МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Кластеризация и классификация текстовых данных с использованием технологий text mining»

Обучающийся В.Д. Борисов _____
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., В.С. Климов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант Е.В. Косс
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

# Аннотация

Название выпускной квалификационной работы: «Кластеризация и классификация текстовых данных с использованием технологий text mining».

Выпускная работа состоит из введения, трёх глав, заключения, таблиц, списка литературы, включая зарубежные источники.

Ключевым вопросом дипломной работы является разработка метода классификации и кластеризации текста, для выявления эмоционального окраса текстов, публикуемых в социальных сетях.

Целью выпускной квалификационной работы является описание процесса разработки метода классификации и кластеризации текста, для выявления эмоционального окраса текстов, публикуемых в социальных сетях.

Выпускная квалификационная работа может быть разделена на следующие логически взаимосвязанные части: анализ предметной области; постановка задачи; выбор программного обеспечения; разработка метода классификации текстовых данных; разработка демонстрационного приложения; тестирование разработанного приложения.

В конце исследования мы успешно тестируем разработанный метод классификации по эмоциональному окрасу, текстов, публикуемых в социальных сетях, который позволит ускорить процесс модерации, предоставив качественные инструменты для автоматизации модерации публикуемых материалов.

Подводя итоги, мы бы хотели подчеркнуть, что данная работа актуальна в решении проблемы автоматизации обработки текстовых данных, публикуемых в социальных сетях не зависимо от площадки, что достигается независимостью встраиваемого модуля, что предоставляет возможность удобной интеграции с сервисами.

# Abstract

The title of a graduation work is: «Clustering and classification of text data using text mining technologies».

The graduation work consists of an introduction, three parts, a conclusion, tables, list of references including foreign sources.

The key issue of the thesis is the development of a method for classifying and clustering text to do the sentiment analysis of the texts published on social networks.

The aim of the graduation work is to give some information about the development of a method of classification and clustering of text to do the sentiment analysis of the texts published on social networks.

The graduation work may be divided into several logically connected parts: the analysis of the subject area; problem statement; software selection; development of a method for classifying text data; development of a demo application; testing the developed application.

Finally, we successfully test the developed classification method of sentiment analysis of the texts published on social networks, which will speed up the moderation process by providing high-quality tools for automating the moderation of published materials.

In conclusion, we would like to stress this work is relevant to solving the problem of automating the processing of text data published on social networks regardless of the site, which is achieved by the independence of the embedded module, which provides an opportunity for convenient integration with services.

# Содержание

# Введение

Всё большее количество информации окружает нас в том числе в текстовом виде. При том всё больше технологий позволяет переводить аудио и видеоконтент в текст. Повсеместная распространённость и доступность интернета позволила людям по всему миру производить всё больше информации в огромнейших объёмах.

Такое количество производимой информации требует систематизации и анализа. Текст отлично подходит для анализа за относительную простоту хранения и обработки, но основная сложность заключается в анализе естественного языка. На помощь в решении данной задачи приходят технологии текст-майнинга (text mining).

Из общеизвестных примеров можно вспомнить социальные сети, количество постов, а значит и информации, в них растёт с каждой минутой. Она полностью соответствует критерием больших данных: большой объём, многообразие и высокая скорость генерации новых данных. По этой причине обрабатывать их вручную практически невозможно.

В такой и подобной ситуации на помощь приходят алгоритмы машинного обучения, которые можно разделить на “без учителя” – кластеризация, и “с учителем” – классификация, их и рассмотрим.

Целью данной квалификационной работы является изучение и анализ технологий текст-майнинга для кластеризации и классификации текстовых данных применимо к социальным сетям для классификации постов по эмоциональному признаку, а именно на предмет негативно окрашенных высказываний.

Для достижения указанной цели в работе необходимо решить следующие задачи:

- Провести анализ задач кластеризации и классификации текстовых данных, решаемых алгоритмами текст-майнинга;
- Провести анализ популярных алгоритмов для решения задач классификации текста;

- Подобрать материалы для тестового анализа;
- Рассмотреть и применить метод обработки естественного языка;
- Применение методов классификации для определения принадлежности к каждой из кластерных групп;
- Разработать иллюстрирующее приложение с использование веб-фреймворка Django.

# 1 Анализ методов кластеризации и классификации

## 1.1 Постановка задачи

В нынешнее время социальные сети получили огромное распространение и популярность. Большинство социальных сетей позволяют кому угодно и когда угодно писать посты, комментировать посты других людей. Количество информации, генерируемой при этом, является на столько большим, что обрабатывать вручную не представляется возможным.

Люди не всегда могут похвастаться особой тактичностью и усидчивостью к собеседнику, поэтому некоторые комментарии или посты можно расценивать как «токсичные». Токсичность в данном контексте — это обобщающее слово для неприятных особенностей поведения других людей. То есть оскорбления, расизм, угрозы и прочее. Для более благоприятного образа социальной сети стоит пресекать или скрывать подобный контент.

На фоне запрета Instagram (принадлежат Meta, признанной в РФ экстремистской организацией и ее деятельность запрещена на территории РФ) эта проблема может быть особо актуальной, на рынке появилось свободное место для замены этой соцсети, а значит необходимо написать алгоритм автоматической модерации.

## 1.2 Концепция машинного обучения

Формальное определение машинного обучения, данное профессором Митчеллом, таково:

“Компьютерная программа, которая, как говорят, извлекает уроки из своего опыта E в отношении некоторого класса задач T и измеряет производительность P, если ее производительность при выполнении задач в T, измеряемая P, улучшается с опытом E.” [2].

Итак, машинное обучение состоит из алгоритмов, которые улучшают E (Опыт), P (Производительность) и T (Выполнение задачи).

– Задача (T): Задача может быть определена как реальная проблема, подлежащая решению. Система должна работать с точками данных для выполнения задач, основанных на ОД. Классификация, регрессия, кластеризация и т. д. — это несколько задач, основанных на машинном обучении.
– Опыт (E): Опыт — это знания, полученные системой из точек данных, предоставляемых алгоритму или модели. Опыт (E) используется для решения задачи (T). Контролируемое и неконтролируемое обучение, а также обучение с подкреплением — это способы приобретения опыта.
– Производительность (P): Производительность — это показатель, который показывает, работает ли используемый алгоритм или модель в соответствии с ожиданиями или нет. Это количественный показатель, который показывает, как модель выполняет задачу (T), используя свой опыт (E). Оценка точности, матрица путаницы и т. д. — это показатели, которые помогают понять производительность (P) [1][5].

Машинное обучение применяется в случаях:

– Машинное обучение может быть использовано для принятия решений, основанных на данных, в сценариях, где не хватает человеческого опыта. Например: Навигация по неизвестным территориям или планетам.
– Он используется в сценариях, где данные динамичны по своей природе, т. е. данные постоянно меняются с течением времени. Например: Сетевое подключение или доступность инфраструктуры в организации.
– Он используется в тех местах, где трудно перевести человеческий опыт в вычислительные задачи. Например, распознавание речи.
– В розничном бизнесе это помогает веб-сайтам давать рекомендации клиентам на основе их истории поиска и предыдущих покупок.

– В банковском секторе машинное обучение является наиболее важным аспектом выявления мошенничества.
– Сегодня в сфере здравоохранения машинное обучение используется для диагностики рака кожи за счет уменьшения человеческих ошибок [22].

Несколько примеров реального применения машинного обучения:

– Прогнозирование погоды
– Анализ фондового рынка
– Распознавание объектов
– Самостоятельное вождение в таких автомобилях, как Tesla.

Машинное обучение отличается от традиционного подхода в программировании и позволяет решать задачи без чёткого понимания предметной области. На рисунке 1 показано наглядное различие этих двух подходов.

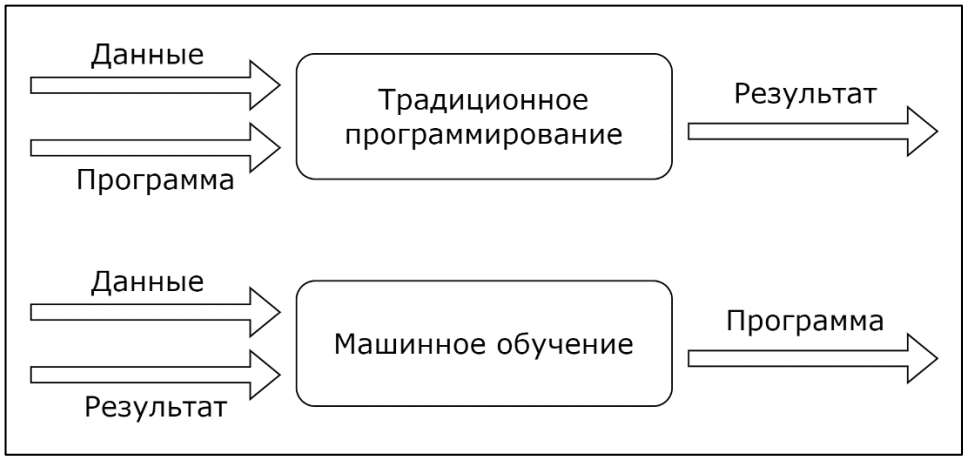


Рисунок 1 – Различие в обработке данных разных подходов

При традиционном подходе программы создаются программистом вручную, и эти программы используют входные данные и запускаются на компьютере для получения выходных данных. В машинном обучении программа создается путем отправки входных и выходных данных в алгоритм. При традиционном подходе правила формулируются или кодируются

вручную, тогда как в машинном обучении алгоритмы формулируют правила на основе предоставленных данных и являются очень мощными [1][8].

### 1.3 Определение больших данных в машинном обучении

Наука о данных — это изучение анализа данных с помощью передовых технологий (Машинное обучение, искусственный интеллект, Большие данные). Он обрабатывает огромное количество структурированных, полу структурированных, неструктурированных данных для извлечения смысла понимания, из которого может быть разработан один шаблон, который будет полезен для принятия решения об использовании новых бизнес-возможностей, улучшении продукта / услуги, в конечном счете, для роста бизнеса [14].

Процесс Data science (Наука о данных) для осмысления больших данных / огромного объема данных, используемых в бизнесе. Рабочий процесс Data science выглядит следующим образом:

- Цель и проблема определения бизнеса – Какова цель организации, какого уровня организация хочет достичь, с какой проблемой сталкивается компания — это рассматриваемые факторы. На основе таких факторов рассматривается, какой тип данных является релевантным.
- Сбор соответствующих данных - соответствующие данные собираются из различных источников.
- Очистка и фильтрация собранных данных – удаляются несущественные данные.
- Исследуйте отфильтрованные, очищенные данные – Находите любую скрытую закономерность, синхронизацию в данных, выводите их на график, диаграмму и т. д. В форме, понятной нетехническому человеку.
- Создание модели путем анализа данных – создание модели, ее проверка.

– Визуализация поиска путем интерпретации данных или создания модели для делового человека.
– Помогите бизнесмену принять решение и сделать шаг в направлении роста бизнеса.

Интеллектуальный анализ данных: это процесс извлечения смысла, скрытой закономерности из собранных данных, которые полезны для принятия бизнес-решения с целью сокращения расходов и увеличения доходов.

Большие данные: это термин, связанный с извлечением значимых данных путем анализа огромного количества сложных, по-разному отформатированных данных, генерируемых с высокой скоростью, которые не могут быть обработаны, обработаны традиционной системой.

Расширение данных изо дня в день: изо дня в день объем данных растет в геометрической прогрессии из-за современных различных источников получения данных, таких как интеллектуальные электронные устройства. Согласно отчету IDC (International Data Corporation), количество новых данных, создаваемых на каждого человека в мире в секунду, к 2020 году составит 1,7 МБ. Общий объем данных в мире к 2020 году достиг примерно 44 зеттабайт (44 триллиона гигабайт), а к 2025 году - 175 зеттабайт. По диаграмме на рисунке 2 видно, что общий объем данных буквально удваивается каждые два года.

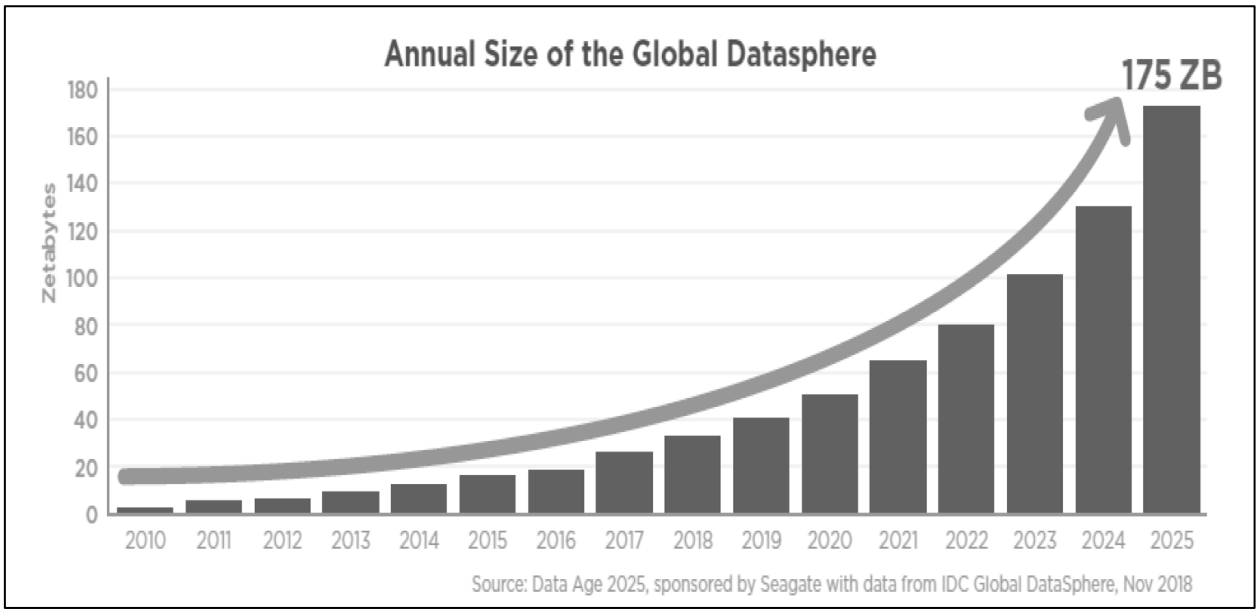


Рисунок 2 – Рост количества данных и прогнозы согласно отчету IDC

Основные источник больших данных:

Социальные сети: В современном мире значительный процент всего населения мира пользуется социальными сетями, такими как Вконтакте, Telegram, Twitter, YouTube и т. д. Каждое действие на таких носителях, как загрузка фотографии, видео, отправка сообщения, комментарий, лайк и т. д., создает данные.

Датчики, размещенные в разных местах: Датчики, размещенные в разных местах города, которые собирают данные о температуре, влажности и т. д. Камера, расположенная рядом с дорогой, собирает информацию о состоянии дорожного движения, создает данные. Камера слежения, размещенная в таких чувствительных местах, как аэропорт, железнодорожный вокзал, торговый центр, также создает большие объёмы данных.

Отзывы клиентов о продукте или услуге различных компаний на их веб-сайте создают данные. Например, такие сайты розничной торговли, как Ozon, AliExpress, Amazon, Walmart, собирают отзывы клиентов о качестве своего продукта, сроках доставки. Телекоммуникационная компания, другая организация-поставщик услуг стремятся улучшить качество обслуживания клиентов с помощью своих услуг. Они создают много данных.

Устройства интернета вещей: Электронные устройства, подключенные к Интернету, создают данные для своих интеллектуальных функций, примерами являются смарт-телевизор, смарт-стиральная машина, смарт-кофеварка, смарт-кондиционер и т. д. Это сгенерированные машиной данные, которые создаются датчиками, хранящимися в различных устройствах.

Например, интеллектуальная печатная машина – она подключена к Интернету. Несколько таких печатных машин, подключенных к сети, могут передавать данные друг другу. Таким образом, если кто-либо загружает копию файла в одну печатную машину, система сохраняет содержимое этого файла, другая печатная машина, находящаяся в другом здании или на другом этаже, может распечатать эту печатную копию файла. Такая передача данных между различными печатными машинами генерирует данные.

В транзакциях электронной коммерции, деловых операциях, банковском деле и на фондовом рынке множество хранимых записей рассматривается как один из источников больших данных. Оплата с помощью кредитной карты, дебетовой карты или другим электронным способом, все это сохраняется в виде данных.

GPS в транспортном средстве, который помогает отслеживать движение транспортного средства, чтобы сократить путь к месту назначения, сократить расход топлива и времени. Эта система создает огромные данные о местоположении и движении транспортного средства [6][8][17].

## 1.4 Виды машинного обучения

### 1.4.1 Обучение с учителем

Контролируемый алгоритм обучения подобен тому, что у вас есть учитель, который контролирует весь процесс. В этом процессе обучающий набор состоит из входных данных, которые сопряжены с правильными выходными данными, т. е. метки предоставляются вместе с данными. В процессе обучения алгоритм будет искать закономерности в данных, которые коррелируют с желаемым результатом. После процесса обучения этот

алгоритм примет новые входные данные и сможет определить, к какой метке относится новый невидимый входной сигнал, на основе предыдущих обучающих данных. На рисунке 3 наглядно изображена схема обучения с учителем.

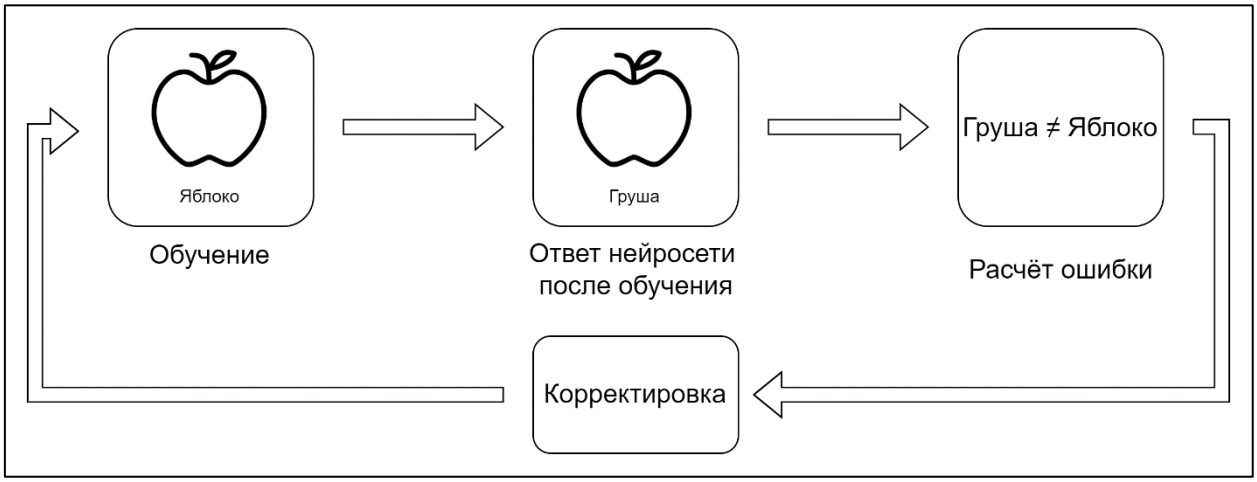


Рисунок 3 – Схема обучения с учителем

Базовая нотация алгоритма контролируемого обучения может быть записана как Y = f (x) (где Y - прогнозируемый результат, который определяется функцией отображения. x — это входное значение) [4][21].

Шаги, связанные с обучением под наблюдением:

- Сначала определите набор данных и соответствующие помеченные данные.
- Разделите набор данных на обучающий набор данных, тестовый набор данных и проверочный набор данных.
- Определите входные характеристики обучающего набора данных, которые должны обладать достаточными знаниями, чтобы модель могла точно предсказать выходные данные.
- Определите подходящий алгоритм для модели, такой как модель опорных векторов, дерево решений и т. д.
- Выполните алгоритм на обучающем наборе данных.

– Оцените точность модели, предоставив тестовый набор. Если модель предсказывает правильный результат, это означает, что наша модель точна.

Алгоритмы, используемые в обучении с учителем:

– Линейная регрессия
– Логистическая регрессия
– Наивный байесовский классификатор
– Алгоритм k-ближайших соседей
– Дерево решений
– Линейный дискриминантный анализ
– Метод опорных векторов

### 1.4.2 Обучение без учителя

Машинное обучение без учителя направлено на выявление ранее неизвестных закономерностей в данных, но в большинстве случаев эти закономерности являются плохим приближением к тому, чего может достичь контролируемое машинное обучение. Кроме того, поскольку вы не знаете, какими должны быть результаты, нет способа определить, насколько они точны, что делает контролируемое машинное обучение более применимым к реальным проблемам. На рисунке 4 изображена схема данного вида обучения.

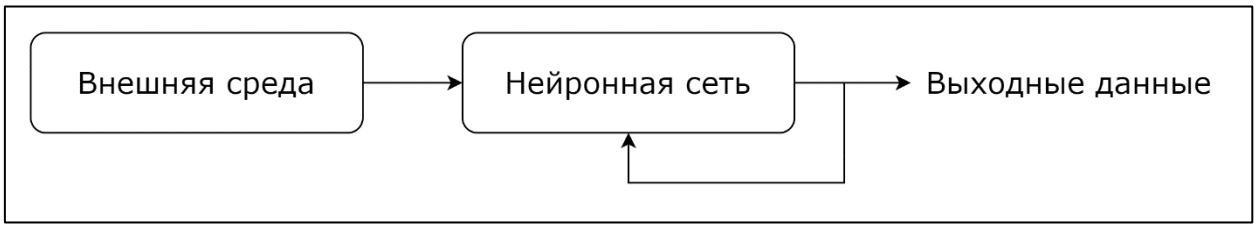


Рисунок 4 – Схема обучения без учителя

Лучшая ситуация для использования неконтролируемого машинного обучения – это когда у вас нет данных о желаемых результатах, таких как определение целевого рынка для совершенно нового продукта, который ваш

бизнес никогда раньше не продавал. Однако, если вы пытаетесь лучше понять свою существующую потребительскую базу, обучение под наблюдением является оптимальным методом [4][12].

Некоторые области применения неконтролируемых методов машинного обучения включают:

- Кластеризация позволяет автоматически разбивать набор данных на группы в соответствии со сходством. Однако часто кластерный анализ переоценивает сходство между группами и не рассматривает точки данных как отдельные лица. По этой причине кластерный анализ - плохой выбор для таких приложений, как сегментация клиентов и таргетинг.
- Обнаружение аномалий может автоматически обнаруживать необычные точки данных в вашем наборе данных. Это полезно для точного определения мошеннических транзакций, обнаружения неисправных частей оборудования или выявления выбросов, вызванных человеческой ошибкой при вводе данных.
- Интеллектуальный анализ ассоциаций определяет наборы элементов, которые часто встречаются вместе в вашем наборе данных. Розничные торговцы часто используют его для анализа корзины, поскольку он позволяет аналитикам обнаруживать товары, которые часто покупаются одновременно, и разрабатывать более эффективные стратегии маркетинга и мерчандайзинга.
- Модели скрытых переменных обычно используются для предварительной обработки данных, такой как уменьшение количества объектов в наборе данных (уменьшение размерности) или разложение набора данных на несколько компонентов.

Шаблоны, которые вы обнаружите с помощью неконтролируемых методов машинного обучения, также могут пригодиться при дальнейшем внедрении контролируемых методов машинного обучения. Например, вы можете использовать неконтролируемый метод для выполнения кластерного

анализа данных, а затем использовать кластер, к которому принадлежит каждая строка, в качестве дополнительной функции в модели контролируемого обучения (см. Машинное обучение с подкреплением). Другим примером является модель обнаружения мошенничества, которая использует оценки обнаружения аномалий в качестве дополнительной функции [12][20][21].

### 1.4.3 Обучение с подкреплением

Машинное обучение с подкреплением – это комбинация контролируемых и неконтролируемых методов машинного обучения.

С помощью более распространенных методов контролируемого машинного обучения вы обучаете алгоритм машинного обучения на “помеченном” наборе данных, в котором каждая запись содержит информацию о результатах. Это позволяет алгоритму выводить закономерности и определять взаимосвязи между вашей целевой переменной и остальной частью набора данных на основе уже имеющейся у него информации. В отличие от этого, неконтролируемые алгоритмы машинного обучения извлекают уроки из набора данных без переменной результата. При обучении с подкреплением алгоритм обучается на основе набора данных, который включает как помеченные, так и немаркированные данные, обычно в основном немаркированные [11][12].

Когда у вас недостаточно помеченных данных для создания точной модели, и у вас нет возможности или ресурсов для получения большего количества данных, вы можете использовать методы с подкреплением, чтобы увеличить размер ваших обучающих данных. Например, представьте, что вы разрабатываете модель, предназначенную для выявления мошенничества для крупного банка. О каком-то мошенничестве вы знаете, но другие случаи мошенничества проходят мимо вашего ведома. Вы можете пометить набор данных известными вам случаями мошенничества, но остальные ваши данные останутся немаркированными. Также вы можете использовать метод обучения

с подкреплением для маркировки данных и переобучения модели с помощью нового помеченного набора данных. Схема описанного алгоритма представлена на рисунке 5.

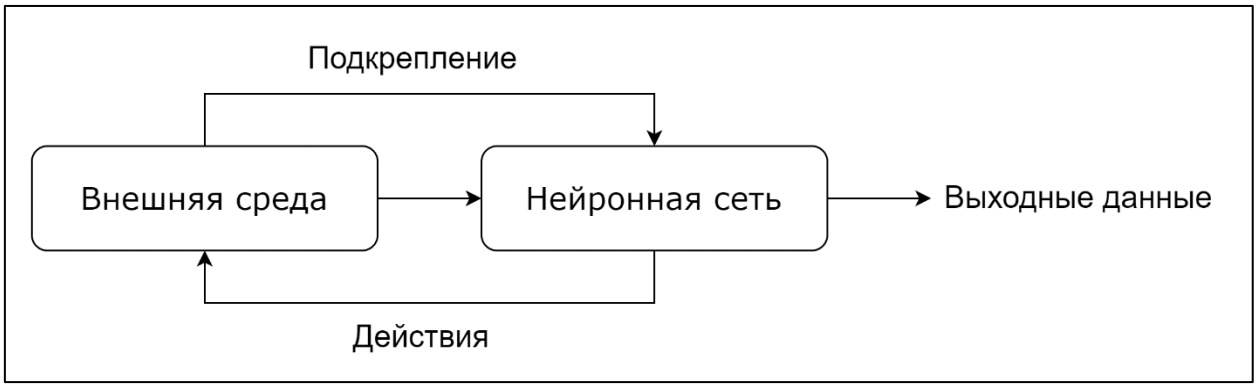


Рисунок 5 – Схема обучения с подкреплением

Затем вы используете переобученную модель к новым данным, более точно выявляя фальсификации с использованием контролируемых методов машинного обучения. Однако нет никакого способа проверить, что алгоритм произвел метки, которые являются точными на 100%, что приводит к менее надежным результатам, чем традиционные контролируемые методы [16][21].

## 1.5 Предварительная обработка текста

### 1.5.1 Токенизация

Токенизация — это первый шаг в текстовой аналитике. Процесс разбиения абзацев текста на более мелкие фрагменты, такие как слова или предложения, называется токенизацией. Наглядный процесс токенизации представлен на рисунке 6. Маркер — это единое целое, которое является строительными блоками для предложений или абзацев [9].

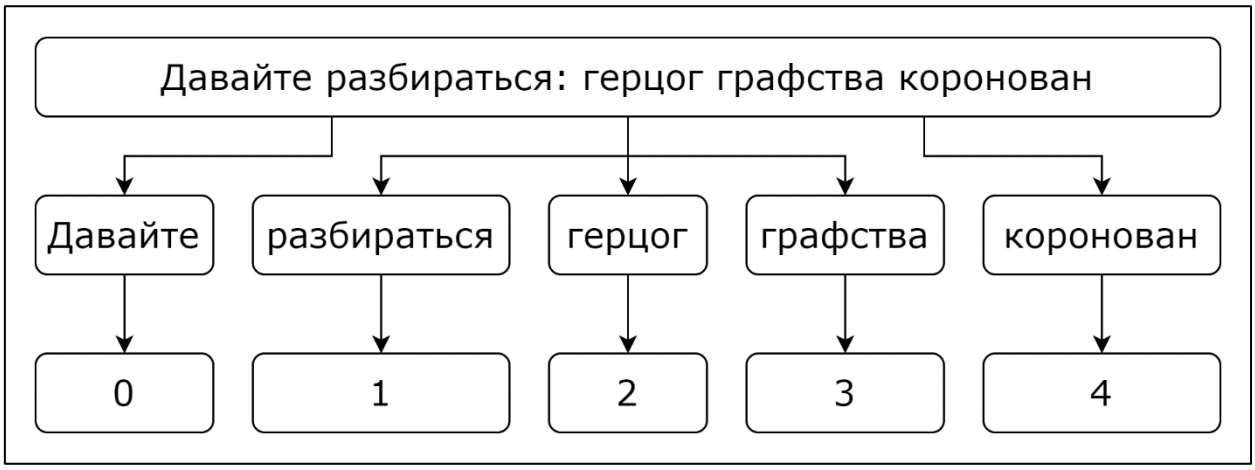


Рисунок 6 – Пример работы токенизации

Предварительная идентификация базового блока является очень важной частью обработки, и ошибки при обработке текста распространяются на весь анализ.

Маркировка слов может показаться простой в языках, где слова разделены специальными символами "пробел". Однако не все языки разделяют слова пробелами (например, китайский, японский, тайский), и при ближайшем рассмотрении становится ясно, что в английском тоже не хватает места. Однако ошибки, допущенные на этом этапе, распространяются на более поздние этапы и вызывают проблемы. Для решения этой проблемы, в дополнение к стандартному токенизатору, было разработано множество передовых методов для решения определенных задач в области токенизации.

Первым шагом в большинстве приложений для обработки текстов является разделение текста на слова. Во всех современных языках, использующих латиницу, кириллицу или греческий, таких как английский или другие европейские языки, словосочетания разделяются пробелами. Таким образом, в таких языках, называемых сегментированными языками, большинство токенов связаны явными разделителями, такими как пробелы или знаки препинания. В таких случаях вы можете использовать алгоритм, который просто разделяет токены на основе пространства между объектами в документе. Потому что более сложная обработка требует больше времени.

Большинство существующих токенизаторов указывают границы токенов пробелом. Итак, если такой токенизатор находит 2 токена непосредственно рядом друг с другом, когда за словом следует запятая, затем вставьте пробел между ними. Поскольку данные обрабатываются на английском языке, в этой задаче используется именно этот метод токенизации.

В английском и других индоевропейских языках точка ставится непосредственно перед предыдущим словом, обычно это еще один знак, указывающий на конец предложения. Однако, если точка следует за аббревиатурой, она является неотъемлемой частью этой аббревиатуры и должна быть обозначена ею. К сожалению, многие сокращения и аббревиатуры не имеют общепринятых стандартов. Наиболее обширный подход к обработке сокращений заключается в ведении списка известных сокращений. Таким образом, во время лексического анализа слов, заканчивающихся точками, его можно сравнить с аналогичным списком, и если есть токен, он будет помечен как 1 токен, а если нет, конечно, точность такого подхода зависит от того, насколько хорошо список сокращений вписывается в анализируемый текст обработано. Текст почти наверняка будет содержать сокращения, которые не включены в список. Кроме того, сокращения в списке могут совпадать с общеупотребительными словами и приводить к неправильной маркировке [9][15][20].

### 1.5.2 Стоп-лист

Стоп-слова — это набор часто используемых слов в языке. Примерами стоп-слов в английском языке являются «он», «она», «в», «на» и т. д. Стоп-слова обычно используются в интеллектуальном анализе текста и обработке естественного языка (NLP) для исключения слов, которые настолько часто используются, что несут очень мало полезной информации [9].

Например, в контексте поисковой системы, если ваш поисковый запрос «что такое стоп-слово?», вы хотите, чтобы поисковая система сосредоточилась

на поиске документов, в которых говорится о стоп-слове, а не документов, в которых просто встречается фраза «что такое».

Это можно сделать, сохранив список стоп-слов (который может быть обработан вручную или автоматически) и предотвратив анализ всех слов из вашего списка стоп-слов. В примере выше слова можно было бы исключить, оставив только слова: «стоп-слово». Это гарантирует, что документы, имеющие актуальную тематику, будут иметь высокий рейтинг в результатах вашего поиска. Другой пример исключения стоп слов представлен на рисунке 7.

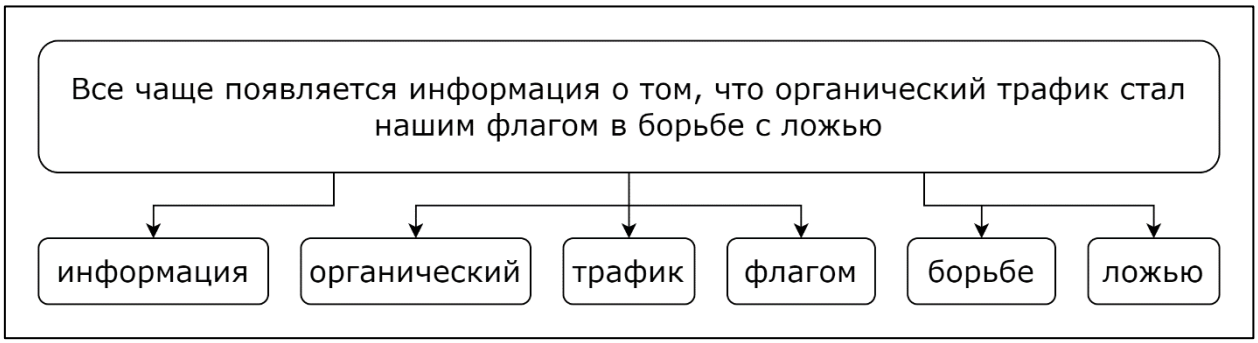


Рисунок 7 – Пример работы стоп-листа

В то время как стоп-слова обычно используются для удаления слов с низким содержанием информации, стоп-слова также могут стать полезными для добавления контекста. Например, в запросе «что такое стоп-слово?», если вы знаете, что на самом деле это вопрос «что такое?», а не вопрос «как использовать?», вы можете дополнительно уточнить результаты, показанные пользователям. Один из способов узнать это - просто посмотреть на слова, не относящиеся к теме.  Как только у вас будет эта контекстная информация вместо того, чтобы ранжировать «Как использовать стоп-слова?» в самом верху результатов поиска, вы можете научить свои алгоритмы ранжировать документы, связанные с «Что такое стоп-слова?», намного выше.

Существуют установленные списки стоп-слов, которые вы можете легко подключить и использовать. Некоторые списки стоп-слов являются

результатом исследовательской работы NLP, а некоторые просто составляются вручную разными людьми.

Хотя довольно легко использовать опубликованный набор стоп-слов, во многих случаях таких стоп-слов будет недостаточно. Например, в клинических текстах такие термины, как «доктор», «пациент» или «мг» / «мкг», встречаются почти в каждом документе, с которым вы сталкиваетесь. Таким образом, эти термины можно рассматривать как потенциальные стоп-слова для анализа и поиска клинического текста.

Аналогично, для твитов (постов социальной сети Twitter) такие термины, как «#», «RT», «@username», потенциально могут рассматриваться как стоп-слова. К сожалению, специфические для языка стоп-слова не охватывают термины, относящиеся к конкретной предметной области. Хорошей новостью является то, что вы можете легко создать свой собственный список стоп-слов для конкретной задачи [6][20].

### 1.5.3 Лемматизация и стемминг

Лемматизация - один из наиболее распространенных методов предварительной обработки текста, используемых в обработке естественного языка (NLP) и машинном обучении в целом. В лемматизации мы пытаемся свести данное слово к его корневому слову. Корневое слово называется основой в процессе поиска основы (Стемминг), и оно называется леммой в процессе лемматизации. Но между ними есть несколько различий, кроме этого. Давайте посмотрим, что это такое. Процесс лемматизации представлен на рисунке 8.

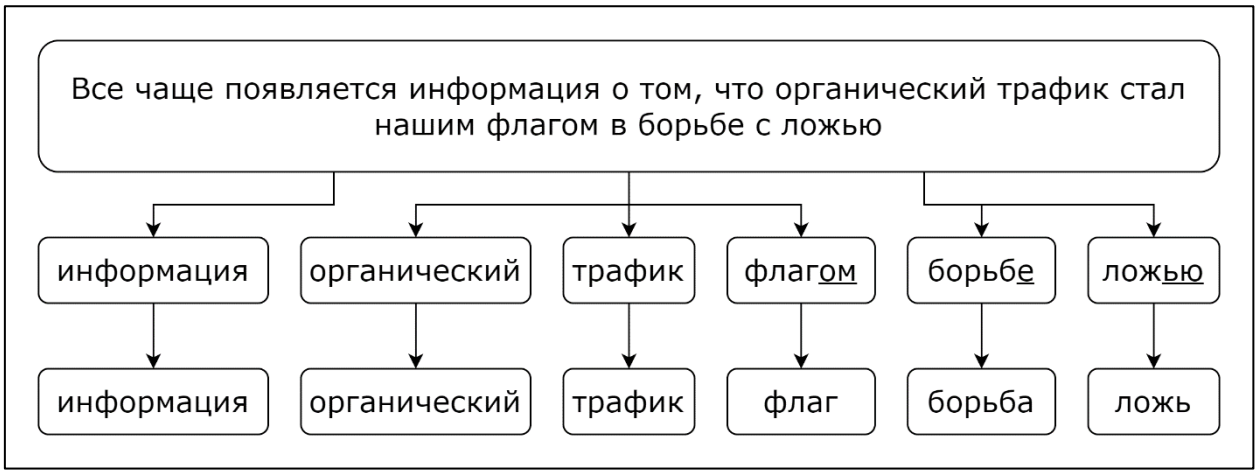


Рисунок 8 – Пример работы лемматизации

Сте́мминг (stemming — находить происхождение) — это процесс нахождения основы слова для заданного исходного слова. Основа слова не обязательно совпадает с морфологическим корнем слова. Задача нахождения основы слова представляет собой давнюю проблему в области компьютерных наук.

При нахождении основы часть слова просто отрубается на конце, чтобы добраться до основы слова. Существуют разные алгоритмы, используемые для определения того, сколько символов нужно отрезать, но алгоритмы на самом деле не знают значения слова на языке, к которому оно принадлежит. С другой стороны, при лемматизации алгоритмы обладают этим знанием. На самом деле, даже можно сказать, что эти алгоритмы ссылаются на словарь, чтобы понять значение слова, прежде чем свести его к корневому слову или лемме.

Таким образом, алгоритм лемматизации будет знать, что слово "лучше" происходит от слова "хороший", и, следовательно, лемма хороша. Но алгоритм стемминга не смог бы сделать то же самое. Там может быть чрезмерное или недостаточное значение, и слово "лучше" может быть сокращено до "луч", либо просто сохранено как "лучше". Но нет никакого способа сделать вывод, что его можно было бы свести к его источнику - слову "хорошо". В этом, по сути, и заключается разница между стеммированием и лемматизацией.

Лемматизация и стемминг являются гораздо более сложными, чем описанный выше процесс. Есть еще много вещей, которые следует рассмотреть в отношении обоих подходов, прежде чем принимать решение. Но на практике редко заметно какое-либо значительное улучшение эффективности и точности продукта, который использует лемматизацию вместо стемминга. Но в большинстве случаев, повышенные затраты, которые требует лемматизация, не оправданы. Так что это зависит от рассматриваемого проекта, для некоторых задач затраты на лемматизацию вполне оправданы, и на самом деле лемматизация в данных случаях является необходимостью [18][24].

### 1.6 Математическая модель нейронных сетей

В нынешнее время нейронные сети имеют большое распространение в обработке информации, в том числе в классификации текста. Нейронные сети работают подобно нейронам в мозгу. Под внешним, «входным» воздействием нейрон может создать выходной импульс, который распространится по аксону дальше. Между поступлением входного импульса и импульсом на выходе проходит сравнительно большое время, приблизительно 1 миллисекунда, в отличие от компьютера, где затраты времени на подобные операции занимают наносекунды.

Высокая «вычислительная эффективность» мозга достигается благодаря одновременной работы его нейронов, в то время как в компьютере все эти процессы происходят последовательно, что является основным тормозящим аспектом в использовании нейронных сетей.

Представим задачу моделирования и обучения искусственной нейронной сети, представляющей собой довольно распространенную топологию, описываемую в системе дифференциальных уравнений с задержками, вызванными конечным временем передачи сигналов с входных сигналов на синаптические выходы (от одного нейрона к другому). Для решения этой задачи оптимальные весовые коэффициенты нейронной сети

определяются с использованием аппарата математической теории оптимального управления, а также используется теория решения дискретных задач с использованием методов нелинейного программирования.

На динамическое поведение нейронной сети влияют такие факторы, как входные данные, топология сети и физические свойства отдельных элементов. При этом учитывается эффект задержки при передаче сигналов от одного нейрона. Взаимодействие нейронов описывается весовыми коэффициентами (w), которые характеризуют силу воздействия нейронов к нейронам: является ли связь возбуждающей, подавляющей, нейтральной или полностью отсутствует. Рассмотрим представление строение нейрона на рисунке 9.

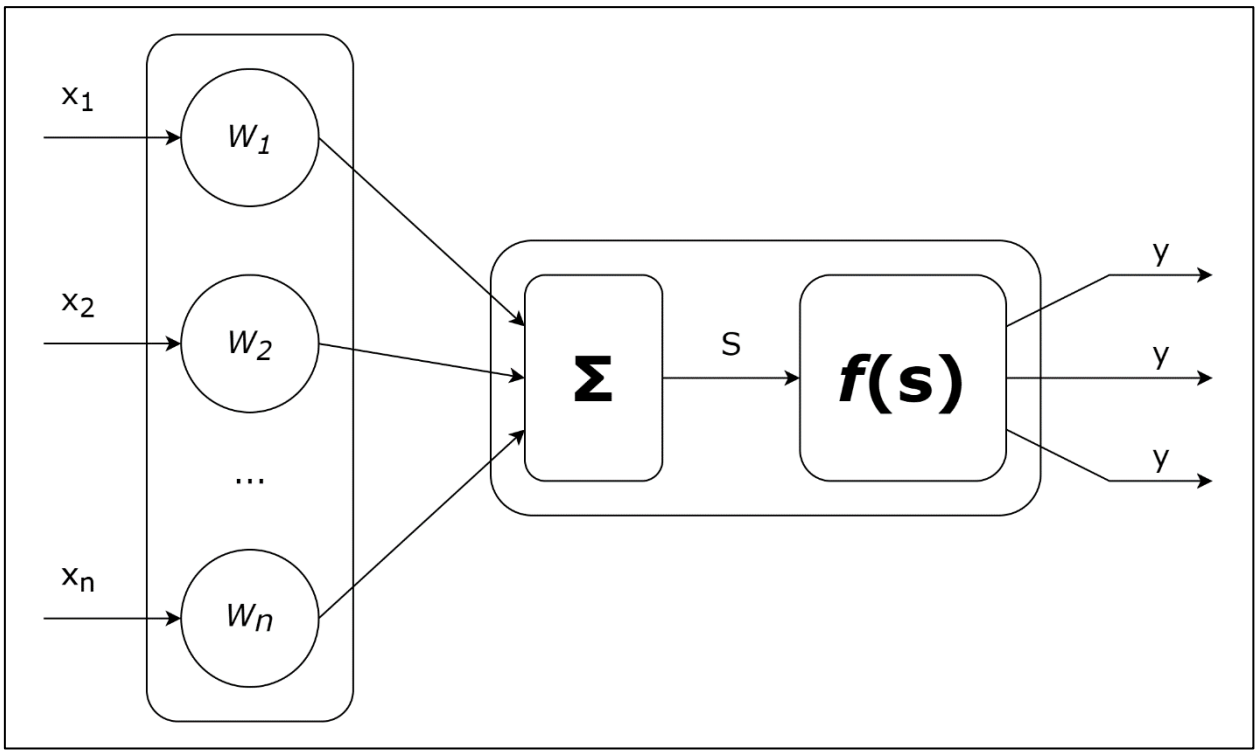


Рисунок 9 – Схема нейрона

Нейроны действуют как относительно простые элементы, которые имитируют работу нейронов в мозге. Искусственно созданные нейроны и их естественные прототипы представляют собой группу синапсов (входов), соединенных с выходом других нейронов ($x_1 \dots x_n$), а выходные соединения

этого нейрона — аксоны ( y ), из которых сигналы возбуждения или торможения поступают в синапсы других нейронов.

Искусственные нейроны состоят из двух элементов: взвешенного сумматора ($\Sigma$) и нелинейного преобразователя (f(s)). На вход искусственного нейрона поступает определенный набор сигналов, каждый из которых является выходным другого нейрона. Каждый входной сигнал умножается на соответствующий вес ($w_1 \dots w_n$), аналогичный синоптической силе, и все выходные данные суммируются для определения уровня активации нейрона. [8][15]

Взвешенный сумматор производит суммирование по формуле:

$$s_{ij} = \sum_{i=1}^{n} x_i w_{ijl} + w_{0jl} \quad (1)$$

где, $n$ – номер входа нейрона;

$i$ – номер входа нейрона;

$j$ – номер нейрона в слое;

$l$ – номер слоя;

$x_i$ – входные сигналы, совокупность входных сигналов образуют входной вектор $X$;

$w_i$ – весовые коэффициенты, совокупность коэффициентов весов образует вектор весов $W$;

$w_0$ – вес, моделирующий пороговый уровень нейрона, часто равен 1.

Другой вариант записи:

$$S = X^T W \quad (2)$$

где, $X^T$ – транспонированный вектор входных сигналов нейрона, подающихся на вход;

$W$ – вектор весов.

Выход сумматора преобразуется нелинейным элементом по формуле:

$$y_{jl} = f(s_{jl}) \quad (3)$$

где, $f(s_{jl})$ – функция активации.

Функция активации подбирается под специфику решаемой задачи. В случае данной работы будет использован сигмоида:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \quad (4)$$

График функции представлен на рисунке 10.

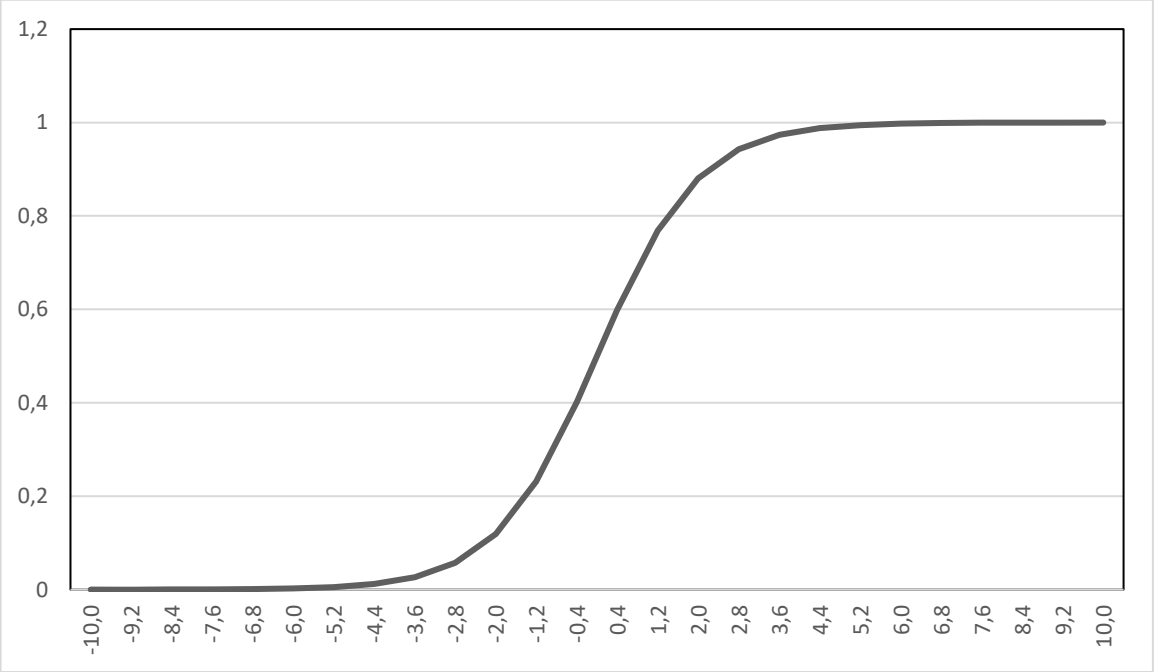


Рисунок 10 – График сигмоиды

Эта функция реализована в библиотеке PyTorch в качестве метода `sigmoid()`, что позволяет не описывать её вручную. Данный метод небинарен и нелинеен. Сигмоида является очень популярной функцией в данное время, так как не приводит к ошибкам в случае больших значений, а также за качественное сглаживание значений взвешенного сумматора. [15][21]

Вывод по разделу 1

В данном разделе мы описали принципы кластеризации и классификации текстовых данных, обозначили основные методы предварительной обработки текста. А также рассмотрели математическую модель нейрона в нейросети.

# 2 Разработка и реализация программных модулей

## 2.1 Выбранные технологии

В качестве языка программирования выбран язык Python. По причине того, что он мне уже знаком, и является основным языком, используемым для работы с машинным обучением и нейронными сетями.

Для отображения интерфейса и прочих данных воспользуемся веб-фреймворком для создания RIA-веб-приложений Django. Он предлагает сервер-ориентированную архитектуру. Использование языка Python позволяет выполнять основную часть логики приложения на стороне сервера, в том числе позволит использовать различные библиотеки для разработки и использования нейронных сетей непосредственно со стороны бэкенда. Для отображения элементов пользовательского интерфейса и взаимодействия с сервером на стороне клиента Django использует собственный набор веб-компонентов или JavaScript-библиотеки Vue, React и Angular [7].

В качестве базы данных будет использован PostgreSQL, объектно-реляционная система управления базами данных. По причине того, что система PostgreSQL является бесплатной, простой в администрировании и знакомой мне.

Для обработки естественного языка используется BERT – нейронная сеть от Google, нацеленная на обработку естественного языка. Эта нейронная сеть повсеместно используется в поисковой строке Google. Так же на базе BERT можно создавать программы с ИИ для обработки естественного языка: отвечать на вопросы, заданные в произвольной форме, создавать чат-ботов, автоматические переводчики, анализировать текст и так далее [3].

## 2.2 Оценка «токсичности» текста

Для определения наличия в тексте оскорблений и угроз будем использовать модель rubert-tiny. Это упрощённая и уменьшенная модель на основе bert-base-multilingual-cased, разработанная исключительно для русского языка. Эта модель полезна, в случаях необходимости точно

настроить ее для относительно простой задачи на русском языке (например, NER или классификация настроений), её преимущество в скорости обработке и размере. Она примерно в 10 раз меньше и быстрее, чем BERT базового размера.

Для определения настроения текста модель необходимо дообучить. Процесс дообучения происходит посредством обучения существующей модели с использованием нового датасета для получения нового назначения. Для дообучения будем использовать данные, предоставленные на соревновании «OK ML Cup», они представлены в виде обезличенных комментариев из социальной сети одноклассники с пометкой эмоционального окраса.

Дообучение модели происходит средствами языка Python и библиотеки PyTorch. Он стал очень популярным и эффективным фреймворком для создания модели глубокого обучения (Deep Learning). Эта библиотека машинного обучения с открытым исходным кодом основана на Torch и предназначена для обеспечения большей гибкости и увеличения скорости реализации глубоких нейронных сетей. В настоящее время PyTorch является наиболее популярной библиотекой для исследователей и практиков ИИ (искусственного интеллекта) во всем мире в промышленности и академических кругах [24][25].

Так как для выбранной задачи в найденном наборе данных различают разные варианты негативного окраса, для описания постов воспользуемся условными величинами

- normal – на сколько текст нейтрален и «безопасен»;
- insult – на сколько текст является оскорбительным;
- obsenity – уровень непристойности текста;
- threat– на сколько текст может считаться угрожающим;
- dungerous – опасность такого текста.

Каждый параметр принимает значение от нуля до одного, где ноль – полное несоответствие параметру, единица – полное соответствие.

На основе полученной из модели оценки можно сделать вывод о «токсичности» текста и принять решение о его скрытии или удалении. На рисунке 11, представлен код метода определяющего эмоциональный окрас полученного на вход текста [23].

```
def check_toxicity_full(text, aggregate=True):
    with torch.no_grad():
        inputs = tokenizer(text, return_tensors='pt',
truncation=True, padding=True).to(model.device)
        test =
torch.sigmoid(model(**inputs).logits).cpu().numpy()
    if isinstance(text, str):
        test = test[0]
    if aggregate:
        return 1 - test.T[0] * (1 - test.T[-1])
    return test
```

Рисунок 11 – Фрагмент кода анализирующий переданный на вход текст

Фрагмент кода записан в отдельный файл для удобства вызова его из стандартного файла views.py веб-фреймворка Django.

### 2.3 Обработка постов

Для обработки постов напишем метод, который будет принимать на вход текст публикации, а возвращать массив из пяти критериев, или по упрощённой схеме будем возвращать одно число, вычисляемое на основе нейтральности и опасности поста, означающее общую «токсичность» текста.

За опорные точки возьмём следующие числа:

- «Токсичность» выше 0.99 – пост является слишком агрессивным и не подходит правилам площадки.
- «Токсичность» выше 0.5, но ниже 0.99 – пост является потенциально агрессивным, и должен быть скрыт, но не удален.
- «Токсичность» ниже 0.5 – обычный пост.

Обернём предыдущий метод (Код представлен на рисунке 12), своего рода функцию активации, которая будет возвращать числа 0, 1 или 2, где 0 –

пост не нарушает правил площадки, 1 – пост должен быть скрыт, 2 – пост должен быть удалён [23].

```
def check_toxicity(text):
    if check_toxicity_full(text) <= 0.5:
        return 0
    elif check_toxicity_full(text) > 0.99:
        return 2
    return 1
```

Рисунок 12 – Метод, разделяющий на кластеры на основе вывода модели

Данный фрагмент кода также записан в отдельный файл для удобства вызова его из стандартного файла views.py веб-фреймворка Django.

### 2.4 Веб-приложение

Напишем демонстрационное веб-приложение с использованием написанных в предыдущем пункте методов. Если текст поста нормальный, он выводится без изменения, если текст помечен как потенциально агрессивный, он выводится под цензурой и показывается только при наведении курсора мыши, вместо удалённых постов показывается сообщение о том, что оно удалено.

В качестве интерфейса демонстрационного приложение будет использоваться веб-страница, для её отображения принято решение использовать популярный веб-фреймворк Django. Напишем с его помощью небольшую форму, в которую можно вводить текст постов, а над ней будут показаны предыдущие посты. UML диаграмма взаимодействия с приложением по этому сценарию изображены на рисунке 13.

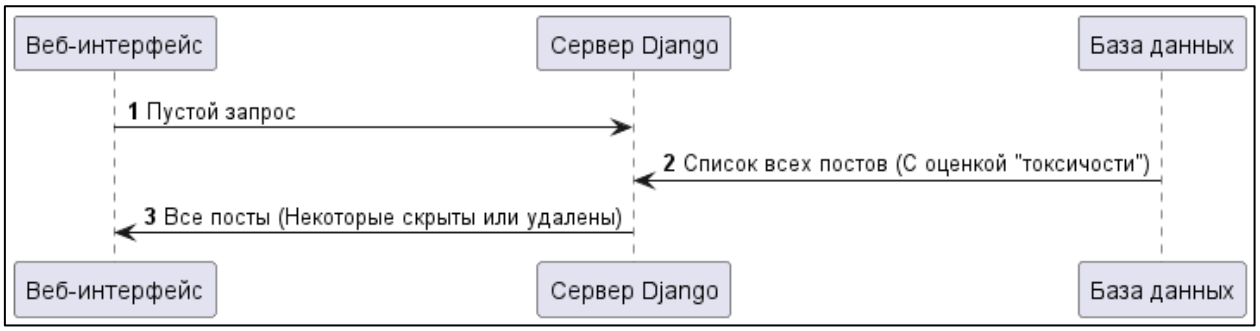


Рисунок 13 – Схема взаимодействия с приложением без добавления поста

После ввода текста пост будет обрабатываться нейросетью и добавляться в базу данных вместе с результатом обработки для дальнейшего формирования страницы. Вариант UML диаграммы с записью в базу данных изображены на рисунке 14 [10][19].

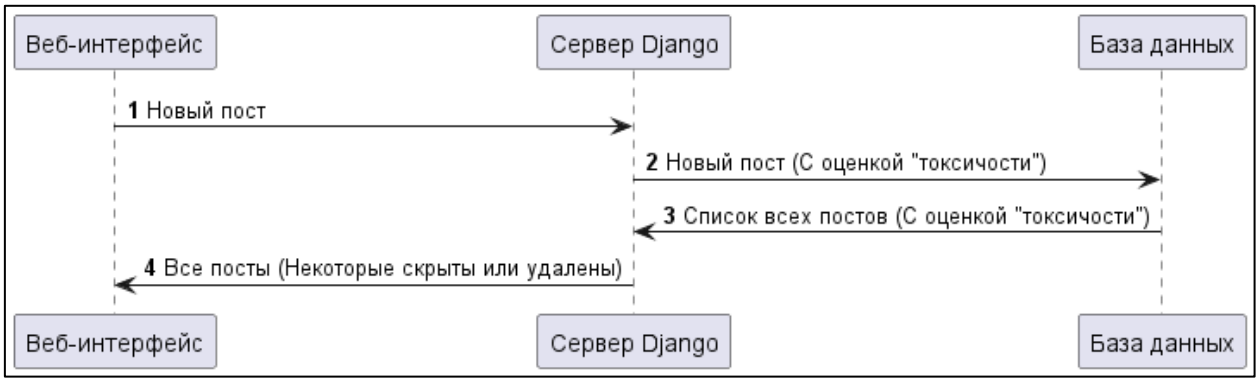


Рисунок 14 – Схема взаимодействия с приложением при добавлении поста

Напишем простой обработчик запросов, который будет принимать полученный текст, если таковой имеется, обрабатывать при помощи обученной модели и вместе с её оценкой записывать этот пост в базу данных. Независимо от наличия текста в запросе, из базы данных формируется список всех постов и предаётся в шаблон странице, где уже формируется красивый список, с учётом оценки.

Код обработчика запросов представлен на рисунке 15. Он возвращает шаблон страницы, описанный в файле `index.html` [7][10][13].

```
def index(request):
    text = request.GET.get("text", "")
    if text != "":
        rc = check_toxicity(text)
        p, created = Post.objects.get_or_create(text=text,
toxicity=rc)

    template = loader.get_template('index.html')
    context = {
        'posts': sorted(list(Post.objects.all()), key=lambda
post: post.id),
    }
    return HttpResponse(template.render(context, request))
```

Рисунок 15 – Код обработчика запросов

Фрагмент кода записан в стандартный файл views.py веб-фреймворка Django и обращается к ранее созданным методам.

Вывод по разделу 2

В данном разделе мы рассмотрели выбранные технологии подробнее и с их помощью написали программную реализацию для классификации текстовых постов по эмоциональному окрасу и автоматическую модерацию в демонстрационном приложении.

# 3 Тестирование и отладка программных модулей

## 3.1 Демонстрационное приложение

Запустим наше приложение дабы удостовериться в его работоспособности. После запуска на компьютере запустится локальный сервер Django, к которому мы сможем обратиться по адресу http://127.0.0.1:8000/. Окно браузера с открытой страницей на рисунке 16.

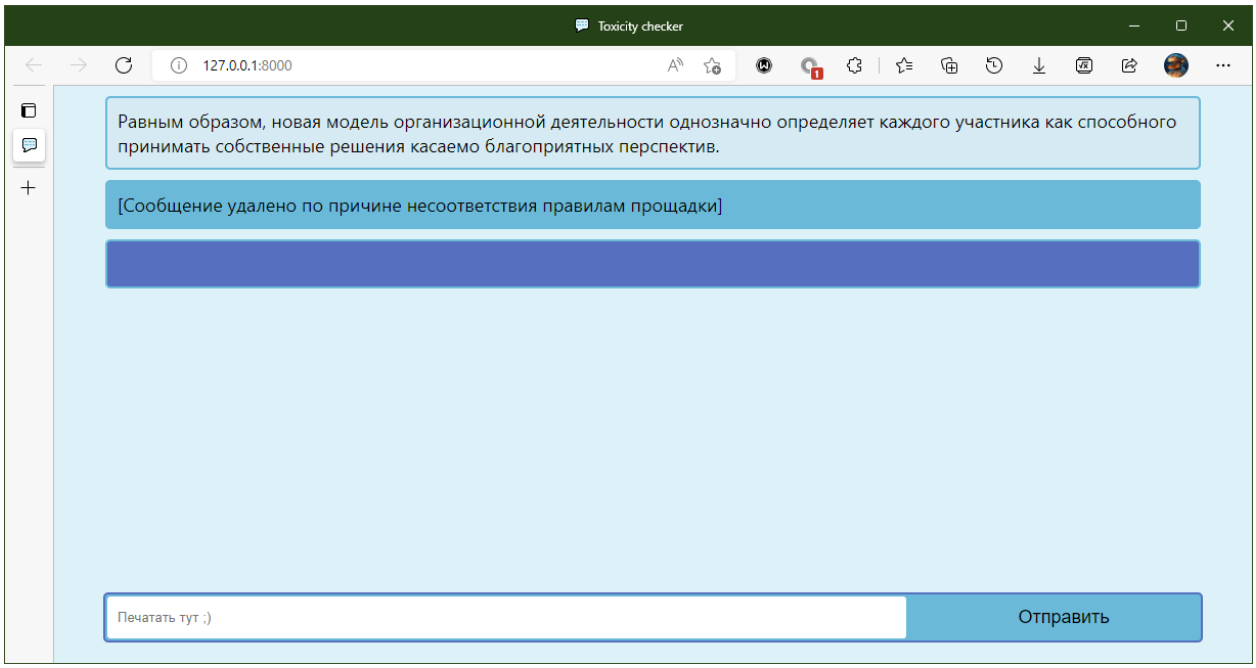


Рисунок 16 – Страница приложения

На странице приложения мы можем видеть предсозданные посты и форму для ввода текста. Попробуем выложить новый пост со следующим текстом: «Известный инсайдер, в преддверии важного события, даёт нам право принимать самостоятельные решения». Поле для ввода текста расположено внизу страницы. Страница после ввода текста представлена на рисунке 17.

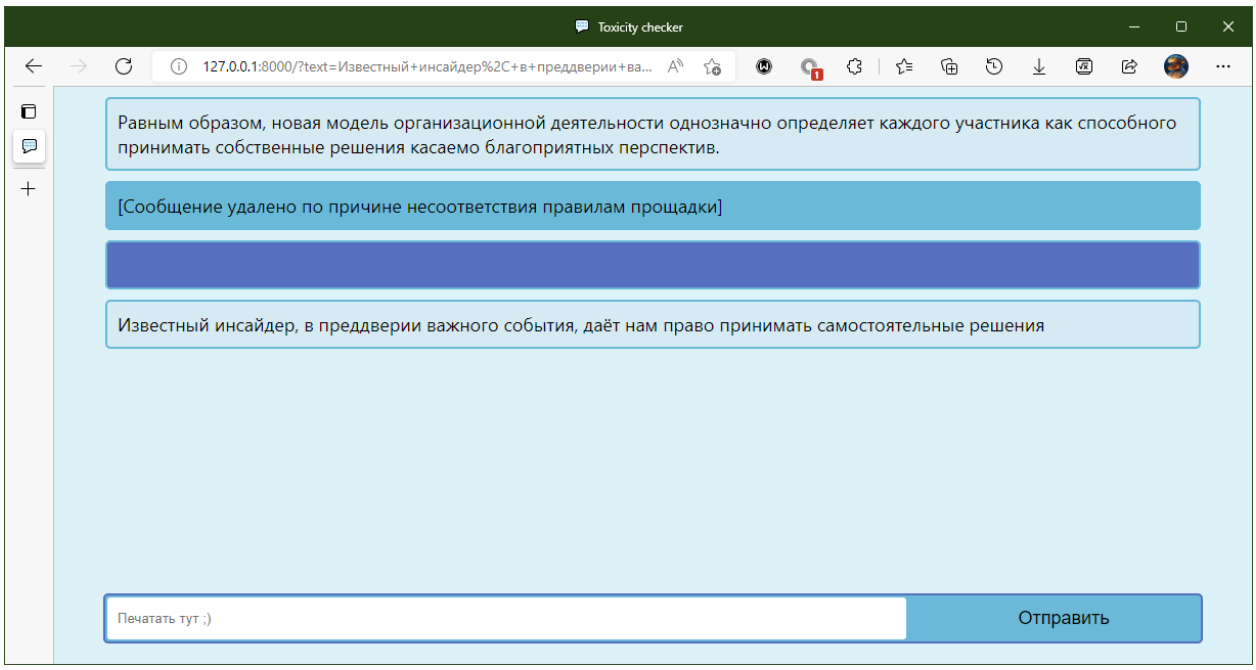


Рисунок 17 – Страница приложения после ввода поста

Как мы видим этот текст является нейтральным и отображается без изменений. Теперь попробуем ввести менее однозначный текст. Такого содержания: «Гореть тебе в аду». Результат на рисунке 18.

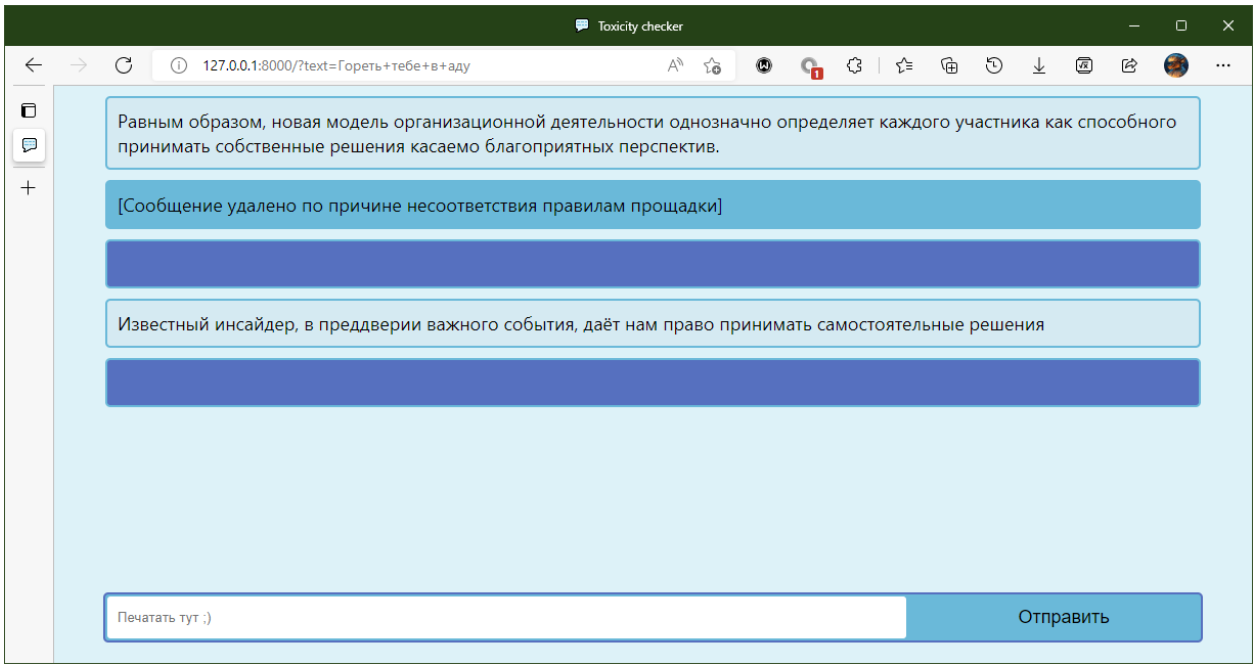


Рисунок 18 – Страница приложения после ввода второго сообщения

Как мы можем видеть текст скрыт, если навести мышь на текст он покажется. На рисунке 19 мышь наведена на последний текстовый блок.

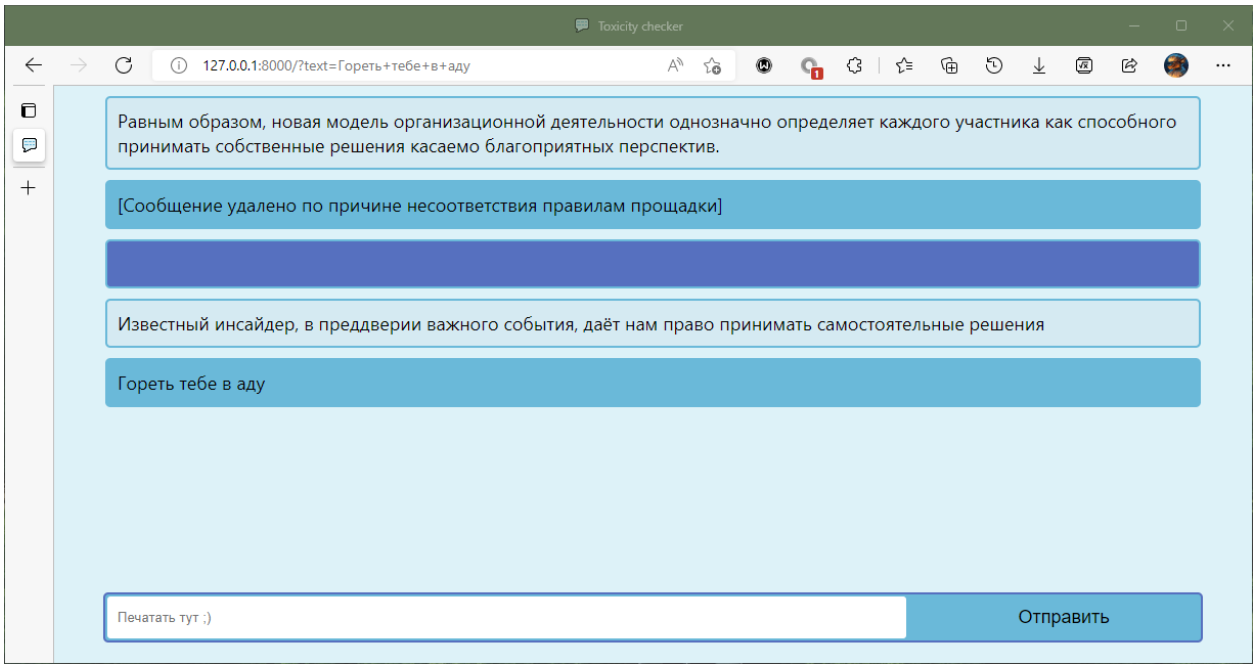


Рисунок 19 – Наведение мыши на скрытый текст

Окончательно для теста введём, что-нибудь очень непристойное. Итоговая страница изображена на рисунке 20

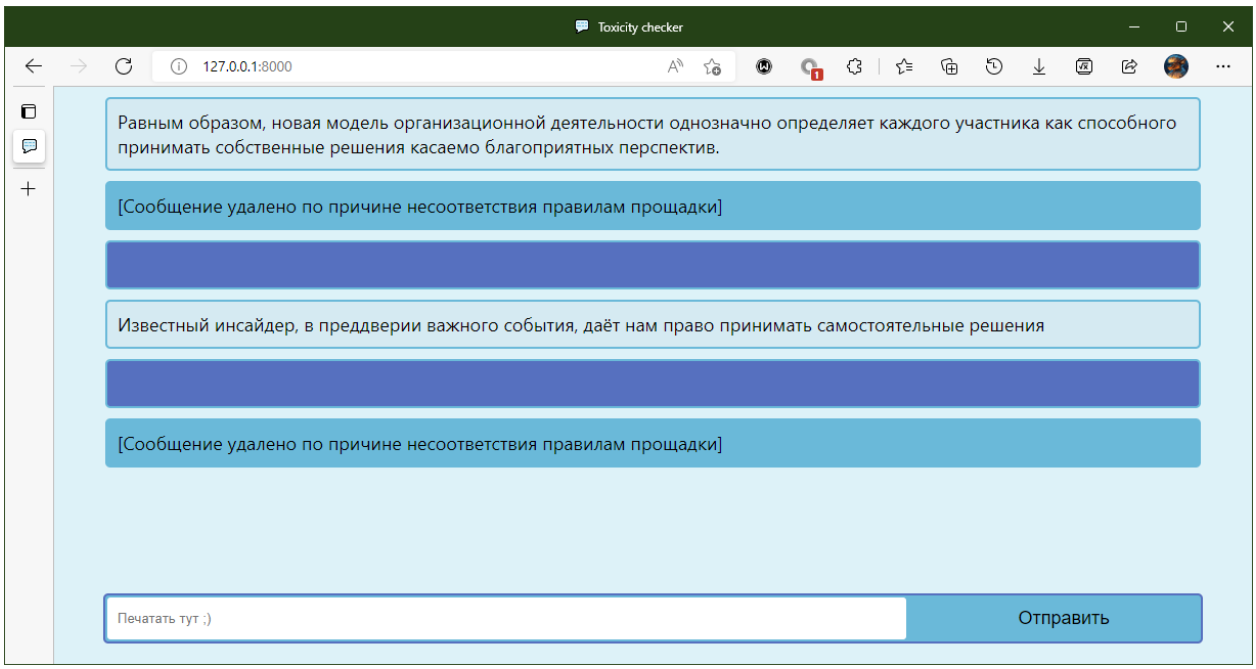


Рисунок 20 – Введено, не подходящего правилам площадки

Тестирование демонстрационного приложение можно считать оконченным.

### 3.2 Синтетические тесты

Подсчитаем точность модели с помощью ROC-кривой (receiver operating characteristic, рабочая характеристика приёмника). ROC является популярным графиком для одновременного отображения компромисса между истинной положительной частотой и ложноположительной частотой для двоичного классификатора при разных порогах классификации. Кривая ROC восходит ко Второй мировой войне, когда она первоначально использовалась для анализа радиолокационных сигналов, а затем для обнаружения сигналов.

Общая производительность классификатора, суммированная по всем возможным пороговым значениям классификации, определяется площадью под кривой ROC, или AUC (Area Under Curve, площадь под кривой). Идеальная кривая ROC будет охватывать верхний левый угол, указывая на высокую частоту истинных положительных результатов и низкую частоту ложных положительных результатов; чем больше AUC, тем лучше классификатор. AUC представляет вероятность того, что модель ранжирует случайный положительный случай выше, чем случайный отрицательный случай. AUC колеблется от 0 до 1, где AUC, равный 0, указывает на то, что модель ошиблась во всех прогнозах, в то время как AUC, равный 1, указывает на то, что модель получила все прогнозы правильно [11][20][21].

Так как перед нами не стоит задачи разделять разные виды настроений в тексте, кроме нейтрального/не нейтрального, поэтому посчитаем площадь под кривой ROC для результирующего ответа. На тестовой выборке из датасета на котором происходило дообучение. Полученная кривая изображена на рисунке 21.

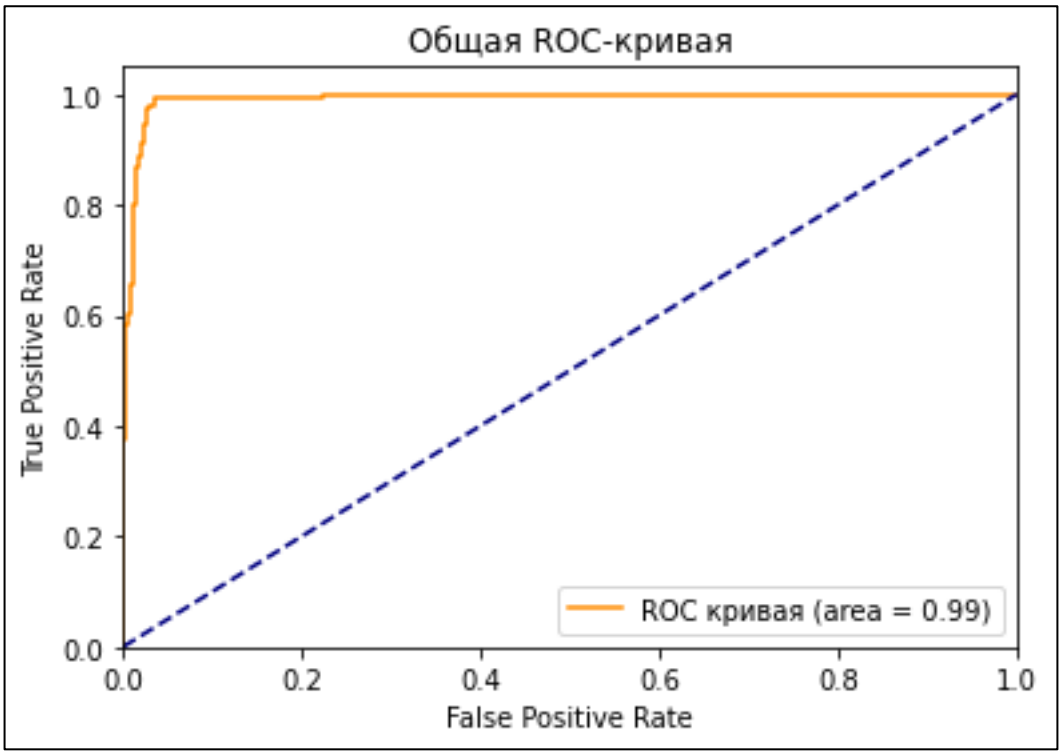


Рисунок 21 – ROC-кривая

Значение AUC ROC для нашей модели равно 0.991. Это очень высокий показатель, скорее всего это обусловлено однородностью датасета на котором производилось дообучение. Так же проверим значения AUC ROC для разных метрик. Получившиеся значения представлены в графике на рисунке 22.

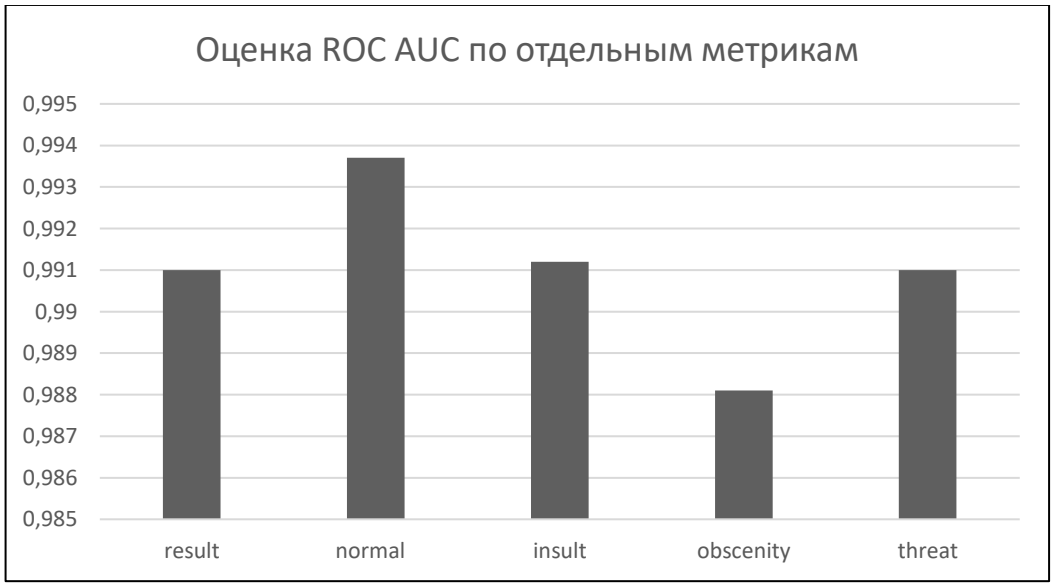


Рисунок 22 – Значение AUC ROC для различных метрик

По полученным данным, мы видим, что нейтральный текст модель различает лучше всего, у него наивысшее значение 0,9937. Показатели AUC ROC на удивление высоки, это означает, что с задачами аналогичными представленным в датасете модель справляется хорошо, но может требовать доработки для специфичных примеров или сообществ.

Вывод по разделу 3

В данном разделе было протестировано демонстрационное приложение, а также проведены синтетические тесты обученной нейросети. Тесты демонстрационного приложения показали корректную работу приложения, а по результатам синтетических тестов можно судить, что на имеющихся данных модель работает корректно, с высокими показателями.

# Заключение

В выпускной квалификационной работе были рассмотрены и проанализированы технологии кластеризации и классификации текстовых данных с применением text mining по отношению к социальным сетям для классификации постов по эмоциональному окрасу и выявлению негативно окрашенных текстов оскорбительного, навязчивого или угрожающего содержания.

В ходе работы был разработан метод анализа и кластеризации текста по эмоциональному окрасу, призванный помечать агрессивные и оскорбительные тексты. А также было разработано демонстрационное приложение для наглядного тестирования разработанного метода, также демонстрирующее возможности применения разработанного метода для автоматической модерации постов.

Метод и приложение были реализованы на современном языке программирования Python с использованием библиотеки PyTorch и веб-фреймворка Django, на основе нейронной сети BERT. Хранение данных осуществляется с помощью базы данных PostgeSQL. Данный метод может быть использован как часть крупного проекта или как отдельный микросервис. А для хранения данных может быть использована любая другая система управления базами данных.

Теоретический анализ литературы позволяет выделить перспективное направление разработки систем анализа текстов по эмоциональному окрасу. Эта работа будет полезна при создании нового социального пространства свободного от агрессивного общения и оскорблений.

А также сама работа представляет собой отличный опыт для меня как для будущего специалиста, благодаря современности и универсальности затронутых в ней технологий и методов. Опыт, полученный во время выполнения данной работы, может и будет применён в моей будущей профессиональной деятельности.

# Список используемой литературы


1. Барский. А. Б. Нейронные сети: распознавание, управление, принятие решений. М.: Финансы и статистика, 2007. 174 с.
2. Васильев А. Н., Тархов Д. А. Принципы и техника нейросетевого моделирования Москва: Гостехиздат, 2015. 334 с.
3. Ваш первый BERT: иллюстрированное руководство. [Электронный ресурс] URL: https://habr.com/ru/post/498144/ (дата обращения 14.05.2022).
4. Вороновский Г. К., Махотило К. В., Петрашев С. Н., Сергеев С. А. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. Х.: ОСНОВА, 1997. 112 с.
5. Галушкин А. И. Нейрокомпьютеры. Учебное пособие. М.: Альянс, 2014. 528 с.
6. Гелиг А. Х., Матвеев А. С. Введение в математическую теорию обучаемых распознающих систем и нейронных сетей. Учебное пособие: моногр. М.: Издательство СПбГУ, 2014. 224 с.
7. Дронов В. Django: практика создания Web-сайтов на Python М.: БХВ-Петербург, 2016. 707 с.
8. Заенцев И. В. Нейронные сети: основные модели. Воронеж: Изд-во Воронежского госуд. ун-та, 1999. 76 с.
9. Кластеризация и классификация больших Текстовых данных с помощью машинного обучения на Java. [Электронный ресурс] URL: https://habr.com/ru/post/526984/ (дата обращения 16.05.2022).
10. Клименко Р. Веб-мастеринг на 100%. М.: Питер, 2015. 614 с.
11. Круглов В. В., Борисов. В.В. Искусственные нейронные сети. Теория и практика. М.: Горячая линия - Телеком, 2002. 382 с.
12. Латыпова Р. Нейронные сети. М.: LAP Lambert Academic Publishing, 2012. 465 с.
13. Лутц М. Программирование на Python. Том 2. М.: Символ-плюс, 2013. 334 с.

14. Осовский. С. Нейронные сети для обработки информации. М.: Финансы и статистика, 2004. 343 с.
15. Редько В. Г. Эволюция, нейронные сети, интеллект: Модели и концепции эволюционной кибернетики. Москва: Наука, 2017. 224 с.
16. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечёткие системы. М.: OCHOBA, 1997. 112 с.
17. Рышард Т. Элементарное введение в технологию нейронных сетей с примерами программ. М.: Горячая линия - Телеком, 2011. 661 с.
18. Уоссермен. Ф. Нейрокомпьютерная техника: теория и практика. М.: Мир, 1992. 236 с.
19. Форсье Д. Django. Разработка веб-приложений на Python. М.: Символ-плюс, 1979. 326 с.
20. Хайкин. С. Нейронные сети. Полный курс. 2-е изд. Пер. с англ. М.: Издательский дом «Вильямс», 2006. 1104 с.
21. Haykin S. Neural networks: a comprehensive foundation. Upper Saddle River, N.J.: Prentice Hall, 1999. 842 с.
22. Lutz M. Programming Python: Powerful Object-Oriented Programming, Fourth Edition. New York City: Shroff, 2011. 1652 с.
23. Python 3.10.4 documentation. [Электронный ресурс] URL: https://docs.python.org/3/ (дата обращения 10.05.2022).
24. Ravichandiran S. Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT. Birmingham: Packt Publishing, 2021. 352 с.
25. Sabharwal N., Agrawal A. Hands-on Question Answering Systems with BERT: Applications in Neural Networks and Natural Language Processing. New York City: Apress, 2021. 184 с.