CSCE-638, Programming Assignment #3 Text Generation

Rohan Chaudhury

UIN: 432001358

Solution:

I have used **Python 3.9.13** to code the solution. With the pdf of this submission, I have also included the code files, generated Html files, and the folders fewshot_examples and test_examples_debug in a separate zip folder. I have also completed the bonus questions and have included their details in this document. I have used the following model and tokenizer for all my experiments:

```
    tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
    model = GPT2LMHeadModel.from_pretrained("gpt2")
```

1 Text Generation Basics:

(2 Points!) Task 1: Generation

I have implemented the self.generate_text() method under the TransformerModel() class without modifying any parameters in the transformer generation APIs. I used the following command to generate the output with this implementation and I ran it in the same folder as the code files:

python generation.py --max 100

Following are the three examples for this implementation and their analysis:

i. **First Provided prompt:**

“Text generation is the task of generating text with the goal of”

Generated output by the model:

“Text generation is the task of generating text with the goal of generating a text file.

The following example shows how to generate a text file with the following command:

$ python text_generate.py

The output will be:

The output will be:

The output will be:

The output will be:

The output will be:

The output will be:

The output will be:

The output will be:

The output will be:

The output will"

Analysis of the quality of the output:

The text generation started out nicely with no grammatical errors and with decent semantic cohesiveness however it started repeating the sentence "The output will be:" after a while. Also, the generated output is of little use to us.

To tackle the repetition issue, I have added the following parameters in the next question (Task 2: Better generations) implementation where we are asked to tune the model by modifying parameters in the transformer generation APIs.

```
no_repeat_ngram_size = 2, repetition_penalty = 1.5
```

ii. Second Provided prompt:

"I am Rohan Chaudhury and I am a Master of Science in Computer Science student studying at"

Generated output by the model:

"I am Rohan Chaudhury and I am a Master of Science in Computer Science student studying at the University of California, Berkeley. I am a member of the Board of Directors of the National Science Foundation and the National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science

Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National Science Foundation's National"

Analysis of the quality of the output:

I am impressed by the fact that it added the name of the University "University of California, Berkeley" at the end of the prompt and added a designation to the person in the output. The text generation started out nicely with no grammatical errors and with decent semantic cohesiveness however it started repeating the sentence "National Science Foundation's" after a while, an issue which is fixed in the next question implementation.

iii. Third Provided prompt:

"I hugely admire Peter Parker and his alter-ego Spider-Man. His motto 'With great power comes"

Generated output by the model:

"I hugely admire Peter Parker and his alter-ego Spider-Man. His motto 'With great power comes great responsibility' is a common theme in the Marvel Cinematic Universe.

But the fact that he's not a superhero is a bit of a surprise.

The actor, who has been in the Marvel Cinematic Universe since the early '90s, has been a big fan of the character since he was a kid.

He's also been a big fan of the character's origin story, which is based on the Marvel Comics character of the same name.

He's"

Analysis of the quality of the output:

In this prompt I tried testing the parametric knowledge of the model and found that it was able to successfully complete the sentence: "'With great power comes great responsibility' is a common theme in the Marvel Cinematic Universe", which is quite impressive. This shows that the model stores a wide range of knowledge in its parameters. The sentences generated after that are grammatically correct and semantically cohesive and does not suffer from repetition issues unlike the previous 2 examples. The sentences talk about a fictional human and portrays him as an actor in the Marvel Cinematic Universe.

(2 Points!) Task 2: Better generations

I tuned my implemented generation method by modifying parameters in the transformer generation APIs, specifically I added the following parameters:

```
When  num_return_sequences=1, I added:
```

```
no_repeat_ngram_size = 2, repetition_penalty = 1.5, top_k=100, top_p=0.95,
temperature=0.8

for num_return_sequence > 1, I added:

 num_beams=num_return_sequences, no_repeat_ngram_size = 2, repetition_penalty
= 1.5, num_beam_groups=num_return_sequences, diversity_penalty=1.5, top_k=100,
top_p=0.95, temperature=0.8
```

I used the following command to generate the output with this implementation and I ran it in the same folder as the code files:

python generation.py --max 100

Following are the four examples for this implementation and their analysis. For comparison, the first 3 example prompts are the same:

i. **First Provided prompt:**

"Text generation is the task of generating text with the goal of"

Generated output by the model:

"Text generation is the task of generating text with the goal of making it readable.

The following example shows how to generate a simple HTML document using JavaScript:

(function(text) { var html = new Text(); // Create an element for this function, and add some content here if (!html &&!documentElement ||!!this[0].innerHTML)){ console.log("<div class="content">"); } elseif ((!documents){ return; }); /* Add more information about what we're doing */ $("./script").append"

Analysis of the quality of the output:

The output is semantically cohesive, grammatically correct and makes great sense. In fact, it spit out code in JavaScript to generate an HTML document which is quite impressive. It does not suffer from the repetition issue as well.

ii. **Second Provided prompt:**

"I am Rohan Chaudhury and I am a Master of Science in Computer Science student studying at"

Generated output by the model:

"I am Rohan Chaudhury and I am a Master of Science in Computer Science student studying at the University of California, Berkeley.

The following is an excerpt from my book The Art of Programming: A Practical Guide to Understanding Your Own Code (University Press). It was published by Oxford Business Review on May 5th 2014"

Analysis of the quality of the output:

The output is semantically cohesive, grammatically correct. Although the output generated is purely imaginary, but still the output makes great sense with regards to a fictional character. It does not suffer from the repetition issue as well.

iii. Third Provided prompt:

“I hugely admire Peter Parker and his alter-ego Spider-Man. His motto 'With great power comes”

Generated output by the model:

“I hugely admire Peter Parker and his alter-ego Spider-Man. His motto 'With great power comes greatest responsibility' is a common theme in the Marvel Cinematic Universe, but it's also one that has been used by other superheroes to justify their actions on Earth as well:

The idea of using an alien race for your own purposes was first introduced into comics when Captain America (Chris Evans) tried to use them against him during World War II.[1] The concept came from Steve Rogers who had fought alongside Iron Man at some point before he became known simply "Captain American". In this case however”

Analysis of the quality of the output:

The output is semantically cohesive, grammatically correct and shows that it has a wide arsenal of knowledge inherently in its parameters. It does not suffer from the repetition issue as well.

iv. Fourth Provided prompt:

“I wonder what the world would be like in the year 4050, will there be”

Generated output by the model:

“I wonder what the world would be like in the year 4050, will there be a new civilization?

The answer is yes. The future of humanity depends on it and we must continue to work towards that goal as long or soon enough for us all to live together peacefully again.”

Analysis of the quality of the output:

I tried to explore the imaginative side of the model here and it performed wonders here. The output is semantically cohesive, grammatically correct and the provided output makes so much sense. It does not suffer from the repetition issue as well.

(3 Points!) Task 3: Evaluation

I have implemented the evaluation metric perplexity score for text (sequence) generation in the self.evaluate_ppl() method under the TransformerModel(). I used the following command to generate the output and I ran it in the same folder as the code files:

python generation.py --ppl

The perplexity score on the WikiText dataset from the implemented code is as follows:

Perplexity = 24.3746.

Perplexity is a method of assessing language models in natural language processing. A probability distribution across full phrases or texts is a language model. Perplexity is described as a sequence's exponentiated average negative log-likelihood. In the event that we have a tokenized sequence:

$$X = (x_0, x_1, \ldots, x_t)$$

,

then X's perplexity is defined as:

$$\text{PPL}(X) = \exp\left\{-\frac{1}{t}\sum_{i}^{t} \log p_\theta(x_i|x_{<i})\right\}$$

Where,

$$\log p_\theta(x_i|x_{<i})$$

denotes the log-likelihood of the ith token based on its preceding tokens in our model.

Perplexed is the English word for bewildered or confused. Perplexity in English language is the inability to handle or comprehend something challenging or unfathomable.

We find ourselves "perplexed" when a young child or newborn speaks incomprehensibly. The reason for this is because their spoken language does not conform to the syntax and grammatical structures of the language that we typically comprehend and speak.

Let's say we've used a ton of well-written blogs and responses to build a machine learning NLP model. The next step is to assess a specific StackOverflow answer's potential for moving to the forefront of the feed. The model that is least "perplexed" when presented with a well-written blog would be chosen from among the numerous models we trained.

So, a measure of a model's "uncertainty" in predicting (assigning probability to) given text is called the perplexity metric in NLP. Shannon's Entropy is related to it. Less perplexity means less entropy (uncertainty). If a model that has been trained on good blogs and is being tested on similarly good blogs assigns a greater probability, then we claim that this model has less perplexity than a model that assigns a lesser probability.

The range of Perplexity is as follows:

The maximum possible value of perplexity is **infinity**

The minimum possible value of perplexity is **1**.

So, the value of perplexity can be within **1** and **infinity**.

2 Sentiment Analysis as Text Generation

(3 Points!) Task 1: Few-shot predictions

1. I have modified the example template in the self.get template() method under the TransformerModel() class to the following:

```python
template = 'The movie review: \"%s\"\nThe observed sentiment: %s' %(doc, lbl)
```

2. I have selected the following examples (a total of 4) from the IMDB dataset provided in PA2, and have put them in corresponding folders under fewshot examples:
    a. For positive reviews, the chosen files are: **cv003_11664.txt, cv015_29439.txt [numbers 3 and 15]**
    b. For negative reviews, the chosen files are: **cv011_13044.txt, cv024_7033.txt [numbers 11 and 24]**

    After careful consideration I chose these files as these files represent the positive and negative sentiments respectively very well. I have not trimmed the reviews. The folder is submitted in the zip file.

3. Ran the code in debug mode with the above-mentioned changes in the code. Also,
    a. tweaked the function "fewshot_sentiment", to trim the test sentences (did not modify the files) only keep the first few sentences.
    b. tweaked the function "fewshot_sentiment", to shuffle the training data

    I used the following command to generate the output and I ran it in the same folder as the code files:

    python sentiment.py --debug

    Following is the output:

    Fewshot accuracy: 0.90

    Logs:

    ```
    $ python sentiment.py --debug

    negative file no. 999 got: The movie review: "two party guys bob their heads to ... sentiment: positive
    Fewshot accuracy: 0.90
    ```

4. Removed --debug and ran the testing with all 200 examples.
    I used the following command to generate the output and I ran it in the same folder as the code files:
    python sentiment.py

    Following is the output:

    Fewshot accuracy: 0.61

    Logs:

    ```
    $ python sentiment.py

    positive file no. 942 incorrect
    positive file no. 919 incorrect
    ```

```
positive file no. 981 incorrect
positive file no. 955 incorrect
.
.
.
.
(skipping the rest of the logs since they are huge and redundant)
.
.
negative file no. 965 incorrect
negative file no. 972 incorrect
negative file no. 977 incorrect

Fewshot accuracy: 0.61
```

So, in debugging mode: Fewshot accuracy: 0.90
And in non-debugging mode: Fewshot accuracy: 0.61

[Extra credits] (2 Points!) Task 2: Visualization

I have utilized the "output attention" feature when calling the GPT-2 model by setting output_attentions=True. Implemented the self.visualize attention() method under TransformerModel() to return the weights (importance) of each input token.

I have chosen the attention score from the last layer to visualize by summing and averaging their values across all the tokenized input embeddings and passed the decoded input embeddings with these scores as return values of the self.visualize attention() function. Also, I have modified the "sentiment.py" file so that it gets a list of decoded tokenized inputs and the corresponding scores from the self.visualize attention() function.

I used the following command to generate the output and I ran it in the same folder as the code files:

```
python sentiment.py -x --debug
```

Submitted the "explained.html" file along with the zip file. Following is a screenshot of the HTML file:

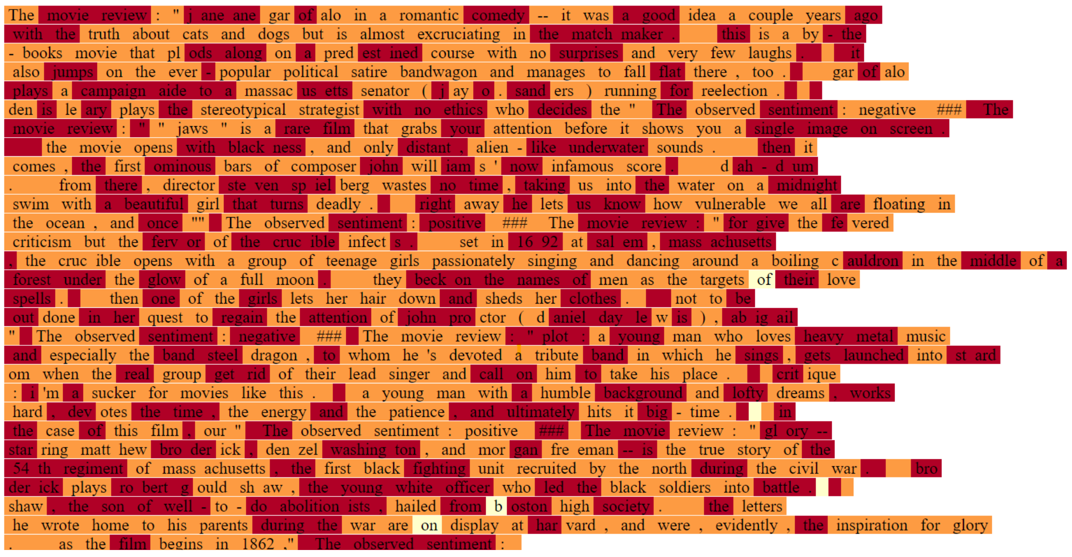

The movie review : " j ane ane gar of alo in a romantic comedy -- it was a good idea a couple years ago
with the truth about cats and dogs but is almost excruciating in the match maker . this is a by - the
- books movie that pl ods along on a pred est ined course with no surprises and very few laughs . it
also jumps on the ever - popular political satire bandwagon and manages to fall flat there , too . gar of alo
plays a campaign aide to a massac us etts senator (j ay o . sand ers) running for reelection .
den is le ary plays the stereotypical strategist with no ethics who decides the " The observed sentiment : negative ### The
movie review : " " jaws " is a rare film that grabs your attention before it shows you a single image on screen .
the movie opens with black ness , and only distant , alien - like underwater sounds . then it
comes , the first ominous bars of composer john will iam s ' now infamous score . d ah - d um
. from there , director ste ven sp iel berg wastes no time , taking us into the water on a midnight
swim with a beautiful girl that turns deadly . right away he lets us know how vulnerable we all are floating in
the ocean , and once "" The observed sentiment : positive ### The movie review : " for give the fe vered
criticism but the ferv or of the cruc ible infect s . set in 16 92 at sal em , mass achusetts
, the cruc ible opens with a group of teenage girls passionately singing and dancing around a boiling c auldron in the middle of a
forest under the glow of a full moon . they beck on the names of men as the targets of their love
spells . then one of the girls lets her hair down and sheds her clothes . not to be
out done in her quest to regain the attention of john pro ctor (d aniel day le w is) , ab ig ail
" The observed sentiment : negative ### The movie review : " plot : a young man who loves heavy metal music
and especially the band steel dragon , to whom he 's devoted a tribute band in which he sings , gets launched into st ard
om when the real group get rid of their lead singer and call on him to take his place . crit ique
: i 'm a sucker for movies like this . a young man with a humble background and lofty dreams , works
hard , dev otes the time , the energy and the patience , and ultimately hits it big - time . in
the case of this film , our " The observed sentiment : positive ### The movie review : " gl ory --
star ring matt hew bro der ick , den zel washing ton , and mor gan fre eman -- is the true story of the
54 th regiment of mass achusetts , the first black fighting unit recruited by the north during the civil war . bro
der ick plays ro bert g ould sh aw , the young white officer who led the black soldiers into battle .
shaw , the son of well - to - do abolition ists , hailed from b oston high society . the letters
he wrote home to his parents during the war are on display at har vard , and were , evidently , the inspiration for glory
. as the film begins in 1862 ." The observed sentiment :

Fig: **Screenshot of explained.html file**

From this visualization we can see which tokens the models had more attentions on and which ones it had less attentions on. This can be very helpful in debugging potential accuracy issues.

[Extra credits] (3 Points!) Task 3: Fine-tuning

For fine-tuning, used the same template and examples in Task 1 to construct training and test samples. Different samples are considered as individual prompts instead of being joined together. Implemented the self.finetune() method under TransformerModel() to fine-tune parameters in self.model and performed prediction on test samples.

Results after Finetuning:

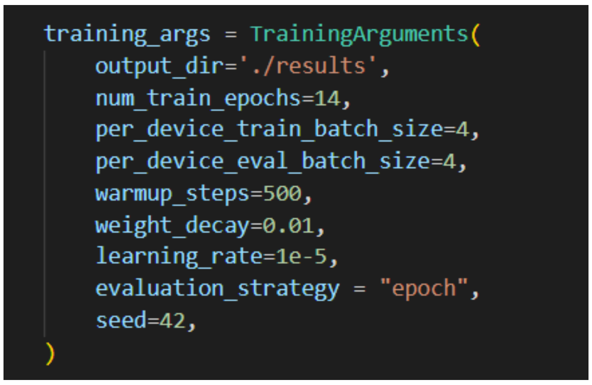
In debugging mode: (Command to run: python sentiment.py --tune --debug)

Accuracy after finetuning: **0.70** (Logs provided at the end in the additional outputs section)

In non-debugging mode: (Command to run: python sentiment.py --tune)

Accuracy after finetuning: **0.51** (Logs provided at the end in the additional outputs section)

Fine-tuned the model with the following arguments:


```
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=14,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    warmup_steps=500,
    weight_decay=0.01,
    learning_rate=1e-5,
    evaluation_strategy = "epoch",
    seed=42,
)
```

On comparing them with the results in Task 1 we can see that:

Results from Task 1 (Fewshot):

> **in debugging mode: Fewshot accuracy: 0.90**
> **And in non-debugging mode: Fewshot accuracy: 0.61**

So the values of fewshot accuracy is greater than fine-tuning accuracy for both debugging and non-debugging mode respetively. This signifies that fine-tuning the model does not help more in increasing the accuracy of the model than the fewshot setup and the model inherently works quite well in fewshot settings. Perhaps fine-tuning the model and then using fewshot might improve the accuracy by a bit more. Regardless, large language models like GPT-2 are expected to work good on language related tasks because they have inherently good knowledge of natural language stored in their parameters.

Additional output logs:

Logs for fine-tuning in debugging mode:

```
~/nlp_project/PA3/PA3-text-generation$  python sentiment.py  --tune --debug

.

.

.

(Redundant  logs removed)

.

.

100%|██████████████████████████████| 336/336 [01:13<00:00,  5.87it/s]***** Running
Evaluation *****

  Num examples = 128

  Batch size = 16

{'eval_loss': 3.9535322189331055,  'eval_runtime':  0.7863, 'eval_samples_per_second':  162.793,
'eval_steps_per_second':  10.175, 'epoch': 14.0}

100%|██████████████████████████████| 336/336 [01:14<00:00,  5.87it/s]


Training completed. Do not forget to share your model on huggingface.co/models =)



{'train_runtime':  74.6892, 'train_samples_per_second':  71.978, 'train_steps_per_second':  4.499,
'train_loss': 16.529673258463543,  'epoch': 14.0}
```

```
100%|██████████████████████████████| 336/336 [01:14<00:00,  4.50it/s]

positive file no. 999 got: The movie review: "truman ( " true-man " ) burbank ... sentiment:
negative

positive file no. 996 got: The movie review: "richard gere can be a commanding actor, ...
sentiment: negative

negative file no. 999 got: The movie review: "two party guys bob their heads to ... " positive

Accuracy after finetuning: 0.70
```

Logs for fine-tuning in non-debugging mode:

```
~/nlp_project/PA3/PA3-text-generation$ python sentiment.py --tune

.

.

.

(Redundant logs removed)

.

.

{'eval_loss': 2.546142578125, 'eval_runtime': 0.7755, 'eval_samples_per_second': 165.055,
'eval_steps_per_second': 10.316, 'epoch': 13.0}

100%|████████████████████████████████████████████████████████████████████████
███████████| 336/336 [01:14<00:00,  5.83it/s]***** Running Evaluation *****

  Num examples = 128

  Batch size = 16

{'eval_loss': 2.6668701171875, 'eval_runtime': 0.7741, 'eval_samples_per_second': 165.349,
'eval_steps_per_second': 10.334, 'epoch': 14.0}

100%|████████████████████████████████████████████████████████████████████████
███████████| 336/336 [01:14<00:00,  5.83it/s]


Training completed. Do not forget to share your model on huggingface.co/models =)



{'train_runtime': 74.878, 'train_samples_per_second': 71.797, 'train_steps_per_second': 4.487,
'train_loss': 16.514676048642112, 'epoch': 14.0}
```

```
100%|██████████████████████████████████████████████████████████████████████
██████████| 336/336 [01:14<00:00,  4.49it/s]

positive file no. 914 incorrect

positive file no. 967 incorrect

positive file no. 915 incorrect

.

.

.

.

(output truncated)

.

.

.

.

negative file no. 923 incorrect

negative file no. 984 incorrect

negative file no. 977 incorrect

negative file no. 990 incorrect

negative file no. 957 incorrect

negative file no. 922 incorrect

Accuracy after finetuning:  0.51
```