Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение эвм и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Метод распознавания эмоций по звучащей речи на основе скрытой марковской модели»

Студент ИУ7-86Б _____ Т. А. Казаева
(Подпись, дата)

Руководитель _____ Ю. В. Строганов
(Подпись, дата)

Нормоконтролер _____ Д.Ю. Мальцева
(Подпись, дата)

2023 г.

РЕФЕРАТ

Расчетно–пояснительная записка 77 с., 10 рис., 16 табл., 41 ист, 1 прил.

Объектом разработки является метод идентификации эмоций по звучащей речи. Цель работы – спроектировать и реализовать метод распознавания эмоций по звучащей речи на основе скрытой марковской модели.

Система идентификации эмоций обучена на корпусе данных DUSHA, разработанного в 2022 году и на данный момент не использованного для систем распознавания эмоций в речи. F-мера разработанного классификатора в среднем ≈35%, по каждому классу отдельно – от 28% до 43%.

Знания об эмоции, которую испытывает человек в данный момент, могут быть использованы во многих сферах HCI, начиная от улучшения качества обслуживания и повышения покупательской способности и заканчивая повышением эффективности коммуникации и оказанием психологической помощи.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениями.

1) HCI (*англ. human-computer interaction*) – область науки, которая изучает взаимодействие между человеком и компьютером, а также создает и улучшает интерфейсы, которые облегчают этот процесс.

2) Эмоция – психическое состояние, которое включает в себя такие элементы, как субъективные ощущения, физиологические изменения и выражается через мимику, жесты и голос.

3) Речь – процесс создания звуковых волн, которые возникают в результате вибрации голосовых связок в гортани, и затем формируются в узнаваемые звуки и слова при помощи движений губ, языка, неба и других артикуляционных органов.

4) Фреймы – отрезки аудиосигнала длительности как правило 10-40 мс, идущие «внахлест», то есть таким образом, чтобы начало очередного фрейма пересекалось с концом предыдущего.

5) Частота основного тона – частота колебания голосовых связок при произнесении тоновых звуков.

6) Джиттер – мера возмущений частоты основного тона, показывающая непроизвольные изменения в частоте смежных вибрационных циклов голосовых складок.

7) Шиммер – мера аналогичная джиттеру, только характеризующая пертурбации амплитуд сигнала на смежных циклах колебаний основного тона.

8) Форманты – пики в огибающей спектра звука, создаваемые акустическими резонансами в голосовом тракте.

9) Кепстр – преобразование Фурье от логарифма спектра мощности.

10) Мел – единица измерения частоты звука, основанная на статистической обработке большого числа данных о субъективном восприятии высоты звуковых тонов.

11) Кластеризация – разделение множества входных векторов данных на кластеры (группы) по степени схожести друг с другом.

12) Матрица несоответствий (*англ. confusion matrix*) – таблица, в которой каждая строка соответствует истинному классу объектов, а каждый столбец – прогнозируемому классу. В ячейках таблицы указано количество аудиофайлов, которым в результате алгоритма распознавания был присвоен тот или иной класс.

ВВЕДЕНИЕ

Человеческая речь является одними из самых естественных средств общения. Она содержит широкий спектр разнообразной информации, в том числе и эмоциональной. Для информационной системы важно не только понимать эмоции, но и их различать: например, исследование [1] показало, что различные эмоции могут влиять на покупательное поведение по-разному. Например, чувство счастья может способствовать большей трате денег, в то время как чувство грусти может привести к сокращению трат. Более того, исследование показало, что люди, испытывающие страх или тревогу, часто проявляют склонность к покупке товаров, которые обеспечивают им защиту или безопасность.

Если в межличностной коммуникации большинство людей справляется с распознаванием речевых эмоций, то искусственные системы этому нужно обучить. Созданием технологий, ответственных за обработку эмоциональной информации в информационных системах, занимается направление, получившее название «аффективные» или «эмоциональные вычисления» (*англ. affective computing*). В области аффективных вычислений существует множество техник обработки речевого сигнала и моделей классификации эмоций.

Цель настоящей работы – разработать метод определения эмоций по звучащей речи на основе скрытой марковской модели. Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести анализ существующих эмоциональных корпусов и выбрать наиболее подходящий для обучения классификатора;
- провести обзор информативных признаков, характеризующих речь и способов их выделения;
- провести обзор классификаторов, используемых в анализе речевых эмоций;
- спроектировать и реализовать метод детектирования эмоций;
- определить качественные характеристики классификатора, реализованного в рамках метода.

1 Аналитический раздел

1.1. Категоризация эмоциональных данных

1.1.1. Дискретное пространство эмоций

Дискретный подход основан на выделении фундаментальных (базовых) эмоций, сочетания которых порождают разнообразие эмоциональных явлений. Разные авторы называют разное число таких эмоций – от двух до десяти. П. Экман на основе изучения лицевой экспрессии выделяет пять базовых эмоций: гнев, страх, отвращение, печаль и радость [2]. Первоначальная версия 1999 года также включала удивление [3]. Р. Плутчик [4] выделяет восемь базисных эмоций, деля их на четыре пары, каждая из которых связана с определенным действием: страх, уныние, удивление и т. д.

На сегодняшний день существование базовых эмоций ставится под сомнение. Теория встречает ряд концептуальных проблем, таких как, например, эмпирическое определение набора базовых эмоций или критерии синхронизации эмоциональных реакций. Однако, многие решения в области автоматического детектирования эмоций основаны на дискретной модели эмоциональной сферы. Например, решение компании «Affectiva» [5].

1.1.2. Многомерное пространство эмоций

Многомерное пространство представляет собой эмоции в координатном многомерном пространстве. В качестве ее источника рассматривают идею В. Вундта о том, что многогранность чувств человека можно описать с помощью трех измерений: удовольствие-неудовольствие, расслабление-напряжение, возбуждение-успокоение. Вундт заключил [6], что эти измерения охватывают все разнообразие эмоциональных состояний. Данные для этой теории были получены с помощью метода интроспекции.

Эмоциональная сфера представляется как многомерное пространство, образованное некоторым количеством осей координат. Оси задаются полюсами первичных характеристик эмоций. Отдельные эмоции – это точки, местоположение которых в «эмоциональном» пространстве определяется степенью выраженности этих параметров.

Один из примеров описываемого подхода – модель Дж. Рассела. В ней

водится двумерный базис, в котором каждая эмоция характеризуется валентностью (*англ. valence*) и интенсивностью (*англ. arousal*). Измерение валентности отражает то, насколько хорошо человек ощущает себя на уровне субъективного переживания от максимального неудовольствия до максимального удовольствия. Измерение активации связано с субъективным чувством энергии и ранжируется в диапазоне от дремоты до бурного возбуждения. Такой подход используется, например, в наборе данных «RECOLA» [7].

Аналогично вопросу о количестве эмоций в дискретной модели, вопрос о количестве измерений остается открытым. Использование только двух критикуется на том основании, что они не позволяют устанавливать различия между отдельными эмоциональными состояниями (например, страх, гнев, ревность, презрение и др. имеют отрицательную валентность и высокую активацию).

1.1.3. Гибридное пространство эмоций

Гибридная модель представляет собой комбинацию дискретной и многомерной модели. Примером такой модели являются «Песочные часы эмоций», предложенные Камбрией, Ливингстоном и Хуссейном [8].

Согласно этой классификации, в отдельной области n-мерного эмоционального пространства различия между эмоциями могут определяться в терминах измерений, имеющих отношение к этой области. Эмоции могут быть сопоставимы по измерениям внутри и вне категорий, и каждая категория может иметь свои отличительные признаки [9]. Каждое измерение характеризуется шестью уровнями силы, с которой выражены эмоции. Данные уровни обозначаются набором из двадцати четырех эмоций. Поэтому совершенно любая эмоция может рассматриваться как и фиксированное состояние, так и часть пространства, связанная с другими эмоциями нелинейными отношениями.

1.2. Корпуса данных речевых эмоций

1.2.1. Корпуса данных на иностранных языках

Корпус данных RAVDESS [10]. Набор содержит записи 24 профессиональных актеров (12 мужчин и 12 женщин), озвучивающих две одинаковые фразы на английском языке с североамериканским акцентом в двух вариантах: речь и пение. На каждого актера набор предоставляет 60 записей. Использовано дис-

кретное эмоциональное пространство, состоящее из семи эмоций: спокойствие, гнев, страх, отвращение и радость. Каждая фраза представлена двумя уровнями эмоциональной интенсивности для каждой из эмоций и безэмоционально.

Корпус данных SAVEE [11] *(англ. Surrey Audio-Visual Expressed Emotion)* состоит из речи четырех актеров мужского пола, говорящих на английском с британским акцентом. Эмоциональная разметка для каждого высказывания соответствует одной из шести базовых эмоций (радость, печаль, гнев, удивление, страх, отвращение) или нейтральному состоянию. Всего было записано 15 уникальных фраз, каждая фраза записывалась безэмоционально дважды.

Корпус данных Emo-DB [12]. Немецкие актеры (5 женщин и 5 мужчин) имитировали эмоции, произнося фразы, интонационно отражающие различные эмоции (гнев, радость, печаль, страх), а также произнести некоторые из них так, чтобы они не несли никакой эмоциональной нагрузки. Смысл фраз не соответствовал интонационному оформлению. Запись файлов для базы данных проводилась в звукоизолированной комнате. Для оценки качества записанной речи в Берлинском университете был проведен перцептивный тест которым было предложено оценить, к какой из эмоций относится прослушанная единожды запись. Сообщения с уровнем распознавания выше 80% и естественностью звучания свыше 60% вошли в итоговый набор.

Корпус данных TESS [13] *(англ. Toronto emotional speech set)* содержит 2800 звуковых дорожек формата «WAV». Набор озвучен только женскими голосами и размечен по 6 базовым эмоциям: гнев, отвращение, страх, счастье, печаль, удивление. Также присутствуют записи безэмоциональной речи.

1.2.2. Корпуса данных на русском языке

Корпус данных RUSLANA [14]. Является первым русскоязычным эмоциональным набором данных. Содержит записи 61 человека (12 мужчин и 49 женщин), которые произносили десять предложений с выражением следующих эмоциональных состояний: удивление, счастье, гнев, грусть и страх. Также предложения были записаны безэмоционально. Таким образом, в сумме база содержит 3 660 записей. Общая продолжительность аудиозаписей составляет более 31 часа.

Корпус данных DUSHA [15] – русскоязычный набор эмоциональных данных. Набор данных состоит из более 300 000 аудиозаписей и разделен на

два домена – «Crowd» и «Podcast». Длительность составляет около 350 часов аудио. Разметка включает в себя 4 класса: радость, грусть, злость и безэмоциональную речь.

Русскоязычный эмоциональный корпус (REC) [16] состоит из 295 видеозаписей университетских зачетов и экзаменов и 510 видеозаписей общения с клиентами в службе одного окна ГУ ИС г. Москвы. К настоящему моменту размечено 192 файла. Разметка содержит информацию о мимике и жестах, которая размещена на временной шкале. Учитывается мимика двух органов: глаза (взгляды вверх и по сторонам продолжительностью больше 0.5 сек) и рот (неречевые движения губ, манипуляции языком и движения челюстью). Движения рук представлены как самостоятельные, так и с использованием сторонних предметов, тела или одежды.

1.2.3. Сравнение существующих эмоциональных корпусов

Сравнение существующих корпусов представлен в таблице 1.1.

Таблица 1.1 – Сравнение существующих эмоциональных корпусов

Название	*Количество эмоций*	*Количество голосов*		*Количество лингвистических единиц*	*Публичный*
		М	Ж		
RAVDESS	8	12	12	2 предл.	да
SAVEE	6	4	0	15 предл.	да
Emo-DB	6	5	5	10 предл.	да
TESS	8	0	2	200 слов	да
RUSLANA	4	12	49	10 предл.	нет
DUSHA	4	-	-	-	да
REC	-	-	-	-	нет

Прочерк в графе «Количество голосов» означает, что практически для каждой записи в корпусе голос уникальный. Прочерк в графе «Количество эмоций в разметке» означает отсутствие эмоциональной разметки. прочерк в графе «Количество лингвистических единиц» означает, что каждая запись в корпусе

использует уникальный набор лингвистических единиц.

Существующие корпуса имеет смысл сравнивать, во-первых, по размеру представленных данных. При увеличении размера корпуса улучшается статистическая значимость результатов, что позволяет получать более точные оценки параметров модели. На данный момент самым большим русскоязычным набором данных является DUSHA.

Также следует обратить внимание на разнообразие корпуса, а именно на количество представленных голосов. Разнообразие обучающего корпуса может существенно влиять на точность классификации. Если корпус содержит небольшое количество голосов, то модель может не обладать достаточной обобщающей способностью и не сможет правильно классифицировать новые данные.

Стоит учитывать, что классификатор может потерять способность обобщения и перестать обнаруживать закономерности в данных при перегрузке эмоциональными классами. В большинстве представленных корпусов за основу взята «большая шестерка» эмоций П. Экмана [2]. Теория шести базовых эмоций в настоящее время хоть и ставится под сомнение, для аффективных вычислений такой подход является наиболее эффективным, поскольку при расширении эмоционального пространства эмоции могут быть слабо различимы.

Также на качество классификации может влиять качество самих аудиофайлов, поскольку информативные признаки, характеризующие эмоцию могут быть чувствительны к шуму или искажениям в записи. Поэтому для корпусов, использующихся в распознавании эмоций из звучащей речи, запись аудиофайлов желательно производить в студийных условиях.

В большинстве существующих корпусов эмоции имитируются. В таком случае далеко не всегда удается отразить реальные эмоциональные состояния человека. Также излишняя театральность при симуляции эмоций может отрицательно повлиять на результат. Однако, симуляция эмоций может быть полезной в тех случаях, когда для получения набора данных с реальными эмоциями участников исследования недостаточно ресурсов.

1.2.4. Интонационный контур при выражении эмоций

Звучащие предложения обладают интонацией: повествовательной, вопросительной, ответной, перечислительной, восклицательной и т.п. Предполагается, что существует связь между выражением эмоции и использованием одного

из существующих в русском языке интонационного контура [17]. В русском языке выделяют 7 интонационных контуров (ИК) [18]:

– **ИК-1** характеризуется понижением тона на ударной части:
«Анна стоит на мосту. Наташа поет.»,
используется для выражения завершенности в повествовательных предложениях;

– при **ИК-2** ударная часть произносится с некоторым повышением тона:
«Кто пьет сок? Как поет Наташа?»,
наблюдается в вопросе с вопросительными словами;

– **ИК-3** характеризуется значительным повышением тона на ударной части:
«Это Антон? Ее зовут Наташа?»,
наблюдается в вопросе без вопросительных слов;

– **ИК-4** характеризуется повышением тона, продолжающееся на безударных слогах:
«А вы? А это?»,
наблюдается в вопросе без вопросительных слов;

– **ИК-5** используется при выражении оценки в предложениях с местоименными словами:
«Какой сегодня день!»,
наблюдается повышение тона на ударной части;

– **ИК-6**, аналогично ИК5, используется при выражении оценки в предложениях с местоименными словами, однако повышение тона происходит на ударной части и продолжается на заударной части:
«Какой сок вкусный!»;

– **ИК-7**, аналогично ИК-1 используется для выражения завершенности в повествовательных предложениях:
«И Антон стоит на мосту.»,
но ударная часть, в отличие от ИК-1, эмоционально окрашена.

ИК в сочетании с некоторыми артикуляционными особенностями может выступать в качестве средства выражения эмоций. Однако, не каждая эмоция может

быть выражена с использованием всех семи ИК. Например, эмоция удивления выражается с помощью ИК-2, ИК-3, ИК-4 и ИК-6. [19] Такая связь важна для систем распознавания эмоций в речи. Результат работы классификатора – набор вероятностей принадлежности высказывания к тому или иному эмоциональному классу. Если вероятности принадлежности высказывания к нескольким эмоциональным классам схожа по значению, то решение принимается на основе дополнительной информации о высказывании. В качестве дополнительной информации имеет смысл использовать номер интонационного контура. На данный момент ни один корпус эмоциональной речи не содержит информацию об ИК.

1.3. Выделение информативных признаков

Задача распознавания эмоций решается непосредственно по оцифрованному сигналу в два этапа: выделение и отбор информативных признаков и классификация (сопоставление признаков).

Сигнал содержит много избыточной для анализа информации. Поэтому для того, чтобы привести сигнал в вид, который будет использован алгоритмом распознавания, требуется выделить набор информативных признаков речевого сигнала. К выделяемому набору признаков предъявляются следующие требования [20]:

– с помощью выделенного набора признаков можно получить наиболее значимую информацию из акустического сигнала;

– размер выборки должен быть минимальным для увеличения быстродействия разрабатываемой системы распознавания эмоций.

Характеристики речевого сигнала, использующиеся для определения эмоций по речи, можно разделить на две группы: просодические и спекртальные.

Просодическими признаками, содержащими информацию об эмоции, являются признаки, основанные на количественной оценке частоты основного тона, интенсивность (энергия) речевого сигнала, темп речи и паузация. К спектральным признакам можно отнести различные кепстральные (например, мел-кепстральные) коэффициенты, частоты первых формант речевого сигнала и их среднеквадратические отклонения и пертурбационные параметры (джиттер и шиммер).

1.3.1. Просодические характеристики

1.3.1.1. Частота основного тона

Значение частоты основного тона зависит от размеров и степени натяжения связок. [21] Кроме самой частоты основного тона оценивается ее среднеквадратическое отклонение. В таблице 1.2 представлена связь между эмоцией и изменением этих характеристик относительно безэмоциональной речи.

Таблица 1.2 – Связь характеристик частоты основного тона и эмоции

Базовая эмоция	*Изменение значений относительно нейтрального состояния*	
	ЧОТ	*СКО ЧОТ*
Радость	выше	выше
Печаль	ниже	ниже
Гнев	ниже	выше
Удивление	ниже	выше
Страх	выше	выше

Методы определения частоты основного тона можно разделить на три категории: основанные на временной динамике сигнала (*англ. time-domain*), основанные на частотной структуре (*англ. frequency-domain*) и комбинированные методы [22].

Перед применением методов, основанных на временной динамике, сигнал предварительно фильтруют, оставляя только низкие частоты. Задаются минимальная и максимальная частоты (например, от 75 до 500 Гц). Частота основного тона не определяется для участков, содержащих негармоничную речь (паузы, шумовые звуки), поскольку это влечет за собой ошибки, которые могут распространяться на соседние фреймы при применении интерполяции или сглаживания. Длину кадра выбирают так, чтобы в ней содержалось как минимум три периода.

В методах, основанных на частотной структуре, анализируется гармоническая структура сигнала. Одним из таких методов является кепстральный ана-

лиз.

Гибридные методы определения имеет смысл рассмотреть на примере алгоритма YAAPT (*англ. Yet Another Algorithm of Pitch Tracking*), который считается гибридным, поскольку использует как частотную, так и временную информацию. YAAPT, как и другие алгоритмы определения ЧОТ состоит из трех этапов: препроцессирование, поиск кандидатов (возможных значений ЧОТ) и выбор наиболее вероятной траектории ЧОТ.

На этапе препроцессирования значения изначального сигнала возводят в квадрат для усиления и восстановления пиков автокорреляции. Затем по спектру преобразованного сигнала рассчитывается базовая траектория ЧОТ. Кандидаты определяются с помощью функции SHC – *от англ. Spectral Harmonics Correlation* согласно 1.1:

$$\mathrm{SHC}(t,\, f) = \sum_{f'=-\mathrm{WL}/2}^{\mathrm{WL}/2} \prod_{r=1}^{\mathrm{NH}+1} S(t,\, rf + f'), \tag{1.1}$$

где $S(t,\, f)$ — магнитудный спектр для фрейма t и частоты f, WL — длина окна (Гц), NH – число гармоник (рекомендуется [23] использовать первые три гармоники).

Далее, как для изначального сигнала, так и для преобразованного производится определение кандидатов на F0, и вместо автокорреляционной функции здесь используется функция NCCF (*от англ. Normalized Cross Correlation*) согласно уравнению 1.2:

$$\mathrm{NCCF}(m) = \frac{\sum\limits_{n=0}^{N-m-1} x(n) \cdot x(n+m)}{\sqrt{\sum\limits_{n=0}^{N-m-1} x^2(n) \cdot \sum\limits_{n=0}^{N-m-1} x^2(n+m)}},\ 0 < m < M_0, \tag{1.2}$$

Далее проводится оценка всех возможных кандидатов и вычисление их веса (*англ. merit*). Вес кандидатов, полученных по аудиосигналу, от амплитуды пика NCCF и от их близости к траектории ЧОТ, определенной по спектру.

Затем для всех пар оставшихся кандидатов рассчитывается матрица цены перехода (*англ. Transition Cost*), по которой находят оптимальную траекторию ЧОТ.

1.3.1.2. Инетнсивность, темп речи и паузация

Громкость (интенсивность) речи измеряется в децибелах (дБ). Связь градаций интенсивности и эмоции представлена в таблице 1.3.

Таблица 1.3 – Связь градаций интенсивности и эмоции

Интенсивность	*Значение* (дБ)
Шепот	< 20
Значительное снижение	20. . . 40
Умеренное снижение	40. . . 50
Нормальная	50. . . 80
Умеренное повышение	80. . . 90
Значительное повышение	90. . . 110
Крик	> 110

Следующей просодической характеристикой является паузация. Короткими паузами принято считать паузы до 3 секунд, средними – от 3 до 7 секунд, длинными – свыше 7 секунд. В таблице 1.4 представлена связь характеристик паузации и эмоции.

Таблица 1.4 – Связь характеристик паузации и эмоции

Базовая эмоция	*Изменение значений относительно нейтрального состояния*	
	количество пауз	длина пауз
Радость	меньше	короче
Печаль	больше	короче
Гнев	меньше	короче
Удивление	больше	длиннее

Темпом речи называют скорость произнесения элементов речи (звуков, слогов, слов). Темп речи изменяется двумя параметрами [24]:

– числом произносимых в единицу времени элементов речи;

– средней длительностью элемента.

Повышение темпа речи осуществляется за счет сокращения длительности гласных и согласных звуков, снижение темпа достигается путем увеличения длительности гласных. Согласно исследованиям в [25], изменение темпа речи связано с проявлением говорящим эмоции. Например, проявление тревожности характеризуется ускорением темпа речи, а холодность и задумчивость – замедлением.

1.3.2. Спектральные характеристики

1.3.2.1. Мел-кепстральные коэффициенты

Основной принцип работы с человеческой речью заключается в том, что звуки, генерируемые человеком, фильтруются формой голосового тракта (язык, зубы и т.д.). Набор этих характеристик и можно представить с помощью мел-кепстральных коэффициентов (*англ. MFCC*) [26].

Шкала Мел (рисунок 1.1) соотносит воспринимаемую частоту или высоту чистого тона (мел) с фактической измеренной частотой (Гц). Люди гораздо лучше различают небольшие изменения высоты звука на низких частотах, чем на высоких [27].

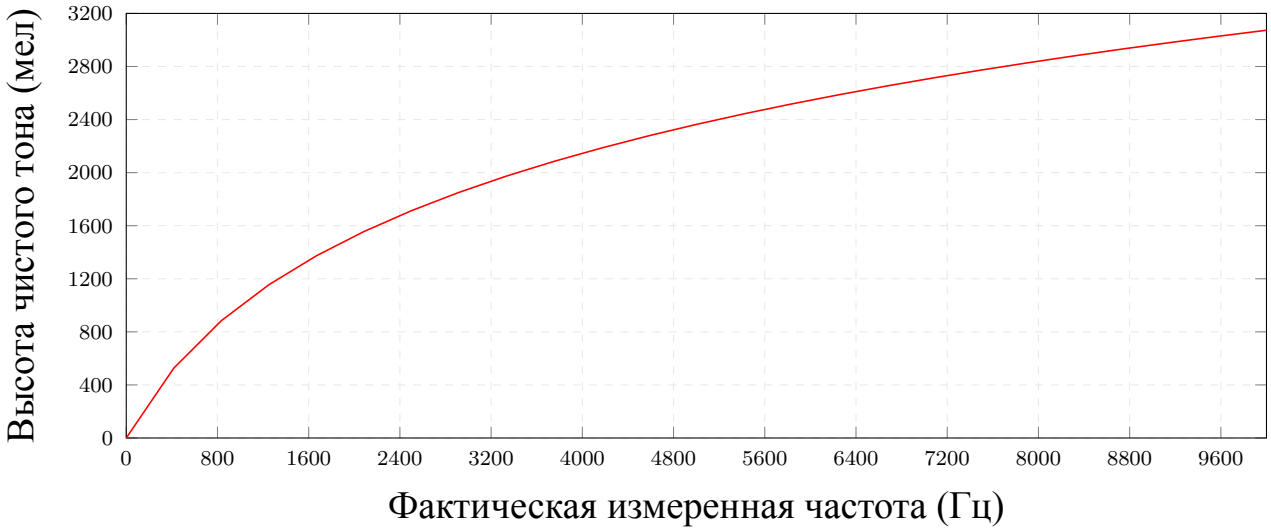

Рис. 1.1 – График зависимости частоты от мел

Вычисление мел-кепстральных коэффициентов заключается в следующем. Для каждого фрейма $x_j(n)$ выполняется дискретное преобразование Фу-

рье (1.3):

$$X_j(k) = \sum_{n=0}^{N-1} x_j(n) w(n) \exp -\frac{2\pi i}{N} kn,\ 0 \le k < N, \tag{1.3}$$

где j – номер фрейма, $w(n)$ – оконная функция Хэмминга, используемая для уменьшения утечки ДПФ на интервале конечной длительности.

Следующим шагом вычисляется банк мел-фильтров из M треугольных фильтров. Для этого треугольные фильтры умножаются на периодограмму и суммируются. Каждый треугольный фильтр моделируются с помощью функции 1.4:

$$H_m(k) = \begin{cases} 0, & k < f(m-1), \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \le k < f(m), \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)}, & f(m) \le k < f(m+1), \\ 0, & k > f(m+1). \end{cases} \tag{1.4}$$

Далее производится расчет логарифмического значения энергии компонент спектра на выходе каждого фильтра (1.5):

$$T_j(m) = \ln \sum_{k=0}^{N-1} P_j(k) H_m(k),\ 0 \le m < M. \tag{1.5}$$

Поскольку ДПФ характеристик синтезированных фильтров 1.4 взаимно пересекаются, а энергии на выходе фильтров существенно коррелируют, для вычисления MFCC необходимо использовать дискретное косинусное преобразование (1.6), чтобы устранить возникающие корреляци:

$$c_j(m) = \sum_{m=0}^{M-1} T_j(m) \cos \left(\frac{\pi n \left(m + \dfrac{1}{2} \right)}{M} \right),\ 0 \le n < M. \tag{1.6}$$

После получения $c_j(m)$, коэффициент $c_j(0)$ отбрасывается, так как он не несет информации о речи и задает постоянное смещение [28].

1.3.2.2. Энергетические и пертурбационные параметры

Выделяют следующие энергетические параметры речи: нижний уровень громкости речи (I_{lower}), верхний уровень громкости речи (I_{higher}), динамический диапазон (ΔI) и амплитуда основного тона (I_{pinch}). Энергетические параметры речи измеряются в децибелах (дБ).

Пертурбационными параметрами называют джиттер и шиммер. Существует несколько параметров их оценки. Локальный джиттер определяется согласно 1.7:

$$\text{jitter}_{\text{loc}} = \frac{1}{N-1}\sum_{i=1}^{N-1}|F_0(i) - F_0(i+1)| \Big/ \frac{1}{N}\sum_{N}^{i=1} F_0(i), \tag{1.7}$$

где $F_0(i)$ – длительность i-го периода основного тона, N – число периодов основного тона.

Другой способ определения джиттера – использование отклонения текущей длительности периода не от предыдущей, а от локально усредненного значения, которое рассчитывается на окне в 3 или 5 выборок [29]:

$$\text{jitter}_{\text{P}} = \frac{1}{N-P+1}\sum_{i=1+\frac{P-1}{2}}^{N-\frac{P-1}{2}} \left| F_0(i) - \frac{1}{P}\sum_{n=i-\frac{P-1}{2}}^{i+\frac{P-1}{2}} F_0(n) \right| \Big/ \frac{1}{N}\sum_{i=1}^{N}|F_0(i)|, \tag{1.8}$$

где P – параметр, определяющий количество периодов используемых для вычисления среднего.

Аналогично джиттеру, для вычисления шиммера существует несколько способов. Простейший – определение средней абсолютной разницы между амплитудами последовательных периодов основного тона $A(i)$, деленной на среднюю амплитуду (выражение 1.9):

$$\text{shimmer}_{\text{loc}} = \frac{1}{N-1}\sum_{i=1}^{N-1}|A(i) - A(i+1)| \Big/ \frac{1}{N}\sum_{N}^{i=1} A(i). \tag{1.9}$$

Существуют варианты, определения шиммера как относительное отклонение амплитуды от локально-усредненного значения на интервале в P выбо-

рок(1.10):

$$\mathrm{shimmer_P} = \frac{1}{N-P+1} \sum_{i=1+\frac{P-1}{2}}^{N-\frac{P-1}{2}} \left| A(i) - \frac{1}{P} \sum_{n=i-\frac{P-1}{2}}^{i+\frac{P-1}{2}} A(n) \right| \Big/ \frac{1}{N} \sum_{i=1}^{N} |A(i)| \,. \tag{1.10}$$

Такая оценка более точна чем 1.9, поскольку на шиммер влияет постепенный равномерный (естественный) спад интенсивности голоса, создающий эффект «дрейфа» амплитуды сигнала. [30]

1.3.2.3. Частоты первых формант речевого сигнала

Форманты возникают под влиянием резонаторов речевого аппарата, поэтому их высотное положение не зависит от основного тона, но зависит от произносимого звука. При произнесении некоторых звуков речи (в основном гласных) происходит усиление обретонов на определенной частоте: например, при произнесении гласной «у» характерно усиление частичных тонов от 200 до 400 Герц, а для гласной «о» – от 400 до 600 Герц.

Определение формант возможно с помощью кепстрального анализа. Речь можно представить как результат свертки инициирующего сигнала гортанных импульсов с частотной характеристикой голосового тракта, выступающего в роли линейного фильтра (1.11):

$$x(t) = s(t) * h(t), \tag{1.11}$$

где $s(t)$ - инициирующий сигнал, $h(t)$ – частотная характеристика голосового тракта. Представляя сигнал во временной области, при переходе к лог-спектру (1.12) и кепстру (1.13) можно получить следующие выражения:

$$\log X(\omega) = \log S(\omega) + \log H(\omega), \tag{1.12}$$

$$X(\bar{\omega}) = S(\bar{\omega}) + H(\bar{\omega}). \tag{1.13}$$

То есть свертка сигналов во временной области эквивалентна их умножению в частотной области или сложению в лог-спектре и кепстре. Это свойство кепстра позволяет разделить инициирующий сигнал и частотную характеристику голосового тракта. При этом информация о тембре и формантах будет находиться на

низких quefrency-значениях (анаграмма от англ. *frequency*, дословно – сачтотах) кепстра, а информация о инициирующем сигнале - на высоких.

Чтобы извлечь тот или иной компонент сигнала, нужно применить т.н. liftering (анаграмма от англ. *filtering*, фильтрация) - аналог частотных фильтров для quefrency-области. Для получения информации о формантах можно использовать следующий фильтр низких quefrency-значений:

$$l(n) = \begin{cases} 1, n < \tau, \\ -1, n \geq \tau. \end{cases} \tag{1.14}$$

Для мужской речи значение τ выбирают в диапазоне от 4 до 8 мс, так как среднее значение частоты мужской речи находится в диапазоне от 128 до 270 Гц. Для женской речи, из-за более высокого среднего значения частоты (256-310 Гц), диапазон, на котором расположены форманты в кепстре, может пересекаться с диапазоном инициирующего сигнала, что мешает однозначному разделению компонент речи.

1.4. Классификаторы, используемые в анализе речи

Автоматическое определение эмоций происходит с помощью классификатора – обученной модели, которая после обучения сможет определять эмоции по записям человеческой речи. Обучение классификатора происходит с использованием одного набора данных на одном языке. Существует множество классификаторов, используемых в схожих с исследуемой задачах – модель гауссовых смесей, метод опорных векторов, случайный лес. Однако, наиболее популярны при распознавании эмоций модели – это скрытая марковская модель (*англ. Hidden Markov Model, HMM*) и искусственная нейронная сеть (*англ. Artificial Neural Networks, ANN*).

1.4.1. Скрытая марковская модель

Скрытые марковские модели предоставляют наиболее подходящий инструмент моделирования в ситуациях когда состояния не являются непосредственно наблюдаемыми. Более детальное объяснение скрытых марковских моделей имеет смысл начать с определения марковских цепей.

Марковская цепь – это конечный автомат, имеющий дискретное число со-

стояний $q_1, \ldots, q_n$ и существует вероятность перехода из состояния q_i в другое состояние q_j: $P\left(S_t = q_j | S_{t-1} = q_i\right)$. Цепь может находиться в состоянии q_i в любой момент времени t, поскольку вермя дискретно. Также согласно марковскому свойству, вероятность следующего состояния зависит только от вероятности предыдущего.

Скрытая марковская модель – это марковская цепь, в которой состояния не являются непосредственно наблюдаемыми. Такую модель можно объяснить как двойной стохастический процесс: скрытый стохастический процесс, который невозможно наблюдать напрямую и процесс, который создает последовательность наблюдений с учетом первого процесса.

При применении скрытой марковской модели при классификации выделяют три основных задачи.

1. *Задача вычисления оценки*. В рассматриваемой модели необходимо определить оценку вероятности последовательности наблюдений.

2. *Задача определения оптимальной последовательности*. С учетом модели и конкретной последовательности наблюдений необходимо определить оценку наиболее вероятной последовательности состояний, которая создает эти наблюдения.

3. *Задача обучения параметров*. С учетом количества последовательностей наблюдений необходимо отрегулировать параметры модели.

Для того, чтобы формализовать представленные задачи, следует ввести следующие обозначения. Модель $\lambda = (A,\ B,\ \pi)$ состоит из матрицы перехода A, матрицы наблюдаемых значений B и начального распределения π. Последовательность наблюдаемых значений D выбирается из алфавита (множества значений скрытых параметров) V.

Суть первой задачи заключается в определении вероятности последовательности наблюдений D по параметрам данной модели $\lambda = (A,\ B,\ \pi)$. Для вычисления вероятности последовательности наблюдений можно вычислить ее оценку для конкретной последовательности состояний, а затем прибавить вероятности для всех возможных последовательностей состояний согласно 1.15:

$$P(D|\lambda) = \sum_{Q} P(D|\, Q,\ \lambda) p(Q|\, \lambda). \tag{1.15}$$

Вторая задача заключается в том, чтобы по модели λ и последовательности D найти оптимальную последовательность состояний Q. Третья задача – главная, заключается в оптимизации параметров модели $\lambda = (A, B, \pi)$ таким образом, чтобы минимизировать $p(D|\lambda)$ при данных D, т.е. найти модель максимального правдоподобия.

1.4.2. Искусственная нейронная сеть

Нейронная сеть – математическая модель, построенная на принципах функционирования биологических нейросетей (сетей нервных клеток живого организма) [31]. Основная парадигма нейронных сетей – это формирование решения из множество простых элементов, подобных нейронам. Эти элементы образуют граф с взвешенными синаптическими связями. Искусственная нейронная сеть обладает следующими свойствами [32]:

– параллельность – в любой момент времени в активном состоянии могут находиться несколько процессов;

– распределенность – каждый из процессов может независимо обрабатывать локальные данные;

– свойство самообучения и подстройки своих параметров при изменении профиля данных.

Единицу, выполняющую вычисления в нейронной сети, называют нейроном. Нейроны обрабатывают входной сигнал и отправляют его дальше по сети. Нейрон представляет собой некую функцию от линейной комбинации всех своих входных сигналов. Основная функция нейрона - сформировать выходной сигнал y в зависимости от сигналов $x_1, \ldots, x_N$, поступающих на его входы. Входные сигналы обрабатываются адаптивным сумматором (1.16):

$$\sum_{i=1}^{N} w_i x_i - T, \tag{1.16}$$

где T – порог нейрона, $w_1, \ldots, w_N$ – знаки весов синапсов. Выходной сигнал поступает в нелинейный преобразователь F с некоторой функцией активации, после чего результат подается на выход (в точку ветвления).

Синапс – связь между нейронами, причём каждый синапс имеет свой вес. Благодаря этому входные данные видоизменяются при передаче. Во время обработки переданная синапсом информация с большим показателем веса станет преобладающей.

Схема классификации на основе нейронных сетей включает в себя следующие шаги.

1. На входной слой нейронов происходит поступление определённых данных.

2. Информация передаётся с помощью синапсов следующему слою, причём каждый синапс имеет собственный коэффициент веса, а любой следующий нейрон способен иметь несколько входящих синапсов. Данные, полученные следующим нейроном – это сумма всех данных для нейронных сетей, которые перемножены на коэффициенты весов.

3. Полученное в итоге значение подставляется в функцию активации (для нормализации входных данных), в результате чего происходит формирование выходной информации.

4. Информация передаётся дальше до тех пор, пока не дойдёт до конечного выхода.

1.5. Постановка задачи

В настоящее время ряд задач, связанных с распознаванием эмоций в речи решается с использованием нейронных сетей или статистических классификаторов. В настоящей работе проектируется метод решения этой задачи с использованием статистического классификатора, а именно – скрытой марковской модели.

Корпус данных должен быть подготовлен к использованию классификатором – а именно, должно быть установлено однозначное соответствие аудиофайл-эмоция согласно представленной в наборе разметке. Под исследуемую задачу по разметке и по языку подходят следующие наборы: RUSLANA и DUSHA. Однако, корпуса RUSLANA на данный момент нет в открытом доступе.

Из аудиозаписей должны быть выделены информативные признаки. Поскольку сигнал должен быть разбит на кадры небольшой длительности, использование просодических признаков не имеет смысла, следовательно, будут использованы спектральные признаки речевого сигнала. В качестве спектральных признаков, извлекаемых из аудиофайлов, было решено использовать мел-кепстральные коэффициенты, поскольку они представляют собой широкий спектр признаков, которые могут быть маркерами конкретной эмоции, таких как громкость, длительность, скорость и ритм. Для обучения классификатора эти признаки должны быть поделены на кластеры. Обученный классификатор используется для распознавания эмоций.

На рисунке 1.2 представлена IDEF0-диаграмма нулевого уровня решаемой задачи.

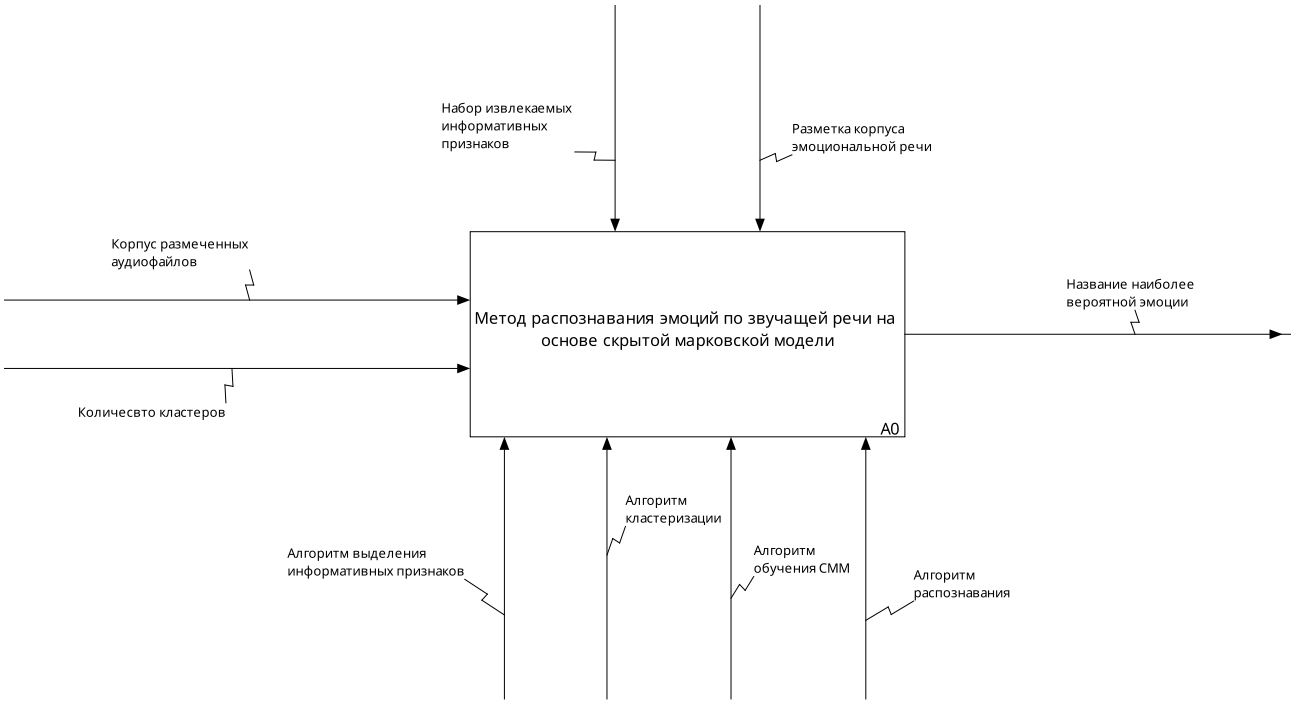

Рис. 1.2 – IDEF0-диаграмма нулевого уровня

2 Конструкторский раздел

2.1. Общая схема метода

Задача распознавания эмоций по звучащей речи сводится к соотношению исходных данные на входе (аудиозаписи звучащей речи) к определенному классу на выходе (виду эмоции). На рисунке 2.1 представлена IDEF0-диаграмма первого уровня решаемой задачи.

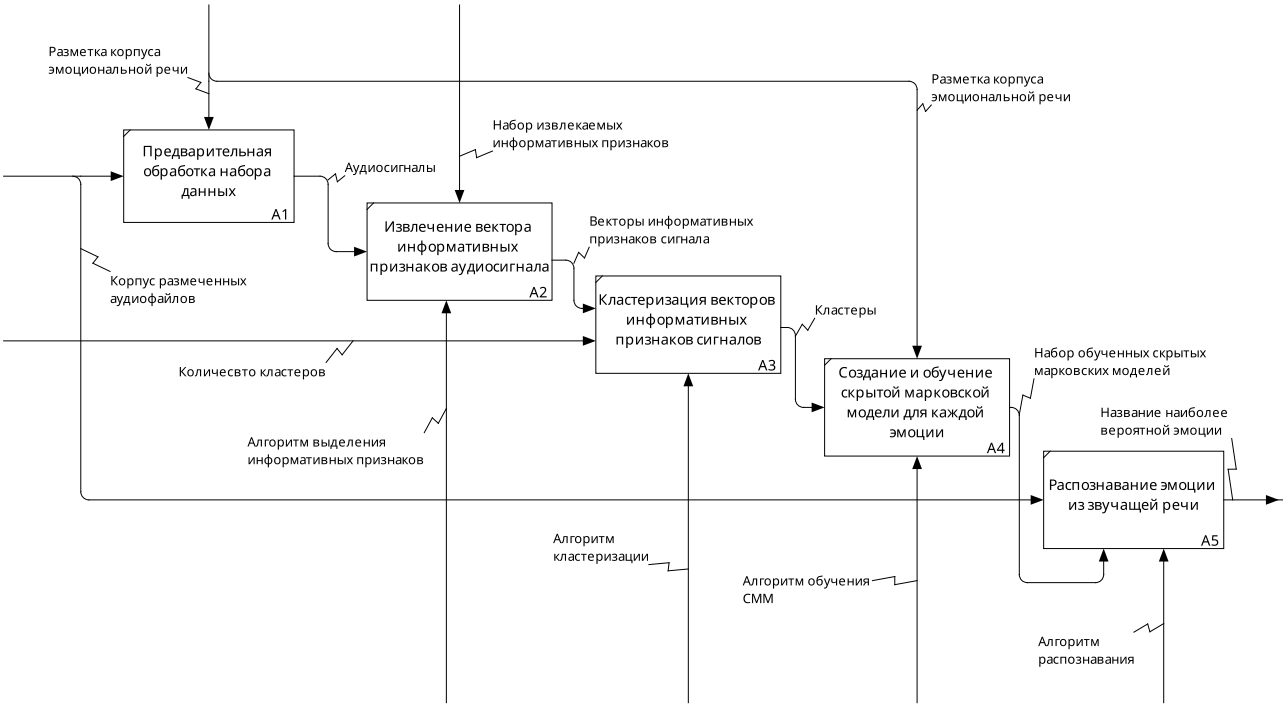

Рис. 2.1 – IDEF0-диаграмма первого уровня

Прежде всего набор данных должен быть предварительно обработан (блок A1). Предварительная обработка включает в себя разделение набора данных на тренировочную (70-80% набора) и тестовую (20-30% набора) выборки. Следующий этап решения задачи (блок A2) – это извлечение информативных признаков из речевого сигнала. Далее необходимо сократить размерность вектора информативных признаков. Для этого производится кластеризация векторов признаков сигналов (блок A3). Этап создания и обучения скрытой марковской модели для каждой эмоции (блок A4) включает в себя определение количества состояний модели, а также вероятностей перехода между состояниями и вероятностей наблюдения признаков в каждом состоянии. В качестве состояний модели будут использованы порядковые номера выделенных кластеров. Обученную марковскую модель можно использовать для распознавания эмо-

ций в звучащей речи (блок А5).

2.2. Проектирование ключевых модулей системы

2.2.1. Формирование вектора информативных признаков

Самым надежным решением при распознавании эмоций в речи принято считать использование мел-кепстральных коэффициентов в качестве информативных признаков [33]. В данной работе решено использовать первые 13 мел-кепстральных коэффициентов.

Формирование вектора информативных признаков, состоящего из 13-ти первых мел-кепстральных коэффициентов представлено на рисунке 2.2.

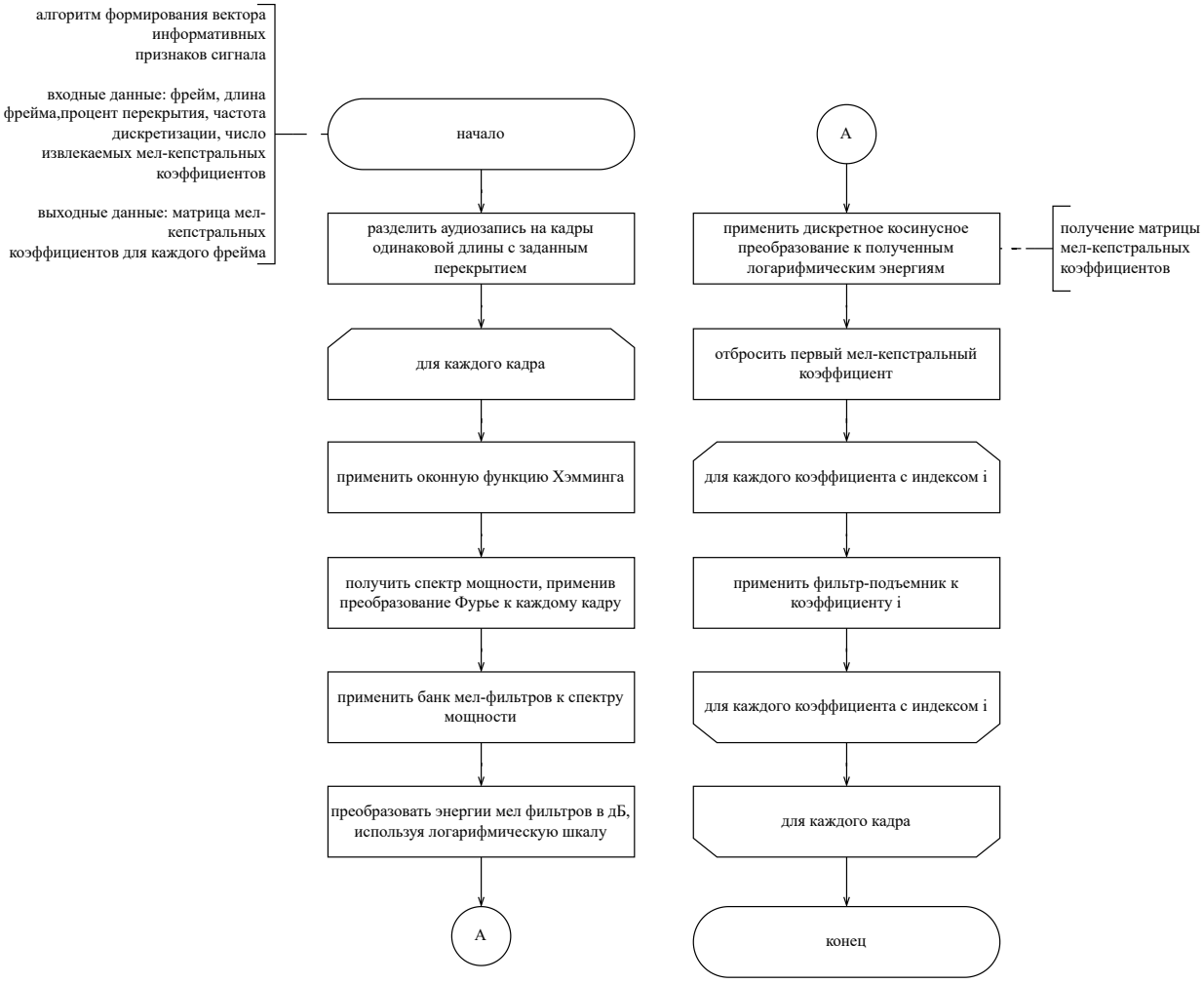

Рис. 2.2 – Алгоритм формирования вектора информативных признаков

Фильтр-подъемник (lifter) применяется для усиления высокочастотных компонентов сигнала, которые могут быть утрачены в процессе преобразования сигнала в мел-кепстральные коэффициенты. Применение фильтра-подъемника

происходит согласно 2.1

$$1 + \frac{\text{lifter}}{2} \cdot \frac{\sin\left(\frac{\pi i}{\text{lifter}}\right)}{1}, \tag{2.1}$$

где i - индекс коэффициента MFCC, а lifter - значение параметра фильтра-подъемника.

2.2.2. Кластеризация

Алгоритм k-means – это неиерархичный метод неконтролируемого обучения. Он позволяет разделить произвольный набор данных на заданное число кластеров так, что объекты внутри одного кластера были достаточно близки друг к другу, а объекты разных не пересекались. Цель этого алгоритма — объединить в группы сходные данные по некоторым заданным критериям. Чаще всего при кластеризации используются меры расстояния. В данной работе в качестве меры расстояния будет реализовано Евклидово (квадратичное) расстояние согласно 2.2:

$$\text{dist}(p,\ q) = \sqrt{(p-q)^2}, \tag{2.2}$$

где p, q – точки вектора входных данных. Точка вектора входных данных имеет 13 измерений, поскольку точкой в данном случае являются значения мел-кепстральных коэффициентов, наблюдаемых на каждом фрейме каждого сэмпла.

Результатом работы алгоритма кластеризации является массив 13-ти мерных точек, являющихся центроидами каждого кластера. После получения центроидов необходимо определить ближайший центроид для каждого набора мел-кепстральных коэффициентов, полученного в модуле выделения информативных признаков. В результате работы всего модуля устанавливается соответствие «кластер-фрейм», необходимое для дальнейшего обучения скрытой марковской модели.

Схема алгоритма кластеризации представлена на рисунке 2.3.

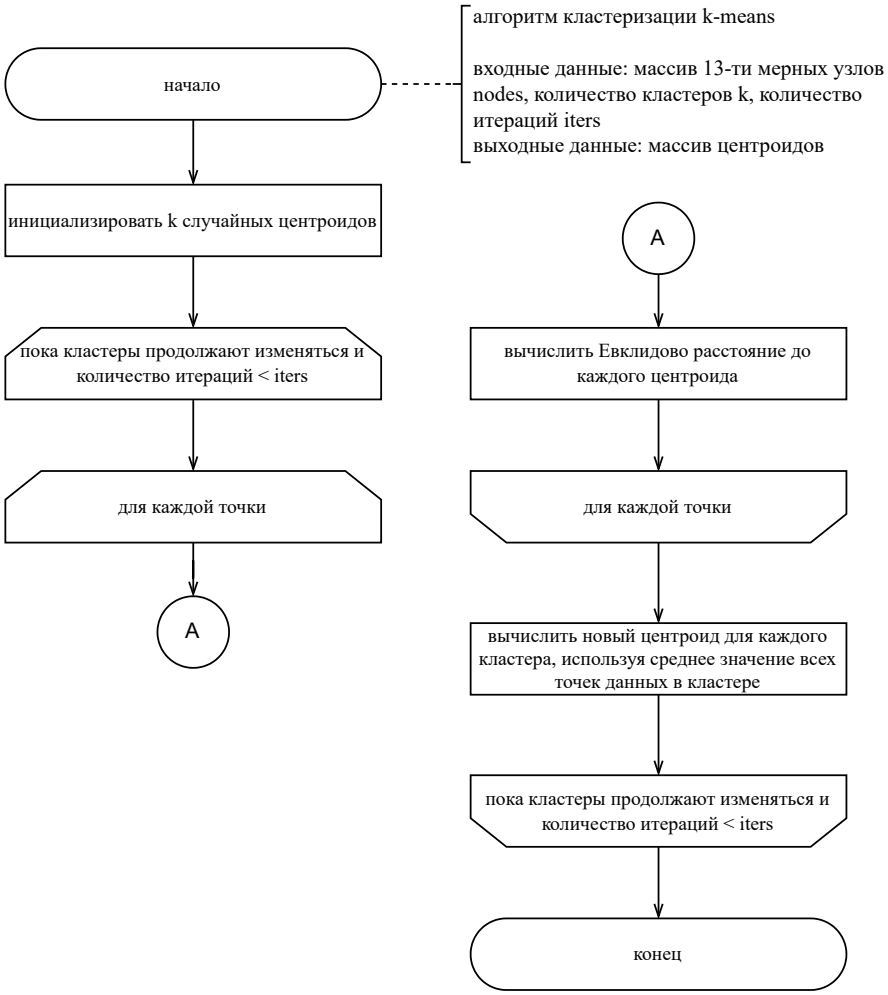

Рис. 2.3 – Алгоритм кластеризации k-means

2.2.3. Создание и обучение скрытых марковских моделей

Обучение скртыой марковской модели – это определение параметров $\lambda = \{A, B, \pi\}$ с учетом количества последовательностей наблюдений $\{O = O_1, \ldots, O_n\}$. Для обучения скрытой марковской модели используется алгоритм Баума — Велша. Стоит отметить, что он применим только в том случае, если предварительно определено количество состояний и наблюдений. Пусть γ – условная вероятность нахождения в определенном состоянии q_i в с учетом последовательности наблюдений (2.3):

$$\gamma_i = P(s_t = q_i \,|\, O,\, \lambda) = \frac{P(s_t = q_i,\, O \,|\, \lambda)}{P(O)}. \tag{2.3}$$

Далее следует ввести переменную α, обозначающую вероятность частичной последовательности наблюдений до момента времени t, находящаяся в состоянии

q_i в момент времени t, согласно 2.4:

$$\alpha_i = P(O_1, O_2, \ldots, O_t,\ S_t = q_i \mid \lambda), \tag{2.4}$$

и переменную β, вероятность частичной последовательности наблюдений от $t+1$ до T, при нахождении в состоянии q_i в момент времени t (2.5):

$$\beta_i = P(O_{t+1}, O_{t+2}, \ldots, O_T,\ S_t = q_i \mid \lambda), \tag{2.5}$$

С учетом этих обозначений следует ввести переменную ξ, обозначающую вероятность перехода из состояния i в момент времени t в состояние j в момент времени $t+1$ с учетом последовательности наблюдений O согласно 2.6:

$$\xi_t(i,\ j) = \frac{\alpha_t(i) a_{i,\,j} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O)}. \tag{2.6}$$

С учетом переменных α и β (2.4 и 2.5) задать ξ удобнее согласно 2.7:

$$\xi_t(i,\ j) = \alpha_t(i) a_{i,\,j} b_j(O_{t+1} \beta_{t+1}(j) \Big/ \sum_i \sum_j \alpha_t(i) a_{i,\,j} b_j(o_{t+1}) \beta_{t+1}(j). \tag{2.7}$$

С учетом введенных обозначений, Баума—Велша можно разделить на 5 этапов.

1. Выполнение прямого прохода, в результате которого вычисляются вероятности наблюдаемых последовательностей до каждого момента времени и вероятности переходов из одного скрытого состояния в другое.

2. Выполнение обратного хода, в результате которого вычисляются апостериорные вероятности скрытых состояний в каждый момент времени.

3. оценка априорных вероятностей – количество случаев нахождения в состоянии i в момент времени t согласно 2.8:

$$\pi_i = \gamma_1(i). \tag{2.8}$$

4. Оценка вероятностей переходов – количество переходов из состояния i в состояние j по отношению к количеству случаев нахождения в состоянии

i согласно 2.9:

$$a_{i,j} = \frac{\sum_{t=1}^{T} \gamma_j(t) 1(v(t) = k)}{\sum_{t=1}^{T} \gamma_j(t)}. \quad (2.9)$$

5. Оценка вероятностей наблюдений – количество случаев нахождения в состоянии j при количестве наблюдений k по отношению к количеству случаев нахождения в состоянии j согласно 2.10:

$$b_{j,k} = \frac{\sum_{t=1}^{T} \gamma_j(t) 1(v(t) = k)}{\sum_{t=1}^{T} \gamma_j(t)}. \quad (2.10)$$

Алгоритм обучения скрытых марковских моделей представлен на рисунке 2.4.

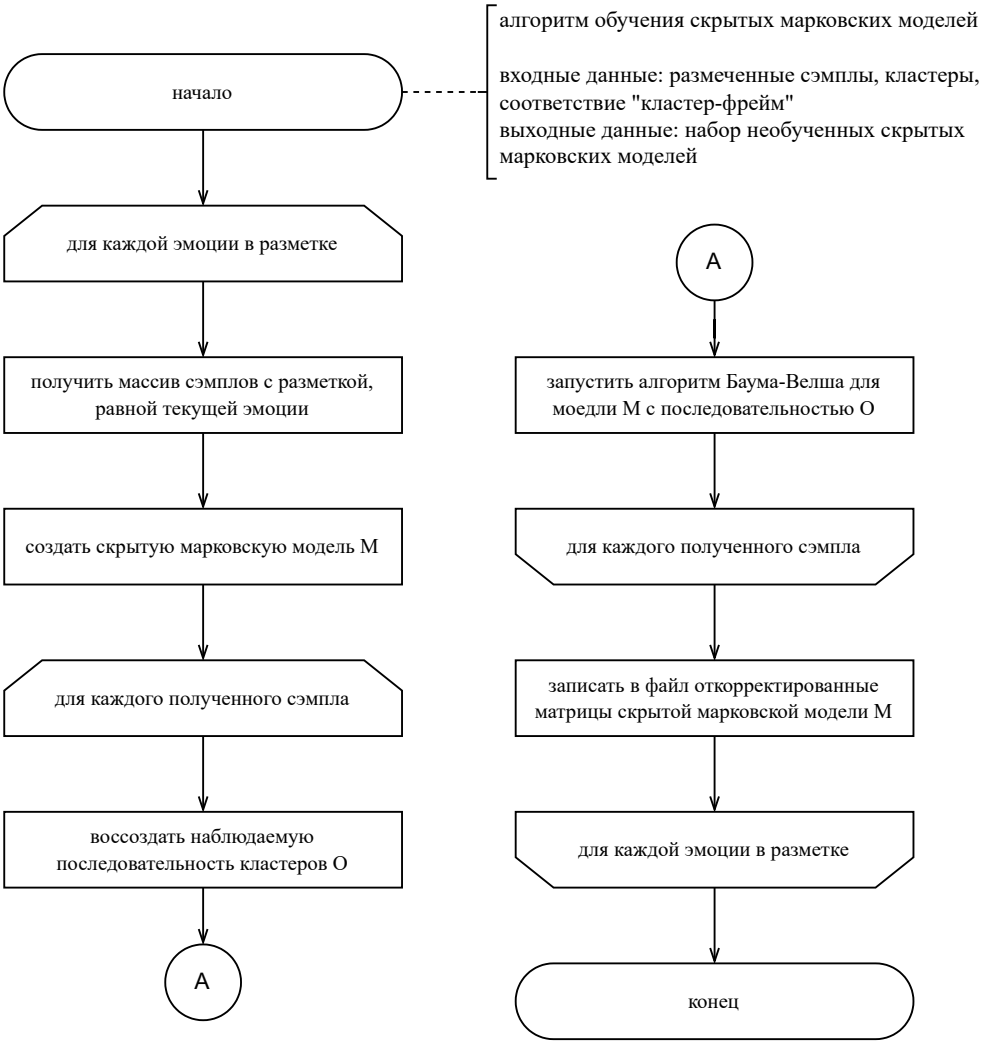

Рис. 2.4 – Обучение скрытых марковских моделей

2.2.4. Определение эмоции из аудиосигнала

Для того, чтобы выделить эмоцию из семпла, необходимо определить вероятности наблюдения последовательности кластеров семпла для каждой марковской модели. В алгоритме прямого прохода вычисляются не только вероятности α, но также и вероятность наблюдать данную последовательность при условии заданной модели. Алгоритм представлен на листинге 2.1.

Листинг 2.1: Алгоритм прямого хода

Исходные параметры: Скрытая марковская модель λ; Последовательность наблюдений O; количество состояний N; Количество наблюдений T

$i \leftarrow 0$; **цикл** $i < T$ **выполнять**
  $\alpha_1(i) = \pi_i b_i(O_1)$; // инициализация
конец цикла
$t \leftarrow 1$; **цикл** $t < T$ **выполнять**
  $j \leftarrow 0$; **цикл** $j < N$ **выполнять**
    $\alpha_t(j) = \sum_i \alpha_{t-1}(i) a_{i,j} b_j(O_t)$ // индукция
  конец цикла
конец цикла
$P(O) = \sum_i \alpha_T(i)$

Алгоритм прямого прохода состоит из трех основных частей: инициализация, индукция и завершение. На этапе инициализации определяются переменные α для всех состояний в начальный момент времени. На этапе индукции вычисляются значения $\alpha_{t+1}(i)$ по значениям $\alpha_t(i)$. На этапе завершения вычисляется значение $P(O \mid \lambda)$ путем суммирования всех значений α_T.

Распознавание происходит следующим образом: для каждого семпла выделяется последовательность наблюдений, состоящая из номеров кластеров, присвоенных каждому фрейму семпла на этапе кластеризации. Для каждой обученной модели запускается алгоритм прямого прохода, из которого определяются вероятности наблюдения данной последовательности кластеров с учетом модели. Полученные вероятности сравниваются. Та модель, для которой вероятность наблюдения максимальная, считается подходящей. Алгоритм распозна-

вания представлен на риснуке 2.5.

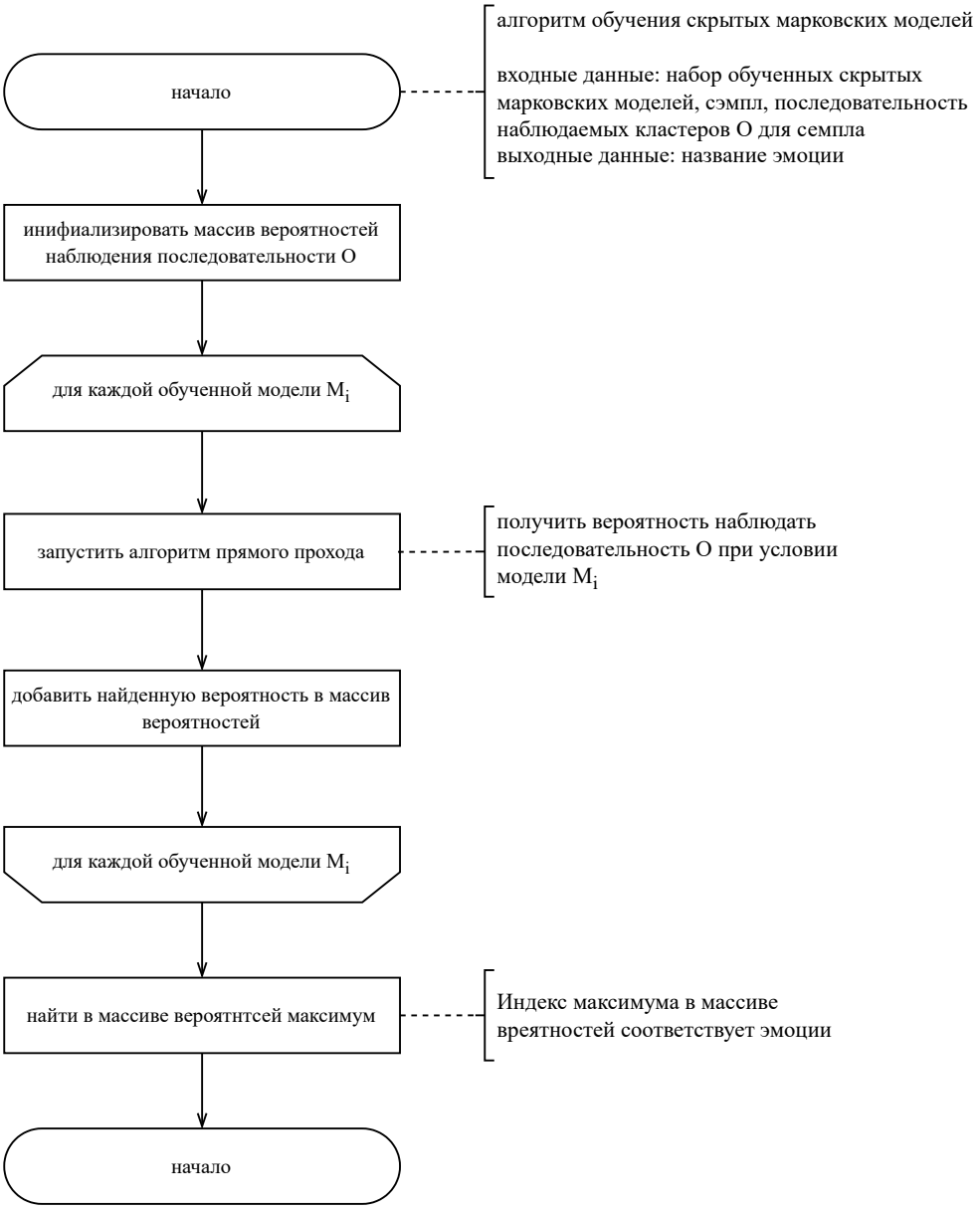

Рис. 2.5 – Выделение эмоции из аудиосигнала

2.3. Описание используемого набора данных

2.3.1. Разметка и структура набора

Для обучения классификатора было решено использовать набор данных DUSHA [15], содержащий записи эмоциональной речи. Набор разделен на два домена – аудиоматериалы, собранные с помощью краудсорсинга (*англ. «Croud»*) и выдержки из русскоязычных подкастов (*англ. «Podcast»*).

При разметке эмоциональных наборов данных существует сложность в неоднозначности интерпретации эмоции аннотаторами. В наборе данных

DUSHA эта проблема решается привлечением к оцениванию двух независимых экспертных групп. В наборе присутствуют только те записи, для которых оценка обоих групп была консистентна. В разметке присутствуют следующие классы эмоций:

- **нейтраль**;
- **позитив**: текст требовалось произносить с улыбкой или смехом, стараться делать выраженные ударения на позитивно окрашенных словах;
- **грусть**: произносить текст требовалось приглушенным голосом, меланхолично;
- **злость или раздражение**: текст требовалось произнести с криком или сквозь зубы, и, аналогично позитиву, стараться делать выраженные ударения на негативно окрашенных словах.

В домене «Crowd» тексты были искусственно сгенерированы на основе записей общения с голосовым ассистентом. Затем они были озвучены двумя способами: эмоционально, с той эмоцией, которую показал на этой записи классификатор BERT [34] и безэмоционально. Домен «Podcast» содержит уже не имитацию эмоции, а естественную речь. Все аудиозаписи были сделаны на профессиональные микрофоны, качество аудио было унифицировано до 16кГц. Аннототоры производили разметку, опираясь исключительно на звуковую дорожку, без учета произнесённого в ней текста.

2.3.2. Содержание набора данных

Объём данных после агрегации с разбивкой по подмножествам приведён в таблице 2.1.

Таблица 2.1 – Объём данных после агрегации

Домен	*Тренировочная выборка*		*Тестовая выборка*	
	Файлы (шт.)	Время	Файлы (шт.)	Время
Crowd	147057	184 ч. 21 мин.	13867	18 ч. 17 мин.
Podcast	78810	70 ч. 08 мин.	10591	09 ч. 24 мин.
Всего	225867	254 ч. 29 мин.	24458	27 ч. 41 мин.

Поскольку разметку осуществляли несколько аннотаторов, требовалась агрегарция разметки. Она производилась по методу Дэвида — Скина с порогом 0.9, выбранным эмпирически [15]. В агрегированном наборе данных имеются следующие поля разметки:

- **audio_path:** путь к аудиофайлу;
- **emotion:** эмоция, которую указал разметчик;
- **speaker_text:** текст, который произнёс диктор (присутствует только в домене Crowd);
- **speaker_emo**: эмоция, которую выражал диктор (присутствует только в домене Crowd);
- **source_id**: уникальный идентификатор диктора или подкаста.

2.4. Проектирование отношений сущностей

На каждом этапе решения поставленной задачи формируется большой объем связанных структурированных данных, который необходимо хранить и дополнять. Для хранения этого набора решено использовать реляционную базу данных. На рисунке 2.6 представлена диаграмма сущностей базы данных в нотации Чена.

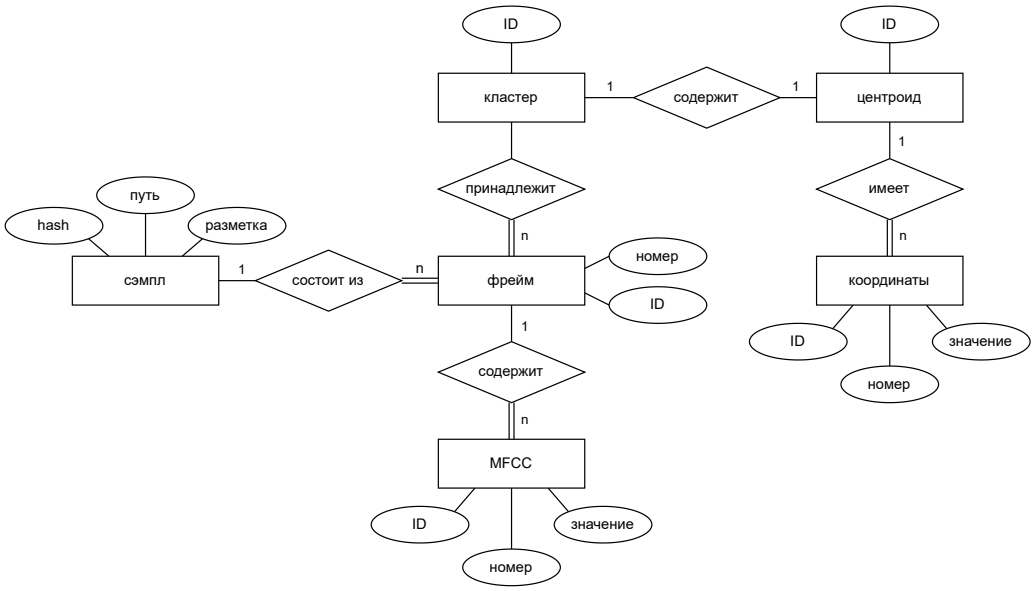

Рис. 2.6 – Диаграмма сущностей базы данных в нотации Чена

Сущность «сэмпл» представляет собой информацию о WAV-файле, который хранится в файловой системе. Сущность содержит поля, необходимые для ее обработки: уникальный хэш для идентификации сущности, абсолютный путь в файловой системе компьютера и разметка, указанная в корпусе. Звуковые дорожки разделены на небольшие фрагменты для дальнейшего анализа. Эти фрагменты представлены сущностью «фрейм». Для работы с фреймами необходимо хранить информацию о его порядковом номере в аудиофайле и идентификатор.

Полученный набор фреймов используется в качестве входных данных для кластеризации. В результате кластеризации каждому фрейму присвоен кластер. Сущность базы данных, хранящая кластер, содержит уникальный идентификатор и информацию о своем центроиде. Центроид, в свою очередь, также содержит идентификатор и набор координат.

3 Технологический раздел

3.1. Выбор средств реализации программного обеспечения

Программное обеспечение состоит из трех модулей: модуль выделения информативных признаков сигнала, модуль кластеризации и модуль создания и обучения скрытых марковских моделей.

Для модулей кластеризации и работы со скрытыми марковскими моделями был выбран язык Golang [35], поскольку процедура кластеризации и обучения скрытых марковских моделей предполагают работу с большими объемами данных. Golang имеет низкие накладные расходы на работу с памятью, что обеспечивает эффективную обработку с большими объемами данных.

Модуль выделения информативных признаков в основном содержит работу с аудиофайлами, поэтому для реализации этого модуля был выбран язык Python [36]. Python имеет широкий спектр библиотек и инструментов для обработки звуковых сигналов, в том числе и для выделения мел-кепстральных коэффициентов.

Для хранения и передачи данных между модулями было решено использовать реляционную базу данных. В качестве СУБД была выбрана PostgreSQL [37], поскольку языки Python и Golang, используемые для написания программного обеспечения, имеют встроенные пакеты работы с данной СУБД.

В качестве средства развертывания программного обеспечения была выбрана утилита Docker [38], поскольку она поддерживает микросервисную архитектуру: Docker обеспечивает возможность разделения приложения на отдельные сервисы, которые могут быть запущены в отдельных контейнерах.

3.2. Компоненты программного обеспечения

3.2.1. Формирование вектора информативных признаков

Компонент формирования вектора информативных признаков выполняет три основные задачи:

– чтение файла разметки, предоставленного разработчиками корпуса DUSHA;

– формирование вектора мел-кепстральных коэффициентов для каждого кадра аудиофайла, прочитанного из файла разметки;

– запись информации об аудиофайлах, их кадрах и векторах мел-кепстральных коэффициентов в базу данных.

Задача чтения файла разметки выполняется классом DatasetProcessor, представленном на листинге 3.1. В качестве аргументов конструктору класса предоставляется абсолютный путь до корпуса DUSHA в файловой системе компьютера и название файла, содержащего разметку.

Листинг 3.1 – Класс для чтения файла разметки

```
class DatasetProcessor:
    def __init__(self, dataset_path, dataset_meta_file):
        self._dataset_path = dataset_path
        self._dataset_meta_file = dataset_meta_file
        self.wavs = []
        self._get_wavs()

    def _get_wavs(self):
        os.chdir(self._dataset_path)
        with open(self._dataset_meta_file, 'r') as mf:
            raw_data = list(mf)
            for i, rec in enumerate(raw_data):
                json_string = json.loads(rec)
                json_string["audio_path"] =
                    _resolve_absolute_path(json_string["audio_path"])

                self.wavs.append(json_string)
```

Метод _get_wavs извлекает из файла разметки пути до аудиофайлов корпуса для дальнейшей работы с ними. В файле разметки присутствует относительный путь от самого файла разметки до аудиофайла корпуса, поэтому для чтения файла необходимо воссановить абсолютный путь аудиофайла корпуса. Это делает функция _resolve_absolute_path.

После того, как пути были извлечены из файла разметки, требуется обработать каждый аудиофайл – открыть его, разделить на кадры и вычислить мел-кепстральные коэффициенты для каждого кадра аудиофайла. Для работы с аудиофайлами реализован модельный класс сущности «Аудиофайл», представленный на листинге 3.2 и модельный класс сущности «Кадр», представленный на листинге 3.3.

Листинг 3.2 – Модельный класс сущности «Аудиофайл»

```
class Sample:
    def __init__(self, uuid, audio_path, emo):
        self.uuid = uuid
        self.audio_path = audio_path
        self.emo = emo

    def __dict__(self):
        return {
            "uuid": self.uuid,
            "audio_path": self.audio_path,
            "emotion": self.emotion,
            "batch": self.batch
        }
```

Листинг 3.3 – Модельный класс сущности «Кадр»

```
class Frame:
    def __init__(self, sample_id, index, mfcc):
        self.id = str(uuid.uuid4())
        self.sample_id = sample_id
        self.sample_num = index
        self.mfcc = mfcc
```

Класс AudioFeaturesExtractor (листинг 3.4) выполняет функции разделения аудиофайла на кадры и вычисления мел-кепстральных коэффициентов для каждого кадра аудиофайла.

Листинг 3.4 – Класс для формирования вектора информативных признаков

```
class AudioFeaturesExtractor:
    def __init__(self, file_path, frame_length=0.2, hop_length=0.1):
        sig, self.sampling_rate =
            librosa.load(file_path, res_type='kaiser_fast')
        self.frame_length = int(self.sampling_rate * frame_length)
        self.hop_length = int(hop_length * self.sampling_rate)
        self.frames =  librosa.util.frame(sig,
            frame_length=self.frame_length, hop_length=self.hop_length)

    def get_mfcc(self, n_mfcc=13):
        mfcc = []
        for frame in self.frames.T:
            mfccs = librosa.feature.mfcc(y=frame,
                                         sr=self.sampling_rate, n_mfcc=n_mfcc)
            mfcc_per_frame = np.mean(mfccs.T, axis=0)
            mfcc.append(mfcc_per_frame)

        return np.array(mfcc)
```

При обработке аудиофайла (чтение, деление на кадры и извлечение мел-кепстральных коэффициентов) была использована библиотека «Librosa» [39], предназначенная для анализа и обработки аудиофайлов.

Аргументами конструктора класса являются путь до аудиофайла в памяти компьютера, длина кадра и размер перекрытия. Метод get_mfcc используется для извлечения вектора мел-кепстральных коэффициентов. Его аргументом является количество извлекаемых коэффициентов.

Класс SampleRepository (листинг 3.5) реализует «репозиторий» сущности «аудиофайл», инкапсулирующий способ хранения данных.

Листинг 3.5 – «Репозиторий» сущности «Аудиофайл»

```
class SampleRepository(Repository):
    def __init__(self, db_client):
        self.db_client = db_client

    def create(self, entity):
        cursor = self.db_client.cnx.cursor()
        query = f"INSERT INTO sample VALUES (%s, %s, %s)"

        cursor.execute(query, (entity.uuid, entity.audio_path, entity.emo))
        self.db_client.cnx.commit()

        cursor.close()

    def get(self):
        cursor = self.db_client.cnx.cursor()
        query = "SELECT uuid, audio_path, emotion FROM sample"
        cursor.execute(query)

        samples = []
        for row in cursor.fetchall():
            uuid, audio_path, emotion = row
            sample = Sample(uuid, audio_path, emotion)
            samples.append(sample)

        cursor.execute(query)
        self.db_client.cnx.commit()

        cursor.close()

        return samples
```

Аналогично для сущности «Кадр» на листинге 3.6 представлен класс FrameRepository, реализующий «репозиторий» сущности.

Листинг 3.6 – «Репозиторий» сущности «Кадр»

```
class FrameRepository(Repository):
    def __init__(self, db_client):
        self.db_client = db_client

    def create(self, entity):
        self._create_frame(entity)
        self._create_mfcc(entity.id, entity.mfcc)

    def get(self):
        pass

    def _create_frame(self, frame):
        cursor = self.db_client.cnx.cursor()
        query = "INSERT INTO frame (uuid, sample_uuid, index) \
                 VALUES (%s, %s, %s)"

        values = (frame.id, frame.sample_id, frame.sample_num)

        cursor.execute(query, values)
        self.db_client.cnx.commit()

        cursor.close()

    def _create_mfcc(self, frame_id, mfcc):
        cursor = self.db_client.cnx.cursor()

        for i, c in enumerate(mfcc):
            query = "INSERT INTO mfcc (uuid, frame_uuid, index, value) \
                     VALUES (%s, %s, %s, %s)"
            values = (str(uuid.uuid4()), frame_id, i + 1, c.item())
            cursor.execute(query, values)
            self.db_client.cnx.commit()

        cursor.close()
```

3.2.2. Кластеризация

Сервис кластеризации выполняет две основные задачи: кластеризация векторов информативных признаков и запись информации о кластерах и их центроидах в базу данных.

Функция KMeans, представленная на листинге 3.7 в качестве входных данных принимает массив векторов информативных признаков каждого кадра каждого аудиофайла и возвращает массив центроидов кластеров, соответствующих этому набору.

Листинг 3.7 – Функция нахождения центроидов кластеров

```
type Node []float64

func KMeans(Nodes []Node, clusterCount int, maxRounds int) ([]Node, error) {
    if len(Nodes) < clusterCount {
        return nil,
            errors.New("amount of nodes is smaller than cluster count")
    }

    stdLen := 0
    for i, Node := range Nodes {
        curLen := len(Node)

        if i > 0 && len(Node) != stdLen {
            return nil, errors.New("data is not consistent dimension-wise")
        }
        stdLen = curLen
    }

    centroids := make([]Node, clusterCount)
    r := rand.New(rand.NewSource(time.Now().UnixNano()))
    for i := 0; i < clusterCount; i++ {
        srcIndex := r.Intn(len(Nodes))
        srcLen := len(Nodes[srcIndex])
        centroids[i] = make(Node, srcLen)
        copy(centroids[i], Nodes[r.Intn(len(Nodes))])
    }

    return initialCentroids(Nodes, maxRounds, centroids), nil
}
```

Для создания общего массива векторов информативных признаков каждого кадра необходимо извлечь их из базы данных.

Для работы с кадрами аудиофайла реализована модельная структура сущности «кадр» (листинг 3.8). Полям структуры присвоены json-метки, указывающие на соответствие полей структуры полям таблицы кадров в базе данных.

Листинг 3.8 – Модельная структура сущности «кадр»

```
type Frame struct {
    ID          string `db:"uuid"`
    SampleUUID  string `db:"sample_uuid"`
    Index       int    `db:"index"`
    ClusterIndex string `db:"cluster_uuid"`
    MFCCs       []float64
}
```

Сущность «кадр» имеет соответствующую сервисную структуру (листинг 3.9), в которой присутствует функция, реализующая извлечение кадров из базы данных и функция, AssignCluster, которая ставит в соответствие кадру аудиофайла ближайший кластер.

Листинг 3.9 – Сервисная структура сущности «кадр»

```
type FrameService struct {
    repo *postgres.FramePostgres
}

func (s *FrameService) GetAll() ([]entity.Frame, error) {
    frames, err := s.repo.GetAll()
    if err != nil {
        return nil, err
    }
    return frames, nil
}

func (s *FrameService) AssignCluster(frame entity.Frame,
                            clusters []entity.Cluster) error {
    centroids := make([]kmeans.Node, len(clusters))
    for i, cluster := range clusters {
        centroids[i] = cluster.Centroid.Value
    }
    nearestIndex := kmeans.Nearest(frame.MFCCs, centroids)
    nearest := clusters[nearestIndex]

    return s.repo.AssignCluster(nearest.ID, frame.ID)
}
```

Структура FramePostgres, экземпляр которой присутсвует в поле сервисной структуры FrameService, реализует «Репозиторий» сущности «кадр», инкапсулирующий работу с базой данных Postgresql. Структура репозитория сущности «Кадр» представлена на листинге 3.10.

Листинг 3.10 – Репозиторий сущности «Кадр»

```
type FramePostgres struct {
    db *sqlx.DB
}
```

Полем структуры является клиентское соединение с базой данных PostgreSQL. Репозиторий содержит метод GetAll, представленный на листинге 3.11. В методе реализовано извлечение всех кадров и их информативных признаков из базы данных и помещает в соответствующий массив.

Листинг 3.11 – Извлечение кадров и их информативных признаков из базы данных

```
func (f *FramePostgres) GetAll() ([]entity.Frame, error) {
    var frames []entity.Frame

    query := `SELECT f.uuid, f.sample_uuid, f.index,
        array_agg(m.value ORDER BY m.index) AS mfccs
    FROM frame f INNER JOIN mfcc m ON f.uuid = m.frame_uuid
    GROUP BY f.uuid, f.sample_uuid, f.index, f.cluster_uuid`

    rows, err := f.db.Query(query)
    if err != nil {
        return nil, err
    }

    for rows.Next() {
        var (
            uuid       string
            sampleUUID string
            index      int
            mfccsStr   string
        )
        err = rows.Scan(&uuid, &sampleUUID, &index, &mfccsStr)
        if err != nil {
            return nil, err
        }

        mfccs, err := parseMFCC(mfccsStr)
        if err != nil {
            return nil, err
        }

        frames = append(frames, entity.Frame{
            ID:         uuid,
            SampleUUID: sampleUUID,
            Index:      index,
            MFCCs:      mfccs,
        })
    }

    return frames, nil
}
```

После получения массива центроидов кластеров необходимо установить соответствие между кадром и кластером, к которому этот кадр принадлежит. Кластер ставится в соответствие кадру в результате выполнения функции Nearest,

представленной на листинге 3.12.

Листинг 3.12 – Функция находления ближайшего кластера

```
func Nearest(in Node, nodes []Node) int {
    count := len(nodes)

    results := make(Node, count)

    cnt := make(chan int)
    for i, node := range nodes {
        go func(i int, node, cl Node) {
            results[i] = distance(in, node)
            cnt <- 1
        }(i, node, in)
    }

    wait(cnt, results)

    minI := 0
    curDist := results[0]

    for i, dist := range results {
        if dist < curDist {
            curDist = dist
            minI = i
        }
    }

    return minI
}
```

Функция Nearest оперирует индексами центроидов. При дальнейшей работе используется не целочисленный индекс кластера, а соответствующие ему модельные структуры сущности «кластер» и сущности «центроид». Модельная структура сущности «кластер» представлена на листинге 3.13.

Листинг 3.13 – Модельные структуры сущностей «кластер» и «центроид»

```
type Cluster struct {
    ID       string    `db:"uuid"`
    Index    int       `db:"index"`
    Centroid Centroid  `db:"centroid_id"`
}

type Centroid struct {
    ID    string    `db:"uuid"`
    Value []float64 `db:"value"`
}
```

Для записи данных о кластере в базу так же имеется соответствующий репозиторий, аналогичный репозиториям остальных сущностей.

Установка соответствия между центроидом кластера и его модельной структурой Cluster реализована в функции constructClusterData сервисного класса ClusterService, представленной на листинге 3.14.

Листинг 3.14 – Установка соответствия между центроидом кластера и его модельной структурой

```
1  func (s *ClusterService) constructClusterData(centroids []entity.Centroid)
2                                                  []entity.Cluster {
3      clusters := make([]entity.Cluster, len(centroids))
4
5      for i, centroid := range centroids {
6          clusters[i] = entity.Cluster{
7              ID:       uuid.New().String(),
8              Index:    i + 1,
9              Centroid: centroid,
10         }
11     }
12     return clusters
13 }
```

Аналогичная функция представлена для установки соответствия между индексом центроида и его модельной структурой Centroid (листинг 3.15).

Листинг 3.15 – Установка соответствия между индексом центроида и его модельной структурой

```
1  func (s *ClusterService) constructCentroidsData(centroidsCoords
2                                                  []kmeans.Node) []entity.Centroid {
3      centroids := make([]entity.Centroid, len(centroidsCoords))
4
5      for i, centroid := range centroidsCoords {
6          centroids[i] = entity.Centroid{
7              ID:    uuid.New().String(),
8              Value: centroid,
9          }
10     }
11
12     return centroids
13 }
```

В методе AssignClusters сервисного класса ClusterService, представленном на листинге 3.16, реализована кластеризация векторов информативных признаков. Метод возвращает массив модельных структур полученных кластеров.

Листинг 3.16 – Кластеризация

```
func (s *ClusterService) AssignClusters(frames []entity.Frame,
                    nclusters, maxRounds int) ([]entity.Cluster, error) {
    nodes := s.collectFramesData(frames)

    centroidsCoords, err := kmeans.KMeans(nodes, nclusters, maxRounds)
    if err != nil {
        return nil, err
    }

    centroids := s.constructCentroidsData(centroidsCoords)
    return s.constructClusterData(centroids), nil
}
```

3.2.3. Создание и обучение скрытых марковских моделей

Компонент создания и обучения скрытых марковских моделей выполняет три функции:

– создание марковской модели для каждой эмоции;

– обучение каждой скрытой марковской модели;

– распознавание эмоций в аудиофайлах.

На листинге 3.17 представлена структура скрытой марковской модели, где «Transitions» – матрица переходов, «Emissions» – матрица эмиссий и «StationaryProbabilities» – вектор начальных состояний.

Листинг 3.17 – Модельная структура скрытой марковской модели

```
type HiddenMarkovModel struct {
    Transitions             [][]float64 `json:"transitions"`
    Emissions               [][]float64 `json:"emissions"`
    StationaryProbabilities []float64   `json:"stationary_probabilities"`
}
```

Обучение скрытой марковской модели с использованием алгоритма Баума — Велша основано на принципе EM (*англ. expectation-maximization*), поэтому изначально матрицы скрытой марковской модели инициализируются случайным образом (E-этап).

На листинге 3.18 представлена случайная инициализация матриц скрытой марковской модели.

Листинг 3.18 – Инициализация скрытой марковской модели

```
func New(nStates, nObs int) *HiddenMarkovModel {
    seed := rand.New(rand.NewSource(0))
    emitProb := allocateRandomMatrix(nStates, nObs, seed)
    transProb := allocateRandomMatrix(nStates, nStates, seed)
    initProb := make([]float64, nStates)
    sum := 0.0
    for i := range initProb {
        initProb[i] = seed.Float64()
        sum += initProb[i]
    }
    for i := range initProb {
        initProb[i] /= sum
    }

    return &HiddenMarkovModel{
        Transitions:            transProb,
        Emissions:              emitProb,
        StationaryProbabilities: initProb,
    }
}
```

Для обучения скрытой марковской модели требуется прочитать данные об аудиофайлах и их кадрах из базы данных. В модельной структуре сущности «кадр» (листинг 3.19) присутствуют не значения мел-кепстральных коэффициентов, а индексы кластеров, к которым принадлежит соответствующий набор значений мел-кепстральных коэффициентов.

Листинг 3.19 – Модельная структура сущности «кадр»

```
type Frame struct {
    ID           string `db:"uuid"`
    SampleID     string `db:"sample_uuid"`
    Index        int    `db:"frame_index"`
    ClusterIndex int    `db:"cluster_index"`
}
```

Модельная структура сущности «аудиофайл» представлена на листинге 3.20.

Листинг 3.20 – Модельная структура сущности «аудиофайл»

```
type Sample struct {
    ID        string `db:"uuid"`
    AudioPath string `db:"audio_path"`
    Emotion   Label  `db:"emotion"`
    Frames    []Frame
}
```

Создание последовательности наблюдаемых в аудиофайле кластеров осуществляется в функции ConstructObservationSequence сервисного класса SampleService (листинг 3.21). В дальнейшем эта последовательность будет использована как входные данные для алгоритмов обучения и распознавания.

Листинг 3.21 – Создание последовательности наблюдений

```
func (s *SampleService) ConstructObservationSequence(sm entity.Sample) []int {
    observations := make([]int, len(sm.Frames))
    for i := 0; i < len(observations); i++ {
        observations[i] = sm.Frames[i].ClusterIndex
    }
    return observations
}
```

Обучение скрытой марковской модели с использованием алгоритма Баума — Велша представлено на листинге 3.22.

Листинг 3.22 – Алгоритм Баума — Велша

```
func (hmm *HiddenMarkovModel) BaumWelch(obs []int, iterations int) {
    var (
        alpha = make([][]float64, len(obs))
        beta  = make([][]float64, len(obs))
        gamma = make([][]float64, len(obs))
        xi    = make([][][]float64, len(obs)-1)
    )
    for i := 0; i < len(obs); i++ {
        alpha[i] = make([]float64, len(hmm.Transitions))
        beta[i] = make([]float64, len(hmm.Transitions))
        gamma[i] = make([]float64, len(hmm.Transitions))
        if i < len(obs)-1 {
            xi[i] = make([][]float64, len(hmm.Transitions))
            for j := 0; j < len(hmm.Transitions); j++ {
                xi[i][j] = make([]float64, len(hmm.Transitions))
            }
        }
    }
    for it := 0; it < iterations; it++ {
        hmm.Forward(obs, alpha)
        hmm.Backward(obs, beta)
        hmm.computeGammaProbs(obs, alpha, beta, gamma)
        hmm.computeXiProbs(obs, alpha, beta, xi)

        hmm.update(obs, gamma, xi)
    }
    hmm.laplaceSmoothing(obs)
}
```

Пересчет матриц скрытой марковской модели в алгоритме Баума — Велша включает три шага: шаг прямого прохода, шаг обратного прохода и шаг обновления.

Шаг прямого прохода, в котором вычисляется вероятность наблюдать текущую последовательность состояний и наблюдений, используя текущие параметры модели (начальные вероятности, матрица переходов и матрица наблюдений), представлен на листинге 3.23.

Листинг 3.23 – Алгоритм прямого прохода

```
1  func (hmm *HiddenMarkovModel) Forward(obs []int, alpha [][]float64) {
2      for i := 0; i < len(hmm.Transitions); i++ {
3       alpha[0][i] = hmm.StationaryProbabilities[i] * hmm.Emissions[i][obs[0]]
4      }
5      for t := 1; t < len(obs); t++ {
6          for j := 0; j < len(hmm.Transitions); j++ {
7              sum := 0.0
8              for i := 0; i < len(hmm.Transitions); i++ {
9                  sum += alpha[t-1][i] * hmm.Transitions[i][j]
10             }
11             alpha[t][j] = sum * hmm.Emissions[j][obs[t]]
12         }
13     }
14 }
```

Шаг обратного прохода, в котором вычисляется вероятность наблюдать оставшуюся часть последовательности состояний и наблюдений, начиная с текущего состояния представлен на листинге 3.24.

Листинг 3.24 – Алгоритм обратного прохода

```
1  func (hmm *HiddenMarkovModel) Backward(obs []int, beta [][]float64) {
2      for i := 0; i < len(hmm.Transitions); i++ {
3          beta[len(obs)-1][i] = 1.0
4      }
5      for t := len(obs) - 2; t >= 0; t-- {
6          for i := 0; i < len(hmm.Transitions); i++ {
7              sum := 0.0
8              for j := 0; j < len(hmm.Transitions); j++ {
9                  sum += hmm.Transitions[i][j] * hmm.Emissions[j][obs[t+1]] *
10                     beta[t+1][j]
11             }
12             beta[t][i] = sum
13         }
14     }
15 }
```

Вычисление γ-вероятностей представлен на листинге 3.25.

Листинг 3.25 – Вычисление γ-вероятностей

```
func (hmm *HiddenMarkovModel) computeGammaProbs(obs []int, alpha [][]float64,
                          beta [][]float64, gamma [][]float64) {
    for t := 0; t < len(obs); t++ {
        sum := 0.0
        for i := 0; i < len(hmm.Transitions); i++ {
            gamma[t][i] = alpha[t][i] * beta[t][i]
            sum += gamma[t][i]
        }

        for i := 0; i < len(hmm.Transitions); i++ {
            gamma[t][i] /= sum
        }
    }
}
```

Вероятности α, β и γ используются при вычислении ξ-вероятностей. Вычисление ξ-вероятностей представлено на листинге 3.26.

Листинг 3.26 – Вычисление ξ-вероятностей

```
func (hmm *HiddenMarkovModel) computeXiProbs(obs []int, alpha [][]float64,
                    beta [][]float64, xi [][][]float64) {
    for t := 0; t < len(obs)-1; t++ {
        sum := 0.0
        for i := 0; i < len(hmm.Transitions); i++ {
            for j := 0; j < len(hmm.Transitions); j++ {
                xi[t][i][j] = alpha[t][i] * hmm.Transitions[i][j] *
                     hmm.Emissions[j][obs[t+1]] * beta[t+1][j]
                sum += xi[t][i][j]
            }
        }

        for i := 0; i < len(hmm.Transitions); i++ {
            for j := 0; j < len(hmm.Transitions); j++ {
                xi[t][i][j] /= sum
            }
        }
    }
}
```

Пересчет матриц скрытой марковской модели в алгоритме Баума — Велша основывается на оценке вероятностей переходов и вероятностей наблюдений, основанных на прямом и обратном проходах через модель, и последующем обновлении этих матриц на основе ожидаемых значений. На листинге 3.27

представлена функция пересчета значений матриц скрытой марковской модели, использующая α, β, γ и ξ вероятности.

Листинг 3.27 – Пересчет значений матриц скрытой марковской модели

```
func (hmm *HiddenMarkovModel) update(obs []int, gamma [][]float64,
                                     xi [][][]float64) {
    for i := 0; i < len(hmm.StationaryProbabilities); i++ {
        hmm.StationaryProbabilities[i] = gamma[0][i]
    }

    for i := 0; i < len(hmm.Transitions); i++ {
        for j := 0; j < len(hmm.Transitions); j++ {
            sumXi := 0.0
            sumGamma := 0.0
            for t := 0; t < len(obs)-1; t++ {
                sumXi += xi[t][i][j]
                sumGamma += gamma[t][i]
            }
            hmm.Transitions[i][j] = sumXi / sumGamma
        }
    }

    for i := 0; i < len(hmm.Emissions); i++ {
        for j := 0; j < len(hmm.Emissions[0]); j++ {
            sumGamma := 0.0
            sumGammaObs := 0.0
            for t := 0; t < len(obs); t++ {
                if obs[t] == j {
                    sumGammaObs += gamma[t][i]
                }
                sumGamma += gamma[t][i]
            }
            hmm.Emissions[i][j] = sumGammaObs / sumGamma
        }
    }
}
```

Для избежания появления нулей в матрице эмиссий скрытой марковской модели перед первой итерацией алгоритма Баума — Велша следует использовать сглаживание. В качестве метода сглаживания был выбран метод сглаживания Лапласа [40].

Сглаживание Лапласа, также известное как аддитивное сглаживание, основано на добавлении небольших положительных значений ко всем элементам матрицы скрытой марковской модели.

Сглаживание Лапласа представлено на листинге 3.28.

Листинг 3.28 – Сглаживание Лапласа

```
const (
    SMOOTHING_FACTOR float64 = 0.5
)

func (hmm *HiddenMarkovModel) laplaceSmoothing(observations []int) {
    nStates := len(hmm.Emissions)
    nObs := len(hmm.Emissions[0])

    emissions := make([][]float64, nStates)
    for i := range emissions {
        emissions[i] = make([]float64, nObs)
    }

    counts := make([]int, nObs)
    for i := 0; i < len(observations); i++ {
        state := observations[i]
        counts[state]++
    }

    total := 0
    for _, count := range counts {
        total += count
    }

    for i := 0; i < nObs; i++ {
        smoothedProbability := (float64(counts[i]) + SMOOTHING_FACTOR)
            / (float64(total) + SMOOTHING_FACTOR*float64(nObs))
        emissions[0][i] = smoothedProbability
    }

    hmm.Emissions = emissions
}
```

Наиболее вероятную эмоцию можно получить после запуска алгоритма прямого хода. В нем вычисляется вероятность наблюдать представленную на вход алгоритма последовательность, при условии заданной модели. Эта вероятность называется вероятностью наблюдения.

Из вычисленных вероятностей наблюдения выбирается максимальная. Скрытая марковская модель, которой принадлежит максимальная вероятность наблюдения считается подходящей для входной последовательности и эмоция, на которой была обучена эта модель, считается наиболее вероятной.

На листинге 3.29 представлен алгоритм получения наиболее вероятной эмоции.

Листинг 3.29 – Получение наиболее вероятной эмоции

```
func FindBestFittedModel(obs []int, models []HiddenMarkovModel) int {
    probabilities := make([]float64, len(models))

    for i, model := range models {
        alpha := make([][]float64, len(obs))
        for i := 0; i < len(obs); i++ {
            alpha[i] = make([]float64, len(model.Transitions))
        }

        model.Forward(obs, alpha)
        probability := 0.0
        for i := 0; i < len(models[0].Transitions); i++ {
            probability += alpha[len(obs)-1][i]
        }
        probabilities[i] = probability
    }

    max := floats.MaxIdx(probabilities)

    return max
}
```

Обученные марковские модели решено хранить в JSON-файле. Функция записи матриц скрытой марковской модели в файл представлена на листинге 3.30.

Листинг 3.30 – Запись матриц скрытой марковской модели в файл

```
func (hmm *HiddenMarkovModel) SaveJSON(filename string) error {
    file, err := os.Create(filename)
    if err != nil {
        return err
    }
    defer file.Close()

    encoder := json.NewEncoder(file)
    if err := encoder.Encode(hmm); err != nil {
        return err
    }

    return nil
}
```

Результат распознавания эмоций представлен в качестве матрицы несоответствий. На листинге 3.31 представлена инициализация матрицы несоответствий с использованием истинных и спрогнозированных значений разметки.

Листинг 3.31 – Инициализация матрицы несоответствий

```
func NewConfusionMatrix(labels []entity.Label, actual,
                        predicted []entity.Label) *ConfusionMatrix {
    confusionMatrix := make(map[entity.Label]map[entity.Label]float64)

    for _, ac := range labels {
        confusionMatrix[ac] = make(map[entity.Label]float64)
        for _, pred := range labels {
            confusionMatrix[ac][pred] = 0.
        }
    }

    for i := 0; i < len(actual); i++ {
        confusionMatrix[actual[i]][predicted[i]]++
    }

    return &ConfusionMatrix{labels: labels, Values: confusionMatrix}
}
```

Матрицу несоответствий перед отображением следует нормализовать. Нормализованная матрица несоответствий обычно представляется в виде долей.

3.3. Тестирование компонент программного обеспечения

Для оценки правильности работы программного обеспечения было проведено тестирование алгоритма обучения скрытой марковской модели. Было выделено два класса эквивалентности, представленных в таблице 3.1.

Таблица 3.1 – Классы эквивалентности

№	*Описание*	*Входные данные*	*Ожидаемый результат*
1	последовательность наблюдаемых состояний содержит различные состояния	эмиссии, наблюдаемые состояния	алгоритм правильно откорректирует матрицу эмиссий
2	последовательность наблюдаемых состояний содержит одно и то же состояние	эмиссии, наблюдаемые состояния	алгоритм правильно откорректирует матрицу эмиссий

Для тестов подготовлены файлы, содержащие значения матрицы эмисси-

ий и последовательности состояний. Выполнение тестов 1-2 представлена на листинге 3.32.

Листинг 3.32 – Выполнение тестов

```
func TestHiddenMarkovModel_BaumWelch(t *testing.T) {
    testCases := []struct {
        name  string
        fEmit string
        fObs  string
        fExp  string
        dim   int
    }{
        {
            name:  "STATES DIFFER",
            fEmit: "etc/test-cases/hmm-diff-emit",
            fExp:  "etc/test-cases/hmm-diff-expect",
            fObs:  "etc/test-cases/hmm-diff-obs",
            dim:   50,
        },
        {
            name:  "SAME STATE",
            fEmit: "etc/test-cases/hmm-same-emit",
            fExp:  "etc/test-cases/hmm-same-expect",
            fObs:  "etc/test-cases/hmm-same-obs",
            dim:   50,
        },
    }

    for _, testCase := range testCases {
        t.Run(testCase.name, func(t *testing.T) {

            err := runBaumWelchTest(testCase.fEmit, testCase.fObs,
                                    testCase.fExp, testCase.dim)

            if err != nil {
                t.Fatal(err)
            }
        })
    }
}
```

Каждый тестовый случай представлен структурой в массиве testCases, в которой присутствует название теста (поле name), файл с матрицей эмиссий (поле fEmit), файл с массивом наблюдаемых состояний (поле fObs), файл с ожидаемым значением матрицы эмиссий после прогона алгоритма (поле fExp) и размерность матрицы эмиссий (поле dim).

Вспомогательная функция, запускающая алгоритм алгоритм Баума — Велша для тестов представлена на листинге 3.33.

Листинг 3.33 – Тестовый прогон алгоритма Баума — Велша

```
func runBaumWelchTest(fEmit, fObs, fExp string, dim int) error {
    emit, err := readMatrixFromFile(fEmit, 1, dim)
    if err != nil {
        return err
    }
    hmm := HiddenMarkovModel{
        Transitions: [][]float64{
            {1},
        },
        Emissions:                emit,
        StationaryProbabilities: []float64{1},
    }
    obs, err := readArrayFromFile(fObs)
    if err != nil {
        return err
    }
    hmm.BaumWelch(obs, 1)
    expectedEmissions, err := readMatrixFromFile(fExp, 1, dim)
    if err != nil {
        return err
    }
    equal := matricesEqual(hmm.Emissions, expectedEmissions, 0.005)
    if !equal {
        return errors.New("emission matrix differs from expected")
    }
    return nil
}
```

3.4. Физические компоненты системы и их размещение на устройствах

Диаграмма развертывания, отражающая особенности физической архитектуры системы, представлена на рисунке 3.1.

Система распознавания размещена на одном устройстве, однако для каждого компонента присутствует собственная среда исполнения: компоненты создания и обучения СММ и кластеризации имеют среду исполнения Golang, в то время как компонент формирования информативных признаков имеет среду исполнения Python3. Точка входа, агрегирующая все функции системы, так же имеет свою среду исполнения.

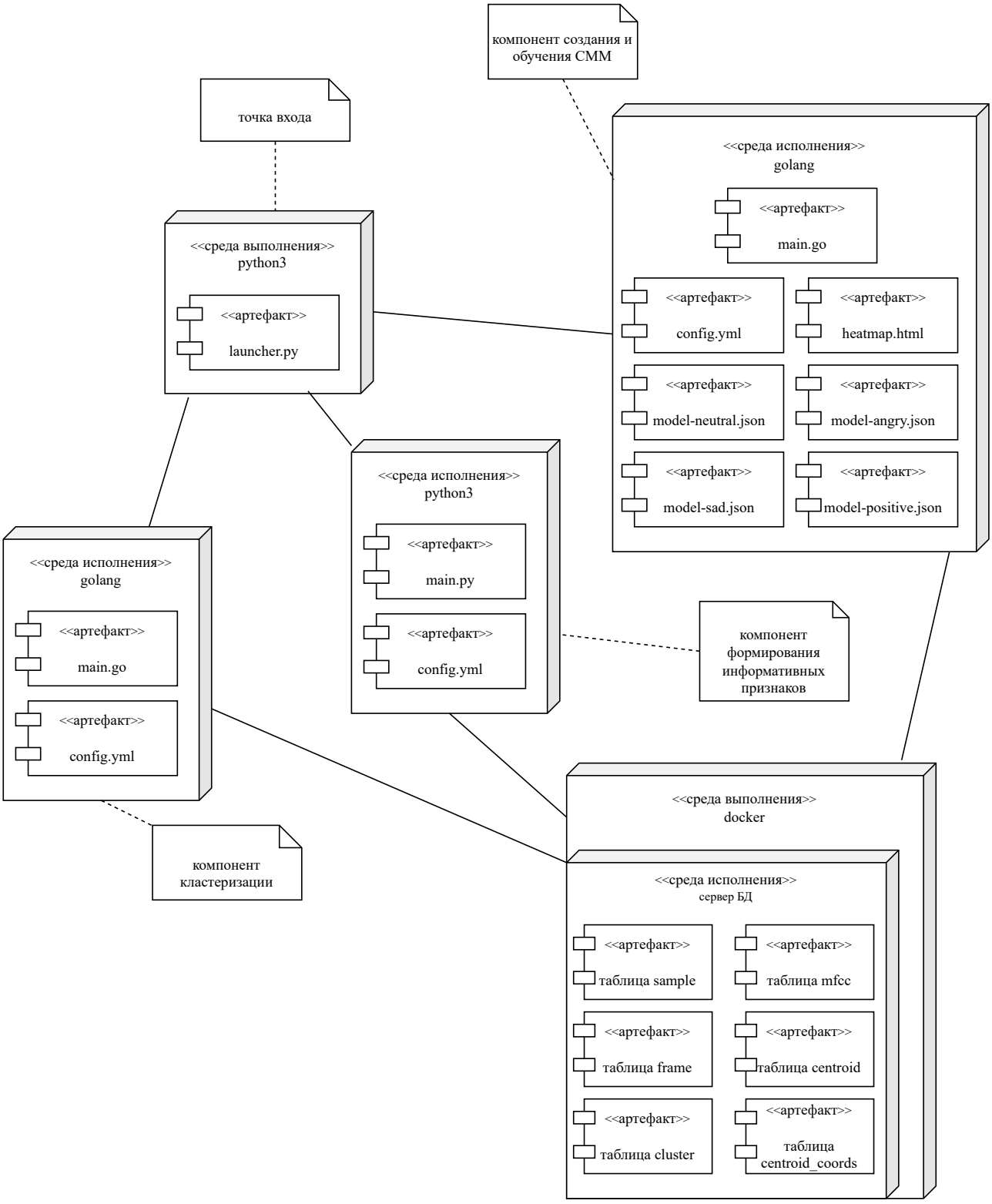

Рис. 3.1 – Диаграмма развертывания

3.5. Формат входных и выходных данных

Компонент формирования вектора информативных признаков принимает на вход файл формата YAML [41], содержащий следующие поля:

– «dataset_path»: абсолютный путь к корневой директории корпуса речи;

– следующие поля для DSN (*англ. Data Source Name*) строки соединения с базой данных:

  – «psql_bind_ip»: сервер, на котором работает база данных;
  – «psql_port»: порт, на котором работает база данных;
  – «psql_user»: имя пользователя базы данных;
  – «psql_password»: пароль для пользователя базы данных;
  – «psql_database»: имя базы данных.

Так же компонент принимает на вход параметр командной строки «dataset_metafile», представляющий относительный путь файла разметки от корневой директории корпуса речи «dataset_path» и параметр «mode», представляющий ключ для выполняемого действия: «from-file», если требуется заполнить базу данных из файла разметки и «add», если требуется добавить файл по его абсолютному пути. Во втором случае принимается на вход также аргумент «audio_path», представляющий собой абсолютный путь до файла, подвергаемого обработке.

Компонент кластеризации так же принимает на вход файл формата YAML, содержащий следующие поля:

– Аналогичные конфигурационному файлу для компонента формирования вектора информативных признаков поля для DSN строки соединения с базой данных («psql_bind_ip», «psql_port», «psql_user», «psql_password», «psql_database») и параметр «ssl_mode», определяющий уровень требуемой безопасности для соединения;

– «cluster_amount»: число кластеров для алгоритма k-means;

– «kmeans_max_rounds»: число итераций для алгоритма k-means.

Так же компонент принимает на вход параметр командной строки «mode», представляющий ключ для выполняемого действия: «cluster-all», если требуется произвести кластеризацию всех кадров в базе данных, и «assign» в случае если требуется соотнести один набор кадров с соответствующими кластерами. Во втором случае принимается на вход также аргумент «audio_path», представляющий собой абсолютный путь до файла, подвергаемого обработке.

Компонент создания и обучения скрытых марковских моделей принимает на вход параметр командной строки «mode», представляющий ключ для выполняемого действия (обучение, проверка или распознавание), принимающий значения «test», «train» и «recognize». Также компонент создания и обучения скрытых марковских моделей принимает на вход файл формата YAML, содержащий следующие поля:

– Аналогичные конфигурационному файлу для компонента кластеризации поля для DSN строки соединения с базой данных;

– «cluster_amount»: число кластеров алгоритма k-means (для задания количества состояний моделей);

– «model_file_path»: шаблон для имени файла скрытой марковской модели, представляющий строку со строковым спецификатором формата, в который во время выполнения алгоритма будет подставлено название эмоции.

Компонент создания и обучения скрытых марковских моделей имеет также выходной параметр – html-файл, содержащий инструкции отображения матрицы ошибок в веб-браузере.

4 Исследовательский раздел

4.1. Предварительная обработка обучающего набора данных

Перед составлением тренировочной и обучающих выборок корпус DUSHA был проанализирован на предмет процентного соотношения каждой классов в разметке. Распределение классов разметки по доменам представлено в таблице 4.1.

Таблица 4.1 – Распределение классов разметки в корпусе DUSHA

Домен	*Crowd*		*Podcast*	
Эмоция	*Количество, шт.*	*Доля, %*	*Количество, шт.*	*Доля, %*
positive	15446	9.40	5909	6.53
sad	23316	14.18	1170	1.29
angry	17120	10.41	2057	2.27
neutral	106850	65.00	81104	89.66
other	1655	1.01%	222	0.25

Из таблицы 4.1 видно, что классы в разметке неравномерно распределены. Поэтому для обучения и проверки скрытой марковской модели был использован не весь набор данных DUSHA, а его часть. Из файлов разметки были извлечены все аудиофайлы, длина которых не превышала 3 секунды и разметка которых не содержала значения «other». В обучающую выборку было включено 1500 аудиофайлов каждого класса разметки. Объем данных обучающей выборки, которая подается на вход классификатору, представлен в таблице 4.2.

Таблица 4.2 – Объем данных обучающей выборки

Подгруппа	*Всего*	*Тренировочная выборка*	*Тестовая выборка*
angry	1 ч. 02 мин. 47 сек.	50 мин. 05 сек.	12 мин. 41 сек.
neutral	1 ч. 02 мин. 23 сек.	50 мин. 02 сек.	12 мин. 20 сек.
positive	1 ч. 02 мин. 01 сек.	50 мин. 32 сек.	12 мин. 29 сек.
sad	1 ч. 02 мин. 53 сек.	51 мин. 57 сек.	12 мин. 55 сек.

4.2. Результат классификации и его оценка

Результат распознавания на тренировочной выборке представлен в виде таблицы 4.3.

Таблица 4.3 – Матрица несоответствий

Экспертная оценка	*Оценка классификатора*			
	angry	*neutral*	*positive*	*sad*
angry	0.443	0.062	0.359	0.136
neutral	0.177	0.118	0.448	0.258
positive	0.308	0.058	0.503	0.131
sad	0.28	0.073	0.346	0.301

По матрице несоответствий можно рассчитать следующие важные для оценки параметры: TP, FP, TN, FN. Значения и расшифровка этих параметров представлены в таблице 4.4.

Таблица 4.4 – Значения TP, TN, FP, FN каждого класса разметки

Класс	*TP*	*TN*	*FP*	*FN*
	Истинно положительные	*Истинно отрицательные*	*Ложно положительные*	*Ложно отрицательные*
angry	533	4095	920	467
neutral	141	4373	780	1059
positive	603	4690	1206	1007
sad	361	4491	462	799

Для оценки производительности классификатора на основе матрицы несоответствий также можно учесть несколько показателей оценки, таких как точность (*англ. precision*), полнота (*англ. recall*) и F1-мера.

Точность системы в пределах класса – это доля элементов действительно принадлежащих данному классу относительно всех элементов которые система

отнесла к этому классу. Рассчитывается согласно 4.1:

$$\mathrm{Precision_c} = \frac{A_{c,\,c}}{\sum\limits_{i=1}^{n} A_{c,\,i}}, \tag{4.1}$$

где A – матрица несоответствий, c – индекс класса для которого вычисляется точность в матрице несоответствий. Значение точности для каждого класса разметки представлено в таблице 4.5.

Таблица 4.5 – Значение точности для каждого класса разметки

Эмоция	angry	neutral	positive	sad	$\sum$
$\mathrm{Precision_c}$	0.367	0.378	0.303	0.365	0.353

Полнота системы – это доля найденных классфикатором элементов принадлежащих классу относительно всех элементов этого класса в тестовой выборке. Рассчитывается согласно 4.2:

$$\mathrm{Recall_c} = \frac{A_{c,\,c}}{\sum\limits_{i=1}^{n} A_{i,\,c}}, \tag{4.2}$$

де A – матрица несоответствий, c – индекс класса для которого вычисляется полнота в матрице несоответствий. Значение полноты для каждого класса разметки представлено в таблице 4.6.

Таблица 4.6 – Значение точности для каждого класса разметки

Эмоция	angry	neutral	positive	sad	$\sum$
$\mathrm{Recall_c}$	0.443	0.117	0.503	0.301	0.341

F-мера представляет собой гармоническое среднее между точностью и полнотой и вычисляется согласно 4.3:

$$\mathrm{F} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \tag{4.3}$$

Значение F-меры для каждого класса представлено в таблице 4.7.

Таблица 4.7 – Значение F-меры для каждого класса разметки

Эмоция	angry	neutral	positive	sad	$\sum$
F_c	0.366	0.281	0.432	0.321	0.347

Наиболее высокое значение F-меры наблюдается у эмоций «ярость/раздражение» и «позитив», наиболее низкое – у классов «нейтраль» и «грусть», причем разница с классом «нейтраль» наблюдается почти в 1.5 раза, с классом «грусть» – в 1.4 раза. Такое отклонение может быть аргументировано следующим образом:

– в обучающей выборке классов с низкой точностью распознавания используется слишком много коротких аудиозаписей (меньше секунды), и значимая часть интонационного рисунка (модуляция голоса и акцентуация при произнесении) фразы становится маловероятной;

– в обучающей выборке классов с низкой точностью распознавания используется слишком много длинных аудиозаписей, вмещающих несколько фраз, обладающих собственным интонационным контуром;

– неточная или неоднозначная разметка классов;

– эмоции «ярость/раздражение» и «позитив» имеют более разнообразные или выраженные характеристики, что способствует лучшей их идентификации.

В таблице 4.8 представлено количество аудиозаписей в трех диапазонах длительности: Менее 1.5 секунд, 1.5-2.5 секунд и более 2.5 секунд.

Таблица 4.8 – Длительности аудиозаписей по классам разметки

Эмоция	*Менее 1.5 секунд, шт.*	*1.5-2.5 секунд, шт.*	*Более 2.5 секунд, , шт.*
angry	20	603	877
neutral	20	622	858
positive	14	597	889
sad	7	479	1014

Можно заметить, что в классе «грусть» преобладают длинные аудиозаписи, содержащие, скорее всего, несколько фраз. Однако, длины в классе «нейтраль» распределены примерно так же как и в классах с высокими показателями распознавания.

Можно также предположить, что для качественного распознавания безэмоциональной речи требуется больший объем данных. В таблице 4.9 представлены замеры качества обучения классификатора с различными размерами обучающей выборки.

Таблица 4.9 – Замеры качества обучения классификатора

Количество элементов, шт.	*F-мера*			
	angry	neutral	positive	sad
80	0	0.196	0.34	0.254
240	0.379	0.179	0.260	0.215
400	0.313	0.270	0.334	0.218
560	0.321	0.140	0.387	0.265
800	0.340	0.145	0.344	0.308
1200	0.366	0.281	0.432	0.321

Можно заметить, что F-мера каждого класса растет с увеличением количества элементов обучающей выборки.

4.3. Зависимость времени классификации от объема обучающей выборки

4.3.1. Замеры времени обучения классификатора

ЭВМ, на которой проводились исследования производительности, обладает следующими техническими характеристиками:

– операционная система EndeavourOS Linux x86_64;

– оперативная память 8 Гб;

– процессор 11th Gen Intel i5-1135G7 (8) @ 4.200GHz.

В таблице 4.10 представлены результаты выполнения замеров скорости обучения классификатора на выборках различной длины.

Таблица 4.10 – Замеры времени обучения классификатора

Количество элементов обучающей выборки, шт	*Время обучения, мс*
80	32333
240	99034
400	121426
560	121696
800	121728
1200	122216

На рисунке 4.1 представлены данные из таблицы 4.10 в виде графика.

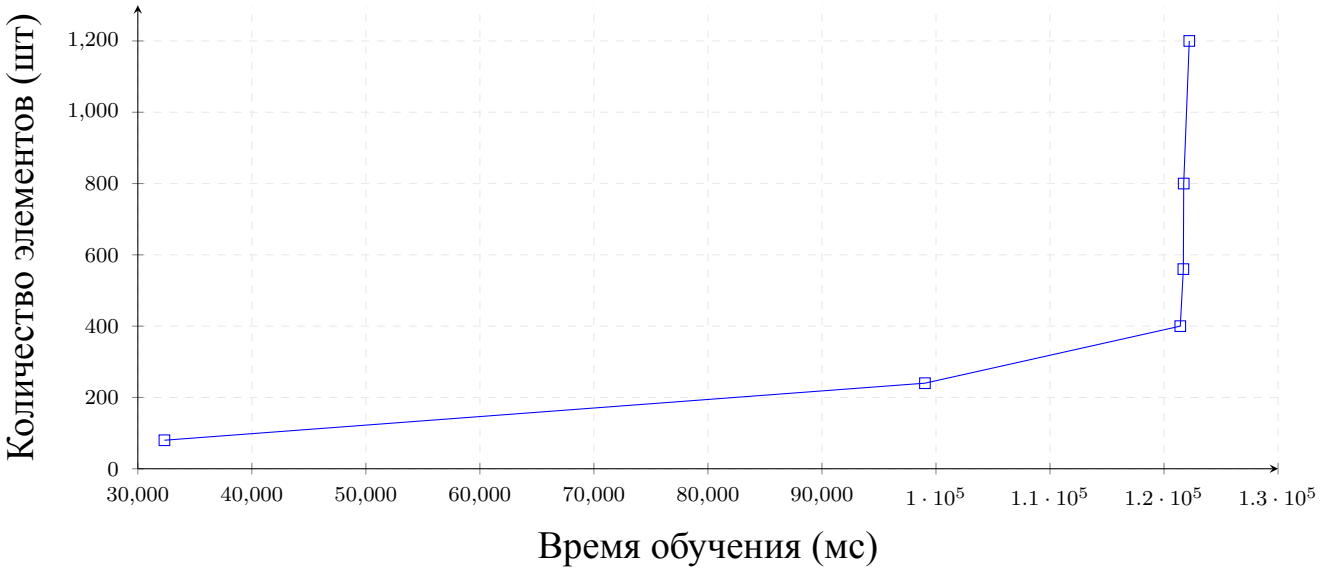

Рис. 4.1 – График зависимости времени обучения от размера тренировочной выборки

Вывод

Классификатор на выборке из 6000 элементов (1200 элементов в тестовой выборке и 300 в тренировочной для каждой эмоции) показал следующие результаты: точность – 0.353, полнота – 0.341 и F-мера – 0.347. При расчете этих параметров для каждого класса было выявлено, что наиболее высокое значение F-меры наблюдается у эмоций «ярость/раздражение» и «позитив», наиболее низкое – у классов «нейтраль» и «грусть», причем разница с классом «нейтраль»

наблюдается почти в 1.5 раза, с классом «грусть» – в 1.4 раза. Такое отклонение, скорее всего, вызвано тем, что для качественной характеристики наиболее «ярких» эмоций, то есть тех, которые имеют наиболее выраженные характеристики, требуется меньший объем обучающих данных. Также было выявлено, что на качество распознавания может влиять количество самостоятельных фраз в аудиозаписи, поскольку каждая фраза имеет свой интонационный контур, и повторение интонационных контуров в аудиозаписи может привести к ухудшению результатов.

ЗАКЛЮЧЕНИЕ

В рамках настоящей работы был разработан и реализован метод распознавания эмоций по звучащей речи. Все поставленные цели были выполнены.

Были проанализированы русскоязычные и иностранные корпуса эмоциональной речи. Русскоязычных корпусов с подходящей эмоциональной разметкой всего 2: RUSLANA и DUSHA. Корпуса RUSLANA нет в открытом доступе, поэтому для обучения классификатора был выбран корпус DUSHA.

Были проанализированы как просодические, так и спектральные признаки, характеризующие эмоцию в речи. Для обучения классификатора были использованы мел-кепстральные коэффициенты, поскольку они более устойчивы к шуму и содержат достаточно широкий набор информации о речи.

Был проведен обзор классификаторов, чаще всего применяющихся в системах распознавания эмоций в речи: скрытой марковской модели и искусственной нейронной сети.

При реализации метода было использовано дискретное пространство эмоций, включающее в себя 4 эмоции: «злость», «радость», «грусть» и «нейтраль».

Классификатор, реализованный в рамках метода, на наборе данных, объем которых составляет 6000 аудиозаписей, показал наиболее высокое качество классификации у эмоций «ярость/раздражение» и «позитив», наиболее низкое – у классов «нейтраль» и «грусть», причем наблюдаемая разница составила почти 1.5 раза. При изменении размера обучающей выборки было выяснено, что положительная тенденция наблюдается у каждого класса разметки. Можно предположить, что для качественной классификации эмоций, имеющих более выраженные признаки, требуется меньший размер обучающей выборки.

В дальнейших исследованиях планируется обучение классификатора на собственном корпусе данных, содержащем информацию не только об эмоции, которую выражал говорящий, но и об интонационном контуре, которым обладает высказывание. За счет связи интонационного контура и некоторых эмоций (например, ИК-6 и эмоции удивления) можно повысить качество классификации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Zhang J.*, *Lu Z.*, *Xu F.* Feeling and caring: When mood intervenes in charitable giving // Journal of Marketing Research. — 2008. — Т. 45, № 2. — С. 213—224.
2. *Ekman P.* Universals and Cultural Differences in Facial Expression of Emotion // Nebraska Symposium on Motivation. Vol. 19. — 1972.
3. *Ekman P.* An Argument for Basic Emotions // Cognition and Emotion. — 1992.
4. *Plutchik R.*, *Kellerman H.* Emotion: Theory, Research, and Experience: Vol. 1. Theories of Emotion. // London: Academic Press. — 1980.
5. Affectica [Электронный ресурс]. — Режим доступа, URL: https://www.affectiva.com/ (дата обращения: 30.9.2022).
6. *Вундт В.* Психология душевных волнений // Психология эмоций. — 1984.
7. RECOLA [Электронный ресурс]. — Режим доступа, URL: https://diuf.unifr.ch/main/diva/recola/ (дата обращения: 1.10.2022).
8. *E. Cambria.* Sentic Computing for social media marketing / E. Cambria [и др.] // Multimedia Tools and Applications - MTA. — 2012. — DOI: 10.1007/s11042-011-0815-0.
9. *Russell J.* Core Affect and the Psychological Construction of Emotion // Psychological Review. — 2003.
10. RAVDESS Emotional speech audio [Электронный ресурс]. — Режим доступа, URL: https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio (дата обращения: 15.3.2023).
11. *Vlasenko B.* Combining frame and turn-level information for robust recognition of emotions within speech / B. Vlasenko [и др.] //. — 01.2007. — С. 2249—2252.
12. EmoDB Dataset [Электронный ресурс]. — Режим доступа, URL: https://www.kaggle.com/datasets/piyushagni5/berlin-database-of-emotional-speech-emodb (дата обращения: 15.3.2023).
13. Toronto emotional speech set (TESS) [Электронный ресурс]. — Режим доступа, URL: https://www.kaggle.com/datasets/ejlok1/toronto-emotional-speech-set-tess (дата обращения: 15.3.2023).

14. *Makarova V.*, *Petrushin V. A.* RUSLANA: A database of Russian emotional utterances // Seventh international conference on spoken language processing. — 2002.

15. *Kondratenko V.* Large Raw Emotional Dataset with Aggregation Mechanism / V. Kondratenko [и др.] // arXiv preprint arXiv:2212.12266. — 2022.

16. Русскоязычный эмоциональный корпус: коммуникативное взаимодействие в реальных эмоциональных ситуациях. [Электронный ресурс]. — Режим доступа, URL: https://events.spbu.ru/eventsContent/files/corpling/corpora2011/Kotov_211.pdf (дата обращения: 15.3.2023).

17. Русская фонетика. Интонационные конструкции. Интонация [Электронный ресурс]. — Режим доступа, URL: http://www.philol.msu.ru/~fonetica/index1.htm (дата обращения: 15.3.2023).

18. Русская фонетика. Интонационные конструкции. Перечень ИК. [Электронный ресурс]. — Режим доступа, URL: http://www.philol.msu.ru/~fonetica/intonac/ik/ik1.htm (дата обращения: 15.3.2023).

19. *Коврижкина Д. Г.* Связь интонации с выражением эмоций (на примере эмоции удивления) // Вестник Ленинградског200государственного университета им. АС Пушкина. — 2014. — Т. 1, № 3. — С. 233—243.

20. *Киселев В.*, *Давыдов А.*, *Ткаченя А.* Система определения эмоционального состояния диктора по голосу. — 2012.

21. *Рабинер Л.*, *Гоулд Б.* Теория и применение цифровой обработки сигналов. — Рипол Классик, 1978.

22. *Gerhard D.* Pitch extraction and fundamental frequency: History and current techniques. — Department of Computer Science, University of Regina Regina, SK, Canada, 2003.

23. *Zahorian S. A.*, *Dikshit P.*, *Hu H.* A spectral-temporal method for pitch tracking // Ninth International Conference on Spoken Language Processing. — 2006.

24. *Ярцева В. Н.* Лингвистический энциклопедический словарь. — Советская энциклопедия, 1990.

25. *Прокофьева Л.* Распознавание эмоций по характеристикам речевого сигнала (лингвистический, клинический, информационный аспекты) / Л. П. Прокофьева [и др.] // Сибирский филологический журнал. — 2021. — № 2. — С. 325—336.

26. Первичный анализ речевых сигналов [Электронный ресурс]. — Режим доступа, URL: https://alphacephei.com/ru/lecture1.pdf (дата обращения: 17.2.2023).

27. *Chauhan P.*, *Desai N.* Mel Frequency Cepstral Coefficients (MFCC) based speaker identification in noisy environment using wiener filter // Proceedings of International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE 2014). — 2014.

28. *Кошеков К.* Метод автоматической классификации эмоционального состояния диктора по голосу / К. Кошеков [и др.] // Динамика систем, механизмов и машин. — 2020. — Т. 8, № 4. — С. 51—59.

29. *Farrús M.*, *Hernando J.*, *Ejarque P.* Jitter and shimmer measurements for speaker recognition // 8th Annual Conference of the International Speech Communication Association; 2007 Aug. 27-31; Antwerp (Belgium): ISCA; 2007. p. 778-81. — International Speech Communication Association (ISCA). 2007.

30. *Baken R. J.*, *Orlikoff R. F.* Clinical measurement of speech and voice. — Cengage Learning, 2000.

31. Нейронные сети [Электронный ресурс]. — Режим доступа, URL: https://logic.pdmi.ras.ru/~sergey/oldsite/teaching/asr/notes-11-neural.pdf (дата обращения: 9.3.2023).

32. *Дюк В.*, *Самойленко А.* Data Mining: учебный курс. — СПб.: Питер, 2001.

33. *Lanjewar R. B.*, *Chaudhari D.* Speech emotion recognition: a review // International Journal of Innovative Technology and Exploring Engineering (IJITEE). — 2013. — Т. 2, № 4. — С. 68—71.

34. *Devlin J.* Bert: Pre-training of deep bidirectional transformers for language understanding / J. Devlin [и др.] // arXiv preprint arXiv:1810.04805. — 2018.

35. Build simple, secure, scalable systems with Go [Электронный ресурс]. — Режим доступа, URL: https://go.dev/ (дата обращения: 20.4.2023).

36. Welcome to Python.org [Электронный ресурс]. — Режим доступа, URL: https://www.python.org/ (дата обращения: 20.4.2023).

37. PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс]. — Режим доступа, URL: https://www.postgresql.org/ (дата обращения: 20.4.2023).

38. Docker: Accelerated, Containerized Application Development [Электронный ресурс]. — Режим доступа, URL: https://www.docker.com/ (дата обращения: 20.4.2023).

39. *McFee B.* librosa: Audio and music signal analysis in python / B. McFee [и др.] // Proceedings of the 14th python in science conference. Т. 8. — 2015.

40. *Boodidhi S.* Using smoothing techniques to improve the performance of Hidden Markov's Model. — 2011.

41. YAML: YAML Ain't Markup Language [Электронный ресурс]. — Режим доступа, URL: https://yaml.org/ (дата обращения: 20.4.2023).

ПРИЛОЖЕНИЕ А