Казахский Национальный Университет имени аль-Фараби

СҰЛТАН ДАНИЯР РАХМАНҚҰЛҰЛЫ

Обнаружение и предотвращение кибербуллинга в онлайн-пользовательском контенте

8D06301 – Системы информационной безопасности

Диссертация на соискание степени Doctor of Philosophy (Ph.D.)

Отечественный научный консультант:
Омаров Батырхан Султанович, PhD,
и.о. доцента, Алматы, Казахстан

Зарубежный научный консультант:
Mateus Mendes, Doctor of Philosophy in
Electrical Engineering and Computer Science,
Polytechnics Instituto Politécnico de Coimbra,
Коимбра, Португалия

Республика Казахстан

Алматы, 2023

СОДЕРЖАНИЕ

НОРМАТИВНЫЕ ССЫЛКИ 4
ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 5
ВВЕДЕНИЕ 6
1. ОБЗОР ЛИТЕРАТУРЫ 10
1.1 Классификация кибербуллинга 12
2. МАШИННОЕ ОБУЧЕНИЕ ДЛЯ ВЫЯВЛЕНИЯ КИБЕРБУЛЛИНГА 18
2.1 Сбор данных 18
2.2 Извлечение признаков 19
2.3 Проектирование свойств 19
2.4 Выбор параметров 23
2.5 Применение алгоритмов машинного обучения 24
2.6 Работа с данными. 25
2.7 Показатели оценки 27
3 ПОДХОДЫ ГЛУБОКОГО ОБУЧЕНИЯ ДЛЯ ВЫЯВЛЕНИЯ КИБЕРБУЛЛИГА 29
1.1 Рекуррентные нейронные сети в задаче выявления кибербуллинга 30
1.2 Использование простых глубоких нейронных сетей в задаче выявления кибербуллинга 31
1.3 Использование сверточных нейронный сетей в задаче классификации текстов 32
1.4 Использование сетей с кратковременной долговременной памятью для выявления кибербуллинга 33
1.5 Использование многослойного персептрона в задаче классификации текстовых данных 34
1.6 Механизм внимания 37
4 МАТЕРИАЛЫ И МЕТОДЫ 40
4.1 Построение парсера. 40
4.2 Очистка и фильтрация данных 43
4.2.1 Фильтрация данных 43
4.2.2 Обработка пропущенных значений 44
4.2.3 Работа с дубликатами и опечатками 44
4.3 Создание простых глубоких нейронных сетей для выявления кибербуллинга 44

4.4 Имплементация рекуррентных нейронных сетей для выявления кибербуллинга 46
4.5 Создание Сверточных нейронных сетей для выявления кибербуллинга 48
4.6 Нейронные сети с долговременной кратковременной памятью (LSTM) 50
4.7 Двунаправленные нейронные сети с долговременной кратковременной памятью(Bi-LSTM) 52
4.8 Создание многослойного персептрона для выявления кибербуллинга 53
4.9 Создание глубокой нейронной сети с использованием механизма внимания 54
4.10 Итоговая модель нейронной сети с использованием механизма внимания 55
4.10.1 Первичная обработка сырого текста 56
4.10.2 Выборка параметров 57
4.10.3 Создание и обучение нейронной сети 58
5 РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ 62
5.2 Результаты обучения рекуррентных нейронных сетей 63
5.3 Результаты использования глубоких нейронные сети 64
5.4 Результаты внедрения сверточных нейронных сетей в выявлении кибербуллинга 65
5.5 Результаты внедрения моделей кратковременной долговременной памяти в задаче классификации 67
5.6 Результаты внедрения многослойного персептрона 69
5.7 Результаты внедрения глубокой нейронной сети с механизмом внимания 70
5.18 Сравнительный анализ результатов исследования 71
6. ВЫВОДЫ И ОБСУЖДЕНИЕ 74
6.1 Постановка исследовательского целя 74
6.2 Ключевые достижения 74
ЗАКЛЮЧЕНИЕ 76
СПИСОК ЛИТЕРАТУРЫ 77

НОРМАТИВНЫЕ ССЫЛКИ

В данной диссертации использовались ссылки согласно следующим стандартам:

– Закон Республики Казахстан «О науке» от 18.02.2011 г. № 407-IV ЗРК.

– ГОСО РК 5.04.034-2011 «Государственный общеобязательный стандарт образования Республики Казахстан. Послевузовское образование. Докторантура». Основные правила заверены Министерством Образования и Науки РК. «17» июнь 2011 ж. №261. Астана 2011.

«Инструкция по оформлению диссертационных работ и авторефератов, МОН РК, Комитет высшей аттестации, Алматы, 2004, МЕСТ 7.1-2003. Библиографическая опись.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

CPU – Central Processing Unit
GPU – Graphical Processing Unit
NLP – Natural Language Processing
SA – Sentiment Analysis
ML – Machine learning
DL – Deep learning
NN – Neural networks
DNN – Deep neural networks
CNN – Convolutional neural networks
RNN – Recurrent neural networks
AUC – Area under curve
ROC – Receiver operating characteristics
NB – Naïve Bayes
LR – Logistic regression
SVM – Support vector machine
KNN – k-nearest neighbors
DT – Decision tree
RF – Random forest
LSTM – Long short-term memory
TF – Term frequency
IDF – Inverse document frequency
SMOTE – Synthetic minority oversampling technique
API – Application programming interface
TP – True positive
TN – True negative
FP – False positive
FN – False negative

ВВЕДЕНИЕ

Актуальность темы исследования. Кибербуллинг - это растущая проблема, которая может иметь серьезные потенциальные проблемы как для жертв, так и для преступников. Это относится к использованию цифровых инструментов, таких как интернет-платформы, социальные сети и мобильные телефоны, для преследования, запугивания или другими способами причинения вреда людям. Кибербуллинг может принимать различные формы, включая публикацию злых или угрожающих сообщений, распространение слухов, обмен постыдным фото-, видео-контентом или исключение кого-либо из социальных групп онлайн.

Последствия кибербуллинга могут быть значительно пагубными. У пострадавших людей могут проявляться симптомы тревоги, депрессии, снижения самооценки и суицидальных мыслей. 75% детей из Казахстана сталкивались с разного рода угрозами в интернете. Это выяснила антивирусная компания ESET, опросив казахстанских интернет-пользователей накануне летних каникул. Кроме того, они могут столкнуться с проблемами со сном, аппетитом, концентрацией внимания и академическим или межличностным функционированием. В некоторых случаях кибербуллинг может привести к физическим последствиям, поскольку объект может испытывать изоляцию или воспринимать себя в опасности. В 2020 году 143 подростка совершили суицид, а 306 несовершеннолетних совершили попытку суицида из за травли в интернет-пространстве.

Люди, которые совершают кибербуллинг, также могут пострадать от негативных последствий. Кибербуллинг менее распространён, чем буллинг. В целом, 5% подростков становились жертвами кибербуллинга или сами принимали участие в кибербуллинге других людей 2-3 раза и более в месяц. Хотя бы один раз подвергались кибербуллингу 12% подростков. Они могут столкнуться с юридическими последствиями, если их действия будут признаны преступными, а также с социальными и профессиональными последствиями, такими как ущерб их репутации, трудности с приобретением друзей или поиском работы. Согласно статистике, больше трети респондентов, которые сталкивались с кибербуллингом, получали оскорбления в личных сообщениях, 24% видели неприятные посты о себе, а почти треть - 31% - читали неприятные комментарии под своими фото.

Одним из способов уменьшить кибербуллинг является информирование людей о последствиях их действий. Это можно сделать с помощью кампаний по информированию общественности, образовательных программ и ресурсов для родителей и учителей. Повышая осведомленность о последствиях кибербуллинга, люди, возможно, с большей вероятностью подумают, прежде чем прибегать к подобному поведению.

Еще одним способом уменьшить кибербуллинг является предоставление поддержки и ресурсов жертвам. Предоставляемая поддержка и ресурсы могут включать консультирование, терапию, а также механизмы для сообщения о случаях кибербуллинга и устранения их последствий. Благодаря этим инструментам жертвы могут чувствовать себя более уверенно и иметь возможность действовать и обращаться за помощью.

Также важно привлекать комитеты к ответственности за свои действия. Это может включать в себя дисциплинарное взыскание с учащихся, которые занимаются кибербуллингом, или возбуждение судебного иска в случаях, когда такое поведение считается преступным. Привлекая виновных к ответственности, это дает понять, что кибербуллинг недопустимо и может иметь серьезные последствия.

Наконец, важно создать культуру уважения и доброты в наших сообществах. Этого можно добиться, поощряя позитивное поведение, такое как доброта и сопереживание, и моделируя это поведение самостоятельно. Создавая культуру уважения и доброты, мы можем создать более инклюзивную и гостеприимную среду для всех.

Цель диссертационной работы - Построение модели глубокой нейронной сети для автоматического обнаружения кибербуллинга в текстовых данных.

Задачи исследования. Для достижения цели исследования были решены следующие задачи:

1. Анализ алгоритмов машинного обучения для задачи бинарной и многоклассовой классификации для обнаружения кибербуллинга.
2. Сбор данных и предварительная обработка данных на казахском языке для обучения алгоритмам ML и DL для выполнения диссертационной работы.
3. Анализ архитектур глубокого обучения. Обучающая реализация различных типов алгоритмов DL, таких как:
   a) Сверточные нейронные сети
   b) Глубокие нейронные сети
   c) Рекуррентные нейронные сети
   d) Сети с кратковременной долговременной памятью
   e) Многослойные персептроны
   f) Глубокие нейронные сети с использованием механизма внимания
   g) Проведение экспериментальных исследований, сравнение, выбор модели, настройка гиперпараметров для улучшения результатов модели.

Объект исследования: социальные сети (Vkontake, Instagram, Youtube, Twitter), новостные проталы(nur.kz, tengri news).

Предмет исследования: Алгоритмы машинного обучения и глубокого обучения для обнаружения кибербуллинга в текстовых данных.

Теоретическая значимость исследования: исследование существующих работ по выявлению кибербуллинга в текстовых данных, анализ инструментов обработки естественного языка.

Практическая значимость исследования: разработка и обучение глубоких нейронных сетей, обработка естественного языка, проведение экспериментов по выявлению кибербуллинга в социально-медийном пространстве.

Научная новизна проведенных исследований:

– Создан языковой набор данных, предварительно обработанный и помеченный вручную для дальнейших задач машинного и глубокого обучения.

– Разработана и обучена глубокая нейронная сеть с механизмом внимания для задачи двоичной и трех-классовой классификации в задаче выявления кибербуллинга.

– Был проведен сравнительный анализ между машинным обучением и алгоритмами глубокого обучения в задаче выявления кибербуллинга.

Результаты диссертации были представлены в 5 научных работах, из них 2 статьи в журналах, рецензируемых в базе Scopus, 2 статьи – в материалах международных конференций, 1 статья – обзорная статья обзорная статья в журнале, рецензируемый в базе Scopus:

1) D. Sultan *et al*, “A Review of Machine Learning Techniques in Cyberbullying Detection”, *Computers, Materials & Continua*, vol. *74, no*. 3, pp. 5625-5640. https://doi.org/ 10.32604/cmc.2023.033682.

2) D. Sultan *et al*, “Cyberbullying-related hate speech detection using shallow-to-deep learning,” *Computer Materials & Continua*, vol. *74, no.* 11, pp. 2115-2131. https://doi.org/10.32604/cmc.2023.033682.

3) M. Anand, K. Bhushan Sahay, M. Altaf Ahmed, D. Sultan, R. Raman Chandan, B. Singh, "Deep learning and natural language processing in computation for offensive language detection in online social networks by feature selection and ensemble classification techniques," Theoretical Computer Science, v. 943, 2023, pp. 203-218, https://doi.org/10.1016/j.tcs.2022.06.020.

4) D. Sultan *et al.*, "Cyberbullying Detection and Prevention: Data Mining in Social Media," *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2021, pp. 338-342, https://doi.org/10.1109/Confluence51648.2021.9377077.

5) D. Sultan *et al.*, "Cyberbullying and Hate Speech Detection on Kazakh-Language Social Networks," *2021 7th IEEE Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart*

Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), NY, USA, 2021, pp. 197-201, https://doi.org/10.1109/BigDataSecurityHPSCIDS52275.2021.00045.

Объем и структура работы. Диссертационная работа состоит из 82 страниц и включает 34 рисунка и 15 таблиц. Содержание включает 6 разделов.

Введение. В данном разделе дано описание актуальности, новизны и основной цели диссертационной работы. Был приведен список основных задач и объекта и предмета исследования, а также теоретической и практической значимости исследования.

Первый раздел дает определение кибербуллинга, обзор на похожие проделанные работы в области выявления кибербуллинга. Представлен обзор результатов других авторов и инструментов обработки естественного языка.

Во втором заделе дается полное описание алгоритмов машинного обучения, теоритическое обоснование осоновных алгоритмов, а также их математическое обоснование. В разделе также подробно описана исследовательская часть работы, которая затрагивает бинарную и многоклассовую классификацию в текстовых данных для выявления кибербуллинга. Сводная информация приведена в виде графиков, таблиц и выводов.

В третьем разделе дается полное описание алгоритмов глубокого обучения, с приведением примеров и их строений. Также в разделе математически описаны все возможные виды слоев при создании нейронных сетей; структура механизма внимания разобрана с точки зрения математических формул и его использованием для классификации текста.

В четвертом разделе описаны проведенные работы по теме исследовательской диссертации. В первую очередь показан и разобран процесс создания парсера для сбора данных в социальных сетях и онлайн-пользовательском контенте, ручной классификации данных, первичной обработки и инструментов начальной обработки сырых данных, такие стемминг, лемматизация, удаление стоп слов и др. Далее приведены примеры использования алгоритмов машинного обучения в обработанных данных и готовых данных на примере использования набора данных Twitter. Далее приведены примеры и коды создания нейронных сетей. В конце раздела приведена предлагаемая модель с полным описанием вышеуказанных процессов.

В пятом разделе представлены результаты имплементированных алгоритмов машинного обучения и глубокого обучения. Приведены сводные результаты всех экспериментов. Также приведена таблица и диаграммы, показывающие прирост алгоритма долговременной кратковременной памяти с использованием механизма внимания по отношению к остальным методам.

В заключении обобщены практические результаты данной диссертационной работы, приведены ее самые значимые достижения в

выявлении кибербуллинга в текстовом контенте с использованием алгоритмов машинного и глубокого обучения.

Личный вклад исследователя. В результате работы был выполнен подробный анализ существующих методов обработки естественного языка. Был полностью разработан парсер для сбора данных с медиа пространства, проведена ручная классификация собранных данных, проведен процесс первичной обработки сырых данных для работы с алгоритмами машинного и глубокого обучения.

1. ОБЗОР ЛИТЕРАТУРЫ

В этом разделе приведены примеры схожих работ, связанные с темой диссертации. Следует отметить, что некоторые специалисты показали гораздо лучшие результаты по английскому, арабскому и другим популярным языкам, но существует меньше исследований, посвященных непопулярным языкам. Кроме того, в нашей теме есть три варианта, где мы протестировали наши модели и архитектуры для трех различных наборов данных: английского, русского и казахского; набор данных для казахского языка был собран нами самими.

Современное пространство повседневного общения характеризуется новой поразительной особенностью - его распространением в виртуальный мир. Если для современных взрослых навыков общения с использованием социальные сети, мгновенных сообщений и чатов являются дополнением к уже приобретенным навыкам живого общения, то современные дети и подростки овладевают обоими этими навыками практически одновременно. Что касается подростков, то можно сказать, что процесс социализации во многом перемещается в Интернет – вместе со знакомствами, связанными группами, освоением различных социальных ролей и норм [1]. Все те коммуникативные процессы, которые происходят в обычном социофизическом пространстве, "дублируются", иногда усиливаются, а иногда компенсируются виртуальным общением, но в любом случае приобретают новые черты. Хотя исторически виртуальное существование явно вторично по отношению к реальному, можно ожидать обратного воздействия и переноса коммуникативных ситуаций и правил, распространенных в Интернете, в “реальное” пространство общения [2].

С развитием информационных технологий в жизни современного подростка произошли значительные изменения: появилась виртуальная реальность, в которой общение и межличностные отношения переходят на новый, незнакомый для них уровень. Запугивание становится более опасным для отдельного человека, поскольку оно может осуществляться с использованием интернет-технологий [3].

Впервые определение "кибербуллинга" было дано Биллом Белси. По его мнению, кибербуллинг - это использование информационно-

коммуникационных технологий, например, электронной почты, мобильного телефона, персональных интернет-сайтов, для преднамеренного, повторяющегося и враждебного поведения человека или группы, направленного на оскорбление других людей [4].

Издевательства в Интернете могут осуществляться 24 часа в сутки, 7 дней в неделю, не оставляя шанса почувствовать себя защищенным, сообщения и комментарии могут приходить неожиданно, в любое время – это оказывает сильное психологическое воздействие на подростка [5]. В Интернете также существует анонимность, благодаря которой подросток может даже не подозревать, что за человек над ним издевается, что может вызвать у него еще больший страх. В отличие от физического насилия, последствия эмоционального насилия в долгосрочной перспективе влияют на психологическое здоровье [6]. Поэтому наша цель - изучить феномен кибербуллинга как формы подавления личности подростка и определить содержание профилактики кибербуллинга с использованием технологии машинного обучения (ML), которая предполагает постановку следующих задач: виды кибербуллинга в онлайн-контенте; определить методы и инструменты автоматического обнаружения кибербуллинга и выражений ненависти.; рассмотрим открытые наборы данных для обучения моделям машинного обучения для автоматического обнаружения кибербуллинга; выделим самые современные методы и проанализируем будущие тенденции.

В этом обзоре литературы были объединены и применены как качественные, так и количественные методы анализа [7-8]. Рис. На рис. 1 показаны этапы рецензирования и количество включенных и исключенных статей. Коллекция статей начинается с определения строки поиска. Эта строка состоит из трех поисковых запросов: “Машинное обучение” и “ Кибербуллинг” с логическим оператором “И" между ними. Были использованы четыре справочные базы данных: платформа Science Direct, цифровая библиотека IEEE Xplore, Springer, онлайн-библиотека Wiley. Мы включили в наше исследование статьи, опубликованные в период с 2015 по 2021 год. На этапе отбора мы исключили десять записей. На этапе отбора мы исключили восемь записей без полных текстов, вторичного анализа и дублированных публикаций. Из оставшихся 22 статей мы оставили 13 для мета-анализа.

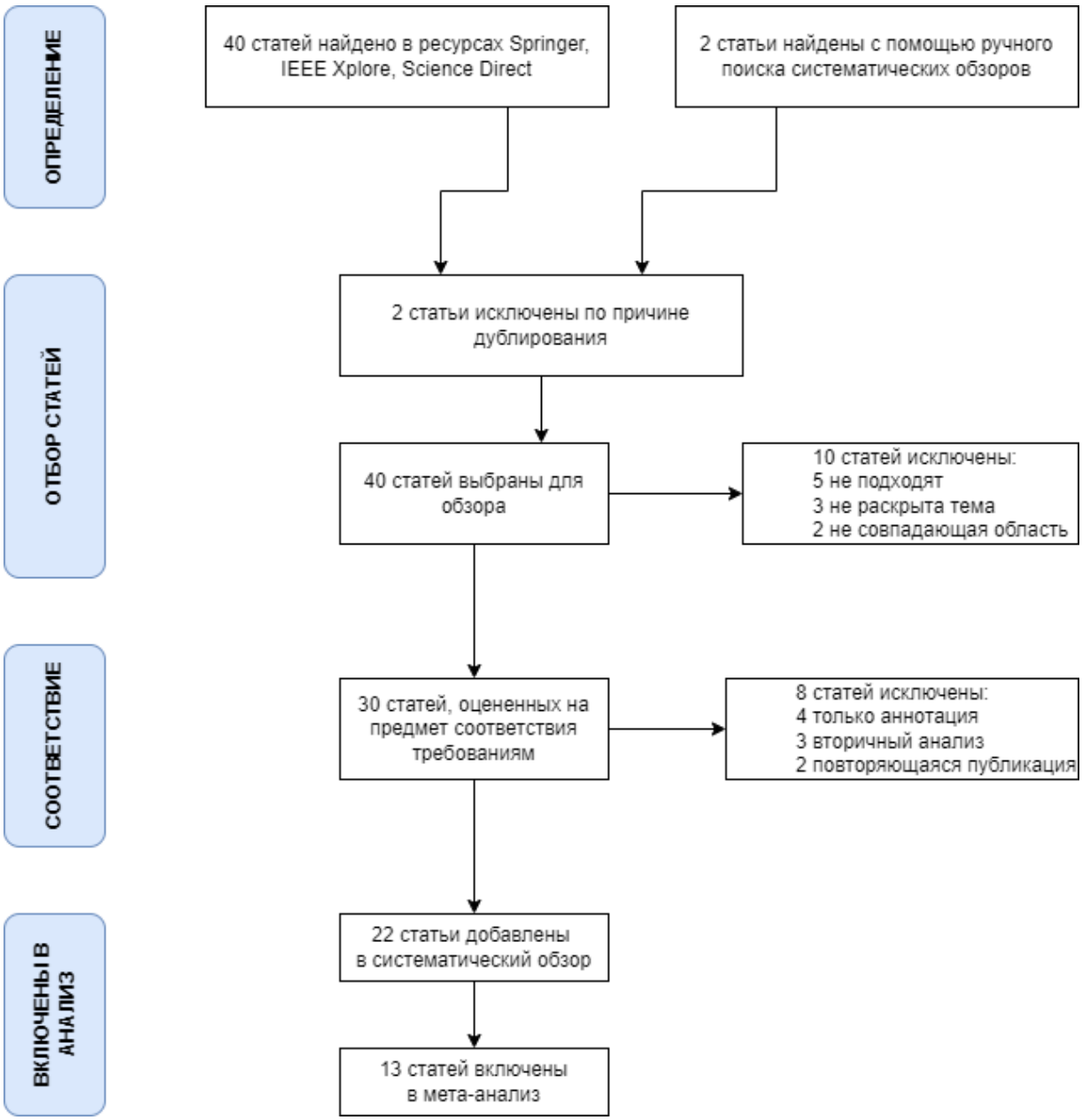


Рисунок 1.1 - Метод систематического обзора литературы

1.1 Классификация кибербуллинга

Цифровая драма - это выражение нью-эйдж, которое относится к формам жестокого обращения и жестокости среди подростков в мире технологий [9]. Определение кибербуллинга Национальным советом по предупреждению преступности таково: “когда Интернет, мобильные телефоны или другие устройства используются для отправки или размещения текста или изображений, предназначенных причинить боль или поставить в неловкое положение другого человека" [10]. StopCyberbullying.org экспертная организация, занимающаяся интернет-безопасностью и неприкосновенностью частной жизни, определяет кибербуллинг как: "ситуацию, когда ребенок, подросток или тинейджер неоднократно подвергается "пыткам, угрозам, домогательствам, унижению, смущению или иным образом становится

мишенью" со стороны другого ребенка или тинейджера с помощью текстовых сообщений, электронной почты, мгновенных сообщений или любого другого другой тип цифровой технологии”.

Кибербуллинг может быть таким же простым, как продолжение отправки электронного письма тому, кто заявил, что не желает больше контактировать с отправителем, но оно также может включать угрозы, сексуальные высказывания, уничижительные ярлыки (например, разжигание ненависти), объединение жертв, делая их предметом насмешек на форумах, и публикация ложных заявлений как факта, направленная на унижение.

Как и традиционное запугивание, кибербуллинг может быть прямым и косвенным [11]. Прямое кибербуллинг - это прямые атаки на ребенка посредством писем или сообщений [12]. В случае косвенного преследования в процесс преследования жертвы вовлекаются другие люди (как дети, так и взрослые), не всегда с их согласия; преследователь может взломать учетную запись жертвы и, подражая ведущему, отправлять сообщения с этой учетной записи друзьям жертвы, разрушая коммуникативное поле жертвы. и вызывающий сомнение в его моральных качествах [13]. Одна из наиболее угрожающих ситуаций - это когда преследователь публикует в сети информацию, которая подвергает жертву опасности, например, он размещает от ее имени объявление о поиске сексуальных партнеров. Как и традиционное запугивание, кибербуллинг включает в себя континуум действий, на одном полюсе которого действия, которые вряд ли признаются окружающими домогательствами, а на другом — насильственное поведение агрессора, которое может даже привести к смерти жертвы.

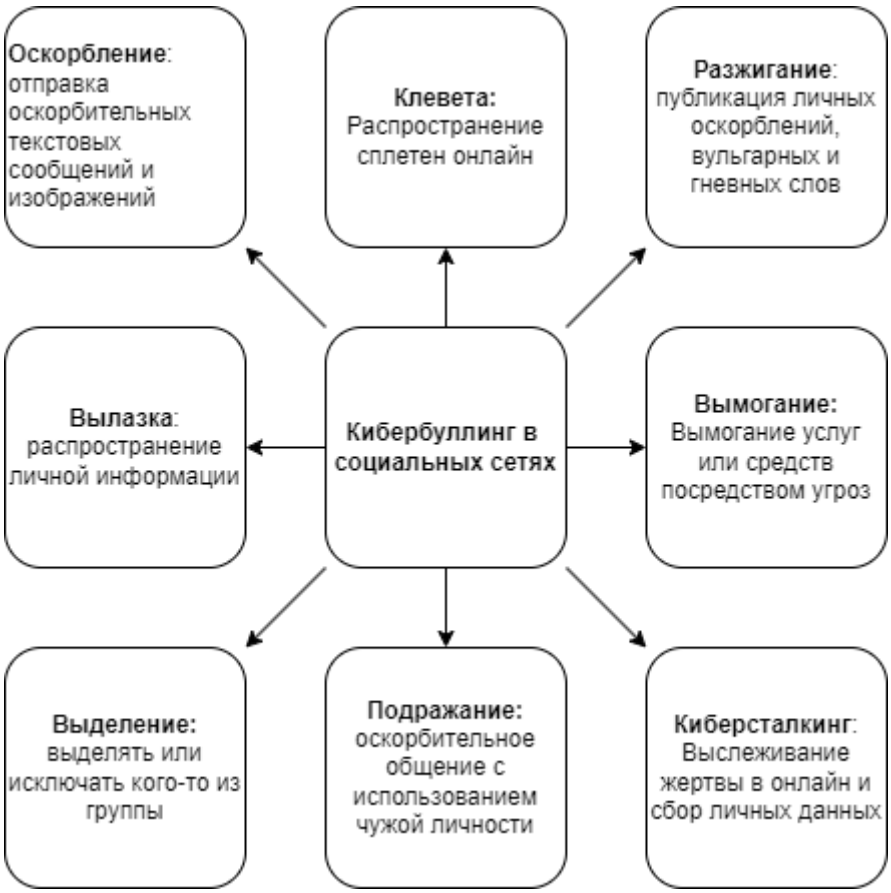


Рисунок 1.2 – Классификация типов кибербуллинга в интернет пространстве

Недавние исследования приводят следующие наиболее распространенные методы домогательств в электронном пространстве. Рисунок 1.2 демонстрирует виды кибербуллинга в социальных сетях.

Наиболее эмоционально жестокой формой кибербуллинга является флейминг, который начинается с оскорблений и перерастает в быстрый эмоциональный обмен репликами, обычно публично, реже в частной переписке [14]. Это происходит между двумя собеседниками с изначально равными позициями, но внезапная агрессия вносит дисбаланс, который усиливается тем фактом, что участник не знает, кого его противник может привлечь на свою сторону в этой битве. Посетители форума, свидетели, могут присоединиться к одной из сторон и завязать бурную переписку, не до конца понимая первоначальный смысл столкновения и часто рассматривая ситуацию как игру, в отличие от инициаторов агрессивного диалога. Вы можете сравнить это с боем "стенка на стенку", где участники не до конца понимают ни то, что послужило причиной конфликта, ни то, что является критерием присоединения товарищей по оружию друг к другу.

Похожей на флейминг, но однонаправленной формой издевательств является домогательство: обычно это настойчивые или повторяющиеся слова и действия, адресованные конкретному человеку, которые вызывают у него раздражение, тревогу и стресс и в то же время не имеют разумной цели [15].

Кибер-домогательства обычно выражаются в повторяющихся оскорбительных сообщениях жертве, от которых она чувствует себя морально уничтоженной, на которые она не может ответить из-за страха или неспособности идентифицировать преследователя, а иногда она также вынуждена платить за полученные сообщения [16].

Другой формой харрасмента является троллинг: кибер-тролли публикуют негативную, тревожащую информацию на веб-сайтах, страницах социальных сетей, даже на мемориальных страницах, посвященных умершим людям, вызывая сильную эмоциональную реакцию. Изначально термин "троллинг" относится к рыбной ловле и означает ловлю на блесну. "Настоящими" троллями обычно называют провокаторов — это те, кто использует "слабые места" других людей для того, чтобы с помощью манипуляций подразнить человека и получить удовольствие от его эмоционального взрыва. В этом случае агрессор испытывает чувство всемогущества из-за власти над жертвой, над ее эмоциональным состоянием.

Схожим по смыслу, но менее манипулятивным и более прямолинейно агрессивным является кибersталкинг — использование электронных коммуникаций для преследования жертвы посредством повторяющихся тревожных и раздражающих сообщений, угроз незаконных действий или ущерба, жертвами которых могут быть получатель сообщений или члены его семьи [17].

Кроме того, так называемые секстеты могут вызывать стыд, беспокойство или страх. Секстинг - это распространение или публикация фото- и видеоматериалов с обнаженными и полуобнаженными людьми [18-20]. Чем старше дети, тем выше вероятность их вовлечения в секстинг. Согласно исследованию, 10% молодых людей в возрасте 14-24 лет отправляли или публиковали свои изображения с сексуальным подтекстом, 15% получали такие сообщения непосредственно от кого-то другого [21]. Среди участников исследования Американской национальной кампании по предотвращению подростковой и нежелательной беременности 71% девочек и 67% мальчиков отправляли "секс-открытки" своим романтическим партнерам; 21% девочек и 39% мальчиков отправляли фотографии с сексуальным подтекстом людям, с которыми они хотели бы общаться. имеют романтические отношения; 15% юношей и девушек отправили их кому-то знакомому только через онлайн-общение [22]. Если одни люди рассылают такие сообщения как часть гармоничных отношений внутри пары, то другие преследуют цели домогательств и нанесения вреда, например, размещая фотографии обнаженной бывшей девушки в Интернете в качестве мести за болезненный разрыв отношений.

Другой формой домогательств в Интернете является распространение клеветы, называемой очернительством: это публикация и распространение унизительной и ложной информации о человеке, его искаженных изображений, в частности в сексуализированном и/или наносящем ущерб его репутации виде и т.д. [23]. Одной из форм клеветы являются "онлайн-слэм-буки". Slam books — это записные книжки, в которых одноклассники публикуют различные оценки и комментарии - "кто самая красивая девочка в классе", "кто одевается хуже всех" и т.д. Соответственно, "онлайн-сламбуки" - это сайты, созданные для развлечения, где одноклассники публикуют похожие оценки и комментарии, часто грубые и неприятные, например, и «Худшая пара в классе". Платформой для этого часто служат развлекательные сайты, ориентированные на студентов и школьников. Некоторые люди посещают их не для того, чтобы посплетничать и оставить комментарий, а просто для того, чтобы проверить, не стали ли они очередным объектом клеветы и злонамеренных развлечений знакомых [24].

Ложная информация также распространяется при выдаче себя за кого-то другого, что называется олицетворением. Преследователь, используя украденный пароль, рассылает негативную, жестокую или неадекватную информацию своим друзьям с аккаунтов жертвы и как бы от ее имени [25]. Жертва испытывает сильное унижение при получении обратной связи и часто теряет друзей. Кроме того, преследователь может использовать пароль для изменения личного профиля жертвы на веб-сайте, размещать там неуместную, оскорбительную информацию, отправлять угрожающие или унизительные электронные письма с адреса жертвы. В крайних случаях преследователь может размещать провокационные оскорбительные сообщения или комментарии на форумах, подписываясь именем жертвы и указывая ее настоящее имя, адрес и номер телефона, тем самым подвергая жертву риску реального преследования и нападения.

Раскрытие секретов и мошенничество, то есть выходка на улицу и обман, предполагает распространение личной, секретной, конфиденциальной информации о жертве в сети. Эта форма похожа на раскрытие секретов "в реальной жизни", которое также сопровождается чувством стыда и страха быть отвергнутым со стороны жертвы, и отличается только количеством возможных свидетелей [26].

Исключение из сообщества, к которому человек чувствует свою принадлежность, может переживаться как социальная смерть. Исключение/остракизм из онлайн-сообществ может произойти в любой среде, защищенной паролем, или путем удаления из "списка друзей". Эксперимент показал, что исключение из онлайн-сообщества снижает самооценку участника и способствует тому, что в следующем сообществе он начинает

вести себя более конформно [27]. Часто после исключения человек присоединяется к другим группам (в частности, тематически посвященным мести первому сообществу), и это позволяет ему частично справиться со своими переживаниями; множество "сообщников" вдохновляют этого человека и укрепляют веру в возможность отомстить за остракизм — самостоятельно или через других. с помощью членов новой группы. При отсутствии прямых оснований это аналог косвенного домогательства, которое выражается в изоляции и неприятии кого-либо из членов группы ("никто не хочет с ним сидеть", "мы с ней не дружим").

Важность для человека его признания со стороны сообщества также используется при публикации видеороликов о физическом насилии/хулиганском нападении (видеозапись нападений/радостных пощечин и прыжков). Happy slapping - хулиганское нападение группы подростков на прохожего, во время которого один из хулиганов снимает происходящее на видеокамеру мобильного телефона [28]. Чтобы усилить чувство унижения жертвы, преследователи размещают видеозапись нападения в Интернете, где тысячи зрителей могут посмотреть ее и прокомментировать. К сожалению, загрузить видео в Интернет гораздо проще, чем удалить его.

Таким образом, основными лейтмотивами домогательств в Интернете являются эксплуатация значимости связанного сообщества для жертвы (привлечение множества свидетелей в разы усиливает чувства стыда, страха, беспомощности и отвержения); бесконтрольное распространение любой (ложной, постыдной, конфиденциальной) информации; провокация гипертрофированных аффективных реакций. обратная связь от жертвы. Цель кибербуллинга - ухудшить эмоциональное состояние жертвы и/или разрушить ее социальные отношения.

2. МАШИННОЕ ОБУЧЕНИЕ ДЛЯ ВЫЯВЛЕНИЯ КИБЕРБУЛЛИНГА

Угрозы, возникающие в сетевой среде, естественным образом стимулируют развитие исследовательского аппарата для изучения их факторов, механизмов и последствий. Эмоциональная природа тех процессов, которые в основном определяют деструктивный характер воздействия негативных сетевых явлений на человека, ставит в приоритет создание инструментов автоматического анализа, позволяющих оценивать выраженность признаков аффективных состояний участников сетевого общения, то есть методов анализа настроений в широком смысле от этого слова. Инжир. 3 иллюстрирует обнаружение кибербуллинга в социальных сетях с использованием методов машинного обучения. На рисунке описан процесс обнаружения кибербуллинга - от сбора данных до классификации текстов о кибербуллинге. В следующих подразделах объясняется каждый этап на рис. 3.

2.1 Сбор данных

Большинство моделей классификации текстов на основе ML полагаются на данные в качестве ключевого компонента. С другой стороны, данные не имеют смысла, если они не используются для получения информации или выводов. Наборы данных для обучения и тестирования выбираются с использованием данных, собранных из социальных сетей. Основываясь на наблюдаемых случаях (помеченных данных), контролируемые модели стремятся предоставить компьютерные подходы для повышения точности классификации в определенных задачах [29]. Хорошая модель для данной задачи не должна ограничиваться выборками только в обучающем наборе [30] и, следовательно, должна содержать немаркированные фактические данные. Объем данных не имеет значения; важно то, точно ли полученные данные отражают активность веб-сайта в социальных сетях [31]. Данные, извлеченные из социальных сетей с использованием ключевых слов, ключевых фраз или хэштегов (например, [31-33]), или данные, извлеченные из социальных сетей с использованием профилей пользователей (например, [34-36]), являются двумя основными методами сбора данных в исследованиях по выявлению киберзапугивания (например, [37]). В разделе "Сбор данных" освещаются проблемы, связанные с различными методологиями сбора данных, и их влияние на производительность метода ML.

Рисунок 2.1 - Метод систематического обзора литературы

2.2 Извлечение признаков

Основная цель извлечения признаков - превратить текст из любой настройки в расписание ключевых слов, которое модель машинного обучения может легко проанализировать. Извлечение признаков также предоставляет информацию о текстах, например, о самой высокой частоте употребления фраз для каждой книги. В контролируемом ML требуется выбор релевантных ключевых слов и определение механизма кодирования этих ключевых слов. Эти поисковые запросы и ключевые фразы могут оказать значительное влияние на способность систем классификации получать оптимальный шаблон [39].

2.3 Проектирование свойств

Поддающийся количественной оценке атрибут задачи, за которым ведется наблюдение, называется признаком [31]. Основная цель построения векторов признаков - предоставить набор обучающих векторов алгоритмам ML, чтобы они могли научиться различать различные виды классов [32]. Успех или неудача большинства моделей ML определяется разработкой функциональных возможностей [33-34]. На успех или неудачу прогноза может повлиять несколько факторов. Характеристики, используемые для обучения модели, являются наиболее важными [35]. Эта работа занимает большую часть работы по разработке моделей прогнозирования киберзапугивания с использованием алгоритмов обучения [35-38]. В этом случае пространство ввода (т.е. характеристики и их комбинации, которые представлены в качестве входных данных для классификатора) должно быть хорошо спроектировано.

Во многих приложениях первым шагом на пути к созданию эффективного классификатора является предложение набора отличительных признаков, которые используются в качестве входных данных для ML-классификатора. Созданные человеком наблюдения могут быть использованы для генерации векторов признаков, которые зависят от того, как объекты связаны с вхождениями классов. Недавнее исследование киберзапугивания

[32-34] выявило связь между несколькими характеристиками, включая пол, возраст и характер пользователя, и распространенностью киберзапугивания. Эти данные могут быть преобразованы в полезную форму, которая позволяет классификатору различать киберзапугивание и нейтральный текст, что позволяет разрабатывать успешные модели обнаружения киберзапугивания. Предложение характеристик является важным шагом к усилению способности классификационных моделей к различению [35-36]. Аналогичным образом, построение успешных классификационных моделей на основе алгоритмов ML требует предоставления набора релевантных характеристик вовлеченности в киберзапугивание на сайтах социальных сетей [37].

На основе передовых исследований были созданы новые характеристики, повышающие точность прогнозирования киберзапугивания. Например, была предложена лексико-синтаксическая функция для прогнозирования ненормативной лексики; эта стратегия более точна, чем классические методы, основанные на обучении [38]. Ахмед и др. использовал наборы данных из Myspace для построения гендерной методологии прогнозирования киберзапугивания на основе гендерной информации из информации профиля. Гендерный фактор был выбран для повышения способности классификатора к различению. Другие исследования [39-40] включали возраст и пол в качестве переменных, однако эти переменные ограничены информацией, предоставленной отдельными лицами в их онлайн-профилях.

В нескольких исследованиях [35, 38, 40-41] рассматривалось предсказание киберзапугивания с использованием нецензурных выражений в качестве характеристики. Аналогичным образом, чтобы сигнализировать об издевательствах, был создан словарь ненормативных фраз, и эти слова использовались в качестве функций для алгоритмов машинного обучения [33, 37]. Использование нецензурных выражений в качестве функций значительно повышает производительность модели. В предыдущей работе [32] количество "плохих" слов и плотность "плохих" терминов были предложены в качестве функций для ввода данных машинного обучения. Согласно полученным результатам, доля ‘плохих" терминов в сообщении указывает на киберзапугивание. Другое исследование [34] создало характеристики запугивания, расширив список предопределенных непристойных терминов и присвоив им различные веса. Эти характеристики были объединены с набором слов и скрытыми семантическими характеристиками и отправлены в алгоритм машинного обучения в качестве входных данных функции.

С точки зрения построения вектора признаков контекстно-ориентированный метод превосходит метод, основанный на списке [37]. С другой стороны, разнообразие и сложность киберзапугивания, возможно, не обязательно подтверждают этот вывод. В нескольких исследованиях [33, 39, 41] рассматривалось, как анализ настроений может помочь классификатору отличать сообщения о киберзапугивании от сообщений, не связанных с киберзапугиванием. В ходе этих исследований была выдвинута гипотеза о том, что особенности настроения являются хорошим показателем

распространенности киберзапугивания. Исследователи предложили модель для идентификации и ассоциирования профилей троллей в Twitter в исследовании, целью которого было создание способов снижения активности киберзапугивания путем обнаружения профилей троллей; они предположили, что обнаружение профилей троллей является важным шагом на пути к выявлению случаев киберзапугивания в социальных сетях [38]. В этом исследовании предлагаются элементы, основанные на содержании твита, времени публикации, языке и местоположении, чтобы улучшить идентификацию автора и оценить, является ли профиль троллем или нет. Ссылка [39] объединила характеристики структуры веб-сайтов SM (близость, степень, взаимосвязь и центральность собственных векторов, а также коэффициент кластеризации) с характеристиками пользователей (например, пол, возрастное поведение) и контента (настроения, длина). Точность машинного обучения повышается при сочетании этих характеристик. Сравнение различных факторов, используемых в литературе по выявлению киберзапугивания, приведенное в табл. 1, демонстрирует, как эти факторы влияют на точность прогнозирования. На вкладке. 1. мы разделили пространство функций на две части: функции, основанные на контенте, и функции, основанные на профиле. Функции, основанные на контенте, включены в его сценический текст, который генерируется пользователями и языком. Функции на основе профиля содержат информацию о пользователе. Например, пол, возраст, предпочтения пользователя. Там мы показываем различные функции, такие как набор слов (BoW), n-грамм, функция программы (PF), бетонный блок (CB), локально интерпретируемая модель-независимое объяснение (LIME), частота терминов - обратная частоте документа (TF-IDF), эмоциональный набор слов (emotion BW), нормальная форма отрицания (NNF). Процесс обучения будет успешным, если созданные характеристики будут включать в себя большое количество аспектов, которые индивидуально ассоциируются с классом. Вот почему в большинстве представленных экспериментов была предпринята попытка развить широкий спектр характеристик. Поведение, связанное со случаями текстового киберзапугивания, должно быть отражено во входных характеристиках. Однако для оценки совокупности признаков следует использовать методы выбора признаков. Чтобы определить, какие характеристики, скорее всего, релевантны или неуместны для классов, используются алгоритмы выбора признаков.

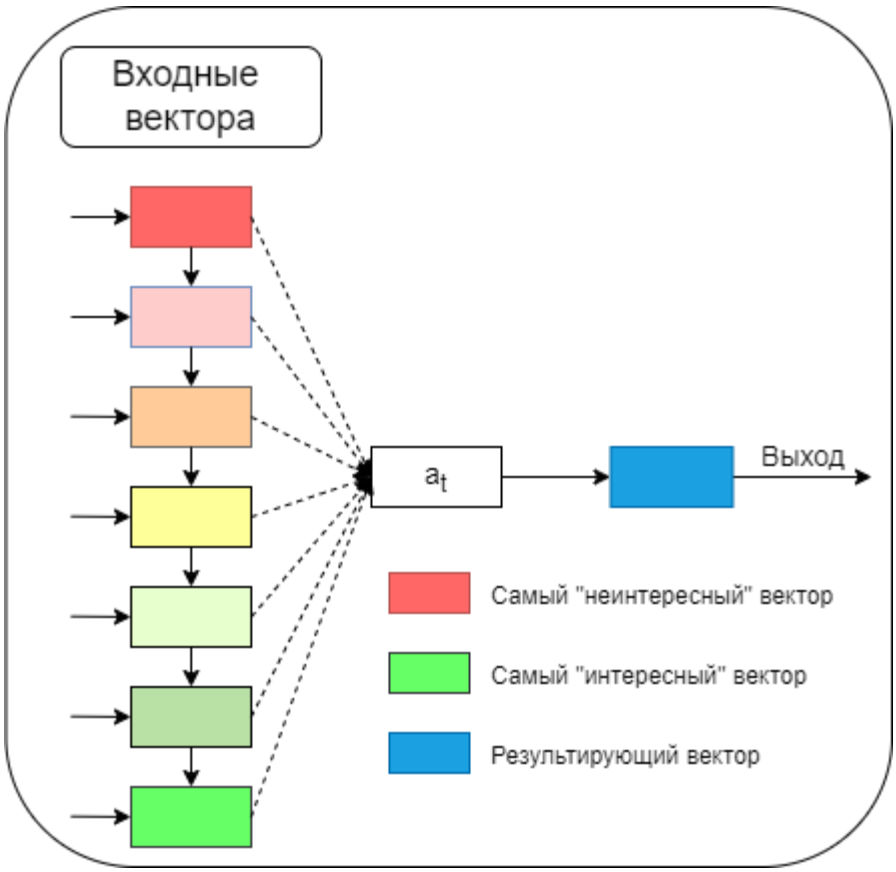


Таблица 2.1: Краткое описание типов признаков, используемых в литературе по прогнозированию киберзапугивания.

Исследование	Функции, основанные на контенте									Функции на основе профиля	
	BoW	n-gram	PF	CB	LIME	TF-IDF	Emotion	BW	Hashtag	NNF	Link
Agustín et al. [30]	✗	✔	✗	✗	✗	✔	✗	✔	✗	✗	✗
A. Pereraa and P. Fernandob [31]	✗	✗	✔	✗	✗	✔	✗	✗	✔	✗	✗
R. Zhao and K. Mao [32]	✔	✗	✗	✔	✔	✗	✗	✗	✔	✗	✔
L Thun et. al [33]	✔	✗	✔	✔	✗	✔	✔	✗	✔	✔	✔
M. López-Vizcaíno et. al [34]	✔	✗	✔	✔	✗	✔	✔	✗	✔	✔	✔
Mohammed Ali Al-garadi [35]	✔	✔	✗	✗	✗	✗	✗	✗	✗	✔	✗
Oguzhan Gencoglu [36]	✗	✗	✗	✗	✗	✗	✔	✗	✔	✔	✔
Zhao Meng [37]	✔	✗	✗	✔	✗	✔	✗	✗	✗	✗	✗
Ahmed et al. [39]	✗	✔	✗	✗	✗	✗	✔	✗	✗	✔	✔
Vimala Balakrishnan [40]	✔	✗	✗	✗	✗	✗	✔	✗	✗	✔	✗
Yasin N. Silva [41]	✔	✗	✗	✗	✗	✗	✗	✗	✗	✔	✗

2.4 Выбор параметров

Несмотря на то, что существует несколько классификаторов классификации текстов, сложность пространственной области представляет собой серьезную проблему [38]. Текст может включать сотни или тысячи различных слов, которые считаются функциями, но многие из них могут быть хаотичными, менее релевантными, зашумленными или избыточными по отношению к целевому классу. Это может привести к введению классификаторов в заблуждение, снижая их текущую эффективность [30, 32]. В результате следует использовать выбор признаков для исключения шумных, менее полезных и избыточных объектов из пространства объектов, уменьшая пространство объектов до управляемого размера и повышая точность и эффективность классификаторов.

Создание подмножества признаков, оценка подмножества, критерии остановки и подтверждение результатов классификации - это четыре основных процесса подхода к выбору признаков [39]. Мы используем поисковый подход для выбора подмножества признаков-кандидатов на первом этапе, которое затем оценивается с использованием критериев приемлемости на втором этапе. Когда требования к остановке выполняются на третьем этапе, создание подмножества и оценка завершаются, и выбирается лучшее подмножество функций из всех кандидатов. Подмножество функций будет проверено с помощью набора проверок на последнем этапе. Подходы к выбору объектов можно разделить на четыре группы в зависимости от того, как генерируются подмножества объектов: модель фильтра [31, 33, 37], модель-оболочка [34, 40], модель встраивания [41] и гибридная модель [36]. Из-за своей точности и производительности большой процент подходов к выбору объектов для классификации текста основаны на фильтрах. [31, 34] обеспечивают тщательное изучение и сравнение альтернативных методов отбора признаков для общих данных.

Алгоритмы выбора признаков редко обучались обнаружению кибербуллинга на онлайн-платформах и сайтах социальных сетей с использованием ML в современных исследованиях. В большинстве рассмотренных исследований (например, [34, 37, 30-32]) не использовался отбор признаков для определения того, какие свойства важны при обучении ML. В некоторых исследованиях [33] для выбора релевантного признака использовался анализ главных компонент и хи-квадрат.

Угрозы, возникающие в сетевой среде, естественным образом стимулируют развитие исследовательского аппарата для изучения их факторов, механизмов и последствий. Эмоциональная природа тех процессов, которые в основном определяют деструктивный характер воздействия негативных сетевых явлений на человека, ставит в приоритет создание инструментов автоматического анализа, позволяющих оценивать выраженность признаков аффективных состояний участников сетевого общения, то есть методов анализа настроений в широком смысле от этого слова. Инжир. 3 иллюстрирует обнаружение кибербуллинга в социальных сетях с использованием методов машинного обучения. На рисунке описан

процесс обнаружения кибербуллинга - от сбора данных до классификации текстов о кибербуллинге. В следующих подразделах объясняется каждый этап на рис. 3.

2.5 Применение алгоритмов машинного обучения

Поскольку мы были в самом начале исследовательского пути, было решено начать эксперименты с алгоритмами машинного обучения по задаче классификации кибербуллинга. Существует несколько алгоритмов машинного обучения, которые могут быть использованы для решения задачи классификации. Ниже приведены примеры и краткое описание популярных алгоритмов, используемых при классификации:

1) Naive Bayes: Это простой, но эффективный алгоритм, который использует теорию вероятностей для классификации текстовых данных. Он часто используется в качестве базовой модели для задач классификации текста[19].
2) Машины опорных векторов (SVMs): Эти алгоритмы направлены на поиск гиперплоскости в многомерном пространстве (пространстве параметров), которая максимально разделяет различные классы[20]. Они эффективны для задач классификации текста с наборами данных малого и среднего размера. Этот тип алгоритмов лучше подходит для дальнейшего понимания задач в ML и для начала экспериментов.
3) Дерево решений: Эти алгоритмы создают древовидную модель решений, которая может быть использована для классификации текстов. Они просты в интерпретации и реализации, но могут быть подвержены переобучению, поэтому для этой модели имеет смысл для начала применить задачу снижения мерности[21].
4) Методы случайного леса: Эти алгоритмы являются расширенной версией деревьев решений, которые создают ансамбль деревьев решений и используют их для классификации текстов. Они менее подвержены переобучению, чем деревья решений по отдельности, и часто лучше справляются с задачами классификации[22].
5) Метод K-ближайших соседей (KNN). Это простой и эффективный метод классификации и регрессии. В обоих случаях входные данные состоят из k ближайших обучающих примеров в пространстве объектов[23]. В классификации KNN результатом является принадлежность к классу. Объект классифицируется большинством количеством его соседей, при этом объекту присваивается класс, наиболее распространенный среди его k ближайших соседей.
6) Логистическая регрессия. Это алгоритм контролируемого обучения, который берет набор помеченных обучающих данных и учится предсказывать двоичную метку новых, невидимых данных[24]. Цель логистической регрессии - найти наилучшие параметры, или веса, которые предсказывают вероятность того, что данный входной сигнал принадлежит к определенному классу.

Важно выбрать правильный алгоритм для задачи классификации текста, исходя из размера и сложности набора данных и имеющихся у нас ресурсов. Поэтому для первых экспериментов было решено работать с готовыми к использованию наборами данных на английском языке. Набор данных Twitter был использован для полного понимания принципов машинного обучения и в качестве отправной точки[25].

2.6 Работа с данными.

Перед вводом необработанных («сырых») данных в алгоритмы машинного обучения данные должны быть переведены в формат, приемлемый для моделей. Так как алгоритмы машинного обучения в глубоком своем понимании являются математическими формулами, то и входные данные должны быть понятными для моделей [26-28]. Вот почему предварительная обработка данных очень важна в задаче машинного обучения. Ниже приведены шаги по подготовке исходных данных для алгоритмов машинного обучения и их непосредственному использованию:

1. Сбор данных.
2. Очистка данных.
3. Преобразование данных.
4. Разделение данных и обучение моделей.
5. Оценка работы алгоритмов.

На этом этапе исследования у нас был набор данных, в частности, у нас был набор данных Twitter, который находится в открытом доступе на kaggle.com. Как только у нас появились данные, отпала необходимость проходить первые 2 этапа, поэтому мы могли их пропустить. Преобразование данных в машинном обучении относится к процессу преобразования необработанных данных в подходящий формат, который может быть легко понят и использован алгоритмами машинного обучения. Поскольку алгоритмы машинного обучения по своей глубокой структуре представляют собой математические формулы, все данные должны быть преобразованы в приемлемый формат, подобные числам. На научном языке это также называется стадией предварительной или первичной обработки данных[31]. Предварительная обработка в нашем исследовании включала в себя 4 этапа: удаление знаков препинания и заглавных букв, маркирование, удаление стоп-слов и лемматизация.

Удаление знаков препинания и заглавных букв при предварительной обработке текстовых задач машинного обучения важно по нескольким причинам. С одной стороны, это помогает нам в снижении уровня шума данных. Когда мы сталкиваемся с концепцией зашумленных данных, устранение знаков препинания и заглавных букв может повысить качество данных за счет уменьшения ненужных отвлекающих факторов и шума, которые могут препятствовать способности алгоритмов машинного обучения распознавать значимые закономерности[32]. В результате данные могут быть

более концентрированными и ориентированными на соответствующую информацию. Это также служит для придания консистенции. Несоответствия в заглавных буквах, такие как использование разными авторами заглавных или строчных букв для одних и тех же слов, могут привести к тому, что алгоритмы машинного обучения будут рассматривать эти слова как разные токены. Удаление заглавных букв гарантирует, что слова обрабатываются последовательно, независимо от их начальной формы[33].

Следующим шагом в предварительной обработке является токенизация. В контексте машинного обучения токенизация относится к процессу разделения данного текста на компоненты или части, такие как слова, фразы, символы или другие значимые элементы, известные как токены. Это важный шаг в подготовке текстовых данных, особенно для обработки естественного языка[34]. Токенизация хороша тем, что она преобразует неструктурированный текст в структурированную форму, которая может быть использована в качестве входных данных для алгоритмов машинного обучения. Существует несколько типов токенизации, таких как токенизация слов, токенизация предложений, токенизация слогов и токенизация символов. Эти типы зависят друг от друга в зависимости от технических характеристик задачи. В нашем случае по той причине, что нам нужно проанализировать слова и зависимости слов, использовался word tokenizer. Пример обозначения слова приведен ниже.

Входной текст: "Hello, today is a beautiful day."
Выходные токены: ["Hello", "today", "is", "a", "beautiful", "day"]

Таки образом, мы разделили целое предложение, на его составные части, что в дальнейшем поможет нам сделать анализ зависимости слов между собой.

Следующий этап предварительной обработки текста после токенизации включает в себя удаление стоп-слов, которые являются обычно используемыми словами, не придающими тексту существенного значения. К примеру, предлоги в тексте не несут в себе какой-либо важной информации, но занимают память при обработке результирующего вектора. Этот процесс является важным компонентом обработки естественного языка и используется для повышения качества используемых входных данных для алгоритмов машинного обучения.

Пример удаления стоп-слов:

Перед удалением стоп-слов:
Входной текст: "Hello, today is a beautiful day."
Токены: ["Hello", "today", "is", "a", "beautiful", "day"]

После удаления сто-слов:
Выходные токены: ["Hello", "today", "beautiful", "day"]

Удаляя стоп-слова "The", "is" и "a", результирующие токены фокусируются на значимых словах входного текста.

Последний шаг, который мы реализовали в нашем исследовании - это стемминг или лемматизация. В задаче машинного обучения стемминг - это инструмент предварительной обработки, который фокусируется на приведении слов к их корневой форме или, как его еще называют, стем[35]. Основная цель стемминга - стандартизировать термины со схожими значениями, даже если они встречаются в разных словоизменениях или производных. Делая это, алгоритмы машинного обучения могут более эффективно распознавать релевантные шаблоны в тексте.

Например, слова "бегущий", "бегун" и "бегал" имеют один и тот же корень - "бег". Стемминг может быть использован для получения корневой формы слов, что позволяет алгоритмам машинного обучения с большей легкостью выявлять семантическое сходство между словами[26].

В заключение следует отметить, что предварительная обработка данных является важным этапом подготовки текстовых данных для алгоритма машинного обучения. Основная цель предварительной обработки - преобразовать необработанные, неструктурированные данные в структурированный, чистый и стандартизированный формат, который можно эффективно использовать в качестве входных данных для моделей машинного обучения. Эти реализованные методы предварительной обработки помогают повысить эффективность, скорость и обобщение алгоритмов машинного обучения за счет уменьшения шума и сложности входных данных.

2.7 Показатели оценки

В исследованиях по обнаружению кибербуллинга в качестве параметров оценки используются точность(accuracy), прецизионность(precision), отзыв(recall), F-мера и площадь под кривой рабочих характеристик приемника (AUC-ROC) [57-58].

$$accuracy = \frac{TP + TN}{P + N} \tag{2.1}$$

$$preision = \frac{TP}{TP + FP} \tag{2.2}$$

$$recall = \frac{TP}{TP + FN} \tag{2.3}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{2.4}$$

Стоит описать формулы, которые используются при расчете показателей эффективности моделей. В нашей работе мы использовали 5 метрик, которые обычно используются в задачах бинарной и многоклассовой

классификации. Перед началом работы с формулами приведем несколько сокращений для информации: TP – Истинно положительный, TN – Истинно отрицательный, FN – ложноотрицательный, FP – Ложноположительный[72].

Точность. Точность - это отношение правильно классифицированных объектов к общему количеству объектов. Он рассчитывается по формуле, приведенной в формуле (1). Он показывает все истинные метки для всех объектов в наборе данных.

Precision. Precision - это отношение правильно предсказанных положительных объектов к общему числу объектов, маркированных как положительные. Он рассчитывается по формуле, приведенной в формуле (4.18). Это показывает, сколько истинных меток было взято из положительной предсказанной метки. Высокий precision означает, что модель делает меньше ложноположительных прогнозов.

Recall: Recall - это доля правильно предсказанных положительных объектов к общему числу положительных объектов в наборе данных. Отзыв измеряет, сколько фактических положительных выборок правильно предсказано моделью. Высокий recall означает, что модель делает меньше ложноотрицательных прогнозов.

Оценка F1 - полезный показатель, когда есть компромисс между точностью и отзывчивостью. Например, в нашей задаче желательно высокое количество повторений, чтобы гарантировать обнаружение максимальное положительных случаев, но высокая точность также важна для минимизации ложноположительных результатов. Оценка F1 помогает найти баланс между этими конкурирующими показателями и выбрать модель, которая хорошо справляется как с precision, так и с recall.

В дальнейшем наша работа ставила целью исследовать феномен кибербуллинга и в Казахстанском медиа пространстве, но здесь мы сразу столкнулись с проблемой. Вследствие отсутствия данных и инструментов мы пришли к выводу, что нам придется все делать самим. По этому поводу был создан план, следуя которому, можно было сделать больше работ касательно темы исследовательской работы. План включал в себя следующие задачи:

1) Собрать набор данных для Казахского языка.
2) Вручную сделать маркировку классов.
3) Собрать языковой корпус для кибербуллинга на казахском языке.
4) Провести анализ машинного и глубокого обучения на полученном наборе данных.

3 ПОДХОДЫ ГЛУБОКОГО ОБУЧЕНИЯ ДЛЯ ВЫЯВЛЕНИЯ КИБЕРБУЛЛИГА

Подходы машинного обучения это конечно хороший инструмент в исследованиях, но у них есть свои недостатки. Например, важно понимать, что качество прогноза модели зависит от качества и количества данных, на которых она обучается, а также от сложности модели. Если данных недостаточно, или если они содержат ошибки или пропуски, то модель может работать некорректно или давать неправильные результаты. Кроме того, машинное обучение ввиду своей простоы может просто не найти скрытых зависимостей между данными[46].

В отличие от машинного обучения, нейронные сети имеют значительные преимущества в задаче классификации текстов. Они позволяют работать с большими объемами текстовых данных и выявлять скрытые зависимости между словами и фразами. Это особенно важно для задач, где необходимо учитывать контекст и смысл слов, например, в задачах анализа тональности или классификации текстов[47]. К тому же, нейронные сети могут использоваться для обучения без учителя, когда задача состоит в поиске скрытых закономерностей в данных. Например, можно использовать автокодировщики, чтобы находить скрытые признаки текстовых данных. Этот способ мы применили при разработке нейронной сети с использованием механизма внимания, которая представлена в конце этой главы. Кроме того, используя нейронные сети, мы открываем себе множество различных возможностей для работы: создавать нейронные сети любого размера, как небольшие, так и объемные нейронные сети, которые на стадии обучения могут использовать десятки миллионов параметров, комбинировать различные архитектуры и всевозможные способы оптимизации для улучшения модели. При работе с алгоритмами машинного обучения мы были ограничены в ресурсах и мощностях, ввиду того, что, сравнивая алгоритмы машинного и глубокого обучения, можно привести аналогию: машинное обучение - это нейронная сеть с единственным внутренним слоем, а глубокие нейронные сети могут состоят из множества слоев, в некоторых случаях количество этих слоев может достигать десятков тысяч. В рамках исследования были реализованы и проведены эксперименты со всеми популярными архитектурами нейронных сетей: рекуррентные нейронные сети, свёрточные нейронные сети, сети с короткой долговременной памятью, многослойные персептроны, обычные настраиваемые глубокие нейронные сети и самая результативная модель – это глубокая нейронная сеть с использованием механизма внимания.

Оборудование. Ввиду того, что нейронные сети требуют больших вычислительных мощностей по сравнению с машинным обучением, было принято решение использовать более продвинутое оборудование для обучения нейронных сетей. Для сравнения ниже приведены описания 2 категории разнородных платформ (CPU и GPU).

CPU: Мы оценили новейшие доступные процессоры CPU, которые есть на рынке, и решили использовать процессор 12th Gen Intel(R) Core(TM) i7-12700KF, частота: 3610 МГц, ядер: 12, логических процессоров: 20. Настроив 4 канала для этого процессора (4 вкладки ОЗУ), мы получили возможность обрабатывать 48 операций параллельно, что означает за один пакет процессор может одновременно принять 48 вычислительных процессов. Это достаточно высокий показатель, но для алгоритмов глубокого обучения данная мощность является очень маленьким. Скорость работы такого процессора составляет 1160 GFLOPS.

GPU: Далее мы рассмотрели вычисление на GPU. Для работы была куплена видеокарта NVIDIA GeForce RTX 3080, с 12 ГБ оперативной памяти, частота видеопамяти 19 000 МГц.

Таблица 3.1 – Сравнение CPU и GPU

	CPU	GPU
Скорость работы, GFLOPS	1 160	30 600
Параллельное вычисление	48 оп/сек	8 704 оп/сек
Частота памяти, МГц	3610	19 000

Как видно из таблицы, можно с уверенностью сказать, что использование GPU дает колоссальное преимущество как в работе с машинном обучением, так и в работе с глубокими нейронными сетями. В дальнейшем все нейронные сети были имплементированы и обучены используя CUDA ядра от NVIDIA.

1.1 Рекуррентные нейронные сети в задаче выявления кибербуллинга

Прежде чем рассказывать про структуру рекуррентных нейронных сетей, немного углубимся в историю. Рекуррентные нейронные сети являются развитием классических нейронных сетей и были созданы в конце 1980-х и начале 1990-х годов[48]. Их основная идея заключается в использовании рекуррентных связей между элементами сети, что позволяет сети обрабатывать последовательности данных, такие как последовательности слов в предложениях или последовательности временных рядов. Другими словами, рекуррентные нейронные сети – это сети с циклами, которые хорошо подходят для обработки последовательностей, к примеру, задачи Seq-to-Seq. Также смысл применения рекуррентных нейронных сетей заключается в том, что мы можем использовать их когда нам важен порядок, когда критически важно соблюдать последовательность входных данных. А в задачах обработки текста расположение слов играет немаловажную роль, так как от постановления слов может в корне поменяться. Существует 4 вида RNN: один к одному, один ко

многим, многие к одному и многие ко многим. В целях наших исследований, подходит вид нейронной сети многие ко многим. Ниже представлен рисунок показывающий структуру такой сети. (Картинка)

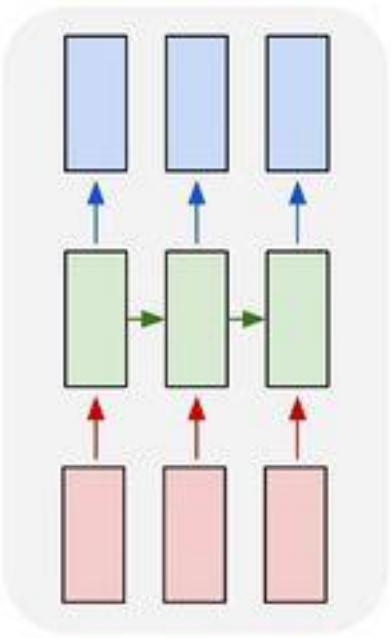


Рисунок 3.1 – структура рекуррентной нейронной сети

Если говорить про расчеты, то ниже приведены формулы. Рассмотрим входную последовательность X = (x_1, x_2, ..., x_t), где x_t - входные данные на шаге времени t. RNN поддерживает скрытое состояние h_t на каждом временном шаге, которое обновляется с использованием предыдущего скрытого состояния h_(t-1) и текущего входного x_t.

Преобразование входных данных в скрытые вычисления:

$$h\_t = \tanh(W\_hh * h\_(t-1) + W\_xh * x\_t + b\_h) \quad (3.1)$$

Вычисление скрытого вывода:

$$o\_t = W\_ho * h\_t + b\_o \quad (3.2)$$

где W_hh, W_xh и W_ho - весовые матрицы, b_h и b_o - члены смещения, а tanh - функция активации (также могут использоваться другие функции активации).

1.2 Использование простых глубоких нейронных сетей в задаче выявления кибербуллинга

Определение: Глубокое обучение - это класс алгоритмов машинного обучения, которые используют несколько слоев для постепенного извлечения функций более высокого уровня из необработанных входных данных. Например, при обработке изображений нижние слои могут идентифицировать

края, в то время как более высокие слои могут идентифицировать понятия, относящиеся к человеку, такие как цифры, буквы или лица[52].

В нашем примере нейронные сети работают по такому же принципу, один слой определяет порядок расположения слов во входном тексте, предоставляя первоначальную информацию о входном потоке данных. Следующий слой уже извлекает зависимость слов между собой и так далее.

Уравнение для созданной нейронной сети представляет собой линейную комбинацию независимых переменных и их соответствующих весов для каждого нейрона. Уравнение нейронной сети выглядит следующим образом:

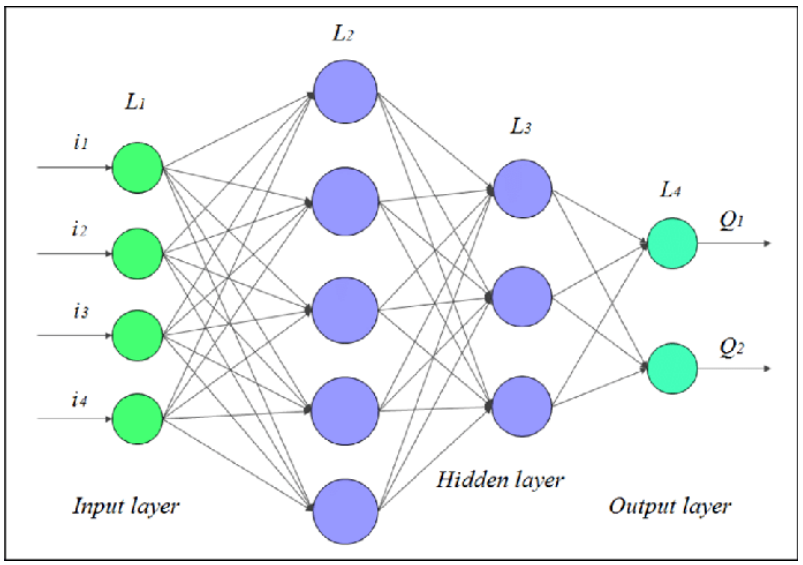


Рисунок 3.2 – Структура простой глубокой нейронной сети

1.3 Использование сверточных нейронный сетей в задаче классификации текстов

Определение: Сверточные нейронные сети - это тип архитектуры нейронных сетей, в основном используемый для обработки изображений и задач компьютерного зрения. Однако они также были успешно применены для классификации текстов благодаря их способности распознавать местные и пространственные особенности. В контексте классификации текста сверточные нейронные сети используются для выявления значимых шаблонов, таких как n-граммы, внутри текста, что может помочь при разделении текста на различные классы[55]. Для этого текстовые документы преобразуются в числовые представления, например, с помощью векторных моделей, таких как word2vec или fastText. После этого числовые представления текстов могут быть использованы как входные данные для сверточной нейронной сети. Сеть может содержать несколько сверточных слоев, которые извлекают важные признаки из текстовых документов, а полностью связанные слои используются для классификации текстов на основе этих признаков.

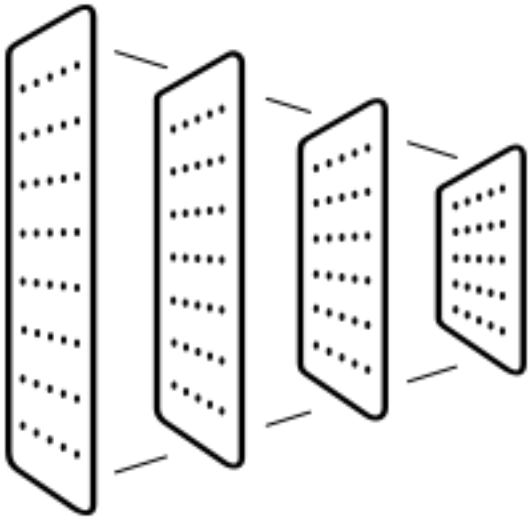


Рисунок 3.3 – Визуальная репрезентация сверточной нейронной сети

Как видно на рисунке 4.3 с каждым следующим слоем количество нейронов становится все меньше и меньше. Это помогает передавать следующим слоям передавать менее значимые связи, что на следующем уровне комбинируя их можно было получить более сложные связи.

1.4 Использование сетей с кратковременной долговременной памятью для выявления кибербуллинга

Определение: Долговременная кратковременная память (LSTM) - это тип архитектуры рекуррентной нейронной сети (RNN), предназначенный для решения проблемы исчезающего градиента, возникающей в традиционных RNN[56]. Технически оно является логической модификацией рекуррентных нейронных сетей с одним отличием. Нейронная сеть кратковременной долговременной памяти имеет уникальную архитектуру на базе рекуррентных нейронных сетей, которая позволяет им хранить информацию и манипулировать ею в течение длительных последовательностей формата Seq[57]. Ввиду этого отличия сети с кратковременной долговременной памятью улучшили методы обработки естественного языка. Ключевыми компонентами данной сети являются:

состояние ячейки – это элемент сети, которая хранит в себе информацию про текущее состояние входной последовательности и обновляется с каждым шагом на основе входных данных, выходных данных и предыдущего состояния сети.

входной шлюз – это элемент сети, который определяет, какая часть входящей информации должна храниться в состоянии ячейки. Он использует функцию активации sigmoid для получения значения 0 и 1, где 0 указывает на то, что никакая информация не должна сохраняться, а 1 указывает на то, что вся входящая информация должна быть сохранена.

выходной шлюз – это элемент сети, который определяет, какая часть состояния ячейки должна быть открыта для следующего слоя. Он также

использует функцию активации sigmoid для получения значения от 0 до 1 по тому же принципу что и входной шлюз.

шлюз выброса – это элемент сети, которая показывает, какая часть из предыдущего состояния ячейки должна быть удалена по результатам функции активации sigmoid. Здесь 0 – все состояние должно быть отброшено и 1 – все состояние должно быть сохранено и передано на следующий шаг.

Таким образом, ключевым особенностью сети долговременной кратковременной памяти является внедрение ячейки памяти, которая может сохранять информацию в длинных последовательностях. Ячейка памяти регулируется тремя элементами: входным и выходным шлюзами и шлюзом выброса. Эти шлюзы отвечают за управление потоком информации в ячейку памяти, из нее и внутри нее[58].

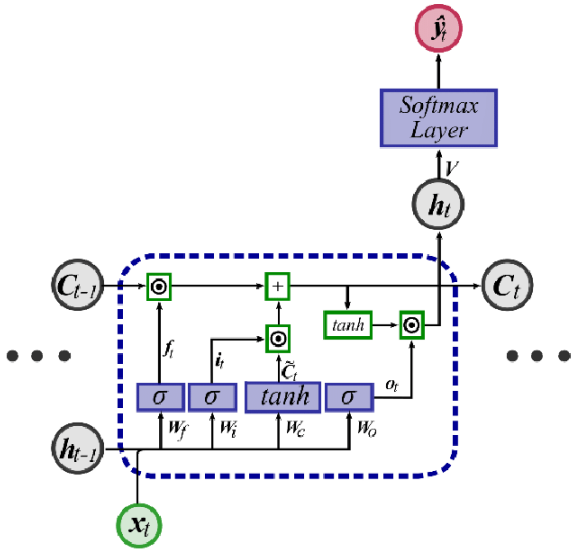


Рисунок 3.4 – Строение долговременной кратковременной памяти

1.5 Использование многослойного персептрона в задаче классификации текстовых данных

Определение: Многослойный персептрон - это тип искусственной нейронной сети, которая состоит из множества слоев взаимосвязанных нейронов. Это сеть с прямой связью, что означает, что информация поступает с входного уровня через скрытые слои (если таковые имеются) на выходной слой без каких-либо циклов[62]. MLP используются для различных задач, таких как классификация, регрессия и аппроксимация функций. Обычно многослойный перцептрон состоит из:

- Входного слоя: Входной слой состоит из нейронов, которые представляют входные данные. Каждый нейрон соответствует определенному признаку, и количество нейронов во входном слое зависит от размерности входных данных.

- Скрытых слоев: MLP может иметь один или несколько скрытых слоев, которые состоят из нейронов, выполняющих нелинейные преобразования входных данных. Скрытые слои позволяют сети изучать сложные шаблоны и представления. Количество скрытых слоев и нейронов в каждом слое может варьироваться в зависимости от сложности задачи и дизайна архитектуры.
- Выходного слоя: Выходной уровень состоит из нейронов, которые производят окончательные прогнозы или классификацию. Количество нейронов в выходном слое зависит от задачи. Для бинарной классификации обычно имеется один выходной нейрон, в то время как для многоклассовой классификации выходных нейронов столько, сколько существует классов. Для задач регрессии количество выходных нейронов соответствует размерности целевой переменной.
- Обратного распространения: MLP обычно обучаются с использованием обучения с учителем, называемого обратным распространением. Этот алгоритм включает в себя вычисление градиента функции потерь относительно каждого веса по правилу цепочки, затем обновление весов с использованием алгоритма оптимизации, такого как градиентный спуск или один из его вариантов.

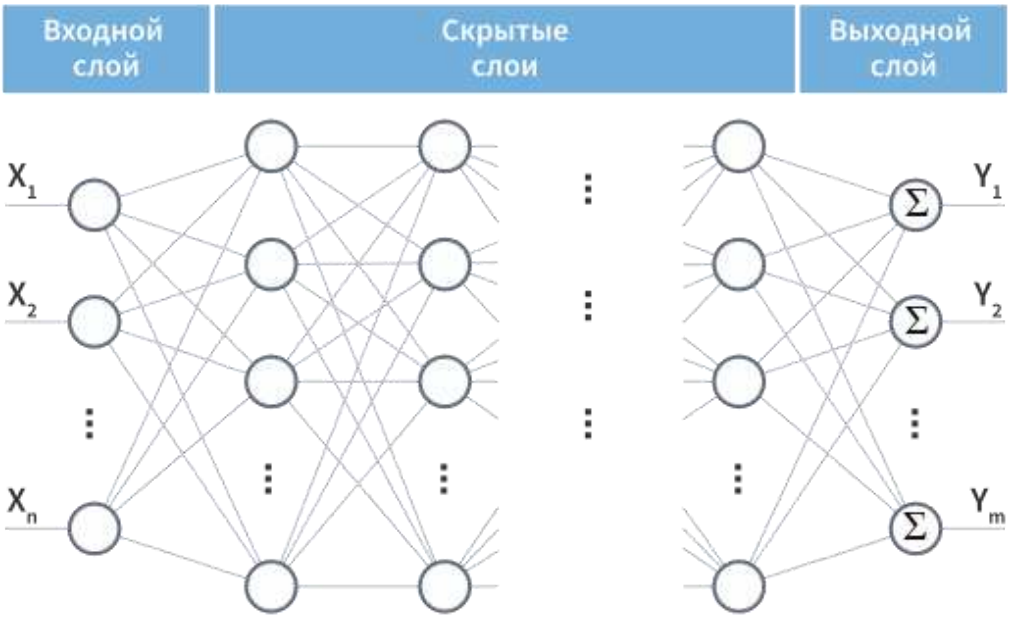


Рисунок 3.5 – Графическое изображение многослойного персептрона

Несмотря на свою относительную простоту, многослойный персептрон имеет некоторые моменты, которые надо учесть при работе с ним. Из-за своей структуры она требует немного другого процесса обработки текста, а также настройки гиперпараметров[63].

Первый момент — это то, что плотные слои не совсем точно могут улавливать зависимости между словами, поэтому сама предобработка данных должна проходить по-другому. Если для предыдущих моделей вполне подходила векторизация с помощью TF-IDF, то для работы с MLP требуется

более комплексные методы векторизации, к примеру Word2Vec, GloVe. После анализа двух алгоритмов векторизации было принято решение использовать Word2Vec[64]. Word2Vec часто считается лучшим, чем некоторые другие методы векторизации, из-за его способности эффективно улавливать семантические и синтаксические связи между словами. Однако важно отметить, что другие методы векторизации слов имеют свои достоинства и могут быть более подходящими в зависимости от конкретного варианта использования. Чаще всего Word2Vec выигрывает из-за того, что он в своей структуре использует CBOW или Skip-grams в зависимости от выбора.

Приведем простой пример векторизации слов с помощью Word2Vec. Возьмем текст "Кибербуллинг - это плохое явление, которое развивается в этом мире и с эти надо бороться" и проведем векторизацию для этого текста. После отработки процесса векторизации мы получим следующий набор векторов

Вектор для слова 'Кибербуллинг': [
-7.1957498e-03 4.2383196e-03 2.1665227e-03 7.4479855e-03
-4.8902300e-03 -4.5656390e-03 -6.0920399e-03 3.2968649e-03
-4.5059021e-03 8.5214656e-03 -4.2858059e-03 -9.1062961e-03
-4.8138797e-03 6.4142672e-03 -6.3652224e-03 -5.2589928e-03
-7.3031816e-03 6.0219853e-03 3.3502064e-03 2.8407574e-03
-3.1341878e-03 6.0382476e-03 -6.1435234e-03 -1.9830675e-03
-5.9785792e-03 -9.9214667e-04 -2.0253430e-03 8.4818425e-03
7.5381853e-05 -8.5764267e-03 -5.4271189e-03 -6.8764468e-03
2.6991712e-03 9.4558690e-03 -5.8152163e-03 8.2646431e-03
8.5294433e-03 -7.0581078e-03 -8.8815102e-03 9.4623314e-03
8.3707273e-03 -4.6886741e-03 -6.7269723e-03 7.8412192e-03
3.7634561e-03 8.0909217e-03 -7.5748521e-03 -9.5199775e-03
1.5798000e-03 -9.8045077e-03 -4.8894407e-03 -3.4583977e-03
9.6187936e-03 8.6196754e-03 -2.8321401e-03 5.8263848e-03
8.2380427e-03 -2.2639779e-03 9.5289340e-03 7.1619805e-03
2.0403073e-03 -3.8494032e-03 -5.0775236e-03 -3.0505764e-03
7.8842333e-03 -6.1884420e-03 -2.9168560e-03 9.1975257e-03
3.4519010e-03 6.0725654e-03 -8.0348616e-03 -7.4956345e-04
5.5232574e-03 -4.7127549e-03 7.4766264e-03 9.3184365e-03
-4.0939171e-04 -2.0669759e-03 -5.9726991e-04 -5.7913996e-03
-8.3868923e-03 -1.5108233e-03 -2.5517803e-03 4.3863910e-03
-6.8645719e-03 5.4149418e-03 -6.7392630e-03 -7.8208866e-03
8.4703164e-03 8.9197345e-03 -3.4805075e-03 3.4926343e-03
-5.7939040e-03 -8.7490240e-03 -5.5091968e-03 6.7523615e-03
6.4174528e-03 9.4369575e-03 7.0569622e-03 6.7549422e-03
]

Далее попробуем найти максимально подходящие слова для слов кибербуллинг в этом тексте:

Наиболее подходящие слова для слов 'кибербуллинг':

[

('надо', 0.12303338199853897),

('в', 0.08089427649974823),

('бороться', 0.05471491068601608),

('и', 0.016177210956811905),

('явление', 0.013262882828712463)

]

В данном примере был использован алгоритм Skip-grams, формула которого приведена ниже:

$$L = 1/T * \Sigma_{t=1\_to\_T} \Sigma_{c=-C\_to\_C, c\neq 0} \log p(w_{t+c}|w_t) \quad (4.14)$$

где T – это количество слов в учебном корпусе(тексте), C - размер окна (контекстное окно), w_t - целевое слово, а w_t +c представляет контекстные слова вокруг целевого слова.

Далее используется функция softmax для преобразования выходных данных нейронной сети в распределение вероятностей по словарному запасу. Вероятность того, что контекстное слово задано целевым словом, определяется по данной формуле:

$$p(w_{t+c}|w_t) = \exp(v'_{t+c}{}^T * v_t) / \Sigma_{j=1\_to\_V} \exp(v'_j{}^T * v_t) \quad (4.15)$$

где v_t – это входной вектор слов (вложение) для целевого слова w_t, v't+c - выходной вектор слов для контекстного слова w_t+c, а V - размер словаря.

1.6 Механизм внимания

Механизм внимания в глубоких нейронных сетях - это механизм, который позволяет модели фокусироваться на определенных частях входных данных при составлении прогнозов[66]. Он обычно используется в задачах обработки естественного языка (NLP), таких как машинный перевод, суммирование текста и анализ настроений; Кроме того, он хорошо подходит для задач бинарной и многоклассовой классификации.

Архитектура уровня внимания включает в себя три компонента:

1. Запрос(Query): вектор, представляющий текущее состояние модели или выходные данные предыдущего слоя.
2. Ключ(Key): Матрица, содержащая информацию о входных данных.
3. Значение(Value): Матрица, представляющая входные данные.

Механизм внимания вычисляет набор оценок внимания между запросом и ключом, которые затем используются для вычисления взвешенной суммы матрицы значений. Результатом уровня внимания является контекстный вектор, который фиксирует соответствующую информацию во входных данных. Математическое представление механизма внимания приведено ниже[67].

$$\text{Attention}(\text{Query}, \text{Key}, \text{Value}) = \text{softmax}(\text{Query} \times \text{Key.T}) \times \text{Value} \quad (4.16)$$

где softmax – это функция, нормализующая показатели внимания, x - операция точечного произведения, а T обозначает транспонированную матрицу.

На практике механизмы внимания могут иметь несколько начал (запросов), что позволяет модели одновременно обрабатывать разные части входных данных. Выходные данные нескольких начал объединяются, и мы можем сформировать конечный выходной сигнал механизма внимания[68].

В связи с тем, что механизм внимания работает по совершенно другому принципу, стоит описать структуру механизма внимания отдельно для полного понимания всей модели обучения. Для начала важно понять, почему механизм внимания упростил машинный перевод и другие задачи обработки естественного языка. Ранее машинный перевод опирался на модели кодировщик-декодер, которые состоят из двух сетей - кодировщика и декодирующего устройства. Роль модели кодировщика заключается в преобразовании входной последовательности в вектор, который инкапсулирует всю информацию о предложении[69]. Затем этот вектор передается модели декодера, которая генерирует требуемый выходной сигнал. Этот уровень модели принимает в качестве входных данных, закодированных в векторы, и по их значению вычисляет “важность” каждого вектора. Он находит те, которые являются более релевантными, и работает с важными векторами, которые представляют больше информации, чем другие векторы[70].

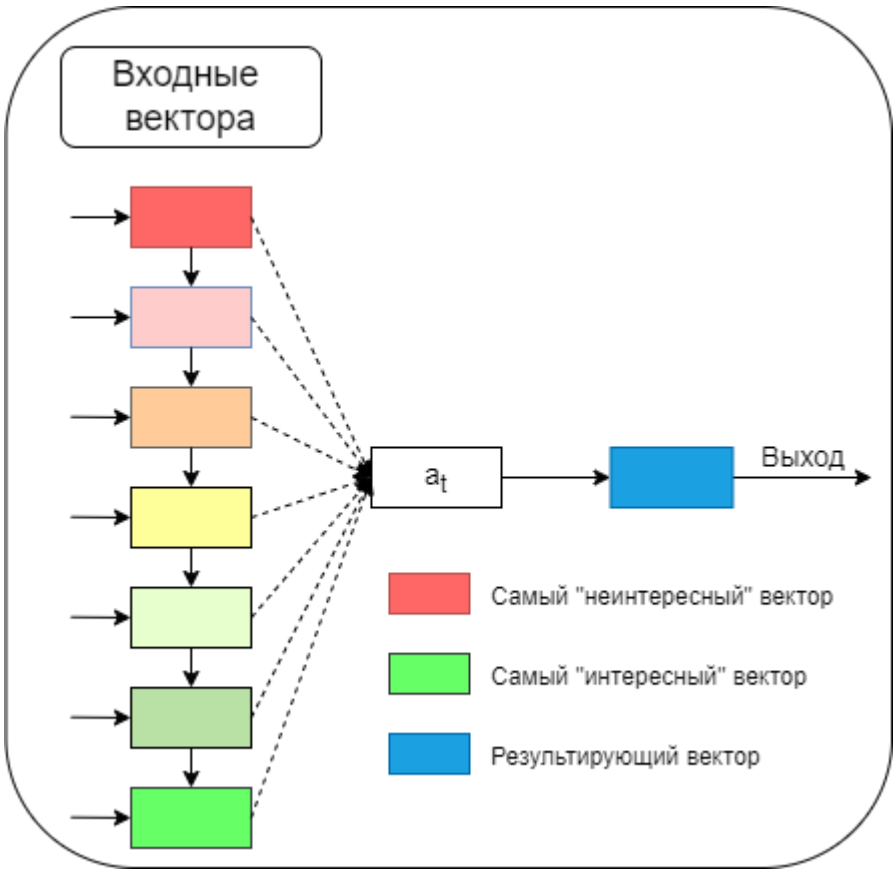


Рисунок 3.6 – Структура механизма внимания

Формула расчета внимания на этом уровне приведена выше при описании структуры механизма внимания. Ячейка a_t хранит в себе информацию про «важные» вектора и передает их на вход контекстного вектора, который является итоговым выходным вектором ячейки внимания[71].

4 МАТЕРИАЛЫ И МЕТОДЫ

Следующим этапом в исследовательском пути был сбор текстовых данных для Казахского языка. Ввиду того, что Казахский язык не является популярным языком в научной сфере невозможно было найти готовые наборы данных для экспериментов по теме исследования. Это привело нас к тому, что было принято решение собрать данные самим. Вручную собирать данные крайне неэффективно, так как человек является очень медленным в развитый век цифровых технологий. Поэтому автоматизация процесса сбора данных является самым приемлемым решением проблемы.

4.1 Построение парсера.

Проанализировав существующую литературу по сбору данных и анализу набора инструментов, которые позволяют автоматически собирать данные, был собран план, по которому можно было собрать нужные данные. Мы начали создавать собственный частично автоматический парсер для сбора данных из открытых источников и социальных сетей. На рисунке показана основная схема построения парсеров, которая описана ниже.

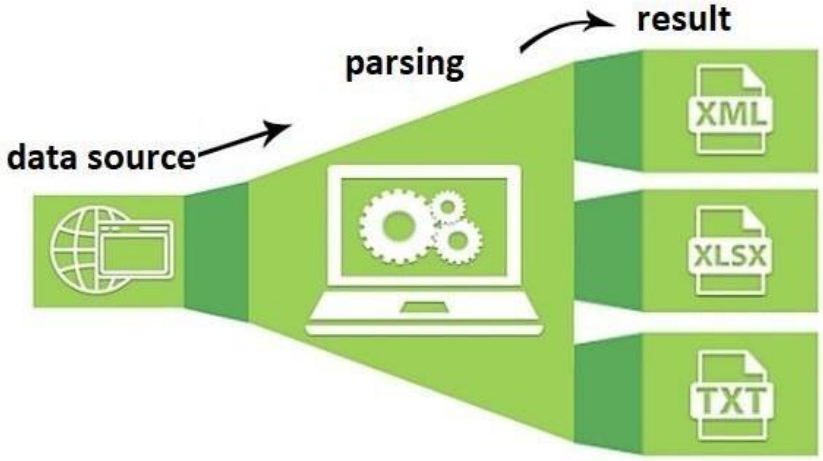


Рисунок 4.1 – Схема работы парсера

После анализа мы выяснили, что есть два типа парсеров: парсер на основе API и пользовательский веб-парсер, способный принимать ключевые слова в запросах и выполнять поиск в веб-страницах и других интернет ресурсах. Каждый из этих типов парсеров имеет свои преимущества и недостатки. Перед нами стоял выбор. При разработке обоих типов парсеров было решено создать собственный парсер без использования каких-либо API, по той причине, что парсеры на основе API ограничивали функциональность, чего могло быть недостаточно. Именно поэтому мы начали разрабатывать собственный синтаксический сборщик. Поскольку мы использовали Python в качестве основного языка программирования, этот язык программирования имел большое количество библиотек для синтаксического анализа данных с любого веб-сайта[37]. Для синтаксического анализа веб-сайта с

использованием Python использовались популярные библиотеки "requests" для извлечения HTML-кода страницы и "BeautifulSoup" для синтаксического анализа HTML.

Основные шаги по разбору данных приведены ниже:

1) Извлечь содержимое веб-страницы.
2) Спарсить HTML-код веб-страницы.
3) Вытащить информацию из полученного кода.

Процесс сбора данных начался с извлечения содержимого веб-страницы путем выбора URL-адреса веб-страницы, которую мы хотели проанализировать [38-40]. Затем мы использовали функцию **requests.get()** для отправки HTTP-запроса на выбранный URL. Эта функция вернула объект response, который содержал информацию о HTTP-запросе и полученном ответе. Чтобы быть уверенными, что запрос прошел успешно, мы проверили код состояния ответа, используя атрибут response.status_code. Он должен вернуть код состояния 200, который указывает на успешный запрос, позволяя нам перейти к разбору содержимого. При успешном запросе мы получили доступ к HTML-содержимому веб-страницы через атрибут response.text, получив исходное HTML-содержимое, необходимое для синтаксического анализа и извлечения информации с помощью BeautifulSoup на следующих этапах.

Листинг 4.1 – Пример передачи данных для создания запроса на сайте

```
headers = {
   'accept':'*/*',
   'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36 OPR/71.0.3770.228'
}
```

На следующем шаге мы проанализировали HTML-содержимое, создав объект BeautifulSoup. Чтобы сделать это, мы передали извлеченный HTML-контент (response.text) и синтаксический анализатор ('html.parser') в качестве аргументов конструктору BeautifulSoup. Полученный в результате объект BeautifulSoup предоставил структурированное представление HTML-содержимого, что позволило нам легко перемещаться и извлекать определенные элементы, такие как блоки кода и теги. Этот шаг жизненно важен для преобразования исходного HTML-содержимого в формат, упрощающий процесс поиска и извлечения определенных фрагментов информации в структуре веб-страницы.

Листинг 4.2 – Пример отправления запроса и получения HTML-кода страницы

```
session = requests.Session()
    request = session.get(base_url, headers = headers)
    if request.status_code == 200:
        soup = bs(request.content, 'html.parser')
        try:
            pagination = soup.find_all('a', {'class':'bloko-button'})
            count = int(pagination[-2].text)
            print(f"Pagication count is {count}")
            for i in range(count):
                url = f'https://web.telegram.org/k/#@kz_bi={i-1}'
                if url not in urls:
                    urls.append(url)
```

На шаге 3 мы извлекли нужную информацию из проанализированного HTML-содержимого, используя различные методы BeautifulSoup, адаптированные для поиска и извлечения определенных элементов по введенному нами значению.

1. Метод find() использовался для поиска первого появления определенного HTML-тега или элемента, необязательно с определенными атрибутами. Этот метод возвращал единственный результат или None, если соответствующий элемент не был найден. Это полезно для извлечения уникальных элементов, таких как заголовки или определенные разделы веб-страницы.
2. Чтобы выполнить поиск по всем вхождениям определенного HTML-тега или элемента, мы запускаем метод **find_all()**, который также может быть объединен с необязательными атрибутами внутри. Этот метод вернул список совпадающих элементов. Например, в теге мы можем найти блок для одного элемента.
3. После определения необходимых элементов мы очистили их содержимое, используя такие методы, как **get_text()/ .text()** для текстового содержимого или **get()** для доступа к значениям атрибутов (например, href для ссылок). Этот шаг позволил нам извлечь интересующую нас фактическую информацию и сохранить ее в более доступном формате, в формате списка, для дальнейшей обработки или анализа.

Выполнив эти описанные шаги, мы смогли ориентироваться в структурированном представлении HTML-кода, предоставляемого BeautifulSoup, и извлекать определенные фрагменты информации с веб-страницы[41]. Создание первой версии парсера представляло для нас незначительную сложность. Потому что потребовалось много времени, чтобы просмотреть HTML-код веб-страницы, чтобы найти теги и их атрибуты. В конце важно отметить, что парсер такого типа уникален и может быть реализован для анализа только одной структуры. Чтобы спарсить другой веб-

сайт, он должен быть написан для структуры именного того ресурса, из которого данные должны были быть получены.

Рисунок 4.2 – Подробная схема построенного парсера.

4.2 Очистка и фильтрация данных

Используя созданный нами парсер, мы собрали набор данных текстов с веб-сайтов и других ресурсов, в результате чего получили неструктурированные и потенциально нерелевантные данные. Это представляет собой еще одну проблему, поскольку использование неочищенных данных препятствует его надлежащему использованию. Чтобы убедиться в актуальности данных и наличии достаточного количества объектов для обучения и тестирования, чрезвычайно важно очистить данные. Существует несколько этапов очистки данных, которые обязательно следует соблюдать.

4.2.1 Фильтрация данных

Мы не можем использовать все данные, которые были собраны. Потому что мы анализируем все данные с веб-ресурсов. Это означает, что созданный нами парсер охватывает всю информацию, даже новости о технологиях или другие темы, которые не имеют отношения к нашей теме. Для этого мы просто написали функцию, которая проверяет, содержит ли каждая ячейка нужные нам ключевые слова. Кроме того, мы составили список разжигающих ненависть слов на казахском языке, поэтому просто включили его в функцию фильтрации. После запуска этой функции мы удалили данные, которые не имеют отношения к теме исследования, и остальные данные готовы к следующим шагам.

4.2.2 Обработка пропущенных значений

Данные, собранные из различных источников, могут содержать недостающие значения, особенно из социальных сетей и комментариев, что может затруднить сбор или привести к необъективным результатам[42]. Обработка отсутствующих значений является важным аспектом очистки данных, поскольку это помогает поддерживать целостность и точность набора данных. Отсутствующие значения могут возникать по различным причинам, таким как ошибки при вводе данных или другие технические проблемы во время сбора данных. Если они не будут приняты во внимание должным образом, пропущенные значения могут привести к необъективному или неполному анализу, что повлияет на производительность и надежность моделей машинного обучения и статистических методов[43]. В этой части работы мы использовали такие методы, как удаление и вменение. При удалении удаляются экземпляры (строки) или определенные точки данных с отсутствующими значениями. Вменение заменяет отсутствующие значения средним значением, медианой или типом значений для данных строки.

4.2.3 Работа с дубликатами и опечатками

Типографские ошибки часто являются результатом человеческого невнимательности и могут встречаться в различных контекстах. Грамматические ошибки могут быть исправлены с помощью целого ряда алгоритмов и методик. Важно устранить эти ошибки, поскольку модели могут по-разному интерпретировать различные характеристики[44]. В данном случае была использована функция автокоррекции. Эта функция работает аналогично таким редакторам, как MS Word, с функциями автозамены. Он исправляет слова на основе предопределенных словарей и алгоритмов. Избежание дублирования также имеет решающее значение для поддержания целостности и качества ваших данных. Потому что это может привести к ситуациям, которые могут снизить точность алгоритмов или привести к неправильному вычислению весов при создании векторов.

После всех этих шагов мы получили частично готовый к использованию набор данных, который позже, конечно, нуждался в первичной предварительной обработке, но отработанные объекты уже были готовы к работе с алгоритмами машинного обучения и нейронными сетями.

4.3 Создание простых глубоких нейронных сетей для выявления кибербуллинга

Определение: Глубокое обучение - это раздел машинного обучения, который фокусируется на искусственных нейронных сетях с несколькими уровнями. Эти сети могут извлекать сложные шаблоны и представления из огромных объемов данных, что делает их пригодными для широкого спектра задач, включая распознавание изображений, обработку естественного языка и

распознавание речи. Было показано, что модели глубокого обучения превосходят традиционные алгоритмы машинного обучения во многих приложениях.

Классификация текста - это задача отнесения данного текста к одной или нескольким предопределенным категориям на основе его содержания. Это обычная задача обработки естественного языка.

```
import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

model = Sequential([

    Dense(64, activation='relu', input_shape=(100,)),

    Dense(32, activation='relu'),

    Dense(16, activation='relu'),

    Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.summary()
```

Выше приведен код имплементированной глубокой нейронной сети с i внутренними плотными слоями. Структура построена таким образом, что в каждом следующем слое количество нейронов меньше чем в предыдущей. Данный метод не является строгим правилом однако такая структура имеет некоторые положительные стороны. Первой особенностью является постепенное снижение мерности векторов, что в свою очередь может помочь в выделении важных закономерностей и удалении нерелевантной информации, что приводит к лучшим результатам и потенциально уменьшает вероятность переобучения. Во-вторых, меньшее количество нейронов в более отдаленных слоях уменьшает количество параметров и вычислительных ресурсов, затрачиываемых для обучения модели, что в свою очередь ускоряет ее обучение и снижает нагрузку на машину. В-третьих, постепенное уменьшение количества нейронов в последующих слоях создает иерархию функций. Идея заключается в том, что слои более низкого уровня изучают базовые функции, а слои более высокого уровня объединяют эти базовые функции в более сложные зависимости[53].

В этой модели расчеты проходят проще чем в других моделях. Говоря простыми словами, тут происходит простой расчет весов и входных данных. Формула расчета итогового выходного слоя приведена ниже.

$$Z = W_1X_1 + W_2X_2 + \ldots + W_nX_n \quad (3.3)$$

где:

Z - это результирующий вектор,

W - распределение весов входных векторов,

X - входные данные

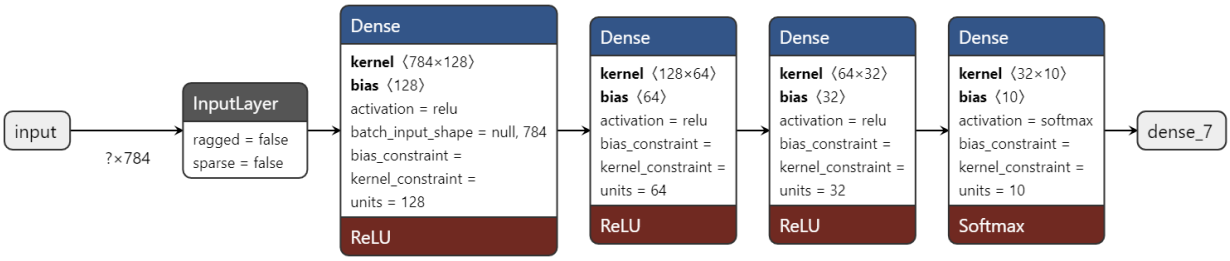


Рисунок 4.3 – Визуальная репрезентация простой

глубокой нейронной сети

На рисунке 4.6 представлено графическое изображение внедренной нейронной сети. Входной слой создает набор векторов размером input_lentg*784, далее передает ее на первый плотный слой. Постепенно каждый следующий слой уменьшая количество нейронов делает предсказание на последний слой, который состоит из одного нейрона. Если этот нейрон после функции softmax пришел в активное состояние, это означает принадлежность к первому классу, а в противоположном случае – ко второму.

4.4 Имплементация рекуррентных нейронных сетей для выявления кибербуллинга

Рекуррентная нейронная сеть (RNN) - это тип искусственной нейронной сети, предназначенной для обработки последовательных данных, где порядок элементов данных играет решающую роль. RNN особенно полезны для задач, связанных с временными рядами или обработкой естественного языка, поскольку они могут моделировать зависимости между элементами в последовательности. RNN имеют уникальную архитектуру, которая позволяет им поддерживать скрытое состояние, которое фиксирует информацию с предыдущих временных шагов или элементов в последовательности.

Для создания модели глубокого обучения на основе рекуррентных слоев был использован популярный модуль TensorFlow, который служит библиотекой для написания и создания нейронных сетей. Ниже приведен код

для создания трехслойной рекуррентной сети для задачи бинарной текстовой классификации.

```
vocab_size = 10000
embedding_dim = 128
max_length = 100
num_classes = 2
hidden_units = 64

model = Sequential([
    Embedding(vocab_size, embedding_dim, input_length=max_length),
    SimpleRNN(hidden_units, return_sequences=True),
    SimpleRNN(hidden_units, return_sequences=True),
    SimpleRNN(hidden_units),
    Dense(num_classes, activation='softmax')
])
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

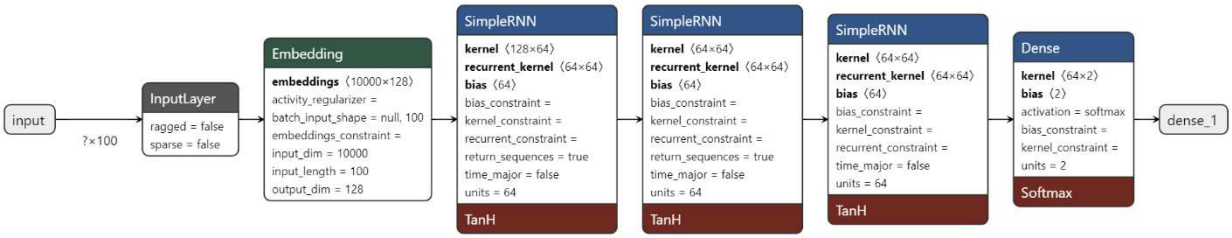


Рисунок 4.4 – Структура созданной рекуррентной нейронной сети

Во внедренной модели рекуррентной нейронной сети (RNN) для классификации текста мы использовали следующие слои: слой встраивания, слой SimpleRNN и слой Dense. Ниже приведен описание основных формул и вычислений, связанных с каждым слоем.

Слой встраивания, как было сказано ранее, используется для преобразования входных токенов (целых чисел, представляющих слова) в непрерывные векторы фиксированного размера. Учитывая размер словаря V (vocab_size) и мерность встраивания D (embedding_dim), уровень встраивания поддерживает обучаемую весовую матрицу W_emb размера (V x D). Для каждого входного токена i соответствующее вложение слова извлекается путем выбора i-й строки весовой матрицы.

Слой SimpleRNN:

$$h_t = \tanh(W_{hh} * h_{t-1} + W_{xh} * x_t + b_h)$$

где W_hh - матрица скрытого к скрытому весу, W_xh - матрица входного к скрытому весу, а b_h - скрытый вектор смещения. Функция tanh используется в качестве функции активации для уровня SimpleRNN. Это вычисление выполняется для каждого простого слоя, при этом выходное

скрытое состояние одного слоя служит входным для следующего слоя только с меньшими значениями.

Слой Dense:
y = W_yh * h_T + b_y
где W_yh - выходная весовая матрица, а b_y - выходной вектор смещения.

Затем выходной вектор y преобразуется с помощью функции активации softmax для получения окончательных вероятностей классификации:

p_i = exp(y_i) / sum(exp(y_j) for j in 1 to num_classes)
где p_i - вероятность принадлежности входного текста к i-й категории.

4.5 Создание Сверточных нейронных сетей для выявления кибербуллинга

В данном примере обучения нейронных сетей мы использовали такое дополнительное понятие как n-граммы, что означает зависимость не только слов, но и словосочетаний и связку нескольких слов. В теории это дало бы преимущество, так как в большинстве случаев слова по отдельности значат меньше, чем слова в связке. Например, слова «глупый», «недоносок», в весовом подсчете дают меньшую значимость по отдельности чем «глупый недоносок» в форме словосочетания. Для работы с со сверточными нейронными сетями были использованы 3 модификации n-грамм, где n означает количество слов в связке. Таким образом, были имплементированы и обучены униграммы(1), биграммы(2) и триграммы(3). По итогам экспериментов был сделан вывод, что самым оптмальным вариантом будет использование биграмм, так как с ростом количество слов в связке, возрастают и затрачиваемые ресурсы, и росто идет экспоненциальнй что ведет к увеличению времени обучения и усложнению модели. А усложнение модели может привести к переобуению. В таблице() приведена статистика использования сверточных нейронных сетей с использованием n-граммы.

Слои объединения: После слоев свертки требуется создать слои объединения, чтобы уменьшить пространственные размеры векторов после свертки, и сохранить наиболее важные объекты. Это также помогает снизить вычислительную сложность модели. В данном примере мы использовали max_pooling по следующим причинам:

1) Выбирая максимальное значение в пределах локального слоя, max_pooling обеспечивает определенную степень инвариантности перевода. Это означает, что если один и тот же объект появляется на входных данных в нескольких разных позициях, сеть все равно способна обнаружить и распознать его. К примеру формы разные слова «бегать».

2) Max_pooling также может обеспечить некоторую устойчивость к шуму во входных данных и отбрасывает более слабые активации, которые могут быть вызваны шумом. В других вариантах нейронных сетей для этого требовалось бы добавить слой dropout, что несомненно повышало бы сложность модели
3) Max_pooling позволяет сети изучать иерархические зависимости, создавая многомасштабное представление входных данных. Таким образом можно найти сначала главные слова, а после связать их с менее значимыми для обнаружения скрытых зависимостей.

```
model = Sequential([
    Embedding(vocab_size, embedding_dim, input_length=max_length),
    Conv1D(128, 5, activation='relu'),
    GlobalMaxPooling1D(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Итоговая созданная модель выглядит следующим образом:

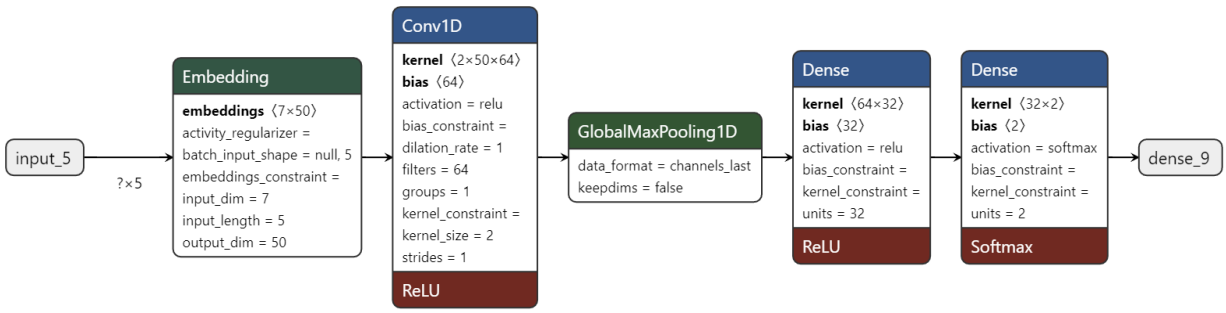


Рисунок 4.5 – Визуальное представление сверточной нейронной сети

Математические формулы использованные в этой модели:

$$C[i] = \text{sum}(F * E[i:i+k-1]) + b \quad (3.4)$$

где C[i] - выходные данные свертки по позициям i, F - матрица фильтра, E[i:i+k-1] - входная подматрица для фильтра в позиции i, k - размер фильтра (размер ядра[1,2,3]) и b - смещение.

$$R[i] = \max(0, C[i]) \quad (3.5)$$

где R[i] - выходной сигнал функции активации по позициям i, а C[i] - входной сигнал от уровня свертки.

$$P = \max(R) \quad (3.6)$$

где P - выходные данные объединяющего слоя, а R - входные данные функции активации.

$$D = A(W * P + b) \quad (3.7)$$

где D - выходные данные плотного слоя, A - функция активации (в данном случае ReLU), W - матрица изученных весов, P - входные данные из объединяющего слоя и b - значение смещения.

$$O = \text{softmax}(W * D + b) \quad (3.8)$$

где O - выходной вектор вероятности принадлежности к классам, softmax - функция активации на последнем слое, W - изученная весовая матрица, D - входные данные из предыдущего плотного слоя и b - значение смещения.

Таблица 4.1 – Сравнительный анализ использования n-gram

	униграммы	биграммы	триграммы
Точность	0.86	0.88	0.87
Потеря	0.649848	0.627139	0.610562
Время обучения (сек)	63.9	80	67

В данной таблице показаны сравнительные характеристики униграмм, биграмм и триграмм. Как видно из результатов, работа с биграммами дает лучшие результаты, но теряется преимущество во времени обучения. По логике время обучения у триграмм должно было быть больше чем у биграмм, но такое произошло потому что самих наборов по три слова в итоговом векторе получилось меньше, что снизило время обработки. Далее пробовать квадрограммы, квинтограммы и т.д. смысла нет, так как есть некая золотая середина в большинстве случаев использование биграмм и триграмм дают лучшие результаты.

4.6 Нейронные сети с долговременной кратковременной памятью (LSTM)

Длительная кратковременная память. Долговременная кратковременная память (LSTM) - это тип рекуррентной нейронной сети, которая способна успешно сохранять информацию в течение длительного периода времени, включая “ячейку памяти”. "Входной элемент", "элемент забывания" и "выходной элемент" в первую очередь отвечают за управление поведением ячейки памяти. Элемент ввода отвечает за активацию процесса ввода информации в ячейку памяти, в то время как элемент забывания отвечает за выборочное стирание некоторой информации, которая хранится в ячейке памяти, и активацию хранилища при подготовке к последующему вводу. И последнее, но не менее важное: информация, которая будет отправлена из ячейки памяти, определяется выходным элементом. Приведен код на Python для создания нейронной сети с использованием длительной кратковременной памяти.

```
vocab_size = 10000
embedding_dim = 128
max_length = 100
num_classes = 5
hidden_units = 64

model = Sequential([
   Embedding(vocab_size, embedding_dim, input_length=max_length),
   LSTM(hidden_units, return_sequences=True),
   LSTM(hidden_units),
   Dense(num_classes, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Приведем пример формул, которые используются при расчете весов в данной модели. Так как слой встраивания используется во всех нейронных сетях, не стоит каждый раз описывать его, поэтому пропустим описание этого слоя.

Далее после слоя встраивания идет слой долговременной кратковременной памяти. Этот слой предназначен для обработки входных последовательностей по одному токену за раз, сохраняя скрытое состояние h и состояние ячейки c. На каждом временном шаге t слой принимает входное вложение x_t, предыдущее скрытое состояние h_{t-1} и предыдущее состояние ячейки c_{t-1} и вычисляет новое скрытое состояние h_t и новое состояние ячейки c_t, используя следующие формулы:

$$i_t = \text{sigmoid}(W_{xi} * x_t + W_{hi} * h_{t-1} + b_i)$$
$$f_t = \text{sigmoid}(W_{xf} * x_t + W_{hf} * h_{t-1} + b_f)$$
$$o_t = \text{sigmoid}(W_{xo} * x_t + W_{ho} * h_{t-1} + b_o)$$
$$g_t = \tanh(W_{xg} * x_t + W_{hg} * h_{t-1} + b_g)$$

$$c_t = f_t * c_{t-1} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

где W_xi, W_hi, W_xf, W_hf, W_xo, W_ho, W_xg и W_hg - весовые матрицы, а b_i, b_f, b_o и b_g - векторы смещения. Функция sigmoid используется для элементов ввода, забывания и вывода, в то время как функция tanh используется для ввода ячейки и активации состояния ячейки. Это вычисление выполняется для каждого слоя LSTM, при этом выходное скрытое состояние одного слоя служит входным значением для следующего слоя.

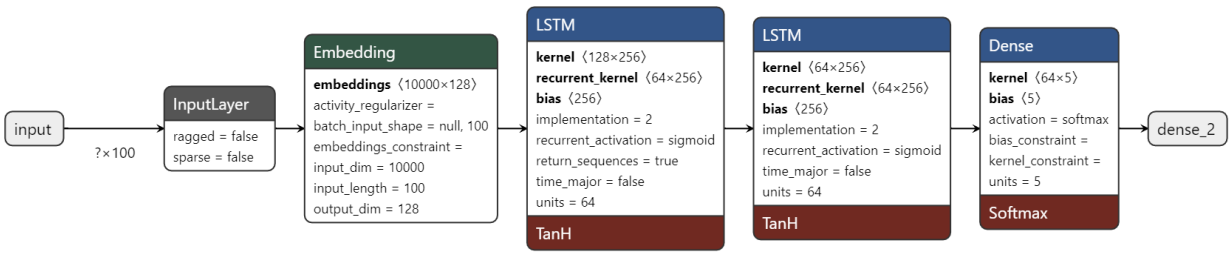


Рисунок 4.6 – Схема нейронной сети с кратковременной долговременной памятью

4.7 Двунаправленные нейронные сети с долговременной кратковременной памятью(Bi-LSTM)

Для внедрения двунаправленной нейронной сети с долговременной кратковременной памятью. Ниже приведенный код создает двухслойную модель Bi-LSTM, использующую двунаправленную оболочку вокруг слоя LSTM. Как видно, по кодам, в создании двух нейронных сетей Bi-LSTM и LSTM нет особой разницы, различия только в названиях самих слоев.

```
model = Sequential([
    Embedding(vocab_size, embedding_dim, input_length=max_length),
    Bidirectional(LSTM(hidden_units, return_sequences=True)),
    Bidirectional(LSTM(hidden_units)),
    Dense(num_classes, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Главное отличие LSTM от Bi-LSTM заключается в том, что Bi-LSTM обрабатывает входную последовательность как в прямом, так и в обратном направлениях. Он состоит из двух отдельных LSTM, один из которых обрабатывает входную последовательность от начала до конца (прямой LSTM), а другой обрабатывает входную последовательность от конца до начала (обратный LSTM). Скрытые состояния обоих LSTM на каждом временном шаге объединяются для формирования конечного выходного скрытого состояния.

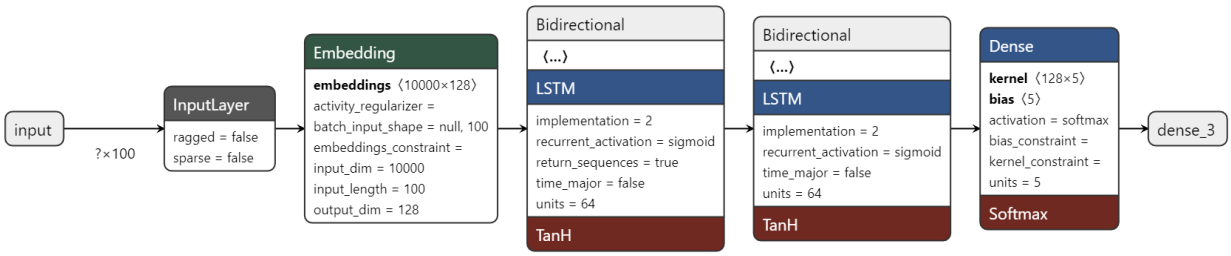


Рисунок 4.7 – Схема двунаправленной нейронной сети с кратковременной долговременной памятью

4.8 Создание многослойного персептрона для выявления кибербуллинга

По структуре многослойный персептрон похож простую глубокую нейронную сеть, по этой причине, мы изменили структуру MLP следующим образом: добавили количество слоев, 2 дополнительных слоя и перед добавленными слоями добавили функцию dropout. Функция dropout сама по себе ничего не делает, она просто удаляет не активированные нейроны[65]. В условии задаем удалить те нейроны, которые не достигли значения 0,5. Это означает то, что нейроны, которые после функции активации не дошли до значения 0,5 будут автоматически удалены и не подлежат допуску к следующему слою. Таким образом затрачиваемые вычислительные ресурсы на первых 3 слоях возрастают, но dropout посредством удаления помогают нам снизить эти расходы на следующих слоях. В итоге у нас получилась более сложная модель, но по вычислительным затратам не сильно превосходит простую нейронную сеть. Ниже представлен код для создания многослойного персептрона с использованием плотных слоев.

```
vocab_size = 10000
num_classes = 5
hidden_units = 64
dropout_rate = 0.5

model = Sequential([
    Dense(hidden_units, activation='relu', input_shape=(vocab_size,)),
    Dense(hidden_units, activation='relu'),
    Dropout(dropout_rate),
    Dense(hidden_units, activation='relu'),
    Dropout(dropout_rate),
    Dense(num_classes, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

В созданной модели многослойного персептрона структура состоит из входного слоя, трех скрытых слоев, двух слоев выброса и выходного слоя.

Входной слой: Для каждого входного вектора x и слоя k плотный слой вычисляет выходной y следующим образом:

$$y = f(W_k * x + b_k)$$

где W_k - весовая матрица, b_k - вектор смещения, а f - функция активации (в данном случае ReLU).

Выходной слой: Для этого слоя вычисление проходят аналогично входному и скрытому слоям, с основным отличием в функции активации. Выходной слой использует функцию активации softmax для получения вероятностей для каждого класса:

$$y = softmax(W_out * h + b_out)$$

где W_out - выходная весовая матрица, b_out - выходной вектор смещения, h - выходные данные последнего скрытого слоя.

softmax(z_i) = exp(z_i) / sum(exp(z_j) for j in 1 to num_classes)

Данная модель обучается с использованием обратного распространения и алгоритма оптимизации (Adam), чтобы минимизировать потери категориальной перекрестной энтропии между прогнозируемыми вероятностями и истинными значениями классов. Веса и смещения всех плотных слоев обновляются в процессе тренировки. Созданная модель многослойного персептрона выглядит следующим образом:

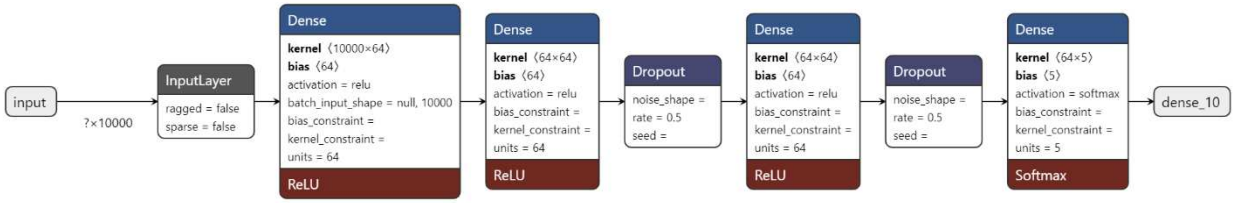


Рисунок 4.8 – Схема многослойного персептрона

4.9 Создание глубокой нейронной сети с использованием механизма внимания

```
vocab_size = 10000
embedding_dim = 128
max_length = 100
num_classes = 2
hidden_units = 64
dropout_rate = 0.5

inputs = Input(shape=(max_length,))
embedding = Embedding(vocab_size, embedding_dim, input_length=max_length)(inputs)
bi_lstm = Bidirectional(LSTM(hidden_units, return_sequences=True))(embedding)
attention = Attention()(bi_lstm)
dropout = Dropout(dropout_rate)(attention)
flat = Flatten()(dropout)
dense = Dense(num_classes, activation='sigmoid')(flat)

model = Model(inputs=inputs, outputs=dense)

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Созданная реализованная нейронная сеть, основанная на внимании, состоит из слоя встраивания, двунаправленного слоя LSTM, пользовательского слоя внимания, слоя отсева и плотного слоя для двоичной классификации.

4.10 Итоговая модель нейронной сети с использованием механизма внимания

По итогам исследования была разработана модель нейронной сети кратковременной долговременной памяти с использованием механизма внимания, которая дала лучшие результаты. Ниже представлена архитектура работы всего процесса, от входа сырых текстовых данных до конечного предсказания принадлежности к классу кибербуллинга.

Как обсуждалось ранее механизм внимания может быть использован в модели классификации, чтобы помочь модели «сосредоточиться» на различных частях входной последовательности при составлении прогнозов принадлежности к классам. Это позволяет модели узнать, какие части входных данных более важны для задачи классификации, и соответствующим образом взвесить их.

Полная структура работы предлагаемой модели выглядит как показано на рисунке (). Данный процесс включает в себя все процессы от входа сырого текста до конечного прогнозирования принадлежности к классу. Далее представлено описание каждой стадии.

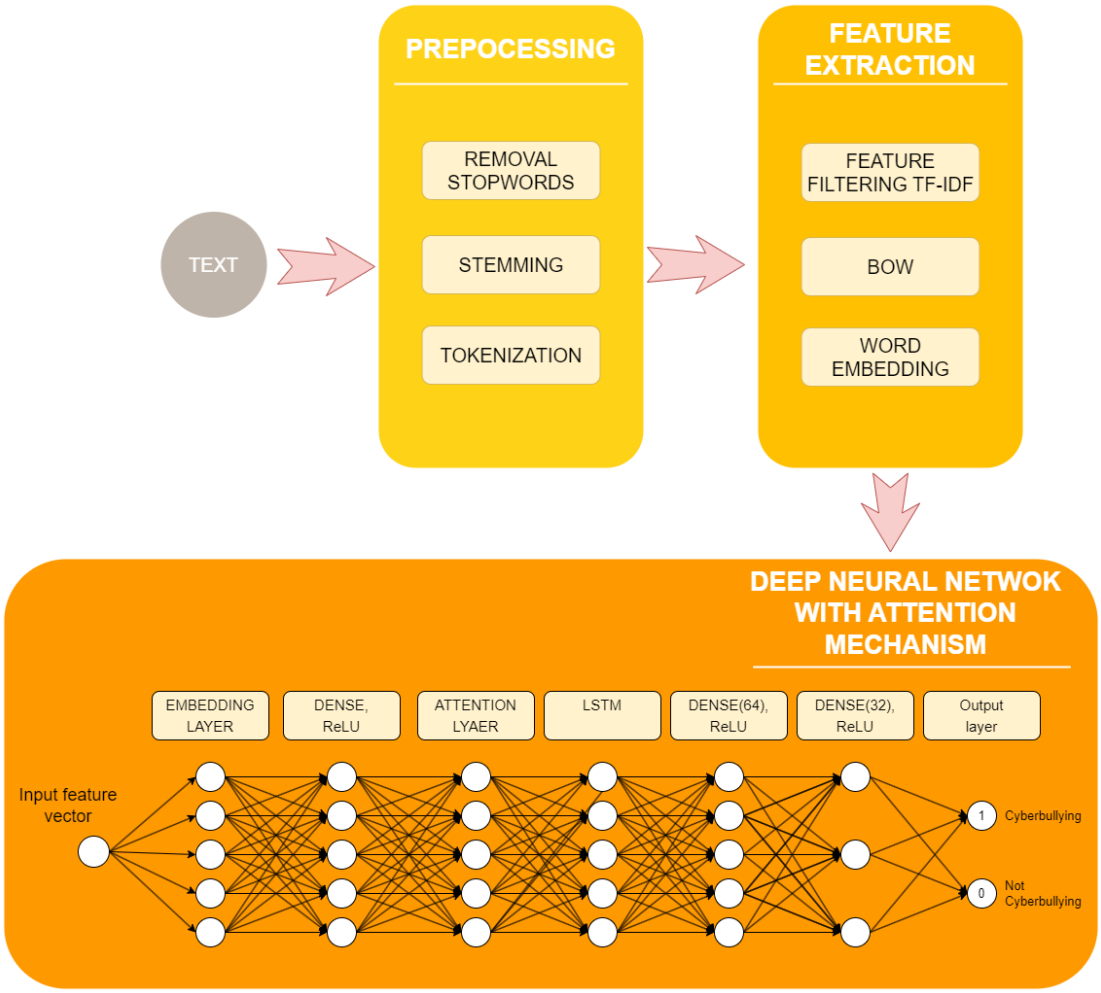


Рисунок 4.9 – Полная схема внедрения нейронной сети с механизмом внимания

4.10.1 Первичная обработка сырого текста

В начале мы уже упомянули, что алгоритмы обучения не могут работать с текстовыми данными напрямую поэтому мы должны сначала перевести их в числовой формат. В данном исследовании мы использовали несколько поэтапных процессов, чтобы получить итоговый набор слов.

Удаление стоп слов. Основная цель удаления стоп-слов - уменьшить размерность текстовых данных и сосредоточиться на более значимых словах, которые несут наиболее релевантную информацию для конкретной задачи. Это может привести к следующим преимуществам: снижение вычислительных затрат, улучшенная производительность модели, уменьшение шума. Все эти определения мы приводили ранее.

```
import nltk
from nltk.corpus import stopwords
import pandas as pd
nltk.download('stopwords')

stop_words = set(stopwords.words('kazakh'))

data = pd.read_csv('Dataset.csv')
word_tokens = nltk.word_tokenize(sample_text)
filtered_text = [word for word in word_tokens if word.lower() not in stop_words]

filtered_text = ' '.join(filtered_text)
print(filtered_text)
```

Стемминг. Повторим, что стемминг - это метод предварительной обработки текста в обработке естественного языка, который сводит слова к начальному виду. Целью stemming является нормализация текстовых данных путем приведения изменяемых или производных слов к их основному значению, что может помочь улучшить производительность моделей за счет уменьшения объема словаря и рассмотрения различных форм слова как одного и того же лексемы.

К примеру слова бегать, бегающий, бегал, бегает. После применения стемминга эти слова будут сведены к их общей основе "бег".

Пример кода, использованный нами приведен ниже:

```
import nltk
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

words = data_stop_words

stemmed_words = [stemmer.stem(word) for word in words]
```

```
for word, stemmed_word in zip(words, stemmed_words):
    print(f"{word} -> {stemmed_word}")
```

Токенизация. Повторим, что токенизация - это процесс разбиения текстовых данных на более мелкие единицы, называемые токенами, обычно это слова или фразы. Токенизация является важным этапом предварительной обработки во многих задачах обработки естественного языка (NLP), таких как классификация текста, анализ настроений и поиск информации. Преобразуя текст в токены, моделям NLP становится легче анализировать и обрабатывать данные. В данном исследовании мы использовали способ пословной токенизации, чтобы каждое слово входило в алгоритм как отдельная единица информации, в котором хранится определенное значение.

```
sentence_tokens = nltk.sent_tokenize(filtered_text)

print("Original text:")
print(sample_text)
print("\nTokenized sentences:")
print(sentence_tokens)
```

Так как в Python есть готовая функция *tokenize,* нам не надо было создавать отдельную функцию, а достаточно вызвать и передать набор данных, который прошел через 2 предыдущих процесса.

В конце этой стадии работы, мы из набора данных, который состоит из необработанных слов, мы получили упорядоченный набор токенов, который могут быть введены в алгоритмы машинного и глубокого обучения.

4.10.2 Выборка параметров

TF-IDF. TF-IDF расшифровывается как частота термина - обратная частоте документа. В процессе классификации текстов, связанных с кабербуллингом, выделение признаков является важным этапом. Чтобы извлечь функции для модели, мы использовали подходы TF-IDF и Word2Vec. Этот метод извлечения текстовых объектов использует статистику слов в качестве основы и представляет собой гибрид двух других алгоритмов, известных как term frequency (TF) и inverse document frequency (IDF), что расшифровывается как term frequency - обратная частота документа. Этот подход просто учитывает те словосочетания, которые согласуются во всех текстах [36]. В результате алгоритм TF-IDF является одним из методов извлечения признаков, который часто применяется при обнаружении текста [37]. Для обработки текста Word2Vec представляет собой нейронную сеть с двумя слоями, которая "векторизирует" слова. Он принимает корпус текста в качестве входных данных и выдает набор векторов в качестве выходных данных, которые являются векторами атрибутов, представляющими слова внутри этой структуры [38]. Методика Word2Vec строит многомерный вектор

для каждого слова, используя модель Skip-gram, непрерывный пакет слов (CBoW) и два скрытых слоя неглубоких моделей [39]. Цель состоит в том, чтобы повысить вероятность того, что:

$$\arg\max_{\theta} \prod_{w \in T} \left[\prod_{c \in C} p(c \mid w; \theta) \right] \tag{4}$$

Word2Vec Embedding. Word2vec - это инструмент для встраивания слов. Это одновременно и эффективно, и энергозатратно. Word2vec использует двухслойную языковую модель нейронной сети для обучения векторным представлениям каждого отдельного слова. На самом деле, приложение включает в себя такие различные модели, как CBoW и Skip-gram. Bow пытается предсказать слово на основе слов, которые его окружают, в то время как Skip-gram пытается предсказать набор слов, основанный только на одном слове. Word2vec способен обучаться на крупномасштабном необъявленном корпусе, несмотря на ограниченные вычислительные ресурсы, благодаря удивительно эффективному дизайну и используемой в нем методике обучения без присмотра. Вложения Word2vec могут быть изучены, и с помощью этого процесса могут быть представлены важные лингвистические связи между словами. Таким образом имплементируя этот процесс мы выявили ключевые зависимости и более важные и менее важные векторы.

Bag-of-Words это способ представления текстовых данных при моделировании. Алгоритмы машинного обучения не способны напрямую обрабатывать сырой текст, поэтому необходимо преобразовать его в числовой формат, точнее, в числовые векторы. При обработке текста для целей машинного обучения, текстовые данные преобразуются в числовые векторы, которые отражают различные языковые характеристики текста. Этот процесс известен как извлечение или кодирование признаков. Метод Bag of Words (BoW) является одним из таких способов. Здесь мы использовали набор биграмм для работы с мешком слов. Термины с частотой встречаемости в документах, которая меньше двух, полностью игнорируются. Несколько различных систем взвешивания терминов, таких как tf-idf и бинарные, также являются жизнеспособными вариантами в этом контексте [41]. Система взвешивания tf-idf - это та, которую мы используем в данном конкретном исследовании. Следующая формула используется для получения веса tf-idf, соответствующего i-му слову в j-м документе:

$$w_{i,j} = TF_{i,j} \times \log\left(\frac{N}{DF_i}\right) \tag{5}$$

4.10.3 Создание и обучение нейронной сети

В главе 3 мы описывали структуру и принцип работы механизма внимания. А здесь представлена модель глубокого обучения, которая

использует эту схему в одном из слоев для каждого нейрона и выдает на следующий слой набор из более «важных» веткоров, которые могут переносить релевантную информацию в себе.

Опишем сначала созданный класс Attention:

```
class Attention(Layer):
    def __init__(self, **kwargs):
        super(Attention, self).__init__(**kwargs)

    def build(self, input_shape):
        self.W_q = self.add_weight(shape=(input_shape[-1], input_shape[-1]),
initializer='glorot_uniform', trainable=True)
        self.W_k = self.add_weight(shape=(input_shape[-1], input_shape[-1]),
initializer='glorot_uniform', trainable=True)
        self.W_v = self.add_weight(shape=(input_shape[-1], input_shape[-1]),
initializer='glorot_uniform', trainable=True)
        super(Attention, self).build(input_shape)

    def call(self, inputs):
        q = tf.matmul(inputs, self.W_q)
        k = tf.matmul(inputs, self.W_k)
        v = tf.matmul(inputs, self.W_v)
        logits = tf.matmul(q, k, transpose_b=True)
        a = Activation('softmax')(logits)
        return tf.matmul(a, v)
```

Здесь функция build отвечает за создание и инициализацию трех обучаемых весовых матриц: W_q (запрос), W_k (ключ) и W_v (значение). Эти весовые матрицы используются для вычисления показателей внимания и результирующего контекстного вектора в методе call. Форма каждой весовой матрицы равна (input_shape[-1], input_shape[-1]), что означает, что количество строк и столбцов равно количеству объектов во входном тензоре.

Функция call возвращает контекстный вектор, который может быть обработан последующими слоями в нейронной сети. Механизм внимания позволяет модели выборочно фокусироваться на релевантных частях входных данных.

Далее распишем саму построенную модель, которая состоит из следующих слоев:

1) Входной слой: Этот слой определяет входную форму модели. Здесь входной слой принимает последовательность целых. Входной текст распределяется на 64 нейрона.
2) Слой Embedding: Этот слой преобразует численные входные данные в плотное векторное пространство, где каждое целое число соответствует

уникальному векторному представлению. В этом примере слой сопоставляет каждое целое число с 64-мерным вектором.

3) Слой внимания: Этот уровень реализует механизм внимания, который оценивает важность каждого элемента в последовательности на основе его доносимой информации. Слой внимания в этом примере использует мехнизм внимания от Keras, который принимает в качестве входных данных выходные данные LSTM и возвращает контекстный вектор, который фиксирует соответствующую информацию во входной последовательности.
4) Слой LSTM: Этот слой применяет алгоритм долговременной кратковременной памяти (Long Shor-term Memory) к входным данным, который позволяет модели фиксировать временные зависимости между элементами последовательности.
5) Выходной слой: Этот уровень применяет “густой” слой с функцией активации softmax к выходам LSTM, который генерирует распределение вероятностей по возможным классам выходных данных.
6) Компиляция модели: Модель создана с использованием алгоритма оптимизации Adam и разреженной(sparse) категориальной функции потери кросс-энтропии, которая обычно используется для задач бинарной и многоклассовой классификации.

После компиляции модели мы получили следующую структуру:

Layer (type)	Output Shape	Param #
embedding_9 (Embedding)	(None, 100, 128)	1280000
time_distributed_1 (TimeDistributed)	(None, 100, 64)	8256
attention_4 (Attention)	(None, 100, 64)	12288
lstm_8 (LSTM)	(None, 64)	33024
dense_16 (Dense)	(None, 64)	4160
dropout_4 (Dropout)	(None, 64)	0
dense_17 (Dense)	(None, 32)	2080
dropout_5 (Dropout)	(None, 32)	0
dense_18 (Dense)	(None, 1)	33

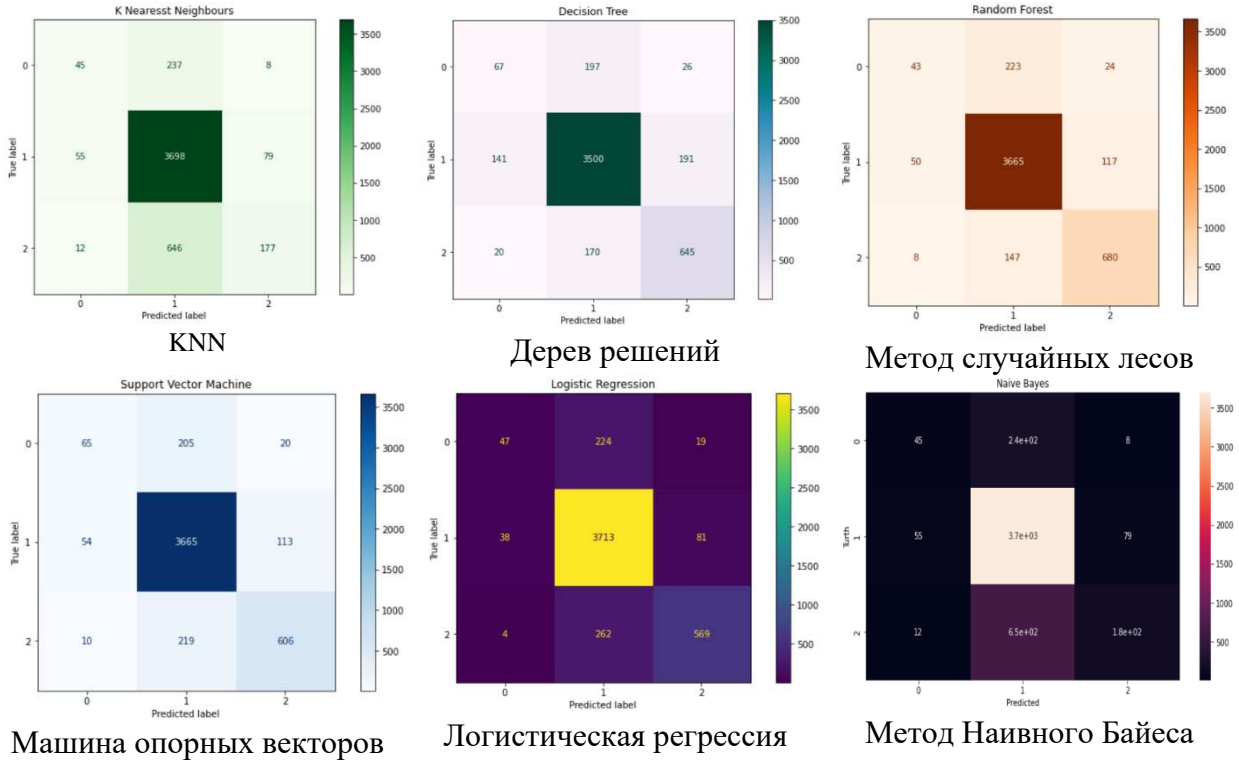
Total params: 1,339,841
Trainable params: 1,339,841
Non-trainable params: 0

Рисунок 4.10 – Отчет компиляции предлагаемой модели нейронной сети

В итоге у нас получилось создать нейронную сеть с почти полутора миллионами обучаемых параметров. Остается лишь передать модели предобработанные данные и запустить обучение.

5 РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ

Здесь показаны результаты применения алгоритмов машинного обучения в наборе данных Twitter. После обучения и тестирования всех классических моделей для задачи бинарной классификации мы получили результаты, как показано ниже. Также важно отметить, что confusion matrix не отображают результаты, а показывают количество правильно или неправильно классифицированных объектов. По этой причине мы использовали все остальные показатели для проведения некоторого сравнительного анализа в задаче бинарной классификации для обнаружения кибербуллинга. Анализ включает в себя: accuracy, precision, recall, F1-scrore и ROC-AUC.



KNN | Дерев решений | Метод случайных лесов

Машина опорных векторов | Логистическая регрессия | Метод Наивного Байеса

Рисунок 5.1. Confusion matrix алгоритмов машинного обучения

Во-первых, анализ показывает, что все 6 протестированных алгоритмов машинного обучения показывают примерно одинаковые результаты.

В заключении мы сравнили результаты алгоритмов машинного обучения, используемых для обнаружения кибербуллинга, в одном и том же наборе данных Twitter. В этом эксперименте мы протестировали 6 классических алгоритмов машинного обучения. Но после нескольких анализов выяснилось, что 4 из них (SVM, DT, Случайный лес и KNN) лучше всего справляются с задачей бинарной классификации.

Ниже предоставлены результаты внедрения алгоритмов глубокого обучения. Далее показан сравнительный анализ преимущества алгоритмов глубокого обучения по сравнению с машинным обучением.

5.2 Результаты обучения рекуррентных нейронных сетей

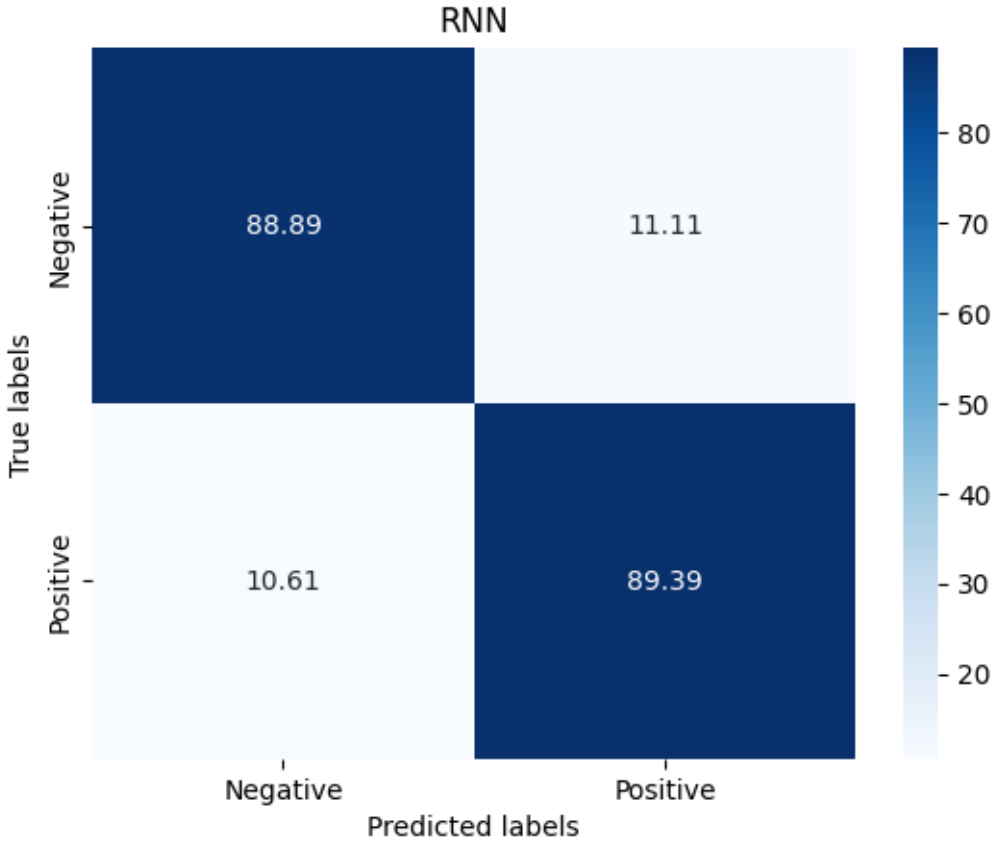


Рисунок 5.2 – Confusion matrix для рекуррентной нейронной сети

Как видно из показанного графика, здесь показаны результаты бинарной классификации рекуррентной нейронной сети. По главной диагонали расположены количество правильно угаданных объектов, для удобства мы вывели показатели не как количество объектов, а их процент по отношению к общему количеству объектов каждого класса, соответственно по побочной диагонали процент объектов, маркированных алгоритмом неправильно. Обычно алгоритмы машинного обучения доходили максимум до 87% точности, здесь же преимущество использования глубокого обучения на лицо. Конечная модель рекуррентной нейронной сети в среднем дала аккуратность 89%. И это не предел.

Таблица 5.1 – отчет классификации для рекуррентной нейронной сети

Class	Precision	Recall	F1-score
1	87.5	88.5	88.0
0	88.1	88.5	88.2
Weighted Avg	87.8	88.5	88.1

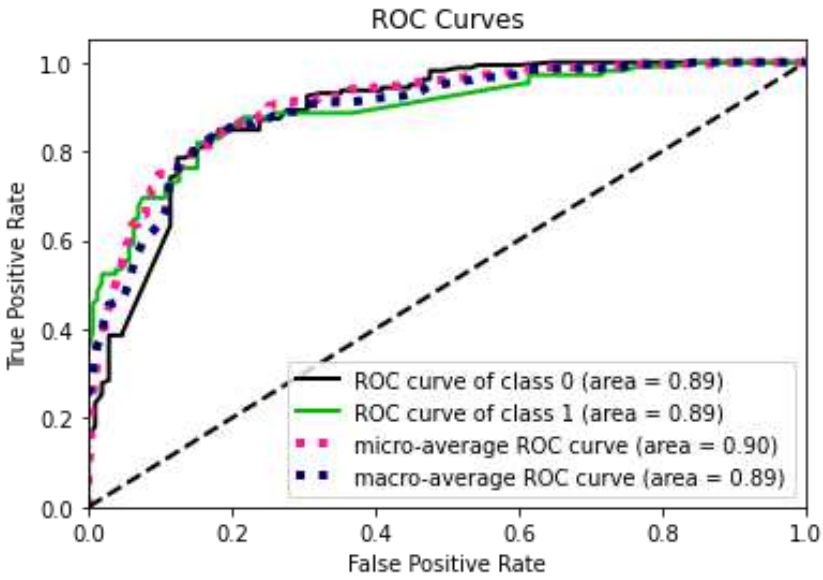


Рисунок 5.3 – ROC-AUC для рекуррентной нейронной сети

Как видно из предоставленных метрик, модель рекуррентной нейронной сети показывает достаточно неплохие результаты с учетом того, что была создана небольшая модель. Если поработать с гиперпараметрами, то предположительно можно получить еще большие результаты, так как данная модель не является лучшей и может быть в дальнейшем усовершенствована.

5.3 Результаты использования глубоких нейронные сети

Как и в предыдущей модели, далее будут представлены результаты обучения простой глубокой нейронной сети в задаче бинарной классификации.

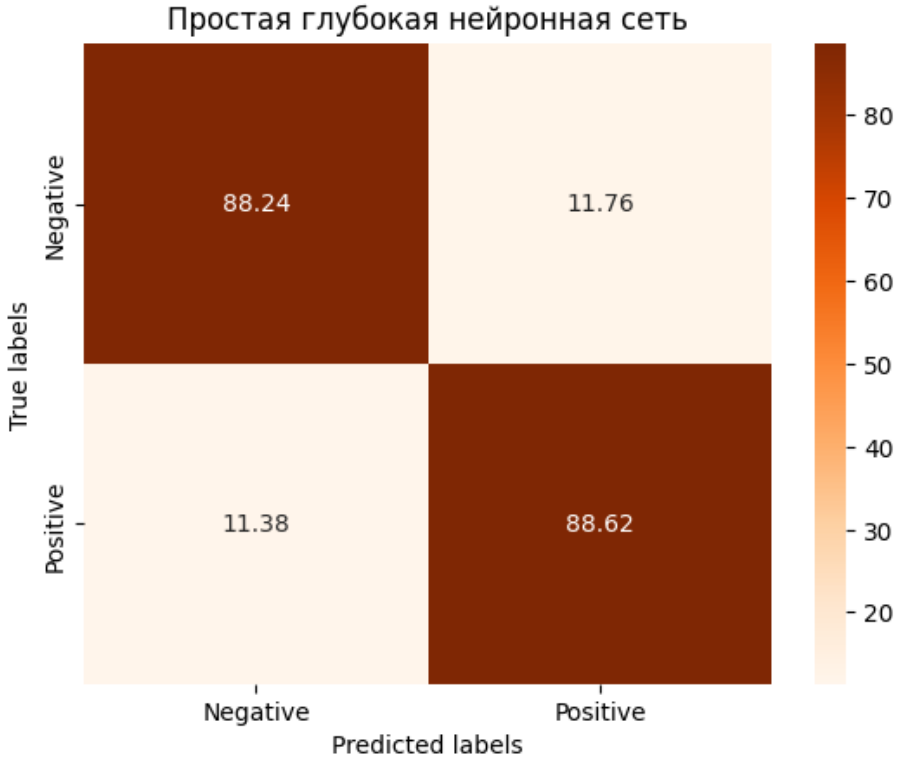


Рисунок 5.4 – Confusion matrix для простой глубокой нейронной сети

Таблица 5.2 – Результаты классификации глубокой нейронной сети

Class	Precision	Recall	F1-score
1	87.0	87.9	87.3
0	87,8	87.9	87.6
Weighted Avg	87.4	87.9	87.5

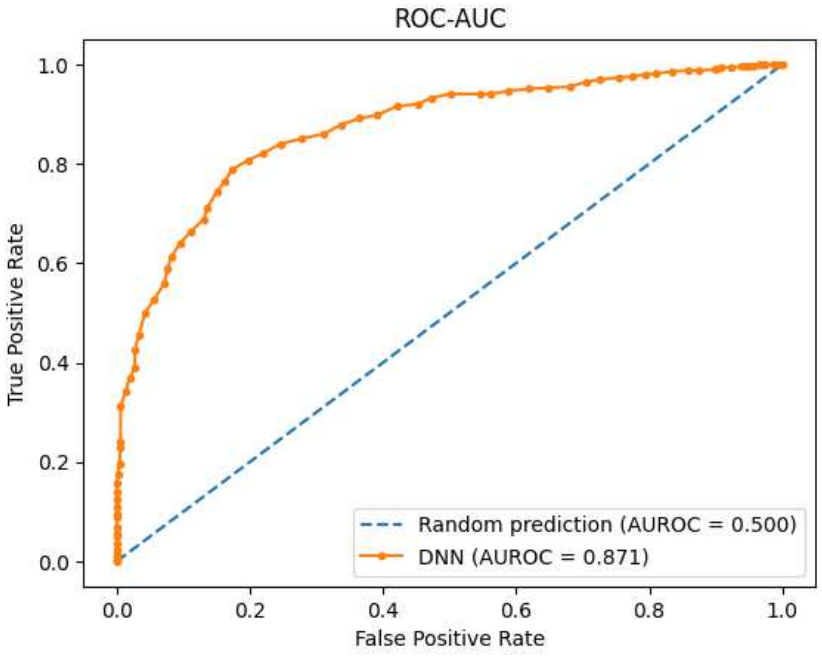


Рисунок 5.5 – ROC-AUC для простой глубокой нейронной сети

Таким образом простая нейронная сеть показала примерно такие же результаты как и рекуррентные нейронные сети. Стоит полагать, что такой результат был достигнут именно благодаря иерархической структуре нейронной сети, которая постепенно уменьшала мерность входных данных на каждом слое[54]. Структура данной нейронной сети представлена и описана в главе 2.

5.4 Результаты внедрения сверточных нейронных сетей в выявлении кибербуллинга

Таблица 5.3 – Сравнительный анализ использования n-gram

	униграммы	биграммы	триграммы
Точность	0.86	0.88	0.87
Потеря	0.649848	0.627139	0.610562
Время обучения (сек)	63.9	80	67

В данной таблице показаны сравнительные характеристики униграмм, биграмм и триграмм. Как видно из результатов, работа с биграммами дает лучшие результаты, но теряется преимущество во времени обучения. По логике время обучения у триграмм должно было быть больше чем у биграмм, но такое произошло потому что самих наборов по три слова в итоговом векторе получилось меньше, что снизило время обработки. Далее пробовать квадрограммы, квинтограммы и т.д. смысла нет, так как есть некая золотая середина в большинстве случаев использование биграмм и триграмм дают лучшие результаты. Ниже приведены итоговые результаты по имплементации сверточных нейронных сетей с использованием биграмм с показанием различных метрик.

По аналогии с предыдущими примерами нейронных сетей ниже приведены результаты.

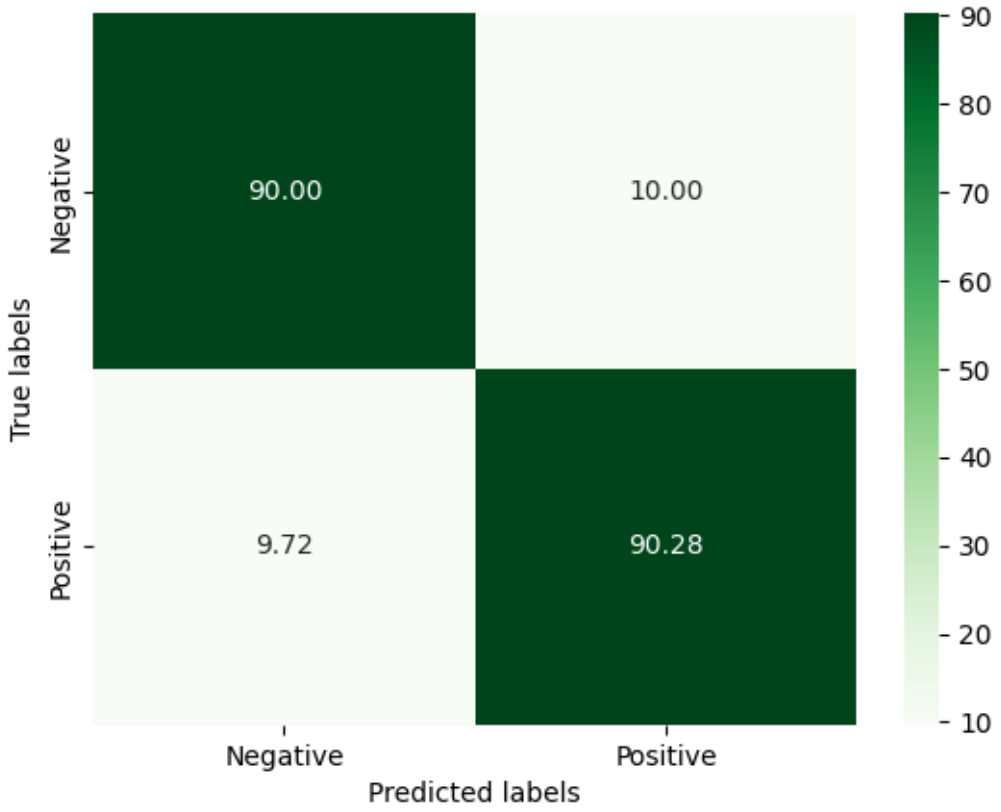


Рисунок 5.6 – Confusion matrix для сверточной нейронной сети

Таблица 5.4 – Классификационный отчет по результатам сверточной нейронной сети

Class	Precision	Recall	F1-score
1	89.0	91.0	89.5
0	89.5	88.9	89.6
Weighted Avg	89.5	88.9	89.2

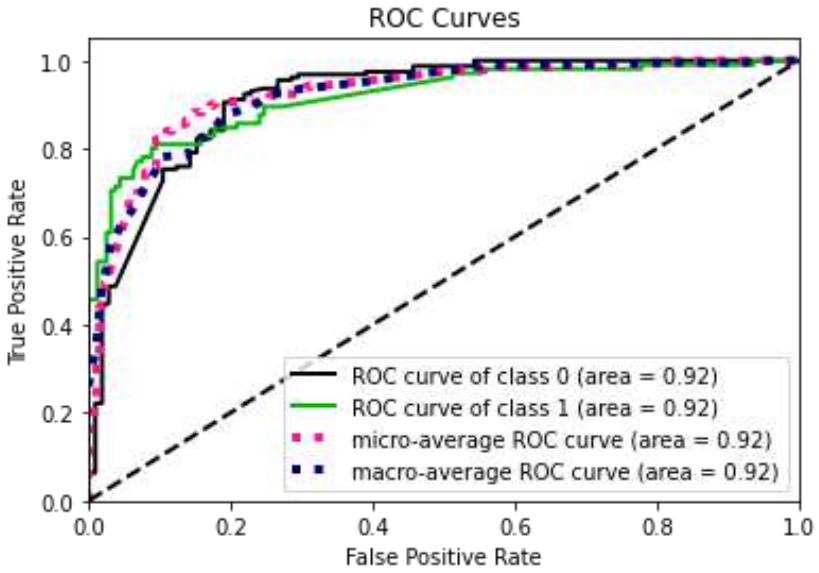


Рисунок 5.7 – Roc-AUC для сверточной нейронной сети

5.5 Результаты внедрения моделей кратковременной долговременной памяти в задаче классификации

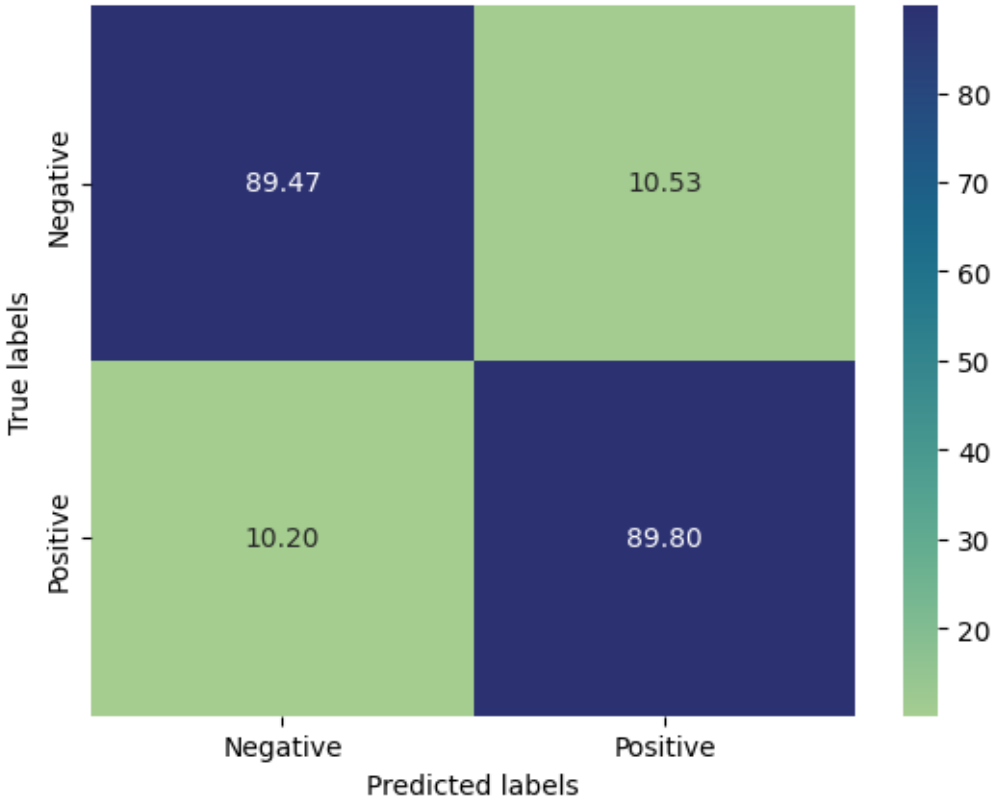


Рисунок 5.8 – Confusion matrix для нейронной сети кратковременной долговременной памяти

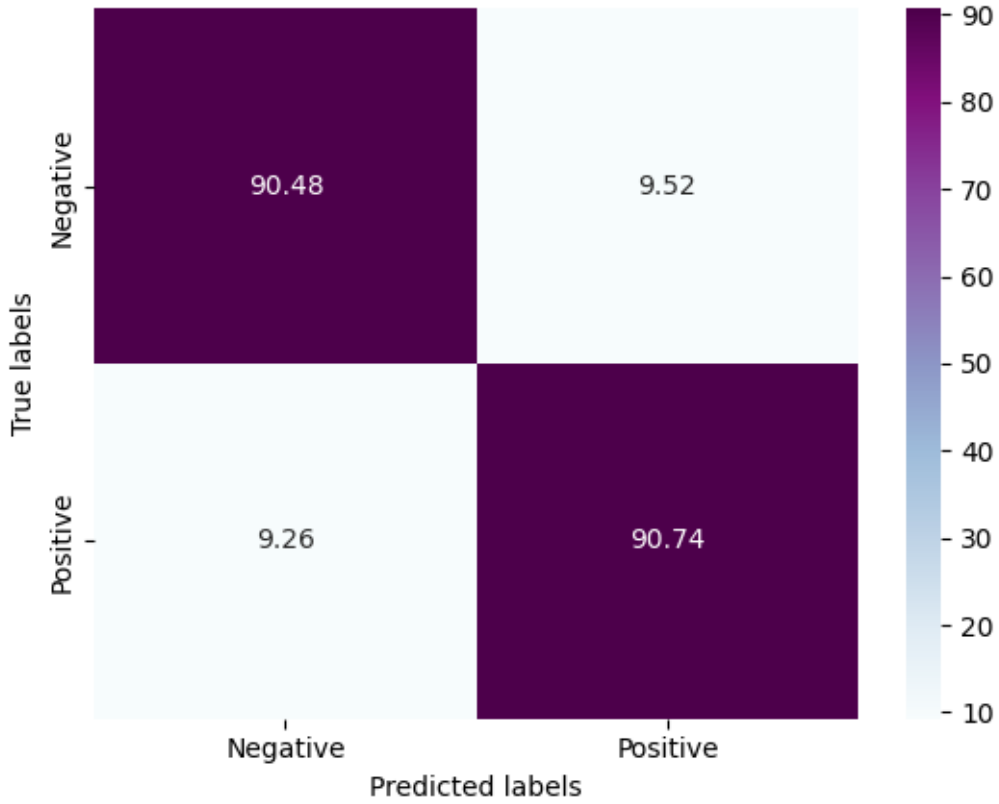


Рисунок 5.9 – Confusion matrix для двунапралвенной нейронной сети кратковременной долговременной памяти

Таблица 5.5 - Классификационный отчет по результатам двунапралвенной нейронной сети кратковременной долговременной памяти

Class	Precision	Recall	F1-score
1	90	92	90.4
0	90.2	91.8	90.8
Weighted Avg	91	90.5	91

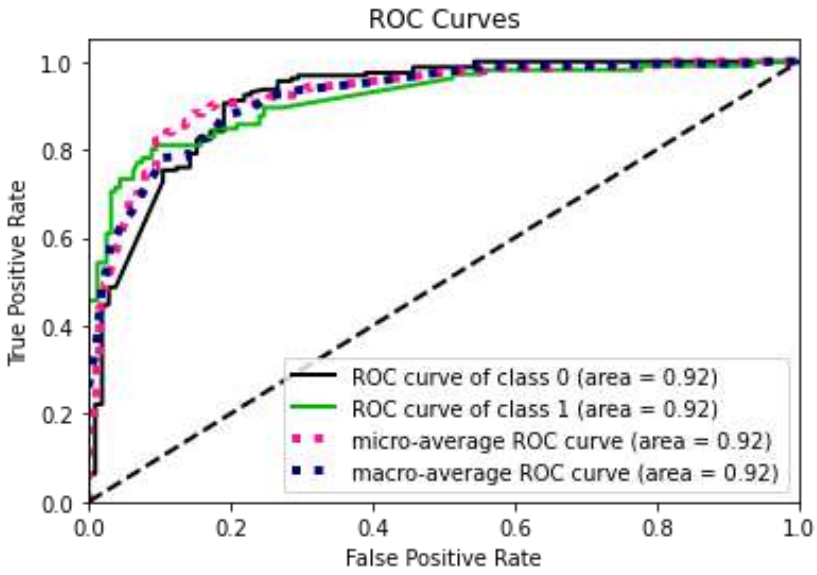


Рисунок 5.10 – ROC-AUC для двунаправленной нейронной сети кратковременной долговременной памяти

Таблица 5.6 - Классификационный отчет по результатам нейронной сети Bi-LSTM

Class	Precision	Recall	F1-score
1	89.5	90	89.7
0	89.7	89	89.2
Weighted Avg	89.5	90.5	89.7

Как видно по результатам использование LSTM и Bi-LSTM дало впервые выйти на уровень точности 90%+. На момент это был лучший результат, который был достигнут. А в сравнении с алгоритмами машинного обучения прирост составил от 5% до 10%, что является достаточно высоким показателем.

5.6 Результаты внедрения многослойного персептрона

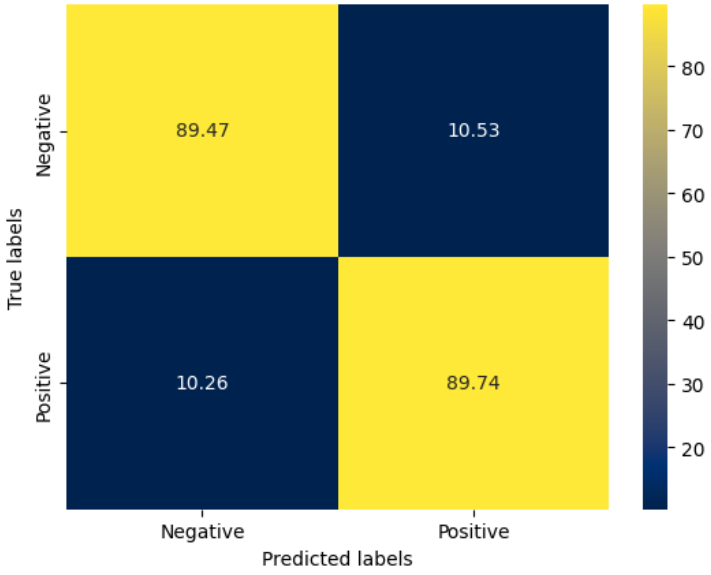


Рисунок 5.11 – Confusion matrix для многослойного персептрона

Таблица 5.7 - Классификационный отчет по результатам многослойного персептрона

Class	Precision	Recall	F1-score
1	90	89	89.7
0	89.5	91	90
Weighted Avg	89.5	90	89.7

5.7 Результаты внедрения глубокой нейронной сети с механизмом внимания

Таблица 5.8 - Классификационный отчет по результатам нейронной сети с использованием механизма внимания

Class	Precision	Recall	F1-score
1	93	94	93.7
0	92.5	93	92.8
Weighted Avg	92.2	93	93

Общая полученная confusion matrix выглядит так, как показано на рисунке ниже. Также стоит отметить, что это тоже одна из ключевых показателей в задачах машинного обучения, которая графически показывает количество или процент правильно и неправильно предсказанных объектов для каждого класса.

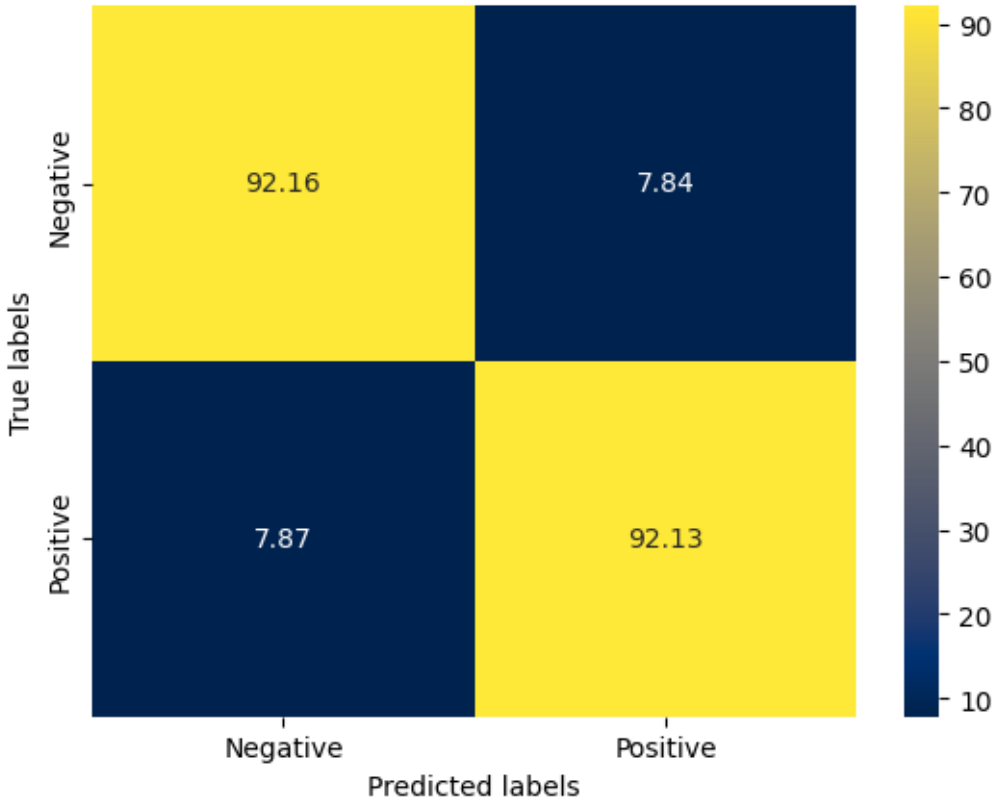


Рисунок 5.12 – Confusion matrix для глубокой нейронной сети с механизмом внимания

5.18 Сравнительный анализ результатов исследования

Ниже приведен сравнительный анализ по итогам исследования, который сравнительно показывает полученные результаты и достижения. Кроме уже показанных результатов здесь прилагается дополнительный рисунок, который показывает прирост лучшей получившейся модели по 4 ключевым показателям.

На рис. 5.13 сравнивается предлагаемая модель и все примененные модели машинного обучения и глубокого обучения по показателям accuracy, precision, recall и F1-score. На рисунке, предлагаемая глубокая нейронная сеть с механизмом внимания превосходит все применяемые методы машинного обучения и другие модели глубокого обучения по всем метрикам. Также этот рисунок показывает процент прироста показателей по каждому из внедренных алгоритмоы и полученных от них результатов.

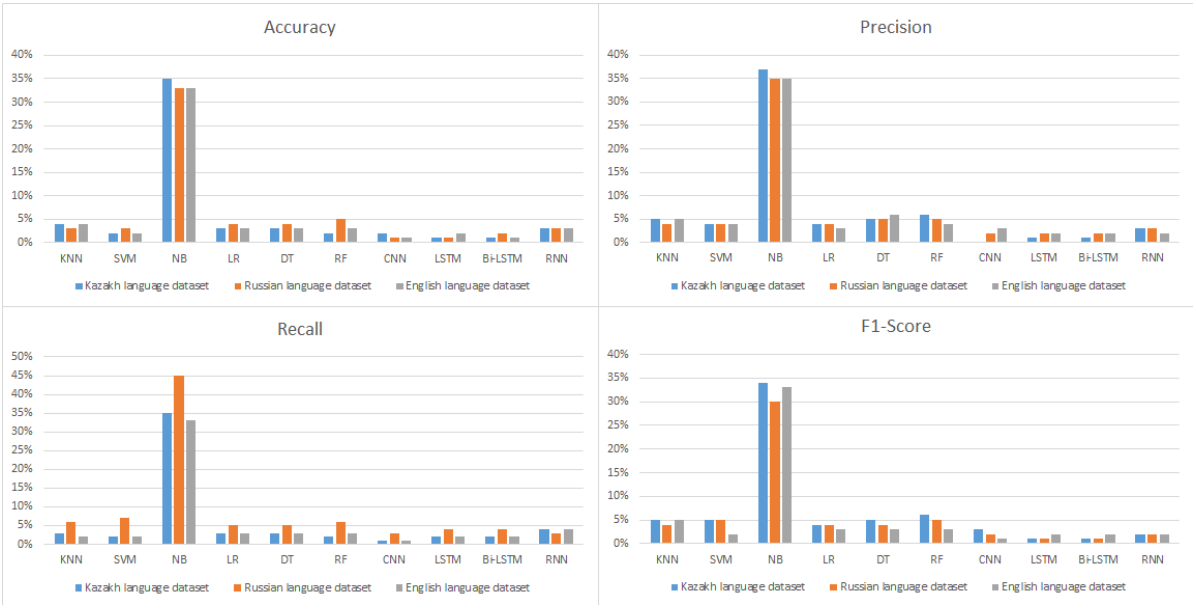


Рис 5.13 - Сравнение метрик для различных наборов данных

Область под кривой рабочих характеристик приемника со всеми извлеченными характеристиками представляет собой оценку эффективности в каждом классе. На рис. 8 сравниваются кривые AUC-ROC всех примененных методов и предлагаемого способа. Важно отметить, что все внедренные глубокие модели продемонстрировали более высокую ценность, чем методы машинного обучения. На рисунке показано, что предлагаемые модели.

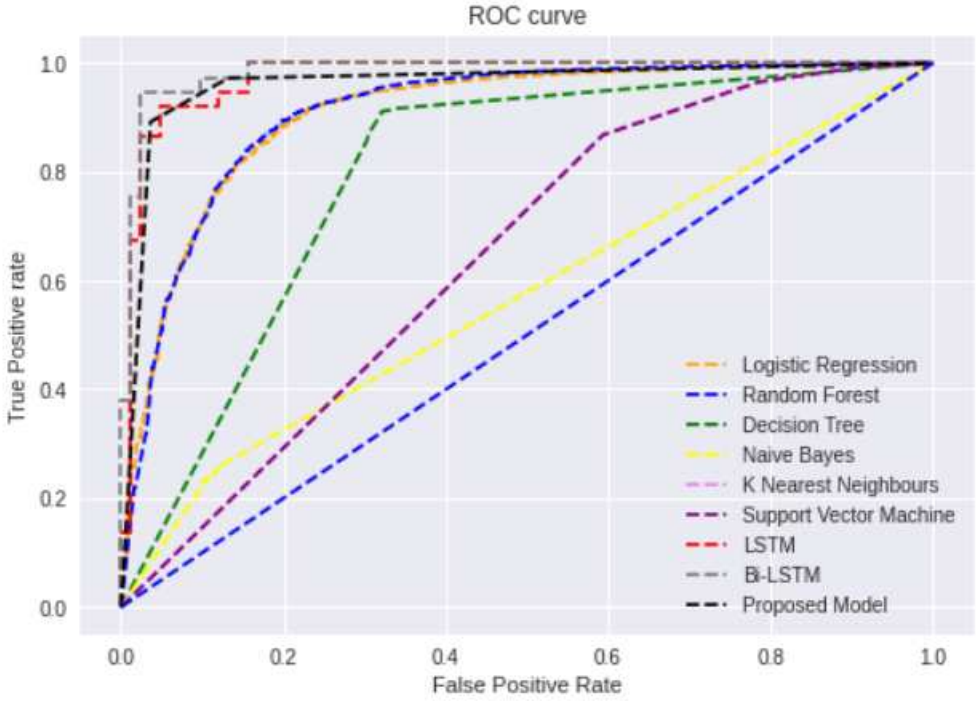


Рис 5.14 – средний показатель кривой под дугой в задаче классификации на наборах данных русского и английского языков.

Таблица 5. – Итоговые результаты выявления кибербуллинга

Dataset	Method	Method	Accuracy	Precision	Recall	F1-score	AUC-ROC
Набор данных русского языка	Глубокое обучение	LSTM	91.0%	92.5%	92.8%	92.5%	92%
		BiLSTM	91.3%	92.6%	93%	92.7%	92%
		CNN	89.2%	89%	89%	88.7%	90%
		Attention	92.1%	93%	92.5%	92.9%	95%
		RNN	89.7%	89.5%	90%	89.9%	94%
	Машинное обучение	LR	87.3%	85.2%	86.2%	85.1%	78%
		RF	85.6%	83.9%	83.1%	83.7%	92%
		DT	87.4%	83.2%	86.3%	85.1%	80%
		NB	60.2%	52.4%	58.5%	64.2%	65%
		KNN	85.1%	85.4%	82.2%	85.6%	77%
		SVM	86.2%	85.3%	83.7%	85.8%	78%
		LSTM	89.2%	89.5%	89.8%	89.6%	92%
Набор данных казахского языка	Глубокое обучение	LSTM	90%	91%	91.5%	91.2%	93%
		BiLSTM	91%	92%	92%	92%	93%
		CNN	88.7%	89%	89%	88.9%	90%
		Attention	91.5%	93%	92.5%	92.9%	94%
		RNN	89%	89%	91%	90%	92%
	Машинное обучение	LR	87.3%	85.6%	87.4%	60.2%	85.1%
		RF	85.2%	83.9%	83.2%	52.4%	85.4%
		DT	86.2%	83.1%	86.3%	58.5%	82.2%
		NB	85.1%	83.7%	85.1%	64.2%	85.6%
		KNN	75%	90%	76%	68%	77%
		SVM	87.3%	85.6%	87.4%	60.2%	85.1%

6. ВЫВОДЫ И ОБСУЖДЕНИЕ

6.1 Постановка исследовательского целя

Основной целью данного исследования была изучить эффективность различных техник и методов выявления кибербуллинга на онлайн-сервисах, уделяя особое внимание социальным сетям [75]. Кибербуллинг - это широко распространенная и растущая проблема, которая затрагивает людей всех возрастов, особенно подростков и молодых. Негативные последствия кибербуллинга могут быть серьезными, включая эмоциональный стресс, депрессию, и даже суицидальные мысли/действия. Поскольку общение онлайн продолжает оставаться неотъемлемой частью жизни людей, крайне важно своевременно и эффективно выявлять кибербуллинг и бороться с ним, чтобы смягчить его вредные последствия [76].

Исследовательскили целями, лежащими в основе этого исследования, заключается в следующем: каковы наиболее эффективные методы обнаружения кибербуллинга в текстовых данных и создать наиболее подходящую модель для повышения точности и скорости обнаружения. Чтобы ответить на этот вопрос, в исследовании были изучены различные подходы к обнаружению кибербуллинга, включая обработку естественного языка (NLP), алгоритмы машинного обучения и алгоритмы глубокого обучения. Кроме того, в исследовании была изучена роль функций пользовательского контента, а также поведенческих и контекстуальных факторов для прогнозирования и выявления авторов кибербуллингосодержащих текстов.

6.2 Ключевые достижения

Анализ исследований показл, что среди различных методов, используемых для обнаружения кибербуллинга, архитектуры глубокого обучения, особенно основанные на рекуррентных нейронных сетях и сверточных нейронных сетях и сетях LSTM, дают наилучшие результаты. Эти модели машинного обучения демонстрируют лучшие результаты с точки зрения ключевых показателей по сравнению с традиционными алгоритмами машинного обучения и другими способами обработки естественного языка.

Модели глубокого обучения лучше справляются с этой задачей, поскольку они могут эффективно улавливать сложные шаблоны и функции в текстовых данных [77]. В частности, сети с долговременной кратковременной памятью (LSTM) с использованием архитектуры внимания, особенно хорошо подходят для обработки и анализа последовательных данных(Seq). Подобные модели могут выявлять и использовать тонкие лингвистические сигналы, указывающие на кибребуллинг, путем выявления зависимостей и взаимосвязей между словами, фразами в тексте.

Эти результаты подчеркивают вожность использования моделей глубокого обучения для решения задач классификации и обработки

естественного обучения[78]. Полученные результаты и выводы открывают новые пути для будущих исследований и разработки еще более точных инструментов и стратегий обнаружения кибербуллинга.

ЗАКЛЮЧЕНИЕ

Данная диссертационная работа была посвящена разработке алгоритмов глубокого обучения для задач классификации случаев кибербуллинга в онлайн-пользовательском контенте. В процессе исследования были изучены различные архитектуры машинного и глубокого обучения, включая рекуррентные нейронные сети, сверточные нейронные сети и, а также инструменты обработки естественного языка, и исследована их эффективность в обнаружении кибербуллинга.

Результаты исследований показали, что модели глубокого обучения, демонстрируют высокую точность в обнаружении кибербуллинга по сравнению с традиционными алгоритмами машинного обучения. Эффективно улавливая сложные закономерности и особенности в текстовых потоках, эти модели хорошо подходят для решения многогранных задач текстовой классификации. В исследовании также подчеркиваются потенциальные преимущества объединения различных архитектур глубокого обучения.

Несмотря на высокие показатели, сохраняются такие проблемы, как потребность в больших объемах классифицированных данных, так как к примеру для работы с текстами казахского языка полученных данных нейдостаточно; вычислительных ресурсах, которые только возрастают с увеличением данных и усложнением структур моделей.

В ходе исследования был проведен качественный анализ существующих работ и работ посвященным обработке естественного языка, которые на примерах английского и арабского языка показывают точность больше 95%. Также были исследованы влияние составных элементов нейронных сетей на точность и другие показатели, была сделана большая работа, посвященная настройке гипер параметров.

Таким образом, эксперименты, проведенные над различными наборами данных, анализ собранных данных и тестирование алгоритмов показали, что использование нейронных сетей может помочь бороться с растущей проблемой кибербуллинга и разработанная модель может быть использована в дальнейших исследованиях и инструментах автоматического нахождения кибербуллинга в пользовательском конетенте.

В диссертации были получены следующие результаты:

1) Собран набор данных, предварительно обработанный и вручную классифицированный, для казахского языка, для дальнейших задач машинного и глубокого обучения.
2) Разработана и обучена глубокая нейронная сеть с механизмом внимания для задачи двоичной и трех-классовой классификации.
3) Был проведен сравнительный анализ между машинным обучением и алгоритмами глубокого обучения в задаче выявления кибербуллинга.

СПИСОК ЛИТЕРАТУРЫ

[1] T. Alsubait and D. Alfageh, “Comparison of machine learning techniques for cyberbullying detection on youtube arabic comments,” International Journal of Computer Science and Network Security, vol. 21, no. 1, pp. 1–5, 2021.

[2] A. Dewani, M. Memon and S. Bhatti, “Cyberbullying detection: Advanced preprocessing techniques & deep learning architecture for roman urdu data,” Journal of Big Data, vol. 8, no. 1, pp. 1–20, 2021.

[3] D. Hall, Y. Silva, Y. Wheeler, L. Cheng and K. Baumel, “Harnessing the power of interdisciplinary research with psychology-informed cyberbullying detection models,” International Journal of Bullying Prevention, vol. 4, no. 1, pp. 47–54, 2021.

[4] K. Arce-Ruelas, “Automatic cyberbullying detection: A Mexican case in high school and Higher Education Students,” IEEE Latin America Transactions, vol. 20, no. 5, pp. 770–779, 2022. [5] T. Ahmed, M. Rahman, S. Nur, A. Islam and D. Das, “Natural language processing and machine learning based cyberbullying detection for Bangla and romanized bangla texts,” TELKOMNIKA (Telecommunication Computing Electronics and Control), vol. 20, no. 1, pp. 89–97, 2021.

[6] B. Omarov, A. Tursynova, O. Postolache, K. Gamry, A. Batyrbekov et al., “Modified UNet model for brain stroke lesion segmentation on computed tomography images,”CMC-Computers, Materials & Continua, vol. 71, no. 3, pp. 4701–4717, 2022.

[7] A. Al-Marghilani, “Artificial intelligence-enabled cyberbullying-free online social networks in smart cities,” International Journal of Computational Intelligence Systems, vol. 15, no. 1, pp. 1–13, 2022.

[8] C. Theng, N. Othman, R. Abdullah, S. Anawar, Z. Ayop et al., “Cyberbullying detection in twitter using sentiment analysis,” International Journal of Computer Science & Network Security, vol. 21, no. 11, pp. 1–10, 2021.

[9] S. Sadiq, A. Mehmood, S. Ullah, M. Ahmad, G. Choi et al., “Aggression detection through deep neural model on twitter,” Future Generation Computer Systems, vol. 114, no. 1, pp. 120–129, 2021.

[10] E. Sarac Essiz and M. Oturakci, “Artificial bee colony-based feature selection algorithm for cyberbullying,” The Computer Journal, vol. 64, no. 3, pp. 305–313, 2021.

[11] C. E. Gomez, M. O. Sztainberg and R. E. Trana, “Curating cyberbullying datasets: A human-AI collaborative approach,” International Journal of Bullying Prevention, vol. 4, no. 1, pp. 35–46, 2022.

[12] S. Salawu, J. Lumsden and Y. He, “A mobile-based system for preventing online abuse and cyberbullying,” International Journal of Bullying Prevention, vol. 4, no. 1, pp. 66–88, 2022.

[13] M. Mladenovic, V. O ´ smjanski and S. V. Stankovi ˇ c, “Cyber-aggression, cyberbullying, and cyber-grooming: A survey and research challenges,” ACM Computing Surveys (CSUR), vol. 54, no. 1, pp. 1–42, 2021. 2130 CMC, 2023, vol.74, no.1

[14] S. R. Sangwan and M. P. S. Bhatia, "Denigrate comment detection in low-resource Hindi language using attention-based residual networks," Transactions on Asian and Low-Resource Language Information Processing, vol. 21, no. 1, pp. 1–14, 2021.
[15] T. T. Aurpa, R. Sadik and M. S. Ahmed, "Abusive Bangla comments detection on Facebook using transformer-based deep learning models," Social Network Analysis and Mining, vol. 12, no. 1, pp. 1–14, 2022.
[16] R. Yan, Y. Li, D. Li, Y. Wang, Y. Zhu et al., "A stochastic algorithm based on reverse sampling technique to fight against the cyberbullying," ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 15, no. 4, pp. 1–22, 2021.
[17] C. J. Yin, Z. Ayop, S. Anawar, N. F. Othman and N. M. Zainudin, "Slangs and short forms of malay twitter sentiment analysis using supervised machine learning," International Journal of Computer Science & Network Security, vol. 21, no. 11, pp. 294–300, 2021.
[18] G. Jacobs, C. Van Hee and V. Hoste, "Automatic classification of participant roles in cyberbullying: Can we detect victims, bullies, and bystanders in social media text?," Natural Language Engineering, vol. 28, no. 2, pp. 141–166, 2022.
[19] A. Jevremovic, M. Veinovic, M. Cabarkapa, M. Krstic, I. Chorbev et al., "Keeping children safe online with limited resources: Analyzing what is seen and heard," IEEE Access, vol. 9, no. 1, pp. 132723–132732, 2021.
[20] K. Kumari, J. P. Singh, Y. K. Dwivedi and N. P. Rana, "Multi-modal aggression identification using convolutional neural network and binary particle swarm optimization," Future Generation Computer Systems, vol. 118, no. 1, pp. 187–197, 2021.
[21] A. M. Abbas, "Social network analysis using deep learning: Applications and schemes," Social Network Analysis and Mining, vol. 11, no. 1, pp. 1–21, 2021.
[22] S. Gupta, N. Mohan, P. Nayak, K. C. Nagaraju and M. Karanam, "Deep vision-based surveillance system to prevent train-elephant collisions," Soft Computing, vol. 26, no. 8, pp. 4005–4018, 2022.
[23] S. Mohammed, W. C. Fang, A. E. Hassanien and T. H. Kim, "Advanced data mining tools and methods for social computing," The Computer Journal, vol. 64, no. 3, pp. 281–285, 2021.
[24] B. Thuraisingham, "Trustworthy machine learning," IEEE Intelligent Systems, vol. 37, no. 1, pp. 21–24, 2022.
[25] V. Rupapara, F. Rustam, H. Shahzad, A. Mehmood, I. Ashraf et al., "Impact of SMOTE on imbalanced text features for toxic comments classification using RVVC model," IEEE Access, vol. 9, no. 1, pp. 78621– 78634, 2021.
[26] O. Sharif and M. M. Hoque, "Tackling cyber-aggression: Identification and fine-grained categorization of aggressive texts on social media using weighted ensemble of transformers," Neurocomputing, vol. 490, no. 1, pp. 462–481, 2022.

[27] K. Kumari, J. P. Singh, Y. K. Dwivedi and N. P. Rana, "Bilingual cyber-aggression detection on social media using LSTM autoencoder," Soft Computing, vol. 25, no. 14, pp. 8999–9012, 2021.
[28] A. Mohamed, E. Amer, N. Eldin, M. Hossam, N. Elmasry et al., "The impact of data processing and ensemble on breast cancer detection using deep learning," Journal of Computing and Communication, vol. 1, no. 1, pp. 27–37, 2022.
[29] A. Sheth, V. L. Shalin and U. Kursuncu, "Defining and detecting toxicity on social media: Context and knowledge are key," Neurocomputing, vol. 490, no. 1, pp. 312–318, 2022.
[30] U. Kursuncu, H. Purohit, N. Agarwal and A. Sheth, "When the bad is good and the good is bad: Understanding cyber social health through online behavioral change," IEEE Internet Computing, vol. 25, no. 1, pp. 6–11, 2021.
[31] A. M. Veiga Simão, P. Costa Ferreira, N. Pereira, S. Oliveira, P. Paulino et al., "Prosociality in cyberspace: Developing emotion and behavioral regulation to decrease aggressive communication," Cognitive Computation, vol. 13, no. 3, pp. 736–750, 2021.
[32] G. Isaza, F. Muñoz, L. Castillo and F. Buitrago, "Classifying cybergrooming for child online protection using hybrid machine learning model," Neurocomputing, vol. 484, no. 1, pp. 250–259, 2022. CMC, 2023, vol.74, no.1 2131
[33] L. Cuoghi and L. Konopelko, "Cyberbullying classification," (accessed on 25 June 2022), 2022. [Online]. Available: https://www.kaggle.com/datasets/andrewmvd/cyberbullying-classification.
[34] D. Bruwaene, Q. Huang and D. Inkpen, "A multi-platform dataset for detecting cyberbullying in social media," (accessed on 25 June 2022), 2022. [Online]. Available: https://dl.acm.org/doi/abs/10.1007/ s10579-020-09488-3.
[35] A. Samoshyn, "Hate speech and offensive language dataset," (accessed on 25 June 2022), 2020. [Online]. Available: https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset.
[36] G. Perasso, N. Carone and L. Barone, "Written and visual cyberbullying victimization in adolescence: Shared and unique associated factors," European Journal of Developmental Psychology, vol. 18, no. 5, pp. 658–677, 2021.
[37] M. Amjad, N. Ashraf, A. Zhila, G. Sidorov, A. Zubiaga et al., "Threatening language detection and target identification in urdu tweets," IEEE Access, vol. 9, no. 1, pp. 128302–128313, 2021.
[38] Ö. Çoban, S. A. Özel and A. Inan, "Deep learning-based sentiment analysis of facebook data: The case of turkish users," The Computer Journal, vol. 64, no. 3, pp. 473–499, 2021.
[39] B. Omarov, N. Saparkhojayev, S. Shekerbekova, O. Akhmetova, M. Sakypbekova et al., "Artificial intelligence in medicine: Real time electronic stethoscope for heart diseases detection," CMC-Computers, Materials & Continua, vol. 70, no. 2, pp. 2815–2833, 2022.

[40] P. Parikh, H. Abburi, N. Chhaya, M. Gupta and V. Varma, “Categorizing sexism and misogyny through neural approaches,” ACM Transactions on the Web (TWEB), vol. 15, no. 4, pp. 1–31, 2021.
[41] S. Kiritchenko, I. Nejadgholi and K. C. Fraser, “Confronting abusive language online: A survey from the ethical and human rights perspective,” Journal of Artificial Intelligence Research, vol. 71, no. 1, pp. 431–478, 2021.
[42] J. A. García-Díaz, M. Cánovas-García, R. Colomo-Palacios and R. Valencia-García, “Detecting misogyny in Spanish tweets. An approach based on linguistics features and word embeddings,” Future Generation Computer Systems, vol. 114, no. 1, pp. 506–518, 2021.
[43] A. Tontodimamma, E. Nissi, A. Sarra and L. Fontanella, “Thirty years of research into hate speech: topics of interest and their evolution,” Scientometrics, vol. 126, no. 1, pp. 157–179, 2021.
[44] X. Chen, H. Xie, G. Cheng and Z. Li, “A decade of sentic computing: topic modeling and bibliometric analysis,” Cognitive Computation, vol. 14, no. 1, pp. 24–47, 2022.
[45] A. S. Srinath, H. Johnson, G. G. Dagher and M. Long, “BullyNet: Unmasking cyberbullies on social networks,” IEEE Transactions on Computational Social Systems, vol. 8, no. 2, pp. 332–344, 2021.
[46] C. Kumar, T. S. Bharati and S. Prakash, “Online social network security: A comparative review using machine learning and deep learning,” Neural Processing Letters, vol. 53, no. 1, pp. 843–861, 2021.
[47] M. Zhu, A. H. Anwar, Z. Wan, J. H. Cho, C. A. Kamhoua et al., “A survey of defensive deception: Approaches using game theory and machine learning,” IEEE Communications Surveys & Tutorials, vol. 23, no. 4, pp. 2460–2493, 2021.
[48] H. Sun and R. Grishman, “Employing lexicalized dependency paths for active learning of relation extraction,” Intelligent Automation & Soft Computing, vol. 34, no. 3, pp. 1415–1423, 2022.
[49] B. Omarov, A. Batyrbekov, K. Dalbekova, G. Abdulkarimova, S. Berkimbaeva et al., “Electronic stethoscope for heartbeat abnormality detection,” in 5th Int. Conf. on Smart Computing and Communication (SmartCom 2020), Paris, France, pp. 248–258, 2020.
[50] F. Bozyigit, O. Do ˘ gan and D. Kilinç, “Categorization of customer complaints in food industry using machine learning approaches,” Journal of Intelligent Systems: Theory and Applications, vol. 5, no. 1, pp. 85–91, 2022.
[51] B. Omarov, S. Narynov, Z. Zhumanov, A. Gumar and M. Khassanova, “A skeleton-based approach for campus violence detection,” Computers, Materials & Continua, vol. 72, no. 1, pp. 315–331, 2022.
[52] S. Bekturov, Қазақ тілі: фонетика, грамматика, морфология, синтаксис , The Science of Microfabrication. 2006.[
[53] A. Fedotov, J. Tussupov, M. Sambetbayeva, I. Idrisova, and A. Yerimbetova, “Development and implementation of a morphological model of kazakh language”,

Eurasian Journal of Mathematical and Computer Applications, vol. 3, no. 3, pp. 69–79, 2015.
[54] A. M. Borodkin, E. Lisin, and W. Strielkowski, “Data algorithms for processing and analysis of unstructured text documents”, Applied mathematical sciences, vol. 8, pp. 1213–1222, 2014.
[55] M. Mowafy, A. Rezk, and H. El-Bakry, “An efficient classification model for unstructured text document”, American Journal of Computer Science and Information Technology, vol. 06, Jan. 2018. doi: 10.21767/23493917.100016.
[56] A. Ittoo, L. M. Nguyen, and A. [den Bosch], “Text analytics in industry: Challenges, desiderata and trends”, Computers in Industry, vol. 78, pp. 96–107, 2016, Natural Language Processing and Text Analytics in Industry, issn: 0166-3615. doi: https://doi.org/10.1016/j.compind.2015.12.001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166361515300646.
[57] A. K. S. Tilve, “Text classification using naïve bayes, vsm and pos tagger”, 2017.
[58] L. Breiman, “Machine learning, volume 45, number 1 - springerlink”, Machine Learning, vol. 45, pp. 5–32, Oct. 2001. doi: 10.1023/A:1010933404324.
[59] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features”, in Machine Learning: ECML-98, C. Nédellec and C. 77 Rouveirol, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142, isbn: 978-3-540-69781-7.
[60] G. I. Webb, “Naïve bayes”, in Encyclopedia of Machine Learning, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 713–714, isbn: 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_576. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_576.
[61] D. G. Kleinbaum, “Introduction to logistic regression”, in Logistic Regression: A Self-Learning Text. New York, NY: Springer New York, 1994, pp. 1–38, isbn: 978-1-4757-4108-7. doi: 10.1007/978-1-4757-4108-7_1. [Online]. Available: https://doi.org/10.1007/978-1-4757-4108-7_1.
[62] T. Hastie, R. Tibshirani, and J. Friedman, “The elements of statistical learning”, Aug, Springer, vol. 1, Jan. 2001. doi: 10.1007/978-0-387-21606-5_7.
[63] M. Korobov, “Morphological analyzer and generator for russian and ukrainian languages”, in Analysis of Images, Social Networks and Texts, M. Y. Khachay, N. Konstantinova, A. Panchenko, D. Ignatov, and V. G. Labunets, Eds., Cham: Springer International Publishing, 2015, pp. 320–332, isbn: 978-3-319-26123-2.
[64] A. Hassan and A. Mahmood, “Deep learning for sentence classification”, 2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT), pp. 1–5, 2017.
[65] K. Kowsari, D. E. Brown, M. Heidarysafa, K. Jafari Meimandi, M. S. Gerber, and L. E. Barnes, “Hdltex: Hierarchical deep learning for text classification”, 2017 16th IEEE International Conference on Machine Learning and Applications

(ICMLA), Dec. 2017. doi: 10.1109/icmla.2017.0-134. [Online]. Available: http://dx.doi.org/10.1109/ICMLA.2017.0-134.
[66] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, Text summarization techniques: A brief survey, 2017. arXiv: 1707.02268 [cs.CL].
[67] J. M. Conroy and D. P. O'leary, "Text summarization via hidden markov models", in Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR '01, New Orleans, Louisiana, USA: Association for Computing Machinery, 2001, pp. 406–407, isbn: 1581133316. doi:10.1145/38395.2384042. [Online]. Available: https://doi.org/10.1145/383952.384042.
[68] S. Miranda-Jiménez, A. Gelbukh, and G. Sidorov, "Conceptual graphs as framework for summarizing short texts", Int. J. Concept. Struct. Smart Appl., vol. 2, no. 2, pp. 55–75, Jul. 2014, issn: 2166-7292. doi: 10.4018/IJCSSA.2014070104. [Online]. Available: https://doi.org/10.4018/IJCSSA.2014070104.
[69] M. Binwahlan, N. Salim, and L. Suanmali, "Fuzzy swarm diversity hybrid model for text summarization", Information Processing Management, vol. 46, pp. 571–588, Sep. 2010. doi: 10.1016/j.ipm.2010.03.004.78
[70] M. Karabalayeva, Z. Yessenbayev, and Z. Kozhirbayev, "Regarding the impact of kazakh phonetic transcription on the performance of automatic speech recognition systems", Oct. 2017.
[71] Z. Kozhirbayev, B. Erol, A. Sharipbay, and M. Jamshidi, "Speaker recognition for robotic control via an iot device", Jun. 2018, pp. 1–5. doi:10.23919/WAC.2018.8430295.
[72] A. Mussina, S. Aubakirov, D. Ahmed-Zaki, and P. Trigo, "Automatic document summarization based on statistical information", Journal of Mathematics, Mechanics and Computer Science, vol. 96, no. 4, pp. 76–87, 2019, issn: 2617-4871. [Online]. Available: https://bm.kaznu.kz/index.php/kaznu/article/view/581.
[73] Z. Kozhirbayev, Z. Yessenbayev, and A. Makazhanov, "Document and word-level language identification for noisy user generated text", English, in IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018 - Proceedings, ser. IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018 - Proceedings, 12th IEEE International Conference on Application of Information and Communication Technologies, AICT 2018 ; Conference date: 17-10-2018 Through 19-10-2018, United States: Institute of Electrical and Electronics Engineers Inc., Oct. 2018. doi: 10.1109/ICAICT.2018.8747138.
[74] B. Myrzakhmetov, Z. Yessenbayev, and A. Makazhanov, "Initial normalization of user generated content: Case study in a multilingual setting", English, in IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018 - Proceedings, ser. IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018-

Proceedings, 12th IEEE International Conference on Application of Information and Communication Technologies, AICT 2018 ; Conference date: 17-10-2018 Through 19-10-2018, United States: Institute of Electrical and Electronics Engineers Inc., Oct. 2018. doi: 10.1109/ICAICT.2018.8747161.
[75] B. Myrzakhmetov and A. Makazhanov, “Initial experiments on russian to kazakh smt”, Research in Computing Science, vol. 117, pp. 153–160, Sep.2016. doi: 10.13053/rcs-117-1-13.
[76] Z. Kozhirbayev, Z. Yessenbayev, and M. Karabalayeva, “Kazakh and russian languages identification using long short-term memory recurrent neural networks”, 2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT), pp. 1–5, 2017.
[77] S. Mukherjee and P. Bhattacharyya, “Sentiment analysis : A literature survey”, CoRR, vol. abs/1304.4520, 2013. arXiv: 1304 . 4520. [Online]. Available: http://arxiv.org/abs/1304.4520.
[78] A. Kumar and A. Jaiswal, “Systematic literature review of sentiment analysis on twitter using soft computing techniques”, Concurrency and Computation: Practice and Experience, vol. 32, no. 1, e5107, 2020, e5107 CPE18-1167.R1. doi: 7910.1002/cpe.5107. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.5107. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5107.
[79] A. Qazi, R. G. Raj, G. Hardaker, and C. Standing, “A systematic literature review on opinion types and sentiment analysis techniques: Tasks and challenges”, Internet Res., vol. 27, no. 3, pp. 608–630, 2017.
[80] L. Martí , N. Sánchez-Pi, J. Molina, and A. C. Garcia, “Anomaly detection based on sensor data in petroleum industry applications”, Sensors, vol. 15, pp. 2774–2797, Feb. 2015. doi: 10.3390/s150202774.