МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра Математического и компьютерного моделирования

Создание модели искусственного интеллекта для автодополнения и генерации текста

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы

направление 01.03.02 — Прикладная математика и информатика

механико-математического факультета

Пушкарева Даниила Игоревича

Научный руководитель
доцент, к.ф.-м.н., доцент _____ С.П. Шевырев

Зав. кафедрой
зав. каф., д.ф.-м.н., доцент _____ Ю.А. Блинков

Саратов 2023

Введение. Современный мир стало невозможно представить без использования искусственного интеллекта (ИИ). Он проникает во все сферы жизни, помогает упростить и автоматизировать процессы, улучшить качество жизни людей. Одной из задач, решаемых с помощью ИИ, является создание систем автодополнения и генерации текста. В данной бакалаврской работе рассмотрена модель искусственного интеллекта, которая способна предсказывать продолжение текста на основе его начала и генерировать новый текст на основе заданных параметров. Эта модель может быть использована в различных областях, таких как маркетинг, журналистика, лингвистика и другие. Одним из ключевых аспектов задачи о создании систем автодополнения и генерации текста являются рассмотрение, выбор и создание модели искусственного интеллекта на основ модели. Существует несколько моделей искусственного интеллекта, которые используются для генерации текста. Они различаются по сложности и способу работы.

1. Модель марковских цепей — это простая модель, которая основана на вероятностях перехода от одного состояния к другому. Она использует историю текста для предсказания следующего слова. В этой модели каждое слово считается состоянием, а вероятность перехода к следующему слову определяется на основе предыдущих слов.

2. Рекуррентные нейронные сети (RNN) — это более сложная модель, которая может учитывать зависимости между словами в тексте. Она использует память для хранения информации о предыдущих словах и применяет ее для генерации следующего слова. RNN может быть обучена на больших объемах текстовых данных и создавать более качественные результаты.

3. Трансформеры — это новая модель, которая использует механизм внимания для генерации текста. Они могут учитывать контекст и зависимости между словами в более широком контексте, чем RNN. Трансформеры могут обучаться на больших объемах данных и создавать более точные результаты.

Каждая из этих моделей имеет свои преимущества и недостатки, и выбор модели зависит от конкретной задачи. Однако все они используются для генерации текста и являются важными инструментами в различных областях, таких как машинный перевод, генерация текстовых описаний и создание искусственных текстов.

Целью данной работы является исследование и создания модели искусственного интеллекта для генерации текста с использованием языка Python. Для того, чтобы разобраться в данном вопросе, для данной бакалаврской работы были поставлены следующие цели:

- Изучение моделей искуссвенного интеллекта
- Изучение моделей нейронных сетей
- Подробный обзор моделей обучения нейронных сетей
- Написание программного кода для создания модели автодополнения и генерации текста.

Структура бакалаврской работы. Бакалаврская работа содержит введение, три раздела, заключение, список использованной литературы и одно приложение. Во введении обоснована актуальность работы, сформулированы цели и задачи. В первом разделе содержится теоретическая часть, описано строение больших языковых моделей, а также предшествующие популярные модели предсказания текста.

Во втором разделе рассмотрена постановка задачи и критерии, которым итоговая языковая модель должна соответствовать.

В третьей части приведена созданная модель и проведены сравнения с установленными изначально целями.

Основное содержание работы. В настоящее время появляется всё больше и больше новых разработок, а также технологий, связанных с искусственным интеллектом, будь то системы, создающие изображения, производящие анализ, предсказания погоды, создающие и анализирующие тексты. Структура искусственного интеллекта может включать несколько компонентов, таких как:

1. Сенсоры – устройства, которые получают данные из окружающей среды, например, микрофоны, камеры, сенсоры движения.
2. Программное обеспечение для обработки данных – это компьютерные программы, которые обрабатывают данные, полученные от сенсоров, и выдают результаты.
3. Алгоритмы машинного обучения – это методы обработки данных, которые позволяют интеллектуальной системе самостоятельно учиться на основе опыта и делать выводы.

4. Базы данных – это хранилища информации, которые используются для обучения и работы искусственного интеллекта.
5. Интерфейсы - это средства взаимодействия между человеком и искусственным интеллектом, например, голосовые ассистенты или чат-боты.
6. Оборудование – это компьютеры и другие устройства, которые используются для работы искусственного интеллекта.

Структура искусственного интеллекта (ИИ) может быть очень сложной и зависит от конкретной задачи, которую нужно решить. Однако все компоненты взаимодействуют между собой, чтобы создать интеллектуальную систему, способную решать сложные задачи и делать выводы на основе данных.

Интеллект является общей структурой, который позволяет человеку мыслить, созерцать мир и приходить к логическому выводу, при решении тех или иных проблем. Естественный интеллект - это способность человека и животных к мышлению, решению задач, обучению и адаптации к окружающей среде. Структура естественного интеллекта включает следующие компоненты:

1. Мозг – основной орган управления и контроля за всеми процессами в организме. Он состоит из миллиардов нервных клеток, которые связываются между собой и передают информацию.
2. Чувства – это ощущения, которые возникают в ответ на воздействие наших органов чувств, таких как зрение, слух, обоняние, вкус и осязание.
3. Память – это способность запоминать информацию и использовать ее для решения задач и принятия решений.
4. Речь – это способность выражать свои мысли и коммуницировать с другими людьми.
5. Мышление – это способность решать задачи, анализировать информацию и делать выводы.
6. Адаптация – это способность приспосабливаться к изменяющейся окружающей среде и менять свое поведение и мышление в соответствии с новыми условиями.

Рекуррентная нейронная сеть (RNN) – вид нейронных сетей, где связи между элементами образуют направленную последовательность. Рекуррент-

ные нейронные сети — сети с циклами, которые хорошо подходят для обработки последовательностей.

Обучение RNN аналогично обучению обычной нейронной сети. Также используется алгоритм обратного распространения ошибки (англ. Backpropagation), но с небольшим изменением. Поскольку одни и те же параметры используются на всех временных этапах в сети, градиент на каждом выходе зависит не только от расчетов текущего шага, но и от предыдущих временных шагов. Например, чтобы вычислить градиент для четвертого элемента последовательности, нам нужно было бы «распространить ошибку» на 3 шага и суммировать градиенты. Этот алгоритм называется «алгоритмом обратного распространения ошибки сквозь время»

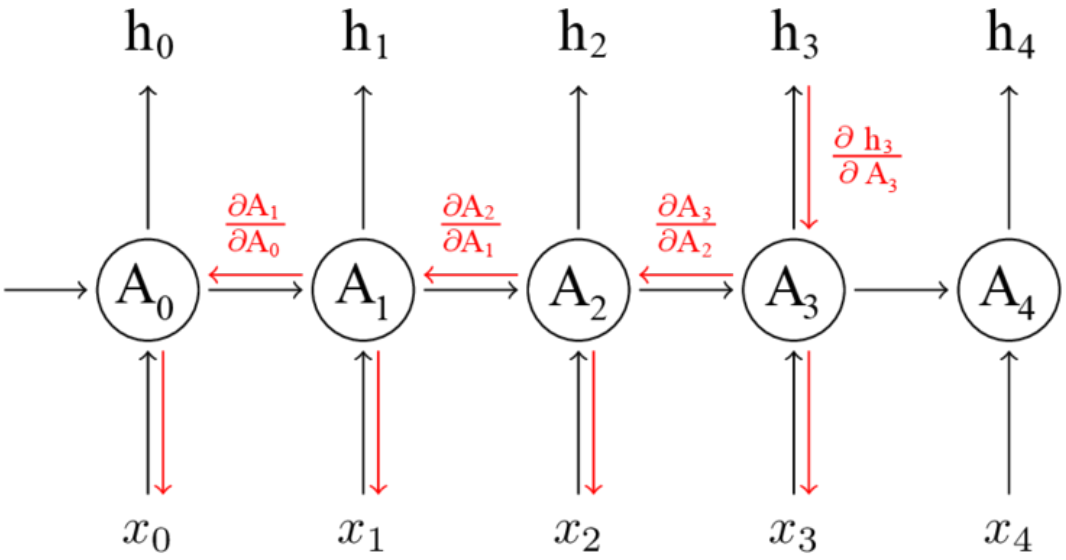

Рисунок 1 — Алгоритм обратного распространения ошибки сквозь время

Используются, когда важно соблюдать последовательность, когда важен порядок поступающих объектов.

- Обработка текста на естественном языке:
- Анализ текста;
- Автоматический перевод;
- Обработка аудио
- Автоматическое распознавание речи;
- Прогнозирование следующего кадра на основе предыдущих;

- Распознавание эмоций;
- Прогнозирование следующего пикселя на основе окружения;
- Генерация описания изображений.

Рекурсивная нейронная сеть использует следующую формулу для вычисления родительского вектора:

$$p_1 = f(W[b; c] + \text{ bias }),$$

- b, c - дочерние векторы - W - обученная матрица весов, $W \in R^{d \times 2d}$ - f - нелинейную функция активации типа гиперболического тангенса - bias - смещение, оно может быть добавлено в качестве дополнительного столбца к W, если к конкатенации входных векторов добавляется 1. Родительские векторы должны иметь одинаковую размерность, чтобы быть рекурсивно совместимыми и использоваться в качестве входных данных для следующей композиции. Деревья могут иметь разную структуру, выбор лучшей подструктуры дерева для сети основывается на их мере. Мера дерева – сумма мер на каждом узле:

$$s(x, y) = \sum_{n \in \text{nodes}(y)} s_n$$

$$\text{label}_p = \text{softmax}\left(W^{\text{label}}\, p\right)$$

Здесь W^{label} — матрица классификаций. Основной задачей и разницей между моделями будет вычисление скрытых векторов $p_i \in R^d$ снизу вверх.

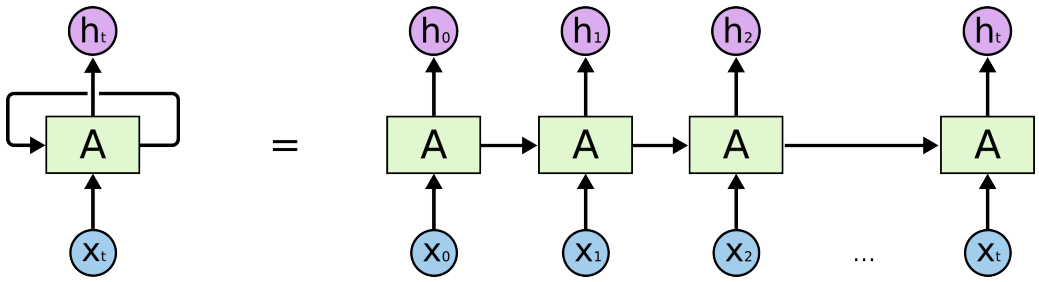

Рисунок 2 — Структура RNN

Для датасета было использовано произведение Александра Сергеевича Грибоедова – Горе от ума.

Данные были переведены в формат Txt и подготовлены для обучения нейронной сети.

Формат TXT, является одним из наиболее распространенных форматов для хранения текстовых данных. В этом формате текст хранится в виде обычного текста без каких-либо форматирований, шрифтов, изображений и других элементов.

Одним из основных преимуществ формата TXT является его универсальность и доступность. Файлы в формате TXT могут быть открыты на любом устройстве или операционной системе без необходимости установки дополнительного программного обеспечения. Кроме того, этот формат является одним из наиболее безопасных для хранения информации, так как он не содержит вредоносного кода или скриптов.

Был выполнен импорт библиотек.

```
import tensorflow as tf

import numpy as np
import os
import time
import requests
```

TensorFlow – это открытая программная библиотека для машинного обучения, разработанная компанией Google. Она позволяет создавать и обучать различные модели глубокого обучения, включая нейронные сети, сверточные нейронные сети и рекуррентные нейронные сети

Для ускорения вычислений библиотекой tensorflow, используется вычислительная мощность графического чипа видеокарты, благодаря этому обучение модели происходит сравнительно быстрее вычислений, проводящихся с использованием только оперативной памяти системы.

Для дальнейшей обработки данных был произведён импорт необходимых данных из хранилища:

```
text = open('/content/drive/MyDrive/Colab Notebooks/data/data.txt', 'rb')
                                .read()
                                .decode(encoding='utf-8')

print(f'Длина текста составляет: {len(text)} символов')
```

После выполненной операции был получен соотвесвтующий вывод:

```
Длина текста составляет: 107853 символов.
```

А также проведена проверка, корректного считывания данных из текста

```
print(text[:300])
```

Где был выполнен вывод соответствующих текстовых данных, заданных изначально в файле:

```
Лизанька
```

```
(вдруг просыпается, встаёт с кресел, оглядывается)

Светает!.. Ах! как скоро ночь минула!
Вчера просилась спать - отказ.
«Ждём друга». - Нужен глаз да глаз,
Не спи, покудова не скатишься со стула.
Теперь вот только что вздремнула,
Уж день!.. сказать им...

(Стучится к Софии.)

Господа,
Эй! Софья Павловна, б
```

Далее выполнен пример преобразования векторизации текста, который позволит понять, как именно модель взаимодействуе Выполнено разделение последовательностей и примеров, где каждая входная последовательность будет содержать символы из текста. Для каждой входной последовательности соотвествующие целевые объекты содержат текст одинаковой длины, исключая смещения на один символ вправо, например входная последовательность сначала обрабатывается K наборами сверточных фильтров с размерностью $1, 2, \ldots, K$. Эти фильтры моделируют локальную и контекстно-зависимую информацию (по аналогии с моделированием униграмм, биграмм, вплоть до K -грамм). Выход сверточного уровня далее обрабатывается шоссейной сетью для дальнейшего выделения параметров. В конце CBHG используется управляемый рекуррентный блок, который для выделения параметров опирается на контекст перед рассматриваемым символом и после рассматриваемого символа.

Для выполнения данной работы была использована функция tf_data.

```
all_ids = ids_from_chars(tf.strings.unicode_split(text, 'UTF-8'))
all_ids
```

Выполнено обучение модели, для этого было указано кол – ко эпох, соответствующее:

```
EPOCHS = 20
```

Запуск процесса обучения с использованием функции $.fit$

```
history = model.fit(dataset, epochs=EPOCHS, callbacks=[checkpoint_callback])
```

Вывод:

```
Epoch 1/20
172/172 [==============================] - 15s 63ms/step - loss: 2.7469
Epoch 2/20
172/172 [==============================] - 12s 60ms/step - loss: 2.0055
Epoch 3/20
172/172 [==============================] - 13s 63ms/step - loss: 1.7275
Epoch 4/20
172/172 [==============================] - 12s 62ms/step - loss: 1.5644
Epoch 5/20
172/172 [==============================] - 12s 62ms/step - loss: 1.4646
Epoch 6/20
172/172 [==============================] - 12s 61ms/step - loss: 1.3948
Epoch 7/20
172/172 [==============================] - 13s 60ms/step - loss: 1.3420
Epoch 8/20
172/172 [==============================] - 12s 61ms/step - loss: 1.2960
Epoch 9/20
172/172 [==============================] - 12s 61ms/step - loss: 1.2566
Epoch 10/20
172/172 [==============================] - 13s 62ms/step - loss: 1.2178
Epoch 11/20
172/172 [==============================] - 14s 62ms/step - loss: 1.1786
Epoch 12/20
172/172 [==============================] - 12s 62ms/step - loss: 1.1404
Epoch 13/20
172/172 [==============================] - 12s 61ms/step - loss: 1.0983
Epoch 14/20
172/172 [==============================] - 12s 61ms/step - loss: 1.0563
Epoch 15/20
172/172 [==============================] - 12s 60ms/step - loss: 1.0085
Epoch 16/20
172/172 [==============================] - 12s 61ms/step - loss: 0.9601
Epoch 17/20
172/172 [==============================] - 12s 61ms/step - loss: 0.9100
Epoch 18/20
```

```
172/172 [==============================] - 12s 61ms/step - loss: 0.8585
Epoch 19/20
172/172 [==============================] - 12s 62ms/step - loss: 0.8067
Epoch 20/20
172/172 [==============================] - 12s 61ms/step - loss: 0.7563
```

Исходя из описанных выше формул и полученных значений, в соответствии с рисунком 3 изображен график, соотвествующий *loss* функции относительно времени, построенный с использованием MathPlotLib:

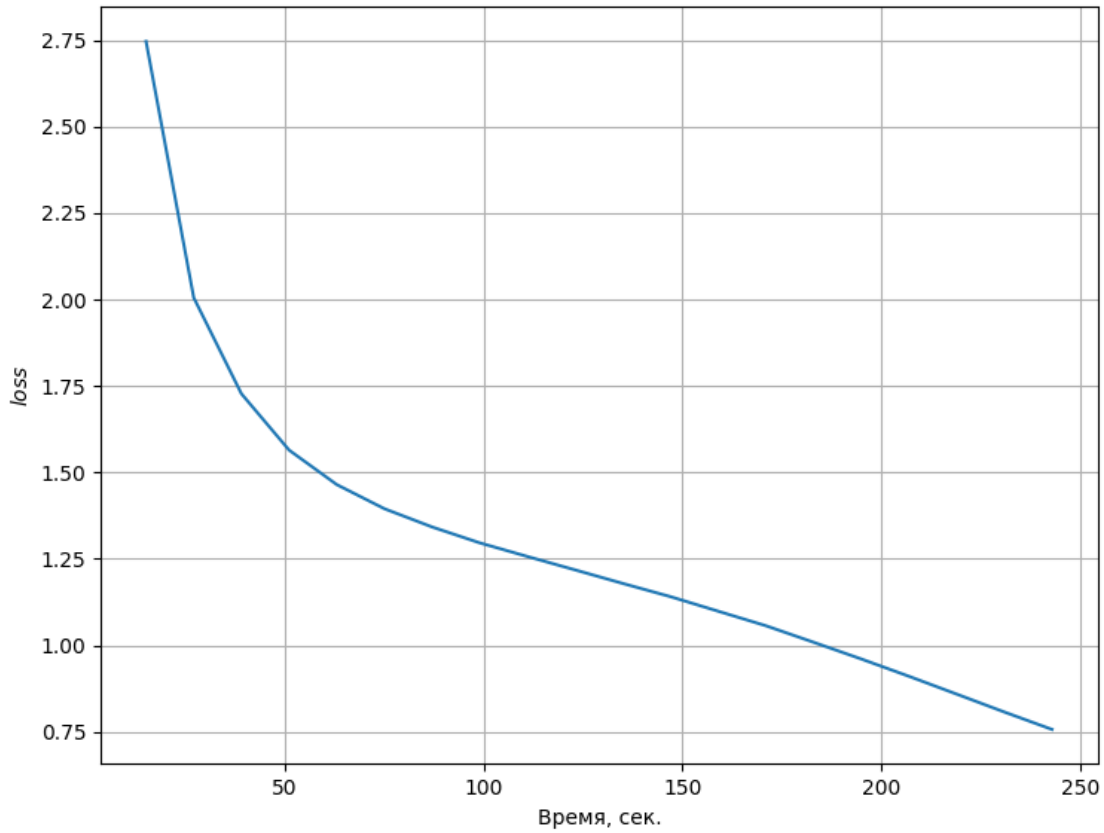

Рисунок 3 — График функции loss, относительно времени

Для этой модели самый простой способ сгенерировать текст – запустить его в цикле и отслеживать внутреннее состояние модели по мере ее выполнения.

```
Лиза
Вертелась перед ним, не помню что врала;
```

```
Ну что же стали вы? поклон, суда&#769;рь, отвесьте.
Осмелюсь я, сударь...

Фамусов
Все умудрились не по ле&#769;там.
А пуще дочери, да сами добряки,

София
Я гнева вашего никак не растолкую.
Он в доме здесь живёт, великая напасть!
И бросилась сюда я со всех ног...

Молчалин
Я только нёс их для докладу,
Что в ход нельзя пустить без справок, без иных,
мотри ты на меня: не хвастаю сложеньем,
Однако бодр и свеж, и дожил до седин;
--------------------------------------------------------------------------------

Run time: 3.7142245769500732
```

Исходя из представленных результатов, модель обучилась на исходном тексте и сгенерировала похожий текст, основываясь на представленных данных. В некоторых местах прослеживается логика построения предложений, а также стиль автора.

Заключение. В ходе проделанной работы были рассмотрены ключевые аспекты задачи о создании модели искусственного интеллекта для автодополнения и генерации текста. Была создана модель искусственного интеллекта, для автодополнения и генерации текста, с использованием технологий открытого исходного кода, с использованием программной библиотеки Tensorflow для Python 3.