Черновик.

Дообучение GPT моделей.

# Общие данные по GPT моделям

GPT - **Generative pre-trained transformer,** класс нейронных сетей основная задача которых угадывать следующее слово в тексте. Как следует из названия – эти сети это прошли обучение, их нужно тренировать под конкретную задачу

Описание процесса:

1. Берем существующую нейронную сеть из открытого доступа
2. Проводим обучение на тренировочных данных, релевантных поставленной задачи

Инструментальные средства:

1. Python c пакетами: torch, transformers
2. Видеокарта с поддержкой Nvidia Cuda

# Эксперимент

Загружаем модули

from transformers import GPT2LMHeadModel, GPT2Tokenizer, TextDataset, DataCollatorForLanguageModeling, Trainer, TrainingArguments  
import torch  
DEVICE = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

Сохраняем обучающий датасет в файл. Датасет межит в переменной **text,** любой текст релевантный решаемой задаче. Лучше так иначе на винде будут проблемы с кодировкой и ничего не заработает

train\_path = 'train\_dataset.txt'  
with open(train\_path, "w", encoding="utf-8") as f:  
 f.write(text)

Берем основную пред обученную модель, с [**Hugging Face**](https://huggingface.co/)

model\_name\_or\_path = "sberbank-ai/rugpt3large\_based\_on\_gpt2"  
tokenizer = GPT2Tokenizer.from\_pretrained(model\_name\_or\_path)  
model = GPT2LMHeadModel.from\_pretrained(model\_name\_or\_path).to(DEVICE)

Создаем датасет

train\_dataset = TextDataset(tokenizer=tokenizer, file\_path=train\_path, block\_size=64)  
if ( 0 == len(train\_dataset) ):  
 raise Exception("Sorry, dataset is empty")  
data\_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)

Задаем параметры обучения

training\_args = TrainingArguments(  
 output\_dir="./finetuned", # The output directory  
 overwrite\_output\_dir=True, # Overwrite the content of the output dir  
 num\_train\_epochs=200 , # number of training epochs  
 per\_device\_train\_batch\_size=32, # batch size for training  
 per\_device\_eval\_batch\_size=32, # batch size for evaluation  
 warmup\_steps=10, # number of warmup steps for learning rate scheduler  
 gradient\_accumulation\_steps=16, # to make "virtual" batch size larger  
 )

Создаем учителя, который выполнит всю магию

trainer = Trainer(  
 model=model,  
 args=training\_args,  
 data\_collator=data\_collator,  
 train\_dataset=train\_dataset,  
 optimizers = (torch.optim.AdamW(model.parameters(),lr=1e-5), None)  
)

Стартуем процесс обучения, на RTX 2060Super процесс обучения с использованием генетического алгоритма на 200 эпох, занимает примерно 20 минут

trainer.train()

Задаем начальную фразу текста, Учили на стихотворении Есенина «Письмо женщине»

text = "Странное задание - писать манул о GPT\nКогда на хабре все разжеванно\n"  
input\_ids = tokenizer.encode(text, return\_tensors="pt").to(DEVICE)

Генерируем продолжение фразы из 100 символов

model.eval()  
with torch.no\_grad():  
 out = model.generate(input\_ids,  
 do\_sample=True,  
 num\_beams=2,  
 temperature=1.3,  
 top\_p=0.9,  
 max\_length=100,  
 )  
  
generated\_text = list(map(tokenizer.decode, out))[0]  
print(generated\_text)

Сохраняем модель для дальнейшего использования

model.save\_pretrained("e1\_model", from\_pt=True)  
trainer.save\_model("e1\_trained")

Смотрим результат:

# Список литературы

1. <https://habr.com/ru/articles/599673/>
2. <https://habr.com/ru/articles/261421/>
3. <https://habr.com/ru/articles/334380/>
4. <https://colab.research.google.com/drive/1U7SX7jNYsNQG5BY1xEQQHu48Pn6Vgnyt?usp=sharing>
5. <https://telin.ugent.be/telin-docs/windows/pytorch/>