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 ИУ7-76Б _____ Т. А. Казаева

(Подпись, дата)

Руководитель _____ Ю. В. Строганов

(Подпись, дата)

2023 г.

РЕФЕРАТ

Расчетно–пояснительная записка 68 с., 8 рис., 5 табл., 41 ист., 0 прил.

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.

ВВЕДЕНИЕ

Человеческая речь является одними из самых естественных средств общения. Она содержит широкий спектр разнообразной информации, в том числе и эмоциональной. 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.

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. В настоящей работе будет рассмотрен метод определения речевых эмоций, использующих скрытые марковские модели – статистический классификатор, который используется для моделирования последовательностей.

1 Аналитический раздел

1.1. Категоризация эмоциональных данных

Одной из главных проблем в исследованиях, связанных с определением эмоционального состояния диктора по голосу, является отсутствие четкого определения эмоции. Подход к классификации эмоций влияет на процесс аннотирования. Сегодня широко используются три подхода к категоризации эмоциональных данных: дискретный, многомерный и гибридный.

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. Первоначальная версия 1999 года также включала удивление. [2; 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, об-

разованное 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. Наборы данных речевых эмоций

1.2.1. Наборы данных на иностранных языках

Набор 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 дискретное 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.

Набор данных SAVEE [11] *(англ. Surrey Audio-Visual Expressed Emotion)* состоит из речи четырех актеров мужского пола, говорящих на английском с британским акцентом. Эмоциональные типы для каждого высказывания соответствуют 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.

Набор 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.

Набор 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. Наборы данных на русском языке

Набор данных RUSLANA. [14] Является первым рускоязычным 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.

Набор данных DUSHA [15] – русскоязычный набор эмоциональных данных. Состоит из нарезки русскоязычных подкастов, содержащие до пяти слов. Все аудиозаписи были сделаны на профессиональные микрофоны. Разметка осуществлялась также согласно дискретной модели и содержала следующие эмоции: радость, грусть, злость. Также присутствуют безэмоциональные записи. Набор содержит примерно из 300 000 аудиозаписей, суммарная длительность которых около 350 часов. В разметке содержится 1,5 млн записей.

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 0,5-1 с) 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 корпуса	*Количество эмоций в разметке*	*Количество голосов*		*Лексикон*	*Есть в открытом доступе*
		мужских	*женских*		
RAVDESS	7	12	12	2 предл.	да
SAVEE	6	4	0	15 предл.	да
Emo-DB	6	5	5	10 предл.	да
TESS	7	0	2	200 слов	да
RUSLANA	4	12	49	10 предл.	нет
DUSHA (crowd)	4	-	-	обширный	да
DUSHA (podcast)	4	-	-	обширный	да
REC	-	-	-	обширный	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.

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. Однако, 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 вчислений 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 (например, частота основного тона или превых формант) 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 датасетов эмоции симулируются, то есть привлекается небольшая группа профессиональных актеров, озвучивающих заданные фразы. 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

Важной особенностью речевого сигнала является его условная стационарность на небольших промежутках (от 20 до 40 мс). [20] По этой причине для оцифровки сигнал разделяется на фреймы. Деление происходит таким образом, чтобы каждая точка перекрывалась дважды. [21]

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

Даже при работе с фреймами, 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: [22]

– 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 характеристики, основанные на количественной оценке ЧОТ, 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, энергетические (джиттер и шиммер) и пертурбационные параметры.

1.3.1. Просодические характеристики

1.3.1.1. Частота основного тона

Значение частоты основного тона зависит от размеров и степени натяжения связок. [23] 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. [24]

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 ос-

новного 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 фрейма 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 анализ. Кепстр – результат преобразования, получаемого применением обратного преобразования Фурье к логарифму спектра мощности сигнала. Иными словами, кепстр является спектром от спектра сигнала.

Алгоритм получения кепстра можно разделить на три этапа.

1. К сигналу применяется дискретное преобразование Фурье.
2. От полученного спектра мощности сигнала берется логарифм, чтобы получить лог-спектр.
3. К лог-спектру применяется обратное преобразование Фурье. На выходе получается спектр от спектра сигнала.

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 [25] 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* (1.2):

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

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
Страх	больше	длиннее

Темпом речи называют скорость произнесения элементов речи (звуков, слогов, слов). Темп речи изменяется двумя параметрами: [26]

– 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 [27], 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*). [28]

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. [29]

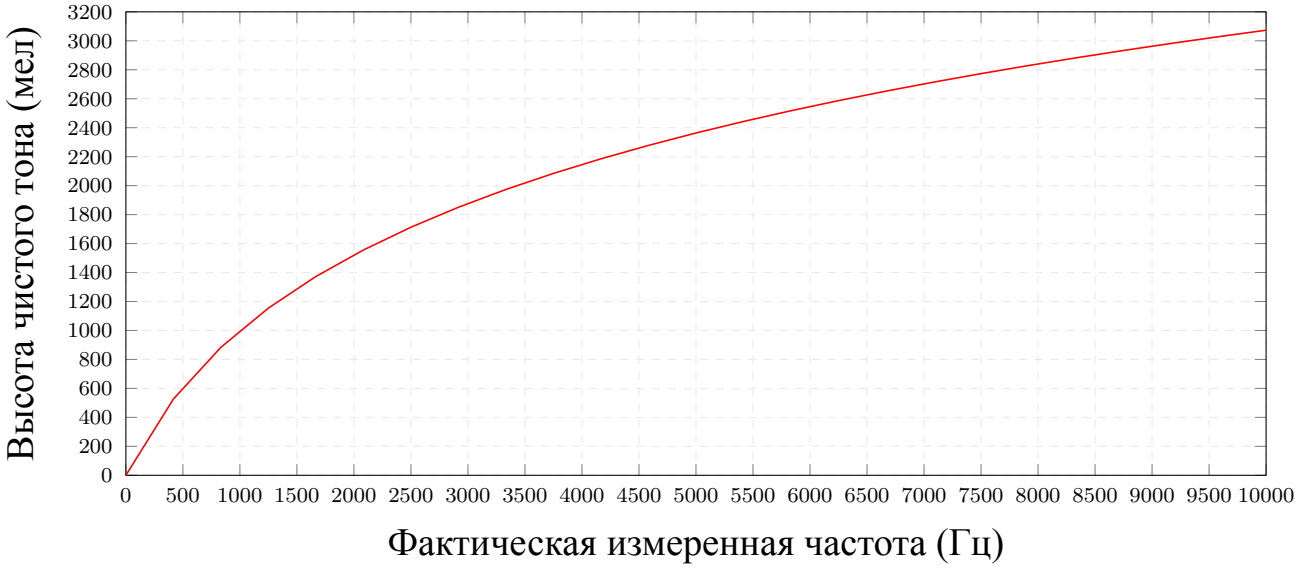

Рис. 1.1 – График зависимости частоты от мел

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

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

После получения $c_j(m)$, коэффициент $c_j(0)$ отбрасывается, так как он не несет информации о речи диктора и задает постоянное смещение. [20]

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 зна-

чения, которое рассчитывается на окне в 3 или 5 выборок [30]:

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

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):

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

Существуют варианты, определения шиммера как относительное отклонение амплитуды от локально-усредненного значения на интервале в P выборок(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. [31]

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 согласно 1.12:

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

при переходе к лог-спектру (1.13) и кепстру (1.14) можно получить следующие выражения:

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

$$X(\bar{\omega}) = S(\bar{\omega}) + H(\bar{\omega}) \tag{1.14}$$

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:

$$l(n) = \begin{cases} 1, n < \tau \\ -1, n \geq \tau \end{cases} \tag{1.15}$$

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, метод k-ближайших соседей. 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 более подходящий и более мощный 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 состояний $q_1, \ldots, q_n$ 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 опре-

делить 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 1.16:

$$P(D|\lambda) = \sum_{Q} P(D|\, Q,\, \lambda) p(Q|\, \lambda) \tag{1.16}$$

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). [32] 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: [33]

– 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 (1.17):

$$\sum_{i=1}^{N} w_i x_i - T, \tag{1.17}$$

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

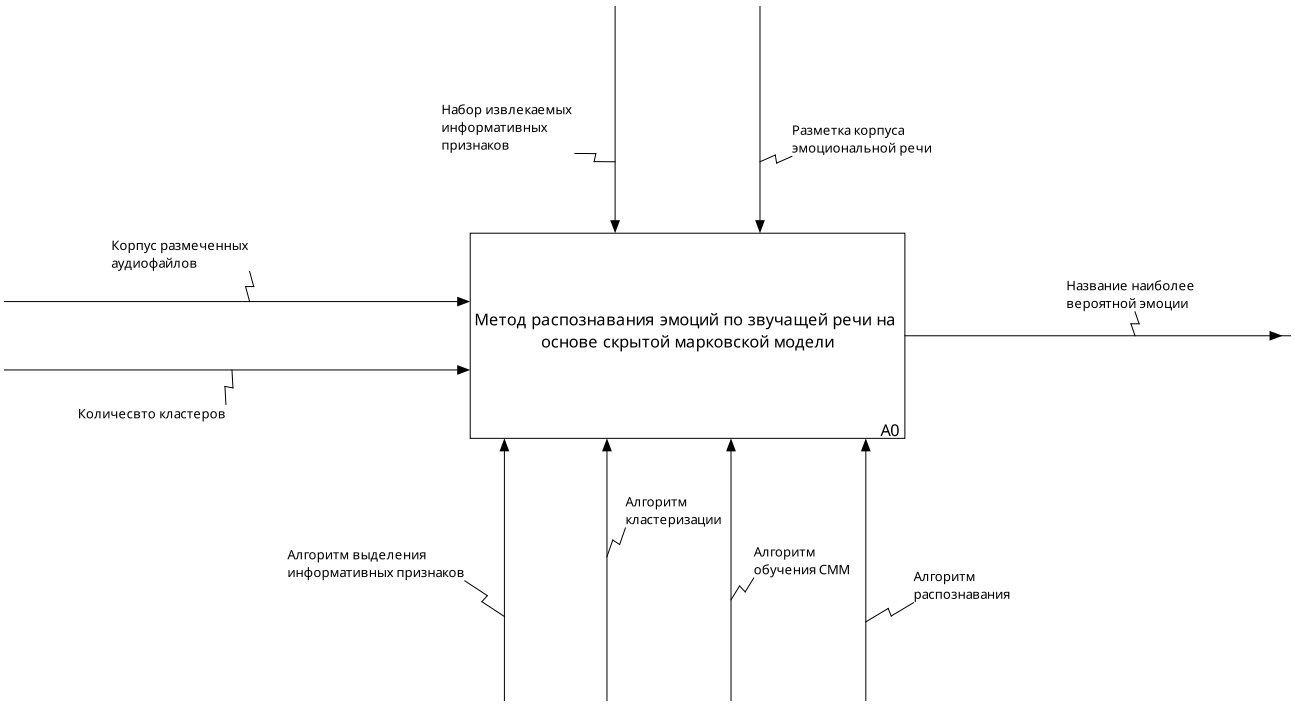

Рис. 1.2 – IDEF0-диаграмма нулевого уровня

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 (фреймы) 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:

– мел-кепстральные коэффициенты представляют собой широкий спектр признаков, которые могут быть маркерами конкретной эмоции, таких как громкость, длительность, скорость и ритм;

– мел-кепстральные коэффициенты более устойчивы к шуму, чем, например, частоты первых 4-х формант или джиттер, поскольку они учитывают особенности восприятия звуковой информации человеческим ухом.

Для обучения классификатора эти признаки должны быть поделены на кластеры. Обученный классификатор используется для распознавания эмоций.

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.

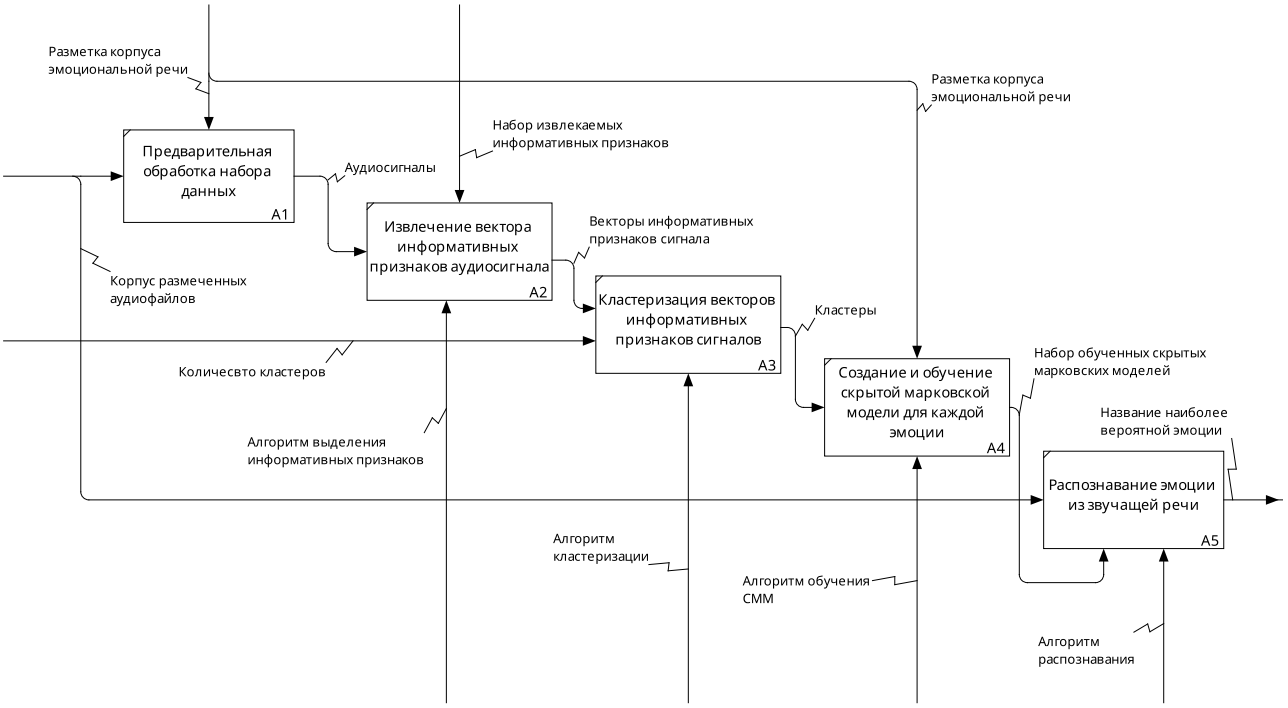

Рис. 2.1 – IDEF0-диаграмма первого уровня

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 [34]. 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.

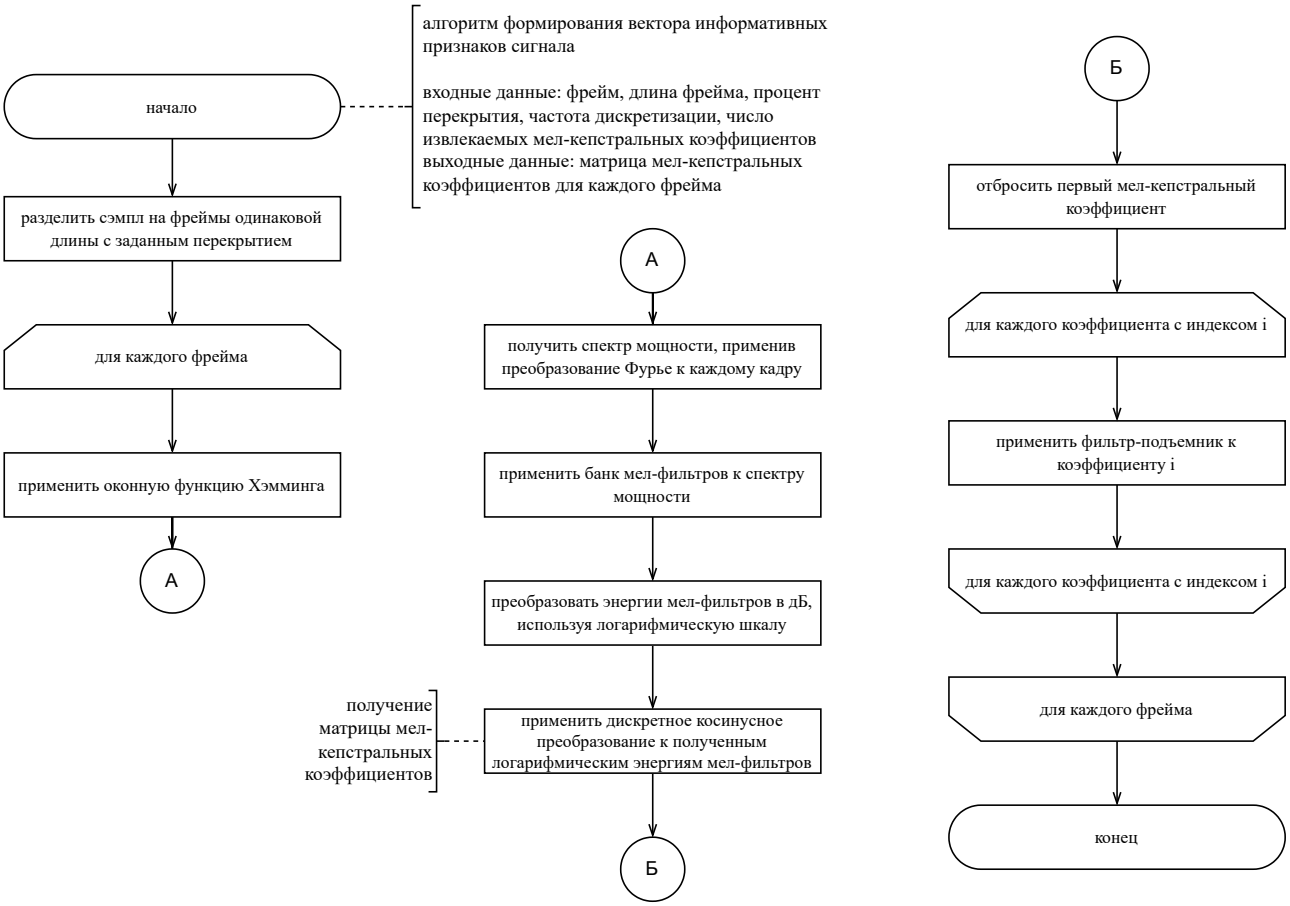

Рис. 2.2 – Алгоритм формирования вектора информативных признаков

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.

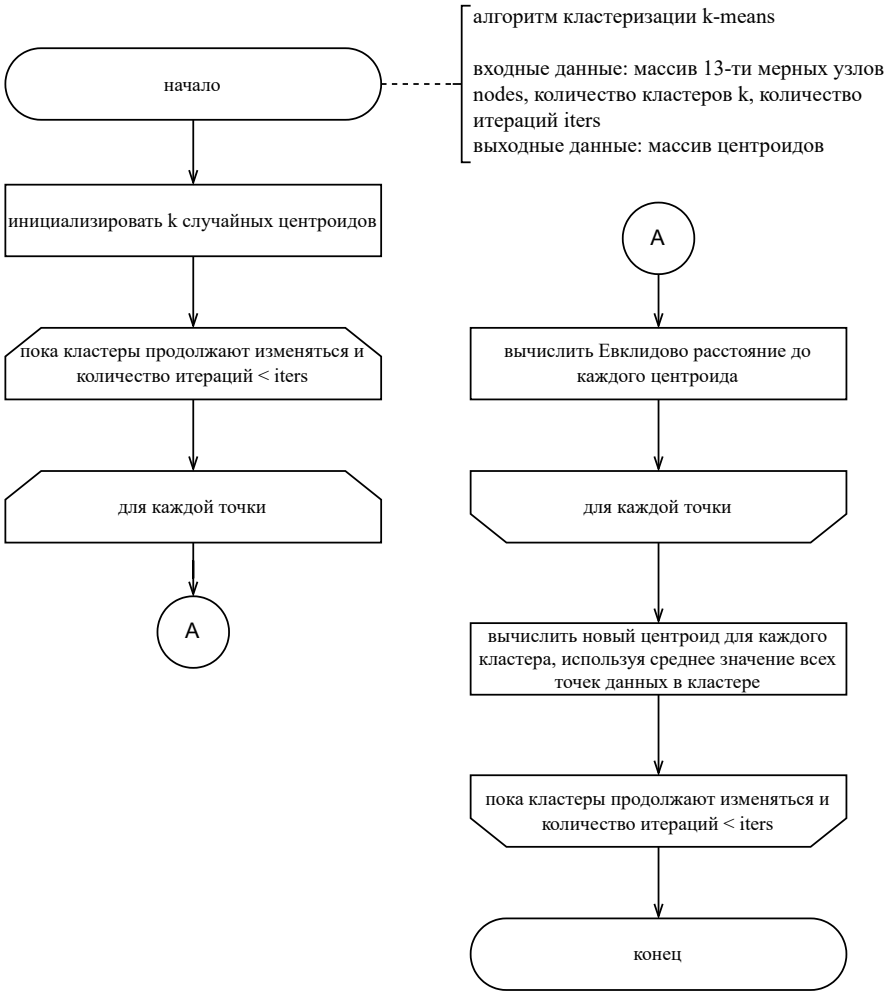

Рис. 2.3 – Алгоритм кластеризации k-means

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.

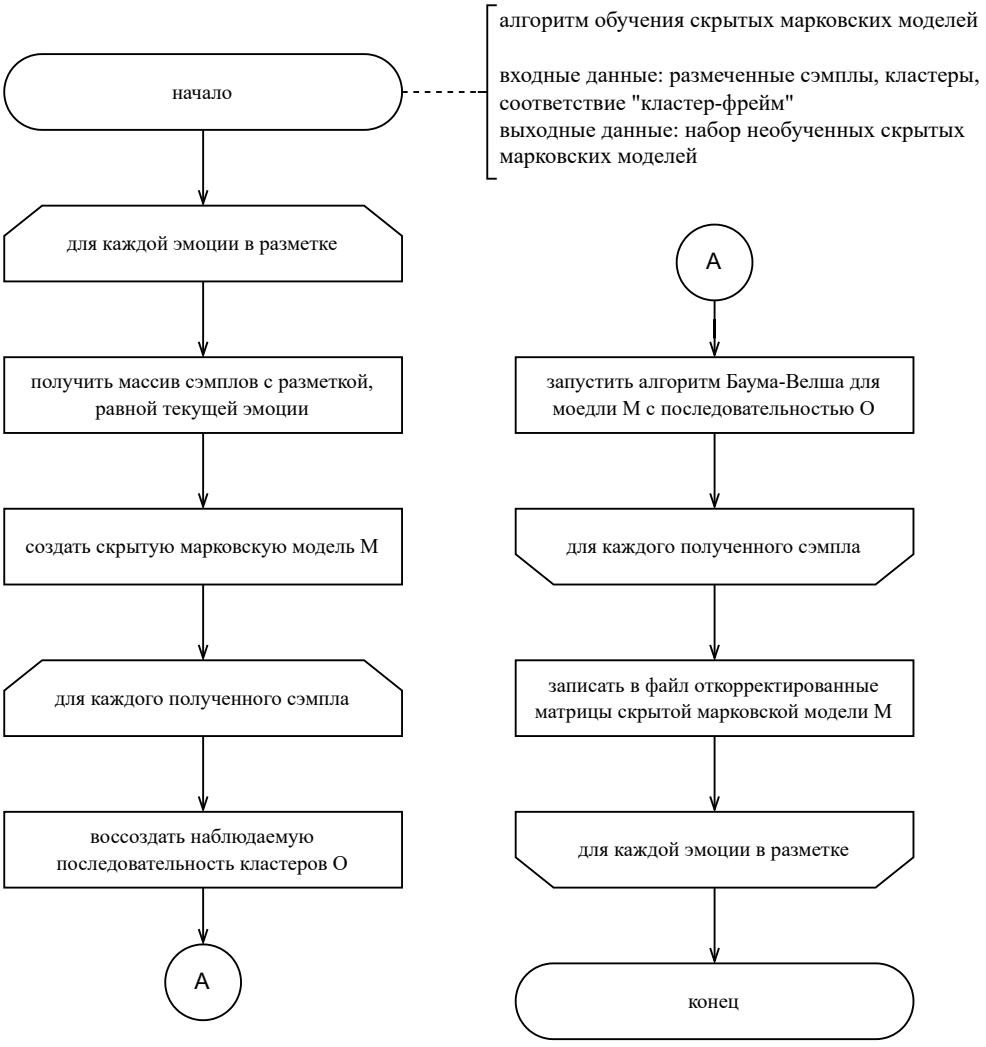

Рис. 2.4 – Обучение скрытых марковских моделей

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.

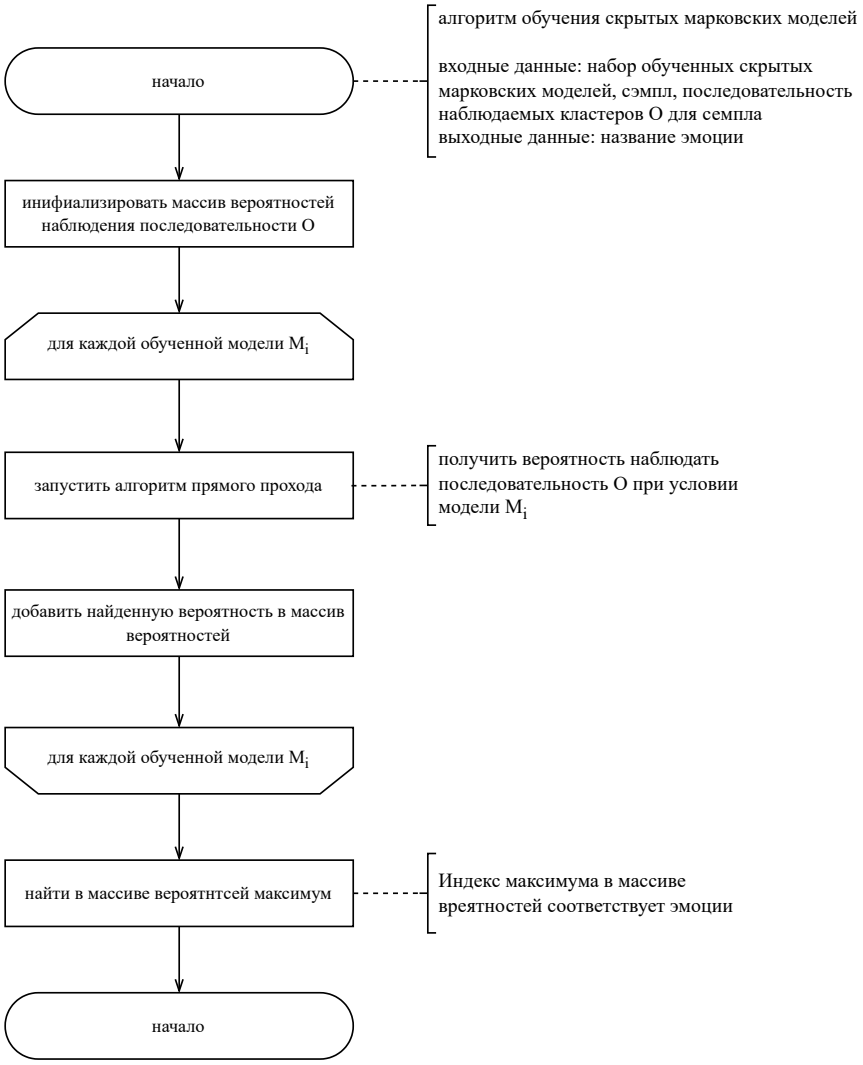

Рис. 2.5 – Выделение эмоции из аудиосигнала

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 [35] 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

Поскольку разметку осуществляли несколько аннотаторов, требовалась агрегарция разметки. Она производилась по методу Дэвида — Скина с порогом 0.9, выбранным эмпирически [15]. 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.

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.

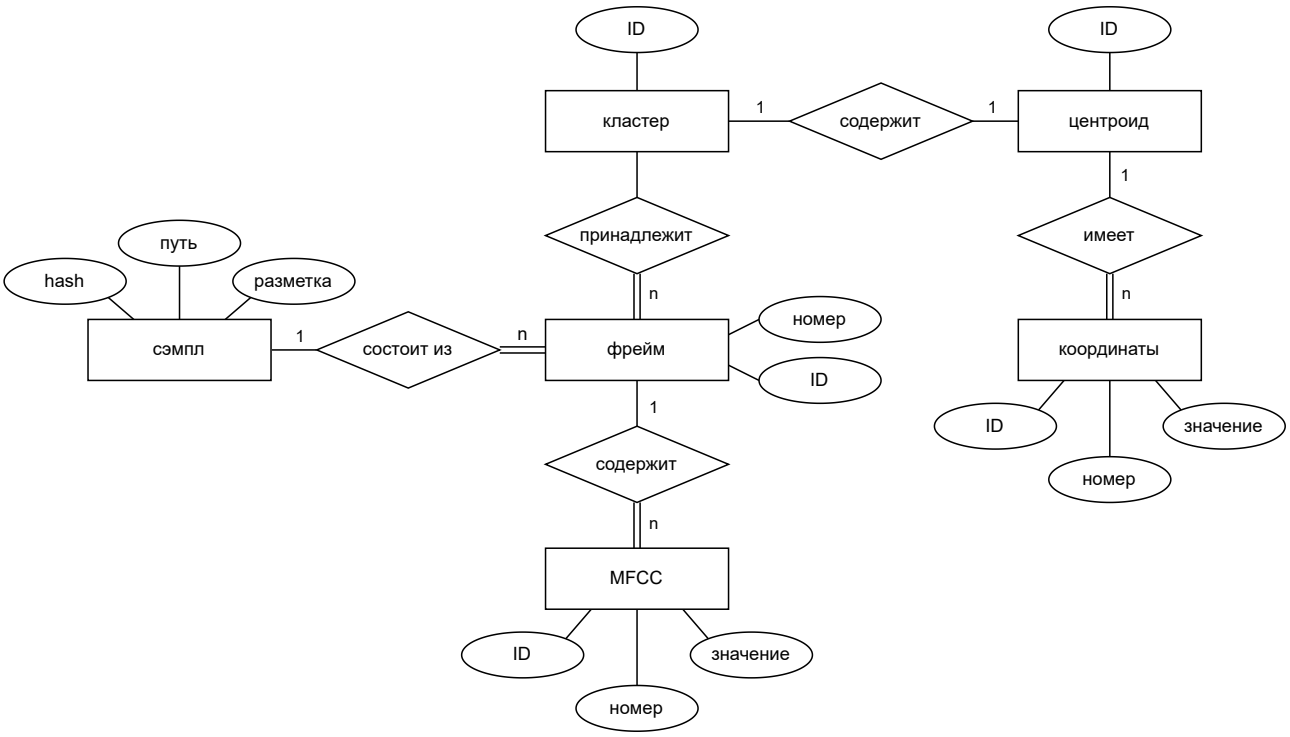

Рис. 2.6 – Диаграмма сущностей базы данных в нотации Чена

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 сэмпле 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.

2.5. Физические компоненты системы и их размещение на устройствах

Здесь диаграмма развертывания

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 [36], 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 [37]. 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 [38], 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 [39], 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

Сервис 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,
            "emo": self.emo
        }
```

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=0.02, hop_length=0.01):
        sig, self.sampling_rate = librosa.load(file_path)

        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:
            mfcc_per_frame = librosa.feature.mfcc(y=frame,
                                  n_fft=self.frame_length, n_mfcc=n_mfcc)
            mfcc.append(mfcc_per_frame.T)

        return np.array(mfcc)
```

При обработке аудиофайла (чтение, деление на кадры и извлечение мел-кепстральных коэффициентов) была использована библиотека «Librosa» [40], 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()
```

Компонент формирования вектора информативных признаков представляет из себя веб-сервер, принимающий HTTP-запросы. Когда клиентское приложение обращается через метод «POST» на конечную точку «*/api/v1/samples*», сервер выполняет функцию чтения аудиофайлов из файла разметки, затем формирует векторы мел-кепстральных коэффициентов для каждого прочитанного аудиофайла и производит запись полученного набора векторов в базу данных для дальнейшего его использования другими компонентами ПО.

3.2.2. Кластеризация

Сервис кластеризации выполняет две основные задачи:

– кластеризация векторов информативных признаков;

– 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. Сервисная структура SampleService, представленная на листинге 3.8 реализует эту логику.

Листинг 3.8 – Функция нахождения центроидов кластеров

```
type SampleService struct {
    repo *postgres.SamplePostgres
}

func (s *SampleService) GetAll() ([]entity.Sample, error) {
    var samples []entity.Sample

    samples, err := s.repo.Get()

    if err != nil {
        return nil, err
    }

    return samples, nil
}

func ... CollectAllFrames(samples []entity.Sample) ([]entity.Frame, error) {

    frames := make([]entity.Frame, 0)

    for i := 0; i < len(samples); i++ {
        for j := 0; j < len(samples[i].Frames); j++ {
            frames = append(frames, samples[i].Frames[j])
        }
    }

    return frames, nil
}
```

Функции структуры оперируют модельной структурой Sample сущности «аудиофайл» (листинг 3.9), полям которой присвоены json-метки, указывающие на соответствие полей структуры полям таблицы аудиофайлов в базе данных.

Листинг 3.9 – Модельная структура сущности «аудиофайл»

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

Последнее поле модельной структуры – кадры аудиофайла. На листинге 3.10 представлена модельная структура сущности «кадр», полям которых присвоены json-метки, указывающие на соответствие полей структуры полям таблицы кадров в базе данных.

Листинг 3.10 – 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
}
```

Каждая используемая сущность имеет соответствующие репозитории. Поля обоих репозиториев – клиентское соединение с базой данных PostgreSQL. На листинге 3.11 представлен репозиторий сущности «аудиофайл».

Листинг 3.11 – Репозиторий сущности «аудиофайл»

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

func NewSamplePostgres(cli *psqlcli.Client) *SamplePostgres {
    return &SamplePostgres{db: cli.DB}
}

func (s *SamplePostgres) Get() ([]entity.Sample, error) {
    var samples []entity.Sample

    err := s.db.Select(&samples,
        `SELECT uuid, audio_path, emotion FROM sample`)
    if err != nil {
        return nil, err
    }

    return samples, nil
}
```

Аналогичную структуру имеет репозиторий сущности «кадр». Кроме операции чтения кадра, в репозитории присутствует логика записи информации о кластере в базу данных. На листинге 3.12 представлен фрагмент репозитория сущности «кадр», выполняющий эту функцию.

Листинг 3.12 – Фрагмент репозитория сущности «кадр»

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

func NewFramePostgres(cli *psqlcli.Client) *FramePostgres {
    return &FramePostgres{db: cli.DB}
}

...

func (f *FramePostgres) AssignCluster(clusterID, frameID string) error {
    query := "UPDATE frame SET cluster_uuid = $1 WHERE uuid = $2"
    _, err := f.db.Exec(query, clusterID, frameID)

    return err
}
```

Кластер ставится в соответствие кадру в результате выполнения функции Nearest, представленной на листинге 3.13.

Листинг 3.13 – 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, а соответствующие ему модельные структуры сущности «кластер» и сущности «центроид». Модельная структура сущности «кластер» представлена на листинге 3.14.

Листинг 3.14 – Модельная структура сущности «кластер»

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

Модельная структура сущности «центроид» представлена на листинге 3.15.

Листинг 3.15 – Модельная структура сущности «центроид»

```
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 3.16.

Листинг 3.16 – Установка соответствия между центроидом кластера и его модельной структурой

```
type ClusterService struct {
    repo   *postgres.ClusterPostgres
    logger logging.Logger
}

func ... constructClusterData(centroids []entity.Centroid) []entity.Cluster {
    clusters := make([]entity.Cluster, len(centroids))
    for i, centroid := range centroids {
        clusters[i] = entity.Cluster{
            ID:       uuid.New().String(),
            Index:    i + 1,
            Centroid: centroid,
        }
    }
    return clusters
}
```

Аналогичная функция представлена для установки соответствия между индексом центроида и его модельной структурой Centroid (листинг 3.17).

Листинг 3.17 – Установка соответствия между индексом центроида и его модельной структурой

```
func ... constructCentroidsData(centroidsCoords []kmeans.Node) ... {
    centroids := make([]entity.Centroid, len(centroidsCoords))
    for i, centroid := range centroidsCoords {
        centroids[i] = entity.Centroid{
            ID:    uuid.New().String(),
            Value: centroid,
        }
    }
    return centroids
}
```

Функция AssignClusters сервисного класса ClusterService реализует логику установки соответствия «кадр-кластер» (листинг 3.18).

Листинг 3.18 – Установка соответствия «кадр-кластер»

```
func ... AssignClusters(frames []entity.Frame, nclusters, maxRounds int) ... {
    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
}

func ... collectFramesData(frames []entity.Frame) []kmeans.Node {
    nodes := make([]kmeans.Node, len(frames))
    for i, fm := range frames {
        nodes[i] = fm.MFCCs
    }
    return nodes
}
```

Для того, чтобы определить качество кластеризации рассчитывается коэффициент силуэта (листинг 3.19). Коэффициент силуэта является мерой качества кластеризации и принимает значения от -1 до 1, где более высокие значения указывают на более качественную кластеризацию. Значения близкие к -1 указывают на перекрытие кластеров, а отрицательные значения указывают на неправильную кластеризацию.

Листинг 3.19 – Определение коэффициента силуэта

```
func SilhouetteCoefficient(observations []Node, centroids []Node) float64 {
    numClusters := len(centroids)
    numObservations := len(observations)
    clusterAssignments, distances := assignToCluster(observations, centroids)
    a := computeAScores(observations, clusterAssignments, distances)
    b := computeBScores(observations, centroids,
            clusterAssignments, numClusters)

    var sumSC float64
    for i := 0; i < numObservations; i++ {
        if b[i] != 0 {
            sc := (b[i] - a[i]) / math.Max(a[i], b[i])
            sumSC += sc
        }
    }
    if numObservations > 0 {
        return sumSC / float64(numObservations)
    } else {
        return 0.0
    }
}
```

Функция вычисления индексов кластеров, к которым относится каждое наблюдение, представлена на листинге 3.20.

Листинг 3.20 – Вычисления индексов кластеров для каждого наблюдения

```
func assignToCluster(obs []Node, centroids []Node) ([]int, [][]float64) {
    ...
    for i := 0; i < numObservations; i++ {
        distances[i] = make([]float64, numClusters)
        for j := 0; j < numClusters; j++ {
            distances[i][j] = distance(obs[i], centroids[j])
        }
        minDist := distances[i][0]
        minIndex := 0
        for j := 1; j < numClusters; j++ {
            if distances[i][j] < minDist {
                minDist = distances[i][j]
                minIndex = j
            }
        }
        clusterAssignments[i] = minIndex
    }
    return clusterAssignments, distances
}
```

Функция принимает на вход набор наблюдений obs и набор центроидов centroids, и возвращает массив clusterAssignments с индексами кластеров, к которым были присвоены наблюдения, а также двумерный массив distances, содержащий расстояния между каждым наблюдением и каждым центроидом. Она используется для вычисления индексов кластеров, к которым каждое наблюдение относится, и расстояний между каждым наблюдением и каждым центроидом, что позволяет затем вычислить новые центроиды в функции updateCentroids.

Вычисление значения $a(i)$ – среднего расстояния между точкой i и всеми другими точками из того же кластера происходит в функции computeAScores (листинг 3.21)

Листинг 3.21 – Вычисления значения $a(i)$

```
1  func computeAScores(observations []Node, clusterAssignments []int,
2                      distances [][]float64) []float64 {
3      numObservations := len(observations)
4      a := make([]float64, numObservations)
5
6      for i := 0; i < numObservations; i++ {
7          clusterIndex := clusterAssignments[i]
8
9          var sumDist float64
10         numSameCluster := 0
11         for j := 0; j < numObservations; j++ {
12             if clusterAssignments[j] == clusterIndex && i != j {
13                 sumDist += distances[i][clusterIndex]
14                 numSameCluster++
15             }
16         }
17         if numSameCluster > 0 {
18             a[i] = sumDist / float64(numSameCluster)
19         }
20     }
21     return a
22 }
```

Конечный шаг для вычисления коэффициента силуэта – это вычисление оценок $b(i)$ (листинг 3.22) для каждого i-го наблюдения. Для каждого i-го наблюдения вычисляется среднее расстояние между i и всеми точками в других кластерах. Минимальное среднее расстояние $b(i)$ затем используется для вычисления коэффициента силуэта.

Листинг 3.22 – Вычисления значения $b(i)$

```
func computeBScores(observations []Node, centroids []Node,
                    clusterAssignments []int, numClusters int) []float64 {
    numObservations := len(observations)
    distances := make([][]float64, numObservations)
    for i := 0; i < numObservations; i++ {
        distances[i] = make([]float64, numClusters)
        for j := 0; j < numClusters; j++ {
            distances[i][j] = distance(observations[i], centroids[j])
        }
    }

    b := make([]float64, numObservations)
    for i := 0; i < numObservations; i++ {
        clusterIndex := clusterAssignments[i]
        minAvgDist := math.Inf(1)
        for j := 0; j < numClusters; j++ {
            if j != clusterIndex {
                var sumDist float64
                numOtherCluster := 0
                for k := 0; k < numObservations; k++ {
                    if clusterAssignments[k] == j {
                        sumDist += distances[i][j]
                        numOtherCluster++
                    }
                }
                if numOtherCluster > 0 {
                    avgDist := sumDist / float64(numOtherCluster)
                    if avgDist < minAvgDist {
                        minAvgDist = avgDist
                    }
                }
            }
        }
        b[i] = minAvgDist
    }
    return b
}
```

Компонент кластеризации также представляет из себя веб-сервер, принимающий HTTP-запросы. Обращение к нему осуществляется с использованием метода «POST» на конечную точку «*/api/v1/clusters*», в этом случае сервер выполняет функцию кластеризации и записи кластеров в базу данных для дальнейшей работы с ними.

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 3.23 представлена модельная структура скрытой марковской модели.

Листинг 3.23 – 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). Функция allocateRandomMatrix (листинг 3.24) создает матрицы, заполненные случайными вероятностями.

Листинг 3.24 – Создание матриц со случайными вероятностями

```
func allocateRandomMatrix(n, m int, seed *rand.Rand) [][]float64 {
    matrix := make([][]float64, n)
    for i := range matrix {
        matrix[i] = make([]float64, m)
        sum := 0.0
        for j := range matrix[i] {
            matrix[i][j] = seed.Float64()
            sum += matrix[i][j]
        }
        for j := range matrix[i] {
            matrix[i][j] /= sum
        }
    }

    return matrix
}
```

На листинге 3.25 представлена случайная инициализация матриц скрытой марковской модели.

Листинг 3.25 – Создание 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. Репозиторий сущности «аудиофайл» идентичен 3.11, модельная структура идентична 3.9. Однако, в модельной структуре сущности «фрейм» (листинг 3.26) присутствуют не значения мел-кепстральных коэффициентов, а индексы кластеров, к которым принадлежит соответствующий набор значений мел-кепстральных коэффициентов.

Листинг 3.26 – 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"`
}
```

Поле Frames модельной структуры Sample содержит последовательность индексов наблюдаемых кластеров. В дальнейшем эта последовательность будет использована как входные данные для алгоритмов обучения и распознавания. Создание последовательности наблюдений осуществляется в функции

ConstructObservationSequence сервисного класса SampleService (листинг 3.27).

Листинг 3.27 – Создание последовательности наблюдений

```
func ... ConstructObservationSequence(sample entity.Sample) []int {
    observations := make([]int, len(sample.Frames))
    for i := 0; i < len(observations); i++ {
        observations[i] = sample.Frames[i].ClusterIndex - 1
    }
    return observations
}
```

Скрытая марковская модель, в качестве состояний использует кластеры, полученные в компоненте кластеризации. Кластеров при обработке большого количества данных может получиться много, поэтому вероятности нахождения скрытой марковской модели в заданном состоянии могут быть небольшими и, следовательно, при их перемножении может возникнуть проблема с числовой точностью.

Процедура прямого прохода, вычисления в которой производятся в логарифмической шкале для предотвращения проблем с числовой точностью, представлена на листинге 3.28.

Листинг 3.28 – Процедура прямого прохода

```
func ... ForwardAlgorithm(obs []int, alpha [][]float64) float64 {
    for i := 0; i < len(hmm.Transitions); i++ {
        alpha[0][i] = math.Log(hmm.StationaryProbabilities[i]) +
        math.Log(hmm.Emissions[i][obs[0]])
    }
    for t := 1; t < len(obs); t++ {
        for j := 0; j < len(hmm.Transitions); j++ {
            sum := math.Inf(-1)
            for i := 0; i < len(hmm.Transitions); i++ {
                sum = logAdd(sum, alpha[t-1][i] +
                math.Log(hmm.Transitions[i][j]))
            }
            alpha[t][j] = sum + math.Log(hmm.Emissions[j][obs[t]])
        }
    }
    logLikelihood := math.Inf(-1)
    for i := 0; i < len(hmm.Transitions); i++ {
        logLikelihood = logAdd(logLikelihood, alpha[len(obs)-1][i])
    }
    return logLikelihood
}
```

Процедура обратного прохода представлена на листинге 3.29. Как и в случае процедуры прямого прохода, вычисления осуществляются в логарифмической шкале.

Листинг 3.29 – Процедура обратного прохода

```
func ... BackwardAlgorithm(obs []int, beta [][]float64) {
    for i := 0; i < len(hmm.Transitions); i++ {
        beta[len(obs)-1][i] = 0.0
    }

    for t := len(obs) - 2; t >= 0; t-- {
        for i := 0; i < len(hmm.Transitions); i++ {
            logSum := math.Inf(-1)

            for j := 0; j < len(hmm.Transitions); j++ {
                logSum = logAdd(logSum,
                math.Log(hmm.Transitions[i][j]) +
                        math.Log(hmm.Emissions[j][obs[t+1]])+beta[t+1][j])
            }

            beta[t][i] = logSum
        }
    }
}
```

Вычисление γ-вероятностей представлен на листинге 3.30.

Листинг 3.30 – Вычисление γ-вероятностей

```
func ... computeGamma(obs []int, alpha [][]float64,
                      beta [][]float64, gamma [][]float64) {
    for t := 0; t < len(obs); t++ {
            sum := math.Inf(-1)

            for i := 0; i < len(hmm.Transitions); i++ {
                gamma[t][i] = alpha[t][i] + beta[t][i]
                sum = logAdd(sum, gamma[t][i])
            }

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

α, β и γ вероятности используются при вычислении ξ-вероятностей. Вычисле-

ние ξ-вероятностей представлено на листинге 3.31.

Листинг 3.31 – Вычисление ξ-вероятностей

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

    for t := 0; t < len(obs)-1; t++ {
        maxVal := math.Inf(-1)
        for i := 0; i < len(hmm.Transitions); i++ {
            for j := 0; j < len(hmm.Transitions); j++ {
                if xi[t][i][j] > maxVal {
                    maxVal = xi[t][i][j]
                }
            }
        }

        sum := 0.0

        for i := 0; i < len(hmm.Transitions); i++ {
            for j := 0; j < len(hmm.Transitions); j++ {
                xi[t][i][j] = math.Exp(xi[t][i][j] - maxVal)

                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
            }
        }
    }
}
```

α, β, γ и ξ вероятности используются при обучении, или корректировке матриц скрытой марковской модели. Алгоритм Баума-Велша (листинг 3.32), выполняющий корректировку матриц модели, принимает в качестве входных данных массив индексов кластеров и количество итераций корректирования матриц.

Листинг 3.32 – Алгоритм Баума-Велша

```
func ... BaumWelch(observationSequence []int, iterations int) {
    var (
        alpha = make([][]float64, len(observationSequence))
        beta  = make([][]float64, len(observationSequence))
        gamma = make([][]float64, len(observationSequence))
        xi    = make([][][]float64, len(observationSequence)-1)
    )

    for i := 0; i < len(observationSequence); 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(observationSequence)-1 {
            xi[i] = make([][]float64, len(hmm.Transitions))

            for j := 0; j < len(hmm.Transitions); j++ {
                xi[i][j] = make([]float64, len(hmm.Transitions))
            }
        }
    }

    hmm.Emissions = LaplaceSmoothing(hmm.Emissions)

    for it := 0; it < iterations; it++ {
        hmm.ForwardAlgorithm(observationSequence, alpha)
        hmm.BackwardAlgorithm(observationSequence, beta)
        hmm.computeGamma(observationSequence, alpha, beta, gamma)
        hmm.computeXi(observationSequence, alpha, beta, xi)

        // Update the model parameters
        hmm.update(observationSequence, gamma, xi)
    }
}
```

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 [41].

Сглаживания Лапласа заключается в добавлении к каждому элементу матрицы эмиссий некоторого малого значения. В данном случае к каждому элементу строки добавляется единица и затем каждая строка матрицы нормализуется. Функция, выполняющая сглаживание Лапласа, представлена на листинге 3.33.

Листинг 3.33 – сглаживание Лапласа

```
func LaplaceSmoothing(emissionMatrix [][]float64) [][]float64 {

    rows := len(emissionMatrix)
    cols := len(emissionMatrix[0])

    smoothedMatrix := make([][]float64, rows)

    for i := 0; i < rows; i++ {
        smoothedMatrix[i] = make([]float64, cols)

        for j := 0; j < cols; j++ {
            smoothedMatrix[i][j] = (emissionMatrix[i][j] + 1) /
                (sum(emissionMatrix[i]) + float64(cols))
        }
    }

    return smoothedMatrix
}
```

Обученные марковские модели решено хранить в JSON-файле. Для каждой эмоции, присутствующей в разметке, создается отдельная скрытая марковская модель. Функция записи матриц скрытой марковской модели в файл представлена на листинге 3.34.

Листинг 3.34 – 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
}
```

Во время первого запуска скрытые марковские модели инициализируются случайным образом (листинг 3.25). Для каждого последующего запуска значения скрытой марковской модели прочитываются из JSON-файла. Функция чтения матриц марковской модели представлена на листинге 3.35.

Листинг 3.35 – Чтение матриц скрытой марковской модели из файла

```
1  func LoadJSON(filename string) (HiddenMarkovModel, error) {
2      file, err := os.Open(filename)
3      if err != nil {
4          return HiddenMarkovModel{}, err
5      }
6      defer file.Close()
7
8      decoder := json.NewDecoder(file)
9
10     var hmm HiddenMarkovModel
11     if err := decoder.Decode(&hmm); err != nil {
12         return HiddenMarkovModel{}, err
13     }
14
15     return hmm, nil
16 }
```

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. Вычисление вероятности наблюдения представлено на листинге 3.28 в строках 17-22.

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.

Интерфейс клиентского приложения состоит из одной веб-страницы, на которой отображается матрица ошибок. Матрица ошибок (*англ. confusion matrix*) представляет собой таблицу, в которой каждая строка соответствует истинному классу объектов, а каждый столбец – прогнозируемому классу. Матрица ошибок позволяет проанализировать результаты работы алгоритма обучения скрытой марковской модели, сравнивая прогнозируемые и фактические значения. В ячейках таблицы указано количество аудиофайлов, которым в результате алгоритма была присвоена та или иная эмоция. Обычно матрица – это квадратная таблица, где диагональные элементы соответствуют правильно классифицированным объектам, а элементы вне диагонали – неправильно классифицированным объектам. На листинге 3.36 представлена инициализация матрицы ошибок с использованием истинных и спрогнозированных значений разметки.

Листинг 3.36 – Инициализация матрицы ошибок

```
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}
}
```

Нормализованная матрица ошибок обычно представляется в виде процентных значений или долей. На листинге 3.37 представлена нормализация матрицы ошибок.

Листинг 3.37 – Нормализация матрицы ошибок

```
func (m *ConfusionMatrix) Normalize() {
    numClasses := len(m.labels)

    rowSums := make([]float64, numClasses)
    totalSum := 0.0
    for i, actualLabel := range m.labels {
        for _, predictedLabel := range m.labels {
            rowSums[i] += m.Values[actualLabel][predictedLabel]
            totalSum += m.Values[actualLabel][predictedLabel]
        }
    }

    normConfusionMatrix := make(map[entity.Label]map[entity.Label]float64)

    for i, actualLabel := range m.labels {
        normConfusionMatrix[actualLabel] = make(map[entity.Label]float64)
        for _, predictedLabel := range m.labels {
            normConfusionMatrix[actualLabel][predictedLabel] =
                m.Values[actualLabel][predictedLabel] / rowSums[i]
        }
    }
    m.Values = normConfusionMatrix
}
```

Компонент создания и обучения марковских моделей, как и два предыдущих, представляет из себя веб-сервер, принимающий HTTP-запросы. Обращение к нему осуществляется с использованием метода «POST» на конечные точки «*/api/v1/train*» и «*/api/v1/test*», осуществляющие процедуры обучения и проверки соответственно.

3.3. Тестирование компонент программного обеспечения

3.4. Формат входных и выходных данных

4 Исследовательский раздел

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Feeling and caring: When mood intervenes in charitable giving / C. K. Hsee, J. Zhang, Z. Lu, F. Xu // 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], URL: https://www.affectiva.com/ (дата обращения: 30.9.2022).
6. *Вундт В.* Психология душевных волнений // Психология эмоций. 1984.
7. RECOLA. [Электронный ресурс], URL: https://diuf.unifr.ch/main/diva/recola/ (дата обращения: 1.10.2022).
8. Sentic Computing for social media marketing / E. Cambria, M. Grassi, A. Hussain, C. Havasi // 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], URL: https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio (дата обращения: 15.3.2023).
11. Combining frame and turn-level information for robust recognition of emotions within speech / B. Vlasenko, B. Schuller, A. Wendemuth, G. Rigoll //. 01.2007. С. 2249—2252.
12. EmoDB Dataset. [Электронный ресурс], 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], 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. Large Raw Emotional Dataset with Aggregation Mechanism / V. Kondratenko, A. Sokolov, N. Karpov, O. Kutuzov, N. Savushkin, F. Minkin // arXiv preprint arXiv:2212.12266. 2022.

16. Русскоязычный эмоциональный корпус: коммуникативное взаимодействие в реальных эмоциональных ситуациях. [Электронный ресурс], 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], URL: http://www.philol.msu.ru/~fonetica/index1.htm (дата обращения: 15.3.2023).

18. Русская фонетика. Интонационные конструкции. Перечень ИК. [Электронный ресурс], 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 Ленинградского государственного университета им. АС Пушкина. 2014. Т. 1, № 3. С. 233—243.

20. Метод автоматической классификации эмоционального состояния диктора по голосу / К. Кошеков, В. Кобенко, Р. Анаятова, А. Савостин, А. Кошеков // Динамика систем, механизмов и машин. 2020. Т. 8, № 4. С. 51—59.

21. Лекция 6. Признаки. Кепстральные коэффициенты. MFCC. [Электронный ресурс], URL: https://logic.pdmi.ras.ru/~sergey/oldsite/teaching/asr/notes-06-features.pdf (дата обращения: 17.2.2023).

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

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

24. Pitch extraction and fundamental frequency: History and current techniques / D. Gerhard [и др.]. Department of Computer Science, University of Regina Regina, SK, Canada, 2003.

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

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

27. Распознавание эмоций по характеристикам речевого сигнала (лингвистический, клинический, информационный аспекты) / Л. П. Прокофьева, И. Л. Пластун, Н. В. Филиппова, Л. Ю. Матвеева, Н. С. Пластун // Сибирский филологический журнал. 2021. № 2. С. 325—336.

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

29. *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.

30. *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).[place unknown]: ISCA; 2007. p. 778-81. International Speech Communication Association (ISCA). 2007.

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

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

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

34. *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.

35. Bert: Pre-training of deep bidirectional transformers for language understanding / J. Devlin, M.-W. Chang, K. Lee, K. Toutanova // arXiv preprint arXiv:1810.04805. 2018.

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

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

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

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

40. librosa: Audio and music signal analysis in python / B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, O. Nieto // Proceedings of the 14th python in science conference. T. 8. 2015.

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

ПРИЛОЖЕНИЕ А