МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
Национальный исследовательский ядерный университет «МИФИ»

Институт
интеллектуальных кибернетических систем

Кафедра № 22 «Кибернетика»

Направление подготовки 01.03.02 Прикладная математика и информатика

Пояснительная записка

к учебно-исследовательской работе студента на тему:

Разработка системы голосовой идентификации

Группа Б17-501

Студент _____ (подпись) Баранова Д.Д. (ФИО)

Руководитель _____ (0-5 баллов) _____ (подпись) Козин Р.Г. (ФИО)

Научный консультант _____ (0-5 баллов) _____ (подпись) _____ (ФИО)

Оценка руководителя _____ (0-15 баллов) Оценка консультанта _____ (0-15 баллов)

Итоговая оценка _____ (0-100 баллов) ECTS _____

Комиссия

Председатель _____ (подпись) _____ (ФИО)

_____ (подпись) _____ (ФИО)

_____ (подпись) _____ (ФИО)

_____ (подпись) _____ (ФИО)

Москва 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

КАФЕДРА КИБЕРНЕТИКИ

Задание на УИР

Студенту гр.	Б17-501		Барановой Д.Д.
	(группа)		(фио)

ТЕМА УИР

Разработка системы голосовой идентификации

ЗАДАНИЕ

№ п/п	Содержание работы	Форма отчетности	Срок исполнения	Отметка о выполнении Дата, подпись рук.
1	**Аналитическая часть**			
1.1	Обзор и сравнение форм биометрической идентификации	Подраздел ПЗ	15.02.2021	
1.2	Обзор методов и существующих подходов для голосовой идентификации	Подраздел ПЗ	10.03.2021	
1.3	Обзор существующих сервисов и областей применения	Подраздел ПЗ	15.03.2021	
2	**Теоретическая часть**			
2.1	Постановка задачи голосовой идентификации	Подраздел ПЗ	10.03.2021	
2.2	Описание способа представления голосов	Подраздел ПЗ	13.03.2021	
2.3	Описание использующегося метода идентификации	Описание метода	15.03.2021	
2.4	Описание способа сравнения голосов	Подраздел ПЗ	17.03.2021	
2.5	Описание используемых метрик качества	Подраздел ПЗ	20.03.2021	
3	**Инженерная часть**			
3.1	Выбор языка и вспомогательных модулей для программной реализации системы	Выбранный язык	23.03.2021	
3.2	Программная реализация моделей и алгоритмов	Исходный код	25.03.2021	
4	**Технологическая и практическая часть**			
4.1	Описание тестовых данных, объема и особенностей	Готовый датасет	01.04.2021	
4.2	Тестирование работы системы идентификации	Подраздел ПЗ	05.04.2021	
	Оформление пояснительной записки (ПЗ) и иллюстративного материала для доклада.	Текст ПЗ, презентация	15.04.2021	

ЛИТЕРАТУРА

•	Zhongxin Bai, Xiao-Lei Zhang, Jingdong Chen. Speaker Recognition Based on Deep Learning: An Overview. 2020. URL: https://arxiv.org/abs/2012.00931
•	Kumar G.Suvarna, Raju, K.A.Prasad, Dr.Mohan Rao CPVNJ, P.Satheesh. Speaker recognition using GMM
•	Hajavi, A., Etemad A. 2019. A Deep Neural Network for ShortSegment Speaker Recognition. URL: https://arxiv.org/pdf/1907.10420.pdf
•	Joon Son Chung, Arsha Nagrani, Andrew Zisserman. 2018. VoxCeleb2: Deep Speaker Recognition. URL: https://arxiv.org/abs/1806.05622
•	Daniel Foley. Gaussian Mixture Modelling (GMM). URL: https://towardsdatascience.com/gaussian-mixture-modelling-gmm-833c88587c7f
•	И. А. Рахманенко, Р. В. Мещеряков. Анализ идентификационных признаков в речевых данных с помощью GMM-UBM системы верификации диктора. URL: https://www.elibrary.ru/item.asp?id=29296333

Дата выдачи задания:		Руководитель			Козин Р.Г.
					(ФИО)
« 1 » февраля 2021 г.		Студент			Баранова Д.Д.
					(ФИО)

Реферат

Пояснительная записка содержит 41 страницу, 11 рисунков, 3 таблицы, 8 формул.

Количество использованных источников – 26.

Ключевые слова: голосовая идентификация, голосовые признаки, модель гауссовых смесей.

Целью данной работы является разработка системы для автоматической идентификации личности говорящего по голосу.

В работе рассматриваются различные существующие подходы к решению задачи идентификации по голосу, классификация разных типов биометрической идентификации, использующиеся методы машинного обучения для идентификации, существующие готовые системы и решения на рынке, востребованность системы голосовой идентификации на рынке биометрических систем идентификации. Затем в работе приводится описание разработанного метода идентификации говорящего, этапы обработки аудиофайла и способа представления голоса в виде вектора признаков. Также в работе дается описание разработки системы, выбор вспомогательных библиотек и модулей для разработки, описание реализованных функций. Приводятся результаты экспериментальных исследований работы реализованной системы идентификации диктора, оценивается влияние архитектуры модели на качество работы. После этого подводятся итоги проведённой работы с кратким описанием результатов по каждому разделу.

Содержание

Введение

Голос говорящего содержит в себе его личные черты, обусловленные уникальным строением органов произношения, например, уникальный голосовой формой тракта, размером гортани, особенностями дикции, акцентом и ритмом [7], индивидуальной манерой речи говорящего. Таким образом, можно автоматически идентифицировать говорящего по его голосу через компьютер.

Идентификация говорящего - это фундаментальная задача обработки речи и голоса, которая находит широкое применение в реальных задачах. Например, она используется для подтверждения личности на горячих линиях и при звонках в call-центры, при голосовой аутентификации в личных интеллектуальных устройствах, таких как сотовые телефоны, автомобили и ноутбуки. Такая идентификация не только гарантирует безопасность транзакций банковской торговли и удаленных платежей, но также автоматизирует ряд задач, ведет к сокращению времени и росту лояльности. Биометрическая идентификация на основе голоса также может широко применяться в судебной практике для расследования подозреваемого на предмет признания его виновным, для наблюдения и автоматической идентификации личности на расстоянии (возможно, без ведома самой личности). Она также может выступать частью задачи автоматического определения количества говорящих в записанном аудиофрагменте и сегментации фрагмента на части, в которых звучит голос определенного диктора.

Несмотря на то, что были достигнуты многие технологические успехи в области биометрической идентификации, остается еще много исследовательских проблем, которые необходимо решить. Разработка системы для идентификации говорящего позволит автоматизировать и ускорить решение многих повседневных задач.

Целью данной работы является построение системы, выполняющей автоматическую идентификацию диктора по его речи в аудиосигнале. В рамках работы решаются следующие задачи:

- Получение обучающей выборки.
- Изучение и разработка метода идентификации диктора.
- Проектирование и разработка оптимальной архитектуры выбранной модели.
- Тестирование работы реализованной системы.

В первом разделе приводится описание и сравнительный анализ различных существующих подходов к идентификации диктора, описание возможных решений, их достоинств и недостатков.

Во втором разделе ставится задача идентификации диктора, выделяются подзадачи, приводится описание метода идентификации и исследование метрик качества разработанной модели.

В третьем разделе приводится список выбранных вспомогательных модулей, а также указывается выбранный язык и среда программирования. Описываются реализованные функций и элементы системы.

В четвёртом разделе приводятся результаты экспериментальных исследований влияния параметров архитектуры выбранной модели на качество работы, приводятся результаты тестирования системы идентификации в результате чего определяется оптимальный метод решения поставленной задачи.

В заключении подводятся итоги проведённой работы с кратким описанием результатов по каждому разделу.

Раздел 1. Анализ существующих подходов к голосовой идентификации

Сейчас существует несколько платных закрытых систем голосовой биометрической идентификации, спроектированных на основе использования различных методов машинного обучения. Качество работы систем не идеально и продолжает совершенствоваться в настоящее время. Интерес к задаче растет.

Несмотря на существование готовых решений, все системы остаются закрыты, их методы работы нигде не описаны и сохраняются в тайне. Описанные в разделе существующие методы решения задачи идентификации диктора по его речи представляют собой лишь примерное описана некоторых существующих решений без деталей реализации и описания недостатков выбранных методов.

1.1 Обзор и сравнение форм биометрической идентификации

Биометрия – это наука, основанная на измерении и описании физических характеристик живых существ. В рамках задач автоматической идентификации личности биометрические данные - это уникальные биологические и физиологические характеристики, которые позволяют установить личность человека. Биометрические данные можно разделить на:

- физиологические (относящиеся к физическому телу – отпечатки пальцев, ладонь руки, радужная оболочка, голос, ДНК и т.д.)
- поведенческие (относящиеся к поведению человека – походка, речь, почерк и т.д.).

Рассматривая динамику развития мирового рынка биометрических технологий, можно увидеть значительный стабильный рост годового дохода рынка от биометрии во всех регионах планеты (рисунок ниже). Исходя из этого можно сделать вывод, что использование биометрических технологий становится все более популярным, а их решения остаются востребованными на рынке.

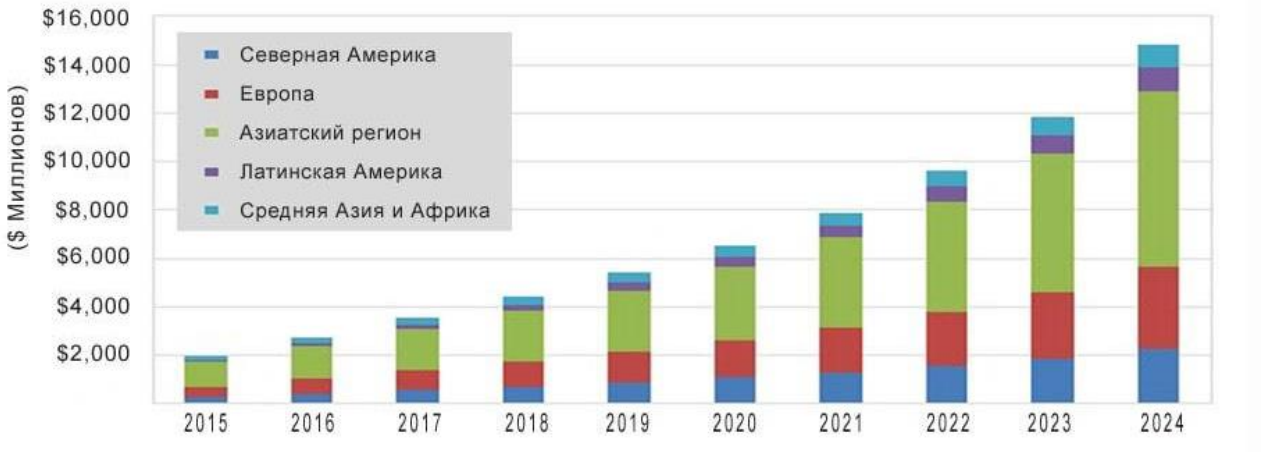

Рис.1 Годовой доход от биометрии по регионам. Прогноз. Мировой рынок: 2015-2024

Среди самых распространенных типов физиологической биометрии – изображение лица человека, отпечатки пальца, рисунок радужной оболочки глаза, голос, рисунки вен на ладони и т.д.

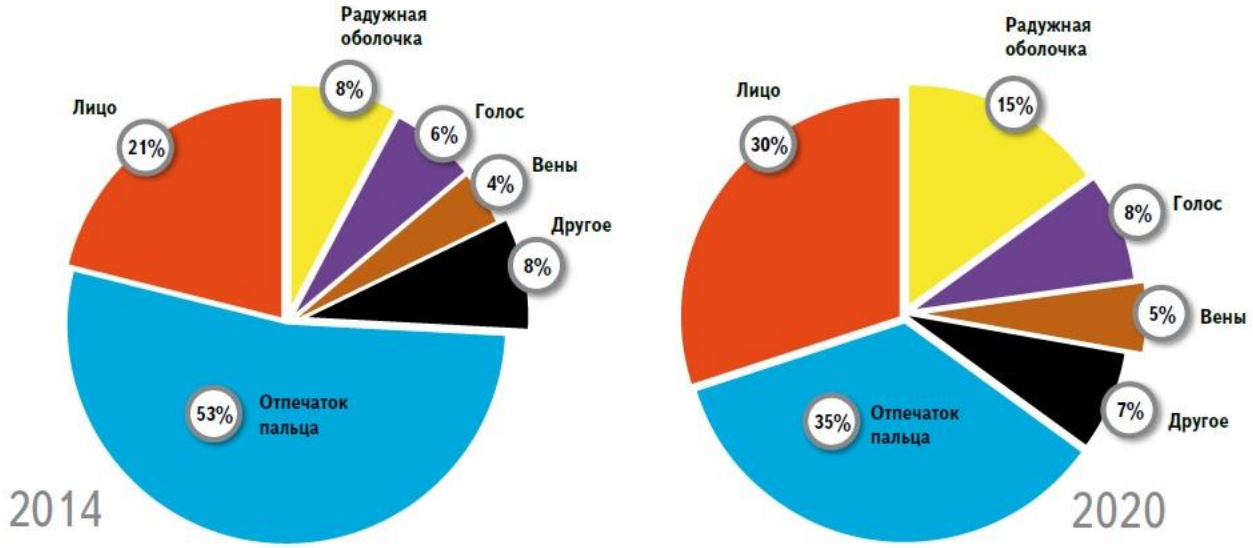

Рис.2 Мировой рынок биометрических технологий. Распределение форм идентификации

Можно наблюдать [8] преобладание на рынке использования технологии идентификации по отпечаткам пальца, что понятно, так как по надежности работы именно она занимает лидирующее место. Однако такой метод идентификации является контактным, что влечет некоторые ограничения, в том числе в условиях пандемии. Также можно заметить, что доля рынка, использующая отпечатки пальца как метод идентификации, постепенно уменьшается, уступая другим методам.

Таблица 1. Значения ошибок идентификации для различных биометрических модальностей [9]

Сравнительная характеристика биометрических систем	Отпечаток пальца	Голос	Радужная оболочка	Лицо
Вероятность «допуска чужого»	2,2%	1.0%	0.1%	0.1%
Вероятность «отказа своему»	2,2%	до 4.0%	1.1-1.4%	до 4.0%
Стоимость системы	Высокая	Низкая	Очень высокая	Высокая

Однако темпы развития технологий голосовой идентификации достаточно велики. По прогнозам ВТБ среднегодовой темп роста рынка голосовых биометрических систем до 2022 года достигнут более 21% и будут занимать второе место после технологии, построенной на использовании радужной оболочки глаза (рисунок ниже). Развитие голосовых технологий может объясняться низкой стоимостью системы по сравнению с другими видами биометрических технологий.

Рис.3 Прогноз среднегодового темпа роста рынка биометрических систем в разрезе технологий до 2022 года

Таким образом, технологии на основе голосовой биометрии сейчас не являются лидерами по востребованности на рынке, однако они быстро развиваются и имеют ряд преимуществ перед другими биометрическими технологиями, что делает их потенциальными лидерами в будущем.

1.2 Обзор методов и существующих подходов для голосовой идентификации

Существует два типа систем идентификации говорящего:

- Текстозависимая. Если произнесенный текст должен быть одинаковым для регистрации диктора в базе и для идентификации, то этот подход называется текстозависимым распознаванием. В такой системе произносимое ключевое слово/фраза могут быть либо общими для всех дикторов, либо уникальными (например, уникальный пароль пользователя).
- Текстонезависимая. Независимые от текста системы чаще используются для идентификации диктора, поскольку они требуют меньшего внимания со стороны говорящего. В этом случае текст при регистрации и идентификации отличается, система считается инвариантна к речи.

Задачу идентификации человека по голосу, независимо от выбранного пути ее решения, можно разбить на следующие основные этапы:

1) Извлечение признаков из аудио сигнала

2) Построение модели диктора, на основе полученных на предыдущем шаге признаков

Процесс определения человека по голосу из представленных в системе образцов голосов, во всех методах состоит в поиске более подходящего по модели диктора, на основании выбранных критериев.

Речевые сигналы - это звуковые колебания, которые распространяются в воздушной среде. Они характеризуются частотой (числом колебаний в секунду), интенсивностью (амплитудой колебаний) и длительностью. На протяжении всего речевого сигнала эти характеристики подвергаются изменениям, и фиксируются с помощью электронно-акустических приборов, таких осциллограф, спектрограф. Затем, при помощи аналого-цифрового преобразователя, аналоговый речевой сигнал переводится в цифровой, который на ЭВМ представлен в виде WAV-файла, с которого уже происходит сбор речевых признаков.

1.2.1 Выбор системы признаков

Характерные особенности голоса человека формируется на основе размера и формы речеобразующего тракта, динамикой его изменения, упругостью и т.д. Основными и самыми популярными характерными параметрами для описания голоса и речи человека, учитывающими описанные особенности, являются MFCC. Практически во всех исследованиях работа ведется с ними, а также иногда дополняется набором из первых и

вторых дельта-функций – локальных оценок производных (в дискретном случае производная функции равна разности значений функции в последовательных точках).

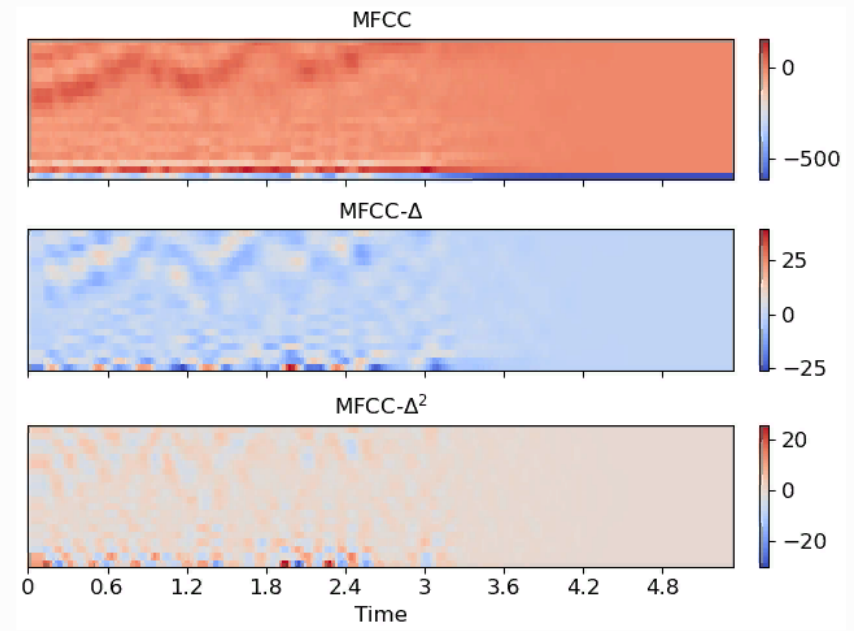

Рис.4 Визуализация MFCC и дельта-функций первого, второго порядков

1.2.2 Выбор классификатора признаков

Современные системы голосовой идентификации и верификации работают в двух режимах.

- Режим обучения. Выделяются характерные признаки голоса человека, формируется его голосовая модель (голосовой отпечаток) на основе этих признаков и выполняется сохранение модели в базе данных.
- Рабочий режим. Выделяются характерные признаки голосового сигнала человека и выполняется поиск в базе данных голосовой модели, соответствующей этим признакам (идентификация личности).

1.2.2.1 GMM

В настоящее время наиболее результативным подходом к решению задач текстонезависимой идентификации личности является построение голосовых моделей на основе моделей гауссовых смесей (Gaussian Mixture Model, GMM) [10, 11]. Сами модели строятся на основе некоторого набора голосовых признаков. Наиболее распространенным методом построения голосовых признаков является формирование вектора мел-частотных кепстральных коэффициентов (Mel-Frequency Cepstral Coefficient, MFCC) из голосовой записи [10, 12].

Однако, несмотря на достаточно хорошие результаты работы в «лабораторных условиях», метод GMM-MFCC не может быть использован для построения реальных систем

голосовой идентификации, так как система имеет сильную зависимость результатов от вида обучающего материала (на основе которого составляется база голосовых моделей и фоновая модель), и условий записи голосового сигнала. Также недостатком являются относительно большая потребность в большом количестве обучающего материала.

1.2.2.2 Сеть Кохонена

В качестве альтернативного способа сравнения голосовых отпечатков может использоваться нейросетевое сравнение при помощи самоорганизующейся сети Кохонена. Сеть обучается без учителя и также имеет две фазы работы – обучение и идентификацию. В процессе обучения обучающий алгоритм подстраивает веса сети так, чтобы получились согласованные выходные векторы. Таким образом, процесс обучения выделяет статистические свойства обучающего множества и группирует сходные векторы в классы. Предъявление на вход вектора из данного класса дает определенный выход. В ходе идентификации будет происходить вычисление сумм в узлах сети после подачи вектора на вход, затем активация найденного узла и получение классифицируемого вектора.

1.2.2.3 Глубокие нейронные сети

Кроме сети Кохонена существующие работы [13] основаны на использовании в качестве классификатора глубокой нейронной сети и нейронной сети со сверточным слоем [14]. Она имеет несколько скрытых слоев, входной слой того же размера, что и вектор признаков, и выходной слой с количеством нейроном, равным количеству зарегистрированных дикторов в системе. Несмотря на различные способы оптимизации способа обучения сети, такой подход делает невозможным быстрое добавление новых дикторов в систему. Для этого необходимо не только заново обучить всю сеть, но и изменить её архитектуру (как минимум – увеличить выходной слой).

1.3 Обзор существующих сервисов и областей применения

Идентификация по голосу всё чаще используется в системах безопасности банков, телекомов и других организаций в виде компонента аутентификации, которая выступает дополнительным фактором парольной защиты. Пользователь должен произнести по телефону пароль, при этом система определяет правильность парольной фразы и дополнительно проверяет уникальный голосовой отпечаток данного пользователя.

Ярким примером является «Сбербанк России» («Сбер»), который внедрил в свою систему возможность записи голоса (сбор голосов начался еще в 2018 году с целью создания Единой биометрической системы (ЕБС)[15] и Единой системы идентификации и аутентификации (ЕСИА)) и возможность голосового подтверждения операций (технология внедрена с 21 января 2021 года [16]). Благодаря новой технологии организация надеется защитить клиентов от мошенничества. Также компания начала предлагать клиентам на

добровольной основе сдать образец голоса через мобильное приложение «Сбербанк онлайн» или записать голосовой образец при звонке в call-центр. Предварительно нужно дать согласие на обработку персональных данных.

Созданная ЕСИА сейчас встроена в систему авторизации пользователей на портале Госуслуг и обеспечивает доступ граждан по его желанию и с его согласия к порталам Госуслуг. После сдачи биометрических данных любой житель страны может пользоваться услугами банков и государственных учреждений без личного присутствия. Биометрическую базу можно будет применять для дистанционной сдачи экзаменов, для прохождения транспортного контроля в аэропортах, для подтверждения права бесплатного проезда на общественном транспорте или оплаты проезда [17].

Из крупнейших решений на мировом рынке голосовой биометрии можно выделить следующих [18]:

- Agnitio (Испания)
- Auraya Systems (Австралия)
- Authentify (ОАЭ)
- KeyLemon (Швейцария)
- Nuance (США)
- ValidSoft (Великобритания)
- Verint Systems (США)
- VoiceTrust (Германия)
- VoiceVault (Великобритания)

Лидером рынка сейчас признается Nuance. Ее решения установлены в Аэрофлоте, а также она известна всем пользователям iPhone, так как Siri построена именно на технологиях Nuance.

1.4 Выводы

Задача идентификации говорящего по голосу является в настоящее время очень актуальной и продолжает набирать популярность. Рынок нуждается в новых более качественных решениях. Все существующие системы идентификации диктора закрыты и не предоставляют описания принципа своей работы.

Наиболее интересной и практически ценной является задача текстонезависимой идентификации. Также система должна иметь возможность быстро добавлять нового диктора в свою базу (без изменения архитектуры всей системы, которое может повлечь ухудшение качества или требовать большое время работы) и не требовать большого объема обучающих данных (нельзя требовать с нового диктора несколько часов его записанной речи).

Раздел 2. Разработка оптимального метода голосовой идентификации

В разделе ставится задача идентификации по голосу и описывается метод идентификации и использующиеся метрики качества идентификации. Задача идентификации разбивается на подзадачи, выделяются этапы работы системы идентификации. Приводятся способы решения поставленных подзадач.

2.1 Постановка задачи голосовой идентификации

Поставим задачу идентификации по голосу. Необходимо создать систему идентификации личности диктора по его голосу в полученной аудиозаписи. Система должна иметь возможность сохранять голос диктора в своей базе и выполнять идентификацию по запросу и получению новой аудиозаписи голоса. Допускается, что идентификация будет происходить по закрытому множеству сохраненных голосов – в этом случае ответ всегда будет существовать и диктор всегда будет найден. Также необходимо, чтобы на ответ системы влиял только голос диктора и никак не влиял произнесенный на аудиозаписи текст – система будет являться текстонезависимой.

Всю задачу идентификации можно разбить на несколько этапов:

- Предобработка сигнала
- Выделение признаков
- Распознавание диктора

После записи голоса средствами записи и получения его на вход системы голос необходимо предварительно обработать. В этот этап входит чистка - удаление посторонних шумов, сетевого гула, выделение главного и удаление других голосов, удаление пауз, усиление высоких частот и нормализация сигнала.

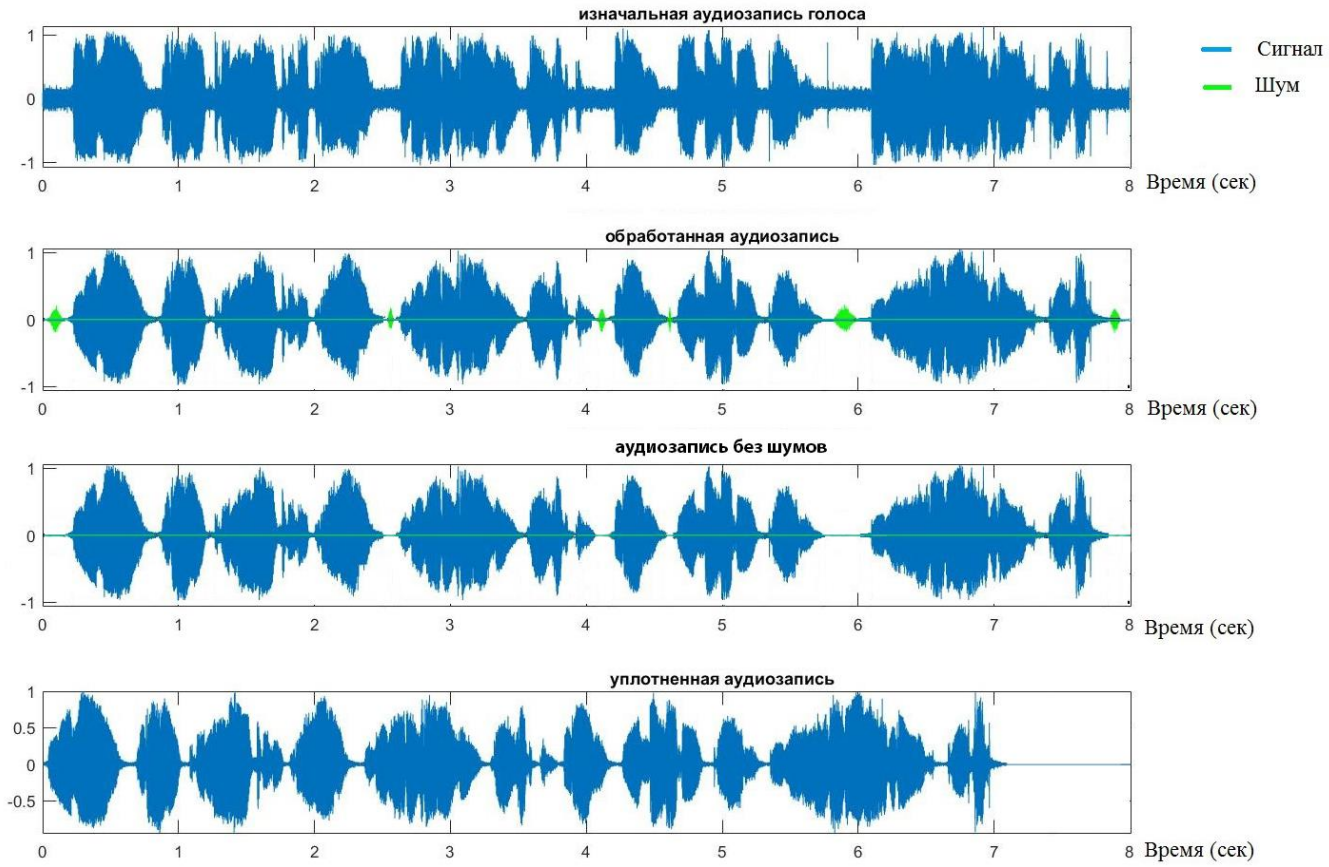

Рис.5 Визуализация этапов предобработки аудиосигнала

Затем запись необходимо представить в виде многомерного вектора признаков, на основе которого будет уже выполняться распознавание диктора.

В данной работе основное внимание будет уделяться последним двум этапам, первый этап предобработки будет рассматриваться вскользь, будут выполнены лишь ограниченное количество необходимых преобразований.

Таким образом, задача голосовой идентификации состоит в идентификации диктора по полученному аудиофайлу с его речью без учета произнесенного текста (в пассивном режиме). При этом задача является закрытой – диктор всегда будет выбираться из дикторов, зарегистрированных в системе.

2.2 Описание способа представления голосов

Звуковой сигнал представляет собой упорядоченный массив значений амплитуды звука, к которому добавлен заголовок, содержащий в себе количество каналов, частоту дискретизации и другую информацию. Однако анализировать данные в таком виде не представляется возможным – они не содержат ключевых признаков, на основе которых метод сможет дать хороший результат.

Одним из способов представления аудиоданных и извлечения из них признаков является получение мел-частотных кепстральных коэффициентов (Mel Frequency Cepstral Coefficients).

Исходный сигнал нарезают на перекрывающиеся фреймы (окна) небольшой длины (10-40 мс). Затем к получившимся фреймам применяют окно Хемминга (чтобы сгладить значения на границах фреймов и уменьшить утечку спектра), делают быстрое преобразование Фурье (FFT):

$$X[k] = \sum_{n=0}^{N-1} x[n] * e^{-2*\pi*i*k*n/N}, 0 \leq k < N \quad (1)$$

и получают спектральную плотность мощности (распределение мощности сигнала в зависимости от частоты, то есть мощность, приходящаяся на единичный интервал частоты).

Затем специальной «гребёнкой» фильтров, расположенных равномерно по мел-шкале (рисунок ниже) делают мел-спектрограмму (каждый мел-фильтр - это треугольная оконная функция, которая суммирует количество энергии на определённом диапазоне частот и тем самым дает мел-коэффициент), после чего логарифмируют полученные результаты (считается, что таким образом понижается чувствительность коэффициентов к шумам) и применяют дискретное косинусное преобразование (DCT) —алгоритм сжатия данных.

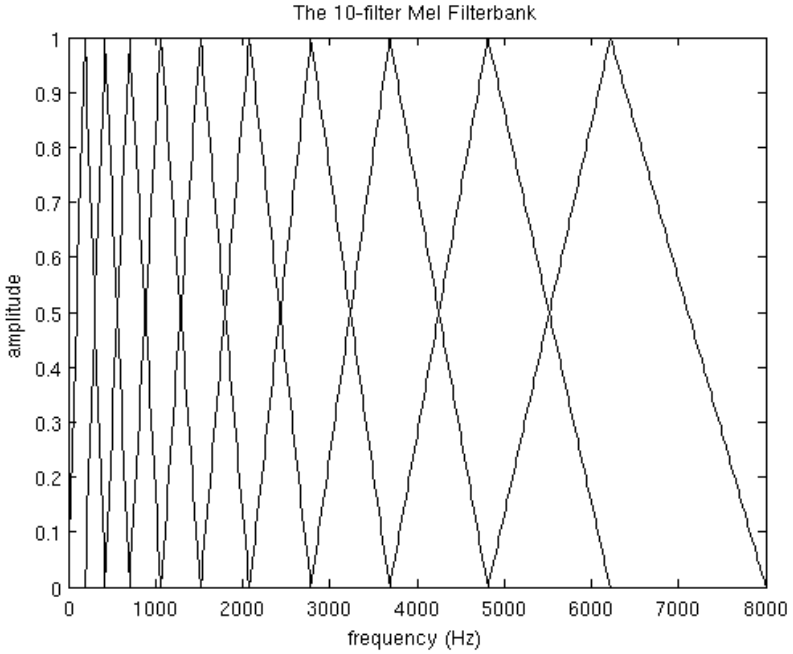

Рис.6 Десять мел-фильтров - треугольных оконных функций

Полученные таким образом коэффициенты представляют из себя некую сжатую характеристику фрейма. При этом, поскольку фильтры, которые мы применяли, были расположены в мел-шкале, коэффициенты учитывают особенности восприятия

человеческого уха, а значит несут больше полезной информации. Принято использовать от 13 до 25 MFCC на фрейм.

Так как индивидуальность голоса часто зависит от скорости речи, особенностей произношения и ускорения, будем также учитывать производные.

2.3 Описание использующегося метода идентификации

На данном этапе необходимо, имея новый набор признаков неизвестного диктора и множество сохраненных в базе наборов известных дикторов, отнести новые данные к верной категории - диктору. Сформулированная задача будет решаться через построение оптимальных для каждого диктора моделей гауссовской смеси распределений.

Модель гауссовой смеси распределений (Gaussian Mixture Model) - это функция, состоящая из нескольких гауссианов, каждый из которых идентифицируется как $k \in \{1, \dots, K\}$, где K - количество кластеров в нашем наборе данных [19]. Другими словами, Модель Гауссовой смеси (GMM) представляет собой смесь многомерных гауссовых распределений вероятностей (многомерных нормальных распределений), которые наилучшим образом моделируют входной набор данных.

Хотя GMM часто классифицируется как алгоритм кластеризации, в нашем случае она будет представлять оценку плотности. Другими словами, результатом подгонки GMM к данным является технически не кластерная модель, а вероятностная модель, описывающая распределение данных. На рисунке ниже данные представлены в виде точек, которые разбиты на кластеры. Точки каждого кластера окрашены своим цветом. Голубые окружности показывают смоделированное GMM распределение входных данных. Показаны результаты работы для двух типов ковариации - covariance_type="spherical" (слева) и covariance_type="full" (справа) [20].

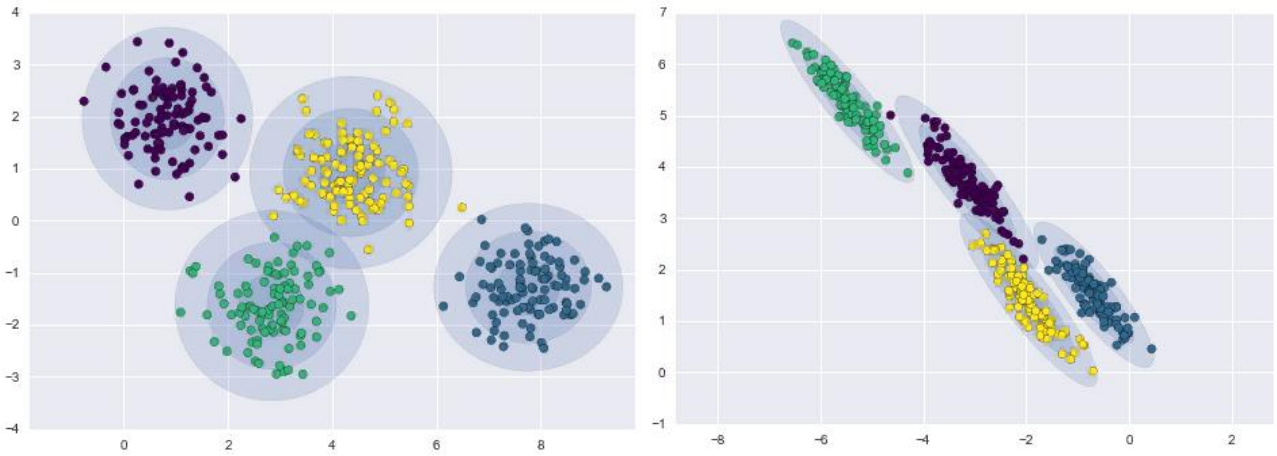

Рис.7 Визуальное представление компонент GMM в зависимости от данных и типа ковариации

Важно выбрать достаточное количество компонент GMM, иначе она будет давать плохой результат (рисунок ниже).

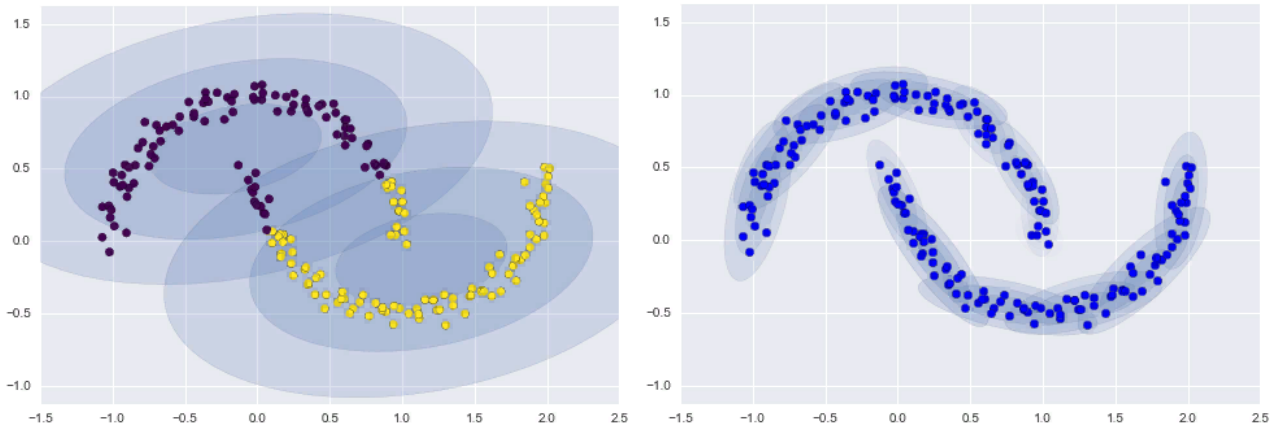

Рис.8 Плохое описание данных (слева – 2 компоненты) и хорошее (справа – 16)

2.4 Описание способа сравнения голосов

GMM удобна, так как представляет собой гибкое средство моделирования произвольного многомерного распределения данных. После этого моделирования будем сохранять все полученные модели GMM для каждого диктора.

Получая на вход новый набор признаков, который необходимо отнести к верной категории – диктору, будем для каждого сохраненной модели GMM диктора определять вероятностные значения присутствия среди данного распределения записанного вектора признаков. Другими словами, будем сравнивать результат от каждой сохраненной GMM. Верной категорией будет признан диктор с максимальным значением вероятности.

2.5 Описание используемых метрик качества

Допустим, необходимо определить, принадлежит ли образец речи Y диктору S (из набора сохраненных дикторов). Сформулируем необходимые гипотезы.

H_0 – речь Y принадлежит диктору S

H_1 – речь Y принадлежит не диктору S

матрица ошибок классификации представлена в таблице ниже. В ней $\mathbb{Y}$ – ответ алгоритма на объекте, $\mathcal{Y}$ – истинная метка объекта.

Таблица 2. Матрица ошибок классификации

	$\mathcal{Y} = Y \in S$	$\mathcal{Y} = Y \notin S$
$\mathbb{Y} = Y \in S$	True Positive (TP)	False Positive (FP)
$\mathbb{Y} = Y \notin S$	False Negative (FN)	True Negative (TN)

Тогда ошибка первого рода будет являться вероятностью «отказа своему» (False Negative), и ошибка второго рода – вероятность «допуска чужого» (False Positive).

Введем метрику точности. Она определяется по формуле:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

Для оценки качества работы алгоритма на каждом из классов по отдельности введем метрики precision (точность) и recall (полнота).

$$precision = \frac{TP}{TP+FP} \quad (3)$$

$$recall = \frac{TP}{TP+FN} \quad (4)$$

$Precision$ можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а $recall$ показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм. Введение $precision$ не позволяет нам записывать все объекты в один класс, так как в этом случае мы получаем рост уровня False Positive (FP). $Recall$ демонстрирует способность алгоритма обнаруживать данный класс вообще, а $precision$ — способность отличать этот класс от других классов.

2.6 Оптимизация метода

Рис.9 Функциональная схема идентификации диктора

Для того, чтобы сохранить образец голоса нового диктора, необходимо создать и обучить GMM, а затем сохранить ее в базе. Однако в процессе обучения возникает трудность – голосовых данных слишком мало. Для хорошего обучения GMM желательно иметь

несколько часов речи диктора. В данной задаче такое требование невозможно. Нужно уметь строить модель на небольшом объеме данных – паре слов или паре фраз.

Будем решать эту проблему с помощью Универсальной фоновой модели (Universal Background Model, UBM, УФМ) [21].

Универсальная фоновая модель (UBM) в задаче распознавания диктора представляет собой некоторую модель, обученную на конечном большом множестве голосов определённых дикторов и содержащую в себе апостериорные знания об устройстве человеческого голоса в целом. UBM также является большой моделью гауссовой смеси (GMM), обученной для представления дикторонезависимого распределения признаков. Модель обучается методом Expectation-maximization (EM) на обучающем множестве. Вектор средних, извлечённый из модели после обучения, называется супервектором средних. Когда необходимо получить признаки для вновь пришедшего произнесения, параметры UBM подстраиваются методом оценки апостериорного максимума (maximum a posteriori probability estimate, MAP) и полученный вектор средних модели уже называется дикторским супервектором средних. Подстроенная модель будет являться конечной GMM для нового диктора, которая и будет сохранена.

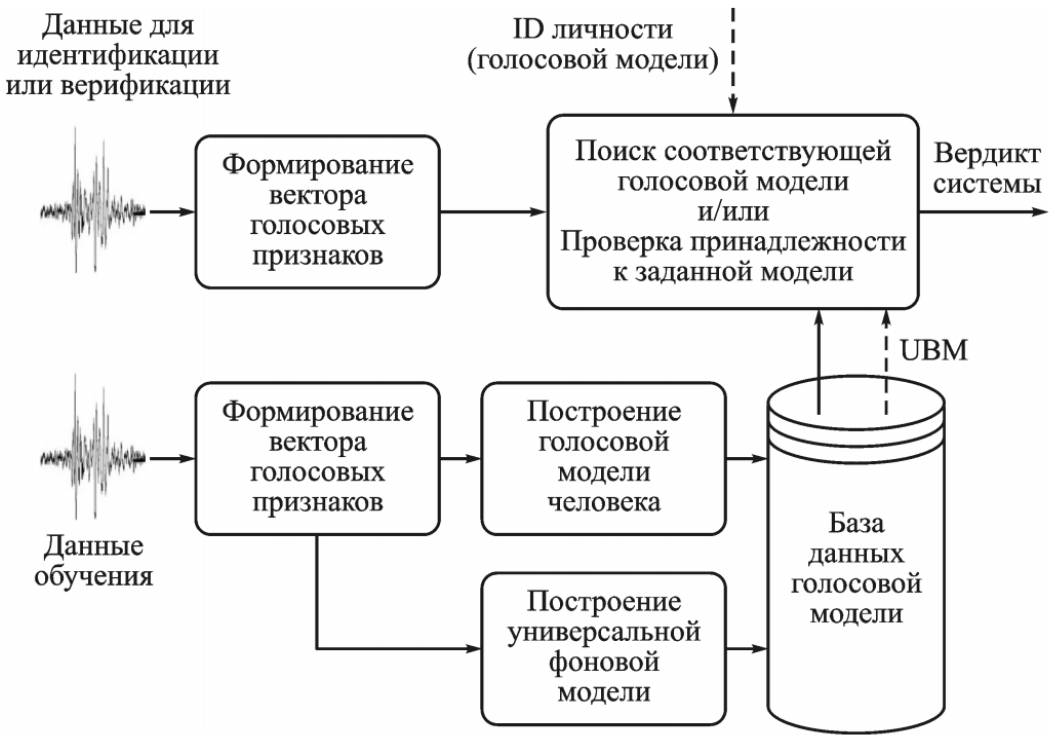

Рис.10 Функциональная схема системы голосовой идентификации/верификации личности

Таким образом, UBM позволяет реализовывать GMM на небольшом количестве входных данных нового диктора, а также позволяет увеличить точность распознавания диктора по сравнению с неадаптированными моделями, которые можно построить на небольшой имеющейся обучающей выборке.

Системы, созданные с использованием UBM, называются системами верификации диктора на основе модели Гауссовых смесей и универсальной фоновой модели (GMM-UBM). Первые варианты подобных систем предложены в [22,23,24,25].

Принцип обучения и работы UBM изображён на рисунке.

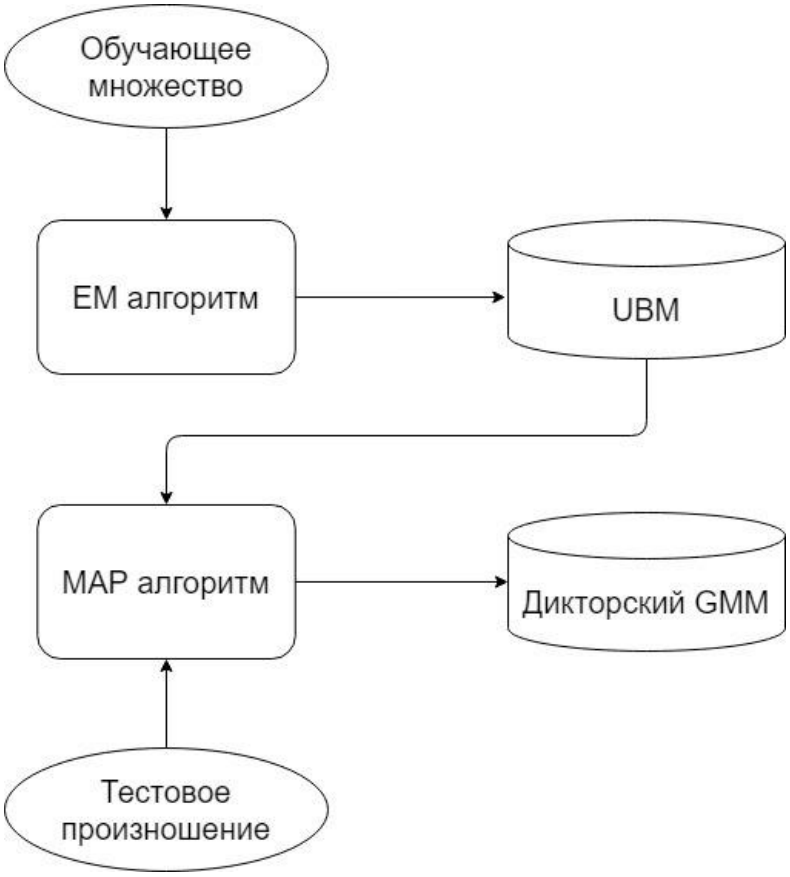

Рис.11 Схема обучения и функционирования UBM-GMM

При создании UBM необходимо, чтобы данные, используемые для обучения модели, были сбалансированными по отношению к дальнейшему применению системы. Другими словами, голоса дикторов должны сильно отличаться, в выборке должны присутствовать как высокие голоса, так и низкие, с разной скоростью речи и разными записывающими системами - сбалансированы и по типу используемых при записи дикторов микрофонов.

В [23] показано, что системы голосовой верификации, производящие адаптацию модели диктора из универсальной фоновой модели, имеют намного лучшую точность, чем системы, в которых модель диктора обучается отдельно от УФМ. Это можно объяснить тем,

что УФМ покрывает большинство классов акустических событий, появляющихся в речи дикторов. Соответственно, во время адаптации модели диктора, часть таких событий, появившихся в речи диктора, изменяют и подстраивают компоненты смеси под конкретного диктора. Оставшаяся часть событий, не встречающаяся в обучающей выборке, копируется из УФМ. Таким образом, это добавляет устойчивости модели к тем акустическим событиям конкретного диктора, которые отсутствовали в обучающей выборке.

Для D-мерного вектора признаков x, поданного на вход, плотность смеси рассчитывается как:

$$P(x|\lambda) = \sum_{k=1}^{M} w_k * g(x|\mu_k, \Sigma_k) \quad (5)$$

Здесь:

x – является D-мерным вектором признаков

$w_k, k = 1, .., M$ – является ли смесь весами, в сумме они = 1

$\mu_k, k = 1, .., M$ – матожидание каждой Гауссианы

$\lambda = (w_k, \mu_k, \Sigma_k), k = 1, .., M$ – параметры каждой GMM

$\Sigma_k, k = 1, .., M$ – ковариация каждой Гауссианы

$g(x|\mu_k, \Sigma_k)$ – плотности Гауссиан, определяемые как:

$$g(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_k|^{\frac{1}{2}}} exp^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)} \quad (6)$$

Обычно используется диагональная ковариационная матрица, а не полная ковариационная матрица, поскольку она более эффективна с вычислительной точки зрения и эмпирически работает лучше.

GMM обучается на наборе обучающих векторов. Параметры GMM вычисляются итеративно с использованием алгоритма максимизации ожидания (EM), и поэтому нет никаких гарантий, что он дважды преобразуется к одному и тому же решению в зависимости от инициализации.

Алгоритм MAP представляет собой тот же алгоритм EM с начальными параметрами модели GMM, равными параметрам UBM. На каждой итерации происходит пересчет и изменение матожидания модели, сохраняя значение ковариации, так как изменение ковариации не улучшает качество работы модели.

Для пересчета матожидания используется максимальная апостериорная адаптация (maximum a posteroiri adaptation):

$$\mu_k^{MAP} = \alpha_k \mu_k + (1 - \alpha_k)\mu_k^{UBM} \quad (7)$$

Здесь:

$\alpha_k = \frac{n_k}{n_k + r_k}$ – коэффициент адаптации матожидания (8)

n_k – количество адаптационных данных

r_k - фактор соответствия (relevance factor)

2.7 Выводы

Для реализации системы идентификации будем использовать метод GMM-UBM на признаках MFCC. Оптимизация в виде UBM позволит избавиться от недостатков и трудностей построения GMM, описанных в предыдущем разделе. GMM, являясь средством моделирования произвольного многомерного распределения данных, будет выполнять функцию классификатора, выдавая плотность смеси на входном векторе признаков для каждого из сохраненных дикторов.

Раздел 3. Разработка системы голосовой идентификации

На данном этапе производился выбор и изучение языка и вспомогательных библиотек для реализации системы, произведена и описана сама реализация и проверена работа метода идентификации.

3.1 Выбор языка и вспомогательных модулей для программной реализации системы

Реализация моделей, алгоритмов обработки аудиофайла, функций построения мел-кепстральных коэффициентов, а также дополнительных методов выполнялась на языке Python3.7. В качестве вспомогательных библиотек использовались библиотеки librosa, pydub и contextlib для загрузки и работы с аудиофайлом, python_speech_features для построения MFCC, webrtcvad для реализации функции детектирования наличия речи в аудиосигнале, joblib для записи данных и моделей, а также Numpy для реализации функций и алгоритмов. Разработка проводилась в программной среде Jupyter Notebook и PyCharm.

3.2 Программная реализация моделей и алгоритмов

Опишем реализованные функции и классы. В процессе работы был реализован класс Frame и функции:

1. def extract_features

Функция, вычисляющая признаки из поданного на вход аудиофрагмента. Функция вычисляет MFCC и добавляет первую и вторую производную.

2. def normalize

Функция нормализации поданного на вход вектора признаков. Многие методы машинного обучения могут вести себя плохо, если отдельные признаки не будут более или менее похожи на стандартные нормально распределенные данные: Гауссовы с нулевым средним и единичной дисперсией. Фукция возвращает нормализованный вектор. Центрирует признаки, удаляя среднее значение каждого признака, а затем масштабирует, деля признаки на стандартное отклонение.

3. def write_wave

Функция, записывающая обработанный аудиофрагмент в .wav файл по указанному пути.

4. class Frame

Класс, представляющий собой фрейм, на множество которых нарезается аудиофрагмент в процессе выделения наполненных голосом фрагментов и удаления тишины.

5. def frame_generator

Функция, генерирующая аудиофреймы (аудиокадры) из поданого на вход аудиофрагмента. Принимает желаемую длительность кадра в миллисекундах, данные и частоту дискретизации. Дает кадры требуемой длительности.

6. def vad_collector

Отфильтровывает аудиофреймы тишины (не наполненные голосом), выдает только озвученный аудиофрагмент. Использует алгоритм скользящего окна с дополнениями по аудиокадрам. Когда более 90% кадров в окне озвучены, устанавливается триггер $triggered$, функция начинает записывать звуковые кадры. Пока 90% кадров в окне не будут озвучены, тригер сохраняет состояние и функция продолжает запись.

7. def map_adaptation

Функция MAP адаптации UBM к GMM для конкретного диктора. Принимает на вход предобученную модель, максимальное количество итераций. Выполняет итерационно подгон параметров UBM, пока не превысит максимальное количество итераций или не будет достигнута указанная точность.

8. def get_data

Загружает аудиофайл и предобрабатывает его. Удаляет тишину и паузы, склеивает и сохраняет. Работает в двух режимах – сохраняет либо все данные, которые получила после удаления пауз, либо какую-то заданную часть. Принимает на вход имя файла, который нужно загрузить и путь, по которому нужно сохранить обработанный файл.

9. def get_features

Функция извлекает вектор признаков заданной размерности из аудиофайла либо загружает сохраненные признаки из файла. В качестве параметров передается имя файла, количество MFCC. Возвращает извлеченные признаки

10. def get_ubm

Создает модель UBM с заданными параметрами и обучает ее на входных признаках либо загружает уже существующую сохраненную модель из файла. Возвращает модель UBM.

11. def get_gmm

Создает модель GMM из поданной на вход UBM и обучает ее на входных признаках либо загружает уже существующую сохраненную модель из файла. Возвращает модель GMM.

12. get_recording

Загружает несколько аудиофайлов из заданной директории, склеивает их и нарезает. Возвращает путь, по которому сохранен обработанный файл.

13. get_speaker

Извлекает признаки из аудиофайла диктора, расположенного по определенному пути. Возвращает вектор признаков

14. create_gmms

Извлекает признаки из указанного файла, загружает модель UBM, обучает UBM с помощью MAP адаптации. Сохраняет обученную модель.

15. test_gmms

Функция создана для тестирования существующих моделей. Считает время работы, а также метрики качества работы каждой модели, расположенной в определенной заданной директории.

3.3 Выводы

В данном разделе выбран программный язык и среда разработки, перечислены вспомогательные библиотеки и описаны функции и классы, реализованные в процессе создания системы идентификации, описан выбранный способ создания веб-сервиса.

Раздел 4. Экспериментальные исследования работы системы

В данном разделе приводятся результаты экспериментальных исследований метода идентификации диктора с помощью конструирования модели GMM-UBM с различными архитектурными параметрами, а также выбор параметров модели для получения системы с оптимальными показателями работы, такими как точность, полнота. Была использована обучающая выборка LibriSpeech объёмом 80 голосов различных дикторов, записанных с помощью разных аудиоустройств. На этой выборке продемонстрировано влияние различных параметров архитектуры модели на результат работы. Проведённые исследования позволили определить оптимальную архитектуру системы.

После этого было проведено сравнение результатов работы на выборках разной длительности для установления минимальной наилучшей длительности для хорошей работы.

4.1 Описание тестовых данных, объема и особенностей

Обучающая и валидационная выборки состоят из базы данных LibriSpeech. Она включает в себя несколько сотен часов английской речи с различными диалектами, записанными на разные устройства. Аудиофайлы чистые, без фонового шума, посторонних голосов и дополнительных артефактов.

Предварительная обработка заключалась в удалении пауз и аудиофрагментов тишины, склеивании фрагментов и нарезке на фрагменты с заданной длительностью для валидации и обучения GMM.

4.2 Тестирование работы системы идентификации

Для определения оптимальных параметров модели UBM было построено 27 моделей различной архитектуры, для каждой модели проведено тестирование работы при 40 добавленных дикторах. Результаты приведены в таблице. Значения Accuracy, Precision, Recall вычислены как среднее арифметическое для каждого класса (диктора). Также для сравнения было измерено среднее время принятия решения классификатором в зависимости от длительности аудиофрагмента речи, поданного на вход для идентификации (столбцы «Время на 40 с» и «Время на 5 с»).

Таблица 3. Качество работы построенных моделей

Компоненты	Ковариация	MFCC	Accuracy	Precision	Recall	Время (на 40 с)	Время (на 5 с)
8	diag	13	0.9946	0.8125	0.8916	2.2191	1.6222
8	diag	20	0.9951	0.8125	0.9	2.2509	1.7025
8	diag	8	0.9942	0.8203	0.8833	2.1254	1.4941

8	full	13	0.9945	0.8375	0.8916	4.71203	3.1141
8	full	20	0.9945	0.8511	0.8916	5.2238	3.4823
8	full	8	0.9945	0.8125	0.898	3.3213	2.1422
8	tied	13	0.9945	0.8502	0.898	4.5222	3.0525
8	tied	20	0.9954	0.8875	0.9083	5.3009	3.5039
8	tied	8	0.9952	0.8527	0.878	3.2472	2.0821
16	diag	13	0.9937	0.8751	0.925	2.45507	1.67226
16	diag	20	0.9954	0.8875	0.9	2.5688	1.7152
16	diag	8	0.9937	0.8125	0.875	2.3292	1.6191
16	full	13	0.9951	0.8501	0.9	6.2387	4.1502
16	full	20	0.9951	0.8512	0.9	8.86304	5.6113
16	full	8	0.9951	0.8524	0.9	4.6579	2.96404
16	tied	13	0.9963	0.8503	0.925	6.2921	4.0401
16	tied	20	0.9963	0.8125	0.925	8.6095	5.5437
16	tied	8	0.9946	0.8208	0.875	4.6362	2.8073
32	diag	13	0.9954	0.8125	0.875	2.82457	2.05204
32	diag	20	0.9958	0.8875	0.925	2.8914	2.0058
32	diag	8	0.9951	0.8513	0.925	2.7116	1.9027
32	full	13	0.9954	0.8629	0.9083	12.5033	8.06306
32	full	20	0.9954	0.8629	0.9083	15.4039	9.91009
32	full	8	0.9954	0.8629	0.9083	7.4432	4.3401
32	tied	13	0.9951	0.8125	0.886	12.8066	8.0172
32	tied	20	0.9954	0.8875	0.9083	15.3268	9.8261
32	tied	8	0.9951	0.8208	0.965	7.3689	4.3601

Таким образом, описанный алгоритм обучения и построения модели GMM-UBM был опробован с различными параметрами:

1. Количество MFCC: 8, 13, 20
2. Тип ковариации в GMM: full, diag, tied
3. Количество компонент: 8, 16, 32
4. С разной длительностью тестовых данных: 5 сек, 20 сек, 40 сек

Для обучения GMM брались аудиофрагменты речи длительностью 20 секунд

4.3 Выводы по результатам экспериментальных исследований

Как видно из таблицы, все построенные модели дают хороший результат работы. Было замечено, что модели с типом ковариации=full требуют намного больший объем памяти по сравнению с моделями других параметров (например, для 16 компонент и MFCC = 20 объем памяти для моделей с типами diag, tied, full равен соответственно 32, 93, 1359 КБ). Таким образом, по объему памяти больше всех требует модель full, меньше всех – diag. Причиной этому может выступать большее число настраиваемых параметров модели. Также модель с типом ковариации full дольше всех обучается на выборке (соответственно, модель с типом ковариации diag обучается быстрее всех).

Время работы процесса идентификации значительно увеличивается при увеличении количества компонент смеси, увеличении длительности входного аудиофрагмента для идентификации, увеличении количества MFCC (размерности вектора признаков) и выборе типа ковариации full.

Как итог, все спроектированные модели дают примерно одинаковый результат. Оптимальной моделью может выступать любая модель с относительно небольшим временем работы. Как пример, оптимальной может быть принята модель с параметрами:

1. Количество компонент UBM: 16
2. Тип ковариации: diag
3. Количество MFCC: 13
4. Длительность обучения GMM: 20 сек
5. Длительность тестового фрагмента: 20 сек

Важно заметить, что длительность тестового фрагмента может отличаться от длительности фрагмента, поданного на вход модели. В процессе предобработки после удаления тишины и пауз фрагмент может значительно сократиться в размере, поэтому оптимальная длительность выбрана «с запасом».

Заключение

В результате выполнения данной учебно-исследовательской работы будет достигнута главная цель - создана система идентификации личности диктора по его голосу в полученной аудиозаписи. В ходе работы были выполнены следующие задачи:

1. Получена обучающая выборка.
2. Изучен и разработан метод идентификации диктора.
3. Спроектирована и разработана оптимальная архитектура выбранной модели.
4. Пройдено тестирование работы реализованной системы.

Достигнутая степень успешного распознавания составляет более 99 процентов на вылидационных данных при точности и полноте более 87 и 90 процентов соответственно.

Исходный код проекта доступен в открытом репозитории Github [26]. В дальнейшем для улучшения работы необходимо увеличивать объем и разнообразие обучающей выборки дли построения модели UBM. Областью применения может служить любая задача идентификации, верификации и аутентификации пользователя системы по его голосу, задача диаризации и задача определения количества говорящих в записанном аудиофайле.

Литература

1. Zhongxin Bai, Xiao-Lei Zhang, Jingdong Chen. Speaker Recognition Based on Deep Learning: An Overview. 2020. URL: https://arxiv.org/abs/2012.00931
2. Kumar G.Suvarna, Raju, K.A.Prasad, Dr.Mohan Rao CPVNJ, P.Satheesh. Speaker recognition using GMM
3. Hajavi, A., Etemad A. 2019. A Deep Neural Network for ShortSegment Speaker Recognition. URL: https://arxiv.org/pdf/1907.10420.pdf
4. Joon Son Chung, Arsha Nagrani, Andrew Zisserman. 2018. VoxCeleb2: Deep Speaker Recognition. URL: https://arxiv.org/abs/1806.05622
5. Daniel Foley. Gaussian Mixture Modelling (GMM). URL: https://towardsdatascience.com/gaussian-mixture-modelling-gmm-833c88587c7f
6. И. А. Рахманенко, Р. В. Мещеряков. Анализ идентификационных признаков в речевых данных с помощью GMM-UBM системы верификации диктора. URL: https://www.elibrary.ru/item.asp?id=29296333
7. T. Kinnunen, H. Li, An overview of text-independent speaker recognition: From features to supervectors, Speech communication. 2010. Стр. 12–40
8. Techportal Биометрия. Стандарты. Технологии. Задачи. Решения. URL: http://www.techportal.ru/security/biometrics/mirovoy-i-rossiyskiy-rynki-biometrii/mirovoy-rynok-marketsandmarkets
9. Ю. Н. Матвеев Технологии биометрической идентификации личности по голосу и другим модальностям. 2012. URL: https://www.researchgate.net/publication/236142366_Tehnologii_biometriceskoj_identifikacii_licnosti_po_golosu_i_drugim_modalnostam_BIOMETRIC_TECHNOLOGIES_OF_PERSON_IDENTIFICATION_BY_VOICE_AND_OTHER_MODALITIES
10. Bimbot F. et al. A tutorial on text-independent speaker verification 2004. Стр. 430-451. URL: https://studylib.net/doc/12607069/a-tutorial-on-text-independent-speaker-verification-pleas...
11. Reynolds D., Rose R. Robust text-independent speaker identification using Gaussian mixture speaker models. 1995. Стр. 72–83
12. Reynolds D. Experimental evaluation of features for robust speaker identification. 1994. Стр. 639–643
13. Бучнева Т.И., Кудряшов М.Ю. Нейронные сети в задаче идентификации диктора по голосу. 2015 URL: http://pmk-vestnik.tversu.ru/issues/2015-2/vestnik-pmk-2015-2-buchneva.pdf

14. Рахманенко Иван Андреевич, Шелупанов Александр Александр, Костюченко Евгений Юрьевич Автоматическая верификация диктора по произвольной фразе с применением свёрточных глубоких сетей доверия URL: https://cyberleninka.ru/article/n/avtomaticheskaya-verifikatsiya-diktora-po-proizvolnoy-fraze-s-primeneniem-svyortochnyh-glubokih-setey-doveriya

15. Единая биометрическая система URL: https://bio.rt.ru/citizens/

16. TAdviser Сбербанк вводит возможность голосового подтверждения операций URL: https://www.tadviser.ru/a/371589

17. Российская газета RG Биометрия. 2021 URL:https://rg.ru/2021/03/10/mincifry-biometriia-dlia-polucheniia-gosuslug-ne-budet-obiazatelnoj.html

18. Алексей Лукацкий Голосовая биометрия. Краткий обзор технологии. 2015. URL: https://www.securitylab.ru/blog/personal/Business_without_danger/147943.php

19. Oscar Contreras Carrasco Gaussian Mixture Models Explained. From intuition to implementation. 2019. URL: https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95

20. Jake VanderPlas In Depth: Gaussian Mixture Models URL: https://jakevdp.github.io/PythonDataScienceHandbook/05.12-gaussian-mixtures.html

21. Stephen M Chu, Daniel Povey Universal background model based speech recognition. 2008. URL: https://www.researchgate.net/publication/224762192_Universal_background_model_based_speech_recognition

22. Matsui T., Furui S. Likelihood normalization for speaker verification using a phoneme-and speaker-independent model. 1995. Стр. 109-116.

23. Reynolds D. A. Comparison of background normalization methods for text-independent speaker verification. 1997 Стр. 963-966

24. Hermansky H., Malayath N. Speaker verification using speaker-specific mappings. 1998 Стр.111-114.

25. Quatieri T. F. et al. Speaker and language recognition using speech codec parameters. 1999 Стр. 787-790

26. https://github.com/Dora9000/voice_recognition

Приложения

Приложение 1. Код скрипта предобработки данных

```
import collections
import contextlib
import librosa
import numpy as np
import python_speech_features
import wave


def extract_features(y, sr, n_mfcc):
    mfcc = python_speech_features.mfcc(signal=y, samplerate=sr, numcep=n_mfcc)
    mfcc = mfcc.T
    mfcc_delta = librosa.feature.delta(mfcc)
    mfcc_delta2 = librosa.feature.delta(mfcc, order=2)
    stacked = np.vstack((mfcc, mfcc_delta, mfcc_delta2))
    return stacked.T


def normalize(feature, eps=1e-14):
    feats_mean = np.mean(feature, axis=0)
    feats_std = np.std(feature, axis=0)
    return (feature - feats_mean) / (feats_std + eps)


def write_wave(path, audio, sample_rate):
    with contextlib.closing(wave.open(path, 'wb')) as wf:
        wf.setnchannels(1)
        wf.setsampwidth(2)
        wf.setframerate(sample_rate)
        wf.writeframes(audio)


class Frame(object):
    def __init__(self, bytes, timestamp, duration):
        self.bytes = bytes
        self.timestamp = timestamp
        self.duration = duration


def frame_generator(frame_duration_ms, audio, sample_rate):
    n = int(sample_rate * (frame_duration_ms / 1000.0) * 2)
    offset = 0
    timestamp = 0.0
    duration = (float(n) / sample_rate) / 2.0
    while offset + n < len(audio):
        yield Frame(audio[offset:offset + n], timestamp, duration)
```

```
47.         timestamp += duration
48.         offset += n
49.
50.
51. def vad_collector(sample_rate, frame_duration_ms, padding_duration_ms, vad,
    frames):
52.     num_padding_frames = int(padding_duration_ms / frame_duration_ms)
53.     ring_buffer = collections.deque(maxlen=num_padding_frames)
54.     triggered = False
55.
56.     voiced_frames = []
57.     for frame in frames:
58.         is_speech = vad.is_speech(frame.bytes, sample_rate)
59.
60.         if not triggered:
61.             ring_buffer.append((frame, is_speech))
62.             num_voiced = len([f for f, speech in ring_buffer if speech])
63.             if num_voiced > 0.9 * ring_buffer.maxlen:
64.                 triggered = True
65.                 for f, s in ring_buffer:
66.                     voiced_frames.append(f)
67.                 ring_buffer.clear()
68.         else:
69.             voiced_frames.append(frame)
70.             ring_buffer.append((frame, is_speech))
71.             num_unvoiced = len([f for f, speech in ring_buffer if not speech])
72.             if num_unvoiced > 0.9 * ring_buffer.maxlen:
73.                 triggered = False
74.                 yield b''.join([f.bytes for f in voiced_frames])
75.                 ring_buffer.clear()
76.                 voiced_frames = []
77.     if voiced_frames:
78.         yield b''.join([f.bytes for f in voiced_frames])
```

Приложение 2. Код скрипта создания модели

```
import numpy as np


def map_adaptation(gmm, data, max_iterations=300, likelihood_threshold=1e-20, relevance_factor=16):
    N = data.shape[0]
    D = data.shape[1]
    K = gmm.n_components

    mu_new = np.zeros((K, D))
    n_k = np.zeros((K, 1))

    mu_k = gmm.means_
    cov_k = gmm.covariances_
    pi_k = gmm.weights_

    old_likelihood = gmm.score(data)
    new_likelihood = 0
    iterations = 0
    while abs(old_likelihood - new_likelihood) > likelihood_threshold and iterations < max_iterations:
        iterations += 1
        old_likelihood = new_likelihood
        z_n_k = gmm.predict_proba(data)
        n_k = np.sum(z_n_k, axis=0)
        for i in range(K):
            temp = np.zeros((1, D))
            for n in range(N):
                temp += z_n_k[n][i] * data[n, :]
            mu_new[i] = (1 / max(n_k[i], 1e-20)) * temp

        adaptation_coefficient = n_k / (n_k + relevance_factor)
        for k in range(K):
            mu_k[k] = (adaptation_coefficient[k] * mu_new[k]) + ((1 - adaptation_coefficient[k]) * mu_k[k])
        gmm.means_ = mu_k

        log_likelihood = gmm.score(data)
        new_likelihood = log_likelihood
    return gmm
```

Приложение 3. Код скрипта идентификации

```
1. import os
2. import shutil
3. import pickle
4. import webrtcvad
5. from sklearn.mixture import GaussianMixture
6. from model import map_adaptation
7. from data_preprocess import *
8. from settings import *
9. import copy
10. import joblib
11. from pydub import AudioSegment
12. import time
13.
14. RESULTS_PATH = '../UIR3/data'
15.
16.
17. def get_data(file, result_file=RESULTS_PATH + '/chunks/full_chunk.wav',
    all_data=True):
18.     y, sr = librosa.load(file, sr=SR)
19.     pre_emphasis = 0.97
20.     y = np.append(y[0], y[1:] - pre_emphasis * y[:-1])
21.
22.     vad = webrtcvad.Vad(3)
23.     audio = np.int16(y / np.max(np.abs(y)) * 32768)
24.
25.     frames = frame_generator(10, audio, sr)
26.     frames = list(frames)
27.     segments = vad_collector(sr, 50, 200, vad, frames)
28.
29.     full_chunk = []
30.     if all_data:  # all data
31.         for i, segment in enumerate(segments):
32.             full_chunk.append(segment[0: len(segment) - int(100 * sr / 1000)])
33.         write_wave(result_file, b''.join([full_chunk[i] for i in
    range(len(full_chunk))]), sr)
34.     else:  # part of 35000
35.         for i, segment in enumerate(segments):
36.             full_chunk.append(segment[0: min(35000, len(segment) - int(100 * sr /
    1000))])
37.         a = 0
38.         s = 0
39.         for i in range(len(full_chunk)):
40.             if a < len(full_chunk[i]):
41.                 a = len(full_chunk[i])
42.                 s = i
43.         write_wave(result_file, b''.join([full_chunk[s]]), sr)
44.     return result_file
```

```
45.
46.
47. def get_features(file, features_to_file, features_from_file=None, mfcc=None):
48.     if features_from_file is not None:
49.         ubm_features = pickle.load(open(features_from_file, 'rb'))
50.     else:
51.         y, sr = librosa.load(file, sr=None)
52.         f = extract_features(np.array(y), sr, n_mfcc=N_MFCC if mfcc is None else
    mfcc, hop=HOP_LENGTH, window=N_FFT)
53.         ubm_features = normalize(f)
54.         pickle.dump(ubm_features, open(features_to_file, "wb"))
55.     return ubm_features
56.
57.
58. def get_ubm(ubm_features, ubm_file=None, ubm_to_file=None):
59.     if ubm_file is not None:
60.         ubm = joblib.load(ubm_file)
61.     else:
62.         ubm = GaussianMixture(n_components=N_COMPONENTS,
    covariance_type=COVARINACE_TYPE)
63.         ubm.fit(ubm_features)
64.         joblib.dump(ubm, ubm_to_file)
65.     return ubm
66.
67.
68. def get_gmm(ubm, gmm_features, gmm_to_file, gmm_file=None):
69.     if gmm_file is not None:
70.         gmm = joblib.load(gmm_file)
71.     else:
72.         gmm = copy.deepcopy(ubm)
73.         gmm = map_adaptation(gmm, gmm_features, max_iterations=1,
    relevance_factor=16)
74.         joblib.dump(gmm, gmm_to_file)
75.     return gmm
76.
77.
78. def clear_dir(path):
79.     shutil.rmtree(path)
80.     os.mkdir(path, mode=0o777, dir_fd=None)
81.
82.
83. def train_ubm():
84.     DATA_PATH = '../LibriSpeech'
85.     features = []
86.     max_cnt = {}
87.     for root, dirs, files in os.walk(DATA_PATH):
88.         for fn in files:
89.             if fn[-4:] == '.wav':
90.                 s = fn[:-4].split('-')
91.                 s = s[0]
```

```
92.                 if max_cnt.get(s, None) is not None and max_cnt.get(s, None) == 1:
93.                     continue
94.                 max_cnt[s] = 1 if max_cnt.get(s, None) is None else max_cnt[s] + 1
95.                 data = get_data(file=root + '/' + fn,
96.                                 result_file=RESULTS_PATH + '/chunks/' + fn,
    all_data=True)
97.                 feature = get_features(file=data,
98.                                        features_to_file=RESULTS_PATH + '/chunks/'
    + fn[-4:] + '.pkl')
99.                 features.append(feature)
100.          features_ = None
101.          for i in range(len(features)):
102.              if i == 0:
103.                  features_ = features[i]
104.              else:
105.                  features_ = np.vstack((features_, np.array(features[i],
    dtype="float16")))
106.          return features_
107.
108.
109.      TRAIN_UBM = False
110.      if TRAIN_UBM:
111.          clear_dir(RESULTS_PATH + '/chunks')
112.          mfcc = [8, 13, 20]
113.          for N_MFCC in mfcc:
114.              features = train_ubm()
115.              n_comp = [8, 16, 32]
116.              cov = ['tied', 'diag', 'full']
117.              for N_COMPONENTS in n_comp:
118.                  for COVARINACE_TYPE in cov:
119.                      get_ubm(features,
120.                              ubm_to_file=RESULTS_PATH +
    '/ubm_models/ubm_{0}_{1}_{2}mfcc.pkl'.format(N_COMPONENTS,
121.
    COVARINACE_TYPE,
122.
    N_MFCC))
123.
124.
125.      def get_recording(path, path_to_save, remember=True):
126.          infiles = []
127.          for root, dirs, files in os.walk(path):
128.              for fn in files:
129.                  if fn[-4:] == '.wav':
130.                      infiles.append(root + '/' + fn)
131.          sound1 = AudioSegment.from_wav(infiles[0])
132.          for i in range(1, len(infiles)):
133.              sound2 = AudioSegment.from_wav(infiles[i])
134.              sound1 = sound1 + sound2
135.          if remember:
```

```
136.             sound1 = sound1[0:DURATION_TO_REMEMBER * 1000]
137.         else:
138.             step = 4
139.             assert (DURATION_TO_REMEMBER * 1000 <
140.                     DURATION_TO_REMEMBER * 1000 + min(DURATION_TO_RECOGNIZE *
    1000, len(sound1) - 1))
141.             assert (step * DURATION_TO_REMEMBER * 1000 +
    min(DURATION_TO_RECOGNIZE * 1000, len(sound1) - 1) <
142.                     len(sound1) - 1)
143.             sound1 = sound1[DURATION_TO_REMEMBER * 1000:
144.                             step * DURATION_TO_REMEMBER * 1000 +
    min(DURATION_TO_RECOGNIZE * 1000, len(sound1) - 1)]
145.         sound1.export(path_to_save, format="wav")
146.         return path_to_save
147.
148.
149.     def parse_params(model_name):
150.         s = model_name[-6:-4]
151.         if s[0] == '_':
152.             s = s[1]
153.         return int(s)
154.
155.
156.     def get_speaker(id, mfcc, remember):
157.         DATA_PATH = '../LibriSpeech/dev-clean/{0}'.format(id)
158.         data = get_data(file=get_recording(DATA_PATH,
159.                                           remember=remember,
160.                                           path_to_save=RESULTS_PATH +
    "/gmm/speaker.wav"),
161.                         result_file=RESULTS_PATH + '/gmm/chunks/' + str(id) +
    '.wav',
162.                         all_data=True)
163.         feature = get_features(file=data,
164.                                features_to_file=RESULTS_PATH + '/gmm/chunks/' +
    str(id) + '.pkl', mfcc=mfcc)
165.         return feature
166.
167.
168.     def create_gmms(cur_ubm):
169.         ubm_name = RESULTS_PATH + '/ubm_models/' + cur_ubm
170.         ids = os.listdir('../LibriSpeech/dev-clean/')
171.         for id in ids:
172.             gmm_features = np.asarray(get_speaker(id, parse_params(cur_ubm[:-
    4]), remember=True))
173.             ubm = get_ubm(None, ubm_file=ubm_name)
174.             if not os.path.exists(RESULTS_PATH +
    '/gmm_models/{0}'.format(cur_ubm[:-4])):
175.                 os.makedirs(RESULTS_PATH + '/gmm_models/{0}'.format(cur_ubm[:-
    4]))
176.             gmm = get_gmm(ubm, gmm_features,
```

```
                            gmm_to_file=RESULTS_PATH +
    '/gmm_models/{0}/gmm_{1}_speaker_{2}.pkl'.format(
                                cur_ubm[:-4],
                                cur_ubm[:-4],
                                id))
            return RESULTS_PATH + '/gmm_models/{0}'.format(cur_ubm[:-4])


        def test_gmms(path, mfcc):
            ids = os.listdir('../LibriSpeech/dev-clean/')
            models = [(joblib.load(path + '/' + user), user) for user in
    os.listdir(path)]
            statistics = {}

            for _, c in models:
                statistics[str(c.split("_")[-1][:-4])] = {'TP': 0, 'TN': 0, 'FP': 0,
    'FN': 0}
            times = []
            for id in ids:
                start_time = time.time()
                gmm_features = np.asarray(get_speaker(id, mfcc, remember=False))
                answer_speaker = 0
                answer_score = -1e9
                for model, name in models:
                    s = np.array(model.score(gmm_features))
                    if answer_score < s:
                        answer_score = s
                        answer_speaker = name
                answer_speaker = str(answer_speaker.split("_")[-1][:-4])
                times.append(time.time() - start_time)
                if answer_speaker == str(id):
                    statistics[answer_speaker]['TP'] += 1
                    for c in statistics:
                        if c == answer_speaker:
                            continue
                        statistics[c]['TN'] += 1
                else:
                    statistics[id]['FN'] += 1
                    statistics[answer_speaker]['FP'] += 1
                    for c in statistics:
                        if c == answer_speaker or c == str(id):
                            continue
                        statistics[c]['TN'] += 1
            print('---- time : ', sum(times) / len(times), min(times), max(times))
            answer = {'acc': [], 'prec': [], 'rec': []}
            for c in statistics:
                TP = statistics[c]['TP']
                TN = statistics[c]['TN']
                FP = statistics[c]['FP']
                FN = statistics[c]['FN']
```

```
224.
225.                 answer['acc'].append((TP + TN) / max(TP + TN + FP + FN, 1))
226.                 answer['prec'].append(TP / max(TP + FP, 1))
227.                 answer['rec'].append(TP / max(TP + FN, 1))
228.             acc = answer['acc']
229.             prec = answer['prec']
230.             rec = answer['rec']
231.             print('acc: ', sum(acc) / len(acc))
232.             print('prec: ', sum(prec) / len(prec))
233.             print('rec: ', sum(rec) / len(rec))
234.
235.
236.     UBMS = os.listdir('../UIR3/data/ubm_models')
237.     for CUR_MODEL in UBMS:
238.         gmm_path = RESULTS_PATH + '/gmm_models/{0}'.format(CUR_MODEL[:-4])
239.         test_gmms(gmm_path, parse_params(CUR_MODEL[:-4]))
240.
```

Приложение 4. Конфигурационный файл

```
SR = 8000  # sample rate
N_MFCC = 13  # number of MFCC to extract
N_FFT = 0.020  # length of the FFT window in seconds
HOP_LENGTH = 0.010  # number of samples between successive frames in seconds
N_COMPONENTS = 16  # number of gaussians
COVARINACE_TYPE = 'full'  # cov type for GMM

DURATION_TO_REMEMBER = 20
DURATION_TO_RECOGNIZE = 5
```