ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.8

АДАПТИВНЫЕ УЧЕБНЫЕ МАТЕРИАЛЫ: ГЕНЕРАЦИЯ ЗАДАНИЙ С ПОМОЩЬЮ ДООБУЧЕННОЙ НЕЙРОННОЙ СЕТИ

Козленко Алексей Игоревич,
Магистрант, Донской государственный технический университет,
кафедра «Институт сквозных технологий»
344000, РФ, г. Ростов – на – Дону, пл. Гагарина 1
E-mail: a.cozlencko@yandex.ru
09.04.02 Интеллектуальные медиатехнологии

Витченко Ольга Викторовна,
Доцент, Донской государственный технический университет,
кафедра «Институт сквозных технологий»
344000, РФ, г. Ростов – на – Дону, пл. Гагарина 1
E-mail: owinf@mail.ru

Аннотация

В статье анализируется подход к реализации дообучения модели для создания персонализированных текстовых учебных заданий. Представлено, как использование глубоких нейросетей обеспечивает автоматизацию разработки учебных материалов, позволяя адаптировать их под потребности учащихся, а также повысить качество и оперативность этого процесса.

Ключевые слова: нейронные сети, автоматизация, текстовые учебные задания, глубокое обучение, образовательные технологии, трансформеры.

ADAPTIVE LEARNING MATERIALS: TASK GENERATION USING A FINE TUNED NEURAL NETWORK

Alexey I. Kozlenko,
Master's student, Don State Technical University,
Department of «Institute of End-to-End Technologies»
344000, Russian Federation, Rostov - on - Don, pl. Gagarina 1
E-mail: a.cozlencko@yandex.ru
09.04.02 Intellectual Media Technologies

Olga V. Vitchenko,
Associate Professor, Don State Technical University,
Department of «Institute of End-to-End Technologies»
344000, Russian Federation, Rostov - on - Don, pl. Gagarina 1
E-mail: owinf@mail.ru

ABSTRACT

The article analyzes the approach to the implementation of the model completion for the creation of personalized text learning tasks. It is presented how the use of deep neural networks provides automation of the development of educational materials, allowing them to be adapted to the needs of students, as well as to improve the quality and efficiency of this process.

Keywords: neural networks, automation, text-based learning tasks, deep learning, educational technologies, transformers.

Персонализация обучения становится все более востребованной, но при этом требует значительных усилий при создании индивидуализированных учебных материалов [1]. В этой статье мы предлагаем решение данной проблемы путем исследования возможности дообучения нейронных сетей для автоматической генерации текстовых учебных заданий. Мы покажем, как такой подход позволяет создавать качественные задания, адаптированные под конкретные потребности учащихся, и при этом значительно сокращает время, необходимое для их разработки.

Для дообучения нейросети генерируются экзаменационные задания в структурированном формате «контекст – задача – данные – ответ».

Этот формат выбран из-за:

1. Простоты: легко представлять и обрабатывать данные.
2. Аннотируемости: компоненты легко размечать и проверять.
3. Полноты: обеспечивает все необходимое для обучения и оценки.
4. Масштабируемости: легко добавлять задания без изменений.

Пред-обученная модель генератора. openai-community/gpt2 [2]. Для решения задачи генерации экзаменационных заданий в текстовой форме были рассмотрены 2 модели, доступные на платформе Hugging Face.

Для генерации экзаменационных заданий выбрана предобученная модель GPT2 (openai-community/gpt2) после сравнения с RuBERT (cointegrated/rubert-tiny) [3]. RuBERT показала ограниченные возможности в генерации структурированного текста, склонность к повторам и нелогичным высказываниям, в отличие от более гибкой GPT2, рисунки 1-2.

```
pipeline(f'{result_2} Исходная строка: {mask_token}.')

[{'score': 0.1714000552892685,
  'token': 30,
  'token_str': ':',
  'sequence': 'Строка состоит из символов 01. Определите максимальное количество идущих подряд символов 0. Исходная строка : :.'},
 {'score': 0.020674072206020355,
  'token': 23408,
  'token_str': '0000',
  'sequence': 'Строка состоит из символов 01. Определите максимальное количество идущих подряд символов 0. Исходная строка : 0000.'},
 {'score': 0.020265251398086548,
  'token': 12181,
  'token_str': '999',
  'sequence': 'Строка состоит из символов 01. Определите максимальное количество идущих подряд символов 0. Исходная строка : 999.'},
 {'score': 0.016715046018362045,
  'token': 20460,
  'token_str': '888',
  'sequence': 'Строка состоит из символов 01. Определите максимальное количество идущих подряд символов 0. Исходная строка : 888.'},
 {'score': 0.01303661148995161,
  'token': 13175,
  'token_str': '1111',
  'sequence': 'Строка состоит из символов 01. Определите максимальное количество идущих подряд символов 0. Исходная строка : 1111.'}]
```

Рисунок 1 – Визуальный анализ результатов генерации: rubert-tiny

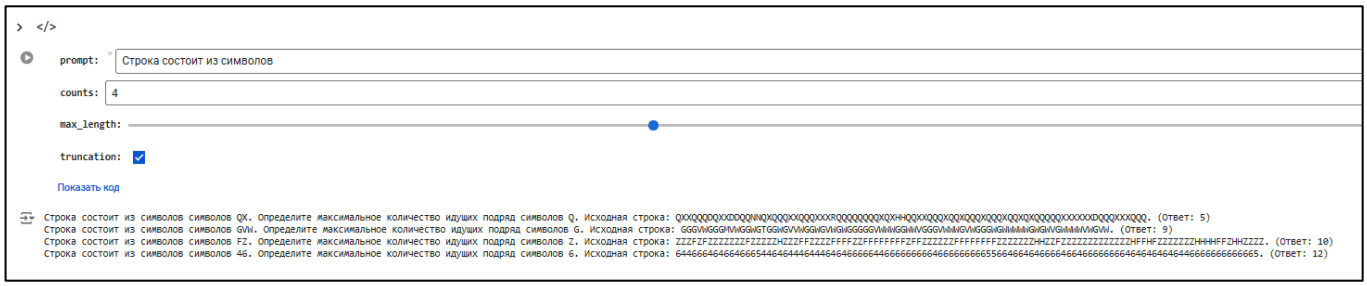


Рисунок 2 – Визуальный анализ результатов генерации: GPT-2

На основе визуального сравнения для дообучения выбрана модель GPT2, как оптимальная для генерации качественных экзаменационных заданий. Для успешного дообучения требуется качественный и разнообразный датасет. Далее будет проведен анализ существующих ресурсов, разработаны методы генерации синтетических данных, а также определены ограничения при формировании обучающего набора.

На первом этапе исследования проанализированы датасеты Hugging Face [4]. Поиск русскоязычных текстовых датасетов с заданиями выявил только «1-800-SHARED-TASKS/xlsum-subset», но он оказался непригодным (ориентирован на суммирование, а не генерацию заданий). Поэтому, в качестве основы для генерации синтетических данных, выбраны задания с сайта «СДАМ ГИА: РЕШУ ЕГЭ» [5], так как они являются корректными экзаменационными заданиями с решениями. Однако, из-за ограниченного количества данных потребуется разработка механизма генерации синтетических образцов, рисунки 3-5.

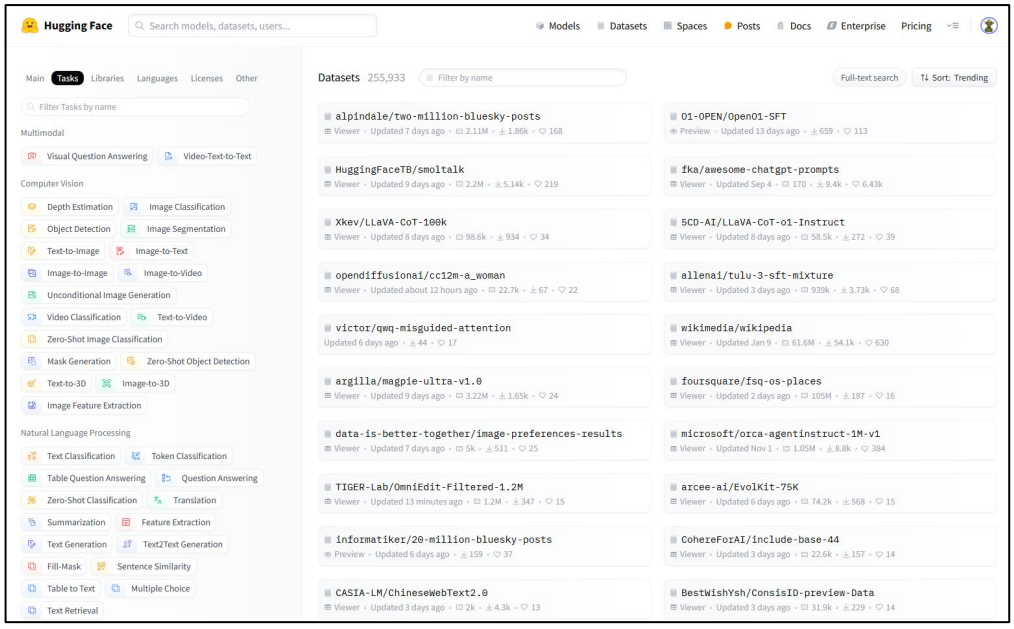


Рисунок 3 – Интерфейс поиска набора данных от Hugging Face

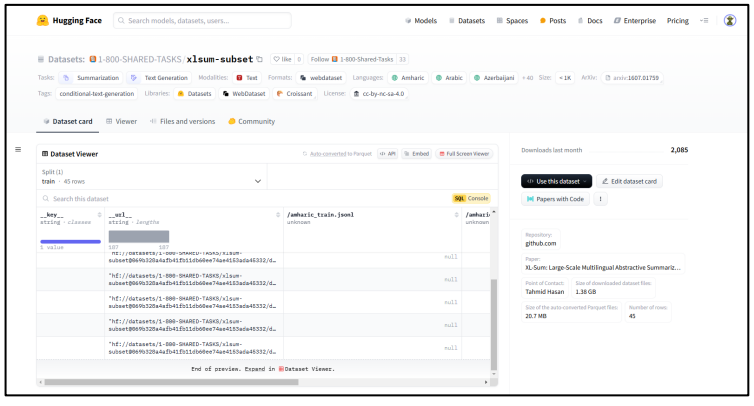
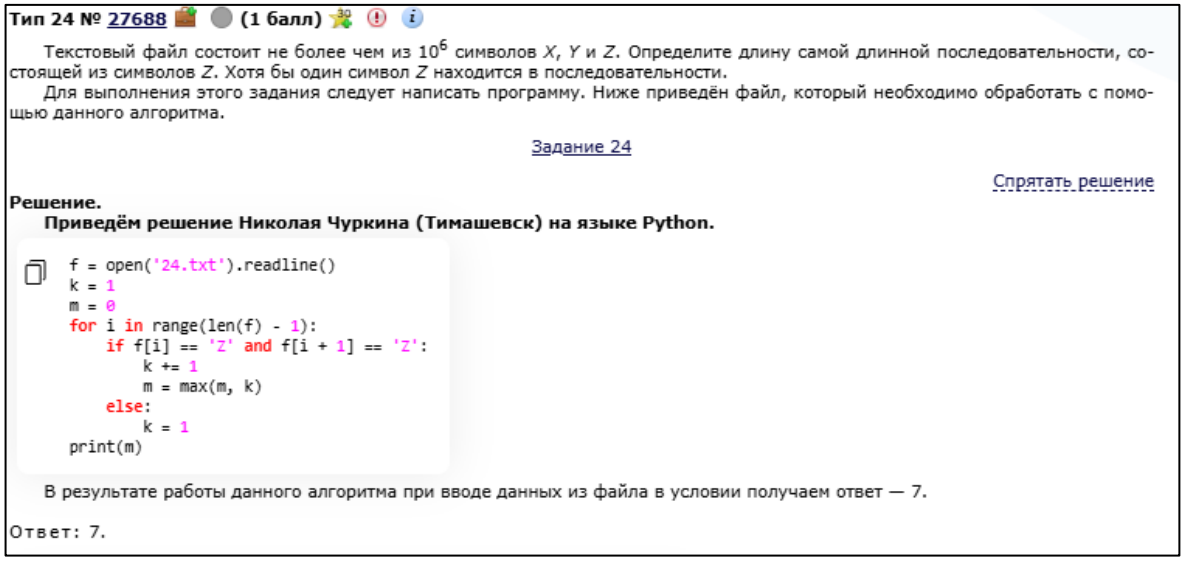


Рисунок 4 – Карточка датасета



Тип 24 № 27688 (1 балл)

Текстовый файл состоит не более чем из 10^6 символов *X*, *Y* и *Z*. Определите длину самой длинной последовательности, состоящей из символов *Z*. Хотя бы один символ *Z* находится в последовательности.

Для выполнения этого задания следует написать программу. Ниже приведён файл, который необходимо обработать с помощью данного алгоритма.

Задание 24

Спрятать решение

Решение.

Приведём решение Николая Чуркина (Тимашевск) на языке Python.

```
f = open('24.txt').readline()
k = 1
m = 0
for i in range(len(f) - 1):
    if f[i] == 'Z' and f[i + 1] == 'Z':
        k += 1
        m = max(m, k)
    else:
        k = 1
print(m)
```

В результате работы данного алгоритма при вводе данных из файла в условии получаем ответ — 7.

Ответ: 7.

Рисунок 5 – Решение задания на поиск подстрок

Разработан ноутбук в Google_Collab [6] для генерации синтетических данных (файл synthetic_data.txt), рисунки 6-11. Он использует функции для:

1. Генерации случайных последовательностей: generate_sequence создает последовательность символов на основе входного слова и набора допустимых символов, добавляя случайные символы и уменьшая повторы.
2. Определения максимального повторения: max_word_count считает сколько раз слово встречается подряд в последовательности.
3. Создания случайных последовательностей: random_sorted_letters генерирует отсортированные буквы, random_sorted_numbers - отсортированные числа, random_substring - подстроку, shuffle_lines - перемешивает строки.
4. Базовой генерации: generate_base создает строку из заданного набора символов, содержащую входное слово. Сгенерировано 10000 записей с различными типами данных (последовательности уникальных символов, двоичные строки, сложные структуры), записанных в выходной файл.

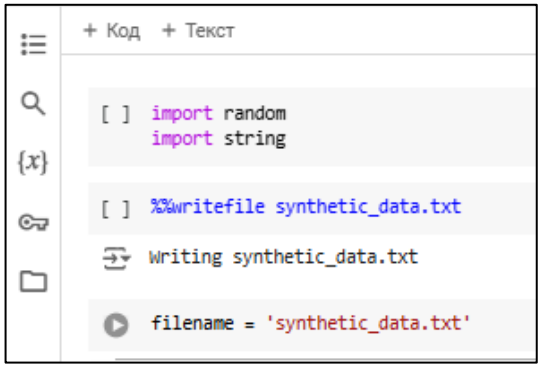


Рисунок 6 – Создание пустого файла для последующей записи

Задача

Структура генерируемых данных.

- Строка состоит из символов [01].
- Определите максимальное количество идущих подряд символов [1].
- Исходная строка: [1111111011110010100111010011111101111111011 10].
- (Ответ: 8)

Рисунок 7 – Структура генерируемых данных

```
[ ]  def generate_sequence(symbols, word, length):

         word_symbols = list(word)

         sequence = []
         for symbol in word_symbols:
             sequence.append(symbol)

         remaining_length = length - len(word)
         for _ in range(remaining_length):

             if random.random() < 0.4:

                 sequence.append(word)

             else:

                 sequence.append(random.choice(symbols))

         for i in range(len(sequence)-1):
             if random.random() < 0.2:
                 sequence[i+1] = sequence[i]

         return ''.join(sequence)[:length]

     def max_word_count(sequence, word):

       sequence = sequence.replace(word, '*')

       max_length = 0
       current_length = 0
       for char in sequence:
         if char == '*':
           current_length += 1
           max_length = max(max_length, current_length)
         else:
           current_length = 0

       return max_length
```

Рисунок 8 – Генерация строки символов и калькуляция ответа

```
[ ]  def random_sorted_letters(count):

       letters = random.sample(string.ascii_uppercase, count)

       return ''.join(sorted(letters))

[ ]  def random_sorted_numbers(count, digits):

       numbers = random.sample(range(digits), count)
       return ''.join(str(x) for x in sorted(numbers))

[ ]  def random_substring(string):

       while True:

         start = random.randint(0, len(string) - 1)
         end = start + random.randint(1, len(string) - start)

         if end - start != len(string):

           return string[start:end]

     def shuffle_lines(filename):

       with open(filename, 'r') as f:
         lines = f.readlines()

       random.shuffle(lines)

       with open(filename, 'w') as f:
         f.writelines(lines)
```

Рисунок 9 – Вспомогательные процедуры

```
[ ]  def generate_base(symbols):

       word     = random_substring(symbols)
       sequence  = generate_sequence(symbols, word, 100)

       return f'Строка состоит из символов [{symbols}]. Определите максимальное количество идущих подряд символов [{word}]. Исходная строка: [{sequence}]. (Ответ: {max_word_count(sequence, word)})'

[ ]  generate_base('01')

→  'Строка состоит из символов [01]. Определите максимальное количество идущих подряд символов [1]. Исходная строка: [1111111101111001010011101111111001100111001111111010000010110001011111111010100111111011111101110]. (Ответ: 8)'
```

Рисунок 10 – Процедура итоговой генерации: пример использования

```
[ ]  from tqdm import tqdm

     for i in tqdm(range(2500)):

       with open(filename, 'a') as f:

         # 1
         f.write(generate_base(random_sorted_letters(2)) + '\n')

         # 2
         f.write(generate_base(random_sorted_letters(3)) + '\n')

         # 3
         f.write(generate_base('01') + '\n')

         # 3
         f.write(generate_base('N#N') + '\n')

     shuffle_lines(filename)

→  100%|██████████| 2500/2500 [00:01<00:00, 2252.85it/s]
```

Рисунок 11 – Заполнение файла

Для дообучения создана вычислительная среда с необходимыми библиотеками (Transformers [7], datasets и др.), структурированным каталогом проекта, и корректно размещенным синтетическим датасетом. Процесс задокументирован и воспроизводим, рисунки 12-18. Основные разделы:

1. Полезные ресурсы: ресурсы для работы с моделями.
2. Установка библиотек: torch, accelerate, transformers для обучения.

3. Загрузка/обработка данных: директории, данные (train_file.txt), функции load_dataset и load_data_collator для подготовки данных.
4. Дообучение: параметры обучения (оптимизатор, эпохи, батч) и запуск обучения GPT-2 с помощью Trainer.
5. Использование: Графический интерфейс для генерации текста с заданными параметрами (тип символов, длина, обрезка), извлекает подстроку и выводит результат. Предназначен для оценки модели.

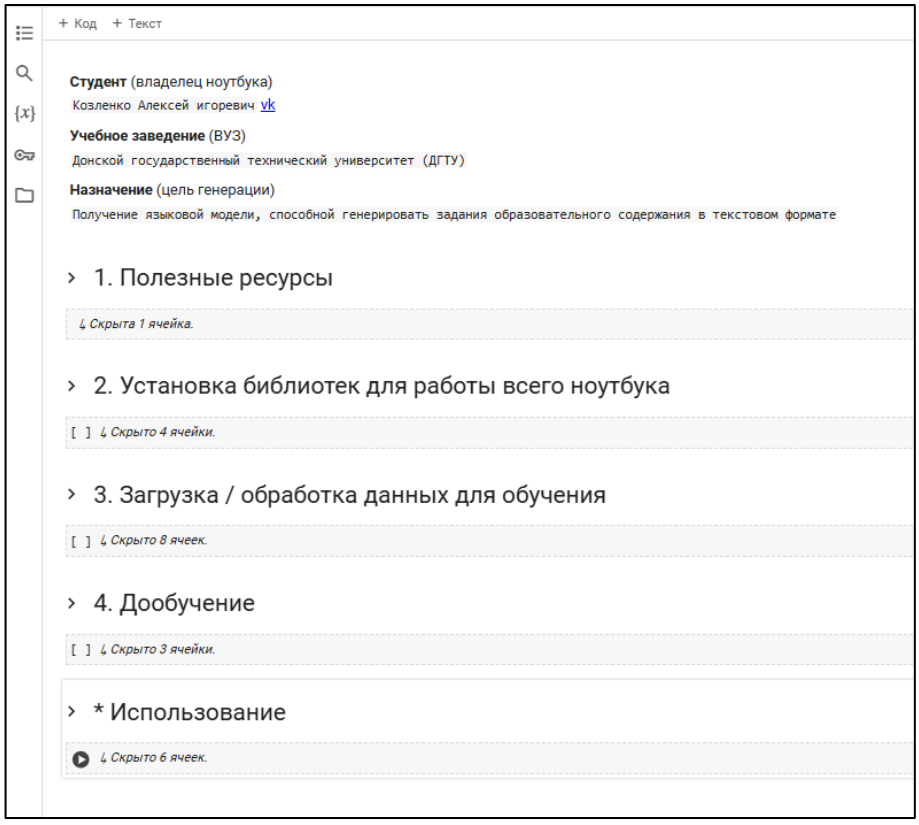


Рисунок 12 – Структура ноутбука для дообучения модели

Рисунок 13 – Полезные ресурсы

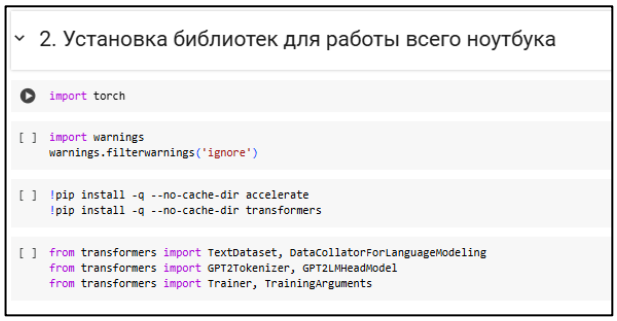


Рисунок 14 – Библиотеки

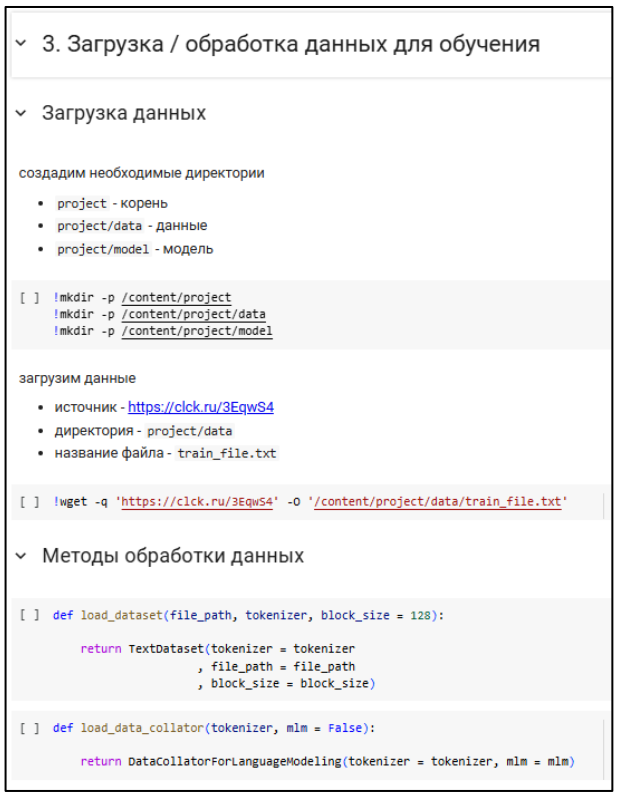
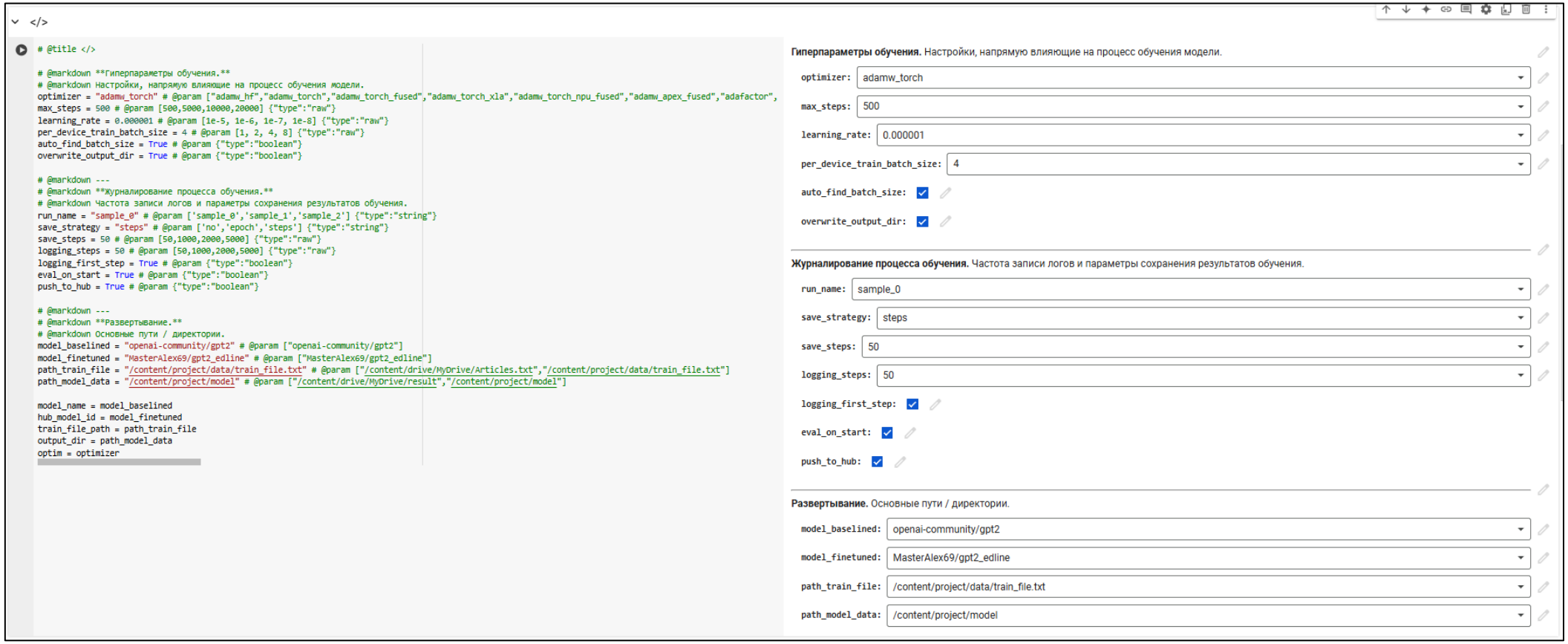


Рисунок 15 – Загрузка данных


@title </>

# @markdown **Гиперпараметры обучения.**
# @markdown Настройки, напрямую влияющие на процесс обучения модели.
optimizer = "adamw_torch" # @param ["adamw_hf","adamw_torch","adamw_torch_fused","adamw_torch_xla","adamw_torch_npu_fused","adamw_apex_fused","adafactor",
max_steps = 500 # @param [500,5000,10000,20000] {"type":"raw"}
learning_rate = 0.000001 # @param [1e-5, 1e-6, 1e-7, 1e-8] {"type":"raw"}
per_device_train_batch_size = 4 # @param [1, 2, 4, 8] {"type":"raw"}
auto_find_batch_size = True # @param {"type":"boolean"}
overwrite_output_dir = True # @param {"type":"boolean"}

# @markdown ---
# @markdown **Журналирование процесса обучения.**
# @markdown Частота записи логов и параметры сохранения результатов обучения.
run_name = "sample_0" # @param ['sample_0','sample_1','sample_2'] {"type":"string"}
save_strategy = "steps" # @param ['no','epoch','steps'] {"type":"string"}
save_steps = 50 # @param [50,1000,2000,5000] {"type":"raw"}
logging_steps = 50 # @param [50,1000,2000,5000] {"type":"raw"}
logging_first_step = True # @param {"type":"boolean"}
eval_on_start = True # @param {"type":"boolean"}
push_to_hub = True # @param {"type":"boolean"}

# @markdown ---
# @markdown **Развертывание.**
# @markdown Основные пути / директории.
model_baselined = "openai-community/gpt2" # @param ["openai-community/gpt2"]
model_finetuned = "MasterAlex69/gpt2_edline" # @param ["MasterAlex69/gpt2_edline"]
path_train_file = "/content/project/data/train_file.txt" # @param ["/content/drive/MyDrive/Articles.txt","/content/project/data/train_file.txt"]
path_model_data = "/content/project/model" # @param ["/content/drive/MyDrive/result","/content/project/model"]

model_name = model_baselined
hub_model_id = model_finetuned
train_file_path = path_train_file
output_dir = path_model_data
optim = optimizer

Гиперпараметры обучения. Настройки, напрямую влияющие на процесс обучения модели.
optimizer: adamw_torch
max_steps: 500
learning_rate: 0.000001
per_device_train_batch_size: 4
auto_find_batch_size:
overwrite_output_dir:
Журналирование процесса обучения. Частота записи логов и параметры сохранения результатов обучения.
run_name: sample_0
save_strategy: steps
save_steps: 50
logging_steps: 50
logging_first_step:
eval_on_start:
push_to_hub:
Развертывание. Основные пути / директории.
model_baselined: openai-community/gpt2
model_finetuned: MasterAlex69/gpt2_edline
path_train_file: /content/project/data/train_file.txt
path_model_data: /content/project/model

Рисунок 16 – Дообучение: настройка параметров

```python
def train_edline_model(save = False):

    # Инициализация и токенизация.
    # Загрузка предобученного токенизатора и модели GPT-2, токенизация тренировочных данных и сохранение токенизатора.
    tokenizer     = GPT2Tokenizer.from_pretrained(model_name)
    train_dataset = load_dataset(train_file_path, tokenizer)
    data_collator = load_data_collator(tokenizer)

    tokenizer.save_pretrained(output_dir)

    model = GPT2LMHeadModel.from_pretrained(model_name)
    model.save_pretrained(output_dir)

    # Аргументы тренировки.
    # Настройка параметров тренировки модели, таких как размер батча и количество эпох, путь сохранения и т.д.
    training_args = TrainingArguments(output_dir                    = output_dir
                                    , overwrite_output_dir          = overwrite_output_dir
                                    , per_device_train_batch_size   = per_device_train_batch_size
                                    , run_name                      = run_name
                                    , save_strategy                 = save_strategy
                                    , logging_steps                 = logging_steps
                                    , save_steps                    = save_steps
                                    , max_steps                     = max_steps
                                    , learning_rate                 = learning_rate
                                    , logging_first_step            = logging_first_step
                                    , auto_find_batch_size          = auto_find_batch_size
                                    , push_to_hub                   = push_to_hub
                                    , hub_model_id                  = hub_model_id
                                    , optim                         = optim
                                    , eval_on_start                 = eval_on_start)

    # Инициализация тренера.
    # Создание объекта Trainer для управления процессом тренировки, используя загруженную модель, данные и параметры.
    trainer = Trainer(model           = model
                    , args            = training_args
                    , data_collator   = data_collator
                    , train_dataset   = train_dataset)

    # Тренировка и сохранение.
    # Запуск тренировки модели и сохранение обученной модели.
    trainer.train()
    if save: trainer.save_model()
```

```python
train_edline_model(save = False)
```

Рисунок 17 – Дообучение: тренировка модели

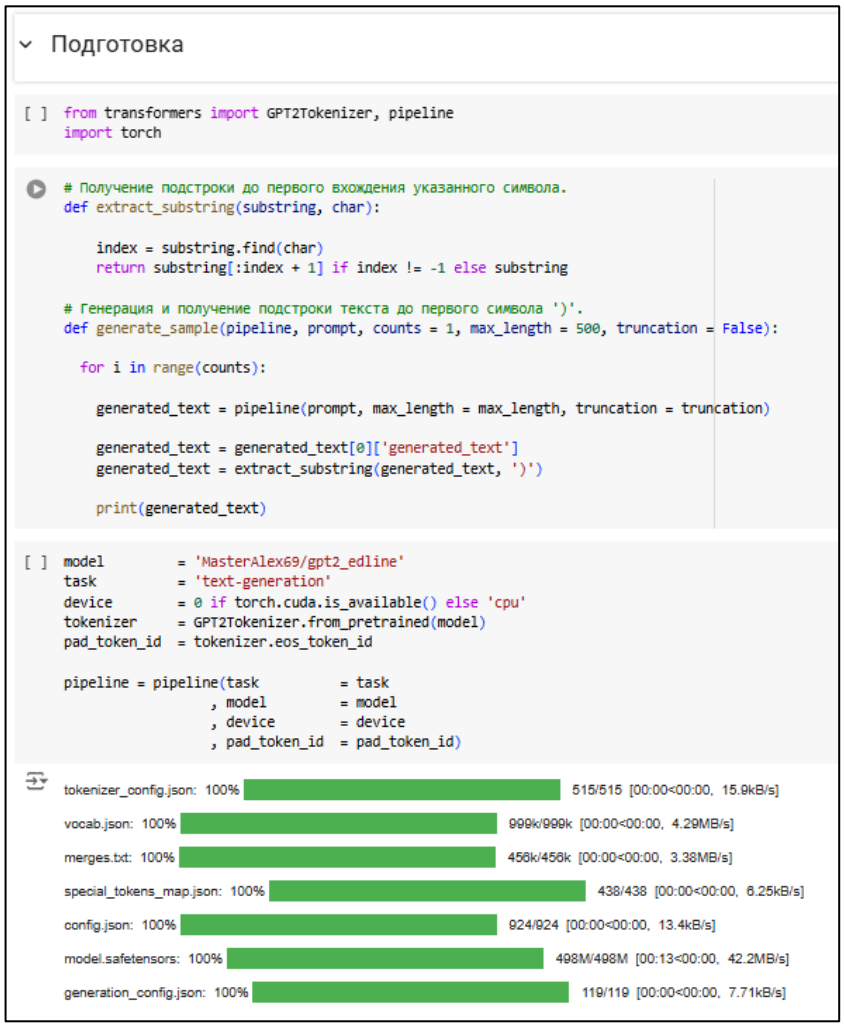


Рисунок 18 – Использование: подготовка / загрузка модели

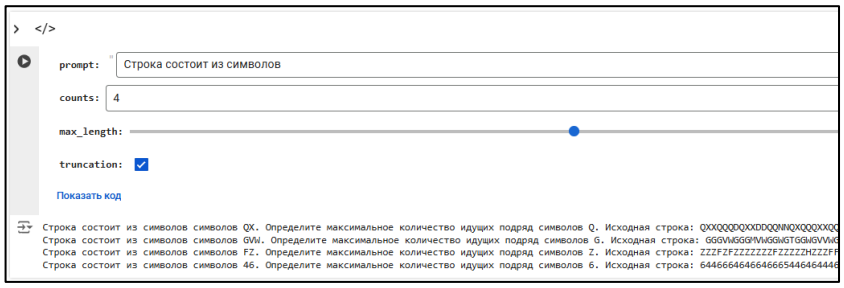


Рисунок 19 – Использование: результаты генерации

Дообученная модель GPT-2 успешно генерирует корректные, но однотипные и простые задачи по информатике на анализ строк. Примеры с запросом «Строка состоит из символов» продемонстрировали способность модели создавать задачи, где требуется найти максимальную последовательность повторяющихся символов.

Недостатки:

1. Повторяемость: Задачи однотипны, не хватает разнообразия в формулировках и проверяемых навыках.
2. Простота: Задачи подходят для начального уровня, необходимо усложнение (более длинные строки, больше символов, доп. условия).

Для улучшения генерации необходимо:

1. Расширить разнообразие типов задач.
2. Усложнить условия задач.

3. Уточнить формулировки.

В целом, модель демонстрирует потенциал для генерации учебных заданий, но требует дальнейшего совершенствования для достижения более высокого качества и разнообразия.

Список литературы:

1. Sheitanova N. Искусственный интеллект в обучении иностранным языкам: возможности ChatGPT в создании учебных текстов //Проблемы когнитивного и функционального описания русского и болгарского языков. – 2024. – Т. 18. – №. 3. – С. 80-99.
2. Lagutina N. S., Lagutina K. V., Kopnin V. N. Automatic Determination of Semantic Similarity of Student Answers with the Standard One Using Modern Models //MODELING AND ANALYSIS OF INFORMATION SYSTEMS. – 2024. – С. 194.
3. Gomez J. F. et al. Algorithmic Arbitrariness in Content Moderation //The 2024 ACM Conference on Fairness, Accountability, and Transparency. – 2024. – С. 2234-2253.
4. Shen Y. et al. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face //Advances in Neural Information Processing Systems. – 2024. – Т. 36.
5. Борчашвили Ф. Т., Болтаева Л. Ш. ЭЛЕКТРОННЫЕ ОБРАЗОВАТЕЛЬНЫЕ ПЛАТФОРМЫ КАК ФУНДАМЕНТ СОВРЕМЕННОГО ШКОЛЬНОГО ОБРАЗОВАНИЯ //Автор проекта и научный редактор. – 2024. – С. 33.
6. Seebut S., Wongsason P., Kim D. Combining GPT and Colab as learning tools for students to explore the numerical solutions of difference equations //Eurasia Journal of Mathematics, Science and Technology Education. – 2024. – Т. 20. – №. 1. – С. em2377.
7. Islam S. et al. A comprehensive survey on applications of transformers for deep learning tasks //Expert Systems with Applications. – 2024. – Т. 241. – С. 122666.

References:

1. Sheitanova N. State intelligence in learning a foreign language: the ability to communicate using educational texts //Problems of cognitive and encyclopedic description of the Russian and Bulgarian languages. – 2024. – Vol. 18. – No. 3. – pp. 80-99.
2. Lagutina N. S., Lagutina K. V., Kopnin V. N. Automatic determination of the semantic similarity of students' answers to the standard using modern models //MODELING AND ANALYSIS OF INFORMATION SYSTEMS. – 2024. – P. 194.
3. Gomez J. F. et al. Algorithmic arbitrariness in content moderation //ACM 2024 Conference on Equity, Accountability and Transparency. - 2024. – pp. 2234-2253.
4. Shen Yu. and others Hugginggpt: Solving artificial intelligence problems using chatgpt and his friends in hugging face //Achievements in the field of neural information processing systems. – 2024. – Vol. 36.
5. Borchashvili F. T., Boltayeva L. S. ELECTRONIC EDUCATIONAL PLATFORMS AS THE FOUNDATION OF MODERN SCHOOL EDUCATION //The author of the project and the scientific editor. – 2024. – p. 33.

6. Sibut S., Wongsason P., Kim D. The combination of GPT and Colab as tools for teaching student's numerical solutions of difference equations //Eurasia Journal of Mathematics, Science and Technology Education. – 2024. – Vol. 20. – No. 1. – p. 2377.
7. Islam S. et al. A comprehensive overview of the use of transformers for deep learning tasks //Expert systems with applications. – 2024. – vol. 241. – p. 122666.