review_generator_v3

December 23, 2022

0.1 Finetuning GPT-2 to generate reviews

[22]:
```
import pandas as pd
from transformers import GPT2LMHeadModel, GPT2Tokenizer
import numpy as np
import random
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import GPT2Tokenizer, GPT2LMHeadModel, AdamW,␣
 ↪get_linear_schedule_with_warmup
from tqdm import tqdm, trange
import torch.nn.functional as F
import csv
# from langdetect import detect
```

Reading in all the data and preparing it

[23]:
```
reviews = pd.read_csv("../tripadvisor_dataset/reviews.csv")
reviews = reviews.applymap(str) # convert to string because there are some rows␣
 ↪with float
```

We extract the English reviews because gpt is an English model

Code can be found here: https://www.kaggle.com/hikmatelhaj/extracting-english-reviews

[24]:
```
reviews = pd.read_csv("reviews_en.csv")
reviews["rating"].value_counts()
```

[24]:
```
5.0    12647
4.0     8595
3.0     2744
1.0     1486
2.0     1351
Name: rating, dtype: int64
```

We take 1000 5-star and 1000 1-star reviews to finetune the model. We don't use more data because it takes a while to finetune

[25]:
```
reviews = pd.concat([reviews.query("rating == 1.0").sample(1000),reviews.
↪query("rating == 5.0").sample(1000)])
reviews = reviews.reset_index() # reset the indices after sampling
```

[27]:
```
reviews
```

[27]:

	index	id	reviewer name	title	date	review	rating
0	25933	4431449	Madelon_B70	Terrible disappointment	May 24, 2016	Based on a recommendation about the location a…	1.0
1	22227	10643135	224mariias	Order your coffee someplace else	January 11, 2020	Yesterday came here and ordered three coffees…	1.0
2	26660	784382	Babslalala	Terrible!	December 8, 2018	I usualy don’t give negative reviews but this …	1.0
3	22993	6878696	boblecostaud	Avoid	August 29, 2019	Customer service does not exist here. We polit…	1.0
4	9819	3676410	Gregor B	Terrible food and service	April 3, 2016	We came here to have dinner and watch a footba…	1.0
…	…	…	…	…	…	…	…
1995	8117	1058490	paratanytarsus	Fabulous Fondue	May 6, 2014	We returned to this restaurant with son Paul a…	5.0
1996	1665	967590	A Tripadvisor reviewer on Facebook	verry nice food	June 11, 2008	verry nice food	5.0
1997	1301	740670	ppaulmm	A jewel of a restaurant in a jewel of a city	September 12, 2016	We visited Ghent for a few days - a great city…	5.0
1998	8345	1894818	Caroline026	Cosy place with excellent food	July 28, 2016	As soon as you are seated in this cosy restaur…	5.0
1999	6175	15194988	AngeloP2608	Check this out!	October 15, 2018	As babybrother from Oak, this restaurant is re…	5.0

[2000 rows x 7 columns]

We create a dataset which tokenizes the reviews. We also limit the review length to 1024 tokens in case a review is longer

[7]:
```
class ReviewData(Dataset):
    def __init__(self, control_code, gpt2_type="gpt2", max_length=1024):
        self.tokenizer = GPT2Tokenizer.from_pretrained(gpt2_type)
        self.revs = []
        for row in reviews['review']:
            self.revs.append(torch.tensor(
                self.tokenizer.encode(f"<|{control_code}|>{row[:
 ↪max_length]}<|endoftext|>")
            ))
        self.rev_count = len(self.revs)

    def __len__(self):
        return self.rev_count

    def __getitem__(self, item):
        return self.revs[item]
```

[8]:
```
dataset = ReviewData(reviews['review'], gpt2_type="gpt2")
```

```
Downloading:   0%|          | 0.00/1.04M [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/456k [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/665 [00:00<?, ?B/s]
```

[9]:
```
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = GPT2LMHeadModel.from_pretrained('gpt2')
```

```
Downloading:   0%|          | 0.00/548M [00:00<?, ?B/s]
```

GPT is a huge model, to limit the calculation. Before performing a gradient descent step, it'll sum up all the gradients of several operations. Then it will divide that sum by the number of accumulated steps, to get an average loss over the training sample.

[10]:
```
#Accumulated batch size

def pack_tensor(new_tensor, packed_tensor, max_seq_len):
    if packed_tensor is None:
        return new_tensor, True, None
    if new_tensor.size()[1] + packed_tensor.size()[1] > max_seq_len:
        return packed_tensor, False, new_tensor
    else:
        packed_tensor = torch.cat([new_tensor, packed_tensor[:, 1:]], dim=1)
        return packed_tensor, True, None
```

0.2 Training the model

The hyperparameters to tune are learning rate, batch size, epochs and optimizer (ADAM)

[11]:
```
import os

def train(
    dataset, model, tokenizer,
    batch_size=16, epochs=10, lr=2e-5,
    max_seq_len=400, warmup_steps=200,
    gpt2_type="gpt2", output_dir=".", output_prefix="wreckgar",
    test_mode=False, save_model_on_epoch=False,
):

    acc_steps = 100
    device=torch.device("cuda")
    model = model.cuda()
    model.train()

    optimizer = AdamW(model.parameters(), lr=lr)
    scheduler = get_linear_schedule_with_warmup(
        optimizer, num_warmup_steps=warmup_steps, num_training_steps=-1
    )

    train_dataloader = DataLoader(dataset, batch_size=1, shuffle=True)
    loss=0
    accumulating_batch_count = 0
    input_tensor = None

    for epoch in range(epochs):

        print(f"Training epoch {epoch}")
        print(loss)
        for idx, entry in tqdm(enumerate(train_dataloader)):
            (input_tensor, carry_on, remainder) = pack_tensor(entry,␣
 ↪input_tensor, 768)

            if carry_on and idx != len(train_dataloader) - 1:
                continue

            input_tensor = input_tensor.to(device)
            outputs = model(input_tensor, labels=input_tensor)
            loss = outputs[0]
            loss.backward()

            if (accumulating_batch_count % batch_size) == 0:
                optimizer.step()
                scheduler.step()
```

```
                optimizer.zero_grad()
                model.zero_grad()

            accumulating_batch_count += 1
            input_tensor = None
        if save_model_on_epoch:
            torch.save(
                model.state_dict(),
                os.path.join(output_dir, f"{output_prefix}-{epoch}.pt"),
            )
    return model
```

[12]:
```
model = train(dataset, model, tokenizer)
```

```
/opt/conda/lib/python3.10/site-packages/transformers/optimization.py:306:
FutureWarning: This implementation of AdamW is deprecated and will be removed in
a future version. Use the PyTorch implementation torch.optim.AdamW instead, or
set `no_deprecation_warning=True` to disable this warning
  warnings.warn(

Training epoch 0
0

2000it [01:20, 24.83it/s]

Training epoch 1
tensor(2.7957, device='cuda:0', grad_fn=<NllLossBackward0>)

2000it [01:20, 24.72it/s]

Training epoch 2
tensor(1.1757, device='cuda:0', grad_fn=<NllLossBackward0>)

2000it [01:22, 24.13it/s]

Training epoch 3
tensor(1.0292, device='cuda:0', grad_fn=<NllLossBackward0>)

2000it [01:20, 24.72it/s]

Training epoch 4
tensor(1.3510, device='cuda:0', grad_fn=<NllLossBackward0>)

2000it [01:21, 24.58it/s]

Training epoch 5
tensor(0.8160, device='cuda:0', grad_fn=<NllLossBackward0>)

2000it [01:24, 23.77it/s]

Training epoch 6
tensor(1.4777, device='cuda:0', grad_fn=<NllLossBackward0>)

2000it [01:21, 24.62it/s]
```

```
Training epoch 7
tensor(0.7970, device='cuda:0', grad_fn=<NllLossBackward0>)

2000it [01:20, 24.70it/s]

Training epoch 8
tensor(0.6082, device='cuda:0', grad_fn=<NllLossBackward0>)

2000it [01:22, 24.11it/s]

Training epoch 9
tensor(0.9392, device='cuda:0', grad_fn=<NllLossBackward0>)

2000it [01:21, 24.49it/s]
```

0.3 Generating

Now generating the text has some parameters.

entry_count: The amount of times to generate

entry_length: Maximum amount of words to generate

top_p: The minimum probablity to filter possible outcomes. An example:

```
Probability as sequence of this sentence: it's hot ...
1% today
2% yesterday
3% tomorrow
...

This will be converted to
1% today
3% today, yeserday
6% today, yesterday, tomorrow
...

If we now set the top_p to 1%, then the chance that the word 'today' get's chosen is higher tha
```

temperature: The same parameter that we used in our previous model

[57]:
```python
def generate(
    model,
    tokenizer,
    prompt,
    entry_count=10,
    entry_length=30,
    top_p=0.8,
    temperature=0.5,
):
    model.eval()
    generated_num = 0
    generated_list = []
```

```
    # print(f"temperature is {temperature}")
    filter_value = -float("Inf")

    with torch.no_grad():

        for entry_idx in trange(entry_count):

            entry_finished = False
            generated = torch.tensor(tokenizer.encode(prompt)).unsqueeze(0)

            for i in range(entry_length):
                outputs = model(generated, labels=generated)
                loss, logits = outputs[:2]
                logits = logits[:, -1, :] / (temperature if temperature > 0␣
↪else 1.0)

                sorted_logits, sorted_indices = torch.sort(logits,␣
↪descending=True)
                cumulative_probs = torch.cumsum(F.softmax(sorted_logits,␣
↪dim=-1), dim=-1)

                sorted_indices_to_remove = cumulative_probs > top_p
                sorted_indices_to_remove[..., 1:] = sorted_indices_to_remove[
                    ..., :-1
                ].clone()
                sorted_indices_to_remove[..., 0] = 0

                indices_to_remove = sorted_indices[sorted_indices_to_remove]
                logits[:, indices_to_remove] = filter_value

                next_token = torch.multinomial(F.softmax(logits, dim=-1),␣
↪num_samples=1)
                generated = torch.cat((generated, next_token), dim=1)

                if next_token in tokenizer.encode("<|endoftext|>"):
                    entry_finished = True

                if entry_finished:

                    generated_num = generated_num + 1

                    output_list = list(generated.squeeze().numpy())
                    output_text = tokenizer.decode(output_list)
                    generated_list.append(output_text)
                    break

            if not entry_finished:
```

```
            output_list = list(generated.squeeze().numpy())
            output_text = f"{tokenizer.decode(output_list)}<|endoftext|>"
            generated_list.append(output_text)

    return generated_list
```

```
100%|      | 1/1 [00:08<00:00,  8.56s/it]
```

[46]:
```
def text_generation_tekst_meegeven(text, temperature=0.5):
    x = generate(model.to('cpu'), tokenizer, text, entry_count=1,␣
 ↪temperature=temperature)
    return x
```

We first try to generate positive reviews

[32]:
```
text_generation_tekst_meegeven("the food was delicious")
```

```
100%|      | 1/1 [00:06<00:00,  6.47s/it]
```

```
['the food was delicious and the service was great. The only thing I would
change is the menu. I would definitely recommend this place
again.<|endoftext|>']
```

[32]:
```
[['the food was delicious and the service was great. The only thing I would
change is the menu. I would definitely recommend this place
again.<|endoftext|>']]
```

[34]:
```
text_generation_tekst_meegeven("We had a great time")
```

```
100%|      | 1/1 [00:01<00:00,  1.70s/it]
```

```
["We had a great time and we'll be back next year.<|endoftext|>"]
```

[34]:
```
[["We had a great time and we'll be back next year.<|endoftext|>"]]
```

Generating positive reviews seems to work great, now let's test the negative reviews.

[54]:
```
text_generation_tekst_meegeven("too expensive food and meals", 0.8)
```

```
temperature is 0.8
```

```
100%|      | 1/1 [00:07<00:00,  7.97s/it]
```

```
['too expensive food and meals were cooked in the house.\n\nDespite this, the
family was able to get some lovely restaurants that would sell you their meals
in their place.<|endoftext|>']
```

[54]:
```
[['too expensive food and meals were cooked in the house.\n\nDespite this, the
family was able to get some lovely restaurants that would sell you their meals
in their place.<|endoftext|>']]
```

[55]:
```
text_generation_tekst_meegeven("bad food ")
```

```
temperature is 0.5

100%|      | 1/1 [00:09<00:00,  9.44s/it]

['bad food       I went with a group of friends and the service…\n1 person
found this review helpful.\n\nReviewed By Date Rating<|endoftext|>']
```

[55]:
```
[['bad food       I went with a group of friends and the service…\n1 person
found this review helpful.\n\nReviewed By Date Rating<|endoftext|>']]
```

[56]:
```
text_generation_tekst_meegeven("dry food and too salty ")
```

```
temperature is 0.5

100%|      | 1/1 [00:08<00:00,  8.37s/it]

['dry food and too salty \xa0for me.\nI was very disappointed with the quality
of the food. The only thing I could think of was that it was not a
good<|endoftext|>']
```

[56]:
```
[['dry food and too salty \xa0for me.\nI was very disappointed with the quality
of the food. The only thing I could think of was that it was not a
good<|endoftext|>']]
```

[58]:
```
text_generation_tekst_meegeven("impolite staff")
```

```
100%|      | 1/1 [00:07<00:00,  7.56s/it]

['impolite staff, who were all in the building.\n\n"I was really upset and
scared, I was really upset," said one woman. "I was<|endoftext|>']
```

[58]:
```
[['impolite staff, who were all in the building.\n\n"I was really upset and
scared, I was really upset," said one woman. "I was<|endoftext|>']]
```

The negative reviews are much harder to generate, even though we have as much positive as negative reviews. To explain this we take a deeper look at the negative reviews. We save it to a csv and we scroll through the negative reviews.

[19]:
```
reviews.where(reviews.rating == 1.0).dropna()["review"].to_csv("bad.csv",␣
↪index=False)
```

```
reviews.where(reviews.rating == 5.0).dropna()["review"].to_csv("positive.csv",␣
 ↪index=False)
```

[20]:
```
reviews.where(reviews.rating == 1.0)["review"].head(5)
```

[20]:
```
0    Had the ribs…terrible. \nImagine chewing on…
1    Ate there twice in ten years. The first time m…
2    Okay. Let me start by saying that I went to th…
3    My wife and I ordered 2 burgers, they took 30 …
4     Kids wanted Italian so we stopped at this res…
Name: review, dtype: object
```

[21]:
```
reviews.where(reviews.rating == 5.0).dropna()["review"].head(5)
```

[21]:
```
1000    Especially the friendliness and happiness from…
1001    One of the best in GentIf you want to taste an…
1002    We found restaurant food in Ghent to be very e…
1003    Best price/quality veggie in Ghent.Appelier is…
1004    Fab ribs & baked potatoes. The waiters are alw…
Name: review, dtype: object
```

After observing the positive and negative reviews, we saw that the negative reviews are much wider and specific than the positive reviews. For example the positive reviews will mostly say that the food is great, that the staff is great etc.

Negative reviews are complexer, people complain more about specific things and not an overall review.

We have also tried generating reviews such as “not good” or “not tasteful”, but still generates a positive review. We think that the ‘not’ doesn’t have the expected result of making the word the opposite. It works better when you use words such as “bad” or “tasteless”.

0.4 Conclusion

This model is a good improvement in comparison with our selfmade model, but it isn’t working perfect.

We think that the problem is that we don’t have enough negative reviews to be able to generate quality negative reviews. It takes more data to learn negative reviews because negative reviews are harder to generalize than positive reviews. We think that adding more negative reviews and training more epochs will make the model much better.

0.5 References:

https://towardsdatascience.com/how-to-fine-tune-gpt-2-for-text-generation-ae2ea53bc272