Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра математического анализа и моделирования
Направление подготовки – 01.03.02 Прикладная математика и информатика
Направленность (профиль) образовательной программы «Прикладная математика и информатика»

ДОПУСТИТЬ К ЗАЩИТЕ
И.о. зав. кафедрой
_____ Н.Н. Максимова
«_____»_____ 2025 г.

БАКАЛАВРСКАЯ РАБОТА

на тему: Нейронная сеть для определения логических причинно-следственных связей на основе корпуса данных естественного языка

Исполнитель
студент группы 1101об _____ Д.В. Тузов
(подпись, дата)

Руководитель
доцент, канд. физ.-мат. наук _____ Н.Н. Максимова
(подпись, дата)

Нормоконтроль
Ассистент _____ В.О. Салмиянов
(подпись, дата)

Благовещенск 2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра математического анализа и моделирования

УТВЕРЖДАЮ
И.о. зав. кафедрой
_____ Н.Н. Максимова
«_____»_____2025 г.

З А Д А Н И Е

К бакалаврской работе студента Тузова Дмитрия Валерьевича

1. Тема бакалаврской работы: Нейронная сеть для определения логических причинно-следственных связей на основе корпуса данных естественного языка (утверждена приказом от 03.04.2025 № 879-уч).

2. Срок сдачи студентом законченной работы: 25.06.2025 г.

3. Исходные данные к бакалаврской работе: научные статьи, официальное руководство Python, учебные и учебно-методические работы, отчет по производственной практике (научно-исследовательской работе), отчет по преддипломной практике, среда разработки – Google Colab, набор данных «Choice of Plausible Alternatives for Russian language (PARus)».

4. Содержание бакалаврской работы (перечень подлежащих разработке вопросов): основные положения науки о данных, задачи и методы машинного обучения, основы нейронных сетей, разработка модели нейронной сети для работы с текстовыми данными, реализация модели посредством языка программирования Python в среде Google Colab, результаты работы нейронной сети.

5. Перечень материалов приложения: листинги вычислительных программ.

6. Консультанты по бакалаврской работе: нормоконтроль – Салмиянов В.О., ассистент.

7. Дата выдачи задания: 14.04.2025 г.

Руководитель бакалаврской работы: Максимова Надежда Николаевна, доцент, канд. физ.-мат. наук.

Задание принял к исполнению (14.04.2025): _____ Тузов Д.В.

РЕФЕРАТ

Бакалаврская работа содержит 75 с., 8 рисунков, 4 таблицы, 1 приложение, 20 источников.

DATA SCIENCE, НЕЙРОННАЯ СЕТЬ, РЕГРЕССИЯ, БИНАРНАЯКЛАССИФИКАЦИЯ, КЛАСТЕРИЗАЦИЯ, МАШИННОЕ ОБУЧЕНИЕ, ИСКУССТВЕННЫЙ НЕЙРОН, ОЧИСТКА ТЕКСТА, ТОКЕНИЗАЦИЯ, ВЕКТОРИЗАЦИЯ, ЭМБЕДДИНГ, ЭПОХИ, АРХИТЕКТУРА НЕЙРОННОЙ СЕТИ, ГИПЕРПАРАМЕТРЫ.

В данной работе представлено построение модели выявления причинно-следственных связей между парами фраз на основе нейронных сетей и проведены её обучению и оценка качества работы.

Цель работы – разработка и реализация нейронной сети, способной классифицировать пары фраз на наличие или отсутствие причинно-следственной связи, а также анализ полученных результатов с использованием метрик машинного обучения и визуализации ошибок.

Основу методологии исследования составляют современные подходы обработки естественного языка посредством бинарной классификации с помощью модели BERT.

Результатом работы является реализация нейронной сети в среде Google Colab посредством языка программирования Python, с использованием специальных библиотек TensorFlow, scikit-learn и sentence-transformers, численная оценка эффективности модели на валидационной выборке, анализ результатов и ошибок.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире задачи обработки естественного языка становятся всё более актуальными, особенно в таких областях, как автоматическое понимание текста, машинное рассуждение и анализ логических связей. Анализ причинно-следственных связей в тексте представляет собой одну из наиболее сложных задач в области обработки естественного языка. Успешное решение этой задачи позволяет не только улучшить качество моделей машинного понимания текста, но и открывает возможности для автоматизации логического мышления в системах искусственного интеллекта. Особенно это важно в таких сферах, как юридический и медицинский анализ текста, обработка новостей, построение рекомендательных систем и создание контента с заданной логикой [2, 3].

С развитием моделей глубокого обучения появилась возможность более точно моделировать логические зависимости между событиями, описанными в тексте. Это позволяет перейти от простых подходов, основанных на ключевых словах и частных словарях, к моделям, учитывающим контекст и семантику предложений [2].

Объектом данного исследования являются пары предложений на русском языке, содержащие предпосылку и два возможных варианта следствия, взятые из открытого источника датасетов Metatext [19]. Предметом исследования является нейронная сеть, основанная на архитектуре, использующей предобученные эмбеддинги предложений для сравнения степени причинности каждого из следствий относительно предпосылки.

Целью работы является разработка и реализация нейронной сети для определения наличия причинно-следственной связи между предложениями на основе предобученных моделей обработки естественного языка.

Задачи по достижению поставленной цели следующие:

– Обзор существующих подходов к моделированию причинности в тексте и современных методов обработки естественного языка;

– постановка концептуальной формулировки задачи;

– предобработка и подготовка текстовых данных;
– построение и обучение нейронной сети;
– настройка гиперпараметров;
– анализ результатов и оценка качества.

Научная новизна данной работы заключается в том, что, предложенный подход к определению причинности через разностное представление эмбеддингов и сравнение двух возможных следствий в рамках одной модели является оригинальным и может быть использован в дальнейших работах по анализу логических связей в тексте.

Бакалаврская работа состоит из введения, трех основных разделов, заключения, библиографического списка и приложения.

В первой главе рассматривается основные понятия из области науки о данных, рассматривается классификация и обзор задач моделирования на данных, также рассматриваются области применения, ключевые технологии для работы с данными и решения соответствующих задач.

Во второй главе рассмотрены методы нейронных сетей в задачах бинарной классификации, основы алгоритмов машинного обучения в задачах бинарной классификации, также рассмотрены базовые концепции нейронных сетей и их архитектур, представлены метрики оценки качества работы нейронной сети в задачах бинарной классификации и библиотеки языка программирования Python для работы с нейронными сетями.

В третьей главе представлена концептуальная и математические постановки задачи, описан процесс предобработки данных, построения и обучения нейронной сети, также приведены результаты работы нейронной сети и их оценка.

В приложении приведён листинг нейронной сети.

1 НАУКА О ДАННЫХ: ОСНОВЫНЕ ПОНЯТИЯ

Говоря о науке о данных, следует изначально понимать, что такое данные в широком смысле использования термина.

Данные – это любая информация, которая может быть собрана, сохранена, обработана и использована для каких-либо целей. Данные охватывают не только цифровую информацию, но и любые сведения, отражающие реальные факты, события, процессы или явления.

У данных нет определённой формы представления, они могут существовать в виде чисел, текстов, изображений, звука, видео и т.д. Данные сами по себе могут не содержать осмысленной интерпретации, они становятся значимыми после анализа, обработки и применения.

Подобно формам данных, существуют различные виды происхождения данных и источники их добычи: наблюдения (например, температура воздуха), измерения (вес, рост), опросы, автоматические системы (сенсоры, датчики) и др. Исходя из этого, данные могут быть: количественными (числовые значения), качественными (описательные характеристики), структурированными (в таблицах, базах данных), неструктурированными (тексты, аудио, видео).

1.1 Определение науки о данных

Наука о данных (Data Science) представляет собой междисциплинарную область знаний, находящуюся на стыке статистики, информатики, машинного обучения и предметных областей, в которых проводится анализ данных. Её основное предназначение заключается в систематическом извлечении ценной информации, скрытых закономерностей и практически значимой информации из больших объёмов структурированных и неструктурированных данных [7].

Согласно классическому определению, предложенному различными исследователями и практиками в области, наука о данных может быть представлена как процесс преобразования сырых данных в понятные и полезные выводы, которые могут быть использованы для принятия решений, прогнозирования, автоматизации или улучшения бизнес-процессов. При этом ключевая особенность

Data Science заключается в её прикладной направленности: любые результаты исследований должны быть интерпретируемыми, воспроизводимыми и применимыми к реальным задачам [7].

Хотя термин «Data Science» стал широко применяться лишь в начале XXI века, истоки этой дисциплины находятся в более ранних периодах развития статистики, математической логики и информационных технологий [18]. Формирование современного понимания Data Science связано с развитием таких направлений, как статистика, особенно многомерный анализ, регрессионное моделирование и байесовские методы; информатика, включая алгоритмизацию, обработку больших массивов данных и баз данных; машинное обучение, которое позволило автоматизировать процесс анализа и предсказания на основе данных; искусственный интеллект, расширивший возможности автоматизации принятия решений и создания адаптивных систем.

Термин «Data Science» был впервые введён в научный оборот ещё в 1960-х годах, но его популярность резко возросла после публикаций специалистов вроде Вильяма Клегга, который в 2001 году описал Data Science как самостоятельную дисциплину, отличную от традиционной статистики. С момента активного внедрения интернета, появления социальных сетей, мобильных устройств, объёмы появляющихся в мире данных начали экспоненциально расти, что потребовало новых подходов к их обработке и анализу [18].

Основными целями науки о данных являются:

– Описание данных: понимание структуры, характера и качества имеющихся данных.

– Диагностика: выявление причин наблюдаемых явлений, анализ исходных данных.

– Прогноз: построение моделей, позволяющих предсказывать будущие события или поведение систем.

– Рекомендация: предоставление обоснованных рекомендаций на основе данных для оптимизации процессов и принятия решений.

В наше время наука о данных играет ключевую роль во многих сферах жизни общества. В условиях цифровой трансформации организаций и повсеместного распространения цифровых технологий, способность эффективно использовать данные становится большим преимуществом как для государственных структур, так и для частных компаний [7].

Примеры влияния Data Science:

– Здравоохранение: использование алгоритмов машинного обучения для диагностики заболеваний по МРТ и другим изображениям.

– Финансы: модели кредитного скоринга и детектирования мошенничества.

– Маркетинг: персонализация рекламы и предложений на основе поведения пользователей.

– Производство: предиктивное обслуживание оборудования с целью снижения простоев.

– Образование: адаптация учебных программ на основе анализа успеваемости студентов.

Наука о данных стала одной из движущих сил цифровой эпохи, и, хотя она привлекает к себе меньше количество внимания, чем другие науки, всё же её влияние на современный мир велико.

1.2 Этапы работы с данными

Процесс анализа данных представляет собой сложный и многошаговый этап, требующую системного подхода, строгой последовательности действий и глубокого понимания как технических аспектов, так и специфики предметной области. В рамках Data Science работа с данными включает несколько ключевых этапов, каждый из которых имеет свою функциональную нагрузку и значение в общей цепочке получения полезной информации. Эти этапы не являются строго линейными – в реальных проектах и задачах часто возникает необходимость возврата на предыдущие шаги, уточнения целей или повторной обработки информации.

Первым и одним из самых важных этапов в любом проекте по анализу данных является постановка задачи, которая заключается в чётком определении целей исследования, формулировании гипотез, а также установлении критериев успешности проекта. Этот этап носит, как правило, междисциплинарный характер, поскольку требует взаимодействия различных научных областей [7, 18].

На данном этапе необходимо ответить на такие вопросы: какова цель анализа (прогнозирование, классификация, кластеризация, детектирование аномалий и т.д.)? Какие данные доступны, и какие ещё требуется собрать? Какие метрики будут использоваться для оценки эффективности решения?

Правильная постановка задачи позволяет избежать потери времени на ненужные вычисления и направляет дальнейшие усилия в нужное русло.

После постановки задачи, следует сбор данных, представляющий собой процесс получения информации из различных источников. Данные могут быть как внутренними (например, корпоративные базы данных, CRM-системы), так и внешними (публичные датасеты, API сторонних сервисов, опросы и т.д.) [7, 18].

Качество и полнота исходных данных напрямую влияют на результат решения поставленной задачи. Поэтому важно не только получить достаточный объём данных, но и обеспечить их релевантность, достоверность и соответствие решаемой задаче. Также на этом этапе проводится первичная оценка структуры данных: количество признаков, типы данных (категориальные, числовые, текстовые), возможные ошибки и т.д.

Следующим особо важным этапом является очистка данных, так как в реальных условиях данные редко бывают идеальными или хотя бы пригодными для работы с ними. Они могут содержать пропуски, дубликаты, выбросы, некорректные значения и т.д. Все эти проблемы способны существенно повлиять на результаты анализа и качество моделей [7, 18].

Основные действия на этапе очистки данных включают удаление или заполнение пропущенных значений, исправление ошибок ввода, устранение дубликатов записей, обнаружение и обработка выбросов, преобразование форматов данных.

Очистка данных может занять значительную часть времени от решения задачи, но она является обязательной процедурой, без которой дальнейшая работа будет некорректной.

После всех подготовительных этапов начинается процесс моделирования, являющийся центральным в большинстве задач, связанных с Data Science. Целью этого этапа является построение математической или статистической модели, способной решать поставленную задачу: предсказывать, классифицировать, группировать или объяснять данные [7, 18].

Моделирование включает в себя:

– Выбор подходящих алгоритмов (линейная регрессия, деревья решений, случайный лес, SVM, нейронные сети и др.).

– Обучение моделей на обучающей выборке.

– Настройку гиперпараметров.

– Кросс-валидацию для оценки устойчивости модели.

Выбор модели зависит от множества факторов: типа задачи, характера данных, требований к интерпретируемости, скорости работы и точности.

После построения модели проводится её оценка качества, которая позволяет понять, насколько хорошо модель справляется с задачей и сделать выводы о дальнейшем улучшении модели. Для этого используются различные метрики, зависящие от типа задачи: для классификации: Accuracy, Precision, Recall, F1-score, ROC-AUC и др.; для регрессии: MAE, MSE, RMSE, R^2 и др.; для кластеризации: silhouette score, Davies–Bouldin index и др.

Также важным аспектом является интерпретация модели, особенно в критических областях, таких как здравоохранение или финансы, где недостаточно просто получить хороший результат – необходимо понимать, почему он был получен и где он может быть применим.

Полученные результаты должны быть правильно интерпретированы и переведены на язык, понятный лицам, для которых результаты и были получены.

Этот этап направлен на извлечение практически значимых выводов, которые могут быть использованы для принятия решений, улучшения процессов, формирования дальнейших гипотез и др.

Работа с данными в рамках науки о данных представляет собой сложный и длительный процесс, состоящий из некоторых ключевых этапов. Каждый из них играет важную роль в достижении конечной цели – извлечения ценной информации из данных и использования этой информации для решения практических задач.

1.3 Классификация задач моделирования на данных

Процесс построения моделей в рамках Data Science предполагает решение разнообразных задач, каждая из которых имеет свою специфику, цели, методы реализации и способы оценки. В зависимости от характера решаемой проблемы, типа входных и выходных данных, а также используемых подходов, все задачи моделирования можно разделить на несколько основных категорий. Правильная классификация задач позволяет выбрать наиболее подходящие алгоритмы, метрики оценки и стратегию обработки данных.

1.3.1 Задачи классификации

Задача классификации заключается в том, чтобы отнести объект к одному из заранее определённых классов на основе его признаков. Это одна из самых распространённых задач машинного обучения, особенно в таких областях, как медицина, маркетинг, финансы, компьютерное зрение и обработка естественного языка [5, 7, 8].

Формальная постановка:

Пусть $X = \{x_1, x_2, ..., x_n\}$ – вектор признаков объекта, где n – размерность пространства признаков. Цель состоит в том, чтобы построить функцию $f : R^n \to Y$, которая ставит в соответствие каждому объекту X его класс $y \in Y$, где $Y = \{y_1, y_2, ..., y_k\}$ – множество возможных классов.

Если $k = 2$, задача называется бинарной классификацией, если $k > 2$ – многоклассовой классификацией.

Примеры применения: определение кредитоспособности клиента, распознавание лиц на изображениях, анализ тональности текста (нейтральный, позитивный, негативный), диагностика заболеваний на основе анализов.

Метрики оценки:

1. Accuracy (доля правильных ответов):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$

где TP – истинно положительные, TN – истинно отрицательные, FP – ложно положительные, FN – ложно отрицательные. Данные показатели берутся из матрицы ошибок, которая визуально показывает доли правильной и неправильной классификации. Зачастую, Accuracy не является основной метрикой, так как она показывает количество верно классифицированных данных, что не исключает случайной классификации, то есть простого угадывания.

2. Precision (точность предсказаний положительного класса):

$$Precision = \frac{TP}{TP + FP}.$$

3. Recall (полнота):

Показывает, какую долю релевантных объектов модель смогла правильно классифицировать из всех возможных.

$$Recall = \frac{TP}{TP + FN}.$$

4. F1-score (гармоническое среднее точности и полноты):

$$F1 = 2\frac{Precision \times Recall}{Precision + Recall}.$$

5. ROC-AUC (площадь под ROC-кривой):

Характеризует способность модели различать классы при разных порогах. Строится на основе TPR (True Positive Rate) и FPR (False Positive Rate).

$$TPR = \frac{TP}{TP + FN};$$

$$FPR = \frac{FP}{FP + TN}.$$

1.3.2 Задачи регрессии

Задача регрессии направлена на предсказание числового значения целевой переменной на основе входных признаков. В отличие от классификации, где результатом является дискретная величина (класс), в регрессии модель возвращает непрерывное значение [7].

Формальная постановка:

Дано множество объектов $X_i \in R^n$, с соответствующими значениями $y_i \in R$. Требуется найти такую функцию $f(X)$, чтобы минимизировать ошибку между предсказанным значением $\hat{y}_i = f(X_i)$ и реальным y_i.

Целевая функция может быть выражена следующим образом:

$$f_{\min} \sum_{i=1}^{N} L(y_i, f(X_i)),$$

где L – функция потерь.

Примеры применения: прогнозирование цены акции, оценка стоимости недвижимости, предсказание уровня продаж, определение возраста человека по фотографии.

Метрики оценки:

1. MAE (Mean Absolute Error):

Показывает среднее абсолютное отклонение предсказанных значений $\hat{y}$ от реальных y.

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|.$$

2. MSE (Mean Squared Error):

Показывает среднее квадратов отклонений предсказаний от реальных значений.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 .$$

3. RMSE (Root Mean Squared Error):

Показывает стандартное отклонение ошибок модели.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}.$$

4. R^2 (коэффициент детерминации):

Показывает долю дисперсии целевой переменной, которую объясняет модель.

$$R^2 = \sqrt{1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}},$$

где $\bar{y}$ – среднее значение целевой переменной.

1.3.3 Задачи кластеризации

Задача кластеризации относится к классу задач обучения без учителя и заключается в группировке объектов по схожести их характеристик. При этом заранее известны только данные, но нет готовых меток классов [7].

Формальная постановка:

Дано множество объектов $X = \{x_1, x_2, ..., x_N\}$, $x_i \in R^n$. Необходимо разбить эти объекты на k групп (кластеров), таких, что внутри каждого кластера объекты были максимально схожи, а между кластерами – различны.

Формально это можно выразить через минимизацию суммарного расстояния до центров:

$$\min_{C_1,...,C_k} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu\|^2,$$

где μ_i – центр кластера c_i.

Примеры применения: сегментация клиентов по поведению, группировка документов по тематике, обнаружение мошеннических транзакций, анализ геномных данных.

Метрики оценки:

Silhouette Score:

Показывает, насколько хорошо объекты внутри одного кластера похожи друг на друга или отличаются.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},$$

где $a(i)$ – среднее расстояние до других объектов в своём кластере, $b(i)$ – минимальное среднее расстояние до объектов другого кластера.

1.3.4 Задачи временных рядов

Анализ временных рядов – это особый класс задач, в которых данные зависят от времени. Эти задачи широко распространены в финансах, метеорологии, энергетике, здравоохранении и других областях [7].

Формальная постановка:

Дано множество наблюдений y_t, где $t = 1, 2, ..., N$. Задача заключается в построении модели $f(y_1, y_2, ..., y_{t-1}) \to y_t$.

Примеры применения: прогнозирование спроса, предсказание отказов оборудования, анализ активности пользователя во времени, финансовое прогнозирование (курсы валют, акции).

Метрики оценки те же, что и в регрессии: MAE, MSE, RMSE, R^2.

1.3.5 Другие специализированные задачи

Помимо перечисленных, в науке о данных существуют и более специализированные задачи, которые могут быть уникальны для конкретных предметных областей:

– Обнаружение аномалий – выявление редких событий, которые значительно отличаются от большинства наблюдений.

– Генеративные модели – создание новых данных, например, изображений, текста, аудио.

– Усиленное обучение – обучение через взаимодействие с окружающей средой и получение наград.

– Объяснение моделей – обеспечение интерпретируемости сложных моделей.

– Автоматизация машинного обучения – автоматизация процесса выбора модели и настройки гиперпараметров.

Все перечисленные типы задач находят своё применение в современных приложениях, от бизнес-аналитики до искусственного интеллекта. Особенно важным является понимание этих задач при переходе к более сложным методам, таким как нейронные сети [7].

1.4 Основные области применения

Data Science является универсальной дисциплиной, находящей применение практически во всех сферах человеческой деятельности, где присутствует необходимость анализа информации, прогнозирования, принятия решений и оптимизации процессов. С развитием технологий и ростом объёмов генерируемых данных роль Data Science становится всё более значимой. Ниже рассматриваются основные области, в которых методы науки о данных активно применяются, с пояснением специфики задач, используемых подходов и достигаемых результатов [18].

Финансы

Финансовая индустрия является одной из самых ранних и активных пользовательниц методов науки о данных. Здесь данные играют ключевую роль в управлении рисками, повышении эффективности операций и улучшении клиентского сервиса.

Основные задачи:

– Кредитный скоринг: модели машинного обучения используются для оценки кредитоспособности клиента на основе исторических данных.

– Детектирование мошенничества: нейронные сети и алгоритмы обнаружения аномалий помогают выявлять подозрительные транзакции в режиме реального времени.

– Алгоритмическая торговля: использование моделей временных рядов и глубокого обучения для автоматического исполнения торговых сделок.

– Прогнозирование финансовых показателей: предсказание курсов валют, цен на акции, доходов компаний и других экономических индикаторов.

– Риск-менеджмент: моделирование вероятности дефолта, оценка рыночных и операционных рисков.

Примеры технологий: Random Forest, XGBoost (для классификации клиентов), LSTM и Prophet (для прогнозирования временных рядов), Autoencoders (для детектирования аномалий).

Маркетинг и реклама

В условиях высокой конкуренции маркетологи всё чаще полагаются на данные для персонализации взаимодействия с клиентами, повышения эффективности кампаний и увеличения конверсии.

Основные задачи:

– Сегментация клиентов: разделение потребителей на группы по демографическим, поведенческим или другим признакам.

– Таргетированная реклама: использование рекомендательных систем и моделей CLV (Customer Lifetime Value) для точечного воздействия на целевую аудиторию.

– Анализ оттока клиентов: прогнозирование вероятности ухода клиента и разработка стратегий его удержания.

– Оптимизация ценообразования: использование данных о спросе и предложении для установления оптимальных цен.

– Контент-маркетинг: анализ тональности, автоматическое создание контента и выбор наиболее эффективных каналов продвижения.

Примеры технологий: K-means, DBSCAN (для кластеризации), Логистическая регрессия, CatBoost (для предсказания оттока), NLP-методы (для анализа отзывов и социальных сетей).

Здравоохранение

Медицинская сфера получает огромное количество данных, включая электронные медицинские карты, результаты лабораторных исследований, изображения (например, МРТ), генетическую информацию и другие виды биомедицинских данных. Использование науки о данных позволяет повысить точность диагностики, оптимизировать лечение и проводить профилактику заболеваний.

Основные задачи:

– Диагностика заболеваний: распознавание патологий по медицинским изображениям (например, рак легких по КТ).

– Прогнозирование развития болезней: модели машинного обучения позволяют выявлять пациентов с высоким риском сердечно-сосудистых заболеваний, диабета и других хронических состояний.

– Персонализированная медицина: адаптация терапии под индивидуальные особенности пациента.

– Обработка медицинского текста: автоматическое извлечение информации из заключений врачей, научных статей и других источников.

– Эпидемиологический анализ: моделирование распространения инфекций и планирование мер реагирования.

Примеры технологий: CNN (Convolutional Neural Networks) (для анализа изображений), BERT, BioBERT (для обработки медицинского текста), Survival Analysis (для прогнозирования исходов лечения).

Логистика и транспорт

Логистические компании и транспортные организации сталкиваются с необходимостью управления сложными системами доставки, маршрутизации и управления запасами. Наука о данных позволяет оптимизировать эти процессы и снизить операционные расходы.

Основные задачи:

– Прогнозирование спроса: определение количества товаров, которые будут востребованы в определённый период.

– Оптимизация маршрутов: минимизация времени и стоимости доставки за счёт анализа трафика, погоды и других факторов.

– Управление складскими запасами: предсказание моментов пополнения и списания товаров.

– Предиктивное обслуживание: предотвращение поломок транспорта и оборудования на основе анализа датчиков IoT.

– Анализ транспортных потоков: выявление зон повышенной нагрузки и планирование инфраструктуры.

Примеры технологий: ARIMA, SARIMA (для прогнозирования спроса), Алгоритмы графов (для оптимизации маршрутов), Time Series Forecasting с использованием LSTM.

Образование

В сфере образования наука о данных используется для повышения качества обучения, персонализации образовательных программ и повышения вовлечённости студентов.

Основные задачи:

– Прогнозирование успеваемости: выявление студентов, находящихся в группе риска неуспеваемости.

– Адаптивное обучение: создание индивидуальных траекторий обучения на основе поведения и результатов ученика.

– Анализ эффективности преподавания: оценка влияния различных методов обучения на успеваемость.

– Автоматическая проверка заданий: использование NLP и компьютерного зрения для автоматической оценки ответов.

Примеры технологий: Деревья решений, случайный лес (для классификации успеваемости), Рекуррентные нейронные сети (для анализа текстовых ответов), Тематическое моделирование (для анализа содержания работ).

Обработка естественного языка (NLP)

Обработка естественного языка (NLP) – это одно из наиболее динамично развивающихся направлений науки о данных, связанное с анализом и интерпретацией текстовой информации.

Основные задачи:

– Анализ тональности: определение эмоциональной окраски текста (положительная, отрицательная, нейтральная).

– Извлечение сущностей (NER): поиск и классификация имён собственных, таких как люди, места, организации.

– Машинный перевод: автоматическое преобразование текста с одного языка на другой.

– Генерация текста: создание текстовых сообщений, новостей, отчётов и т.д.

– Выявление причинно-следственных связей: одна из ключевых задач данной работы, заключающаяся в определении отношения между событиями в тексте.

Примеры технологий: Word2Vec, GloVe, FastText (для представления слов), Трансформеры (BERT, RoBERTa, GPT) (для понимания и генерации текста), RNN, LSTM (для последовательной обработки текста).

Энергетика

В энергетическом секторе наука о данных используется для оптимизации производства, распределения и потребления энергии, а также для повышения устойчивости энергетических систем.

Основные задачи:

– Прогнозирование спроса и предложения энергии.

– Оптимизация работы электростанций.

– Обнаружение утечек и потерь в сетях.

– Интеграция возобновляемых источников энергии.

– Анализ состояния инфраструктуры.

Примеры технологий: Модели временных рядов, Глубокое обучение для анализа датчиков, Алгоритмы оптимизации для управления нагрузкой.

Наука о данных находит применение в широком спектре отраслей, обеспечивая автоматизацию, повышение точности прогнозов, оптимизацию процессов и поддержку принятия решений. Универсальность методов Data Science позволяет использовать их как в коммерческих целях, так и в научных исследованиях, государственном управлении, здравоохранении и других важных сферах жизни общества.

1.5 Ключевые технологии и инструменты

С развитием Data Science появилось множество программных технологий, библиотек, фреймворков и платформ, которые позволяют эффективно собирать,

обрабатывать, анализировать и моделировать данные. Эти инструменты обеспечивают полный цикл работы с данными – от их извлечения до развертывания моделей в производственной среде. Правильный выбор технологий играет ключевую роль в успехе любого проекта, поскольку влияет на производительность, точность, масштабируемость и удобство реализации решений.

1.5.1 Языки программирования

Выбор языка программирования зависит от характера задачи, уровня сложности анализа и специфики предметной области. Наиболее популярными языками в науке о данных являются:

Язык программирования Python

Python является наиболее распространённым языком в сообществе Data Science благодаря своей простоте, читаемости кода и богатой экосистеме библиотек. Он поддерживает как классические методы машинного обучения, так и современные нейронные сети [6, 13, 14, 15, 17].

Основные преимущества: простота освоения, поддержка параллельных вычислений и GPU-ускорения, интеграция с Jupyter Notebook и другими IDE, широкое комьюнити и активная поддержка.

Популярные библиотеки: Pandas, NumPy – для работы с данными; Matplotlib, Seaborn, Plotly – для визуализации; Scikit-learn – для классических алгоритмов машинного обучения; TensorFlow, Keras, PyTorch – для глубокого обучения; NLTK, tspaCy, Transformers (от Hugging Face) – для обработки естественного языка.

Язык программирования R

R – это язык и среда программирования, специально разработанная для статистического анализа и графической визуализации данных. Он особенно популярен среди учёных, экономистов и статистиков.

Основные преимущества: мощные средства статистики и анализа, богатые возможности визуализации (ggplot2), поддержка широкого спектра статистических тестов и моделей.

SQL

SQL (Structured Query Language) используется для взаимодействия с реляционными базами данных. В рамках Data Science он применяется для извлечения, фильтрации и предварительной обработки данных, прежде чем они будут переданы в аналитические или модельные системы.

Основные преимущества: универсальность и стандартизация, высокая эффективность при работе с большими объёмами структурированных данных, интеграция с большинством СУБД (PostgreSQL, MySQL, Oracle и др.).

1.5.2 Библиотеки и фреймворки для анализа и моделирования

Для выполнения конкретных задач в области науки о данных используются специализированные библиотеки и фреймворки, которые значительно упрощают работу с данными, автоматизируют процессы и повышают точность результатов.

Pandas и NumPy

Эти библиотеки служат основой для работы с данными в Python. Pandas предоставляет удобные структуры данных, такие как DataFrame и Series, а NumPy – мощные многомерные массивы и математические функции [1, 6, 13, 14, 15, 17].

Scikit-learn

Scikit-learn – это одна из самых популярных библиотек машинного обучения в Python. Она содержит реализации большого числа алгоритмов классификации, регрессии, кластеризации, снижения размерности и оценки качества моделей [6, 13, 14, 15, 17].

TensorFlow и PyTorch

Эти фреймворки предназначены для разработки и обучения нейронных сетей. TensorFlow предлагает высокий уровень абстракции и поддержку Keras, тогда как PyTorch отличается гибкостью и удобством при работе с исследовательскими проектами и экспериментами [1, 6, 10, 13, 14, 15, 17].

NLTK, spaCy, Transformers

Для обработки естественного языка (NLP) используются специализированные библиотеки:

NLTK – содержит набор инструментов для токенизации, леммитизации и других NLP-задач.

spaCy – оптимизированная библиотека для промышленной обработки текста.

Transformers – позволяет использовать предобученные модели, такие как BERT, GPT, RoBERTa и другие.

1.5.3 Инструменты визуализации данных

Визуализация играет важную роль в интерпретации данных, выявлении закономерностей и презентации результатов. Существует несколько мощных библиотек и платформ для создания графиков, диаграмм и интерактивных панелей.

Matplotlib и Seaborn

Matplotlib – базовая библиотека для построения графиков, поддерживающая широкий спектр типов диаграмм. Seaborn строится на основе Matplotlib и предоставляет более эстетичные и информативные визуализации.

Plotly и Bokeh

Эти библиотеки позволяют создавать интерактивные визуализации, которые можно внедрять в веб-приложения и отчёты.

Tableau и Power BI

Tableau и Power BI – это инструменты бизнес-аналитики, не требующие глубоких знаний программирования. Они позволяют быстро создавать визуализацию, проводить анализ и делиться результатами с коллегами.

1.5.4. Средства хранения и управления данными

Работа с большими объемами данных требует использования систем хранения и управления, которые обеспечивают надежность, масштабируемость и быстрый доступ к информации.

Реляционные СУБД

MySQL, PostgreSQL, Oracle – подходят для хранения структурированных данных и обеспечения ACID-совместимости.

NoSQL базы данных

MongoDB, Cassandra, Redis – используются для работы с неструктурированными или полуструктурированными данными, где важна горизонтальная масштабируемость.

Big Data платформы

Apache Hadoop – распределенная система хранения и обработки больших данных.

Apache Spark – обеспечивает обработку данных в режиме реального времени и поддерживает SQL, машинное обучение и потоковые вычисления.

1.5.5 Интерактивные среды и инструменты разработки

Для работы с данными и написания кода используются различные интерактивные среды и IDE.

MATLAB

ППП MATLAB – это пакет прикладных программ, представляющий собой платформу для технических вычислений, включая инструменты для обработки естественного языка (NLP) и глубокого обучения. MATLAB предлагает интегрированную среду с акцентом на удобство разработки и визуализацию. Ключевые возможности: Text Analytics Toolbox – токенизация и нормализация текста, извлечение n-грамм и ключевых фраз, поддержка предобученных word2vec моделей; Deep Learning Toolbox – готовые слои для LSTM, CNN, Transformers, автоматическая дифференциация, поддержка GPU/CPU вычислений.

Jupyter Notebook

Jupyter Notebook – это популярная интерактивная среда, которая позволяет сочетать код, текст, формулы и визуализации в одном документе. Она широко используется для прототипирования, обучения и презентаций.

Google Colab и Kaggle Kernel

Облачные версии Jupyter Notebook, предоставляющие бесплатный доступ к GPU/TPU и готовым окружениям для работы с данными.

IDE

VS Code – легковесная и мощная среда разработки с плагинами для Python, Git, Jupyter и других инструментов.

PyCharm – профессиональная IDE от JetBrains, ориентированная на разработку на Python.

Выбор конкретных технологий зависит от характера задачи, доступных ресурсов, требований к производительности и масштабируемости. Однако общая тенденция заключается в том, что Python становится стандартом в этой области, а интеграция между различными инструментами позволяет строить сложные аналитические системы, способные решать самые разнообразные задачи.

2 НЕЙРОННЫЕ СЕТИ В ЗВДВЧАХ БИНАРНОЙ ККЛАССИФИКАЦИИ

2.1 Задача бинарной классификации

Бинарная классификация представляет собой одну из наиболее распространённых задач машинного обучения, в которой модель обучается предсказывать принадлежность объекта к одному из двух возможных классов. Эта задача имеет широкое применение в различных областях – от медицины и финансов до маркетинга и обработки естественного языка (NLP) [5, 8, 11, 12].

2.1.1 Постановка задачи бинарной классификации

Формальная постановка задачи:

Пусть дано множество объектов $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$,

где $x_i \in R^n$ – вектор признаков i -го объекта, $y_i \in (0,1)$ – целевая переменная, принимающая одно из двух значений: 0 или 1.

Задача состоит в том, чтобы найти такую функцию $f : R^n \to \{0,1\}$, которая для любого нового объекта x_k могла бы предсказать его класс $y_{pred} = f(x_k)$, максимально точно соответствующий истинному значению y.

Модель f может быть как детерминированной, так и вероятностной. Например, в случае использования логистической регрессии или нейронной сети, выходом модели является вероятность принадлежности к положительному классу:

$$P(y = 1 | x) = \sigma(\omega^T x + b),$$

где ω – вектор весов, b – смещение (bias), $\sigma(z) = \frac{1}{1 + e^{-z}}$ – сигмоидная функция активации.

Класс предсказывается по правилу:

$$y_{pred} = \begin{cases} 1, & P(y = 1 | x \geq 0 \\ 0, & \end{cases},$$

где θ – порог принятия решения (обычно устанавливается равным 0.5).

Целевая функция:

Для обучения моделей в задачах бинарной классификации используется функция потерь, минимизация которой позволяет улучшить качество предсказаний. Наиболее популярной является логистическая (биномиальная) функция потерь:

$$L(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})],$$

где $\hat{y} = P(y = 1 | x)$ – предсказанная вероятность положительного класса.

Общая функция потерь для всей выборки выглядит следующим образом:

$$L(y, \hat{y}) = -\frac{1}{N}[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)].$$

Эта функция также называется binary cross-entropy loss и широко применяется при обучении логистической регрессии, нейронных сетей и других моделей [5, 8, 11, 12].

2.1.2 Примеры прикладных задач бинарной классификации

Бинарная классификация находит применение во множестве реальных задач, некоторые из которых приведены ниже:

Медицинская диагностика

Определение наличия или отсутствия заболевания у пациента на основе результатов анализов, анамнеза и прочих данных. Например, предсказание наличия сердечно-сосудистых заболеваний на основании показателей давления, уровня холестерина, возраста и др. [3, 8].

Анализ тональности текста

В NLP часто требуется определить, является ли отзыв положительным или отрицательным. Это задача бинарной классификации, где классы: {позитивный, негативный} [2, 9].

Обнаружение спама

Классификация электронных писем на спам и не спам. Признаками могут быть наличие ключевых слов, частота встречаемости определённых выражений, длина сообщения и т.д. [9]

Кредитный скоринг

Прогнозирование способности клиента погасить кредит. Классы: {кредитоспособный, некредитоспособный}. Признаки: доход, кредитная история, количество детей и т.д.

Выявление мошенничества

Обнаружение подозрительных финансовых операций. Признаками могут быть сумма транзакции, время совершения, географическое местоположение и другие метаданные.

Определение наличия причинно-следственной связи в тексте

В рамках данной работы задача бинарной классификации заключается в определении, существует ли причинно-следственная связь между двумя предложениями или нет. [9]

Например:

Прошёл дождь. Дорога стала скользкой $\rightarrow$ есть связь.

Собака лает. Солнце светит $\rightarrow$ нет связи.

Такие задачи требуют глубокого понимания семантики текста и часто решаются с помощью моделей на основе нейронных сетей, таких как BERT, LSTM, CNN и другие архитектуры [9].

Задача бинарной классификации играет важную роль в машинном обучении и науке о данных. Её формальная постановка позволяет строго подходить к процессу построения моделей, а использование вероятностного подхода обеспечивает интерпретируемость результатов. Благодаря широкому спектру приложений, начиная от финансового анализа и заканчивая обработкой естественного языка, задача бинарной классификации является ключевой для дальнейшего изучения методов машинного и глубокого обучения.

2.2 Основы алгоритмов машинного обучения для задач бинарной классификации

Задача бинарной классификации может быть решена с помощью множества алгоритмов машинного обучения, каждый из которых имеет свои особенности, преимущества и ограничения. Выбор алгоритма зависит от специфики задачи и подбирается индивидуально.

2.2.1 Логистическая регрессия

Формальная постановка:

Логистическая регрессия – это линейная модель, предназначенная для решения задач бинарной классификации. Она оценивает вероятность принадлежности объекта к положительному классу $y = 1$, используя сигмоидную функцию активации.

Модель определяется следующим образом:

$$P(y = 1 \mid x) = \sigma(\omega^T x + b),$$

где: $x \in R^n$ – вектор признаков, $\omega \in R^n$ – вектор весов, $b \in R$ – смещение (bias), $\sigma(z) = \frac{1}{1 + e^{-z}}$ – сигмоидная функция.

Класс предсказывается по порогу:

$$y_{pred} = \begin{cases} 1, & P(y = 1 \mid x \geq 0 \\ 0, & \end{cases}.$$

Обычно θ = 0.5.

Обучение модели:

Для обучения используется функция потерь binary cross-entropy:

$$L(y, \hat{y}) = -\frac{1}{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)],$$

где $\hat{y}_i = \sigma(\omega^T x_i + b)$.

Оптимизация проводится с использованием градиентного спуска или его модификаций:

$$\omega = \omega - \eta \frac{\partial L}{\omega},\ b = b - \eta \frac{\partial L}{b},$$

где η — шаг обучения (learning rate).

Преимущества: простота реализации, интерпретируемость коэффициентов, быстрое обучение.

Недостатки: предполагает линейную разделимость данных, не подходит для сложных нелинейных зависимостей без преобразования признаков.

2.2.2 Деревья решений

Формальная постановка:

Дерево решений строится путём рекурсивного разбиения выборки на подвыборки, начиная с корневой вершины. На каждом шаге выбирается признак, который наилучшим образом разделяет данные на однородные группы.

Критерии качества разбиения:

– Информационный выигрыш:

$$IG(Y, X) = H(Y) - H(Y|X),$$

где $H(Y)$ – энтропия целевой переменной, $H(Y|X)$ – условная энтропия.

– Gini Impurity:

$$G(p) = 1 - \sum_{i=1}^{k} p_i^2,$$

где p_i – доля объектов класса i в текущем узле.

Преимущества: интуитивно понятная визуализация, не требует масштабирования признаков, хорошо работает с категориальными данными.

Недостатки: склонность к переобучению, высокая вариация при небольших изменениях данных.

2.2.3 Случайный лес

Формальная постановка:

Случайный лес представляет собой ансамбль деревьев решений, каждое из которых обучается на случайной подвыборке исходных данных, а также на случайном подмножестве признаков. Итоговое решение принимается голосованием:

$$y_{pred} = mode(f_1(x), f_2(x), ..., f_T(x)),$$

где $f_t(x)$ – предсказание t-го дерева, T – количество деревьев.

Преимущества: устойчивость к переобучению, хорошая точность на большинстве задач, возможность оценки важности признаков.

Недостатки: требует больше вычислительных ресурсов, меньше интерпретируем, чем одно дерево.

2.2.4 Наивный байесовский классификатор (Naive Bayes)

Формальная постановка:

Наивный байесовский классификатор основан на теореме Байеса и предположении о независимости признаков:

$$P(y\left|x\right.) = \frac{P(x|y)P(y)}{P(x)}.$$

Преимущества: очень быстрое обучение, работает хорошо даже с малым количеством данных.

Недостатки: сильно зависит от предположения о независимости признаков, что редко выполняется на практике.

2.2.5 Метод k-ближайших соседей (kNN)

Формальная постановка:

kNN – непараметрический метод, основанный на хранении всей обучающей выборки. При предсказании для нового объекта x_{new} находятся k ближайших соседей из обучающей выборки, и класс определяется голосованием:

$$y_{pred} = \arg\max_c \sum_{i=1}^{k} c,$$

где c – индикаторная функция.

Расстояние обычно вычисляется по евклидовой метрике:

$$d(x_i, x_j) = \sqrt{\sum_{f=1}^{n} (x_{if} - x_{jf})^2}$$

Преимущества: прост в понимании и реализации, не требует этапа обучения.

Недостатки: высокая вычислительная сложность при большом объёме данных, чувствителен к шуму и масштабу признаков.

Существует множество алгоритмов машинного обучения, применимых к задаче бинарной классификации. Каждый из них имеет свою математическую основу, область применения и особенности поведения на различных типах дан-

ных. От выбора метода зависит не только качество модели, но и её интерпретируемость, скорость работы и возможность дальнейшего улучшения с помощью глубокого обучения.

2.3 Метрики качества обучения в задачах бинарной классификации

Одним из ключевых аспектов построения моделей машинного обучения является оценка их качества. В отличие от регрессионных задач, где оценка модели основывается на численном расхождении между предсказанными и истинными значениями, в бинарной классификации используется набор специализированных метрик, которые учитывают дискретную природу целевой переменной и могут учитывать несбалансированность классов.

Матрица ошибок

Для понимания большинства метрик необходимо рассматривать матрицу ошибок – таблицу, которая показывает, как часто модель правильно или неправильно классифицирует объекты. Общий вид матрицы ошибок представлена на рисунке 1.

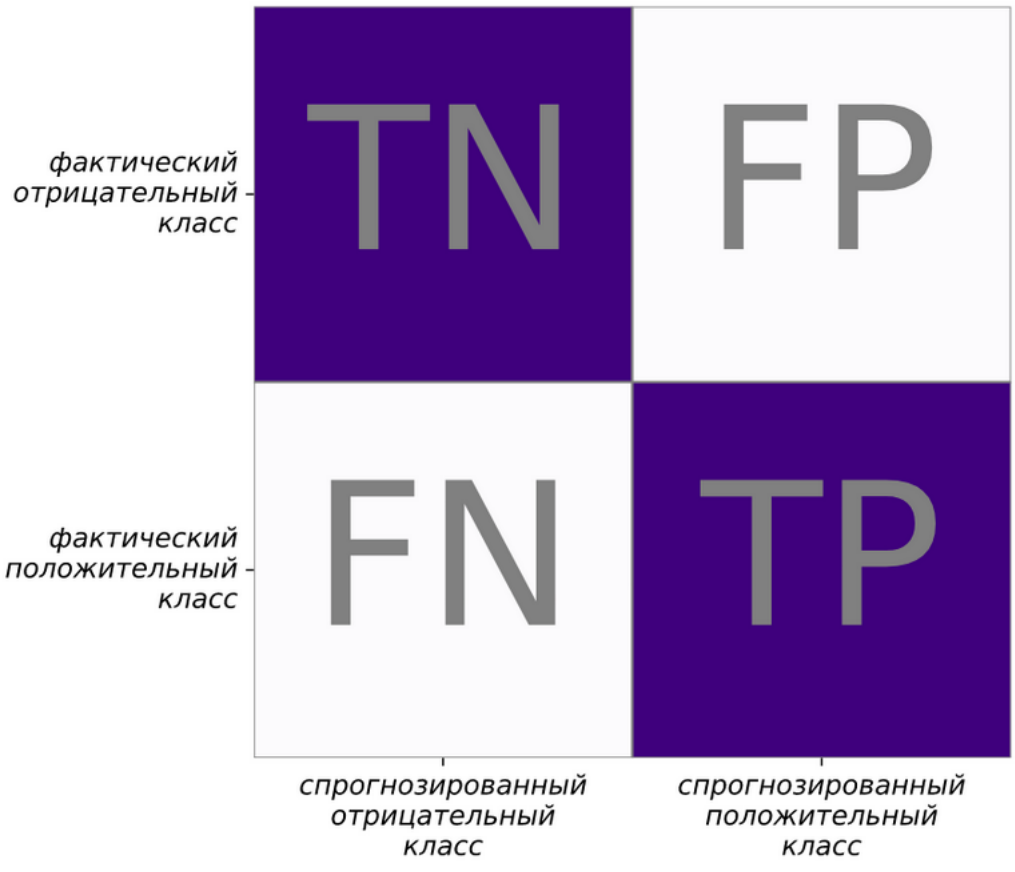

Рисунок 1 – Матрица ошибок

Обозначения на матрице ошибок: TP (True Positive) – модель правильно предсказала положительный класс, TN (True Negative) – модель правильно предсказала отрицательный класс, FP (False Positive) – модель ошибочно предсказала положительный класс, FN (False Negative) – модель ошибочно предсказала отрицательный класс.

На основе этих значений строятся все основные метрики качества.

Accuracy (точность)

Формула:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

Accuracy показывает долю правильных предсказаний среди всех сделанных. Это самая простая и интуитивно понятная метрика.

Accuracy может быть обманчивой, если классы сильно несбалансированы. Например, если 99% данных принадлежит классу 0, то модель, всегда возвращающая 0, будет иметь Accuracy = 99%, но при этом не будет полезной.

Precision (точность предсказаний положительного класса)

Формула:

$$Precision = \frac{TP}{TP + FP}.$$

Precision показывает, какая доля положительных предсказаний действительно относится к положительному классу. Иначе говоря, это мера того, насколько модель уверена в своих положительных прогнозах.

Если модель выдаёт 100 положительных результатов, из которых только 80 верны, Precision равен 0.8.

Recall (полнота, чувствительность)

Формула:

$$Recall = \frac{TP}{TP + FN}.$$

Recall показывает, какую долю положительных случаев модель смогла обнаружить. То есть, насколько полно модель определяет положительный класс.

Если всего в выборке 100 положительных объектов, а модель нашла только 75 из них, Recall равен 0.75.

F1-score (гармоническое среднее точности и полноты)

Формула:

$$F1 = 2\frac{Precision \times Recall}{Precision + Recall}.$$

F1-score представляет собой гармоническое среднее между Precision и Recall и позволяет сбалансировать эти два показателя. Он особенно полезен, когда важно учитывать, как ложноположительные, так и ложноотрицательные ошибки.

ROC-AUC (Area Under the ROC Curve)

ROC-кривая – это график зависимости True Positive Rate (TPR) от False Positive Rate (FPR) при изменении порога принятия решения.

Формулы:

$$TPR = \frac{TP}{TP + FN},\ FPR = \frac{FP}{FP + TN}.$$

ROC-AUC – это площадь под этой кривой и принимает значения от 0 до 1. Чем выше значение AUC, тем лучше модель различает классы. AUC = 1 – идеальная модель, AUC = 0.5 – случайное угадывание, AUC < 0.5 – модель хуже случайной.

ROC-AUC инвариантна к балансу классов и хорошо подходит для сравнения моделей.

PR-Curve и AUC-PR

Кроме ROC-кривой, также используется Precision-Recall Curve (PR-curve), которая показывает зависимость Precision от Recall при разных порогах.

AUC-PR – площадь под этой кривой – особенно полезна, когда данные сильно несбалансированы.

AUC-PR более информативна, чем ROC-AUC, в случае сильно несбалансированных данных.

В задачах бинарной классификации существует широкий спектр метрик, позволяющих адекватно оценить качество модели. Выбор метрики зависит от характера задачи, баланса классов, требований к типу ошибок и области применения. Такие метрики, как Accuracy, Precision, Recall и F1-score, позволяют оценить конкретные аспекты производительности модели, тогда как ROC-AUC и AUC-PR дают более общее представление о её способности различать классы.

2.4 Базовые концепции искусственных нейронных сетей

Искусственные нейронные сети (ANN) представляют собой вычислительные модели, вдохновлённые биологическими нейронными системами. Они состоят из множества взаимосвязанных элементов – искусственных нейронов, которые обрабатывают информацию с помощью нелинейных преобразований. Нейронные сети стали основой глубокого обучения и нашли широкое применение в задачах распознавания образов, обработки естественного языка, прогнозирования и других сложных задачах машинного обучения, включая бинарную классификацию.

2.4.1 Понятие искусственного нейрона

Искусственный нейрон является базовым элементом нейронной сети и представляет собой упрощённую математическую модель биологического нейрона. Он принимает входные сигналы, взвешивает их, суммирует и передаёт результат через функцию активации, чтобы получить выходное значение. Подобным образом работает биологический нейрон в мозге, принимая входные сигналы через специальные каналы – дендриты, и выдающие результат обработки импульса чрез концевую ветвь аксона. Визуализация биологического нейрона и математической модели искусственного нейрона представлены на рисунках 2 и 3 соответственно.

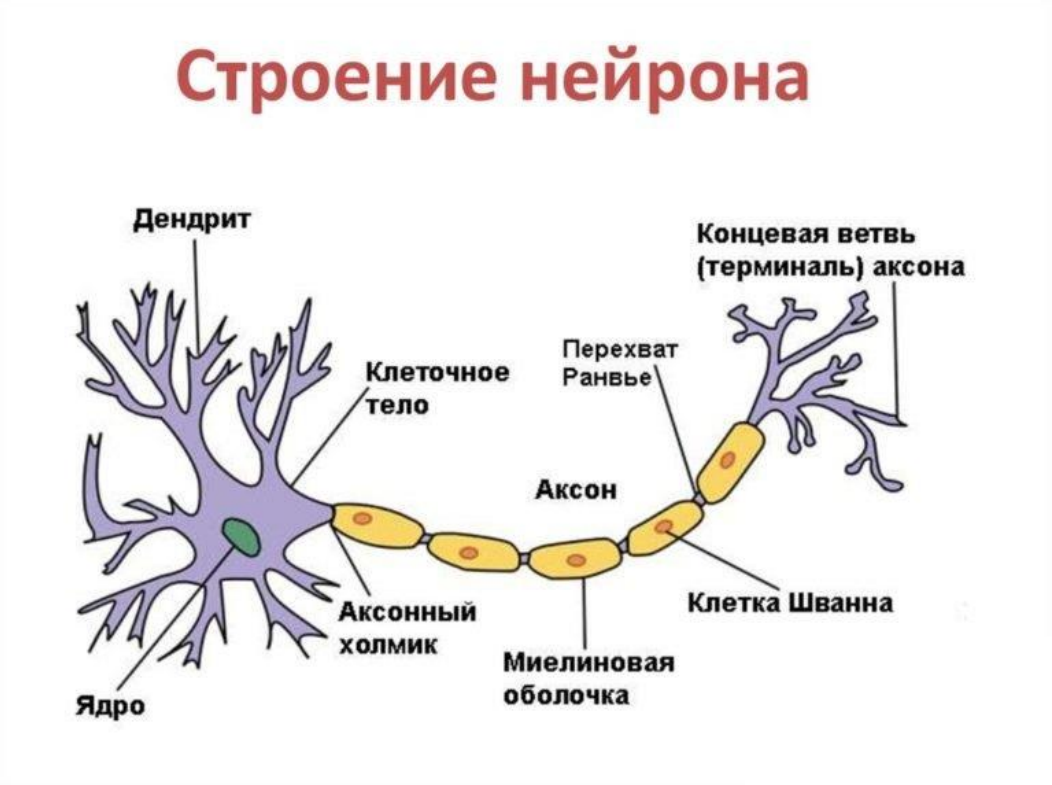

Рисунок 2 – Биологический нейрон

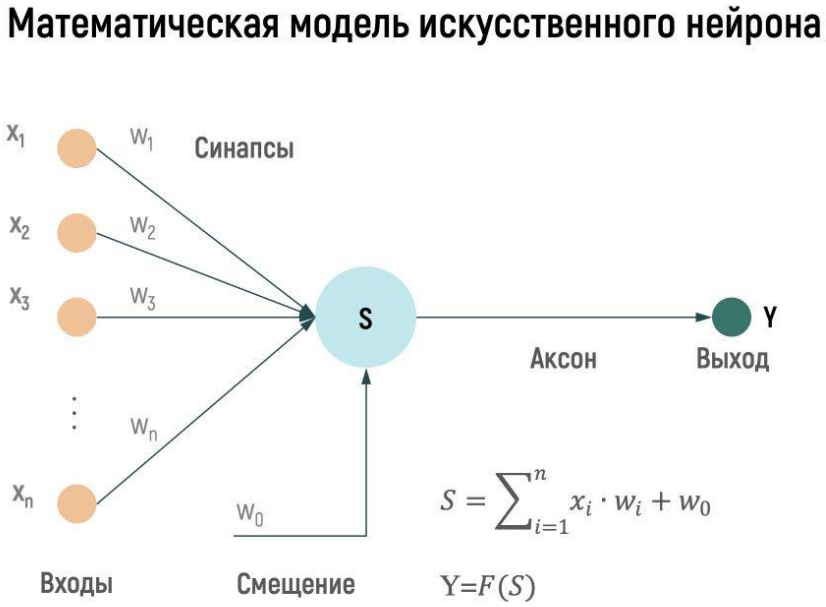

Рисунок 3 – Искусственный нейрон

Формальная постановка:

Пусть $x = (x_1, x_2, ..., x_n) \in R^n$ – входной вектор признаков, $\omega = (\omega_1, \omega_2, ..., \omega_n) \in R^n$ – вектор весов, $b \in R^n$ – смещение (bias).

Выход нейрона определяется следующим образом:

$$y = f(\sum_{i=1}^{n} \omega_i x_i + b),$$

где f – функция активации.

Основные типы функций активации:

Сигмоидная функция (Sigmoid):

$$f(z) = \frac{1}{1+e^{-z}}.$$

Преобразует входной сигнал в диапазон (0, 1), используется в задачах бинарной классификации.

Гиперболический тангенс (Tanh):

$$f(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}.$$

Выходной сигнал находится в диапазоне (-1, 1). Предпочтительнее сигмоиды в скрытых слоях из-за центрированности.

ReLU (Rectified Linear Unit):

$$f(z) = \max(0, z).$$

Наиболее популярная функция активации в современных нейронных сетях благодаря своей простоте и эффективности.

Softmax:

Используется на последнем слое при многоклассовой классификации:

$$f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}},\ i = 1,...,k.$$

Обучение нейрона:

Обучение заключается в подборе параметров ω и b таким образом, чтобы минимизировать ошибку предсказания на обучающей выборке. Для этого используется функция потерь и метод градиентного спуска.

Например, если использовать бинарную кросс-энтропию как функцию потерь:

$$L(y, \hat{y}) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y}),$$

то обучение проводится с использованием обратного распространения ошибки, где вычисляются частные производные функции потерь по весам и смещению:

$$\frac{\partial L}{\partial \omega_i} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial \omega_i}, \; \frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial b},$$

и параметры обновляются по правилу:

$$\omega_i = \omega_i - \eta \frac{\partial L}{\partial \omega_i}, \; b = b - \eta \frac{\partial L}{\partial b},$$

где η – скорость обучения (learning rate).

2.4.2 Архитектуры нейронных сетей

Архитектура нейронной сети определяет её структуру: количество слоёв, количество нейронов в каждом слое, способ соединения между слоями и тип используемых функций активации.

Персептрон (Perceptron)

Персептрон – это простейшая нейронная сеть, состоящая из одного нейрона. Используется для линейной бинарной классификации. Обучается с помощью правила персептрона:

$$\omega_i = \omega_i + \eta(y - \hat{y})x_i,$$

где y – истинная метка, $\hat{y}$ – предсказанная метка.

Многослойный персептрон (MLP)

Многослойный персептрон состоит из входного слоя, одного или нескольких скрытых слоёв и выходного слоя.

Каждый слой содержит множество нейронов, соединённых с нейронами соседних слоёв. Все связи имеют свои веса, и каждый нейрон применяет функцию активации.

Входной слой представляет собой начальный уровень сети, на который подаются исходные данные. Каждый нейрон этого слоя соответствует одному признаку объекта. Например, если задача заключается в классификации текста, то входными данными могут быть числовые представления слов или предложений [4].

Для входного вектора $x \in R^n$, весов первого слоя $W^{(1)} \in R^{dn}$ и смещений $b^{(1)} \in R^d$:

$$h^{(1)} = f(W^{(1)}x + b^{(1)}),$$

где $h^{(1)} \in R^d$ – выход первого скрытого слоя.

Скрытые слои находятся между входным и выходным слоями. Они отвечают за извлечение скрытых закономерностей и построение нелинейных зависимостей. Чем больше количество скрытых слоёв, тем сложнее модель может быть, что позволяет ей лучше справляться с нетривиальными задачами, такими как анализ естественного языка, компьютерное зрение и прогнозирование временных рядов [4].

Для второго слоя:

$$h^{(2)} = f(W^{(2)}h^{(1)} + b^{(2)}).$$

Выходной слой содержит один или несколько нейронов, в зависимости от задачи. Например, для бинарной классификации используется один нейрон с сигмоидной функцией активации. Для многоклассовой классификации – несколько нейронов с функцией Softmax. Для регрессии – один или несколько нейронов без активации или с линейной активацией [4].

На выходном слое:

$$\hat{y} = f(W^{(L)}h^{(L-1)} + b^{(L)}).$$

MLP позволяет моделировать нелинейные зависимости и широко применяется в задачах классификации и регрессии.

Свёрточные нейронные сети (CNN)

CNN предназначены для работы с данными, имеющими пространственную структуру (например, изображения, текст). Они содержат свёрточные слои, в которых применяются фильтры (ядра) для извлечения локальных признаков [16].

В задачах классификации текста, например, свёртки могут быть применены к векторным представлениям слов для выявления важных семантических паттернов.

$$c_i = f(\sum_{j=1}^{k} \omega_j x_{i+j} + b),$$

где k – размер фильтра, c_i – элемент выходного вектора.

CNN часто используются вместе с операциями пулинга (pooling), такими как max-pooling. Операция пулинга – это один из ключевых элементов свёрточных нейронных сетей (CNN), который используется для уменьшения размерности данных, выделения наиболее значимых признаков и повышения устойчивости модели к небольшим сдвигам и деформациям входного сигнала. Пулинг применяется после свёрточных слоёв и позволяет агрегировать информацию внутри локальных регионов.

Целью операции пулинга является снижение размерности – уменьшение числа параметров и вычислений в сети, увеличение инвариантности – модель становится менее чувствительной к мелким изменениям позиции объекта в пространстве (например, небольшой сдвиг текста или изображения), обобщение признаков – выделение наиболее значимых характеристик в локальной области:

$$p = \max(c_i, c_{i+1}, ..., c_{i+k})$$

Рекуррентные нейронные сети (RNN)

RNN разработаны для обработки последовательностей данных, таких как временные ряды или текст. В отличие от MLP, RNN сохраняет внутреннее состояние h_t, которое зависит от текущего входа x_t и предыдущего состояния h_{t-1} [16].

$$h_t = f(W_h h_{t-1} + W_x x_t + b),\ y_t = g(W_y h_t + b_y),$$

где f и g – функции активации.

Однако простые RNN страдают от проблемы исчезающего градиента, поэтому чаще используются более сложные варианты, такие как LSTM и GRU.

LSTM (Long Short-Term Memory):

LSTM содержит три ключевых компонента: входной затвор i_t, забывание f_t и выходной затвор o_t:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f),$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_f),$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C),$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t,$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o),$$

$$h_t = o_t \odot \tanh(C_t),$$

где $\odot$ – поэлементное умножение.

GRU (Gated Recurrent Unit):

GRU – упрощённая версия LSTM, объединяющая входной и забывательный затворы в единую функцию обновления:

$$z_t = \sigma(W_z[h_{t-1}, x_t]),$$

$$r_t = \sigma(W_r[h_{t-1}, x_t]),$$

$$\tilde{h}_t = \tanh(W[r_t \odot h_{t-1}, x_t]),$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t.$$

Трансформеры (Transformers)

Трансформеры – относительно новая, но очень мощная архитектура, которая использует механизм внимания (attention) вместо рекуррентных связей.

Self-Attention:

Для заданного набора входных векторов $Q, K, V \in R^{nd}$, вычисления выглядят так:

$$Attention(Q, K, V) = soft\max(\frac{QK^T}{\sqrt{d_k}})V,$$

где d_k – масштабирующий коэффициент.

Трансформеры легли в основу моделей BERT, RoBERTa, GPT и других SOTA решений в NLP, включая задачи бинарной классификации текста.

Искусственные нейронные сети строятся на основе простых элементов – искусственных нейронов, которые выполняют взвешенное суммирование входов и применяют нелинейную функцию активации. Совокупность таких нейронов образует слои, а совокупность слоёв – архитектуру сети.

Выбор архитектуры нейронной сети зависит от типа данных, характера задачи и доступных вычислительных ресурсов. Для задач бинарной классификации, особенно в области обработки естественного языка, наиболее популярны многослойные персептроны, рекуррентные сети (LSTM, GRU) и трансформеры.

2.5 Библиотеки Python для построения и обучения нейронных сетей

Python стал стандартным языком программирования в области машинного и глубокого обучения благодаря своей простоте, читаемости кода и богатой экосистеме библиотек. В частности, существует множество мощных фреймворков и инструментов, которые позволяют строить, обучать и оценивать нейронные сети различной сложности – от простых многослойных персептронов до сложных трансформеров. Эти библиотеки обеспечивают как низкоуровневую реализацию моделей, так и высокоуровневые API, упрощающие разработку и эксперименты [14, 15, 17].

TensorFlow

TensorFlow – это открытая библиотека для численных вычислений и машинного обучения, разработанная Google Brain Team. Она предоставляет гибкую платформу для создания моделей глубокого обучения, поддерживает GPU/TPU-ускорение и позволяет работать как на уровне операций с тензорами, так и с использованием высокоуровневых API, таких как Keras.

Основные возможности: поддержка автоматического дифференцирования, гибкая графовая модель вычислений (Eager Execution и графы), интеграция с Keras для удобного построения моделей, возможность распределённого обучения и оптимизации для производительности.

Преимуществами являются широкое комьюнити и документация, поддержка мобильных и встраиваемых устройств через TensorFlow Lite и возможность сохранения и загрузки моделей в форматах SavedModel и HDF5.

К недостаткам относятся сложность в освоении по сравнению с PyTorch из-за графовой модели вычислений (в режиме не Eager Execution) и меньшая интерактивность при исследовательских задачах.

PyTorch

PyTorch – это библиотека с открытым исходным кодом, разработанная Facebook (Meta), которая стала особенно популярной в научном сообществе благодаря своей динамичности вычислений, что делает её особенно удобной для исследования и экспериментов.

Основные возможности: динамическое построение вычислительных графов, поддержка автоматического дифференцирования через torch.autograd, интеграция с CUDA для работы на GPU, модульность: можно легко создавать собственные слои и архитектуры.

Преимуществами являются высокая гибкость и возможность быстрого прототипирования, активное развитие и поддержка сообщества.

Недостатки – меньше готовых решений для промышленного развёртывания по сравнению с TensorFlow, менее зрелые инструменты для сериализации и оптимизации модели.

Keras

Keras – это высокоуровневый API для построения и обучения нейронных сетей. Он может работать поверх TensorFlow, Theano или CNTK (на данный момент официально поддерживается только TensorFlow). Keras отличается простотой использования и позволяет быстро строить модели даже новичкам [1, 10].

Основные возможности: позволяет создавать модели с помощью Sequential API или Functional API, поддерживает обучение на CPU и GPU, обширная коллекция предобученных моделей и набор callback-функций.

Преимущества: простота освоения и использования, хорошая документация, подходит для быстрой реализации и тестирования моделей.

Недостатки: меньше контроля над деталями обучения по сравнению с TensorFlow и PyTorch, не предназначен для исследований сложных архитектур.

Hugging Face Transformers

Hugging Face Transformers – это библиотека, предоставляющая доступ к сотням предобученных моделей NLP, основанных на архитектуре трансформера. Библиотека активно развивается и поддерживает как PyTorch, так и TensorFlow.

Основные возможности: загрузка и использование предобученных моделей (BERT, RoBERTa, GPT, T5 и др.), автоматическая обработка текста с помощью Tokenizer, поддержка fine-tuning, inference и pipeline-интерфейса, удобство работы с корпусами данных через datasets.

Преимущества: широкая коллекция моделей, простая интеграция с другими фреймворками, поддержка NLP-специфичных функций (токенизация, пайплайны, метрики).

Недостатки: требует понимания базовых принципов работы трансформеров, может быть избыточной для задач вне NLP.

Scikit-learn

Хотя scikit-learn не является фреймворком для глубокого обучения, он играет важную роль в подготовке данных перед обучением нейронных сетей и в оценке качества моделей.

Основные возможности: предобработка данных: StandardScaler, LabelEncoder, OneHotEncoder, разделение выборки: train_test_split, оценка моделей: accuracy_score, classification_report, confusion_matrix и др.

Преимущества: простота и скорость, богатые средства оценки и визуализации, совместимость с NumPy и Pandas.

Недостатки: не поддерживает глубокое обучение, не предназначен для работы с большими объёмами данных.

Fast.ai

Fast.ai – это высокоуровневая библиотека, построенная на основе PyTorch, которая упрощает процесс обучения моделей и содержит готовые решения для компьютерного зрения, обработки естественного языка и рекомендательных систем.

Основные возможности: автоматический подбор гиперпараметров и скорости обучения, поддержка transfer learning и fine-tuning, встроенные методы визуализации и анализа ошибок.

Преимущества: высокий уровень абстракции, подходит для быстрого старта и обучения.

Недостатки: меньше контроля по сравнению с PyTorch, меньше подходит для продвинутых исследований.

ONNX (Open Neural Network Exchange)

Описание:

ONNX – это открытый формат представления моделей машинного и глубокого обучения, который обеспечивает совместимость между различными фреймворками. Это позволяет экспортировать модели, обученные в одной библиотеке, и использовать их в другой среде.

Основные возможности: экспорт моделей из PyTorch, TensorFlow, Keras и других, поддержка оптимизации и выполнения на разных устройствах, поддержка ONNX Runtime для высокопроизводительного вывода.

Преимущества: переносимость между фреймворками, поддержка оптимизации и deployment.

Недостатки: требует дополнительных усилий для корректного экспорта, не все слои поддерживаются напрямую.

DVC (Data Version Control)

DVC – это система управления данными и ML-экспериментами, аналогично Git для кода, но для данных и моделей. Она помогает отслеживать изменения в наборах данных, моделях и метриках.

Основные возможности: версионирование данных и моделей, воспроизводимость экспериментов, интеграция с Git и CI/CD.

Преимущества: удобство отслеживания изменений, поддержка облачного хранения (AWS, GCP, Azure).

Недостатки: не обучает модели, а управляет ими, требует понимания версионного подхода.

MLflow

Основные возможности: отслеживание параметров и метрик экспериментов, управление жизненным циклом модели, развёртывание моделей в production.

Преимущества: поддержка нескольких фреймворков, интерфейс веб-панели, поддержка REST API для взаимодействия.

Недостатки: требует запуска сервера, не предназначен для непосредственного построения моделей.

Jupyter Notebook / Colab / Kaggle Kernel

Хотя технически это не библиотеки, эти интерактивные среды являются ключевыми инструментами для разработки и тестирования нейронных сетей.

Преимущества: интерактивная работа с данными, визуализация результатов, поддержка GPU/TPU.

Недостатки: не предназначены для production-разработки, ограниченная масштабируемость.

Python предлагает широкий спектр библиотек и фреймворков, позволяющих эффективно строить, обучать и оценивать нейронные сети. TensorFlow и PyTorch являются лидерами в области глубокого обучения, тогда как Keras и Hugging Face Transformers обеспечивают удобство работы с уже готовыми архитектурами. Дополнительные инструменты, такие как ONNX, DVC и MLflow, позволяют управлять жизненным циклом моделей и обеспечивают воспроизводимость экспериментов.

Выбор конкретной библиотеки зависит от целей проекта: для исследований и экспериментов предпочтителен PyTorch, для промышленной реализации – TensorFlow или Keras, для задач обработки естественного языка – Hugging Face Transformers.

3 НЕЙРОННАЯ СЕТЬ ДЛЯ АНАЛИЗА ТЕКСТОВЫХ СООБЩЕНИЙ

3.1 Постановка задачи

3.1.1 Содержательная и концептуальная постановка задачи

Данные представляют собой набор текстовых данных, состоящий из:

– Фраза-предпосылка – исходное утверждение, которое несёт в себе определённую смысловую нагрузку.

– Две фразы-следствия – два возможных продолжения исходной фразы-предпосылки, где одно возможное продолжение связано логически с предпосылкой, а другое – нет.

– Метки – Одно из двух значений, определяющее, какая фраза-следствие является продолжением фразы-предпосылки. 0, если правильной является первая фраза, 1, если вторая.

Пример представления данных:

– Фраза-предпосылка: «Моё тело отбрасывает тень на траву.»;

– Возможное следствие 1: «Солнце уже поднялось.»;

– Возможное следствие 2: «Трава уже подстрижена.»;

– Метка: 0 (Верной предпосылкой является первая).

Модель должна анализировать все три имеющиеся фразы и определять, какая из двух фраз-следствий является логическим продолжением исходной фразы-предпосылки. Для этого модель должна обучиться на наборе размеченных данных, чтобы впоследствии правильно классифицировать новые входные данные [19].

Концептуально задача представляет собой задачу машинного обучения с учителем. Конкретно, это задача бинарной классификации, где входные данные – это пара текстовых последовательностей (p_i, c_{i1}) и (p_i, c_{i2}), в которых p_i – фраза-предпосылка, а c_{i1} и c_{i2} – это возможные фразы-следствия. Выходными

данными является метка $y_i \in \{0,1\}$, указывающая, какое из двух следствий логически связано с предпосылкой. Целью задачи является определить, какое из двух следствий логически связано с предпосылкой.

3.1.2 Математическая постановка задачи

Математически задача может быть сформулирована следующим образом:

$P = \{p_1, p_2, \ldots, p_n\}$ – множество сообщений-предпосылок, каждое сообщение p_i представлено в виде последовательности токенов.

$C = \{(c_{11}, c_{12}), (c_{21}, c_{22}), \ldots, (c_{n1}, c_{n2})\}$ – множество фраз-следствий,

где c_{i1} и c_{i2} – два возможных следствия для предпосылки p_i.

$Y = \{y_1, y_2, \ldots, y_n\}$ – множество меток,

где $y_i \in \{0,1\}$: $y_i = 0$, если c_{i1} является следствием p_i; $y_i = 1$, если c_{i2} является следствием p_i

Цель: найти функцию $f : (P, C) \to Y$ которая для каждой пары $(p_i, (c_{i1}, c_{i2}))$ предсказывает метку y_i, указывающую, какая из двух фраз c_{i1} или c_{i2} является следствием предпосылки p_i.

Обучение модели: для обучения модели используется набор данных:

$D = \{(p_i, (c_{i1}, c_{i2}), y_i)\}_{i=1}^{N}$,

где N – количество обучающих примеров.

Модель будет обучаться с использованием алгоритма обратного распространения ошибки, минимизируя функцию потерь:

$L(f(p_i, (c_{i1}, c_{i2})), y_i)$,

где L – функция потерь, отражающая разницу между предсказанным и истинным значением.

Оценка качества: для оценки качества работы модели будут использоваться метрики, такие как точность, полнота и F1-мера, которые позволят количественно оценить эффективность классификации текстовых сообщений. Также будет использована матрица ошибок, для количественного наблюдения классов, которые модель различает правильно и на которых ошибается.

3.2 Предобработка текста

Предобработка текстовых данных является одним из ключевых этапов в задачах NLP, так как исходные данные зачастую представлены в виде неструктурированного текста. Такой формат вполне понятен для человека, но для обучения нейронной сети необходимо привести данные к виду, который будет корректно восприниматься моделью и легко преобразован для чтения компьютером.

3.2.1 Очистка

Очистка текстовых данных представляет собой удаление лишних символов, нормализацию строк и устранение ошибок в данных. Этот этап позволяет повысить качество представления информации и снизить влияние помех в данных на работу модели.

Основные задачи, которые ставятся на данном этапе:

Удаление специальных символов и HTML-разметки.

В исходных данных могут встречаться символы, не относящиеся к содержательной части текста, такие как теги разметки, эмодзи, артефакты форматирования и прочее. Их присутствие может отвлекать модель от реальных закономерностей и вносить шумы в представления признаков, тем самым ухудшая качество работы модели.

Приведение текста к единому регистру.

Языки, особенно русский, очень чувствительны к регистру, но не всегда это влияет на смысл. Поэтому для того, чтобы, во-первых, уменьшить размер словаря и, во-вторых, сделать модель менее чувствительной к случайным различиям в написании слов, весь текст переводится в нижний регистр.

Удаление знаков препинания.

Удаление пунктуации позволяет уменьшить количество уникальных токенов и повысить устойчивость модели к вариациям в оформлении текста. Наличие знаков препинания в тексте важно для человека, хоть и не всегда, но для обучения модели они не несут какой-либо смысловой нагрузки, поэтому для упрощения обучения и повышения качества работы модели производится удаление всех знаков препинания.

Для реализации этапа очистки текста в Python была использована библиотека re (regular expressions), а также стандартные функции Python для работы со строками. Пример данных до и после очистки представлен в таблице 1.

Таблица 1 – Очистка данных

Текст до очистки	Текст после очистки
Из-за дождя дорога стала скользкой.	из-за дождя дорога стала скользкой
Солнце светит, птицы поют.	солнце светит птицы поют
Я проснулся… Потом я пошёл гулять!	я проснулся потом я пошёл гулять
Тестирование: Проверка связи?	тестирование проверка связи

Такие изменения позволяют значительно улучшить качество входных данных и повысить продуктивность последующих этапов предобработки текста.

Механизм реализации этапа очистки не был реализован вручную, однако полностью соответствующий изложенному выше описанию алгоритм очистки внедрён в инструментарий модели BERT, используемой в работе. Поэтому этап очистки текстовых данных не был пропущен, а был реализован неявным образом.

3.2.2 Токенизация

Токенизация – процесс разбиения текста на отдельные лексические элементы: слова, подслова, символы или фразы. Эти элементы затем сопоставляются с индексами из словаря модели, что позволяет использовать их в качестве входных данных для нейронной сети. Таким образом последовательности символов преобразуются в дискретные единицы (токены) и используются моделью.

Токенизация может быть выполнена по-разному, в зависимости от того, какая модель используется. Токенами могут быть целые слова (Word-level tokenization), части слов (Subword tokenization) или отдельные символы (Character-level tokenization). Пример токенизированных данных приведён в таблице 2.

Таблица 2 – Токенизация текста

Вид токенизации	Текст до токенизации	Текст после токенизации
Word-level tokenization	Из-за дождя дорога стала скользкой	[из-за, дождя, дорога, стала, скользкой]
Subword tokenization	Обезьяноподобный	[обезьян, о, подобн, ый]
Character-level tokenization	Семь	[с, е, м, ь]
Комбинированный	Пошёл дождь	[пошёл, до, ж, дь]

В данной работе токенизация не проводилась вручную, она происходила неявно. Для реализации использовалась модель Sentence-BERT с библиотекой sentence-transformers, которая сама вызывает токенизатор, поддерживающий русский язык, что позволяет производить токенизацию качественно. Модель использует субсловную токенизацию, что позволяет ей эффективно обрабатывать формы слов в разных падежах, временах и лицах.

3.2.3 Векторизация

После очистки и токенизации наступает этап векторизации, суть которого заключается в преобразовании слов или целых предложений в числовое представление, пригодное для работы нейронной сети. Этот шаг является особенно важным, так как большинство моделей машинного обучения оперируют числами, а не строками, и качество этого преобразования напрямую влияет на эффективность модели.

Задача векторизации заключается в том, чтобы создать плотные числовые представления текста, которые будут отражать семантическое сходство между предложениями, будут интерпретируемыми моделью глубокого обучения и будут обладать устойчивостью к вариациям в формулировках одного и того же события.

Числовые представления, получаемые на этапе векторизации, называются эмбеддингами. Эмбеддинг – числовой вектор фиксированной длины, который представляет собой компактное и информативное числовое представление некоторой единицы данных: слова, предложения или даже абзаца. Эмбеддинги поз-

воляют учитывать семантическую близость между различными элементами текста. Например, если два слова имеют похожий смысл, то их эмбеддинги будут находиться близко друг к другу в многомерном пространстве.

Математически эмбеддинг представляется так:

Для любого текстового элемента $s \in L$ эмбеддинг определяется как отображение:

$$Embedding(s) = v \in R^d,$$

где L – множество всех возможных строк на заданном языке, d – размерность эмбеддинга, v – вещественный вектор, описывающий семантику текста.

В данной работе для векторизации текста было использовано семейство библиотек SBERT, заранее предобученная под русский язык. Данная модель получает эмбеддинги, превращая всё предложение в один вектор, учитывающий его смысл и структуру. Пример получения эмбеддинга: *Машина попала в аварию* $\rightarrow [0.42, 0.11, ..., -0.09]$.

3.3 Архитектура нейронной сети и настройка гиперпараметров

После этапа подготовки данных следует построение самой нейронной сети, предназначенной для классификации причинно-следственной связи между фразами. Для решения данной задачи в работе была использована специальная архитектура Siamese Network, которая позволяет сравнивать два возможных следствия с предпосылкой и выбирать наиболее логичное из них.

Siamese Network представляет собой тип нейронной сети, состоящей из двух или более одинаковых подсетей, которые обрабатывают разные входы и сравнивают их на уровне скрытых признаков. Такой подход обусловлен эффективностью сравнения пар объектов.

В данной работе Siamese Network использовался в модифицированном виде:

– Три входа: предпосылка, первое следствие, второе следствие;

– Для каждого из следствия вычисляется разность с предпосылкой, эти разности объединяются и передаются на полносвязную часть сети, которая делает окончательное решение о наличии причинно-следственной связи.

Такая организация позволила модели явно сравнивать каждое следствие с предпосылкой.

3.3.1 Слои и компоненты модели

Входные слои.

Модель имеет три отдельных входа: Предпосылка, первое следствие, второе следствие. Каждый вход представляет собой числовой вектор размерности 384, полученный на этапе векторизации с помощью Sentence-BERT.

Для каждого из двух следствий вычисляется разностный вектор относительно предпосылки:

$$\Delta_1 = v_p - v_1;$$

$$\Delta_2 = v_p - v_2,$$

где v_p – эмбеддинг предпосылки, v_1 и v_2 – эмбеддинги первого и второго следствий соответственно.

Разностные векторы объединяются в единый входной сигнал для полносвязной части модели:

$$x = [\Delta_1; \Delta_2] \in R^{768}.$$

Такое представление содержит информацию о том, насколько каждое из следствий отличается от исходного события и даёт возможность модели сравнивать эти различия между собой.

Эти операции позволяют модели сравнить степень семантического отличия каждого из следствий от предпосылки, что служит основой для дальнейшей классификации.

Полносвязные слои.

После объединения данные поступают на несколько полносвязных слоёв, каждый из которых выполняет преобразование:

$$h^{(l)} = f(W^{(l)} h^{(l-1)} + b^{(l)}),$$

где $h^{(l)}$ – выход l-го слоя, $W^{(l)}$ и $b^{(l)}$ – веса и смещения, f – функция активации.

Были использованы два скрытых слоя: первый – с 256 нейронами, и второй – с 128 нейронами. В обоих слоях использовалась ReLU-активация.

ReLU-функция $f(z) = \max(0, z)$ обеспечивает модель нелинейностью и устойчивостью к проблеме исчезающего градиента.

После каждого полносвязного слоя применялся Batch Normalization, который нормализует значения входящих данных по формуле:

$$\hat{x} = \frac{x - \mu}{\sigma},$$

где μ и σ – среднее значение и стандартное отклонение.

Batch Normalization способствует ускорению сходимости за счёт нормализации входных значений, снижению зависимости от начального значения весов, повышению устойчивости к вариации масштаба входных данных.

Для регуляризации и предотвращения переобучения после каждого активационного слоя применяется Dropout со значением коэффициента 0.5:

$$h_i^{'} = \begin{cases} 0, & \textit{с вероятностью } p \\ \dfrac{h_i}{1-p}, & \textit{иначе} \end{cases}.$$

Это позволило повысить обобщающую способность модели, уменьшив зависимость от отдельных нейронов.

Выходной слой.

На выходном слое используется один нейрон с сигмоидной функцией активации:

$$y_{pred} = \sigma(Wy + b),$$

где $\sigma(z) = \dfrac{1}{1+e^{-z}}$ – сигмоидная функция, возвращающая вероятность принадлежности к положительному классу.

3.3.2 Функция потерь, оптимизация и гиперпараметры

Для данной задачи была использована функция потерь binary cross-entropy loss, которая определяется как:

$$L(y, \hat{y}) = -[y \log(\hat{y}) + (1-y)\log(1-\hat{y})].$$

Эта функция потерь хорошо работает с сигмоидной активацией и чувствительна к ошибкам в обоих направлениях: ложноположительным и ложноотрицательным.

Для минимизации функции потерь использовался оптимизатор Adam – алгоритм стохастического градиентного спуска, адаптирующий скорость обучения для каждого параметра.

Обновление весов:

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon},$$

где m_t – оценка первого момента (среднего), v_t – оценка второго момента (дисперсии), η – темп обучения (learning rate), ε – малое число для стабилизации вычислений.

Было установлено значение η равным 10^{-4}, что обеспечивало стабильное обучение без риска взрывного роста градиентов.

Для предотвращения переобучения были приняты меры контроля над переобучением. Был внедрён механизм ранней остановки, который прекращает обучение, если качество на валидационной выборке не улучшается за заданное количество эпох.

Итоговая модель была обучена с использованием специальных гиперпараметров, представленных в таблице 3.

Таблица 3 – Гиперпараметры модели

Параметр	Значение
Оптимизатор	Adam
Learning Rate	10^{-14}
Функция потерь	Binary Cross-Entropy
Метрики	Accuracy, Recall, F1-score, матрица ошибок.
Размер батча	64
Число эпох	50 (с ранней остановкой)
Dropout	0.5
Эмбеддинги	384
Скрытые слои	256, 128, ReLU
Выходной слой	1, Sigmoid

Обучение модели производилось на основе заранее подготовленных эмбеддингов, полученных с помощью модели BERT. Это позволило значительно ускорить процесс, так как модель не обучалась заново.

Процесс обучения состоял из следующих этапов:

– Инициализация весов: веса инициализировались случайными значениями согласно стандартной инициализации Glorot;

– Прямое распространение: данные проходили через все слои;

– Вычисление ошибки: ошибка рассчитывалась с использованием binary cross-entropy;

– Обратное распространение: градиенты вычислялись и применялись для корректировки весов;

– Валидация: на каждом шаге оценивалось качество модели на валидационной выборке;

– Ранняя остановка: обучение прекращалось, если перестаёт наблюдаться улучшение на валидационной выборке.

3.4 Анализ результатов обучения

После завершения этапа обучения нейронной сети была проведена полная оценка её эффективности на валидационной выборке. Основные задачи, которые ставятся на данном этапе:

– Определение качества модели в контексте задачи;

– Выявление слабых мест модели;

– Сравнение различных метрик и их интерпретация в контексте задачи;

– Визуализация результатов для наглядного анализа.

Для оценки качества работы нейронной сети использовались основные метрики оценки: доля верных ответов (Accuracy), точность (Precision), полнота (Recall), F1-score (гармоническое среднее между Precision и Recall), матрица ошибок.

На рисунке 4 представлена динамика метрики Accuracy на обучающей и валидационных выборках в зависимости от эпох обучения.

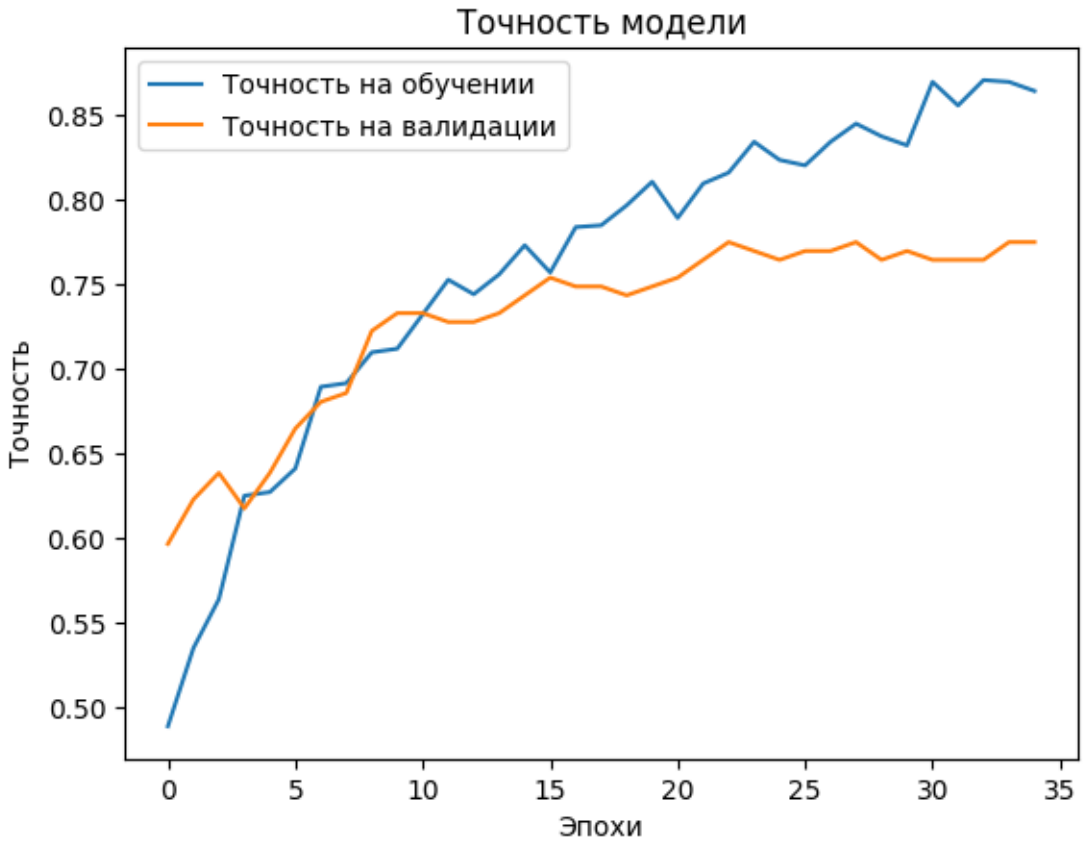

Рисунок 4 – Точность модели (Accuracy)

Обучающая точность демонстрирует постепенное повышение с течением времени, что указывает на корректное обучение. Валидационная точность стабилизируется на уровне около 77%, что говорит о хорошей обобщающей способности модели.

На рисунке 5 отражено значение функции потерь на разных этапах обучения.

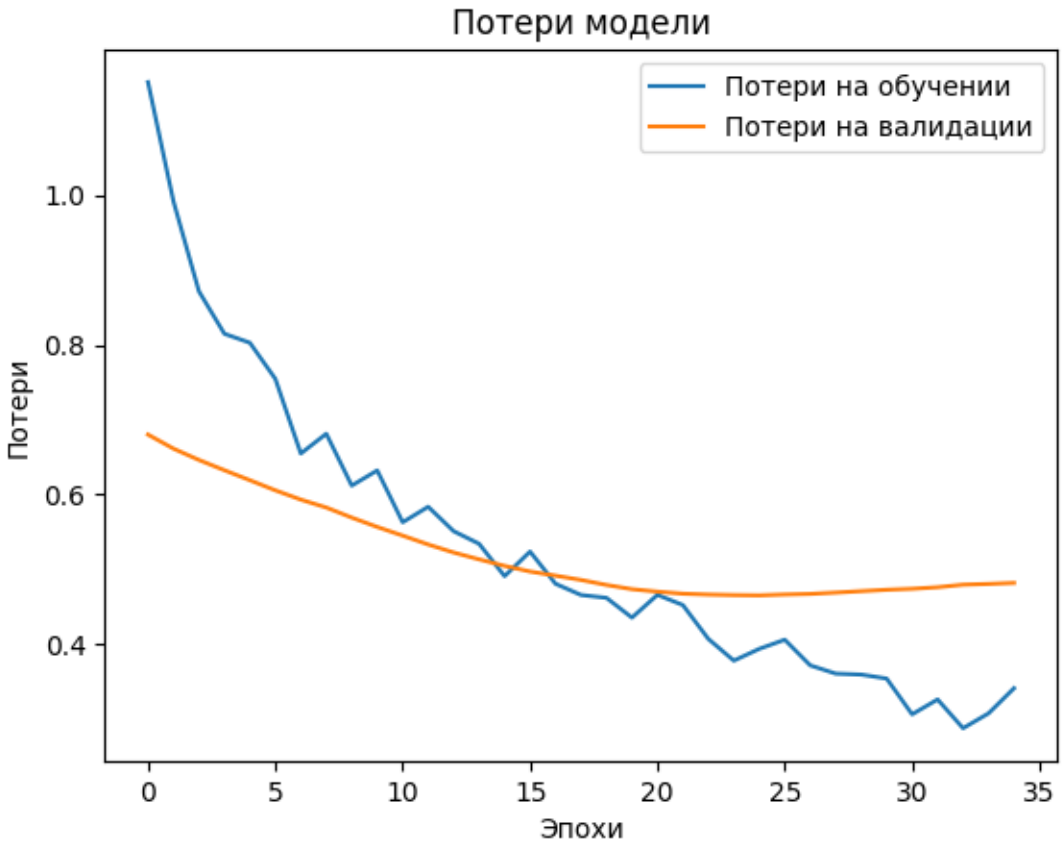

Рисунок 5 – Потери модели

Обучающие потери снижаются в первые несколько эпох и затем стабилизируются. Валидационные потери также уменьшаются и не растут к концу обучения, что подтверждает отсутствие переобучения.

В общем виде, эти два графика в совокупности свидетельствуют о том, что модель обучалась стабильно, без значительного расхождения между обучающей и валидационной выборками.

Для оценки эффективности модели были рассчитаны ключевые метрики машинного обучения. Все значения представлены в таблице 4.

Таблица 4 – Метрики оценки модели

Метрика	Значение
Accuracy	79.58%
Precision	71.26%
Recall	81.58%
F1-score	76.07%
ROC-AUC	85.50%
FPR	21.74%
FNR	18.42%

Значение метрики Precision означает, что из всех случаев, где модель предсказала наличие причинно-следственной связи, более 70% оказались правильными. Оставшиеся примеры являются ложноположительными, то есть модель ошибочно определила связь там, где её нет.

Значение метрики Recall означает, что модель смогла обнаружить более 80% истинных случаев наличия причинности, что говорит о её чувствительности к положительному классу.

Значение метрики F1-score показывает гармоническое среднее между Precision и Recall. Значение выше 75% говорит о достаточно сбалансированной работе модели.

Значение метрики ROC-AUC, график которой представлен на рисунке 6, указывает на очень хорошую определяющую способность модели. Значение более 85% говорит о том, что модель уверенно различает положительные и отрицательные пары предложений.

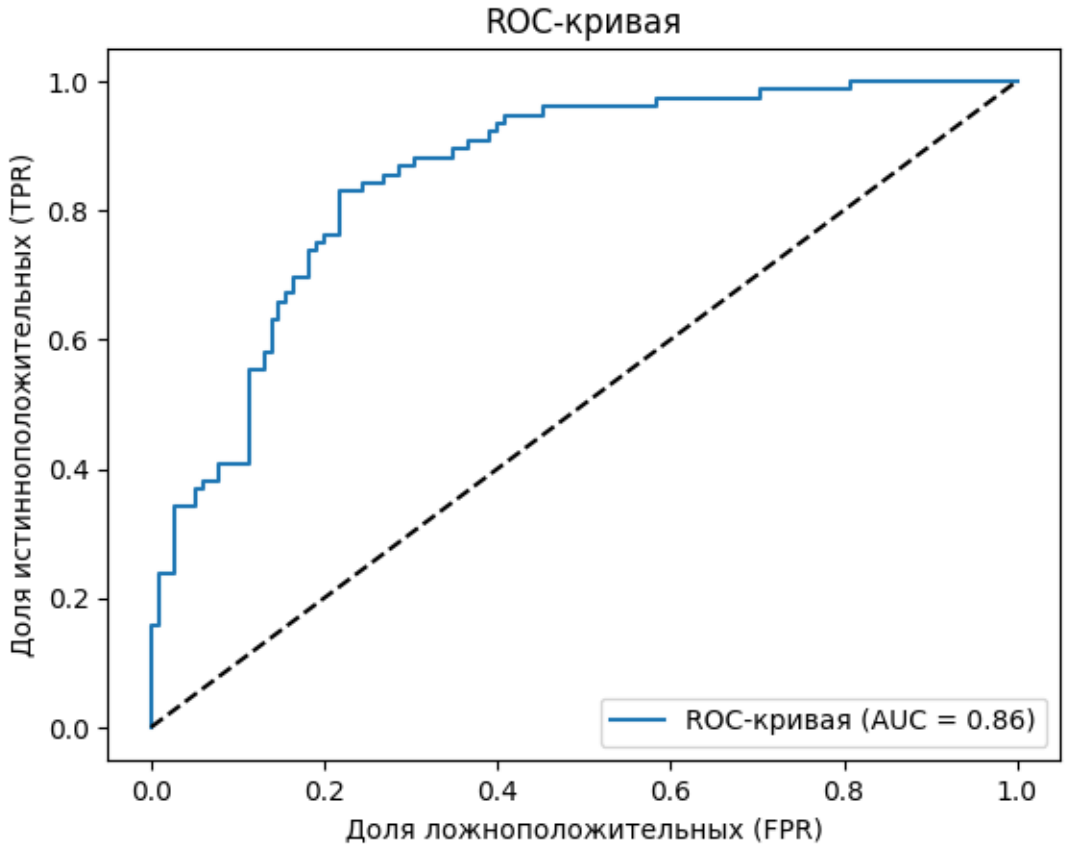

Рисунок 6 – График ROC-кривой

ROC-кривая была построена на основе значений TPR и FPR при различных порогах вероятности. Полученная площадь под кривой свидетельствует о том, что модель работает не как случайный классификатор.

Значения TPR и FPR исходят из матрицы ошибок, представленной на рисунке 7. Матрица ошибок позволяет наглядно оценить, как модель справляется с каждым из двух классов.

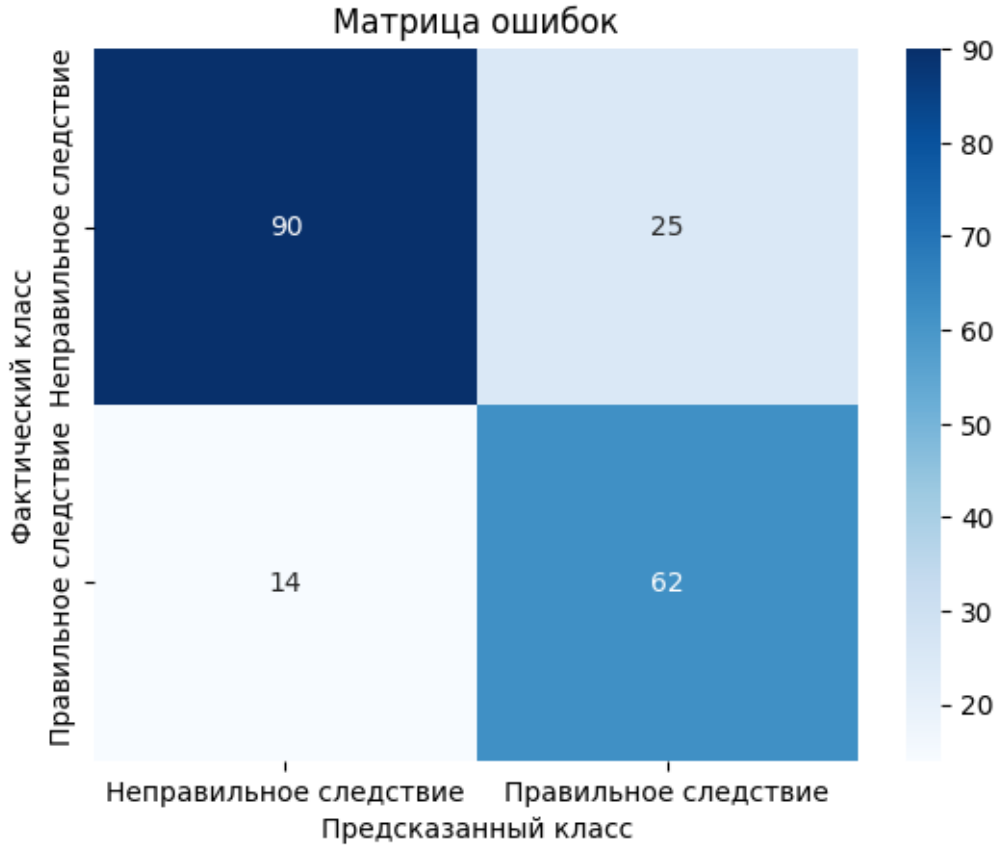

Рисунок 7 – Матрица ошибок

По матрице ошибок можно сделать вывод, что из 191-го случая модель правильно определят 152 случая. Модель чаще допускает ошибки типа ложноотрицательные, чем ложноположительные, что связано с тем, что она склонна быть более чувствительной к истинным положительным примерам из-за несбалансированности классов.

Для удобства анализа качества работы модели, были выведены примеры ошибочных предсказаний, представленные на рисунке 8.

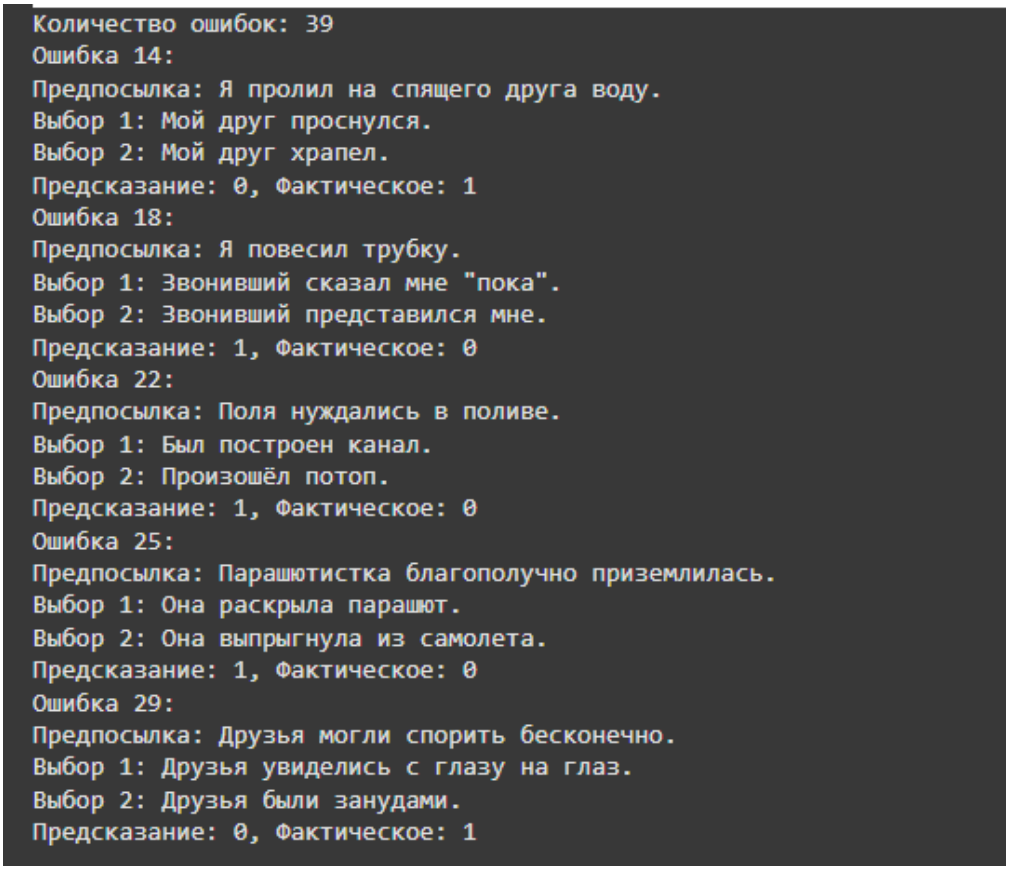

Рисунок 8 – Ошибочные предсказания

На основе анализа ошибок, выявленных на валидационной выборке, можно сделать следующие выводы:

Основной проблемой ошибочности модели является несовершенство набора данных. Начиная тем, что набор несбалансирован, и заканчивая тем, что метки в некоторых фразах определены неправильно. Также встречаются варианты, в которых оба следствия подходят по смыслу к предпосылке, что отражает некачественность датасета [20].

Проблемы из-за несбалансированности классов удалось отчасти сгладить с помощью SMOTE и class_weight. С помощью этих инструментов было искусственно увеличено количество образцов из миноритарного класса, и увеличено влияние редкого класса через взвешивание функции потерь [20].

Построение и обучение модели происходило в среде Google Colab на базе GPU ресурсов, что обеспечивало высокую скорость обучения.

ЗАКЛЮЧЕНИЕ

В наше время задачи обработки естественного языка становятся всё более актуальными, и постоянно притягивают к себе всё больше новых областей: от рекламы до медицины и дальше. Одной из сложнейших и недостаточно исследованных задач в этой области является выявление причинно-следственных связей между предложениями в тексте, что требует не только семантического понимания текста, но и способности моделировать логические зависимости. Классические методы обработки естественного языка, основанные на словарях и статистическом подходе, со временем становятся всё менее эффективными, и поэтому современные методы машинного обучения, включая глубокие нейронные сети, становятся более продуктивными в этой области, так как они позволяют автоматизировать процесс анализа текстов и значительно повысить точность классификации.

В ходе написания выпускной квалификационной работы были выполнены все поставленные задачи и достигнута поставленная цель. Был проведен литературный обзор и анализ существующих методов и средств решения задачи бинарной классификации. Был произведен обзор моделей нейронных, а также обзор сред разработки и языков программирования для решения поставленной задачи. Были сформулированы концептуальная и математическая постановки задачи. Была построена и обучена нейронная сеть посредством языка программирования Python в среде Google Colab. Были оценены результаты работы нейронной сети и определены причины ошибок в работе.

Построенная нейронная сеть показала высокие показатели точности работы, модель способна различать случаи наличия и отсутствия причинно-следственных связей между предложениями.

По результатам работы опубликованы тезисы докладов

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Ананьева, А.А. Особенности библиотек программного комплекса Keras для создания искусственных нейронных сетей / А.А. Ананьева, В.Н. Ланцов // Прикладная математика и информатика: современные исследования в области естественных и технических наук: Сборник материалов IX Международной научно-практической конференции (школы-семинара) молодых ученых, Тольятти, 18 – 20 апреля 2023 года. – Тольятти: Тольяттинский государственный университет, 2023. – С. 386-390. – Режим доступа: https://www.elibrary.ru/item.asp?id=54932566.

2 Гапоненко, В.А. Построение нейросетевой модели анализа тональности текста / В.А. Гапоненко, И.В. Николаева // Информационное общество: современное состояние и перспективы развития: Сборник материалов XVII международного форума, Краснодар, 15 – 20 июля 2024 года. – Краснодар: Кубанский государственный аграрный университет им. И.Т. Трубилина, 2024. – С. 58-61. – Режим доступа: https://www.elibrary.ru/item.asp?id=75132088.

3 Заварукин, А.С. Применение искусственного интеллекта для анализа медицинских документов / А.С. Заварукин // Ceteris Paribus. – 2022. – № 5. – С. 56-60. – Режим доступа: https://www.elibrary.ru/item.asp?id=48491577.

4 Канаев, М.М. Некоторые вопросы обучение многослойных нейронных сетей / М.М. Канаев // Современные технологии: актуальные вопросы, достижения и инновации: сборник научных трудов. – Махачкала: Типография ФОРМАТ, 2022. – С. 24-27. – Режим доступа: https://www.elibrary.ru/item.asp?id=53707245.

5 Коржова, М.Е. Бинарная классификация фактографических данных / М.Е. Коржова // Будущее науки - 2024: сборник научных статей 11-й Международной молодежной научной конференции, Курск, 18 – 19 апреля 2024 года. – Курск: ЗАО «Университетская книга», 2024. – С. 79-83. – Режим доступа: https://www.elibrary.ru/item.asp?id=66075738.

6 Кошелев, Н.Д. Работа с нейронными сетями в Python и Julia / Н.Д. Кошелев // Труды международного симпозиума "Надежность и качество". – 2024. – Т. 1. – С. 342-343. – Режим доступа: https://www.elibrary.ru/item.asp?id=67990639.

7 Кузнецов, А.А. Классификация data science и способы применения / А.А. Кузнецов, А.С. Толоконцева // Теория и практика современной науки. – 2020. – № 9(63). – С. 181-183. – Режим доступа: https://www.elibrary.ru/item.asp?id=44128186.

8 Минязев, Р.Ш. Подходы к построению нейросети для бинарной классификации рентгенограмм / Р.Ш. Минязев, А.А. Румянцев, А.А. Баев, Т.Д. Баева // Известия Российской академии наук. Серия физическая. – 2020. – Т. 84, № 12. – С. 1758-1762. – Режим доступа: https://www.elibrary.ru/item.asp?id=44154377.

9 Муренко, Ю.М. Эффективное использование рекуррентных нейронных сетей в обработке текстовых данных / Ю.М. Муренко // Современная философия и социальные науки глазами молодых ученых: Сборник статей и эссе. – Москва: Московский городской педагогический университет, 2024. – С. 155-160. – Режим доступа: https://www.elibrary.ru/item.asp?id=59904185.

10 Назаренко, П.А. Применение библиотеки Keras в курсе «Нейросетевые алгоритмы обработки данных» / П.А. Назаренко // IX Российская научно-методическая конференция профессорско-преподавательского состава, научных сотрудников и аспирантов: МАТЕРИАЛЫ КОНФЕРЕНЦИИ, Самара, 05 – 08 апреля 2021 года. – Самара: Поволжский государственный университет телекоммуникаций и информатики, 2021. – С. 104. – Режим доступа: https://www.elibrary.ru/item.asp?id=45847406.

11 Николаев, П.Л. Классификация книг по жанрам на основе текстовых описаний посредством глубокого обучения / П.Л. Николаев // International Journal of Open Information Technologies. – 2022. – Т. 10, № 1. – С. 36-40. – Режим доступа: https://www.elibrary.ru/item.asp?id=47560526.

12 Павлюченко, М.В. Анализ зависимости точности бинарной классификации текстов от применения мета-функций для различных алгоритмов классификации / М.В. Павлюченко, Т.В. Кабанова // Математическое и программное обеспечение информационных, технических и экономических систем : Материалы VIII Международной молодежной научной конференции, Томск, 26 – 30 мая 2021 года / Под общей редакцией И.С. Шмырина. Том 306. – Томск: Национальный исследовательский Томский государственный университет, 2021. – С. 42-47. – Режим доступа: https://www.elibrary.ru/item.asp?id=47402352.

13 Протодьяконов, А.В. Алгоритмы Data Science и их практическая реализация на Python: Учебное пособие / А.В. Протодьяконов, П.А. Пылов, В.Е. Садовников. – Москва: Общество с ограниченной ответственностью "Издательство "Инфра-Инженерия", 2022. – 392 с. – Режим доступа: https://www.elibrary.ru/item.asp?id=65484681.

14 Соков, И.А. Исследование обучения нейронной сети в реальном времени на Python / И.А. Соков // Образование. Наука. Производство: Сборник докладов XV Международного молодежного форума, Белгород, 23 – 24 октября 2023 года. – Белгород: Белгородский государственный технологический университет им. В.Г. Шухова, 2023. – С. 341-344. – Режим доступа: https://www.elibrary.ru/item.asp?id=59457488.

15 Терлецкий, А.С. Нейронные сети и искусственный интеллект: Основы нейронных сетей на языке Python / А.С. Терлецкий, Е.С. Терлецкая. – Липецк: Липецкого государственного педагогического университета имени П.П. Семенова-Тян-Шанского, 2023. – 76 с. – ISBN 978-5-907792-40-1. – Режим доступа: https://e.lanbook.com/book/439343.

16 Трифонов, К.В. Сравнение свёрточной и рекуррентной архитектур нейронных сетей при решении задачи анализа тональности текста / К.В. Трифонов // Молодой исследователь Дона. – 2024. – Т. 9, № 2(47). – С. 41-44. – Режим доступа: https://www.elibrary.ru/item.asp?id=65524235.

17 Чотчаев, Б.А. Из опыта изучения реализации нейронных сетей посредством Python / Б.А. Чотчаев // Мир студенческой науки: сборник статей II Международного научно-исследовательского конкурса, Пенза, 10 февраля 2024 года. – Пенза: Наука и Просвещение (ИП Гуляев Г.Ю.), 2024. – С. 7-9. – Режим доступа: https://www.elibrary.ru/item.asp?id=60101689.

18 Юмашева, Ю.Ю. Архивы в информационную эпоху и время Data Science / Ю.Ю. Юмашева // Историко-культурное наследие в институциональном измерении: Материалы II Уральского историко-архивного форума, посвященного 30-летию подготовки документоведов и 20-летию подготовки специалистов в сфере туризма и гостеприимства в Уральском федеральном университете, Екатеринбург, 06-09 октября 2022 года / Отв. редактор Л.Н. Мазур. – Екатеринбург: Уральский федеральный университет имени первого Президента России Б.Н. Ельцина, 2022. – С. 347-351. – Режим доступа: https://www.elibrary.ru/item.asp?id=49953697.

19 Metatext.io [Электронный ресурс]: – Режим доступа: https://metatext.io/datasets/choice-of-plausible-alternatives-for-russian-language-(parus)-(superglue).

Список публикаций автора по теме работы

20 Тузов, Д.В. Нейронная сеть для определения логических причинно-следственных связей на основе корпуса данных естественного языка / Д.В. Четыркин // Молодёжь XXI века: шаг в будущее: материалы XXVI региональной научно-практической конференции (Благовещенск, 16 мая 2025 г.). – Благовещенск: Благовещенский гос. пед. университет, 2025. – *в печати*.

ПРИЛОЖЕНИЕ А

Листинг программы построенной нейронной сети

```
# Установка необходимых библиотек
!pip install torch==2.0.1 torchvision==0.15.2 torchaudio==2.0.2 --index-url https://download.pytorch.org/whl/cu118
!pip install transformers tensorflow scikit-learn seaborn jsonlines sentence-transformers

# Импорт библиотек
import jsonlines
import numpy as np
import tensorflow as tf
from sentence_transformers import SentenceTransformer  # Исправленный импорт
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Dropout, BatchNormalization, Lambda, Subtract, Concatenate
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc, precision_recall_curve, \
    accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.utils import class_weight
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
import seaborn as sns

# Анализ версий библиотек
print("Версии библиотек:")
print("TensorFlow:", tf.__version__)
!pip show transformers | grep Version
print("\n")

# Проверка доступности GPU
print("GPU доступен:", tf.config.list_physical_devices('GPU'))
print("\n")

# Функция для загрузки данных
def load_data(file_path, is_test=False):
    premises = []
    choices1 = []
    choices2 = []
```

```
    labels = []
    with jsonlines.open(file_path) as reader:
        for obj in reader:
            if all(key in obj for key in ['premise', 'choice1', 'choice2']):
                premises.append(obj['premise'])
                choices1.append(obj['choice1'])
                choices2.append(obj['choice2'])
                if not is_test:
                    labels.append(obj['label'])
                else:
                    labels.append(0)  # Заглушка, если меток нет
            else:
                print(f"Пропущен объект из-за отсутствия ключей: {obj}")
    return premises, choices1, choices2, labels

# Загрузка данных
train_data = load_data('drive/MyDrive/VK Coffee/PARus1/train.jsonl')
val_data = load_data('drive/MyDrive/VK Coffee/PARus1/val.jsonl')
test_data = load_data('drive/MyDrive/VK Coffee/PARus1/test.jsonl', is_test=True)

# Объединение данных
premises = train_data[0] + val_data[0]
choices1 = train_data[1] + val_data[1]
choices2 = train_data[2] + val_data[2]
labels = train_data[3] + val_data[3]

# Разделение данных на обучающую и тестовую выборки
X_premises_train, X_premises_val, y_train, y_val = train_test_split(premises, labels,
test_size=0.2, random_state=42)
X_choice1_train, X_choice1_val = train_test_split(choices1, test_size=0.2, ran-
dom_state=42)
X_choice2_train, X_choice2_val = train_test_split(choices2, test_size=0.2, ran-
dom_state=42)

# Проверка распределения классов
print("Распределение классов в обучающей выборке:", np.bincount(y_train))
print("Распределение классов в валидационной выборке:", np.bincount(y_val))

# Вычисление весов классов для балансировки
class_weights = class_weight.compute_class_weight('balanced', clas-
ses=np.unique(y_train), y=y_train)
class_weights = dict(enumerate(class_weights))
print("Веса классов:", class_weights)

# Загрузка предобученного Sentence-BERT
```

```
sentence_model = SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2')

# Преобразование текстовых данных в числовые эмбеддинги
def get_sentence_embeddings(texts):
    return sentence_model.encode(texts, convert_to_tensor=False)

# Получение эмbeddингов для обучающих данных
X_premises_train_embeddings = get_sentence_embeddings(X_premises_train)
X_choice1_train_embeddings = get_sentence_embeddings(X_choice1_train)
X_choice2_train_embeddings = get_sentence_embeddings(X_choice2_train)

# Применение SMOTE к числовым эмbeddингам
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(
    np.hstack([X_premises_train_embeddings,          X_choice1_train_embeddings,
X_choice2_train_embeddings]),
    y_train
)

# Разделение обратно на отдельные части
X_premises_train_resampled = X_train_resampled[:, :384]  # Размерность Sentence-
BERT: 384
X_choice1_train_resampled = X_train_resampled[:, 384:768]
X_choice2_train_resampled = X_train_resampled[:, 768:]

# Получение эмbeddингов для валидационных данных
X_premises_val_embeddings = get_sentence_embeddings(X_premises_val)
X_choice1_val_embeddings = get_sentence_embeddings(X_choice1_val)
X_choice2_val_embeddings = get_sentence_embeddings(X_choice2_val)

# Создание Siamese Network
input_dim = 384  # Размерность эмbeddингов Sentence-BERT

# Входные слои
input_premise = Input(shape=(input_dim,), name='premise_input')
input_choice1 = Input(shape=(input_dim,), name='choice1_input')
input_choice2 = Input(shape=(input_dim,), name='choice2_input')

# Сравнение эмbeddингов
diff1 = Subtract()([input_premise, input_choice1])
diff2 = Subtract()([input_premise, input_choice2])

# Конкатенация разностей
concatenated = Concatenate()([diff1, diff2])
```

```
# Полносвязные слои
x = Dense(256, activation='relu')(concatenated)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)

x = Dense(128, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)

output_layer = Dense(1, activation='sigmoid')(x)

# Сборка модели
model = Model(inputs=[input_premise, input_choice1, input_choice2], outputs=out-
put_layer)

# Компиляция модели
optimizer = Adam(learning_rate=1e-4)
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accura-
cy'])

# Ранняя остановка
early_stopping = EarlyStopping(monitor='val_loss', patience=10, re-
store_best_weights=True)

# Обучение модели
history = model.fit(
   [X_premises_train_resampled, X_choice1_train_resampled,
X_choice2_train_resampled],
   np.array(y_train_resampled),
   epochs=50,
   batch_size=64,
   validation_data=(
      [X_premises_val_embeddings, X_choice1_val_embeddings,
X_choice2_val_embeddings],
      np.array(y_val)
   ),
   callbacks=[early_stopping],
   class_weight=class_weights
)

# Визуализация процесса обучения
plt.plot(history.history['accuracy'], label='Точность на обучении')
plt.plot(history.history['val_accuracy'], label='Точность на валидации')
plt.title('Точность модели')
plt.xlabel('Эпохи')
```

```
plt.ylabel('Точность')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Потери на обучении')
plt.plot(history.history['val_loss'], label='Потери на валидации')
plt.title('Потери модели')
plt.xlabel('Эпохи')
plt.ylabel('Потери')
plt.legend()
plt.show()

# Оценка модели
y_pred = model.predict(
    [X_premises_val_embeddings, X_choice1_val_embeddings, X_choice2_val_em-
beddings]
)

# Оптимальный порог вероятности
fpr, tpr, thresholds = roc_curve(y_val, y_pred)
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
print(f"Оптимальный порог вероятности: {optimal_threshold:.2f}")

y_pred_classes = (y_pred > optimal_threshold).astype(int).flatten()

# Вычисление метрик
accuracy = accuracy_score(y_val, y_pred_classes)
precision = precision_score(y_val, y_pred_classes)
recall = recall_score(y_val, y_pred_classes)
f1 = f1_score(y_val, y_pred_classes)
roc_auc = roc_auc_score(y_val, y_pred)

# Вывод численных результатов
print("\nЧисленные результаты:")
print(f"Доля верных ответов (Accuracy): {accuracy * 100:.2f}%")
print(f"Точность (Precision): {precision * 100:.2f}%")
print(f"Полнота (Recall): {recall * 100:.2f}%")
print(f"F1-score: {f1 * 100:.2f}%")
print(f"AUC-ROC: {roc_auc * 100:.2f}%")

# Процент ошибок
cm = confusion_matrix(y_val, y_pred_classes)
TN, FP, FN, TP = cm.ravel()
false_positive_rate = FP / (FP + TN) if (FP + TN) > 0 else 0
```

```
false_negative_rate = FN / (FN + TP) if (FN + TP) > 0 else 0

print(f"Процент ложноположительных ошибок (False Positive Rate): {false_positive_rate * 100:.2f}%")
print(f"Процент ложноотрицательных ошибок (False Negative Rate): {false_negative_rate * 100:.2f}%")

# ВИЗУАЛИЗАЦИЯ: Матрица ошибок
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Неправильное следствие', 'Правильное следствие'],
            yticklabels=['Неправильное следствие', 'Правильное следствие'])
plt.title('Матрица ошибок')
plt.xlabel('Предсказанный класс')
plt.ylabel('Фактический класс')
plt.show()

# ВИЗУАЛИЗАЦИЯ: ROC-кривая и AUC
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f'ROC-кривая (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')  # Случайный классификатор
plt.title('ROC-кривая')
plt.xlabel('Доля ложноположительных (FPR)')
plt.ylabel('Доля истинноположительных (TPR)')
plt.legend()
plt.show()

# Анализ ошибок
incorrect_indices = np.where(y_pred_classes != y_val)[0]
print(f"Количество ошибок: {len(incorrect_indices)}")
for i in incorrect_indices[:10]:  # Выведем первые 10 ошибок
    print(f"Ошибка {i}:")
    print(f"Предпосылка: {premises[i]}")
    print(f"Выбор 1: {choices1[i]}")
    print(f"Выбор 2: {choices2[i]}")
    print(f"Предсказание: {y_pred_classes[i]}, Фактическое: {y_val[i]}")
```