ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННО БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
УНИВЕРСИТЕТ «ДУБНА»

Институт системного анализа и управления

Кафедра системного анализа и управления

КУРСОВАЯ РАБОТА

По курсу

«Теория принятия решений»

Тема: «Принятие решений в задачах машинного обучения: нейросетевой подход к выявлению сгенерированного контента»

Выполнил: студент гр. 2011

Трусов Иван Александрович

_____

(подпись студента)

Проверил: к.т.н., доцент Крейдер О.А.

Дата защиты:_____

Оценка:_____

(подпись руководителя)

Дубна 2025

Оглавление

Введение

В современном мире появление и развитие нейросетевых генеративных моделей, стало очень распространённым. Создаваемого объёма контента значительно увеличилось. Люди чаще стали использовать их для написания своих работ или для получения источника информации. В связи с этим появилась проблема различать текст, написанный человеком или сгенерированным нейросетью.

С развитием генеративных моделей вопрос автоматического определения сгенерированного контента встаёт всё более остро. Современные большие языковые модели (*LLM*) — такие, как *ChatGPT*, — способны порождать тексты, на первый взгляд, не отличимые от текстов, написанных людьми. При ближайшем рассмотрении в них, конечно же, встречаются логические ошибки, подмена фактов, синтаксическая структура, несвойственная текстам, написанным людьми, а также весьма необычный выбор лексических средств. Однако автоматическое определение «искусственности» текста или хотя бы чёткая формализация задачи всё ещё вызывает существенные трудности. Актуальность данной задачи, с одной стороны, заключается в необходимости определять сгенерированный контент для борьбы с неправомерным использованием крупных языковых моделей — созданием фейк-ньюс или разведением ботов, — а с другой, для получения устойчивого критерия, предназначенного для оценки качества генерации и последующего улучшения языковых моделей.

В данной курсовой работе будет рассмотрена возможность с помощью машинного обучения классифицировать текст на сгенерированность и несгенерированность.

Постановка задачи

Цель

Повышение качества распознавание контента созданными генеративными нейросетями.

Исходные данные

Датасет созданный с помощью генеративных нейронных сетей, таких как *ChatGPT-4, GIGAChat, Le Chat, Perplexity, DeepSeek*. Датасет был разбит на 2 категории *POSITIVE* (сгенерированный текст) и *NEGATIVE* (текст, написанный человеком).

Модельное представление

Модель будет основана на *spaCy, TextCategorizer* поможет классифицировать текст на категории сгенерированный и несгенерированный. Модель можно будет использовать на любых устройствах.

Ожидаемый результат

Обученная модель нейронной сети, классифицирующая текст на сгенерированность и несгенерированность.

Оценка результата

Классификация текста на сгенерированность и несгенерированность, точность нейронной сети не должна быть ниже 80%, возможность интегрирование нейронной сети на веб-сервисы.

Теоретическая часть

Любая сфера человеческой деятельности связана с принятием решений.

Решение — это не только выбор одной или нескольких альтернатив, но и процесс осуществления такого выбора.

В узком смысле принятие решений — это заключительный акт деятельности по выявлению, анализу различных вариантов решения, направленный на выбор и утверждение лучшего варианта решения. В узком плане решение можно трактовать как результат выбора, тогда оно представляет собой предписание к действию (план работы, вариант проекта).

В широком смысле принятие решения — это процесс, протекающий во времени, осуществляемый в несколько этапов. Другими словами, это совокупность всех этапов и стадий по подготовке (разработке) решения, включая заключительный этап непосредственного принятия решения [5].

История принятия решений идёт ещё с 5000 года до нашей эры. Основные этапы развития принятия решений приведены в таблице (см. табл. 1).

Таблица 1. Этапы развития принятия решений

Год	Индивидуум или этническая группа	Основной вклад в развитие
до н\э		
5000	Шумеры	Письменность, регистрация фактов
4000	Египтяне	Признание необходимости планирования, организации и контроля
2600	Египтяне	Централизация в организации управления
1800	Хаммурапи	Использование свидетелей и письменных документов для контроля, установление минимальной заработной платы, признание допустимости перекладывания ответственности
1491	Евреи	Концепция организации, скалярный принцип
600	Навуходоносар	Контроль за производством и стимулирование через заработную плату
400	Сократ	Формулировка принципа универсальности менеджмента
400	Ксенофонт	Признание менеджмента как особого вида искусства
175	Калон	Использование описаний работ
Н\э		
20	Иусус Христос	Единоличные Золотое правило Человеческие отношения
1100	Газали	Требования к менеджеру
1835	Маршалл, Логган	Признание и обсуждение относительной важности менеджмента
1881	Джозеф Вартон	Разработка для колледжа курса предпринимательского менеджмента
1900	Фредерик У Тейлор	Научный менеджмент, системный подход, кадровый менеджмент, необходимость координации операций между трудом и менеджментом, функциональная организация, оценка функций

1916	Гери Файоль	Первая полная теория менеджмента, его функции, принципы, признание необходимости преподавания менеджмента
1919	Моррис Л Кук	Различные области использования менеджмента
1927	Элтон Мэйо	Социологическая концепция групповых устремлений
1943	Линделл Урвик	Сведение воедино и корреляция принципов менеджмента
1949	Норберт Винер	Разработка системного анализа в теории информации
1976	Ромари Стюарт	Альтернатива и ограничение действий менеджера в различных ситуациях
1985	Том Питерс	Отношение к потребителям как к людям, а к персоналу организации как к важному ресурсу развития бизнеса

Данная таблица показывает наиболее существенные этапы развития управленческой мысли. И если, по всей видимости, вехи, относящиеся к глубокой древности, не подвергаются сегодня сомнению, то акценты и интерпретация вклада в развитие управленческой мысли, более близкие к нам по времени, могут быть расставлены и по-другому.

При этом мы не можем отрицать явного всплеска управленческой мысли, приходящегося именно на наше время.

Поскольку каждое решение — это проекция в будущее, а будущее всегда содержит элемент неопределенности, поэтому для руководителя (лица, принимающего решения) важно правильно определить степень риска, с которым сопряжена реализация принятых решений [6].

Словосочетание «принятие решений» в настоящее время используется очень широко. Говорят, что наилучший вариант решения может быть получен путем математических расчетов, и есть случаи, когда это возможно. Говорят о компьютерах или роботах, принимающих решения, и это тоже имеет место [7].

С развитием технологий и увеличением объема данных традиционные методы принятия решений зачастую оказываются недостаточно эффективными для решения сложных задач. В таких условиях на помощь приходят современные методы искусственного интеллекта, в частности машинное обучение.

Искусственный интеллект (ИИ) — это имитация процессов человеческого интеллекта машинами, компьютерными системами [4].

Области искусственного интеллекта:

1. Компьютерное зрение;
2. Обработка естественного языка;
3. Распознавание и синтез речи;
4. Системы поддержки принятия решений;
5. Машинное обучение.

Машинное обучение включает первый этап обучения или тренировки, когда машина постепенно «учится» выполнять задачу, и второй этап — реализацию, когда машина закончила обучение.

Для того, чтобы научить машину определять, содержит ли изображение автомобиль или самолет, мы должны начать с предоставления ей тысяч изображений, содержащих самолет или автомобиль. Каждый раз, когда на входе системе дается изображение, нейронная сеть (или «нейросеть»), состоящая из соединенных между собой искусственных нейронов (в действительности — это множество математических функций, вычисляемых компьютером), обрабатывает это изображение и выдает выходной ответ:

1. Если ответ правильный, мы ничего не делаем и переходим к следующему изображению.
2. Если ответ неверный, мы немного корректируем внутренние параметры машины, то есть силу связей между нейронами, чтобы ее выходной сигнал приближался к желаемому ответу.

Со временем система настраивается и в конечном итоге сможет распознать любой объект, будь то изображение, которое она видела ранее, или любое другое.

Искусственная нейронная сеть — это программируемая, математическая модель построенная по принципу работы биологических нейронных сетей человека [4].

Тогда, машинное обучение — это, по сути, процесс оптимизации параметров искусственной нейронной сети, направленный на минимизацию ошибок на обучающих данных путем итеративного сравнения ответов системы с эталонными и корректировки внутренних связей. Это позволяет создавать модели, которые учатся на опыте и улучшают свою работу со временем.

Само понятие машинное обучение появилось недавно, в 1950-х годах, когда учёные на основах логики и алгоритмов обходов графов, искусственного интеллекта, раздвинули границы применения. В 1957 году Фрэнк Розенблатт, вдохновлённый когнитивной теорией Дональда Хебба, построил перцептрон — первую обучающуюся машину. Эта машина стала эталонной моделью машинного обучения (см. рис. 1).

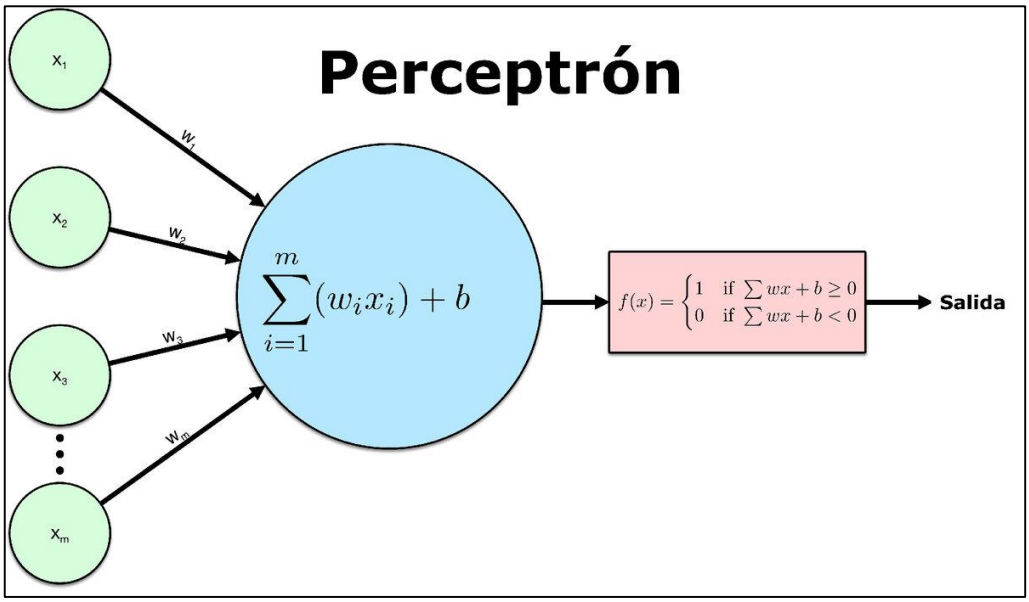

Рисунок 1. Архитектура перцептрона

Архитектура перцептрона выглядит следующим образом:

Первый слой — это сенсорные элементы, которые задают, что мы имеем на входе.

Второй слой — ассоциативные элементы. Их связи с сенсорным слоем жестко заданы и определяют переход к более общей, чем в сенсорном слое, ассоциативной картине описания.

Третий слой — регулирующие элементы, определяющие выход перцептрона в целом. Элемент данного слоя складывает друг с другом взвешенные сигналы от ассоциативных элементов и, если превышен определённый порог, генерирует выходной сигнал равный 1, иначе выход с перцептрона равен -1 [10].

Существуют виды машинного обучения, которые делятся на два вида: с учителем и без учителя.

При машинном обучении с учителем тренировочные данные обязательно должны быть как-то маркированы. Именно машинное обучение с учителем большинство людей подразумевают под машинным обучением. Модель с учителем может измениться в лучшую сторону только путем измерения различий между ожидаемыми выходными данными (метками) и полученными предсказаниями.

При машинном обучении без учителя модель обучается выполнять задачу без каких-либо меток на основе одних только исходных данных. Также методиками машинного обучения без учителя являются алгоритмы понижения размерности. В данном виде машинного обучения модель ищет закономерности в связях между самими точками данных. Сделать модель без учителя «умнее» (точнее) можно, предоставив ей больше данных.

Машинное обучение без учителя позволяет машине обучаться непосредственно на данных без какой-либо помощи со стороны человека. Тренировочные данные могут быть не упорядочены, не структурированы и никак не маркированы.

В данной работе будет рассмотрено принятие решения в задаче машинного обучения с учителем в области обработки естественного языка

Обработка естественного языка (*NLP*) — это область исследований в компьютерных науках и искусственном интеллекте (ИИ), занимающаяся работой с такими языками, как английский или китайский. Обработка обычно включает в себя перевод естественного языка в данные (числа), с помощью которых компьютер может получить информацию об окружающем мире. И это понимание мира иногда используется для создания отражающего его текста на естественном языке [1].

Примерно в 2013 году обработка естественного языка и чат-боты начали получать все большее распространение в нашей жизни. Сначала поиск *Google* напоминал лишь работу с предметным указателем — инструментом, для использования которого не требовалось особых навыков. Однако скоро он стал интеллектуальнее и начал понимать поисковые запросы, все более близкие к естественному языку. Далее еще больше усложнялась функциональность автодополнения в смартфонах. По центру зачастую было указано именно интересующее пользователя слово.

Именно с 2013 года, начал появляться сгенерированный контент

Сгенерированный контент — это материалы, созданные автоматически с помощью искусственного интеллекта, в частности нейросетевых моделей, а не напрямую человеком.

Такой контент часто характеризуется определёнными свойствами:

1. Суммаризацией;
2. Симплификацией;
3. Ответ, привязанный к вопросам.

Суммаризация текста — это процесс создания краткой версии исходного материала с сохранением его ключевых идей и смысловой нагрузки. По сути, это искусство сжатия информации без потери значимого содержания. Например, представьте, что вы сжимаете огромный текстовый файл, но вместо случайной потери данных вы осознанно оставляете только самое важное. Именно поэтому, суммаризация помогает экономить время, делая при этом информацию более удобной для восприятия [11].

Преобразование текста с целью удаления языковых конструкций, осложняющих его восприятие, называется симплификацией, или упрощением текста. Симплификация

текста в автоматизированном режиме позволяет реализовать правила упрощения текстов с помощью инструментов обработки стандартного языка. С использованием механизма автоматизированной симплификации текста может быть обеспечен равный доступ к текстовой информации для различных категорий населения [12].

Такое свойство сгенерированного контента, как ответ, привязанный к вопросам, характеризуется тем, что формирует релевантный и тематически связанный ответ.

В данной работе будет рассматриваться подход к выявлению генерированного с помощью библиотеки *spaCy* в языке программирования *Python*.

SpaCy — это бесплатная библиотека с открытым исходным кодом для расширенной обработки естественного языка (*NLP*) в *Python* [2].

Рассмотрим, как устроена архитектура *spaCy* (см. рис. 2)

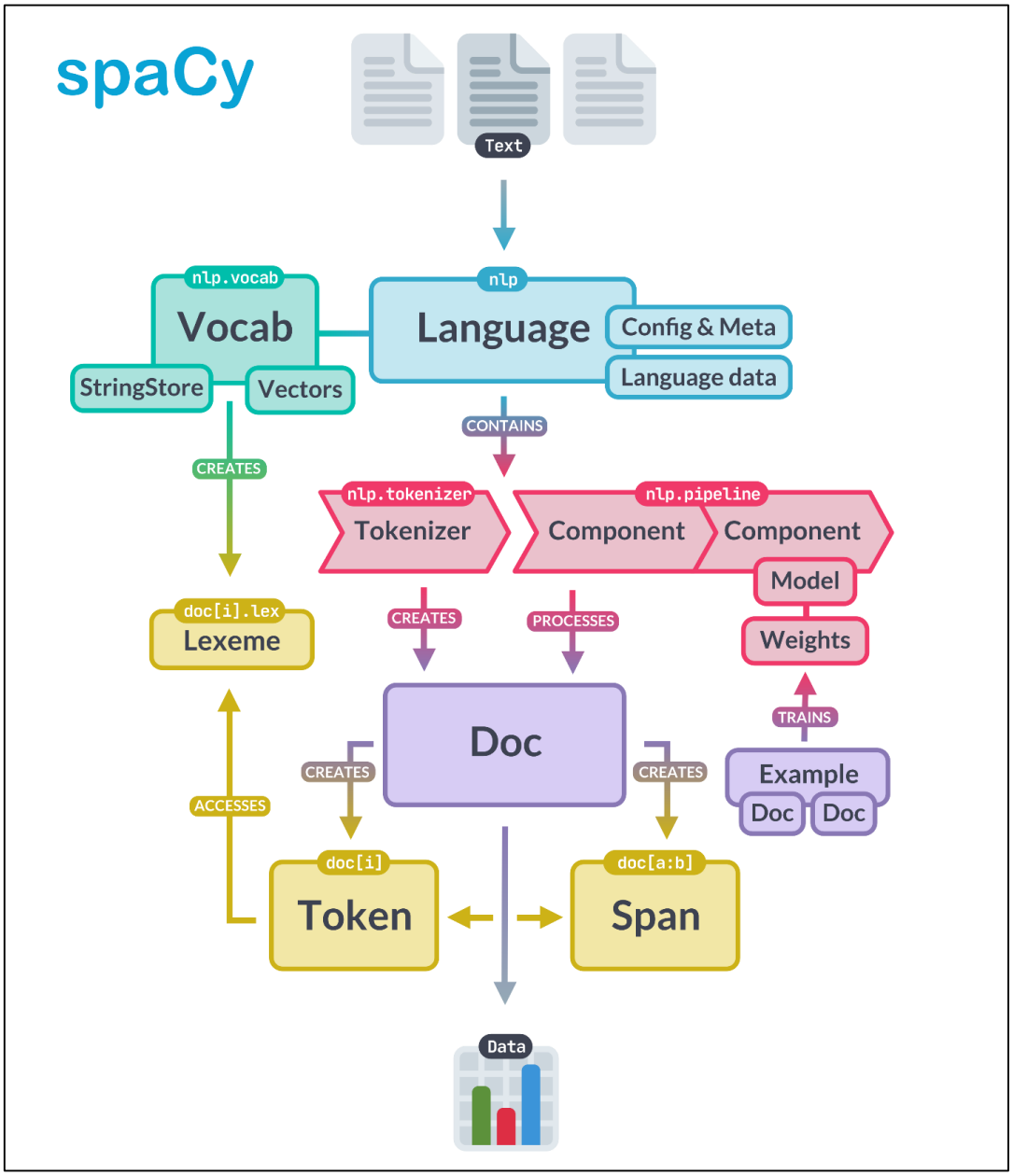

Рисунок 2. Архитектура *spaCy*

Подробнее об архитектуре:

Центральными структурами данных в *spaCy* являются следующие *Language* класс тем *Vocab* и *Doc* объект. Класс используется для обработки текста и превращения его в объект. Обычно он хранится в виде переменной с именем. Объекту принадлежит последовательность лексем и все их аннотации. За счет централизации строк, векторов слов, мы избегаем хранения нескольких копий этих данных. Это экономит память и обеспечивает наличие единого источника достоверной информации [2].

Текстовые аннотации также предназначены для обеспечения единого источника достоверной информации: объект владеет данными, а *DocSpan* и *Token* это взгляды, которые указывают на него. Объект конструируется с помощью метода *DocTokenizer*, а затем модифицируются компонентами трубопровода. Объект координирует эти компоненты. Для этого нужно Исходный текст и отправляет его по конвейеру, возвращая документ с аннотацией. Он также координирует обучение и сериализацию.

- *Doc* — контейнер для доступа к лингвистическим аннотациям;
- *DocBin* — набор объектов для эффективной двоичной сериализации; Также используется для обучающих данных;
- *Example* — коллекция обучающих аннотаций, содержащая два объекта: справочные данные и прогнозы;
- *Language* — класс обработки, который превращает текст в объекты; В разных языках реализуются свои подклассы; Переменная обычно называется *nlp*;
- *Lexeme* — запись в словарном запасе; Это тип слова без контекста, в отличие от лексемы слова; Поэтому у него нет тега части речи, разбора зависимостей и т;д;
- *Span* — кусочек от объекта *Doc*;
- *SpanGroup* — именованная коллекция пролетов, принадлежащая *Doc*;
- *Token* — отдельный токен, то есть слово, знак препинания, пробел и т. д.

Конвейер обработки состоит из одного или нескольких компонентов конвейера, которые призвали по порядку. Генератор маркеров выполняется перед компонентами. Компоненты могут быть добавлены с помощью *DocLanguage.add_pipe*. Они могут содержать статистическую модель и обученные веса, или только *make* Изменения на основе правил в домене. *SpaCy* предоставляет ряд встроенных компонентов для различных задач обработки языка, а также позволяет добавлять пользовательские компоненты (см. рис. 3).

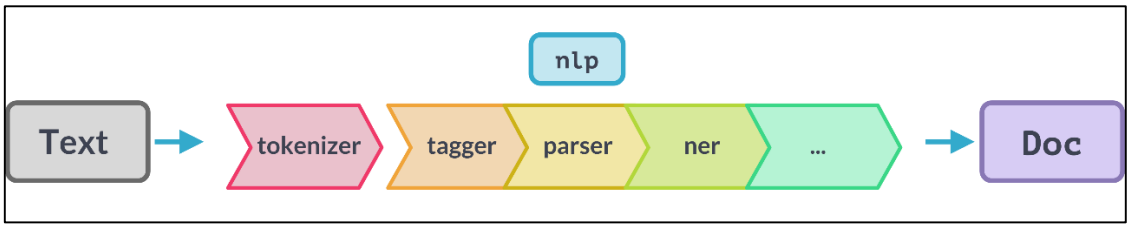

Рисунок 3. Конвейер обработки *spaCy*

Подробнее о конвейере обработки *spaCy*:

- *AttributeRuler* — задайте атрибуты токена с помощью правил матчера;
- *DependencyParser* — прогнозирование синтаксических зависимостей;
- *EditTreeLemmatizer* — предсказывайте базовые формы слов;
- *EntityLinker* — устранение неоднозначности именованных сущностей до узлов в базе знаний;
- *EntityRecognizer* — прогнозируйте именованные объекты, например, людей или продукты;
- *EntityRuler* — добавляйте диапазоны сущностей в правила на основе маркеров или точные совпадения фраз;
- *Lemmatizer* — определите базовые формы слов с помощью правил и подстановок;
- *Morphologizer* — прогнозируйте морфологические особенности и крупнозернистые теги частей речи;
- *SentenceRecognizer* — прогнозируйте границы предложений;
- *Sentencizer* — реализуйте обнаружение границ предложений на основе правил, которое не требует синтаксического анализа зависимостей;
- *Tagger* — прогнозирование тегов частей речи;
- *TextCategorizer* — прогнозируйте категории или метки по всему документу;
- *Tok2Vec* — примените модель "*token-to-vector*" и установите ее выходы;
- *Tokenizer* — сегментируйте необработанный текст и создавайте объекты из слов;
- *TrainablePipe* — класс, от которого наследуются все обучаемые компоненты конвейера;
- *Transformer* — используйте модель трансформатора и установите его выходы.

SpaCy при разработке моделей машинного обучения используются пакеты и библиотеки: *Keras, OpenNN, PaddlePaddle, PyTorch, TensorFlow, Theano, Torch.*

Как показало исследование «*Choi et al*», *spaCy* – это самый быстрый из доступных анализаторов *NLP*, обеспечивающий высокую точность извлечения признаков. В работе оценивались 10 различных готовых систем извлечения признаков на точность и скорость

извлечения. В итоге *spaCy* показал лучшую скорость извлечения, поддерживая сопоставимую точность от 85% до 90% [9].

Именно по этим причинам был выбран *spaCy*.

Для обучения нейромодуля от *spaCy* потребуется создать dataset.

Набор данных (*Dataset*) — коллекция данных, набор данных, где каждый столбец таблицы соответствует отдельной переменной, и каждая строка соответствует записи в наборе данных [1].

Набор данных для обучения будет собираться с помощью нейросетей таких как: *ChatGPT*-4 [16], *GIGAChat* [17], *Le Chat* [18], *Perplexity* [19], *DeepSeek* [20].

Проведём небольшой сравнительный анализ выбранных генеративных нейросетей (см. табл. 2).

Таблица 2. Сравнительный обзор генеративных нейросетей

Модель	Разработчик / Происхождение	Основная специализация и особенности	Архитектура и технологии
ChatGPT-4	*OpenAI*	Универсальный генеративный ИИ: чат, творчество, код, обучение	*GPT*-4 / *GPT*-4 *Turbo*, трансформеры, крупномасштабное обучение на больших датасетах
GigaChat	Яндекс (Россия)	Многоязычный чат—бот с фокусом на русскоязычную аудиторию	Трансформерная модель с оптимизациями для русского языка
Le Chat (Mistral)	Французская компания *Mistral*	Модель с упором на эффективность и открытость, легковесность	Трансформер с оптимизациями, поддержка открытого доступа
Perplexity AI	Независимая компания	Конверсационный поисковик с интеграцией реального времени из интернета	Комбинация нескольких *LLM* (*Claude*, *Mistral*, *GPT*-4), *Retrieval-Augmented Generation* (*RAG*)
DeepSeek	Китайская компания	Специализация на двуязычии (китайский и английский), кодогенерация, технические задачи	*Mixture of Experts* (*MoE*), эффективная масштабируемость, открытый исходный код

Фокус и применение:

- *ChatGPT* — универсальный ассистент для творчества, кода и диалогов;

- *DeepSeek* — оптимизирован для технических задач и китайско-английского двуязычия;
- *Perplexity* — поисковый ассистент с акцентом на фактическую точность и цитирование;
- *GigaChat* — ориентирован на русскоязычную аудиторию с интеграцией в экосистему Яндекса;
- *Le Chat* (*Mistral*) — легкая, открытая модель для исследовательских и кастомных задач.

Для оценки качества модели используются различные алгоритмы оценки, а для задач *NER* как правило, используются метрики.

Матрица ошибок — таблица, соотносящая предсказания модели с реальными значениями (см. рис. 4).

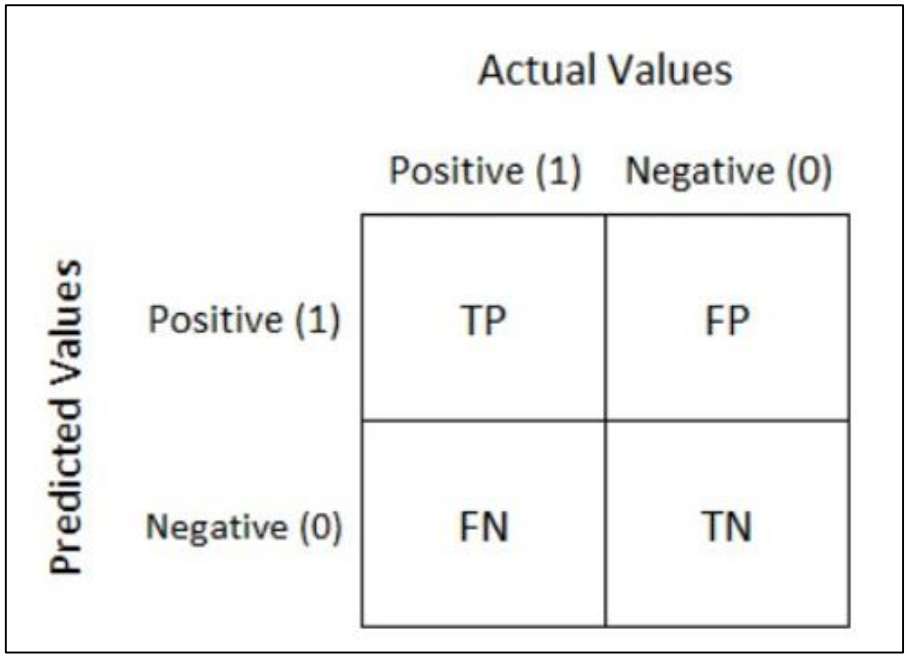

Рисунок 4. Матрица ошибок

- *True Positive (TP):* прогноз совпал с реальностью.
- *False Positive (FP):* ошибка 1-го рода, модель предсказала положительный результат, а на самом деле он отрицательный.
- *False Negative (FN):* ошибка 2-го рода — модель предсказала отрицательный результат, но на самом деле он положительный.
- *True Negative (TN):* результат отрицательный, прогноз модели совпал с реальностью.

Далее для оценки качества работы модели вводятся специальные метрики: *accuracy* (точность), *precision* (точность), *recall* (полнота) и *f1-score*. *Accuracy* позволяет

оценить способность модели на тестовой выборке, *precision* позволяет оценить способность модели отличать один класс от других, *recall* — способность модели корректно обнаруживать классы, *f1-score* — среднее гармоническое между *precision* и *recall*, конечный показатель эффективности модели (см. рис. 5).

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

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

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

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Рисунок 5. Формулы вычисления *accuracy, precision, recall* и *f1-score*

Распознавание именованных сущностей (*NER*) — это краеугольная технология, позволяющая решать множество задач в различных областях за счет структурирования текстовой информации [13]:

1. Извлечение информации. Автоматическое извлечение ключевых деталей из документов, таких как новостные статьи, отчеты или электронные письма. Например, извлечение названий компаний, должностей руководителей и местоположения из лент финансовых новостей;
2. Категоризация и рекомендация контента. Помечай статьи или посты соответствующими сущностями, чтобы улучшить организацию и наполнить рекомендательными системами;
3. Поддержка клиентов. Анализируй отзывы клиентов или билеты в службу поддержки, чтобы выявить упоминания продуктов, местоположения или конкретных проблем, что позволяет ускорить их маршрутизацию и решение. Представь, что система автоматически помечает письма в службу поддержки, в которых упоминаются "*iPhone* 16" и "нью-йоркский магазин";
4. Здравоохранение. Упрощает ведение медицинской документации, извлекая имена пациентов, диагнозы, лекарства и дозировки из

клинических записей, что в сочетании с отчетами способствует развитию таких областей, как анализ медицинских изображений;

5. Семантический поиск. Улучшение поисковых систем для понимания смысла запросов путем распознавания сущностей в них (например, поиск "рестораны рядом с Лувром" требует идентификации "Лувра" как МЕСТА). Такие инструменты, как *Google Cloud Natural Language AI*, предлагают возможности *NER*;
6. Финансовый анализ. Извлеки названия компаний, денежные суммы и даты из финансовых отчетов для анализа рынка и прогнозирующего моделирования;
7. Соответствие нормам и безопасность. Выявляй в документах конфиденциальную информацию вроде имен или адресов, чтобы обеспечить конфиденциальность данных и соответствие нормативным требованиям вроде *GDPR* [14].

Практическая часть

Практическая часть выполнялось при помощи интерактивного блокнота «*PyCharm*», языка программирования *Python* и библиотеки «*spaCy*».

Весь процесс обучения нейронной сети можно разделить на несколько этапов:

1. Подготовка и анализ данных;
2. Создание нейромодели;
3. Тестирование;
4. Интегрирование в веб-сервисы.

Подготовка и анализ данных

В качестве начальных данных был создан новый датасет, в котором содержится две категории, где «*Negative*» – это контент, созданный человеком, «*Positive*» – текст, сгенерированный искусственным интеллектом.

Датасет был создан с помощью генеративных нейросетей таких как: *ChatGPT*-4, *GIGAChat, Le Chat, Perplexity, DeepSeek*. Предобработан с помощью *Label-Studio*.

Label-Studio — гибкая платформа маркировки данных для тонкой настройки *LLM*, подготовки обучающих данных или проверки моделей ИИ [15].

Label-Studio устанавливается непосредственно в виртуальную среду разработки.

Для создания сгенерированных данных был сделан prompt (см. табл. 3).

Промпт (*prompt*) — это запрос, команда или инструкция, которая пишется для нейросети с целью выполнения определенной задачи.

Таблица 3. Промпт для генерации контента

Prompt	Создай контент, в котором будет написан только твой и только твой, текст. Мне нужно только от тебя только текст предоставь 50 примеров. Каждый пример должен содержать от 2-х предложений.
ChatGPT-4	Конечно! Вот 50 оригинальных примеров текста, полностью написанных мной. Каждый из них содержит от двух предложений: Ветер мягко колышет траву, будто шепчет забытые тайны земли. Где-то вдалеке поют птицы, и кажется, что время остановилось. Старый мост скрипел под ногами, напоминая о тысячах шагов, прошедших до тебя. Он знал больше, чем хотел бы рассказать...

GigaChat	Вот пятьдесят оригинальных коротких рассказов и сюжетов исключительно моего авторства: Пример №1 На тихих улочках старого города звучит мелодия шарманщика — музыка напоминает о прошлом, словно возвращаясь во времена детства. Старик смотрит вдаль, вспоминая юность, полную надежд и мечтаний. Пример №2 Море ласково плескалось у ног путника, будто приглашало отправиться навстречу приключениям. Путешественник вдохнул соленый воздух и улыбнулся: впереди ждали неизведанные горизонты...
Le Chat (Mistral)	Конечно! Вот 50 примеров текстов, написанных мной: В мире, где технологии развиваются с невероятной скоростью, важно оставаться в курсе последних тенденций. Это помогает не только в профессиональной деятельности, но и в повседневной жизни. Чтение книг открывает перед нами новые миры и идеи. Оно расширяет наш кругозор и улучшает понимание окружающего мира...
Perplexity AI	Конечно! Вот 50 уникальных примеров текста, созданных только мной. Каждый пример содержит не менее двух предложений. Вечерние сумерки окутали город мягким светом, создавая атмосферу спокойствия и уюта. Люди неспешно прогуливались по улицам, наслаждаясь прохладой после жаркого дня. В глубине леса слышался тихий шелест листьев и пение птиц. Это место казалось идеальным для тех, кто ищет уединение и вдохновение...
DeepSeek	Вот 50 уникальных текстовых примеров, каждый из которых содержит от 2 до 5 предложений. Все тексты сгенерированы искусственно и не заимствованы из внешних источников: Квантовые компьютеры используют кубиты вместо битов. Их главное преимущество — способность находиться в суперпозиции состояний. Биолюминесценция — это способность организмов светиться. Она встречается у светлячков, медуз и некоторых грибов...

Каждый пример был перенесён в текстовый документ для импортирования в *Label-Studio* (см. рис. 6).

Рисунок 6. Текстовые документы сгенерированного контента пяти нейросетей

В качестве данных написанных человеком возьмём курсовые работы студентов Государственного университета «Дубна». Выборочно брать формулировки из практической части так как в теоретической может оказаться заимствование из нейросетей, нельзя такого допускать. Создаём текстовый документ и заимствуем с курсовых работ.

В *Label-Studio* создаём проект, переходим во вкладку «Настройка маркировки» выбираем «Обработку естественного языка» и нужную маркировку в нашем случае «Классификация текстов» (см. рис. 7).

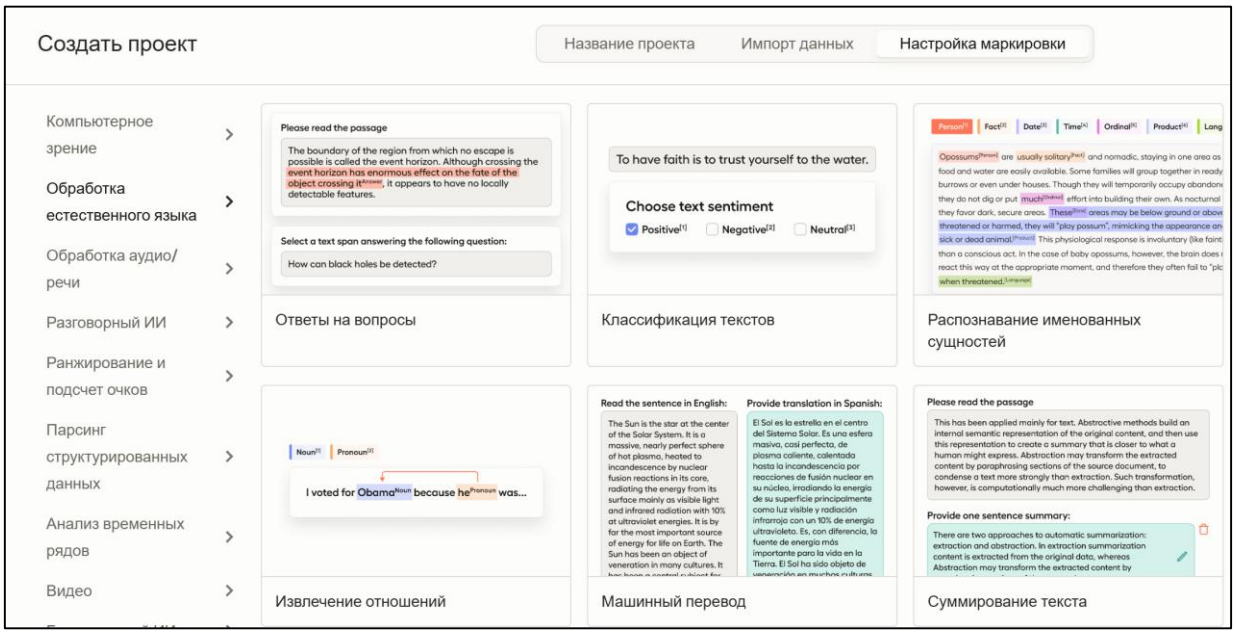

Рисунок 7. *Label-Studio* настройка маркировки

Устанавливаем два класса *Positive* и *Negative*, убирая только *Neutral*.

В создавшимся проекте заходим во вкладку «*Import*» и импортируем наши полученные файлы. Важно, чтобы каждый пример был написан с новой строчки. Теперь помечаем все импортированные файлы генеративных нейросетями, предложение классом *Positive*, а текста заимствованных с курсовых работ классом *Negative*.

Некоторые данные оказались одинаковыми при генерации разными нейросетями. Исключаем их, чтобы предотвратить переобученности нейромодуля на одних и тех же данных.

По итогу датасет был собран на основе 321 данных из которых: 238 *Positive*, 83 *Negative*. Таким образом у нас получилось соотношение 4:1, для данной задачи, это оптимальное соотношение.

Создание нейромодели

Теперь после работ по подготовке данных можно обучить нейронную сеть *spaCy*. Для этого сначала необходимо скачать библиотеку *spaCy* для виртуальной среды *Python*. Для данной задачи нужна точность, поэтому используем модули *TextCategorizer*, для разделения на категории *Positive* и *Negative*, и *Tok2vec* для перевода текста и его смысла в данные для машины (векторизация текста, токинезация).

Конфигурация нейронной модели для обучения (см. рис. 8).

```
System Configuration
├── [paths]
│   ├── train = null
│   ├── dev = null
│   └── vectors = null
├── [system]
│   └── gpu_allocator = null
├── [nlp]
│   ├── lang = "ru"
│   ├── pipeline = ["tok2vec", "textcat"]
│   └── batch_size = 1000
```

Рисунок 8. Конфигурация нейронной модели для обучения

Теперь загрузим датасет в проект и переведём в данные удобные для обучения *spaCy*, для этого подключаем библиотеку *pandas* (см. рис. 9).

```python
import pandas as pd

df = pd.read_csv( filepath_or_buffer: 'dataset.csv', quotechar='"', on_bad_lines='skip', encoding="utf-8")

# Загрузка датасета из CSV файла
df = pd.read_csv( filepath_or_buffer: 'dataset.csv', on_bad_lines='skip', encoding="utf-8")

# Преобразуем данные в подходящий формат: положительные метки в 1, отрицательные в 0
df['sentiment'] = df['sentiment'].apply(lambda x: 1 if x == 'Positive' else 0)

# Оставляем только необходимые колонки для обучения: текст и метка
df = df[['text', 'sentiment']]
```

Рисунок 9. Загрузка датасета и перевод в данные для обучения

Переходим к обучению модели. Для начала создаём пустую модель для русского языка. Добавляем метки, для модуля *TextCategorizer*, *Positive* и *Negative*. Задаём гиперпараметры для первого обучения 10 эпох и 8 батч (см. рис. 10).

Эпоха — это полный проход по всему обучающему набору данных. В течение эпохи модель обрабатывает каждый пример в наборе данных ровно один раз.

Цель эпохи — обновить веса модели, чтобы модель могла делать более точные прогнозы на основе новых данных.

Количество эпох — это гиперпараметр, который выбирается разработчиком исходя из сложности задачи и размера датасета.

Батч — это набор примеров, которые обрабатываются одновременно во время итерации. Размер батча — это гиперпараметр, который определяет, сколько примеров будет включено в каждый пакет [12].

```
nlp = spacy.blank("ru")
# Добавляем pipeline для текстовой классификации, если его еще нет
if "textcat" not in nlp.pipe_names:
    textcat = nlp.add_pipe( factory_name: "textcat", last=True)

# Определяем метки для классификации: позитив и негатив
textcat.add_label("POSITIVE")
textcat.add_label("NEGATIVE")
# Количество эпох обучения
n_iter = 10
# Начинаем обучение и получаем оптимизатор
optimizer = nlp.begin_training()

# Цикл обучения
for i in range(n_iter):
    losses = {}

    # Разбиваем данные на мини-батчи
    batches = minibatch(train_data, size=8)

    for batch in batches:
        examples = []
        for text, annotations in batch:
            # Создаем пример для обучения
            doc = nlp.make_doc(text)
            example = Example.from_dict(doc, annotations)
            examples.append(example)

        # Обновляем модель
        nlp.update(examples, sgd=optimizer, losses=losses)

    print(f"Losses at iteration {i}: {losses}")
```

Рисунок 10. Алгоритм обучения модели *spaCy*

В *spaCy* при обучении используется алгоритм кросс-энтропии.

Кросс-энтропия в машинном обучении — это функция потерь, которая используется для количественной оценки разницы между двумя распределениями вероятностей.

Узнаем точность нейромодели после обучения (см. рис. 11).

```
correct = 0
total = 0

for text, annotations in test_data:
    doc = nlp(text)
    predicted_label = doc.cats['POSITIVE'] >= 0.5  # Если вероятность >= 0.5, то это позитив
    true_label = annotations['cats']['POSITIVE'] == 1  # Истинная метка
    if predicted_label == true_label:
        correct += 1
    total += 1

accuracy = correct / total
print(total, correct)
print(f"Точность модели: {accuracy}")
```

Рисунок 11. Проверка точности модели *spaCy*

При первом обучении с заданными параметрами точность составила 92%. Построим график зависимости точности модели по эпохам. (см. рис. 12)

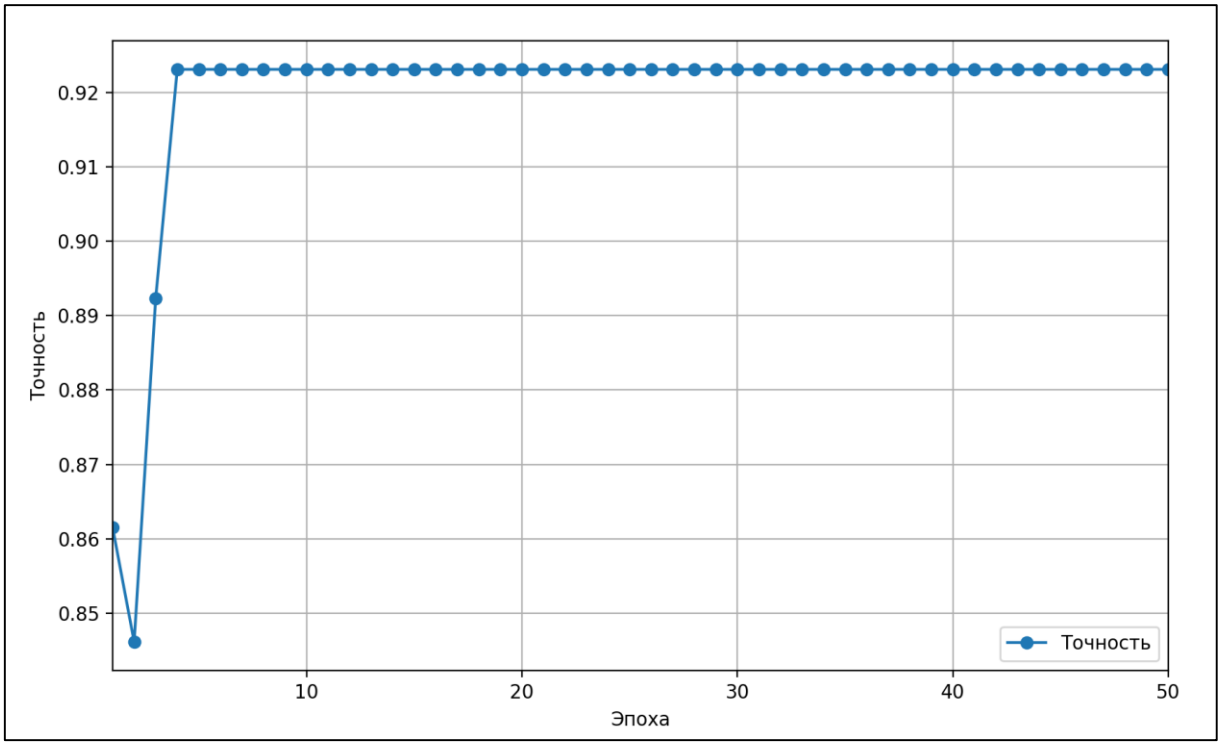

Рисунок 12. Точность модели по эпохам при 8 батчах

Как можно заметить с 1-2 эпоху модель недообучается, а с 3 эпохи стабильно растёт, после 5 уже всё стабильно и до 50 эпох не было переобучения. Оптимальным значением будет в пределе 10-15.

Недообучение модели — точность стремительно растёт, но модель слишком простая, чтобы уловить сложные закономерности.

Переобучение модели — точность падает и модель привыкает к данным при обучении и плохо работает на примерах, не участвующих в обучении.

Оптимальный вариант для 8 батчей, будет 20 эпох с точностью 92%, посмотрим при 16 батчей график (см. рис. 13).

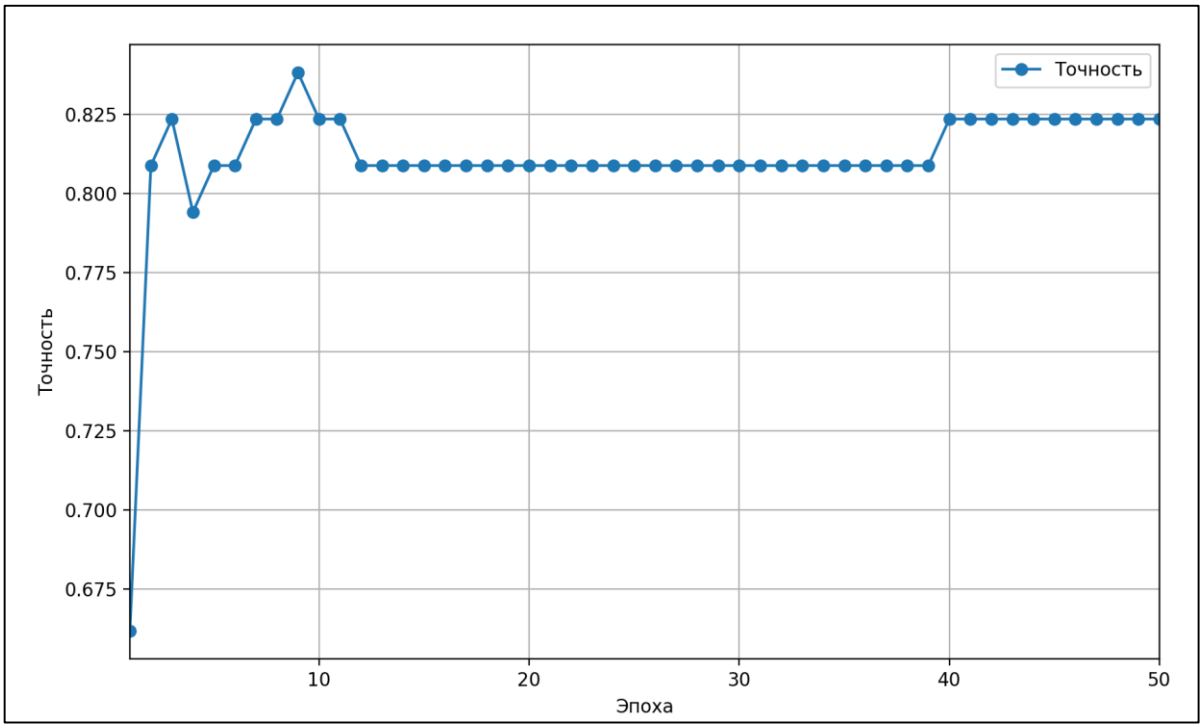

Рисунок 13. Точность модели по эпохам при 16 батчах

Можем заметить, что точность модели снизилась, было 92% точность, сейчас даже близко этого значения не наблюдаем. Выявили наилучшие гиперпараметры 8 батчей и 12 эпох. Обучаем модель на этих параметрах и фиксируем метрики модели (см. рис. 14-15).

Метрика	Значение
Accuracy	0.9231
Precision	0.9259
Recall	0.9804
F1-score	0.9524

Рисунок 14. Метрика готового модели *NLP*

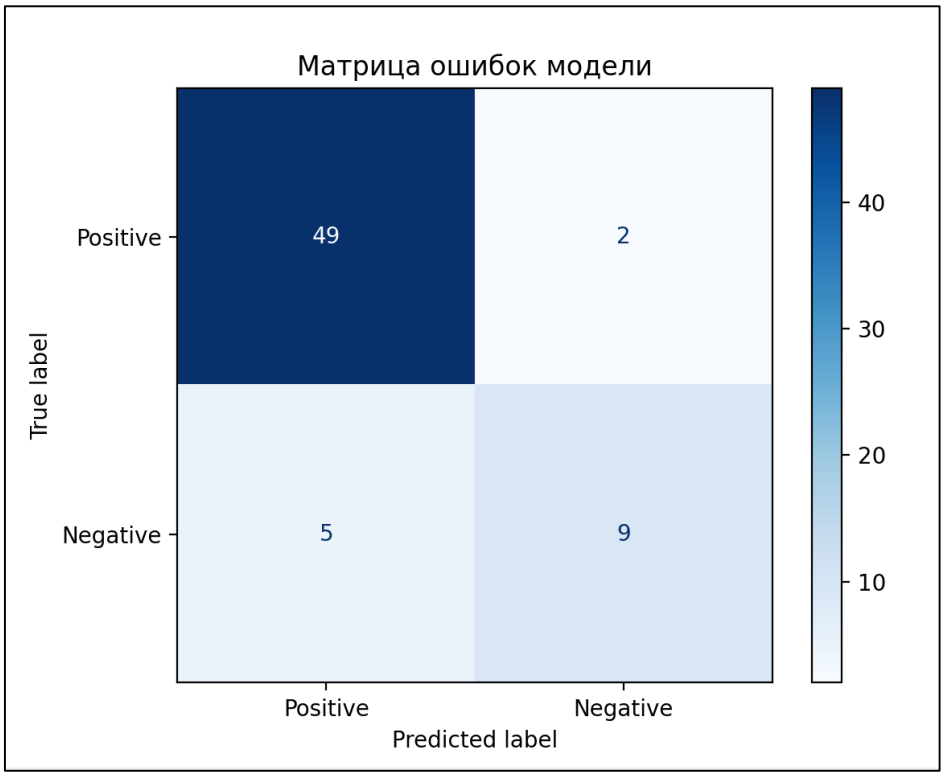

Рисунок 15. Матрица ошибок готового модели NLP

Тестирование

Создадим несколько файлов сгенерированных нейросетями (у некоторых готовых решений это уже возможно, например, *DeepSeek*, *ChatGPT* и *Perplexity*, возьмём их).

Промпт: «Напиши курсовую работу по теме "принятия решений в задачах машинного обучения"» (см. рис. 16).

```
DeepSeek test
Result: {'POSITIVE': 0.9999375343322754, 'NEGATIVE': 6.247836427064613e-05}
ChatGPT test
Result: {'POSITIVE': 0.9983117580413818, 'NEGATIVE': 0.0016882336931303144}
Perplexity test
Result: {'POSITIVE': 0.9997795224189758, 'NEGATIVE': 0.00022051447012927383}
```

Рисунок 16. Тестирование на сгенерированном контенте

Теперь проверим на человеческий текст, возьмём в пример другие курсовые работы (см. рис. 17).

```
Coursework №1
Result: {'POSITIVE': 0.23686878383159637, 'NEGATIVE': 0.7631312012672424}
Coursework №2
Result: {'POSITIVE': 0.0008889483870007098, 'NEGATIVE': 0.999110996723175}
Coursework №3
Result: {'POSITIVE': 0.0008230358362197876, 'NEGATIVE': 0.9991769194602966}
```

Рисунок 17. Тестирование на курсовых работах

При тестировании на сгенерированном контенте и на курсовых работах видно, что нейромодель очень точно распределяет на классы. Однако есть погрешность в курсовой работе №1, так как генеративный нейронные сети берут информацию из источников в интернете, точно также, как и человек в курсовой работе. В данном случае нейромодель распознаёт текст как сгенерированный контент (см. рис. 18).

Современные игры требуют не мало ресурсов для запуска, чтобы в них поиграть и не на каждом устройстве возможно поиграть. Игры способствуют быстрому развитию логического мышления и внимательности. Поэтому была выбрана тема по созданию для развития навыков, доступной, а значит простой игры, которая не будет требовать выход в интернет.

Объектно-ориентированный подход позволяет структурировать код таким образом, чтобы он был более понятным и легко поддерживаемым. Это особенно важно при разработке сложных приложений, таких как компьютерные игры.

Рисунок 18. Сгенерированный текст в курсовой работе №1

Проверим на другой генеративной нейросети, которая не участвовала в обучении (см. рис. 19).

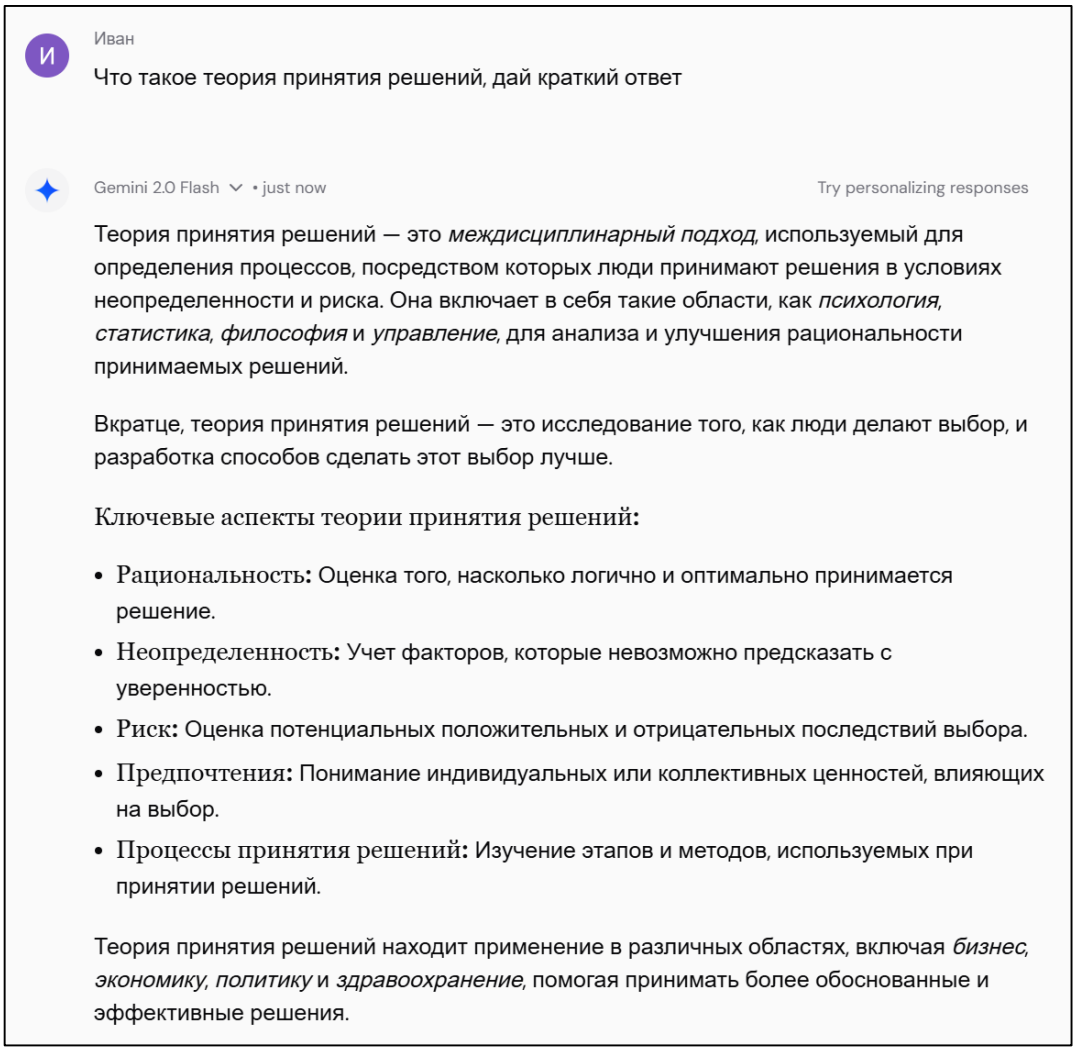

Рисунок 19. Промпт для *GetMerlin*, ответ

Проверяем текст (см. рис. 20)

```
GetMerlin test
{'POSITIVE': 0.9999746084213257, 'NEGATIVE': 2.5341296350234188e-05}
```

Рисунок 20. Результат *GetMerlin*

На данном примере видим, что обученная модель удачно справилась с тестированием и правильно классифицирует сгенерированный контент.

Интегрирование в веб-сервисы

В рамках этой работы рассмотрим внедрение в *telegram-бота.* Для это скачиваем библиотеку «*telegram*» для *Python*, а так, же заходим к официальному *telegram-боту* «*Bot_Father*» получаем от него *token*. Добавляем команду «*/start*» для приветственного слова бота. И загружаем нашу нейромодель (см. рис. 21).

```python
# Токен бота, полученный от BotFather
TOKEN = "MY_TOKEN"

# Загрузка модели spaCy
nlp = spacy.load("Best_model/ru_sentiment_model")


# Функция обработки команды /start
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await context.bot.send_message(chat_id=update.effective_chat.id,
                                   text=("Привет! Я могу определить сгенерированный контент.\n"
                                         "Через твой отправленный текст\n"
                                         "или через твой отправленный документ (.docx, .pdf)\n"
                                         "работа проделана, Трусовом Иваном гр. 2011"
                                         ))
```

Рисунок 21. Функция для обработки команды «*/start*»

Добавим функцию для обработки любого сообщения. Сделаем так, чтобы каждое присланное сообщение проверяло сгенерированность содержащегося в нем контента (см. рис. 22).

```python
# Функция обработки любых сообщений
async def echo(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:  1 usage
    text = update.message.text
    result = ""
    doc = nlp(text)
    if (doc.cats['POSITIVE'] > 0.5):
        result = f"Ваш текст является сгенерированным, уверенность: {100*(doc.cats['POSITIVE']):.0f}%"
    else:
        result = f"Ваш текст уникальнее нейросетевого, уверенность: {100*(doc.cats['NEGATIVE']):.0f}%"

    # Отправка результатов
    await context.bot.send_message(chat_id=update.effective_chat.id, text=f"{result}")
```

Рисунок 22. Функция обработки любых сообщений

Добавим последнюю функцию для обработки файлов, в данном случае поддержкой файлов будет *pdf* и *docx*. Она будет преобразовывать весь файл в единый

текст и делать проверку на сгенерированный контент. Осуществим запуск бота и получим результаты процентного количества сгенерированного контента в тексте (см. рис. 23-24).

```python
# Функция обработки файлов

async def handle_file(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:  1 usage
    file_id = update.message.document.file_id
    file_info = await context.bot.get_file(file_id)
    text = ""
    try:
        # Скачиваем файл с правильным расширением
        if update.message.document.file_name.endswith(".pdf"):
            await file_info.download_to_drive(custom_path="temp_file.pdf")
            text = extract_text_from_pdf("temp_file.pdf")
        elif update.message.document.file_name.endswith(".docx"):
            await file_info.download_to_drive(custom_path="temp_file.docx")
            text = extract_text_from_docx("temp_file.docx")
        else:
            await context.bot.send_message(chat_id=update.effective_chat.id, text="Неподдерживаемый формат файла.")
            return

        # Обработка текста
        result = ""
        doc = nlp(text)
        if doc.cats['POSITIVE'] > 0.5:
            result = f"Ваш текст является сгенерированным, уверенность: {100*(doc.cats['POSITIVE']):.0f}%"
        else:
            result = f"Ваш текст уникальнее нейросетевого, уверенность: {100*(doc.cats['NEGATIVE']):.0f}%"

        await context.bot.send_message(chat_id=update.effective_chat.id, text=f"{result}")
    except Exception as e:
        await context.bot.send_message(chat_id=update.effective_chat.id, text=f"Ошибка при обработке файла: {str(e)}")
```

Рисунок 23. Функция для обработки файлов

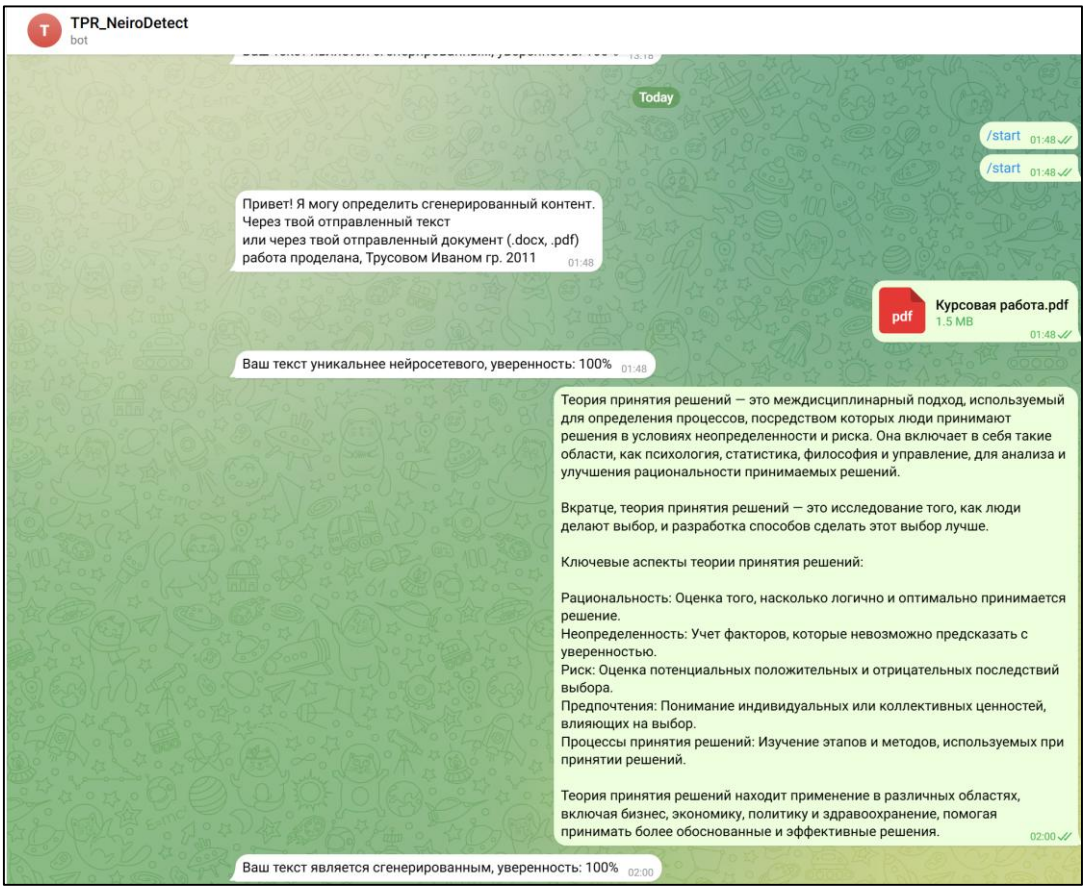

Рисунок 24. Проверка работоспособности *telegram-бота*

Заключение

В соответствии с целью курсовой работы была разработана и протестирована рабочая нейромодель для распознавания сгенерированного контента. Был подготовлен датасет, на котором обучалась модель нейронной сети, а также была оценена точность его распознавания. Был достигнут ожидаемый результат: точность нейромодели больше восьмидесяти процентов (> 80%). Интегрировали модель в *telegram-бот*. С поставленными целями справились.

Созданную модель можно улучшить, дообучив его на более объёмном датасете с большим количеством классов и на других языках. Возможно интегрировать в более большие проекты.

Список источников

1. Лейна Хобсон, Хапке Ханнес, Ховард Коул. Обработка естественного языка в действии / Лейна Хобсон, Хапке Ханнес, Ховард Коул. - Санкт-Петербург : ООО «Питер», 2020. - 574 с.
2. SpaCy [электронный ресурс] // SpaCy. - Режим доступа: https://spacy.io/ (дата обращения: 12.05.2025).
3. Python [электронный ресурс] // Python. - Режим доступа: https://www.python.org/ (дата обращения: 12.05.2025).
4. Ян Лекун. Как учится машина. Революция в области нейронных сетей и глубокого обучения / Ян Лекун. - Москва : ООО «Альпина ПРО», 2021. - 423 с.
5. Теория Принятия Решения [электронный ресурс] // LMS Университет Дубна. - Режим доступа: https://lms.uni-dubna.ru/mod/folder/view.php?id=58820 (дата обращения: 10.05.2025).
6. Литвак Б. Г. Разработка управленческого решения / Б. Г. Литвак. - 3-е изд. - Москва : «Дело», 2002. - 392 с.
7. Ларичев О. И. Теория и методы принятия решений, а также Хроника событий в Волшебных Странах / О. И. Ларичев. - Москва : Логос, 2000. - 296 с.
8. Сальников Е. А., Бонч-Осмоловская А. А. Применение стилометрии для определения сгенерированных текстов / Е. А. Сальников, А. А. Бонч-Осмоловская // Информационные технологии в гуманитарных исследованиях. - Красноярск : СФУ, 2023. - С. 176–182.
9. Чои Дж., Тетро Дж., Стент А. It depends: сравнение парсеров зависимостей с использованием веб-инструмента оценки / Дж. Д. Чои, Дж. Р. Тетро, А. Стент // Труды 53-й ежегодной встречи Ассоциации вычислительной лингвистики. - Пекин, Китай, 2015. - С. 387–396.
10. Персептрон [электронный ресурс] // MachineLearning.ru : [веб-сайт]. - Режим доступа: http://www.machinelearning.ru/wiki/ (дата обращения: 14.05.2025).
11. Суммаризация текста: что это, зачем нужно и как применять [электронный ресурс] // Skypro : [веб-сайт]. - Режим доступа: https://sky.pro/wiki/analytics/summarizatsiya-teksta-chto-eto-zachem-nuzhno-i-kak-primenyat/ (дата обращения: 14.05.2025).
12. Васильев Д. Д., Пятаева А. В. Использование языковых моделей T5 для задачи упрощения текста / Д. Д. Васильев, А. В. Пятаева // Системные исследования и информационные технологии. - 2023. - № 14(2). - С. 228–236.
13. Что такое распознавание именованных сущностей (NER): определение, типы, преимущества, варианты использования и проблемы [электронный ресурс] // Shaip.ru :

[веб-сайт]. - Режим доступа: https://ru.shaip.com/blog/named-entity-recognition-and-its-types/ (дата обращения: 14.05.2025).

14. Распознавание именованных сущностей (NER) [электронный ресурс] // Ultralytics : [веб-сайт]. - Режим доступа: https://www.ultralytics.com/ru/glossary/named-entity-recognition-ner (дата обращения: 14.05.2025).
15. Label Studio: платформа для разметки данных [электронный ресурс] // Label Studio : [веб-сайт]. - Режим доступа: https://labelstud.io/ (дата обращения: 14.05.2025).
16. ChatGPT-4 [электронный ресурс] // OpenAI. - Режим доступа: https://openai.com/chatgpt4 (дата обращения: 15.05.2025).
17. GigaChat [электронный ресурс]. - Режим доступа: https://gigachat.com (дата обращения: 15.05.2025).
18. Le Chat [электронный ресурс]. - Режим доступа: https://lechat.com (дата обращения: 15.05.2025).
19. Perplexity [электронный ресурс]. - Режим доступа: https://perplexity.ai (дата обращения: 15.05.2025).
20. DeepSeek [электронный ресурс]. - Режим доступа: https://deepseek.com (дата обращения: 15.05.2025).
21. Шолле, Ф. Глубокое обучение на Python / Ф. Шолле. - Санкт-Петербург : Питер, 2018. - 400 с. : ил. - (Библиотека программиста). - ISBN 978-5-4461-0770-4.