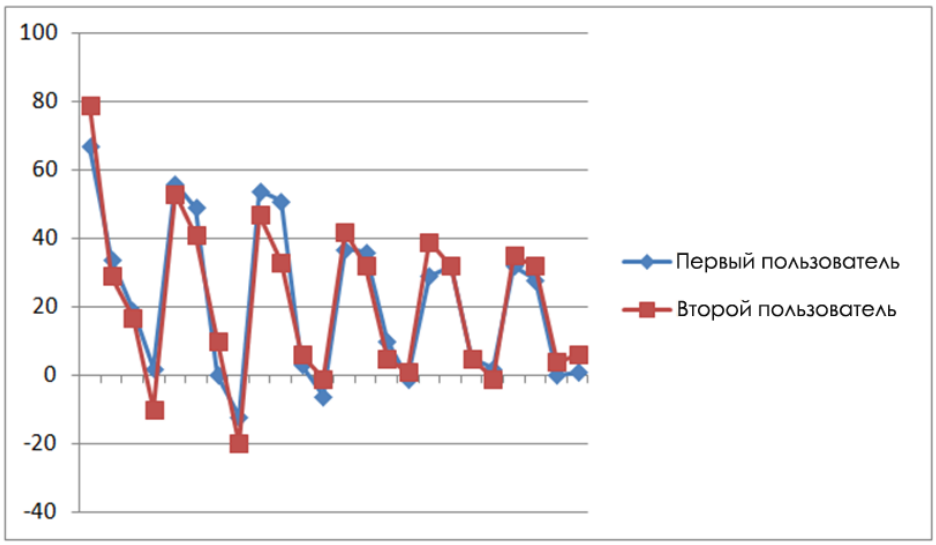

Рисунок 2.12 График MFCC двух разных личностей

На графике видно разницу между коэффициентами записи речи разных пользователей. Что касается двух разных записей речи, которые произносились одним и тем же человеком, то результат виден на рисунке:

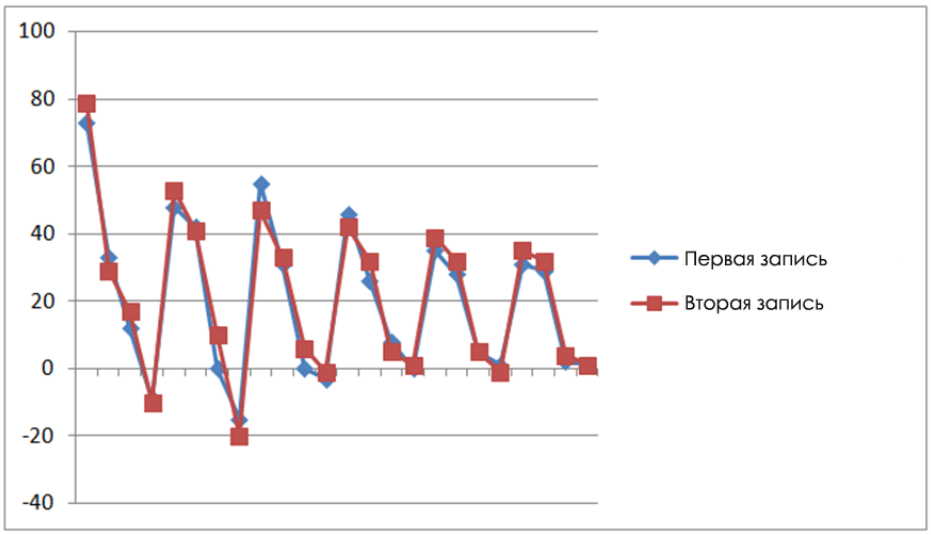

Рисунок 2.13 График MFCC одного человека

Невооруженным глазом видно, что разница между мел-частотными кепстральными коэффициентами не существенна.

После получения всех коэффициентов, записанный сигнал нужно сравнить со своим эталоном из базы данных. Для определения того, совпали эти сигналы или нет, вычисляем Евклидово расстояние между созданным шаблоном и эталоном:

$$rasst(C1, C2) = \sqrt{\sum_{k=1}^{n}(C1_k - C2_k)^2}\,, \quad (45)$$

где C1 и C2 являются массивами с мел-частотными кепстральными коэффициентами.

Вычисленные расстояния между 14 различными записанными речами, некоторые из которых совпадают, показаны на рисунке:

=

	0	1	2	3	4	5	6	7	8	9	10
0	0	0.479	0.434	0.442	0.399	0.584	0.532	0.434	0.442	0.399	0.498
1	0.479	0	0.271	0.28	0.362	0.504	0.514	0.271	0.28	0.362	0.453
2	0.434	0.271	0	0.256	0.26	0.562	0.551	0	0.256	0.26	0.517
3	0.442	0.28	0.256	0	0.359	0.53	0.536	0.256	0	0.359	0.526
4	0.399	0.362	0.26	0.359	0	0.652	0.585	0.26	0.359	0	0.557
5	0.584	0.504	0.562	0.53	0.652	0	0.274	0.562	0.53	0.652	0.384
6	0.532	0.514	0.551	0.536	0.585	0.274	0	0.551	0.536	0.585	0.334
7	0.434	0.271	0	0.256	0.26	0.562	0.551	0	0.256	0.26	0.517
8	0.442	0.28	0.256	0	0.359	0.53	0.536	0.256	0	0.359	0.526
9	0.399	0.362	0.26	0.359	0	0.652	0.585	0.26	0.359	0	0.557
10	0.498	0.453	0.517	0.526	0.557	0.384	0.334	0.517	0.526	0.557	0
11	0.401	0.391	0.438	0.465	0.486	0.481	0.444	0.438	0.465	0.486	0.335
12	0.399	0.361	0.38	0.422	0.386	0.497	0.464	0.38	0.422	0.386	0.393
13	0.442	0.28	0.256	0	0.359	0.53	0.536	0.256	0	0.359	0.526
14	0.399	0.362	0.26	0.359	0	0.652	0.585	0.26	0.359	0	...

Рисунок 2.14 Таблица вычисленных расстояний

2.3 Определение алгоритма инициализации и оценки параметров модели

Существует проблема инициализации начальных параметров модели, при обучении модели гауссовых смесей. Результат работы системы распознавания при этом сильно зависит от исходных значений параметров модели.

В данной работе используется алгоритм инициализации начальных параметров, который основан на кластеризации векторов признаков речевого сигнала. В качестве алгоритма кластеризации используется алгоритм K-means++, евклидово расстояние в котором используется в качестве меры искажения.

Алгоритм K-means++ - это модификация алгоритма K-means. Суть улучшения заключается в поиске более «хороших» изначальных значений центроидов кластеров. Алгоритм k-means не определяет то, как выполняется данный этап у алгоритма, и из-за этого является нестабильным. Центр первого кластера выбирается рандомно, после этого каждый следующий центр избирается из оставшихся точек данных с вероятностью, пропорциональной квадрату расстояния до самого ближнего существующего центра кластера. После этого применяется стандартный алгоритм K-means. Данный метод дает достаточно хорошее снижение погрешности итогового результата. Хотя начальный выбор в алгоритме захватывает дополнительное время, главная часть алгоритма K-means сходится довольно быстро.

Алгоритм k-means применяется к объектам, представляющимся точками в d-мерном векторном пространстве. Таким образом, это кластеры набора d-мерных векторов, $D = \{x_i | i = 1, \ldots, N\}$, где $x_i \in R^d$ обозначает i-ый объект или «точку данных». Алгоритм k-means связывает все точки данных в D так, что каждая точка x_i попадает только в определенный один из k разделов. Можно отслеживать, какая именно из точек находится в конкретном кластере, если назначить номер кластера каждой точке. Точки с таким же номером кластера находятся в одном и том же кластере, в то время как точки с различными номерами кластера находятся в разных кластерах. Это можно обозначить как кластерный составной вектор m длинной N, где m_i будет номером кластера x_i.

Значение k из входных данных алгоритма будет главным. Обычно, значение k основывается на нескольких критериях, например, на предварительном знании о том, какое количество кластеров будет в D; на том, какое количество кластеров нужно для нынешнего приложения, или какие типы кластеров, которые найдены путем изучения/экспериментирования с различными значениями k. Каким образом выбран k неважно для того, чтобы понять, как разделяет набор данных D k-means.

В k-means, каждый из k кластеров представлен одной точкой в Rd. Обозначим этот набор представителей кластера как множество C = {cj|j=1, ….., k}. Эти представители k кластеров также называются состояние кластера или центройды кластера: мы обсудим причину этого после описания целевой функции k-means.

В алгоритмах кластеризации точки группируются некоторым понятием «близости» или «подобия». В k-means мера близости по умолчанию Евклидово расстояние. В частности можно с готовностью показать, что k-means пытается минимизировать следующую неотрицательную функцию стоимости.

Иначе говоря, k-means старается уменьшить итоговый квадрат Евклидова расстояния между каждой из точек x_i и ее ближайшим представителем кластера c_j. Целевая функция k-means:

$$cost = \sum_{i=1}^{N}(\arg min_j \, ||x_i - c_j||_2^2) \quad (45)$$

Алгоритм k-means делит D итеративным способом, который чередуется между двумя шагами: переприсваиванием всем точкам в D номера кластера и между обновлением представителей кластера, которые основаны на точках данных в каждом кластере.

Работа алгоритма. Первым делом, представители кластера инициализируются, путем выбора k из R^d. Выбрать эти начальные источники можно с помощью случайной выборки из набора данных, если устанавливать их как решение кластеризации небольшого подмножества данных, или, если нарушить глобальное среднее значение данных k раз. В алгоритме инициализируется случайно выбранные k точки. Затем происходит выполнение алгоритма итерации до сходимости за пару шагов:

— Присвоение данных. Каждой из точек данных присваивается ее ближайший представитель, при этом происходит произвольное нарушение связей, которое приводит к разделению данных;

— Перемещение «средних». Каждый из представителей кластера направляется к центру, который состоит всех точек данных, ему

присвоенных. Данное множество точек, единственный оптимальный представитель для этого множества, не что иное, как срединная точка данных. Благодаря этому представителя кластера обычно называют срединным элементом кластера или центроидом кластера.

Алгоритм сходится, когда присвоения (и значения C_j) не подвергаются изменениям. Целевая функция k-means станет уменьшаться каждый раз, когда будут изменения в присвоении или шагах измерения, и сближение гарантировано за конечное число итераций.

Как говорилось ранее, выбрать оптимальное значение k довольно трудно. Необходимо пользоваться другими критериями выбора k, таким способом можно решить проблему выбора модели. Простейшее решение этой проблемы - пробовать по несколько различных значений k и выбирать кластеризацию, минимизирующую целевую функцию k-means. К сожалению, значение целевой функции информативно не так сильно, как можно надеяться в данном случае. Для последующей оценки параметров модели применяется EM-алгоритм, на каждой итерации которого вычисляются следующие значения:

$$p_i' = \frac{1}{T}\sum_{t=1}^{T} p(i|\bar{x_t}, \lambda) \qquad (46)$$

$$\mu_l' = \frac{\sum_{t=1}^{T} p(i|\bar{x_t},\lambda)\bar{x_t}}{\sum_{t=1}^{T} p(i|\bar{x_t},\lambda)} \qquad (47)$$

$$\sum i = \frac{\sum_{t=1}^{T} p(i|\bar{x_t},\lambda)(\bar{x_t}-\overline{\mu_l})(\bar{x_t}-\overline{\mu_l})^T}{\sum_{t=1}^{T} p(i|\bar{x_t},\lambda)} \qquad (48)$$

Эти шаги нужно выполнять до схождения параметров, или до достижения максимального числа итераций. В данной работе используется максимальное число итераций, равное 100. При этом значение апостериорной вероятности вычисляется по формуле:

$$p(i|\bar{x_t}, \lambda) = \frac{p_i b_i(\bar{x_t})}{\sum_{k=1}^{M} p_k b_k(\bar{x_t})} \qquad (49)$$

2.4 Определение числа компонентов модели гауссовых смесей

Одной из основных проблем при обучении модели гауссовых смесей является выбор числа компонентов модели. Теоретического решения этой задачи не существует.

Выбрать оптимальное число компонентов модели можно, перебрав и оценив точность распознавания при заданном числе компонентов.

В процессе выбора оптимального числа компонентов в данной работе было выявлено, что точность распознавания повышается при увеличении числа компонентов до 5, после чего этот процесс заметно замедляется. Это наблюдение позволяет сделать вывод о том, что для решения поставленной в рамках данной работы задачи, необходимо число компонент, равное 5.

2.5 Тестирование модели личности по голосу

Для тестирования были использованы тренировочные и тестовые записи, которые включали в себя 20 голосов различных людей. Проведенное тестирование помогло выявить оптимальное число компонент модели гауссовых смесей для поставленной задачи, которое равняется 5. При указанном числе компонент система идентификации показала точность 95%. С уверенностью можно говорить о применимости модели гауссовых смесей для решения задачи автоматической идентификации диктора по голосу.

Зависимость числа корректно идентифицированных дикторов от числа компонент модели гауссовых смесей:

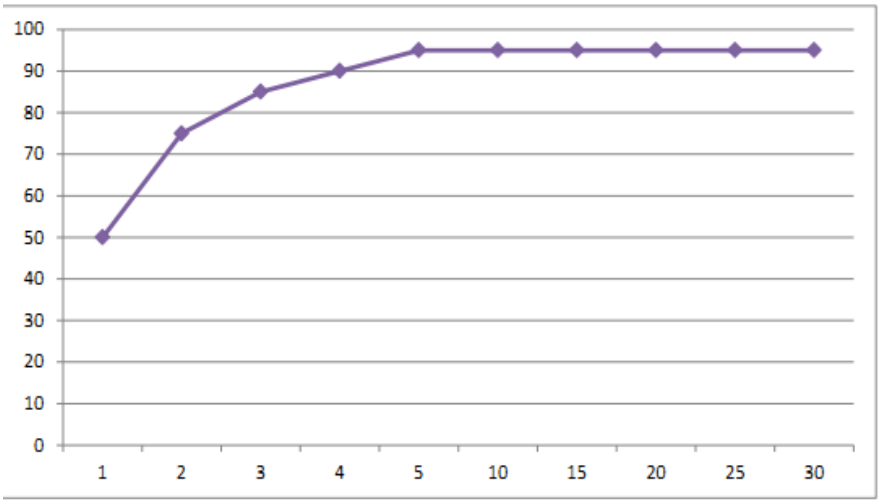

Рисунок 2.15 Результаты тестирования системы. По горизонтальной оси – число компонентов модели, по вертикальной – точность идентификации

2.6 Выводы

В рамках данной работы проводились исследования уже существующих методов решения задачи автоматической идентификации диктора по голосу.

В результате был проведен полный обзор предметной области и реализован один из лучших алгоритмов, используемых для решения поставленной задачи, который основан на применении модели гауссовых смесей. Компоненты гауссовых смесей моделируют уникальные особенности голоса человека, это позволяет с высочайшей точностью отличать по голосу разных людей.

Кроме того, было установлено, что для существенного увеличения скорости обучения и для повышения точности идентификации, целесообразно использовать алгоритм K-means++ для инициализации начальных параметров модели.

Глава 3 Результаты компьютерного моделирования модели идентификации личности по голосу

3.1 Схема работы модели идентификации личности по голосу

Программное средство состоит из нескольких частей, из самой программы и базы данных пользователей.

Основная программа представляет собой приложение с интерфейсом, предназначенным для работы с людьми. В ее функции входит:

— Регистрация новых пользователей;

— Идентификация пользователей;

— Получение информации о программном средстве.

Для регистрации, требуется войти в специальную вкладку интерфейса, ввести логин и записать свой голос, который будет сохранен в базе данных.

Для того, чтобы идентифицировать пользователя нужно ввести логин, который был зарегистрирован ранее, и еще раз сказать фразу или несколько фраз, произнесенные при регистрации.

Неотъемлемой частью программного средства является база данных, состоящая из:

— Файла, хранящего все логины зарегистрированных пользователей;

— Файлов, где раздельно хранится необходимая для идентификации или не идентификации информация о записанном голосе каждого из пользователей.

Функциональная схема представлена на рисунке 3.1.

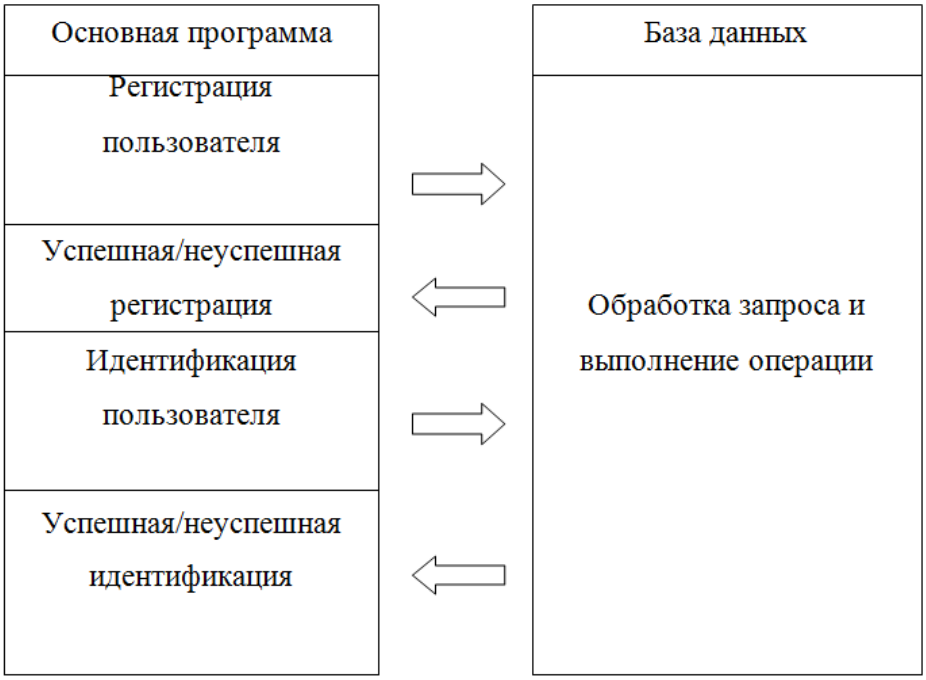

Рисунок 3.1 Функциональная схема программного средства

В то время, когда работает программа, между ее составляющими будут протекать некоторые потоки информации. Начинаются они с микрофона, на который был записан голос, и заканчиваются данными, видимыми на экране. Вся информационная схема представлена на рисунке:

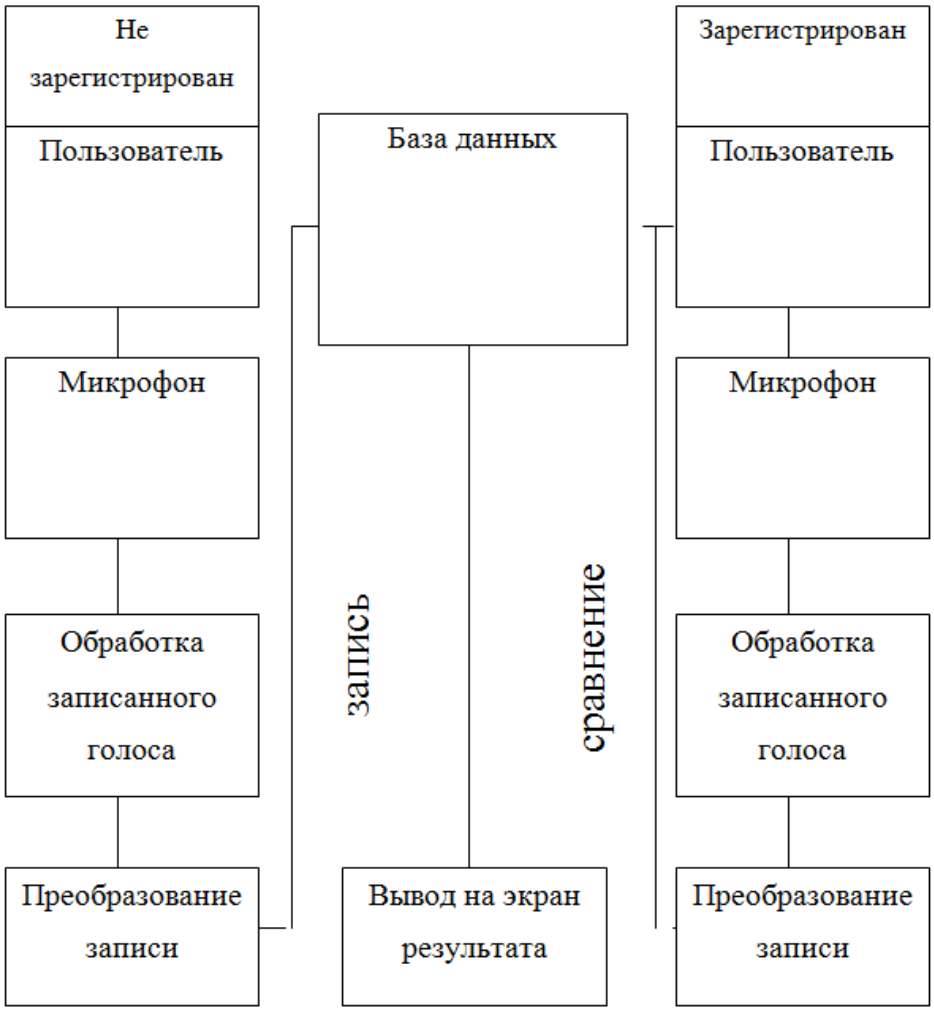

Рисунок 3.2 Информационная схема программного средства

Основная программа имеет графический интерфейс и предназначена для регистрации новых пользователей, а так же для идентификации пользователя по голосу. На вход основной программы поступает человеческая речь, а на выход - кепстральные коэффициенты, которые сравниваются и выдают положительный или отрицательный результат.

База данных состоит из главного файла, хранящего все логины зарегистрированных пользователей и из файлов, хранящих кепстральные коэффициенты отдельно каждого из пользователей.

Во время работы программы происходит прямое взаимодействие основной программы и базы данных. При регистрации в основной программе происходит сохранение данных пользователей в базе данных, а так же при идентификации происходит прямое обращение к базе данных, с целью сравнения кепстральных коэффициентов, которые были записаны, и вывода результата на экран.

Полная блок-схема основного алгоритма программного средства показана на рисунке:

Рисунок 3.3 Блок-схема основного алгоритма программного средства

3.2 Листинг программы

Листинг программы находится в Приложении А.

3.3 Результат работы программы

База данных программного средства хранится в зашифрованных файлах. Она состоит из файла, в котором хранятся логины зарегистрированных пользователей, и из множества файлов, хранящих мел-частотные кепстральные коэффициенты каждого пользователя по отдельности.

Основная программа состоит из панели инструментов, которой можно воспользоваться для выхода из программы, а так же для того, чтобы узнать информацию о программе и ее разработчике. При нажатии кнопки «выход», осуществляется полный выход из программы:

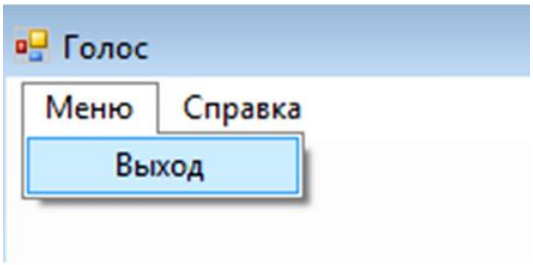

Рисунок 3.6 Меню с выходом из программы

Для того, чтобы узнать информацию о программе нужно нажать кнопку «Справка»:

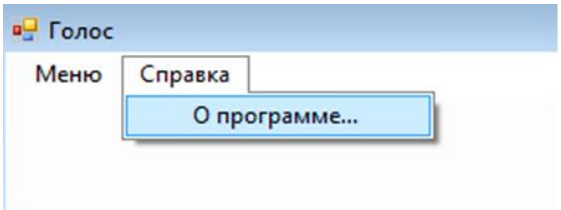

Рисунок 3.7 Справка о программе

Окно с информацией о программе содержит название программы, информацию о разработчике и год разработки программы :

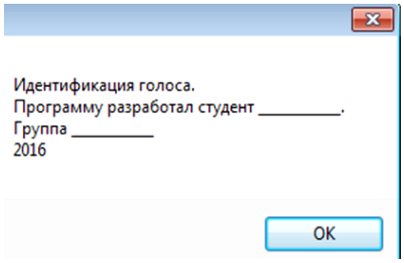

Рисунок 3.8 Информация о программе

Вводить логин нужно в специальном поле:

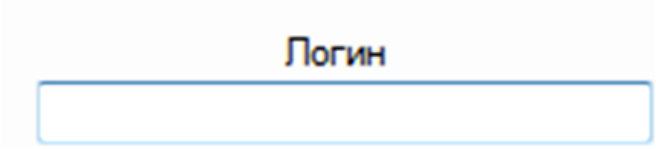

Рисунок 3.9 Поле для ввода логина

В случае, если введенный логин не найден в базе данных, то на экран производится вывод сообщения «ПОЛЬЗОВАТЕЛЬ НЕ НАЙДЕН!»:

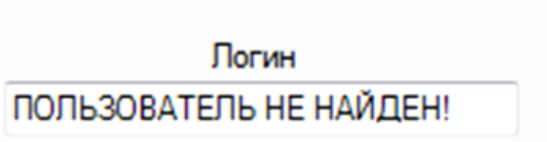

Рисунок 3.10 Регистрация нового пользователя

Если вы являетесь незарегистрированным в системе пользователем, то вам необходимо пройти процедуру регистрации нового пользователя:

Рисунок 3.11 Регистрация нового пользователя

При нажатии на ссылку «Новый пользователь» появляется кнопка «Записать голос» :

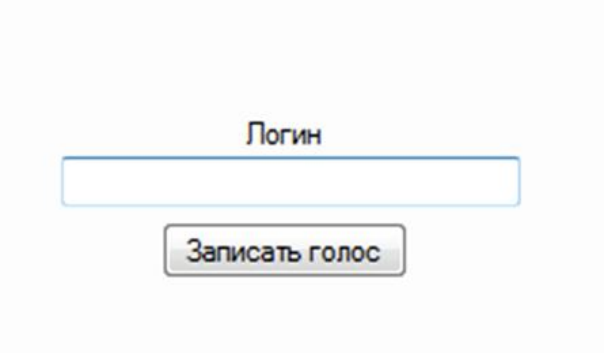

Рисунок 3.12 Запись голоса нового пользователя

В специальном поле ввода запишется будущий логин пользователя. При нажатии кнопки «Записать голос», пользователь произносит определенную фразу, например, его имя и фамилию. При успешной регистрации всплывает окно «Пользователь добавлен!»:

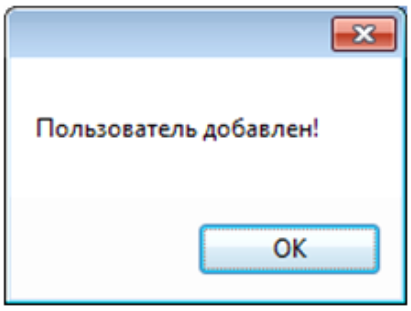

Рисунок 3.13 Успешная регистрация пользователя

Данное окно подтверждает успешную регистрацию, это значит, что пользователь добавлен в базу данных программы и может пройти процедуру идентификации.

После введения логина нужно нажать кнопку «Идентификация». Далее нужно произнести речь, которая говорилась при регистрации нового пользователя. В случае не прохождения идентификации, появится надпись «Не совпадение!», то значит, что записанный голос не совпал с голосом, хранящимся в базе данных. В подтверждении этого, на экран выведутся спектрограммы записанного голоса и голоса из базы данных:

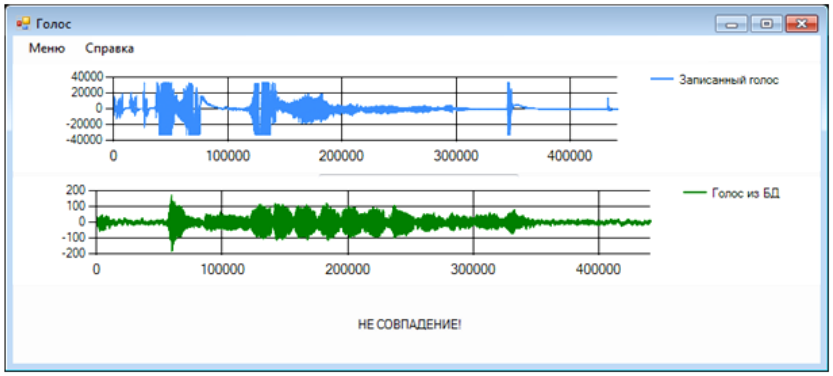

Рисунок 3.14 Неудачная идентификация

При совпадении голосов выводится сообщение «Совпадение!» и на экране показываются спектрограммы совпавших голосов. Это подтверждает успешную идентификацию:

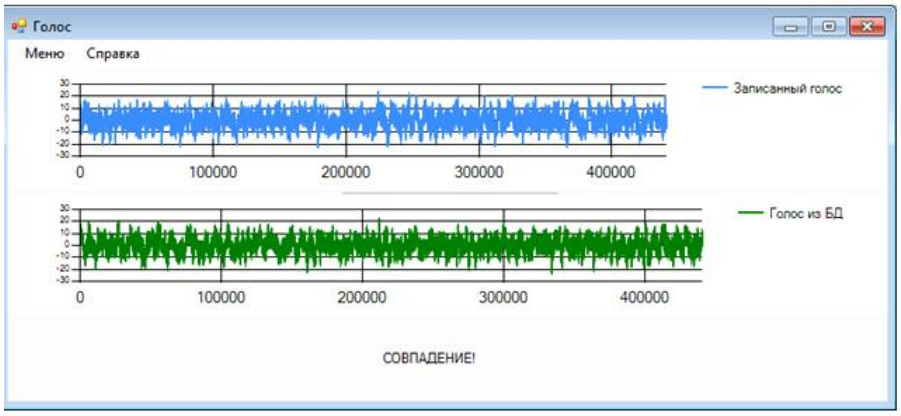

Рисунок 3.15 Успешная идентификация

При тестировании программного средства было задействовано три личности, каждый из которых произнес одинаковую фразу («звукозапись»), с целью нахождения ошибок первого и второго рода, а именно:

— Вероятности события, когда злоумышленник получает доступ к системе;

— Вероятность события, когда пользователь, который зарегистрирован в системе не получает доступ к системе.

Так же в экспериментальной части был найден порог евклидового расстояния, который помог определению того, прошел идентификацию пользователь или нет. Данный порог равнялся 0.4. Это обозначает, что если

расстояние меньше числа 0.4, то на записях один и тот же человек, а если больше – то значит, речь произнесли разные люди.

На рисунке 3.16 представлена таблица евклидовых расстояний между разными речами людей. Был записан эталон, а злоумышленники делают попытки пройти идентификацию через какого-то пользователя. С 1 по 10 показаны попытки одного человека, а с 11 по 20 - другого.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0.584	0.532	0.504	0.609	0.545	0.566	0.575	0.584	0.621	0.564	0.536	0.498	0.401	0.399	0.521	0.511	0.438	0.42	0.406	0.498

Рисунок 3.16 Евклидово расстояние между записью пользователя и злоумышленников

На рисунке видно, что из 20 попыток идентификацию получилось пройти один раз, но так как порог был пройдет только на 0,001, можно сделать вывод, что вероятность ошибки первого рода равняется 1 к 20, то есть 5%.

На следующем рисунке показана таблица сравнения эталона и записи нужного человека. С ее помощью мы можем определить ошибку второго рода, когда у пользователя нет доступа к системе:

	0	1	2	3	4	5	6	7	8	9	10
0	0	0.308	0.341	0.339	0.362	0.388	0.157	0.162	0.195	0.203	0.212
1	0.308	0	0.479	0.434	0.442	0.399	0.304	0.314	0.329	0.33	0.335
2	0.341	0.479	0	0.271	0.28	0.362	0.359	0.341	0.342	0.356	0.329
3	0.339	0.434	0.271	0	0.256	0.26	0.337	0.317	0.291	0.311	0.278
4	0.362	0.442	0.28	0.256	0	0.359	0.359	0.347	0.324	0.33	0.306
5	0.388	0.399	0.362	0.26	0.359	0	0.355	0.323	0.3	0.336	0.3
6	0.157	0.304	0.359	0.337	0.359	0.355	0	0.129	0.165	0.141	0.181
7	0.162	0.314	0.341	0.317	0.347	0.323	0.129	0	0.144	0.16	0.17
8	0.195	0.329	0.342	0.291	0.324	0.3	0.165	0.144	0	0.143	0.117
9	0.203	0.33	0.356	0.311	0.33	0.336	0.141	0.16	0.143	0	0.138
10	0.212	0.335	0.329	0.278	0.306	0.3	0.181	0.17	0.117	0.138	0
11	0.16	0.318	0.324	0.322	0.34	0.364	0.151	0.153	0.174	0.17	0.191
12	0.273	0.423	0.287	0.247	0.248	0.367	0.271	0.274	0.255	0.252	0.238
13	0.193	0.381	0.292	0.302	0.321	0.33	0.186	0.189	0.18	0.193	0.182
14	0.205	0.418	0.307	0.265	0.307	0.316	0.221	0.21	0.187	0.218	0.184
15	0.162	0.334	0.308	0.307	0.323	0.368	0.157	0.152	0.155	0.163	0.169
16	0.142	0.29	0.339	0.353	0.37	0.377	0.155	0.147	0.196	0.205	0.211
17	0.143	0.291	0.341	0.338	0.364	0.354	0.141	0.134	0.185	0.194	0.204
18	0.164	0.337	0.311	0.32	0.336	0.335	0.166	0.148	0.159	0.195	0.166
19	0.189	0.351	0.31	0.295	0.317	0.334	0.188	0.169	0.143	0.184	...

Рисунок 3.17 Евклидово расстояние записей речи одного человека

Из таблицы видно, что вероятность ошибки второго рода при ситуациях, когда человек находится в одинаковом психологическом и физическом состоянии, сводится к 0 (если посмотреть на нулевой столбец, то можно увидеть, что ни одно из значений не превышает значение 0,4). Из 20 экспериментальных записей все 20 прошли идентификацию.

3.4 Расчет затрат на программно - аппаратную часть

Целью расчетов, которые приводятся в этом разделе, является определение себестоимости разработанной программы.

Цикл разработки любого из современных программных продуктов состоит из нескольких этапов [30]:

— Планирование - определение сроков разработки программного обеспечения;

— Проектирование - исследование, моделирование, выработка требований к проектированию, концептуальное проектирование, детальное проектирование, выработка требований к разработке;

— Разработка - программная реализация;

— Тестирование;

— Ввод в эксплуатацию.

Для нахождения себестоимости необходимо учесть:

— Трудоемкость разработки;

— Оплату работы программиста;

— Покупку программного обеспечения;

— Затраты электроэнергии, которая расходуется техническими средствами;

— Накладные расходы;

— Единый социальный налог.

В создании данной работы использовались следующие технические средства:

— Ноутбук ASUS N75S;

— Принтер HP 4500.

Закупочная цена компьютера составляет 50 500 руб, заакупочная цена принтера - 3400 руб.

Общая стоимость технических средств:

$$Ц_{тс}=Ц_{к}+Ц_{п}$$

где $Ц_{к}$ –цена компьютера, $Ц_{п}$-цена принтера

Следовательно:

$$Ц_{тс} = 50500+3400 = 54900руб$$

Для разработки программы применялось следующее программное обеспечение:

— Платформа Windows 7 = 9200руб

— Пакет прикладных программ MatLab = 15000руб

— Офисный пакет Microsoft Office 2007 Professional = 13 000 руб.

Общая стоимость программного обеспечения:

$$Ц_{о}=Ц_{тс}+Ц_{по}$$

где $Ц_{тс}$ –цена технических средств, $Ц_{по}$-цена программного обеспечения

Общая стоимость технических средств и программного обеспечения:

$$Ц_{о} =54900+15000+13000 =82900руб$$

ПЭВМ, на которой велась разработка программы, потребляет электроэнергию сети переменного тока, напряжением 220 В. Согласно

технической документации, мощность, которую суммарно потребляет компьютер и принтер, составляет:

$$М_с= 250 \text{ Вт·ч.}$$

Расход денежных средств, затраченный на энергопотребление технических средств можно найти по формуле, руб.:

$$Р_э=К_{дн}·В_{раб}·М_с·Ц_{эн},$$

где $К_{дн}$ – период написания программы, дн., $В_{раб}$ – длительность рабочей смены, ч., $М_с$ – мощность, потребляемая техническими средствами, кВт·ч, $Ц_{эн}$ – стоимость электроэнергии по действующим тарифам, р./кВт·ч

$К_{дн}$=60 дней

$В_{раб}$=5 часов;

$Ц_{эн}$=1,5 рубля за кВт·ч.

Из этого следует:

$$Рэ=60·5·0,25·1,5 = 112,5$$

Исходя из количества времени работы программиста, которое составило 60 рабочих пятичасовых дней, можно найти количество фактического отработанного времени, ч.:

$$Т_ф=К_{дн}·В_{раб},$$

где $Т_ф$ – фактически отработанное время, ч., $К_{дн}$ – количество отработанных дней, дн., $В_{раб}$ – продолжительность рабочего дня, ч.

$$Т_ф=60·5=300.$$

Заработную плату программиста считаем в размере 250 руб. в час, получаем основную заработную плату, руб.:

$$З_{осн}=Т_ф·Т_ч,$$

где $Т_ч$ – часовая тарифная ставка программиста.

$$З_{осн}=300\cdot250=75000руб$$

Далее определяем единый социальный налог. В соответствии со ставкой он составляет 26% от расходов на оплату труда, что составляет, руб.:

$$О_{соц}=Робщ\cdot0,26$$

$$О_{соц}=75000\cdot0,26=19500руб.$$

Для того, чтобы определить отпускную цену, необходимо учесть:

- Прибыль от реализации 15% ;
- Налог на добавленную стоимость 18%.

$Ц_н = Т + П_н$

где $П_н$ = прибыль от реализации проекта, Т= итог затрат на создание программы.

$$Т=Ц_{тс}+Ц_{по}+Р_э+О_{соц} = 82900+112,5+19500=102512,5 руб$$

$$П_н = Т*0.15 =102512,5*0.15=15,376 руб$$

$$Ц_{н=}102512,5+15,376=117888 руб$$

Цена пакета программы с учетом НДС, руб:

$$Ц_п=Ц_н+НДС$$

где НДС-налог на стоимость

$$НДС = Ц_н * 0.18$$

$$НДС=117888*0.18=21219руб$$

$$Ц_{п=}117888+21219=139107руб$$

3.5 Выводы

В рамках данной работы было разработано программное средство, предназначенное для идентификации личности по голосу, для допуска пользователей, удачно прошедших идентификацию и отказа в допуске для тех, кто не прошел идентификацию.

Точность идентификации достигается 95%, что говорит о том, что реализованный алгоритм можно успешно применять для решения поставленной задачи.

Общий объем единовременных затрат, включающий в себя оплату труда разработчика и оплату аппаратно-программного комплекса составляет 102512 рублей. Цена пакета программы с учетом НДС равняется 139107 рублям.

Глава 4 Безопасность жизнедеятельности

4.1 Общие требования

Охрана труда - система законодательных актов, социально-экономических, организационных, технических, гигиенических и лечебно-профилактических мероприятий и средств, которые обеспечивают безопасность, сохранность здоровья и работоспособности человека в процессе труда. Охрана труда помогает выявлять и изучать причины, из-за которых происходят несчастные случаи на производстве, профессиональные заболевания, аварии, взрывы, пожары. Так же охрана труда разрабатывает систематические мероприятия и требования, которые позволяют устранять эти причины и создают безопасные и благоприятные для человека условия труда.

Производственные здания, оборудование, технологические процессы обязаны отвечать всем требованиям, которые обеспечивают здоровые и безопасные условия труда. Данные требования должны выполнять в первую очередь проектировщики. Согласно санитарным нормам СН 245-71 каждому рабочему положен объём производственного помещения не менее 15 $м^3$ и площадь не менее 4,5 $м^2$.

4.2 Неблагоприятные факторы и средства защиты от них

В наше время работники вычислительных центров сталкиваются с воздействием многих физических вредных факторов, например, с повышенным уровнем шума, отсутствием или недостатком естественного освещения, повышенной температурой внешней среды, недостаточной освещенностью рабочей зоны, электрическим током, статическим электричеством и других.

Длительное нахождение человека в зоне воздействия вредных факторов способно привести к различным заболеваниям.

Работа пользователя на компьютере сопровождается незначительным шумом. Главный источник - шумящее оборудование (мфу и т.п.). Шум оказывает на человека вредное физиологическое воздействие, заключающееся как в повреждении слухового аппарата, так и в негативном влиянии на нервную систему, что способствует замедлению психологических реакций.

Уровень шума на рабочем месте не должен превышать установленных значений для видов работ «Санитарными нормами допустимых уровней шума на рабочих местах», а именно 60 дБ.

Для снижения уровня шума в помещениях с ПЭВМ используют звукопоглощающие материалы с максимальными коэффициентами звукопоглощения для отделки помещений (разрешённых органами Госсанэпиднадзора России), которые подтверждены специальными акустическими расчётами.

Помимо акустических колебаний, которые передаются по воздуху, так же могут присутствовать и механические колебания, которые передаются через конструкции и почву. Под воздействием вибраций в нервной и костно-суставной системах происходит изменение, возможно повышение артериального давления. Вибрации необходимо устранять, еще при установке оборудования требуется уделять особое внимание уравновешиванию положения машин.

Электромагнитное излучение - один из самых неблагоприятных факторов. Длительное воздействие электрического поля низкой частоты может вызвать функциональные нарушения центральной нервной и сердечно-сосудистой систем человека, а также изменения в кровяном составе. Биологическое действие электромагнитных полей высоких частот обладает тепловым и аритмическим эффектами.

Облучение электромагнитным полем большой интенсивности приводит к разрушениям в тканях и органах. При длительном хроническом воздействии электромагнитных полей малой интенсивности происходят различные нервные и сердечно-сосудистые расстройства (мигрени, утомляемость, нарушение сна).

Персонал, который работает с компьютером, подвергается не только электромагнитному излучению, но и ультрафиолетовому, низкоэнергетическому рентгеновскому и периодическим разрядам накапливающегося потенциала статического электричества. Для максимального обеспечения защиты работника необходимо приобретать оборудование, которое прошло сертификацию на стандарт не ниже ТСО’03.

Помещения, в который находятся компьютеры относятся к категории Д - пожароопасные. Категория характеризуется применением негорючих веществ в холодном состоянии.

По правилам устройства электроустановок такие помещения относятся к классу П-III - пожароопасные, характеризующиеся наличием твёрдых горючих веществ. Проектируемое помещение относится к классу помещений с повышенной опасностью, которые обладают электроустановками, имеющими соединение с землёй.

Электрические установки, к которым можно отнести почти все оборудование ЭВМ, потенциально представляют для человека большую опасность из-за того, что при эксплуатации или проведении профилактических работ человек может коснуться частей, которые находятся под напряжением.

Для того, чтобы предотвратить электротравматизм требуется правильная организация обслуживания действующих электроустановок вычислительного комплекса, своевременное проведение ремонтных, монтажных и профилактических работ. Для обеспечения снижения напряжения между оборудованием, которое находится под напряжением, и землёй до безопасной величины, нужно заземлять все металлические нетоковедущие части электрооборудования, которые могут оказаться под напряжением при пробоях изоляции электропроводки. Защитное заземление-это наиболее простая и эффективная мера защиты от поражения электрическим током.

К современному освещению на производстве предъявляются особые требования как гигиенического, так и технико-экономического характера. Правильно спроектированное освещение сможет обеспечить высокий уровень работоспособности, окажет положительное психологическое воздействие на работников, повлияет на повышение производительности труда. Используют два вида освещения: естественное и искусственное.

Напряжённость труда является характеристикой трудового процесса, которая в основном отражает нагрузку на центральную нервную систему. Условия и характер труда на рабочих местах характеризуются гигиенической классификацией труда, позволяющей произвести количественную оценку выраженности вредных факторов производственной среды, уровня напряжённости и тяжести трудового процесса. На данном основе устанавливается приоритетность проведении оздоровительных мероприятий. Для снижения напряженности труда необходимо делать перерывы в работе, менять темп работы, оборудовать рабочее место таким образом, чтобы оно обеспечивало нормальное функционирование опорно-двигательного аппарата и кровообращения. На данной основе устанавливается приоритетность проведении оздоровительных мероприятий.

4.3 Выводы

В ходе исследования факторов, неблагоприятно действующих на человека, были рассмотрены основные средства защиты от источников опасности, даны рекомендации по устранению данных факторов.

Заключение

В рамках данной работы проводились исследования существующих методов решения задачи автоматической идентификации диктора по голосу.

В работе был проведен полный обзор предметной области. Реализовался один из лучших алгоритмов, который можно использовать для решения поставленной задачи, основанный на применении модели гауссовых смесей. Данная модель считается передовой в виду того, что компоненты гауссовых смесей могут моделировать особенности голоса, индивидуальные для каждого человека, это позволяет отличать голоса людей с высочайшей точностью.

Было установлено, что использование алгоритма K-means++ для инициализации начальных параметров модели приводит к существенному увеличению скорости обучения и повышению точности идентификации. Помимо проверки точности идентификации проводилась оценка времени обучения модели гауссовых смесей. На полной выборке, при использовании кластеризации K-means для инициализации начальных параметров модели при числе компонентов модели, равном 5, время обучения равнялось 7 минутам 35 секундам. При использовании K-means++ время обучения равнялось 5 минутам 3 секундам, это говорит о преимуществе использованного алгоритма. Для тестирования программы были составлены шаблоны пользователей, которые состояли из голосов 20 человек.

В результате тестирования было определено оптимальное число компонент модели гауссовых смесей для решения поставленной задачи, оно

равняется 5. При данном числе компонент система идентификации показывает точность распознавания, равную 95%, что говорит о применимости модели гауссовых смесей для решения задачи автоматической идентификации диктора по голосу.

Список используемой литературы

1. Кузнецов В. Автоматический синтез речи / В. Кузнецов, А. Отт. - Таллинн: Валгус, 1989. – 56 с.;
2. Рабинер Л.Р. Цифровая обработка речевых сигналов / Л.Р. Рабинер, Р.В. Шафер. - М: Радио и связь, 1981. – 78 с.;
3. Михайлов А.А. Основные биометрические системы / А.А. Михайлов, А.А. Колосков, Ю.И. Дронов // Алгоритм безопасности. 2016. – 110 с.;
4. Рабинер Л. Теория применения цифровой обработки сигналов / Л. Рабинер, Б. Гоулд. - М: Мир, 1978. – 26 с.;
5. Каганов А.Ш. Соотношение перцептивных признаков и формативных частот гласных в речевом потоке. Сборник трудов. XIII сессия российского акустического общества / А. Ш. Каганов, В.Г. Михайлов. - М: 2000. – 98 с.;
6. Литюк В.И. Методы расчета и проектирование цифровых многопроцессорных устройств обработки радиосигналов. Методическое пособие № 2231 часть 3 / В. И. Литюк. - Таганрог: 1995. – 33 с.;
7. Рамишвили Г.С. Речевой сигнал и индивидуальность голоса / Г.С. Рамишвили. - Тбилиси: 1976. – 53 с.;
8. Сорокин В.Н. Верификация диктора по спектрально-временным параметрам речевого сигнала / В. Н. Сорокин, А. И. Цыплихин // Информационные процессы. 2010. – 21 с.;
9. Линдсей П. Переработка информации у человека / П. Линдсей, Д. Нордман. - М: Мир, 1974. –. 44 с.;
10. Татарченко Н. В. Биометрическая идентификация в интегрированных системах безопасности / Н. В. Татарченко, С. В. Тимошенко // Специальная техника. 2002. – 32 с.;

11. Фаин В. С. Распознавание образов и машинное понимание естественного языка / В. С. Фаин. - М: Наука, 1987. – 100 с.;
12. Оппенгейн А.В. Цифровая обработка сигналов / А. В. Оппенгейн, Р. В. Шафер. - М: Радио и связь, 1979. – 68 с.;
13. Татарченко И. В. Концепция интеграции унифицированных систем безопасности / И. В. Татарченко, Д. С. Соловьев // Системы безопасности. 1973. – 34 с.;
14. Макхоул Дж. Векторное квантование при кодировании речи / Дж. Макхоул // — ТИИЭР, 1985. – 22 с.;
15. Фланаган Дж. Анализ, синтез и восприятие речи / Дж. Фланаган. - М: Связь, 1968. – 44 с.;
16. Комарцова Л.Г. Нейрокомпьютеры. Учебное пособие для вузов / Л. Г. Комарцова, А. В. Максимов. - М: МГТУ им. Н.Э.Баумана, 2002. – 140 с.;
17. Абрамов А. М. Системы управления доступом / А. М. Абрамов, О. Ю. Никулин, А. И. Петрушин. - М: Оберег-РБ, 1998. – 88 с.;
18. Маркел Дж. Д. Линейное предсказание речи / Дж. Д. Маркел, А. Х. Грэй. - М: Радио и связь, 1980. – 75 с.;
19. Кузин М.В. Идентификация по голосу. Скрытые возможности / М. В. Кузин // Information Security. 2006. – 55 с.;
20. Попов Э. В. Общение с ЭВМ на естественном языке / Э. В. Попов. - М: Наука, 1982. – 99 с.;
21. Флорен М. В. Организация управления доступом / М. В. Флорен // Защита информации «Конфидент». 1995. – 56 с.;
22. Садыхов Р.Х. Модели гауссовых смесей для верификации диктора по произвольной речи / Р. Х. Садыхов, В. В. Ракуш // Доклады БГУИР. 2003. – 33 с.;
23. Chow D. Speaker Identification Based on Perceptual Log Area Ratio and Gaussian Mixture Models / D. Chow, H. Waleed, A. Robust. - Auckland, New Zealand: 2002. – 65 с.;
24. Горелик А. Л. Методы распознавания / А. Л. Горелик, В. А. Скрипкин. - М: Высшая школа, 1989. – 87 с.;

25. Кинтцель Т. Руководство программиста по работе со звуком / Т. Кинтцель. - М: 2000. – 90 с.;
26. Зиндер Л.Р. Общая фонетика / Л. Р. Зиндер. - М: Высшая школа, 1979. – 140 с.;
27. Furui S. Digital Speech Processing, Synthesis and Recognition / S. Furui. - New York: 1989. – 44 с.;
28. Childers D. G. The Cepstrum: A Guide to Processing / D. G. Childers, D. P. Skinner, R. C. Kemerait // Proceedings of the IEEE Vol. 65. 1977. – 45 с.;
29. Прохоров Ю. В. Вероятность и математическая статистика / Ю. В. Прохоров. - М: Большая Российская Энциклопедия, 1999. – 100 с.;
30. Константайн Л. Разработка программного обеспечения / Л. Константайн, Л. Локвуд. - СПб: Питер, 2004. – 56с.

```
function check_identified(value,
threshold, name)
                    if value < 1-threshold,
                       disp([name '     identified! ' num2str(value)]);
                    else
                       disp([name ' not identified! ' num2str(value)]);
                    end
                    end
%% Data Input
                  fs = 16000;
                  data_rm1 =
                  wavread('../data/TIMIT/Rag/Male/Speaker1.WAV');
                  data_rm2 =
                  wavread('../data/TIMIT/Rag/Male/Speaker2.WAV');
                  data_rm3 =
                  wavread('../data/TIMIT/Rag/Male/Speaker3.WAV');
                  data_rm4 =
                  wavread('../data/TIMIT/Rag/Male/Speaker4.WAV');
                  data_rf1 =
                  wavread('../data/TIMIT/Rag/Female/Speaker1.WAV');
                  data_rf2 =
                  wavread('../data/TIMIT/Rag/Female/Speaker2.WAV');
                  data_rf3 =
                  wavread('../data/TIMIT/Rag/Female/Speaker3.WAV');
                  data_rf4 =
                  wavread('../data/TIMIT/Rag/Female/Speaker4.WAV');
                  data_sm1 =
                  wavread('../data/TIMIT/Suit/Male/Speaker1.WAV');
                  data_sm2 =
                  wavread('../data/TIMIT/Suit/Male/Speaker2.WAV');
                  data_sm3 =
                  wavread('../data/TIMIT/Suit/Male/Speaker3.WAV');
                  data_sm4 =
```

```
wavread('../data/TIMIT/Suit/Male/Speaker4.WAV');
data_sf1 =
wavread('../data/TIMIT/Suit/Female/Speaker1.WAV');
data_sf2 =
wavread('../data/TIMIT/Suit/Female/Speaker2.WAV');
data_sf3 =
wavread('../data/TIMIT/Suit/Female/Speaker3.WAV');
data_sf4 =
wavread('../data/TIMIT/Suit/Female/Speaker4.WAV');
training_data1 = data_rm1;
training_data2 = data_rm2;
training_data3 = data_rm3;
training_data4 = data_rm4;
training_data5 = data_rf1;
training_data6 = data_rf2;
training_data7 = data_rf3;
training_data8 = data_rf4;
testing_data1 = data_sm1;
testing_data2 = data_sm2;
testing_data3 = data_sm3;
testing_data4 = data_sm4;
testing_data5 = data_sf1;
testing_data6 = data_sf2;
testing_data7 = data_sf3;
testing_data8 = data_sf4;
%%
%% Feature Extraction (MFCCs)
training_features1 = melcepst(training_data1, fs);
training_features2 = melcepst(training_data2, fs);
training_features3 = melcepst(training_data3, fs);
training_features4 = melcepst(training_data4, fs);
training_features5 = melcepst(training_data5, fs);
training_features6 = melcepst(training_data6, fs);
training_features7 = melcepst(training_data7, fs);
training_features8 = melcepst(training_data8, fs);
testing_features1  = melcepst(testing_data1, fs);
```

```
testing_features2  = melcepst(testing_data2, fs);
testing_features3  = melcepst(testing_data3, fs);
testing_features4  = melcepst(testing_data4, fs);
testing_features5  = melcepst(testing_data5, fs);
testing_features6  = melcepst(testing_data6, fs);
testing_features7  = melcepst(testing_data7, fs);
testing_features8  = melcepst(testing_data8, fs);
%% Get some interesting graphs
% subplot(2,1,1),
a = melcepst(training_data1, fs);
% subplot(2,1,2),
% a = melcepst(training_data5, fs);
%% Feature Matching (Minimum-Distance Classifier)
delta = 0.88; % threshold for identification
mean_training_feature1 = mean(training_features1);
mean_training_feature2 = mean(training_features2);
mean_training_feature3 = mean(training_features3);
mean_training_feature4 = mean(training_features4);
mean_training_feature5 = mean(training_features5);
mean_training_feature6 = mean(training_features6);
mean_training_feature7 = mean(training_features7);
mean_training_feature8 = mean(training_features8);
mean_testing_feature1 = mean(testing_features1);
mean_testing_feature2 = mean(testing_features2);
mean_testing_feature3 = mean(testing_features3);
mean_testing_feature4 = mean(testing_features4);
mean_testing_feature5 = mean(testing_features5);
mean_testing_feature6 = mean(testing_features6);
mean_testing_feature7 = mean(testing_features7);
mean_testing_feature8 = mean(testing_features8);
d11 = mean((mean_training_feature1 -
mean_testing_feature1).^2);
d12 = mean((mean_training_feature1 -
mean_testing_feature2).^2);
d13 = mean((mean_training_feature1 -
mean_testing_feature3).^2);
```

```
d14 = mean((mean_training_feature1 -
mean_testing_feature4).^2);
d15 = mean((mean_training_feature1 -
mean_testing_feature5).^2);
d16 = mean((mean_training_feature1 -
mean_testing_feature6).^2);
d17 = mean((mean_training_feature1 -
mean_testing_feature7).^2);
d18 = mean((mean_training_feature1 -
mean_testing_feature8).^2);
d21 = mean((mean_training_feature2 -
mean_testing_feature1).^2);
d22 = mean((mean_training_feature2 -
mean_testing_feature2).^2);
d23 = mean((mean_training_feature2 -
mean_testing_feature3).^2);
d24 = mean((mean_training_feature2 -
mean_testing_feature4).^2);
d25 = mean((mean_training_feature2 -
mean_testing_feature5).^2);
d26 = mean((mean_training_feature2 -
mean_testing_feature6).^2);
d27 = mean((mean_training_feature2 -
mean_testing_feature7).^2);
d28 = mean((mean_training_feature2 -
mean_testing_feature8).^2);
d31 = mean((mean_training_feature3 -
mean_testing_feature1).^2);
d32 = mean((mean_training_feature3 -
mean_testing_feature2).^2);
d33 = mean((mean_training_feature3 -
mean_testing_feature3).^2);
d34 = mean((mean_training_feature3 -
mean_testing_feature4).^2);
d35 = mean((mean_training_feature3 -
mean_testing_feature5).^2);
```

```
d36 = mean((mean_training_feature3 -
mean_testing_feature6).^2);
d37 = mean((mean_training_feature3 -
mean_testing_feature7).^2);
d38 = mean((mean_training_feature3 -
mean_testing_feature8).^2);
d41 = mean((mean_training_feature4 -
mean_testing_feature1).^2);
d42 = mean((mean_training_feature4 -
mean_testing_feature2).^2);
d43 = mean((mean_training_feature4 -
mean_testing_feature3).^2);
d44 = mean((mean_training_feature4 -
mean_testing_feature4).^2);
d45 = mean((mean_training_feature4 -
mean_testing_feature5).^2);
d46 = mean((mean_training_feature4 -
mean_testing_feature6).^2);
d47 = mean((mean_training_feature4 -
mean_testing_feature7).^2);
d48 = mean((mean_training_feature4 -
mean_testing_feature8).^2);
d51 = mean((mean_training_feature5 -
mean_testing_feature1).^2);
d52 = mean((mean_training_feature5 -
mean_testing_feature2).^2);
d53 = mean((mean_training_feature5 -
mean_testing_feature3).^2);
d54 = mean((mean_training_feature5 -
mean_testing_feature4).^2);
d55 = mean((mean_training_feature5 -
mean_testing_feature5).^2);
d56 = mean((mean_training_feature5 -
mean_testing_feature6).^2);
d57 = mean((mean_training_feature5 -
mean_testing_feature7).^2);
```

```
d58 = mean((mean_training_feature5 -
mean_testing_feature8).^2);
d61 = mean((mean_training_feature6 -
mean_testing_feature1).^2);
d62 = mean((mean_training_feature6 -
mean_testing_feature2).^2);
d63 = mean((mean_training_feature6 -
mean_testing_feature3).^2);
d64 = mean((mean_training_feature6 -
mean_testing_feature4).^2);
d65 = mean((mean_training_feature6 -
mean_testing_feature5).^2);
d66 = mean((mean_training_feature6 -
mean_testing_feature6).^2);
d67 = mean((mean_training_feature6 -
mean_testing_feature7).^2);
d68 = mean((mean_training_feature6 -
mean_testing_feature8).^2);
d71 = mean((mean_training_feature7 -
mean_testing_feature1).^2);
d72 = mean((mean_training_feature7 -
mean_testing_feature2).^2);
d73 = mean((mean_training_feature7 -
mean_testing_feature3).^2);
d74 = mean((mean_training_feature7 -
mean_testing_feature4).^2);
d75 = mean((mean_training_feature7 -
mean_testing_feature5).^2);
d76 = mean((mean_training_feature7 -
mean_testing_feature6).^2);
d77 = mean((mean_training_feature7 -
mean_testing_feature7).^2);
d78 = mean((mean_training_feature7 -
mean_testing_feature8).^2);
d81 = mean((mean_training_feature8 -
mean_testing_feature1).^2);
```

```
d82 = mean((mean_training_feature8 -
mean_testing_feature2).^2);
d83 = mean((mean_training_feature8 -
mean_testing_feature3).^2);
d84 = mean((mean_training_feature8 -
mean_testing_feature4).^2);
d85 = mean((mean_training_feature8 -
mean_testing_feature5).^2);
d86 = mean((mean_training_feature8 -
mean_testing_feature6).^2);
d87 = mean((mean_training_feature8 -
mean_testing_feature7).^2);
d88 = mean((mean_training_feature8 -
mean_testing_feature8).^2);
% check_identified(d11, delta, 'd11'); % false negative ;
positive
% check_identified(d12, delta, 'd12'); % negative       ;
negative
% check_identified(d13, delta, 'd13'); % negative       ;
negative
% check_identified(d21, delta, 'd21'); % negative       ; false
positive
% check_identified(d22, delta, 'd22'); % positive       ; positive
% check_identified(d23, delta, 'd23'); % false positive ;
negative
% check_identified(d31, delta, 'd31'); % negative       ; false
positive (negative w delta .88)
% check_identified(d32, delta, 'd32'); % false positive ;
negative
% check_identified(d33, delta, 'd33'); % positive       ; positive
% 2/3 correctly identified            ; 3/3 correctly identified
% 1 false negative, 2 false positives ; 2 false positives
check_identified(d11, delta, 'd11');
check_identified(d12, delta, 'd12');
check_identified(d13, delta, 'd13');
check_identified(d14, delta, 'd14');
```

```
check_identified(d15, delta, 'd15');
check_identified(d16, delta, 'd16');
check_identified(d17, delta, 'd17');
check_identified(d18, delta, 'd18');
check_identified(d21, delta, 'd21');
check_identified(d22, delta, 'd22');
check_identified(d23, delta, 'd23');
check_identified(d24, delta, 'd24');
check_identified(d25, delta, 'd25');
check_identified(d26, delta, 'd26');
check_identified(d27, delta, 'd27');
check_identified(d28, delta, 'd28');
check_identified(d31, delta, 'd31');
check_identified(d32, delta, 'd32');
check_identified(d33, delta, 'd33');
check_identified(d34, delta, 'd34');
check_identified(d35, delta, 'd35');
check_identified(d36, delta, 'd36');
check_identified(d37, delta, 'd37');
check_identified(d38, delta, 'd38');
check_identified(d41, delta, 'd41');
check_identified(d42, delta, 'd42');
check_identified(d43, delta, 'd43');
check_identified(d44, delta, 'd44');
check_identified(d45, delta, 'd45');
check_identified(d46, delta, 'd46');
check_identified(d47, delta, 'd47');
check_identified(d48, delta, 'd48');
check_identified(d51, delta, 'd51');
check_identified(d52, delta, 'd52');
check_identified(d53, delta, 'd53');
check_identified(d54, delta, 'd54');
check_identified(d55, delta, 'd55');
check_identified(d56, delta, 'd56');
check_identified(d57, delta, 'd57');
check_identified(d58, delta, 'd58');
```

```
check_identified(d61, delta, 'd61');
check_identified(d62, delta, 'd62');
check_identified(d63, delta, 'd63');
check_identified(d64, delta, 'd64');
check_identified(d65, delta, 'd65');
check_identified(d66, delta, 'd66');
check_identified(d67, delta, 'd67');
check_identified(d68, delta, 'd68');
check_identified(d71, delta, 'd71');
check_identified(d72, delta, 'd72');
check_identified(d73, delta, 'd73');
check_identified(d74, delta, 'd74');
check_identified(d75, delta, 'd75');
check_identified(d76, delta, 'd76');
check_identified(d77, delta, 'd77');
check_identified(d78, delta, 'd78');
check_identified(d81, delta, 'd81');
check_identified(d82, delta, 'd82');
check_identified(d83, delta, 'd83');
check_identified(d84, delta, 'd84');
check_identified(d85, delta, 'd85');
check_identified(d86, delta, 'd86');
check_identified(d87, delta, 'd87');
check_identified(d88, delta, 'd88');
%% Feature Matching (Gaussian Mixture Model)
No_of_Clusters = 2;
No_of_Iterations = 10;
[training_idx1, training_mu1, training_sigma1] =
GMM(training_features1', No_of_Clusters, No_of_Iterations);
[training_idx2, training_mu2, training_sigma2] =
GMM(training_features2', No_of_Clusters, No_of_Iterations);
[training_idx3, training_mu3, training_sigma3] =
GMM(training_features3', No_of_Clusters, No_of_Iterations);
%%
size(training_features1), size(training_features2),
size(training_features3)
```

```
size(testing_features1), size(testing_features2),
size(testing_features3)
size(training_mu1), size(training_mu2), size(training_mu3)
[pc11, idx11] = Cluster_Probability(testing_features1',
training_mu3)
% [pc12, idx12] = Cluster_Probability(testing_features2,
training_mu1)
% [pc13, idx13] = Cluster_Probability(testing_features3,
training_mu1)
%
% [pc21, idx21] = Cluster_Probability(testing_features1,
training_mu2)
% [pc22, idx22] = Cluster_Probability(testing_features2,
training_mu2)
% [pc23, idx23] = Cluster_Probability(testing_features3,
training_mu2)
%
% [pc31, idx31] = Cluster_Probability(testing_features1,
training_mu3)
% [pc32, idx32] = Cluster_Probability(testing_features2,
training_mu3)
% [pc33, idx33] = Cluster_Probability(testing_features3,
training_mu3)
pc11
Cluster_Probability(testing_features1', training_mu1)
Cluster_Probability(testing_features1', training_mu2)
log(Cluster_Probability(testing_features1', training_mu1)) -
log(Cluster_Probability(testing_features1', background_mu))
```

%% Data Input

```
fs = 8000;
training_data1 = wavread('01_train.wav');
training_data2 = wavread('02_train.wav');
training_data3 = wavread('03_train.wav');
testing_data1  = wavread('01_test.wav');
testing_data2  = wavread('02_test.wav');
testing_data3  = wavread('03_test.wav');
```

```
%% Feature Extraction (MFCCs)
training_features1 = melcepst(training_data1, fs);
training_features2 = melcepst(training_data2, fs);
training_features3 = melcepst(training_data3, fs);
testing_features1  = melcepst(testing_data1, fs);
testing_features2  = melcepst(testing_data2, fs);
testing_features3  = melcepst(testing_data3, fs);
%% Feature Matching (Minimum-Distance Classifier)
delta = 0.85; % threshold for identification
mean_training_feature1 = mean(training_features1);
mean_training_feature2 = mean(training_features2);
mean_training_feature3 = mean(training_features3);
mean_testing_feature1 = mean(testing_features1);
mean_testing_feature2 = mean(testing_features2);
mean_testing_feature3 = mean(testing_features3);
d11 = mean((mean_training_feature1 -
mean_testing_feature1).^2);
d12 = mean((mean_training_feature1 -
mean_testing_feature2).^2);
d13 = mean((mean_training_feature1 -
mean_testing_feature3).^2);
d21 = mean((mean_training_feature2 -
mean_testing_feature1).^2);
d22 = mean((mean_training_feature2 -
mean_testing_feature2).^2);
d23 = mean((mean_training_feature2 -
mean_testing_feature3).^2);
d31 = mean((mean_training_feature3 -
mean_testing_feature1).^2);
d32 = mean((mean_training_feature3 -
mean_testing_feature2).^2);
d33 = mean((mean_training_feature3 -
mean_testing_feature3).^2);
check_identified(d11, delta, 'd11'); % false negative
check_identified(d12, delta, 'd12'); % negative
check_identified(d13, delta, 'd13'); % negative
```

```
check_identified(d21, delta, 'd21'); % negative
check_identified(d22, delta, 'd22'); % positive
check_identified(d23, delta, 'd23'); % false positive
check_identified(d31, delta, 'd31'); % negative
check_identified(d32, delta, 'd32'); % false positive
check_identified(d33, delta, 'd33'); % positive
% 2/3 correctly identified
% 1 false negative, 2 false positives
%% Feature Matching (Gaussian Mixture Model)
No_of_Clusters = 2;
No_of_Iterations = 10;
[training_idx1, training_mu1, training_sigma1] =
GMM(training_features1', No_of_Clusters, No_of_Iterations);
[training_idx2, training_mu2, training_sigma2] =
GMM(training_features2', No_of_Clusters, No_of_Iterations);
[training_idx3, training_mu3, training_sigma3] =
GMM(training_features3', No_of_Clusters, No_of_Iterations);
%%
size(training_features1), size(training_features2),
size(training_features3)
size(testing_features1), size(testing_features2),
size(testing_features3)
size(training_mu1), size(training_mu2), size(training_mu3)
[pc11, idx11] = Cluster_Probability(testing_features1',
training_mu3)
% [pc12, idx12] = Cluster_Probability(testing_features2,
training_mu1)
% [pc13, idx13] = Cluster_Probability(testing_features3,
training_mu1)
%
% [pc21, idx21] = Cluster_Probability(testing_features1,
training_mu2)
% [pc22, idx22] = Cluster_Probability(testing_features2,
training_mu2)
% [pc23, idx23] = Cluster_Probability(testing_features3,
training_mu2)
```

```
%
% [pc31, idx31] = Cluster_Probability(testing_features1,
training_mu3)
% [pc32, idx32] = Cluster_Probability(testing_features2,
training_mu3)
% [pc33, idx33] = Cluster_Probability(testing_features3,
training_mu3)
pc11
Cluster_Probability(testing_features1', training_mu1)
Cluster_Probability(testing_features1', training_mu2)
log(Cluster_Probability(testing_features1', training_mu1)) -
log(Cluster_Probability(testing_features1', background_mu))
```

Приложение Б