МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И
МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена Трудового Красного Знамени федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Разрешаю
допустить к защите
Зав. кафедрой
Городничев М.Г.
_____ 2023 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

НА ТЕМУ

«Разработка эффективного метода определения нецензурной лексики
в режиме реального времени»

Студент: Лапин Виктор Андреевич _____

Руководитель: Мкртчян Грач Маратович _____

Москва 2023 г.

Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

(название полностью)

«Утверждаю»
Зав. кафедрой Городничев М.Г.
«____»_____ 2023 г.

З А Д А Н И Е
на выпускную квалификационную работу

Студенту Лапину Виктору Андреевичу гр. БВТ1901

Направление (специальность) 09.03.01 «Информатика и вычислительная техника»

Форма выполнения выпускной квалификационной работы Бакалаврская работа

(Дипломный проект, дипломная работа, магистерская диссертация, бакалаврская работа)

Тема выпускной квалификационной работы «Разработка эффективного метода определения нецензурной лексики в режиме реального времени»

Утверждена приказом ректора №87-с от 24 января 2023 г.

1. Исходные данные

1) Закон РФ N 2124-1 "О средствах массовой информации"
2) J. Meng и др. Overview of the Speech Recognition Technology
3) Somshubra Majumdar, Boris Ginsburg. MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition
4) Oleg Rybakov и др. Streaming keyword spotting on mobile devices

2. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)	Объем работы в % и сроки выполнения по разделам
Введение	5% − 23.03.2023
Раздел 1. Теоретическая часть	20% − 10.04.2023
Раздел 2. Проектирование системы	45% − 15.05.2023
Раздел 3. Разработка и тестирование	25% − 25.05.2023
Заключение	5% − 01.06.2023

3. Вопросы конструктивных разработок

4. Разработка вопросов по экологии и безопасности жизнедеятельности

5. Технико-экономическое обоснование (подлежащее расчету)

6. Перечень графического материала (с точным указанием обязательных чертежей)

1) Диаграммы исследований использования нецензурной лексики
2) Схема процесса удаления нецензурной лексики из аудиосигнала
3) Визуализация различных признаковых описаний сигнала
4) Схема модели
5) График точности при обучении модели
6) Матрица ошибок модели
7) Сравнение сигналов до и после обработки

7. Консультанты по ВКР (с указанием относящихся к ним разделов проекта):

_____ _____
(подпись) (ФИО)

_____ _____
(подпись) (ФИО)

8. Срок сдачи студентом законченной ВКР: 21 мая 2023

Дата выдачи задания: 27 января 2023

Руководитель _____ Мкртчян Грач Маратович
(подпись) (ФИО)

почасовая нагрузка
(штатная или почасовая)

Задание принял к исполнению _____
(подпись студента)

Примечание: Настоящее задание прилагается к законченной ВКР

ОТЗЫВ РУКОВОДИТЕЛЯ

на выпускную квалификационную работу студента группы БВТ1901
Лапина Виктора Андреевича
на тему:
«Разработка эффективного метода определения нецензурной лексики в режиме реального времени»

В настоящее время проблема необходимости обнаружения нецензурной лексики довольно актуальна в различных сферах, в первую очередь, в прямых трансляциях. Разработка системы на основе машинного обучения, способной автоматизировать этот процесс, может быть значимой для этих отраслей. Ввиду этого факта, тема выпускной квалификационной работы Лапина Виктора Андреевича является актуальной.

Студент реализовал систему, позволяющую определять нецензурную лексику в аудиопотоке в реальном времени, что позволяет снизить задержку трансляции и избежать попадания в эфир неприемлемого контента.

При работе над ВКР, Лапин В.А. продемонстрировал достаточный уровень теоретических и практических навыков, проявил себя старательным и дисциплинированным, способным самостоятельно решать поставленные задачи на высоком уровне.

В результате выполнения работы Лапин В.А. показал, что владеет всеми компетенциями, предусмотренными учебным планом по направлению 09.03.01 – «Информатика и вычислительная техника».

Существенные замечания к работе отсутствуют.

Учитывая приведённые факты, выпускная квалификационная работа студента 4 курса бакалавриата Лапина Виктора Андреевича может быть допущена к защите на ГЭК, а автор заслуживает по результатам защиты присвоения квалификации (степени) «бакалавр» по направлению подготовки 09.03.01 – «Информатика и вычислительная техника».

Ассистент кафедры
«Математическая кибернетика и
информационные технологии» Г.М. Мкртчян

Аннотация

Выпускная квалификационная работа на тему: «Разработка эффективного метода определения нецензурной лексики в режиме реального времени».

Работа включает: 58 страниц, таблиц — 4, рисунков — 8, формул — 9, приложений — 3, используемых источников – 57.

Ключевые слова: нейронные сети, автоматическое распознавание речи, обнаружение ключевых слов, обработка звуковых сигналов, нецензурная лексика.

Выпускная квалификационная работа состоит из введения, трёх основных разделов и заключения.

Во введении обосновывается актуальность исследования, описываются цели и задачи работы.

В первом разделе исследуются основные факторы употребления нецензурной лексики, рассматриваются программные и аппаратные средства, используемые для обнаружения нецензурной лексики, и выявляются их достоинства и недостатки.

Во втором разделе рассматриваются технологии, позволяющие решить поставленную задачу. Производится сравнение и выбор наиболее подходящей модели и сбор данных для её обучения.

В третьем разделе представлена реализация системы, включая обучение модели, оценку её эффективности и разработку прикладной программы для фильтрации нецензурной лексики на её основе.

В заключении представлены выводы по выполненной работе, возможности развития и область применения проведённого исследования.

Содержание

Введение

На сегодняшний день проблема определения и фильтрации нецензурной лексики имеет важную роль во многих сферах информационной деятельности, но основной является сфера телевизионных и интернет-трансляций в прямом эфире. С ростом информатизации общества всё большую популярность набирают мероприятия, проводимые в прямом эфире, так как они позволяют зрителям взаимодействовать с организаторами и ведущими мероприятия, а также с другими зрителями.

Однако, такая возможность грозит попаданием в эфир нежелательного контента, такого как нецензурные выражения, к примеру при звонке зрителя в студию или при дискуссиях среди приглашённых гостей. Попадание в эфир таких выражений может вызвать негативную реакцию со стороны аудитории, а также привести к нежелательным последствиям для организаторов трансляции.

Обнаружение и фильтрация нецензурной лексики в настоящее время происходит преимущественно вручную, что требует особой концентрации внимания и наличия задержки в трансляции, достаточной для реакции человека, что может быть критичным для некоторых трансляций.

В данной выпускной квалификационной работе представлено проектирование и разработка системы, позволяющей с минимальным вмешательством человека определять и фильтровать нецензурные выражения в аудиопотоке в реальном времени с использованием современных технологий машинного обучения.

Целью данной выпускной квалификационной работы является получение эффективной системы, позволяющей минимизировать вероятность распространения нецензурной лексики в прямых трансляциях для того, чтобы

сохранить лояльность аудитории и избежать негативных последствий для организаторов трансляции.

Для достижения цели работы, были поставлены следующие задачи:

– исследование предметной области для полного понимания существующей проблемы, возможных негативных последствий, а также исследование существующих методов её решения;

– изучение и сравнение методов, которые могут быть использованы для решения обозначенной проблемы, включая современные методы машинного обучения в области обработки речи и подготовка к реализации алгоритма для решения проблемы;

– разработка системы для решения обозначенной проблемы, включая разработку алгоритма и непосредственно разработку приложения, позволяющего применить данный алгоритм на практике.

1. Теоретическая часть

1.1. Описание предметной области

В современном русском языке использование нецензурной лексики в большинстве случаев считается неприемлемым и оскорбительным. Однако, несмотря на это, в некоторых сферах и контекстах употребление нецензурных слов может быть распространено. Например, нецензурная лексика довольно часто используется в разговорной речи, а также в различных художественных произведениях для выражения эмоций и настроения [1]. Использование нецензурной лексики распространено и в критических ситуациях, которые могут перерастать в конфликты и провоцировать правонарушения [2].

Отношение в обществе к нецензурным выражениям на сегодняшний день неоднозначно. По данным ВЦИОМ на 2019 год [3], нецензурную лексику хотя бы раз в неделю использует 37 % опрошенных. При этом, по сравнению с аналогичным опросом 2008 года, доля опрошенных, использующих нецензурную лексику, снизилась на 7 %. При этом, 18 % из них используют в своей речи нецензурные выражения ежедневно. Диаграмма с результатами исследования представлена на рисунке 1.1.

Рисунок 1.1 – Результат исследования об использовании нецензурной лексики

Таким, образом, нецензурная лексика прочно вошла в разговорную речь большого количества людей. Однако, употребление нецензурных выражений, особенно в общественных местах или на большую аудиторию, остаётся табуированным. Также, многие люди считают, что использование нецензурной лексики может говорить о низком уровне культуры и образования говорящего, а также об агрессивности и неуважении к окружающим [1].

В соответствии с законодательством Российской Федерации, употребление нецензурных выражений может быть квалифицировано как административное правонарушение. В частности, статья 20.1 Кодекса Российской Федерации об административных правонарушениях предусматривает ответственность за использование нецензурной лексики в общественных местах в отношении человека или группы лиц [4]. Такое поведение может быть квалифицировано как мелкое хулиганство и привести к наложению штрафа.

В 2021 году вступили в силу поправки к закону «Об информации», которые обязывают новостные сайты и социальные сети удалять публикации, содержащие нецензурные выражения, оскорбляющие человеческое достоинство и моральные принципы [5]. Для удаления несоответствующего контента администрации этих площадок даются 1 сутки.

Отдельно рассматривается тема о недопустимости употребления нецензурных выражений в средствах массовой информации. Согласно общепринятым правилам этики, а также исходя из установленных нормативно-правовых актов, средства массовой информации обязаны избегать использования в изготавливаемой продукции нецензурных выражений. Такая позиция подтверждается и социологическими исследованиями, проведёнными среди населения.

По данным исследования Фонда общественного мнения, 74 % опрошенных негативно относятся к использованию нецензурной лексики в СМИ. При этом, 84 % опрошенных одобряют штрафы за использование таких выражений в СМИ [2]. Диаграммы результатов данного исследования представлены на рисунке 1.2.

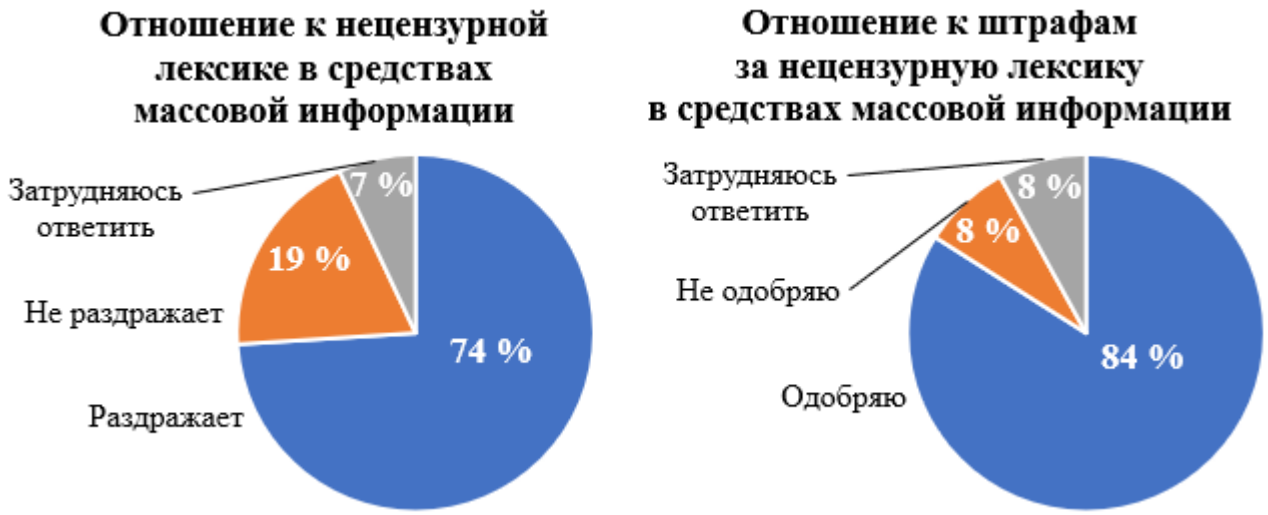

Рисунок 1.2 – Исследование отношения к нецензурной лексике в СМИ

Использование нецензурной лексики в средствах массовой информации, а в частности, в эфирах радиостанций и телевизионных каналов законодательно не допускается. В статье 4 закона РФ «О средствах массовой информации» установлены правила, которые запрещают распространение материалов, содержащих, в том числе, нецензурную лексику [6].

Статья 13.21 Кодекса РФ об административных правонарушениях определяет административную ответственность за изготовление или распространение продукции средства массовой информации, содержащей нецензурную лексику [4]. Нарушение приведённого выше нормативно-правового акта может привести к наложению крупного штрафа на телеканал или радиостанцию.

Выявление фактов употребления нецензурных выражений в продукции средств массовой информации является задачей Федеральной службы по

надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзор) и его территориальных органов.

После вступления в силу в 2013 году поправок в закон о СМИ, которые запретили использование нецензурных выражений, за 8 месяцев было вынесено 40 предупреждений о нарушении приведённого выше правила, о чём сообщается в публикации пресс-службы Роскомнадзора [7]. При наличии двух и более предупреждений в течение 12 месяцев ведомство может подать иск об аннулировании лицензии СМИ.

В действующем законодательстве отсутствует точный перечень слов и выражений, считающихся нецензурными. Однако, Роскомнадзор выпустил рекомендательный документ [8] по данному вопросу. В документе приведены допустимые способы упоминания нецензурных слов в официальных документах и в средствах массовой информации. Эти способы не должны допускать однозначного определения конкретных нецензурных слов в общем контексте. К примеру, замена одной буквы звёздочкой или многоточием не допускается, если при таком способе маскировки можно однозначно установить, какое нецензурное слово было скрыто таким образом. При этом, такая формулировка, как «слово на букву „б“» допускается.

Таким образом, документ определяет, что «к нецензурным словам и выражениям относятся четыре общеизвестных слова, начинающихся на „х“, „п“, „е“, „б“, а также образованные от них слова и выражения». Следовательно, к нецензурным следует относить как формы приведённых слов, так и все образованные от этих слов языковые единицы.

Одновременно с этим перечнем, для выявления, какие конкретно слова являются нецензурными, ведомство приводит ссылки на различные словари нецензурной лексики. В указанном документе, среди прочих, приводятся следующие словари:

– Большой толковый словарь русского языка [9];

– Словарь русской брани [10];

– Самый полный Словарь ненормативной лексики и фразеологических единиц [11].

В спорных ситуациях, когда нельзя однозначно определить, является ли конкретное слово нецензурным, необходимо проведение лингвистического исследования с привлечением независимых специалистов. В частности, такое исследование может быть проведено специалистами Института русского языка РАН, на основании заключения которого можно принимать решение о привлечении СМИ к ответственности.

Использование в российских СМИ нецензурной лексики на других языках не считается нарушением и не влечёт к ответственности редакцию. При этом, остальные грубые и неприличные слова и выражения, которые не относятся к нецензурным, могут быть транслированы в эфире, но только после 21:00. Нецензурные слова, которые заменены звуковым сигналом, могут быть транслированы, в том числе и раньше 21:00 [12].

В телевизионных и радиовещательных программах маскировка нецензурной лексики допускается с замещением нецензурных слов звуковым сигналом. Наиболее часто для этих целей применяется синусоидальный сигнал с частотой 1 кГц, однако практикуется использование и других менее раздражающих звуковых эффектов или замена нецензурных слов на тишину. При этом важно, чтобы нецензурные выражения невозможно было точно определить в контексте произносимой фразы.

Пример процесса замены нецензурных слов на звуковой сигнал показан на рисунке 1.3. Сначала в исходной аудиозаписи необходимо найти нецензурные слова, они выделены на рисунке. Далее найденные слова в аудиозаписи полностью заглушаются. После этого на тех интервалах, где

находились нецензурные слова, размещается звуковой сигнал. В примере, приведённом на рисунке, в аудиозапись добавляется наиболее часто используемый синусоидальный сигнал частотой 1 кГц. При этом амплитуда сигнала не должна быть максимальной, так как основная аудиозапись большую часть времени звучит намного тише. Амплитуда выбирается таким образом, чтобы соответствовать средней громкости исходной аудиозаписи.

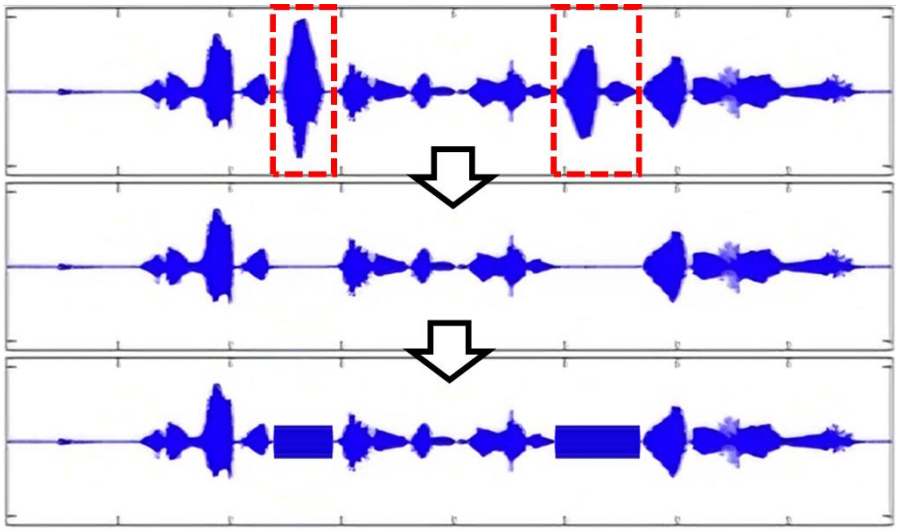

Рисунок 1.3 – Маскировка нецензурных слов в аудиосигнале

В связи с приведёнными выше сведениями, существует необходимость определения нецензурной лексики в звуковых данных. В сфере эфирного вещания потребность выявления нецензурных выражений возникает как в производстве прямых трансляций, так и при предэфирной подготовке материала для того, чтобы он соответствовал указанным ранее нормативно-правовым актам.

Также необходимость выявления нецензурных выражений может возникать и в других сферах деятельности. К примеру, в социальной сфере анализ речи и выявление нецензурных выражений может способствовать раннему выявлению конфликтов и принятию превентивных мер по их недопущению. В этом случае работа со звуком в ручном режиме также не является целесообразной. К примеру, для мониторинга звукового сигнала с

большого количества камер видеонаблюдения с целью выявления потенциально конфликтных ситуаций, требуется такое же количество сотрудников, что не оправдывает финансовые затраты.

1.2. Обзор существующих решений

Процесс обнаружения нецензурной лексики в настоящее время происходит в ручном режиме. Развитие цифрового эфирного телевидения, а также электронных СМИ требует ручной обработки большого количества материала для подготовки к выходу в эфир. Этот процесс связан с большими трудозатратами и требует большого количества ручной монотонной работы.

При работе с прямыми трансляциями такой подход особенно сложен. Для выявления и своевременного цензурирования нежелательных слов и выражений требуется повышенная концентрация внимания и скорость реакции инженера, ответственного за проведение трансляции. Для того, чтобы инженер смог вовремя среагировать и предпринять необходимые меры по фильтрации контента, необходима задержка трансляции до нескольких десятков секунд, что может являться критичным в некоторых случаях.

1.2.1. Аппаратные средства

Для удаления нецензурной лексики из аудиопотока может использоваться специальное оборудование, такое как микшерные пульты с кнопкой отключения или замены сигнала, при нажатии которой вместо нецензурных выражений в эфир будет подана тишина либо звуковой сигнал. На микшерных пультах, не оборудованных такой кнопкой, может применяться снижение уровня громкости входного сигнала с микрофона до нуля.

Также для этих целей может использоваться специальное устройство, называемое «*dump box*», которое позволяет удалить из эфира нежелательный звуковой контент. Одним из производителей такого оборудования является американская компания *Eventide*. Для возможности удаления фрагмента аудиосигнала, устройством создаётся буфер трансляции длительностью в несколько секунд. При нажатии кнопки буфер сбрасывается, а задержка обнуляется. Для того, чтобы создать новый буфер, устройство незаметно для слушателя увеличивает естественные паузы в речи, тем самым через некоторое время буфер снова будет иметь длительность в несколько секунд [13]. При таком способе слушатель может даже не заметить, что из трансляции был вырезан материал. Однако, недостатком такого метода является то, что длительность вырезаемого фрагмента всегда фиксирована, а для сброса буфера всё так же требуется участие человека.

1.2.2. Программные средства

С ростом цифровизации многие устройства в различных производственных отраслях стали заменяться аналогичными по назначению программными продуктами. В сфере эфирного телевизионного и радиовещания также наблюдается такой процесс.

К примеру, микшерные пульты, занимающие достаточно много пространства в студии, всё чаще встречаются в виде программных продуктов, что позволило оснастить их большим набором средств цифровой обработки звука и звуковых эффектов. Для решения задачи удаления нецензурной лексики может использоваться программный микшер по аналогии с физическим.

Также существуют и специализированные программные продукты, предназначенные конкретно для задачи удаления неприемлемого контента. К примеру, программный продукт MediaDelayEditor, разработанный компанией

SI-MEDIA, позволяет выполнять редактирование входящего сигнала во время его приёма, давая возможность редактору удалять нежелательный контент из видео или аудио. По заявлению компании-разработчика, приложение позволяет управлять воспроизведением в прямом эфире, однако, задержка между поступлением сигнала и его выходом в эфир, заявляемая разработчиком, составляет от 5 до 15 минут [14]. Трансляция с такой величиной задержки во многих случаях уже не может считаться прямым эфиром, но позволяет качественно отредактировать контент так, что зрители могут даже не заметить самого факта редактирования. Интерфейс программы поддерживает русский язык, однако существенным ограничением является то, что компания-разработчик зарегистрирована в Италии. В настоящее время это может представлять трудности в работе с данным программным обеспечением российскими вещательными компаниями в связи с введением санкций, ограничивающих использование иностранного программного обеспечения.

Также существуют программные продукты, позволяющие редактировать прямые эфиры с намного более низкой задержкой. К примеру, приложение CensorMX, разработанное сингапурской компанией Etere, позволяет быстро отключать или заменять звуковой сигнал при проведении прямых трансляций. При этом задержка трансляции является настраиваемой величиной и может быть изменена в зависимости от различных потребностей, при этом ограничения длительности задержки отсутствуют, ограничиваясь только объёмом памяти [15]. Программное обеспечение CensorMX позволяет настраивать рабочее пространство, предоставляя возможность наиболее удобного размещения компонентов под конкретные нужды. Имеется возможность управления несколькими серверами с одного клиента, что существенно снижает эксплуатационные затраты.

Компания-разработчик имеет российский филиал, что позволяет без дополнительных трудностей использовать данное программное обеспечение российскими вещательными компаниями.

Представленные программные продукты требуют мониторинга эфира человеком, что означает необходимость иметь задержку в трансляции для своевременного обнаружения нецензурной лексики и принятия мер по её скрытию из выходного сигнала. Также, при таком способе контроля за эфиром, на точность и своевременность выявления нецензурных выражений существенно влияет человеческий фактор.

С развитием интеллектуальных компьютерных систем, всё большее распространение получают системы распознавания речи. Такие системы могут применяться и для решения поставленной в данной работе задачи. Их преимуществом будет являться неподверженность человеческим ошибкам, возможность непрерывной работы, а также снижение задержки между поступлением сигнала и его отправкой в эфир по сравнению с выполнением такой задачи человеком. На данный момент существуют зарубежные решения для распознавания нецензурной лексики в реальном времени.

Одним из них является Intel Bleep — приложение, предназначенное для фильтрации нецензурных и оскорбительных выражений в аудиопотоке в реальном времени. Программа имеет множество настроек и может быть сконфигурирована на определение и последующее удаление различных видов оскорбительных выражений. Данное приложение предназначено для работы под управлением операционной системы Windows. В данный момент это приложение находится в стадии разработки и имеет поддержку только английского языка [16].

Существуют также и системы распознавания речи, которые позволяют выполнять полную транскрибацию аудио в текстовое представление, заменяя

при этом нецензурную лексику на другие символы. Одним из таких решений является Soniox — сервис, позволяющий обрабатывать несколько аудиопотоков параллельно, получая на выходе текстовое представление с отфильтрованной нецензурной лексикой, которая маскируется символом звёздочки [17]. Данный продукт может использоваться как облачный сервис, так и имеет возможность работы на собственной инфраструктуре. Поддерживается несколько языков, включая русский.

Существуют и разработанные в России сервисы распознавания речи, одним из которых является Yandex SpeechKit. Сервис позволяет распознавать речь в аудиопотоке в реальном времени, однако длительность одной сессии соединения ограничена 5 минутами, так как он предназначен в основном для разработки голосовых ассистентов [18]. В распознанном тексте возможно скрытие нецензурных слов с помощью символов-заменителей, это является одной из опций в параметрах нормализации. Также поддерживается расширение словаря, что позволяет внести в него нецензурные выражения для более точной работы системы. Данная система распознавания речи доступна как облачный сервис, так и поддерживает работу на локальной инфраструктуре.

Также существуют и другие сервисы, позволяющие обнаруживать нецензурную лексику в аудио. Например, Speech Analytics — сервис, разработанный ООО «Речевая аналитика», который поддерживает распознавание нецензурных выражений по заданному словарю, но предназначен для работы с записями телефонных звонков и работает только с предварительно сохранёнными файлами.

Представленные сервисы распознавания речи позволяют выявить нецензурную лексику, но для того, чтобы удалить её из выходного аудиосигнала, требуется разработка дополнительного программного обеспечения. Возможность работы с некоторыми системами только как с

облачными сервисами создаёт дополнительную задержку при передаче данных по сети, а также создаёт дополнительные точки отказа, которыми являются сетевое оборудование и линии связи.

Так как выходными данными сервисов распознавания речи является полная транскрипция текста, для обработки аудиопотока требуется задержка в несколько секунд. Такая величина задержки меньше, чем при обработке данных человеком, но также не оптимальна. Существуют алгоритмы, которые позволяют выполнить поставленную задачу с меньшей задержкой.

1.3. Выводы

В данном разделе были исследованы основные факторы употребления нецензурной лексики, приведены результаты исследований, отражающие отношение в обществе к нецензурной лексике. Отдельно рассмотрен вопрос о запрете использования нецензурной лексики в средствах массовой информации и приведены нормативно-правовые акты, регулирующие этот вопрос.

Рассмотрены программные и аппаратные средства удаления нецензурной лексики в прямых трансляциях. В результате анализа выявлено, что в настоящий момент большинство методов маскирования нецензурной лексики требует непосредственного участия человека. Следовательно, целесообразной является разработка алгоритма, позволяющего автоматизировать этот процесс, и позволяющего уменьшить задержку прямой трансляции по сравнению с ручным методом.

Такая система должна находить нецензурные слова в аудиопотоке, при этом иметь как можно более низкую задержку. В следующем разделе будут рассмотрены методы, позволяющие решить эту задачу, а также будет проведено их сравнение.

2. Проектирование системы

Использование методов машинного обучения для анализа аудиоданных в настоящее время быстро распространяется. Алгоритмы в этой области постоянно совершенствуются и находят применение во многих прикладных областях. Существуют модели, которые позволяют анализировать и классифицировать звуковые сигналы, распознавать речь, определять эмоциональное состояние и идентифицировать говорящего, а также решать многие другие задачи [19]. Для того, чтобы выполнять эти задачи, на вход модели необходимо подать набор признаков, описывающих конкретный объект, в данном случае фрагмент аудиосигнала.

2.1. Представление цифровых аудиоданных

При работе с цифровым звуком звуковой наиболее часто сигнал представляется в виде набора числовых значений, полученных с использованием импульсно-кодовой модуляции (PCM). Импульсно-кодовая модуляция заключается в преобразовании аналогового сигнала посредством трёх операций: дискретизации по времени, квантования по амплитуде и кодирования [20].

Дискретизация (sampling) — преобразование непрерывной функции в дискретную. Частота дискретизации — это количество замеров уровня сигнала (отсчётов) в секунду. В цифровой аудиозаписи наиболее часто применяется частота дискретизации 44,1 кГц.

Квантование (quantization) — разбиение диапазона отсчётных значений сигнала на конечное число уровней и округление этих значений до одного из двух ближайших к ним уровней. Например, при кодировании PCM с глубиной 16 бит возможно закодировать 65 536 различных значений.

Кодирование (coding) — получение числового эквивалента квантованной величины в виде комбинации цифр (кода).

В результате получается последовательность значений, которые описывают значение уровня амплитуды звуковой волны в каждый момент времени звучания записи. Такая последовательность является наиболее простым представлением звукового сигнала в виде признаков. Её преимуществом является сохранение полной информации о сигнале.

Однако, такое представление является крайне неэффективным по причине высокой размерности данных, что приводит к использованию значительных вычислительных ресурсов для обработки моделью. К примеру, для описания монофонического звукового сигнала длительностью в 1 секунду и с частотой дискретизации 44,1 кГц потребуется 44 100 признаков. При этом многие признаки будут избыточными или неинформативными, так как аудиосигнал может содержать шум, тишину и паузы между словами. При различной длительности сигналов количество признаков также будет различаться, что приведёт к невозможности модели обрабатывать сигналы с разной длительностью. Для решения этих проблем применяются различные алгоритмы извлечения признаков из аудиосигнала.

2.2. Извлечение признаков

Для снижения размерности входных данных и для того, чтобы избежать описанных проблем, применяются различные способы извлечения признаков из звукового сигнала [21]. Одним из подходов является спектральный анализ. Его суть заключается в том, что исходный сигнал со сколь угодно необходимой точностью возможно декомпозировать на составляющее. Другими словами, любой сложный сигнал может быть представлен в виде составляющих его синусоид с определенными частотами и амплитудами. Таким образом, спектр

сигнала — это представление сигнала уже в двух измерениях, что позволяет судить, как именно во времени распределяется энергия сигнала по частотам.

Существуют различные способы извлечения признаков из аудиосигнала, например частота пересечения нуля (Zero Crossing Rate, ZCR), кодирование с линейным предсказанием (Linear Predictive Coding, LPC), перцептивное линейное предсказание (Perceptual Linear Prediction, PLP) и другие.

Наиболее часто для задач обработки речи в качестве признаков используются мел-частотные кепстральные коэффициенты (MFCC), так как они наиболее точно соответствуют восприятию звука человеком, и для их вычисления существуют алгоритмы, обладающие достаточно высокой производительностью [22].

Метод основан на изменении человеческого голоса с критической пропускной способностью с помощью частотных треугольных фильтров. Они располагаются на линейных интервалах в области низких частот и на логарифмических интервалах в области высоких частот, что позволяет выделить из речевого сигнала фонетически важные признаки. Размеры окна рассчитываются с помощью мел-шкалы: первое окно очень узкое и показывает, сколько энергии содержится в районе нуля герц, по мере увеличения частоты размер окна становится шире, потому что с увеличением частоты звуки становятся менее различимыми для человеческого уха.

Первым этапом является применение кратковременного преобразования Фурье (Short-Time Fourier Transform, STFT). Аудиосигнал разбивается на короткие интервалы, и для каждого из них вычисляется спектральное представление с помощью преобразования Фурье, что позволяет получить информацию о спектральной составляющей звука.

После этого к полученным результатам применяются мел-фильтры. При этом спектральная информация преобразуется из частотной шкалы в мел-

шкалу, которая основана на восприятии звуков человеком и лучше соответствует восприятию различных частот. Существуют разные варианты формул для перехода между частотой (Гц) и высотой звука в мелах. Наиболее распространённый вариант рассчитывается по формуле 2.1:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right), \quad (2.1)$$

где f — исходная частота.

На следующем этапе происходит вычисление косинусного преобразования Фурье (DCT) для получения коэффициентов, называемых MFCC. DCT сжимает спектральную информацию и выделяет наиболее значимые частотные компоненты [23].

На рисунке 2.1 сверху показано представление амплитуды исходного звукового сигнала, ниже показана спектрограмма этого сигнала, а затем — спектрограмма его мел-частотных кепстральных коэффициентов.

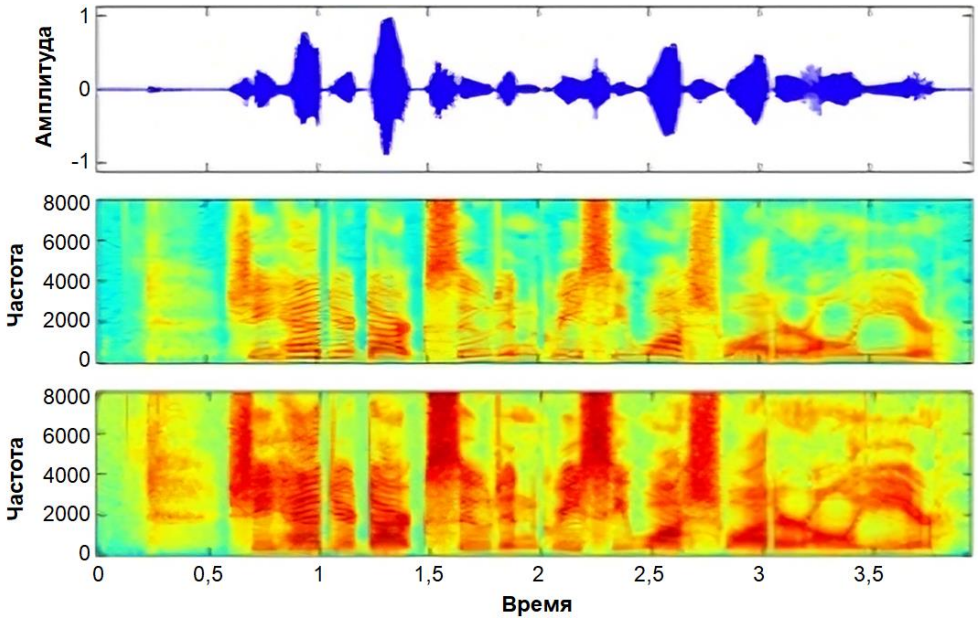

Рисунок 2.1 – Визуализация различных признаковых описаний сигнала

После извлечения признаков, они могут быть поданы на вход какой-либо модели для последующей обработки. Для решения задачи, поставленной в данной работе, могут быть применены два алгоритма: автоматическое

распознавание речи (ASR — automatic speech recognition), и распознавание ключевых слов (KWS — keyword spotting), которые будут рассмотрены далее.

2.3. Автоматическое распознавание речи

Автоматическое распознавание речи — это процесс преобразования звукового сигнала с произносимыми словами, фразами и предложениями в текстовый вид [24].

С развитием технологий машинного обучения, появляется всё больше возможностей для создания качественных и эффективных систем распознавания речи. В этой сфере широко применяются нейронные сети и глубокое обучение. Это позволяет выполнять обработку звукового сигнала и получать в результате наиболее вероятное текстовое представление записанных слов, фраз и предложений. Для обучения современных моделей распознавания речи используются большие наборы данных, которые включают в себя аудиозаписи и соответствующую им транскрипцию.

Структуру модели распознавания речи условно можно представить как последовательность из трёх логических блоков или подмоделей. К ним относятся акустическая модель, лексическая модель и языковая модель. Соответствующая схема представлена на рисунке 6.

Акустическая модель применяется с целью оценки схожести фрагментов входного сигнала со звуками (фонемами) в лингвистическом смысле — наименьшими значимыми единицами в языке. В основе акустической модели могут лежать скрытые марковские модели (HMM), глубокие нейронные сети (DNN), а также другие технологии [25]. Результатом обработки данных акустической моделью является распределение вероятностей различных фонем для конкретного фрагмента входных данных.

Далее полученные результаты обрабатываются лексической моделью. Лексическая модель отвечает за нахождение связей между фонемами и отдельными словами и основывается на словаре языка, который содержит список слов и их транскрипций, состоящих из последовательностей фонем. Лексическая модель может быть представлена в виде статистических моделей, таких как скрытые марковские модели, или в виде графа, который описывает возможные последовательности слов. Результатом работы лексической модели является список слов с соответствующими им вероятностями.

Следующим этапом распознавания речи является языковая модель. Языковая модель определяет, каким образом слова в предложении могут быть связаны между собой и какой порядок слов наиболее вероятен. Языковая модель может учитывать грамматику языка, предыдущие слова в предложении или дополнительную информацию о контексте. В автоматическом распознавания речи для этого чаще всего используются статистические методы. Языковые модели также могут быть основаны на статистических методах, таких как n-граммы или с применением нейронных сетей.

Комбинирование описанных моделей позволяет системе распознавания речи принимать во внимание акустическую информацию, словарь языка и контекст для качественной и точной интерпретации речи. Такая схема не является единственной, к примеру, на практике лексическая и языковая модели могут быть объединены вместе для улучшения производительности.

Существуют точные и качественные модели для распознавания речи, обученные с использованием мощных вычислительных устройств на больших наборах данных. Однако, их применение в задачах распознавания слов, не встречающихся в обучающих данных, затруднено. Такие ситуации могут возникать при распознавании речи, содержащей технические термины и слова, которые используются в узкоспециализированных профессиональных областях.

Такая проблема также присутствует и в задаче распознавания нецензурной лексики, так как в обучающих наборах данных обычно используются тексты, в которых нецензурная лексика встречается довольно редко.

Чтобы решить эту проблему, нет необходимости заново обучать акустическую модель. Для конкретного языка она уже содержит все звуки, необходимые для формирования новых слов, даже не входящих в обучающий набор данных. Вместо этого можно модифицировать только словарь лексической модели и языковую модель. Такой процесс называется адаптацией модели [26].

Для оценки качества моделей автоматического распознавания речи могут быть использованы различные метрики. Наиболее распространённый показатель, используемый для оценки моделей — частота ошибок в словах (word error rate, WER) [27], который может быть рассчитан по формуле 2.2:

$$WER = \frac{S + D + I}{N}, \quad (2.2)$$

где S — замены, то есть случаи, когда найденное алгоритмом слово не соответствует тому, которое было произнесено на самом деле,

D — удаления — слова, которые присутствовали в аудиозаписи, но были пропущены при распознавании,

I — вставки — слова, которых не было в исходной аудиозаписи, но которые были включены в выходные данные,

N — общее число произнесённых слов.

Метрика WER наиболее часто используется при оценке моделей автоматического распознавания речи, так как достаточно просто вычисляется и легко интерпретируется. Однако, эта метрика не учитывает, насколько ошибки искажают смысл оригинального текста. К примеру, если фактическое и

предсказанное слово отличаются только окончанием, при расчёте этой метрики это равнозначно тому, что эти два слова полностью различны [28].

Для того, чтобы учитывать это, можно использовать другую метрику — частоту ошибок в символах (char error rate, CER). Метрика CER рассчитывается по той же формуле, что и WER, но вместо слов используются символы. При этом пробелы и знаки препинания (если выходные данные были нормализованы) не учитываются. Эта метрика основана на расстоянии Левенштейна — строковой метрике, позволяющей измерить расстояние между двумя словами на основании того, сколько букв необходимо изменить в первом, чтобы получить второе. Также можно сказать, что WER использует расстояние Левенштейна на уровне слов, а CER — на уровне символов. Использование метрики CER позволяет более глубоко понять, какие изменения присутствуют в результате работы модели в сравнении с исходным текстом.

В общем случае, применение технологий автоматического распознавания речи может решить задачу обнаружения нецензурной лексики, поставленную в данной работе. Однако, задержка между передачей данных модели и получением результата может достигать нескольких секунд, что в некоторых случаях, включающих проведение прямых трансляций, может быть слишком долгим. Для работы в реальном времени предпочтительнее использовать алгоритмы обнаружения ключевых слов, которые будут рассмотрены далее.

2.4. Обнаружение ключевых слов

Обнаружение ключевых слов — это подраздел автоматического распознавания речи, используемый в задачах поиска в непрерывной речи конкретных слов. В отличие от общего распознавания речи, где система производит распознавание всей речь и преобразование в текст, распознавание

ключевых слов фокусируется на определенных заданных словах или фразах, которые требуются для конкретной задачи [29].

Алгоритмы обнаружения ключевых слов используется в различных областях, например, для активации голосовых ассистентов, другими словами, их «пробуждения», когда произносится их имя. Также, алгоритмы обнаружения ключевых слов могут применяться для анализа записей, телефонных звонков на предмет угроз, опасных ситуаций и для выявления темы разговора и настроения говорящего.

В отличие от автоматического распознавания речи, где выходными данными является текст, модель обнаружения ключевых слов является классификатором, который определяет, содержит ли текущий фрагмент ключевое слово или фразу. На основе выхода классификатора система определяет, содержится ли ключевое слово или фраза в аудиосигнале. Это может быть выражено в виде бинарного результата (наличие/отсутствие ключевого слова) или вероятностями для каждого из необходимых для распознавания ключевых слов.

Обнаружение ключевых слов является первой выявленной проблемой в контексте распознавания речи. Одним из первых исследований в этой области было распознавание ключевых слов с помощью кодирования с линейным предсказанием, опубликованное в 1977 году [30]. Данный метод начинается с предположения, что речевой сигнал воспроизводится звуком с определённым тоном с последующим добавлением различных акустических фильтров. Такая модель называется «источник-фильтр», при этом источником являются голосовые связки, а фильтрами — компоненты речевого тракта. Источник характеризуется интенсивностью (громкостью) и частотой (высотой). Речевой тракт образует последовательность, для которой характерны определённые резонансы, вызывающие форманты.

Алгоритм анализирует речевой сигнал, оценивая форманты, удаляя их влияние из речевого сигнала и оценивая интенсивность и частоту оставшегося тона. Процесс удаления формант называется обратной фильтрацией, а оставшийся сигнал после вычитания отфильтрованного смоделированного сигнала называется остатком. Комбинация выявленных фильтров, а также характеристики источника определяют звуки и слова.

Ещё одним из ранних методов обнаружения ключевых слов является сравнение с использованием динамической трансформации временной шкалы (DTW), описанный в 1970-х годах [31]. Так как речь различается по скорости, длительности и темпу, возникают нелинейные колебания речевого рисунка по оси времени, которые необходимо устранить. Если производить обычное евклидово сопоставление двух записей, такой способ будет неэффективным.

Данный метод позволяет выравнивать и сопоставлять две временные последовательности, которые имеют разную скорость или длительность. В задаче обнаружения ключевых слов данный алгоритм может быть использован для сравнения и сопоставления речевых аудиозаписей, таких как эталонная запись слова и запись, в которой необходимо выявить наличие этого слова.

Метод заключается в том, что для выравнивания двух временных рядов, в данном случае аудиозаписей, алгоритм производит «деформацию» оси времени одной или обеих последовательностей. На рисунке 7 показано сравнение стандартного евклидова сопоставления и сравнения с использованием динамической трансформации временной шкалы.

Наилучшее выравнивание вычисляется путём составления матрицы расстояний от каждого элемента в первой последовательности до каждого элемента второй последовательности. Затем вычисляется путь с минимальной стоимостью — оптимальный путь трансформации временной шкалы. После этого выполняется сама трансформация временной шкалы, происходит

сопоставление полученных сигналов и делается вывод об их сходстве или различии.

Алгоритмы кодирования с линейным предсказанием и динамической трансформации временной шкалы положили начало развитию систем обнаружения ключевых слов. С развитием компьютерных технологий, а в частности машинного обучения и искусственных нейронных сетей, а также с увеличением вычислительной мощности, для решения таких задач всё чаще стали применяться и эти методы.

Исследование нейронных сетей началось в 1940-х годах с формирования понятия нейрона и способов смоделировать поведение клеток мозга с помощью вычислительных средств [32]. Искусственный нейрон имеет намного следующую структуру: у него есть несколько входов, на которых он принимает различные сигналы, преобразует их и передает другим нейронам. Первая нейронная сеть, названная перцептроном, была представлена и реализована в конце 1950-х годов и могла быть использована для решения задач классификации [33].

Исследования в области нейронных сетей значительно расширились в 1980-е годы, когда были проанализированы нейронные сети с обратными связями и было предложено использовать эти разработки для решения задач оптимизации. В эти годы был существенно развит алгоритм обратного распространения ошибки, который используется при обновлении весов многослойного перцептрона. Это итеративный градиентный алгоритм, который используется с целью минимизации ошибки работы многослойного перцептрона и получения желаемого выхода [34].

Искусственные нейронные сети имеют несколько разновидностей, при этом самые простые — однослойные и многослойные нейронные сети, состоящие из полносвязных слоёв. Полносвязный слой — слой нейронов, в

котором каждый нейрон соединён со всеми нейронами на предыдущем уровне, причём каждая связь имеет свой вес.

Однослойная нейронная сеть — сеть, в которой сигналы от входного слоя сразу подаются на выходной слой, который и преобразует сигнал и сразу же выдаёт ответ.

Многослойная нейронная сеть — нейронная сеть, состоящая из входного, выходного и одного или нескольких скрытых слоёв. Многослойные нейронные сети имеют намного больше возможностей для применения в различных задачах, чем однослойные, так как могут моделировать функции практически любой степени сложности [35].

С развитием искусственных нейронных сетей, кроме простых однослойных и многослойных нейронных сетей появились также свёрточные и рекуррентные нейронные сети.

Свёрточные нейронные сети первоначально применялись для обработки изображений. Данный тип нейронных сетей получил название по выполняемой операции свёртки, суть которой в том, что каждый ядро свёртки поэлементно проходит по тензору, выполняя операцию умножения, а результат суммируется и записывается в аналогичную позицию выходного тензора. Признаки на выходе являются взвешенными суммами признаков на входе, при этом весами являются значения ядра свёртки [36].

Основными видами слоёв в свёрточной нейронной сети являются свёрточные (convolutional), пулинговые (pooling) и полносвязные (fully-connected) слои.

Свёрточный слой нейронной сети выполняет описанную операцию свёртки над входными данными, при этом значения ядра свёртки являются параметрами для обучения. То есть для различных нейронов выходного слоя

используется одна и та же матрица весов, которой является ядро свёртки. Это определяет особенность свёрточного слоя — относительно небольшое количество параметров, которые необходимо обучить, которое существенно меньше, чем при использовании полносвязных слоёв применительно к такой же задаче. Ещё одним обучаемым параметром является константный сдвиг (bias).

Слой пулинга (подвыборки, субдискретизации) представляет собой нелинейное полученного тензора признаков, при этом группа элементов уплотняется до одного, проходя нелинейное преобразование. Для получения выходного значения могут использоваться различные функции. Наиболее часто используются функции максимума и среднего значения.

Данный слой позволяет уменьшить размер выходного тензора, что позволяет избежать переобучения и получить более сжатое представление исходных данных для последующих слоёв. Такое уплотнение используется, так как необходимые закономерности, требующие изучения входных данных на более подробном уровне, были найдены на предыдущих и текущем слоях.

В свёрточных нейронных сетях обычно есть несколько свёрточных слоёв, и после каждого из них находится пулинговый слой. При этом входной тензор, изначально представляющий исходные данные, такие как изображение или спектрограмму в случае распознавания ключевых слов, постепенно преобразуется в более абстрактные карты признаков. После прохождения нескольких слоёв свёртки и пулинга, остаётся большой набор каналов, хранящих небольшое число данных, которые интерпретируются как самые абстрактные понятия, выявленные из исходных данных и с которыми эффективно работают полносвязные слои.

После этого данные объединяются и передаются на обычную полносвязную нейронную сеть, которая также может состоять из нескольких

слоёв. При этом полносвязные слои работают с данными уже небольшой размерности, что позволяет увеличить эффективность этих слоёв.

Использование свёрточных нейронных сетей показывает хорошие результаты при решении задач автоматического распознавания речи и обнаружения ключевых слов, при этом имея достаточно низкую задержку [37].

Ещё одним типом нейронных сетей, которые могут использоваться для распознавания ключевых слов, являются рекуррентные нейронные сети. В таких сетях реализуется механизм обратной связи — сигнал с выходных нейронов или нейронов скрытого слоя частично передаётся обратно на входы нейронов входного слоя. Это означает, что выход какого-либо нейрона определяется не только его весами и входным сигналом, но еще и предыдущими выходами, так как они снова вернулись на входы [38].

За счёт наличия таких циклов, рекуррентные нейронные сети хорошо подходят для обработки последовательностей, к которым относится аудиосигнал. Речь представляет собой временную последовательность с сильными временными зависимостями, и рекуррентные нейронные сети позволяют эффективно работать с такими зависимостями [39].

Для обучения рекуррентных нейронных сетей наиболее часто используется алгоритм обратного распространения ошибки, но с небольшим изменением, так как градиент на каждом выходе зависит не только от расчетов текущего шага, но и от предыдущих временных шагов. Этот алгоритм называется алгоритмом обратного распространения ошибки сквозь время (Backpropagation Through Time, BPTT).

Частным случаем рекуррентных сетей являются двунаправленные сети. В таких сетях между слоями существуют связи как в направлении от входного слоя к выходному, так и в обратном. Такая сеть представляет собой две однонаправленные рекуррентные сети, одна из которых обрабатывает входную

последовательность в прямом порядке, а другая — в обратном, что позволяет эффективно работать с речевым сигналом.

Однако, двунаправленные рекуррентные нейронные сети не поддерживают работу в потоковом режиме, который предусматривает подачу на вход нейронной сети не всей последовательности, а итеративно по частям, что необходимо для возможности обработки потокового аудиосигнала в реальном времени.

Ещё одной разновидностью рекуррентных нейронных сетей является сеть с долговременной и кратковременной памятью (Long short term memory, LSTM). Таким способом удалось обойти проблему исчезновения или зашкаливания градиентов в процессе обучения методом обратного распространения ошибки, которые присутствовали в рекуррентных сетях. За счёт специальных вентилей «забывания» такая сеть может сохранять память о тысячах и даже миллионах временных интервалов в прошлом.

Начиная с 2007 года LSTM приобрела популярность и позволила достичь хороших результатов в задачах обработки речи, и в частности, в распознавании ключевых слов, показав существенное улучшение результатов по сравнению с предшествующими моделями [40].

Использование рекуррентных нейронных сетей показывает хорошие результаты при обработке речи и распознавании ключевых слов, однако, такие сети обычно имеют достаточно большую задержку при обработке данных, поэтому их использование в задачах, требующих как можно более быстрого ответа, является менее целесообразным, чем использование свёрточных нейронных сетей.

2.5. Обзор моделей

В настоящее время существует множество архитектур моделей для решения задачи обнаружения ключевых слов. Эти модели различаются по скорости, задержке, объёму требуемой оперативной памяти. Также модели различаются по типам устройств, на которых они могут быть запущены. Модели с низкими требованиями к ресурсам и памяти могут использоваться на встраиваемых системах и мобильных устройствах, а более производительные и требующие больше ресурсов находят применение в работе на мощных компьютерах и серверах. При этом такие модели имеют более высокую точность и низкую задержку.

Точность модели обнаружения ключевых слов является одним из главных параметров при её оценке. При этом на разных датасетах точность модели может существенно различаться. В разных датасетах может быть различное количество примеров, а также для качественного обучения может использоваться аугментация данных, к примеру, добавление шума.

Одним из наиболее часто используемых наборов данных для оценки моделей является Google Speech Commands [41]. За последние 5 лет на этот датасет ссылались более 250 научных публикаций. Он имеет несколько версий, наиболее часто используется версия 1 с распознаванием 12 команд (Google Speech Commands V1 12), в которой содержится 64 727 аудиозаписей, произнесённых 1881 разными людьми.

На рисунке 8 представлен график, показывающий возрастание точности моделей для обнаружения ключевых слов из датасета Google Speech Commands V1 12. Линия соединяет точки, представляющие модели, которые превзошли результат предыдущей наиболее точной модели.

В настоящий момент наиболее точными моделями на этом датасете являются TripletLoss-res15, BC-ResNet-8 и Wav2KWS [42].

Модель TripletLoss-res15 основывается на архитектуре остаточной нейронной сети ResNet. Это архитектура свёрточной нейронной сети (CNN), которая может работать с сотнями или тысячами свёрточных слоёв, решая проблему исчезающего градиента путём пропуска некоторых слоёв на начальных этапах обучения. В этой модели используется комбинация метрик Triplet Loss и вариант метода k-ближайших соседей вместо кросс-энтропийной функции потерь. Метрика Triplet Loss, которая нашла широкое применение в области компьютерного зрения, а в частности, для повторной идентификации личности. Однако, эта метрика показала хорошие результаты и в области обнаружения ключевых слов [43]. Данная модель имеет точность 98,56 %, что является на данный момент лучшим результатом при распознавании ключевых слов датасета Google Speech Commands V1 12.

Модель BC-ResNet-8 также основана на архитектуре остаточной нейронной сети ResNet, но с некоторыми модификациями, называемой BC-ResNet, что означает «транслируемая остаточная нейронная сеть», что позволяет адаптировать модель для использования на устройствах с различными ресурсами. Данный метод конфигурирует большинство остаточных функций как одномерную временную свёртку, но в то же время допуская двумерную свёртку вместе с использованием транслируемого остаточного соединения, которое расширяет временной вывод до частотно-временного измерения. Это позволяет представлять признаки аудиосигнала более эффективно и с гораздо меньшими вычислительными затратами, чем обычные свёрточные нейронные сети. Такая архитектура позволяет достичь высокой точности при небольшом размере модели и низкой вычислительной нагрузке, что может быть необходимо для работы на встраиваемых и

мобильных устройствах [44]. Данная модель имеет точность 98 % при распознавании ключевых слов из датасета Google Speech Commands V1 12.

Модель Wav2KWS использует архитектуру Wav2vec 2.0, которая преобразует звуковой сигнал в векторное представление. Данная модель была разработана для того, чтобы решить проблему отсутствия больших датасетов для обнаружения ключевых слов в конкретных областях. Создание или получение больших обучающих наборов данных, которые необходимы для надежного обнаружения ключевых слов, остаётся сложной задачей. Для решения этой проблемы применяется метод трансферного обучения (Transfer Learning), который позволяет применять модели, обученные на решение одной задачи для решения другой. При этом, модель может изначально обучаться на больших датасетах, а использоваться для тех задач, для которых таких объёмов данных для обучения нет. Данная модель была протестирована на нескольких языках и показала достаточно хорошую точность при распознавании ключевых слов [45]. Данная модель имеет точность 97,9 % при распознавании ключевых слов из датасета Google Speech Commands V1 12.

Рассмотренные модели показывают хорошую точность, близкую к 100 %, на датасете Google Speech Commands V1 12. Однако, при решении задачи, поставленной в данной работе точность будет отличаться, так как для обучения требуется специализированный датасет. Также, для того чтобы была возможность обрабатывать аудио в реальном времени, модель должна поддерживать принятие входных данных не полностью, а по фрагментам, и постепенно классифицировать их. Не все модели можно запустить в таком режиме. Модели, которым необходим доступ ко всей входной последовательности, не могут работать в потоковом режиме. К ним относятся, например, двунаправленные рекуррентные нейронные сети. Те модели, которые поддерживают такой режим работы, называются потоковыми.

Во время обучения модели обычно используется непотоковый режим. Для того, чтобы конвертировать модель в потоковый режим, может потребоваться ручное переписывание модели. Эта задача может быть решена с помощью библиотеки на основе Keras, которая позволяет автоматически преобразовать модель из непотокового режима в потоковый [46]. С помощью этой библиотеки уже была выполнена конвертация в потоковый режим наиболее востребованных моделей распознавания ключевых слов.

Для решения задачи, поставленной в данной работе, была выбрана потоковая модель на основе архитектуры MatchboxNet [47]. В настоящее время она занимает шестое место по точности с результатом 97,48 % при распознавании ключевых слов из датасета Google Speech Commands V1 12. Данная модель использует значительно меньше параметров, чем большинство современных моделей. Следовательно, процесс обучения требует намного меньше времени, модель имеет более высокую пропускную способность, и не требует значительных вычислительных ресурсов и памяти. Это позволяет без больших затрат обрабатывать несколько аудиопотоков параллельно. Данная модель устойчива к шуму, а также имеет достаточно большое количество материалов в открытом доступе, следовательно, проста в обучении.

2.6. Описание выбранной модели

Модель MatchboxNet — сквозная (end-to-end) остаточная нейронная сеть для обработки речевых команд, разработанная в 2020 году. Данная модель основана на архитектуре свёрточной нейронной сети QuartzNet, которая широко используется в задачах распознавания речи [48]. Модель состоит из блоков одномерной свёртки с разделением по временному каналу, что позволяет уменьшить размер модели, пакетной нормализации, функции

активации ReLU и регуляризации методом Dropout. Схема модели MatchboxNet BxRxC показана на рисунке 2.2.

Параметр B задаёт количество остаточных блоков модели, каждый из которых содержит R подблоков. Все подблоки в каждом блоке содержат C выходных каналов.

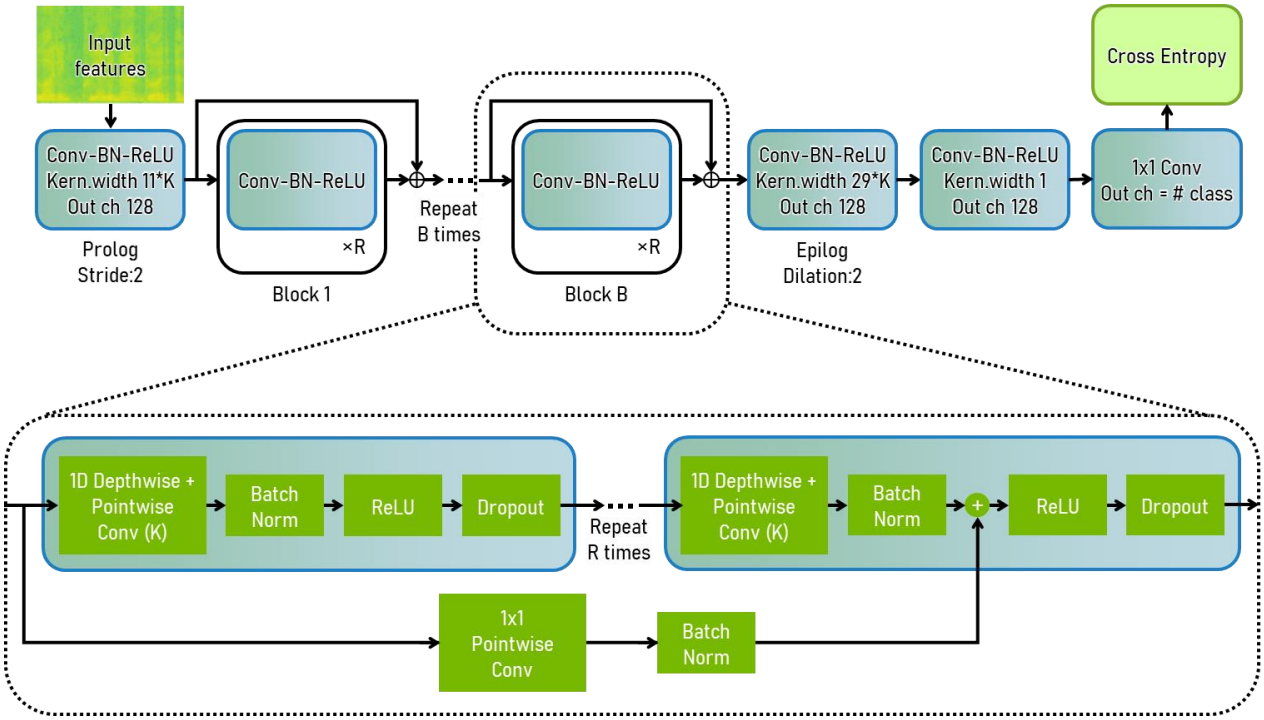

Рисунок 2.2 – Схема модели MatchboxNet

Базовый подблок состоит из одномерной свёртки с разделением по временному каналу, точечных свёрток 1×1, пакетной нормализации, функции активации ReLU и Dropout.

Модели, использующие свёртки с разделением по временному каналу, имеют гораздо меньшее число параметров, чем модели с обыкновенными свёртками, а также менее склонны к переобучению. Каждый модуль свертки с разделением по временному каналу состоит из двух частей: свёртки по глубине и точечной свёртки.

Свёртки по глубине применяют один фильтр на входной канал (входная глубина). Одномерные свертки с разделением по временному каналу могут

быть разделены на одномерный свёрточный слой по глубине с размером ядра K, который работает на каждом канале отдельно, но в течение K временных кадров, и точечный свёрточный слой, который работает в каждом временном кадре независимо, но во всех каналах.

Точечные свёртки 1×1 — свёртки, используемые для создания линейной комбинации выходных данных свёртки по глубине. Пакетная нормализация и активация ReLU применяются к выходным данным обоих слоёв.

Пакетная нормализация — метод нормализации, который позволяет повысить производительность и стабилизировать работу нейронной сети. Для того, чтобы выбрать параметры нормализации, используется не полный датасет, а каждый пакет отдельно [49]. Такой способ нормализации позволяет каждому слою меньше зависеть от других и ускоряет сходимость модели. Результатом нормализации являются данные, имеющие нулевое математическое ожидание и единичную дисперсию.

Функция активации ReLU — одна из наиболее часто используемых функций активации в искусственных нейронных сетях [50]. Она возвращает значение аргумента, если он положительный, и 0 в противном случае. Математически её можно записать в виде формулы 2.3:

$$f(x) = \max(0, x) \tag{2.3}$$

Регуляризация методом исключения, или Dropout — один из методов предотвращения переобучения модели. Для этого используется исключение некоторой части нейронов с некоторой вероятностью на разных итерациях обучения нейронной сети. При этом, более обученные нейроны получают больший вес по сравнению с менее обученными. Такой подход позволяет значительно повысить скорость обучения, качество обучения на тренировочных данных и качество предсказаний на тестовых данных.

Кроме описанных составляющих, модель MatchboxNet также содержит четыре дополнительных подблока: один прологовый свёрточный слой перед первым остаточным блоком и три эпилоговых свёрточных подблока перед последним слоем Softmax.

Так как модель состоит из нескольких идентичных блоков, имеется возможность её масштабирования. Это можно сделать, увеличив глубину B×R или увеличив число каналов C. Масштабируемость позволяет использовать модель в различных целях и на различных устройствах.

Так как модель основана на слоях, поддерживающих потоковую передачу, она может быть конвертирована в потоковый режим. Существует два основных режима потоковой передачи: с внешним и с внутренним состоянием.

Потоковый режим с внешним состоянием предполагает, что все буферы, которые необходимы для потоковой передачи, добавляются как дополнительные входы модели. Поэтому управление этими буферами должно производиться явно.

Потоковый режим с внутренним состоянием основан на буферах, которые добавляются в качестве дополнительных весов модели. Такой способ упрощает разворачивание модели, так как нет необходимости управлять этими буферами явно [46].

2.7. Сбор датасета

Датасет — это набор данных, который содержит информацию, используемую в различных видах анализа и машинного обучения. Он предоставляет данные, на которых модель будет основывать свои предсказания и выявлять закономерности.

Использование качественных датасетов является одним из важных факторов, влияющих на качество моделей машинного обучения. Поэтому выбор правильного датасета и его предобработка — критически важные шаги в процессе обучении моделей.

Для обучения модели обнаружения ключевых слов необходим датасет, содержащий аудиозаписи требуемых слов и метки ключевых слов — их текстовую транскрипцию. Структура датасета для распознавания ключевых слов может быть различной в зависимости от конкретной задачи и используемого подхода.

Наиболее часто используется следующая структура: датасет содержит несколько директорий, названных по необходимым ключевым словам, в каждой из них содержатся файлы с аудиозаписями этих слов [41]. Файлы обычно сохраняются в несжатом формате, например wav, что позволяет ускорить обучение, так как не тратится время на распаковку, но такой способ хранения увеличивает размер датасета. Длительность файлов с аудиозаписями обычно не превышает одной секунды.

Кроме того, датасет должен содержать и данные, не относящиеся к необходимым ключевым словам для того, чтобы модель могла качественно предсказывать отсутствие ключевых слов, тем самым снижая количество ложных срабатываний. Такими данными обычно являются тишина, фоновый шум и другие слова.

Одной из сложностей при решении задачи обнаружения нецензурной лексики является отсутствие готовых специализированных датасетов на русском языке. Для решения этой проблемы было принято решение собрать новый датасет на основе нескольких существующих датасетов, находящихся в открытом доступе и используемых для обучения моделей автоматического распознавания речи.

Для обработки были выбраны следующие датасеты:

– SOVA RuDevicesAudiobooks — записи аудиокниг на русском языке, содержащие 298 часов записей объёмом более 30 ГБ [51];

– SOVA RuYoutube — транскрипции русскоязычных видеороликов, допускающих повторное использование в соответствии с лицензией Creative Commons, содержит более 17 тысяч часов записей объёмом более 1,8 ТБ и разбит на 37 архивов объёмом около 50 ГБ [51];

– SOVA RuDevices — живая речь на русском языке с аннотациями, полученными путём расшифровки голосовых сообщений, содержит 101 час записей объёмом 10,42 ГБ [51];

– Russian Open Speech To Text (STT/ASR) — 118 000 аудиозаписей от 66 человек, которые были собраны с помощью краудсорсинговой платформы, объёмом более 26 ГБ [52].

Для того, чтобы из указанных датасетов сформировать датасет нецензурных слов, были разработаны два скрипта, один из которых использован для обработки датасетов SOVA, а второй — для обработки датасета Russian Open Speech To Text, так как эти датасеты имеют различную структуру. Эти скрипты позволяют найти нецензурные слова и сохранить их в отдельные аудиофайлы, при этом группируя их по директориям.

Для обработки датасетов RuDevicesAudiobooks, RuYoutube и RuDevices может быть использован один скрипт, так как они имеют одинаковую структуру и представлены в виде zip-архива. Исходный код скрипта представлен в приложении А.

Так как объём исходных датасетов достаточно большой, а нецензурные слова встречаются не в каждом файле, загрузка и распаковка полностью всех этих данных была бы нецелесообразной. Вместо этого используется библиотека

remotezip, позволяющая получить доступ к файлам zip-архива без загрузки полного содержимого с удалённого веб-сервера.

Для того, чтобы получить датасет нецензурных слов, сначала считывается список файлов из архива. После этого в оперативную память последовательно загружаются все текстовые файлы, содержащие аннотации к аудиозаписям. Если в каком-либо слове из текста содержится нецензурная часть, то далее уже производится загрузка в оперативную память соответствующей аудиозаписи.

Так как аннотации содержат только текст без обозначений времени начала и окончания каждого слова, необходимо уточнить эту информацию для искомых нецензурных слов. Для того, чтобы установить точные границы нецензурного слова в аудиозаписи, используется библиотека автоматического распознавания речи Vosk, основанной на наборе инструментов Kaldi [53].

С помощью данной библиотеки происходит повторная аннотация аудиозаписи, при этом в выходных данных теперь присутствует время начала и окончания каждого слова, а также значение уверенности для этого слова. Если это значение больше 0,5, то из исходной аудиозаписи вырезается фрагмент, содержащий нецензурное слово и сохраняется в соответствующую директорию. Проверка значения уверенности позволяет избежать ложных срабатываний и попадания в датасет несоответствующих данных.

Для обработки датасета Russian Open Speech To Text использован модифицированный скрипт, так как он имеет отличающуюся структуру. Вместо аннотаций в текстовых файлах в нём присутствует csv-файл, содержащий имя файла с аудиозаписью и соответствующий текст. Для этого датасет был предварительно загружен и распакован. Для выявления аудиозаписей с нецензурной лексикой файл аннотаций считывался с помощью библиотеки

Pandas и далее обрабатывались соответствующие аудиозаписи. Скрипт для обработки датасета Russian Open Speech To Text представлен в приложении Б.

Наибольшее количество нецензурных слов было получено из датасета RuDevices, так как он содержит живую речь, несмотря на то что его объём не самый большой из обработанных датасетов. В результате обработки данными скриптами четырёх представленных датасетов, было получено свыше 11000 записей, сгруппированных в более чем 640 классов.

Для моделей распознавания ключевых слов такое количество классов слишком велико, поэтому они были проанализированы и обобщены в 10 классов, содержащих в основном корни нецензурных слов и наиболее часто встречающиеся формы слов. Полученные классы представлены в таблице 2.1.

Таблица 2.1 – Полученные классы датасета

Номер класса	Начальная буква	Количество букв	Форма слова
1	«е»	4	качественное прилагательное мужского рода единственного числа в краткой форме
2	«е»	5	глагол повелительного наклонения единственного числа, возвратный
3	«ё»	6	качественное прилагательное в начальной форме
4	«е»	5	глагол начальной формы
5	«е»	3	глагол настоящего времени первого лица единственного числа
6	«е»	4	глагол прошедшего времени мужского рода единственного числа
7	«х»	3	существительное начальной формы
8	«п»	5	существительное начальной формы
9	«п»	5	существительное единственного числа в дательном падеже
10	«б»	3	междометие

Преобразование данных к этим 10 классам было выполнено с помощью повторного прохода скрипта, приведённого в приложении 2, но уже отдельно для каждого из 10 требуемых классов.

На практике модель будет определять не только 10 приведённых слов, а также другие нецензурные слова, составной частью которых они являются, что значительно расширяет фактическое количество определяемых классов нецензурных слов, при этом позволяя эффективно использовать алгоритм обнаружения ключевых слов.

Для устранения дисбаланса классов, который может привести к ухудшению качества классификации [54], для классов, в которых количество примеров превышало 200, некоторые примеры были удалены. Для тех классов, для которых не хватало данных, примеры были дополнительно записаны или получены из открытых источников. Все примеры, которые превышали длительность в 1 секунду, были скорректированы для соответствия этому ограничению длительности.

Для того, чтобы модель смогла определять отсутствие ключевых слов, дополнительно был добавлен класс unknown, содержащий 20 000 примеров, на основе данных из датасета RuDevices. Для этого был использован скрипт, представленный в приложении 2, но без проверки на вхождение слова в заранее составленный список.

В результате получен датасет, содержащий 10 классов с 200 примерами в каждом. Все примеры были дополнительно проверены вручную и при необходимости скорректированы для избежания попадания в примеры посторонних частей слов и других слов.

2.8. Выводы

В данном разделе были рассмотрены и проанализированы основные алгоритмы, позволяющие решить задачу обнаружения нецензурной лексики, и имеющие возможность работы в реальном времени, в том числе с использованием машинного обучения. Представлена историческая справка о развитии алгоритмов обработки речи и о современных тенденциях в этой области. Приведены сведения об обработке цифрового звука и рассмотрены способы представления аудиосигнала в виде признаков, которые могут быть использованы как входные данные для моделей машинного обучения.

В результате сравнения алгоритмов наиболее подходящим для решения поставленной задачи был выбран алгоритм обнаружения ключевых слов. Среди наиболее востребованных моделей была выбрана нейронная сеть MatchboxNet, так как она поддерживает работу в потоковом режиме, легко масштабируется и имеет высокую пропускную способность и не требовательна к ресурсам.

Также, в данном разделе была решена проблема отсутствия специализированных датасетов нецензурной лексики на русском языке, что позволит обучить выбранную модель для решения поставленной задачи.

В следующем разделе будет проведено обучение и тестирование выбранной модели, а также разработка программы, позволяющей использовать модель для обработки реальных данных.

3. Разработка и тестирование

3.1. Обучение модели

Для обучения и последующего тестирования модели датасет был разделён на три выборки — обучающую, валидационную и тестовую в соответствии с таблицей 3.1.

Таблица 3.1 – Размеры выборок

Выборка	Количество примеров	
	Классы ключевых слов	Класс unknown
Обучающая	1600 (160 × 10 классов)	16000
Валидационная	200 (20 × 10 классов)	2000
Тестовая	200 (20 × 10 классов)	2000

Обучающая выборка — это набор примеров, использованных во время обучающего процесса и использованных для настройки параметров модели, таких как веса и смещения.

Валидационная выборка — это набор примеров, которые могут использоваться для настройки гиперпараметров и контроля процесса обучения. Модель не обучается непосредственно на этих данных, но они косвенно влияют на модель, поэтому отделение валидационного набора от тестового положительно влияет на качество обучения.

Тестовая выборка — это набор примеров, который не используется непосредственно в процессе обучения и настройки гиперпараметров модели и применяется для объективной оценки окончательной модели.

Для обучения используется алгоритм оптимизации Adam (adaptive moment estimation — адаптивная оценка момента). Этот алгоритм оптимизации основан на стохастическом градиентном спуске, который обновляет веса

нейронов в нейронной сети. Оптимизатор Adam обновляет скорость обучения для каждого веса индивидуально. Этот оптимизатор является наиболее часто используемым, так как имеет ряд преимуществ: простота настройки, низкие требования к памяти и высокая скорость работы [55].

Так как обучение модели сразу на всей обучающей выборке, состоящей из 17600 примеров, требует больших вычислительных мощностей, обучение производится относительно небольшими пакетами (batch). Размер пакета подобран так, что при обучении соблюдается баланс нагрузки на центральный процессор и на диск и составляет 440 примеров на каждый пакет.

Количество шагов обучения (steps) было выбрано равным 6000. Каждый шаг — это обучение модели с использованием одного пакета и одно обновление градиента. Исходя из размеров пакета и общего количества примеров можно рассчитать количество итераций (iterations) — число шагов (пакетов), за которое будет произведено обучение на всей обучающей выборке один раз. В данном случае количество итераций составляет 40 шагов (пакетов).

Число эпох (epoch) — это количество полных проходов обучающей выборки через нейронную сеть в процессе обучения. Оно может быть вычислено как частное от деления общего количества шагов обучения на количество итераций и в данном случае составляет 150 эпох.

При обучении используется переменный темп обучения (variable learning rate). Темп обучения — один из параметров алгоритма оптимизации, который настраивает размер шага градиентного спуска на каждой итерации при движении к минимальному значению функции потерь.

Переменный темп обучения задаётся расписанием темпа обучения (learning rate schedule) и может быть представлен как линейное, ступенчатое или экспоненциальное затухание. В данном случае выбрано ступенчатое затухание в соответствии с таблицей 3.2.

Таблица 3.2 – Расписание темпа обучения

Количество итераций	2000	2000	1000	1000
Темп обучения	0,001	0,0005	0,0001	0,00002

При стандартном методе градиентного спуска скорость обучения поддерживается постоянной на протяжении всего обучения. Высокий темп обучения позволяет модели обучаться быстрее, но получая в результате не самый оптимальный набор весов. Меньший темп обучения может позволить модели получить более оптимальный или даже глобально оптимальный набор весов, но обучение может занять значительно больше времени.

Переменный темп обучения допускает большие изменения веса в начале процесса обучения и небольшие изменения или тонкую настройку ближе к концу процесса обучения. Это позволяет улучшить производительность обучения, при этом сохраняя стабильность обучения.

В процессе обучения оценка метрики accuracy модели на валидационной выборке производилась каждые 400 шагов. График изменения accuracy представлен на рисунке 3.1.

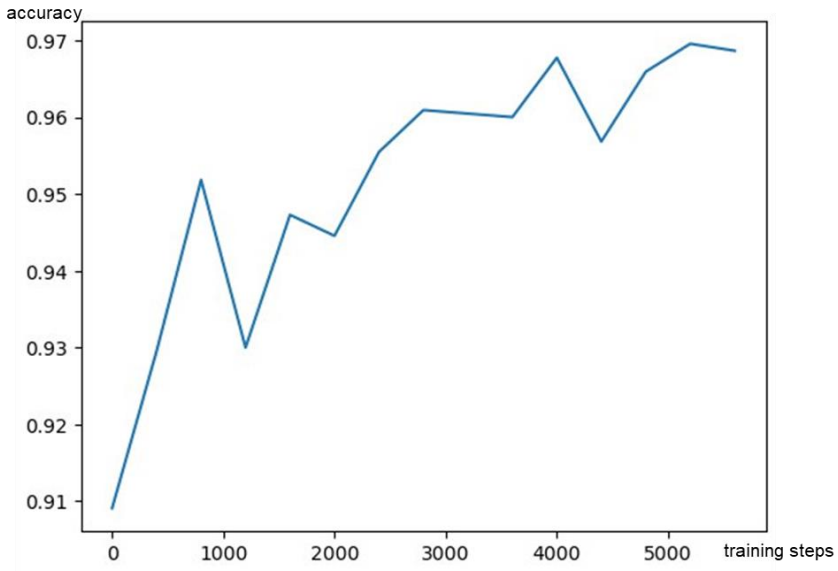

Рисунок 3.1 – График accuracy при обучении

В результате обучения модели наилучший показатель метрики accuracy на валидационной выборке составил 96,77 %.

3.2. Оценка качества модели

Оценка качества полученной модели — важный этап, который позволяет понять, насколько модель соответствует поставленной задаче и применимо ли данное решение на практике. Выбранная модель для решения задачи определения нецензурной лексики является классификатором, поэтому в данном подразделе будут рассчитаны основные метрики, используемые при оценке классификаторов. Модель обучена на распознавание 10 классов нецензурных слов и 1 класса с остальными словами, и для оценки её качества может быть использовано два подхода.

Первый из них — это оценка полученной модели как бинарного классификатора. Такой подход соответствует задаче, поставленной в данной работе, так как требуется только выявление наличия или отсутствия нецензурных слов без конкретизации по конкретным словам.

Второй подход — это оценка модели как 11-классового классификатора, что позволяет оценить саму модель вне контекста задачи.

Исходными данными для расчёта метрик качества классификационной модели является количество её правильных и неправильных предсказаний на тестовой выборке датасета. Чтобы получить необходимую информацию, с использованием модели были вычислены предсказания для 2200 аудиозаписей, из которых 200 содержали нецензурные слова.

На основании правильных ответов и результатов, предсказанных моделью, можно построить матрицу ошибок (confusion matrix). Матрица ошибок — это таблица со всеми возможными комбинациями предсказанных

моделью и фактических значений, в ячейках которых записано количество соответствующих примеров [56].

На рисунке 3.2 слева представлена матрица ошибок в случае, если рассматривать модель как бинарный классификатор, а справа — если рассматривать модель как 11-классовый классификатор.

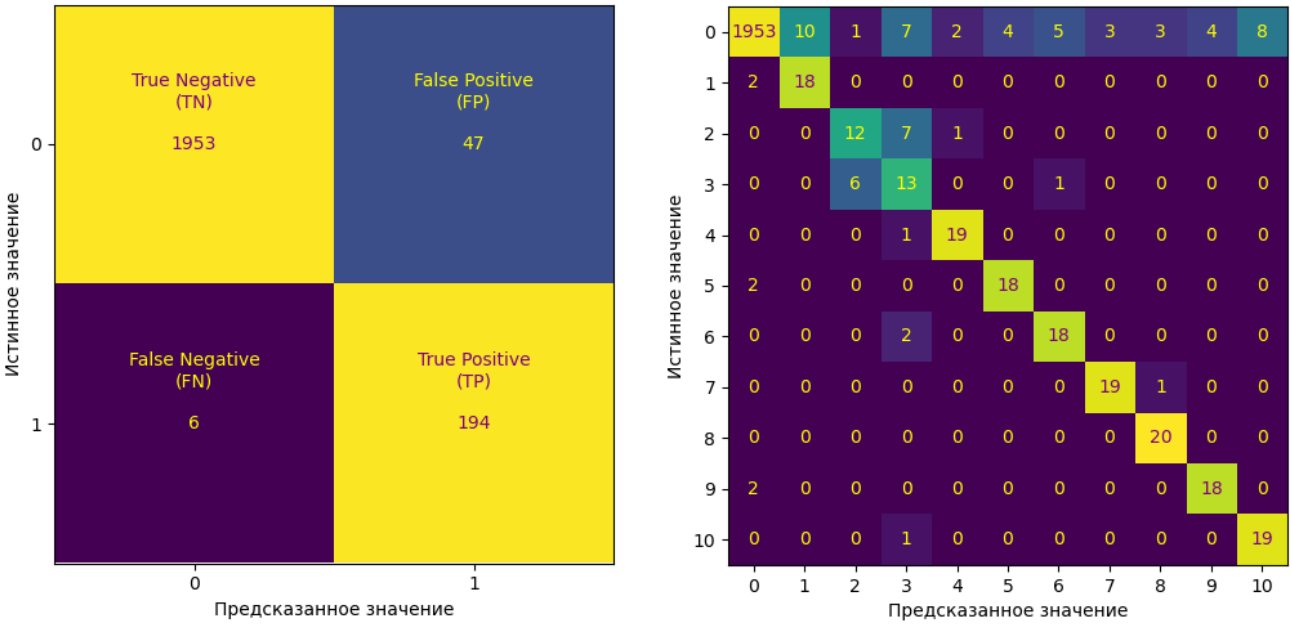

Рисунок 3.2 – Матрицы ошибок

При рассмотрении бинарной классификации ячейки таблицы можно также обозначить следующим образом:

– TP (True positive) — модель верно определила, что в аудиозаписи содержится нецензурная лексика;

– TN (True negative) — модель верно определила, что в аудиозаписи не содержится нецензурной лексики;

– FP (False positive) — в аудиозаписи отсутствует нецензурная лексика, но модель определила её наличие (ошибка первого рода);

– FN (False negative) — в аудиозаписи присутствует нецензурная лексика, но модель не определила её наличие (ошибка второго рода).

При рассмотрении матрицы ошибок многоклассовой классификации можно отметить, что модель может путать некоторые нецензурные слова друг с другом. В рамках поставленной задачи такие ошибки модели не являются критичными, однако позволяют оценить саму модель и то, на каких словах она чаще всего ошибается. Кроме того, данная информация может быть полезной в других задачах, допускающих применение данного алгоритма.

Для оценки качества модели в задачах классификации наиболее часто используемой метрикой является значение точности (accuracy). Данная метрика показывает долю правильных ответов среди всех предсказаний модели. В случае бинарного классификатора метрика accuracy может быть рассчитана по формуле 3.1:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (3.1)$$

Полученное значение метрики accuracy при рассмотрении модели как бинарного классификатора составляет 0,9759.

В случае многоклассовой классификации метрика accuracy может быть рассчитана напрямую из матрицы ошибок. Для этого необходимо сложить все элементы на главной диагонали и разделить на общее количество примеров. Полученное значение составляет 0,9668.

Различие в значениях объясняется тем, что в случае бинарной классификации за ошибку не считается определение моделью одного нецензурного слова, в то время как в действительности аудиозапись содержит другое нецензурное слово, следовательно значение accuracy получается выше, чем при 11-классовой классификации.

Метрика accuracy имеет некоторые существенные ограничения, особенно в контексте несбалансированных наборов данных. В данном случае класс unknown содержит в 10 раз больше примеров, чем все остальные классы

вместе взятые. Метрика accuracy будет отдавать предпочтение классу unknown, что приведёт к вводящей в заблуждение оценке [50].

Для более правильной оценки модели с несбалансированными классами могут использоваться такие метрики как точность (precision), полнота (recall) и F-мера (F-measure).

В случае бинарного классификатора метрика precision рассчитывается по формуле 3.2:

$$Precision = \frac{TP}{TP + FP}. \quad (3.2)$$

Метрика precision отражает, насколько надёжна модель при классификации положительных классов, при этом допуская как можно меньше ложных срабатываний. В данном случае метрика составляет 0,8050.

Метрика recall рассчитывается по формуле 3.3:

$$Recall = \frac{TP}{TP + FN}. \quad (3.3)$$

Метрика recall отражает, насколько надёжна модель при классификации положительных классов, при этом допуская как можно меньше ложных отрицательных ответов. В данном случае метрика составляет 0,97.

F-мера — это метрика, объединяющая в себе precision и recall. В зависимости от задачи одна из метрик precision или recall может быть важнее, тогда используется модифицированная метрика F-beta с параметром, отвечающим, какой из двух метрик отдаётся предпочтение. В данной задаче precision и recall считаются одинаково важными, в таком случае F-мера рассчитывается по формуле 3.4:

$$F\text{-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall}. \quad (3.4)$$

Поскольку F-мера более чувствительна к распределению данных, она подходит для оценки моделей классификации несбалансированных наборов данных. В данном случае метрика составляет 0,8798.

Данные метрики были рассчитаны для бинарного классификатора, однако для многоклассовой классификации расчёт производится другими методами. Для этого многоклассовая классификация представляется как несколько бинарных классификаций, при этом у каждой бинарной классификации стоит задача отделить один класс от остальных. Такой подход классификации называется один против остальных (One-vs-Rest, OVR). Для каждой бинарной классификации подсчитывается количество значений в ячейках матрицы ошибок TP, FP, TN, FN [56].

Для расчёта метрик многоклассовой классификации существует три способа усреднения: микро-усреднение, макро-усреднение и взвешенное.

В случае микро-усреднения сначала нужно рассчитать общее количество истинных положительных срабатываний (TP), ложных положительных (FP) и ложноотрицательных (FN) предсказаний по всем классам. Далее метрики precision, recall и F-мера вычисляются по приведённым выше формулам. При этом все три метрики принимают одинаковое значение, равное 0,9668. Это значение также равно и значению метрики accuracy. Микро-усреднение даёт одинаковый вес каждому примеру, независимо от его класса и количества примеров в классе. Поэтому при оценке многоклассового классификатора с несбалансированными классами использование микро-усреднения не даёт объективной оценки модели.

Второй способ усреднения — макро-усреднение. При таком способе вычисляются метрики precision, recall и F-мера для каждого класса по представленным формулам, а затем полученные значения усредняются. Макро-усреднение даёт равный вес каждому классу, независимо от количества

примеров. Такой способ больше подходит для оценки модели, так как он предназначен для многоклассовых классификаторов. В результате расчётов метрика precision составила 0,7583, recall — 0,8797, а F-мера — 0,8110.

Третий способ усреднения — взвешенное, при котором общие метрики precision, recall и F-мера вычисляются как средневзвешенное значение precision, recall и F-меры в отдельных классах. Данное усреднение будет работать как макро-усреднение, но вместо того, чтобы делить сумму метрик на количество классов, каждому классу даётся коэффициент на основе доли, которую он занимает в наборе данных. При таком способе усреднения метрика precision составила 0,9731, recall — 0,9668, а F-мера — 0,9691.

Такой способ подходит для оценки многоклассовых классификаторов с дисбалансом классов в том случае, когда необходимо придать большее значение классам с большим количеством примеров. В данном случае класс unknown с наибольшим количеством примеров не является приоритетным при расчёте данной метрики.

Ещё одной метрикой, используемой для оценки классификаторов, является ROC-AUC — площадь под кривой ROC. Чем выше показатель ROC-AUC, тем качественнее классификатор, при этом значение 0,5 соответствует случайному классификатору. Значение менее 0,5 означает, что классификатор инвертирован, при этом если поменять местами метки классов, классификатор может оказаться достаточно точным. Для того, чтобы вычислить метрику, потребуются значения TPR (True positive rate) и FPR (False positive rate). При хорошей классификации надо получить максимальный TPR при минимальном FPR. Для вычисления значения TPR используется та же формула, что и для метрики recall. Для расчёта FPR используется формула 3.5:

$$FPR = \frac{FP}{TN + FP}. \quad (3.5)$$

Для бинарного классификатора значение метрики ROC-AUC рассчитывается по формуле 3.6:

$$ROC\text{-}AUC = \frac{1 + TPR - FPR}{2}. \quad (3.6)$$

В результате вычислений значение ROC-AUC составило 0,9733.

При расчёте метрики ROC-AUC для 11-классового классификатора, также могут быть использованы приведённые ранее методы усреднения значений. В результате при микро-усреднении метрика составила 0,9818, при макро-усреднении — 0,9370, а при взвешенном — 0,9696. В данном случае макро-усреднение также является более показательным для оценки модели, так как такой способ усреднения подходит для оценки многоклассовых классификаторов и даёт одинаковый вес каждому классу. Все рассчитанные метрики сведены в таблицу 3.3.

Таблица 3.3 – Рассчитанные метрики

Интерпретация классификатора и усреднение		**Accuracy**	**Precision**	**Recall**	**F-мера**	**ROC-AUC**
Бинарный		0,9759	0,8050	0,9700	0,8798	0,9733
11-класс.	**Микро**	0,9668	0,9668			0,9818
	Макро		0,7583	0,8797	0,8110	0,9370
	Взвешенное		0,9731	0,9668	0,9691	0,9696

Вычисленные значения метрик показывают, что обученная модель показывает хорошие результаты при решении задачи, поставленной в данной работе. Для того, чтобы модель было возможно протестировать в прикладной задаче и проверить её работу на реальных данных, было разработано демонстрационное приложение.

3.3. Разработка приложения

Для демонстрации работы модели было разработано приложение на языке Python, позволяющее определять нецензурную лексику во входном аудиосигнале с минимальной задержкой и заменять её на тон с частотой 1 кГц. Исходный код программы представлен в приложении В.

В качестве входных данных используется потоковый цифровой аудиосигнал, полученный с выбранного устройства ввода или устройства ввода по умолчанию, к примеру линейного или микрофонного входа. Выходные данные также в виде потокового аудиосигнала направляются на выбранное устройство вывода или устройство вывода по умолчанию, например на линейный выход.

После запуска приложения происходит инициализация, состоящая из нескольких этапов. Первый этап — это разбор входных параметров, при этом все параметры являются опциональными. Параметры передаются в программы в виде аргументов командной строки. Программа может быть запущена со следующими параметрами:

– при запуске программы с параметром -d осуществляется вывод доступных звуковых устройств в системе и соответствующих индексов, после чего программа завершается;

– параметр -i устанавливает индекс устройства ввода в целочисленном формате, при отсутствии данного аргумента выбирается системное устройство ввода по умолчанию;

– параметр -o устанавливает индекс устройства вывода в целочисленном формате, при отсутствии данного аргумента выбирается системное устройство вывода по умолчанию;

– параметр -l определяет задержку в миллисекундах и задание размера буфера, значение по умолчанию 1000 мс. Слишком маленькое значение

может стать причиной того, что нецензурное слово уже будет выведено на устройство вывода до того, как оно будет обнаружено алгоритмом;

После разбора аргументов выполняется загрузка параметров модели, и самой модели. После этого загружаются наилучшие веса модели, полученные в результате обучения и модель переводится в потоковый режим с внутренним состоянием. Использование внутреннего состояния обусловлено тем, что нет необходимости явно работать с буферами состояний.

После инициализации запускается основной цикл программы, в ходе которого происходит выявление нецензурной лексики и её замена. Данный процесс происходит следующим образом. Входной сигнал считывается с устройства ввода помощью библиотеки SoundDevice [57] блоками по 20 мс. Каждый блок помещается во входную очередь.

Из входной очереди каждый блок направляется на вход модели и в конец двухсторонней очереди, которая является буфером и по умолчанию вмещает 50 блоков, то есть аудиосигнал длительностью в 1 секунду.

Так как модель имеет внутренний экстрактор признаков, на вход она принимает необработанный сигнал. Модель настроена на работу с аудиосигналом с частотой дискретизации 16 кГц, поэтому перед подачей блока на вход модели происходит передискретизация (ресемплинг) сигнала для приведения его частоты дискретизации к требуемому значению. На выходе модель выдаёт значение по каждому из классов, и то значение, которое оказывается больше остальных, и принимается за предсказанный класс.

Если модель выявляет нецензурное слово, то из начала двухстороннего буфера удаляются 5 блоков аудиосигнала и заменяются на 5 блоков синусоидального тона частотой 1 кГц. Модель выдаёт предсказание для каждого блока, поэтому для одного нецензурного слова будет последовательно выдано несколько предсказаний и будет несколько раз проведена замена на 5

блоков синусоидального сигнала, что обеспечивает замену всего нецензурного слова или части, включающей его корень, что не позволяет определить само слово в контексте фразы в соответствии с требованиями Роскомнадзора.

Так как модель может определить нецензурное слово только после того, как оно было произнесено, двухсторонний буфер должен иметь вместимость, определяемую предполагаемой длительностью нецензурных слов. По умолчанию это значение составляет 50 блоков. Как только весь буфер заполняется, блоки передаются в выходную очередь, из которой уже с помощью библиотеки SoundDevice отправляются на устройство вывода. В результате на выходе получается аудиосигнал с заменёнными нецензурными словами на тон частотой 1 кГц.

3.4. Тестирование приложения

Для тестирования разработанной программы была создана аудиозапись, содержащая 19 слов, из которых 10 нецензурных, которые чередуются с обычными словами. Нецензурные слова выбраны так, что каждое из них относится к соответствующему классу модели. Аудиозапись была подана на линейный вход для обработки программой, а результирующий сигнал был снят с линейного выхода. На рисунке 3.3 сверху визуализирован исходный сигнал, где выделены нецензурные слова, а снизу представлен сигнал после обработки программой, где выделены области с синусоидальным сигналом.

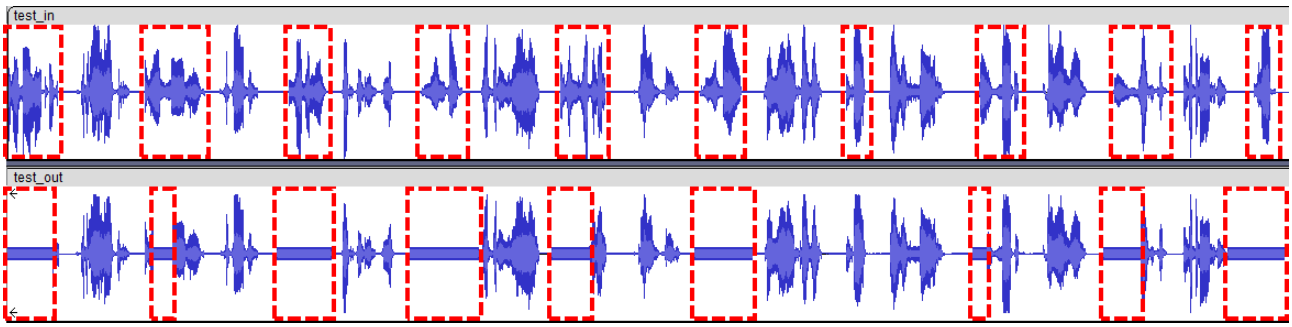

Рисунок 3.3 — Входной сигнал и сигнал после обработки программой

В результате работы программы 9 из 10 нецензурных слов были обнаружены и полностью или частично заменены на звуковой сигнал таким образом, что основная часть нецензурного слова была скрыта сигналом. Одно нецензурное слово не было обнаружено. Слова, которые не являются нецензурными, полностью сохранились в результирующей записи.

3.5. Выводы

В данном разделе было выполнено обучение нейронной сети для распознавания нецензурных слов с использованием специально собранного датасета. Полученная модель была оценена с помощью различных метрик, которые показали, что модель даёт хорошие результаты при распознавании нецензурных слов.

Для демонстрации работы модели и для её тестирования на реальных данных было разработано демонстрационное приложение, позволяющее фильтровать нецензурную лексику и заменять её на звуковой сигнал.

В результате тестирования разработанного приложения большинство нецензурных слов были отфильтрованы, при этом остальные слова не были затронуты программой, что позволяет использовать разработанное приложение для предотвращения попадания нецензурной лексики в выходной аудиосигнал.

Заключение

В рамках данной выпускной квалификационной работы было проведено исследование предметной области, в частности, исследование факторов использования нецензурной лексики, отношение общества к рассматриваемому явлению и его возможные негативные последствия. Были выявлены сферы, в которых может возникнуть необходимость выявления нецензурных выражений в реальном времени.

Проведён анализ существующих возможностей для решения поставленной проблемы, выявлены достоинства и недостатки существующих методов, и на основании этого принято решение о проектировании и разработке системы, позволяющей автоматизировать данный процесс с использованием методов машинного обучения. В процессе исследования был проведён разбор технологий, позволяющих выполнять обнаружение нецензурных слов и для решения этой проблемы выбрана современная технология обнаружения ключевых слов с использованием нейронных сетей.

Для обучения нейронной сети был собран датасет нецензурных слов с использованием данных из открытых источников. С использованием датасета была обучена модель и рассчитаны её метрики, которые показали достаточную точность полученной модели в решении поставленной задачи. На основе обученной модели было разработано приложение, позволяющее использовать модель на реальных данных.

Полученные в результате тестирования данные показывают эффективность разработанного метода выявления нецензурной лексики и доказывают возможность и целесообразность применения использованных технологий для решения этой проблемы.

Несмотря на полученную высокую точность модели, существует возможность её развития, включая добавление других оскорбительных и непристойных выражений для возможности фильтрации контента под определённую возрастную категорию, а также расширение датасета для более точного определения неподходящего контента.

В целом, разработанная система может найти применение в различных задачах, не ограничиваясь организацией прямых трансляций, а проведённое исследование позволит создавать и совершенствовать современные системы контентной фильтрации на основе методов машинного обучения, отвечающие всем предъявляемым требованиям. Дальнейшие исследования в этой области позволят добиться ещё более лучших результатов в решении подобных задач.

Список использованных источников

1. К. А. Гришева, М. В. Кондратей, В. В. Ткаченко. Ненормативная лексика, основные факторы ее употребления. — Текст: непосредственный // Молодой ученый. № 6 (65). 2014. 654-658.
2. «Могучее слово – мат!» Допустимо ли употребление мата? И нужно ли за него наказывать СМИ? // Фонд Общественное Мнение [Электронный ресурс] – Режим доступа: https://fom.ru/obshchestvo/10849
3. Великий и могучий: россияне о чистоте русского языка // Всероссийский центр изучения общественного мнения [Электронный ресурс] – Режим доступа: https://wciom.ru/analytical-reviews/analiticheskii-obzor/velikij-i-moguchij-rossiyane-o-chistote-russkogo-yazyka-
4. Кодекс Российской Федерации об административных правонарушениях [Текст]: от 30.12.2001 № 195-ФЗ; ред. от 28.04.2023 // Собрание законодательства РФ.
5. О внесении изменений в Федеральный закон «Об информации, информационных технологиях и о защите информации» и отдельные законодательные акты Российской Федерации [Текст]: федеральный закон от 05.12.2022 № 478-ФЗ // Собрание законодательства РФ.
6. О средствах массовой информации [Текст]: федеральный закон от 27.12.1991 № 2124-1; ред. от 29.12.2022 // Собрание законодательства РФ.
7. Четыре слова, которые запретили в российских СМИ // Официальный сайт Федеральной службы по надзору в сфере связи, информационных технологий и массовых коммуникаций [Электронный ресурс] – Режим доступа: https://rkn.gov.ru/press/publications/news23414.htm
8. Рекомендации по применению Федерального закона от 05.04.2013 №34-ФЗ «О внесении изменений в статью 4 Закона Российской Федерации «О средствах массовой информации» и статью 13.21 Кодекса Российской

Федерации об административных правонарушениях» // Официальный интернет-ресурс Федеральной службы по надзору в сфере связи, информационных технологий и массовых коммуникаций [Электронный ресурс] – Режим доступа: https://16.rkn.gov.ru/directions/mass-communications/control-nadzor-smi/spravka/p15638/

9. Большой толковый словарь русского языка [Текст]: А-Я / РАН. Ин-т лингв. исслед.; Сост., гл. ред. канд. филол. наук С. А. Кузнецов. - Санкт-Петербург: Норинт, 1998.
10. Словарь русской брани матизмы, обсценизмы, эвфемизмы [Текст]: Сост., Мокиенко В. М., Никитина Т. Г. - Санкт-Петербург: Норинт, 2004.
11. Самый полный словарь ненормативной лексики: [20 000 слов и фразеологических единиц] [Текст]: Сост., Квеселевич Д.И. – М.: АСТ, 2011.
12. Роскомнадзор накажет СМИ только за четыре матерных слова // Официальный сайт Федеральной службы по надзору в сфере связи, информационных технологий и массовых коммуникаций [Электронный ресурс] – Режим доступа: https://rkn.gov.ru/press/publications/news23346.htm
13. BD600 BROADCAST PROFANITY DELAY OPERATOR’S MANUAL [Электронный ресурс] – Режим доступа: https://cdn.eventideaudio.com/uploads/2021/09/BD600-Manual.pdf
14. SIMEDIA MediaDelayEditor (live censorship) [Электронный ресурс] – Режим доступа: https://www.si-media.tv/site/automation_censorship/
15. ETERE LIVE CENSORSHIP DOCUMENTATION [Электронный ресурс] – Режим доступа: https://www.etere.com/DocView/5468/LIVE-CENSORSHIP.aspx
16. Intel Bleep Beta User Guide [Электронный ресурс] – Режим доступа: https://game.intel.com/giveaway/bleepbeta/user-guide/

17. Soniox Docs: Transcribe Streams [Электронный ресурс] – Режим доступа: https://soniox.com/docs/how-to-guides/transcribe-audio/transcribe-streams.html
18. Документация Yandex Cloud: Распознавание речи [Электронный ресурс] – Режим доступа: https://cloud.yandex.ru/docs/speechkit/stt/
19. Ambuj Mehrish, Navonil Majumder, Rishabh Bhardwaj, Soujanya Poria. A Review of Deep Learning Techniques for Speech Processing (2023).
20. J.J O'Reilly. Pulse Code Modulation. // Telecommunication Principles. Springer, Dordrecht (1984).
21. Ratnadeep Deshmukh, Kishori Ghule. Feature Extraction Techniques for Speech Recognition: A Review // International Journal of Scientific and Engineering Research. 2015. 143-147.
22. Wei Han, Cheong-Fat Chan, Chiu Choy, Kong-Pang Pun. An efficient MFCC extraction method in speech recognition. // IEEE international symposium on circuits and systems. 2006.
23. Ratnadeep Deshmukh, Pratik Kurzekar, Dr. Vishal Waghmare, Pukhraj Shrishrimal. A Comparative Study of Feature Extraction Techniques for Speech Recognition System. International Journal of Innovative Research in Science, Engineering and Technology. 2014.
24. J. Meng, J. Zhang and H. Zhao. Overview of the Speech Recognition Technology // 2012 Fourth International Conference on Computational and Information Sciences, Chongqing, China. 2012. 199-202.
25. Shobha Bhatt, Anurag Jain, Amita Dev. Acoustic Modeling in Speech Recognition: A Systematic Review
26. F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss A Dynamic Language Model for Speech Recognition // International Journal of Advanced Computer Science and Applications(IJACSA), Volume 11 Issue 4. 2020.

27. María González, Julián Moreno, José Luis Martínez, Paloma Martínez. An Illustrated Methodology for Evaluating ASR Systems. 2011.
28. Wang, Y., Acero, A., Chelba, C. Is Word Error Rate a Good Indicator for Spoken Language Understanding Accuracy // IEEE Workshop on Automatic Speech Recognition and Understanding. St. Thomas, US Virgin Islands. 2003.
29. Iván López-Espejo, Zheng-Hua Tan, John Hansen, Jesper Jensen. Deep Spoken Keyword Spotting: An Overview. 2021.
30. R. Christiansen, C. Rushforth. Detecting and locating key words in continuous speech using linear predictive coding. // IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 25, no. 5. 1977. 361-367.
31. Sakoe, H., Chiba, S. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. // IEEE transactions on acoustics, speech, and signal processing. 1978. 43–49.
32. Анисимова Э.С. ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ: ИСТОРИЯ И МЕТОДОЛОГИЯ // Экономика и социум. №2-5 (15). 2015.
33. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6). 1958. 386–408.
34. Rumelhart D.E., Hinton G.E., Williams R.J. Learning Internal Representations by Error Propagation. // Parallel Distributed Processing, vol. 1. Cambridge, MA, MIT Press. 1986. 318—362.
35. Степанов, П. П. Искусственные нейронные сети / П. П. Степанов. // Молодой ученый. № 4 (138). 2017. 185-187.
36. Keiron O'Shea, Ryan Nash. An Introduction to Convolutional Neural Networks. 2015.
37. Abdel-Hamid, Ossama, Abdel-Rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn and Dong Yu. Convolutional Neural Networks for Speech Recognition. // IEEE/ACM Transactions on Audio, Speech, and Language Processing 22. 2014. 1533-1545.

38. Robin M. Schmidt. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. 2019.
39. Xiangang Li, Xihong Wu. Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition. 2014.
40. Fernández, Santiago; Graves, Alex; Schmidhuber, Jürgen. An Application of Recurrent Neural Networks to Discriminative Keyword Spotting // Proceedings of the 17th International Conference on Artificial Neural Networks: journal. Berlin, Heidelberg: Springer-Verlag. 2007. 220—229.
41. Pete Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. 2018.
42. Keyword Spotting on Google Speech Commands // Papers With Code [Электронный ресурс] – Режим доступа: https://paperswithcode.com/sota/keyword-spotting-on-google-speech-commands
43. Roman Vygon, Nikolay Mikhaylovskiy. Learning Efficient Representations for Keyword Spotting with Triplet Loss. 2021.
44. Byeonggeun Kim, Simyung Chang, Jinkyu Lee, Dooyong Sung. Broadcasted Residual Learning for Efficient Keyword Spotting.
45. Deokjin Seo, Heung-Seon Oh, Yuchul Jung. Wav2KWS: Transfer Learning From Speech Representations for Keyword Spotting. 2021.
46. Oleg Rybakov, Natasha Kononenko, Niranjan Subrahmanya, Mirko Visontai, Stella Laurenzo. Streaming keyword spotting on mobile devices. 2020.
47. Somshubra Majumdar, Boris Ginsburg. MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition. 2020.
48. Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, Yang Zhang. QuartzNet:

Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions. 2019.

49. S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. 2015.
50. Prajit Ramachandran, Zoph Barret, V. Le Quoc. Searching for Activation Functions. 2017.
51. SOVA Dataset [Электронный ресурс] – Режим доступа: https://github.com/sovaai/sova-dataset
52. Russian Open Speech To Text (STT/ASR) Dataset [Электронный ресурс] – Режим доступа: https://www.kaggle.com/datasets/tapakah68/audio-dataset
53. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N.K., Hannemann, M., Motlícek, P., Qian, Y., Schwarz, P., Silovský, J., Stemmer, G., & Veselý, K. (2011). The Kaldi Speech Recognition Toolkit. The Kaldi speech recognition toolkit. // IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. 2011.
54. Mateusz Buda, Atsuto Maki, Maciej A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. 2017.
55. Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization // 3rd International Conference for Learning Representations, San Diego. 2015.
56. Margherita Grandini, Enrico Bagli, Giorgio Visani. Metrics for Multi-Class Classification: an Overview. 2020.
57. SoundDevice Documentation [Электронный ресурс] – Режим доступа: https://python-sounddevice.readthedocs.io/en/0.4.6/

Приложение А

Исходный код программы для обработки датасета SOVA

```
import sys
import os
import wave
import json
from tqdm import tqdm
from remotezip import RemoteZip
from vosk import Model, KaldiRecognizer, SetLogLevel
from zipfile import ZipFile

words = ["\u0431\u043B\u044F", "\u043F\u0438\u0437\u0434",
         "\u0445\u0443\u0439", "\u0445\u0443\u044F",
         "\u0445\u0443\u0435", "\u0445\u0443\u0451",
         "\u0445\u0443\u044E", "\u0445\u0443\u0451\u043C",
         "\u0435\u0431\u0430", "\u0435\u0431\u043B",
         "\u0435\u0431\u0438", "\u0435\u0431\u044B",
         "\u0451\u0431\u044B", "\u0435\u0431\u0443",
         "\u0435\u0431\u0435\u0442", "\u0435\u0431\u0451\u0442",
         "\u0435\u0431\u0435\u0439", "\u0435\u0431\u0435\u043D",
         "\u043E\u0435\u0431", "\u043E\u0451\u0431",
         "\u0430\u0451\u0431", "\u0430\u0435\u0431"]

dir = os.path.dirname(__file__)
if not os.path.isdir(os.path.join(dir, 'processed')):
  os.mkdir(os.path.join(dir, 'processed'))
data_dir = os.path.join(dir, 'processed', '1')
if not os.path.isdir(data_dir):
  os.mkdir(data_dir)

model = Model(model_name="vosk-model-ru-0.42")

def get_data():
  i = 0
  with RemoteZip(sys.argv[1]) as zf:
    length = len(zf.namelist())
    for file in tqdm(zf.namelist()):
      if file.endswith('.txt'):
        with zf.open(file) as f:
          string = f.read().decode("utf-8")
          if any(w in string for w in words):
            wf0 = zf.open(file[:-3] + "wav")
            wf = wave.open(wf0)
            rec = KaldiRecognizer(model, 16000)
            rec.SetWords(True)
            rec.SetPartialWords(True)
            while True:
              data = wf.readframes(2000)
```

```
37                if len(data) == 0:
38                  break
39                rec.AcceptWaveform(data)
40              jres = json.loads(rec.FinalResult())
41              if "result" in jres:
42                for word in jres["result"]:
43                  if word["conf"] >= 0.5 and any(w in word["word"] for
                    w in words):
44                    wf.setpos(int(word["start"] * wf.getframerate()))
45                    data = wf.readframes(int((word["end"] -
                      word["start"]) * wf.getframerate()))
46                    if not os.path.isdir(os.path.join(data_dir,
                      word["word"])):
47                      os.mkdir(os.path.join(data_dir, word["word"]))
48                    with wave.open(os.path.join(data_dir, word["word"],
                      str(i) + ".wav"), 'w') as outfile:
49                      outfile.setnchannels(wf.getnchannels())
50                      outfile.setsampwidth(wf.getsampwidth())
51                      outfile.setframerate(wf.getframerate())
52                      outfile.setnframes(int(len(data) /
                        wf.getsampwidth()))
53                      outfile.writeframes(data)
54                    i += 1
55
56  get_data()
```

Приложение Б

Исходный код программы для обработки датасета Russian Open Speech To Text

```
import sys
import os
import wave
import json
import pandas as pd
from vosk import Model, KaldiRecognizer, SetLogLevel

words = ["\u0431\u043B\u044F", "\u043F\u0438\u0437\u0434",
         "\u0445\u0443\u0439", "\u0445\u0443\u044F",
         "\u0445\u0443\u0435", "\u0445\u0443\u0451",
         "\u0445\u0443\u044E", "\u0445\u0443\u0451\u043C",
         "\u0435\u0431\u0430", "\u0435\u0431\u043B",
         "\u0435\u0431\u0438", "\u0435\u0431\u044B",
         "\u0451\u0431\u044B", "\u0435\u0431\u0443",
         "\u0435\u0431\u0435\u0442", "\u0435\u0431\u0451\u0442",
         "\u0435\u0431\u0435\u0439", "\u0435\u0431\u0435\u043D",
         "\u043E\u0435\u0431", "\u043E\u0451\u0431",
         "\u0430\u0451\u0431", "\u0430\u0435\u0431"]

dir = os.path.dirname(__file__)
if not os.path.isdir(os.path.join(dir, 'processed')):
  os.mkdir(os.path.join(dir, 'processed'))
data_dir = os.path.join(dir, 'processed', '0')
if not os.path.isdir(data_dir):
  os.mkdir(data_dir)

def get_data():
  i = 0
  sp = pd.read_csv(os.path.join(sys.argv[1], "df.csv"))
  files = sp[sp['text'].str.
  contains('|'.join(words))]['audio_id'].tolist()
  length = len(files)
  for file in files:
    wf = wave.open(os.path.join(sys.argv[1], "audio_files",
    file + ".wav"), "rb")
    rec = KaldiRecognizer(model, 16000)
    rec.SetWords(True)
    rec.SetPartialWords(True)
    while True:
      data = wf.readframes(2000)
      if len(data) == 0:
        break
      rec.AcceptWaveform(data)
    jres = json.loads(rec.FinalResult())
    if "result" in jres:
```

```
      for word in jres["result"]:
        if word["conf"] >= 0.5 and any(w in word["word"] for w
        in words):
          wf.setpos(int(word["start"] * wf.getframerate()))
          data = wf.readframes(int((word["end"] - word["start"]) *
          wf.getframerate()))
          if not os.path.isdir(os.path.join(data_dir, word["word"])):
            os.mkdir(os.path.join(data_dir, word["word"]))
          with wave.open(os.path.join(data_dir, word["word"],
          str(i) + ".wav"), 'w') as outfile:
            outfile.setnchannels(wf.getnchannels())
            outfile.setsampwidth(wf.getsampwidth())
            outfile.setframerate(wf.getframerate())
            outfile.setnframes(int(len(data) / wf.getsampwidth()))
            outfile.writeframes(data)
          i += 1

get_data()
```

Приложение В

Исходный код программы для фильтрации нецензурных слов

```
import os
import json
import queue
import argparse
from collections import deque
import tensorflow as tf
import numpy as np
import sounddevice as sd
import resampy
from kws_streaming.models import utils
from kws_streaming.models import model_utils
from kws_streaming.layers.modes import Modes
import kws_streaming.models.ds_tc_resnet as ds_tc_resnet

# command-line arguments parsing

parser = argparse.ArgumentParser()
parser.add_argument("-d", help="Show list of audio devices and exit",
                    action="store_true")
parser.add_argument("-i", type=int, help="Input device index")
parser.add_argument("-o", type=int, help="Output device index")
parser.add_argument("-l", type=int, help="Latency in ms")
args = parser.parse_args()

default_samplerate = sd.query_devices(device=
sd.default.device[0])['default_samplerate']
sd.default.samplerate = default_samplerate
n_blocks = 50

if args.d:
  print(sd.query_devices())
  parser.exit(0)
if args.i:
  sd.default.device[0] = args.i
if args.o:
  sd.default.device[1] = args.o
if args.l:
  latency = args.l
  n_blocks = latency / 20
  if n_blocks < 5:
    n_blocks = 5

block_size = round(default_samplerate / n_blocks)

# flags loading and model init
```

```
44
45 tf.compat.v1.disable_eager_execution()
46 dir = os.path.dirname(__file__)
47 with tf.compat.v1.gfile.Open(os.path.join(dir, 'ds_tc_resnet',
   'flags.json'), 'r') as fd:
48   flags_json = json.load(fd)
49
50 class DictStruct(object):
51   def __init__(self, **entries):
52     self.__dict__.update(entries)
53
54 flags = DictStruct(**flags_json)
55 total_stride = 1
56
57 pools = model_utils.parse(flags.ds_pool)
58 strides = model_utils.parse(flags.ds_stride)
59 time_stride = [1]
60 for pool in pools:
61   if pool > 1:
62     time_stride.append(pool)
63 for stride in strides:
64   if stride > 1:
65     time_stride.append(stride)
66 total_stride = np.prod(time_stride)
67
68 flags.data_stride = total_stride
69 flags.data_shape = (total_stride * flags.window_stride_samples,)
70 flags.batch_size = 1
71
72 model_non_stream_batch = ds_tc_resnet.model(flags)
73 model_non_stream_batch.load_weights(os.path.join(dir, 'ds_tc_resnet',
   'best_weights')).expect_partial()
74
75 model_stream = utils.to_streaming_inference(model_non_stream_batch,
   flags, Modes.STREAM_INTERNAL_STATE_INFERENCE)
76
77 q = queue.Queue()
78 buffer = deque()
79 out_q = queue.Queue()
80
81 start_idx = 0
82 buffered_chunks = 0
83
84 # callback for each audio block
85
86 def callback(indata, outdata, frames, time, status):
87   if status:
88     print(status, file=sys.stderr)
89   q.put(indata.copy())
```

```
  try:
    outdata[:] = out_q.get_nowait()
  except queue.Empty:
    pass

# reading and processing audio

with sd.Stream(blocksize = block_size, channels=1,
callback=callback):
  while True:
    indata = q.get()
    stream_update = indata
    stream_update[np.abs(stream_update) < 1e-8] = 0
    stream_update = np.expand_dims(stream_update.flatten(), 0)

    # resampling and classification of a current block
    resampled_data = resampy.resample(stream_update, block_size, 320,
    filter="kaiser_best")
    stream_output_prediction = model_stream.predict(resampled_data)
    stream_output_arg = np.argmax(stream_output_prediction)
    if stream_output_arg > 1:
      for i in range(5):
        buffer.popleft()
      for i in range(5):
        t = (start_idx + np.arange(block_size)) / default_samplerate
        t = t.reshape(-1, 1)
        buffer.appendleft(0.1 * np.sin(2 * np.pi * 1000 * t))
        start_idx += block_size
    if buffered_chunks < n_blocks:
      buffer.append(indata[:])
      buffered_chunks += 1
    else:
      buffer.append(indata[:])
      get = buffer.popleft()
      out_q.put(get)
```