Harry Potter's Adventure With GPT2 And LSTM!

Team Members: Kit Chung Yan and Tyler Nguyen

Project Description:

The aim of this project is to leverage the power of language models, specifically GPT-2, to generate original chapters in the style of J.K. Rowling's Harry Potter series. The task involves fine-tuning the GPT-2 model on existing Harry Potter chapters, enabling the model to learn the intricate language patterns, storytelling style, and thematic elements unique to the world of Harry Potter. The project allows for the exploration of machine-generated storytelling. By training the model on multiple Harry Potter texts, it has the potential to generate new and captivating chapters that adhere to the established narrative style. It is just funny to see what the model can write based off such a popular and loved series everyone knows about. The project also builds upon concepts covered in the class, such as fine-tuning language models, working with text data, and using pre-trained models. It serves as a practical application of the knowledge gained during the course. In addition to fine-tuning the GPT-2 model, we will also be training an LSTM model and then compare the results on how effective each model compares with one another in text generation.

Resources:

Here are the 5 resources that we will be using:

- https://monkeylearn.com/sentiment-analysis/<--(General Info on SA)
- https://link.springer.com/article/10.1007/s10462-022-10144-1 <--(Research Article)
- https://www.frontiersin.org/articles/10.3389/frobt.2019.00053/full <--(Example of Harry Potter Movie character sentiment anaylsis)
- https://huggingface.co/blog/sentiment-analysis-python <--(An example of approach of doing sentiment anayalsis task using Transformers)
- https://www.kaggle.com/code/tuckerarrants/text-generation-with-huggingface-gpt2
- https://github.com/amephraim/nlp/blob/master/texts/J.%20K.%20Rowling%20-%20Harry%20Potter%201%20-%20Sorcerer's%20Stone.txt <--(Harry Potter Text)

Abstract:

The problem that we're trying to solve is identifying the accuracy of generating Harry Potter Book text. Essentially, we're expanding our previous homework and try to refine the GPT-2 transformer model and LSTM model to see how well each model can generate long text. The reason why we chose to do Harry Potter text was because those text are usually very long and it contain enough

data to prevent underfitting/overfitting. To calculate the accuracy of how well GPT-2 generates text, we decided to train an LSTM model which resorts in using sequential data generation rather than using a transformer model. The approaches taken to solve this problem was to first setup our SCC and Colab environment. The second step was to collect Harry Potter text, preprocess the data, fine-tuning our pretrained GPT-2 model from HW06 and reused the LSTM model from HW05. Results are what we expected to be GPT-2 Transformer Model generating better quality sentences compared to the LSTM Model. We will be using PyTorch for most of the network training.

Presentation of the results:

Again to reiterate, for this project, we are trying to compare text generation of Harry Potter Text using GPT-2 pretrained Transformer model finetuning it's pretrained model with an LSTM model. We think this would be an interesting topic to focus on because it's really cool to see just how machines can generate text based on the parameters that you give to train from. For this project, we reused most of the codes from HW06 and HW05 as all the codes were already available and all we needed was the Harry Potter text to start generating text. We did use some outside source for LSTM text generation and we decided to use tensorflow to train our LSTM model instead of using PyTorch. We reused 3 Harry Potter Books encoded as raw txt file:

- Harry Potter and the Sorcerer's Stone
- Harry Potter Prisoner of Azkaban
- Harry Potter The Goblet of Fire

The approach that we decided to run this project was to use both the SCC and Google Colab environment to train both our models depending on how much memory it would take with training. The SCC environment was definitely more efficient because it had enough memory and a insane GPU to train our model in thus reducting the time needed to train. But the most important process was to gather the data and format it to put it into our training model. After that was completed, we just needed to train the model and anayalze the results after that.

Here is a snippet of how we processed each of the Harry Potter Books.

Preprocessing the Harry Potter Text

So before we plugged our data into our dataset, we had to preprocess it first by splitting each of the books up by it's chapters. Then, we get rid of any uncessary characters like special markings that would prevent GPT-2 from generating any unwanted characters. In the end, we just wanted to see how well it can generate the text itself given all the Harry Potter books to train from. Essentially, we concatenate every chapter from every book in order to have a list of chapters from each of the book series. The plan then is to then generate multiple paragraphs of text given an intial prompt to start generating from and then keep updating the prompt for n amount of iterations from the last

100 words. This way, we can keep generating text and updating the prompt based on the previous text generation.

```
from google.colab import drive

drive.mount('/content/drive')
from PIL import Image

img = Image.open('/content/drive/My Drive/HarryTxt.png')
display(img)
```

```
Mounted at /content/drive
```

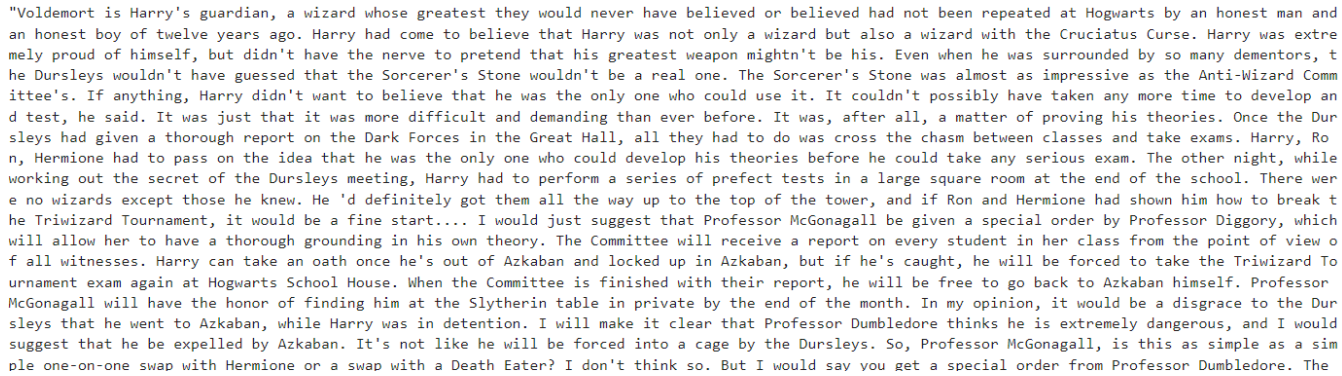
```
# file_path = "/content/drive/My Drive/J. K. Rowling - Harry Potter 1 - Sorcerer's Stone.txt"
file_path = "J. K. Rowling - Harry Potter 1 - Sorcerer's Stone.txt"
```

```
with open(file_path, 'r') as file:
    content = file.read()
chapters = content.split('CHAPTER')
chapters = chapters[1:]
def cleaner(chapters):
  text = ((nltk.word_tokenize(chapters)))
  #print(set(cleaned.split(' ')))
  clean_text = [token for token in text if token != "'"]
  clean_text = [token for token in clean_text if token != "``"]
  clean_text = [token for token in clean_text if token != "--"]
  clean_text = [token for token in clean_text if token != "-"]
  clean_text = [token for token in clean_text if token != "..."]
  clean_text = [token for token in clean_text if token != "...."]
  return ' '.join((clean_text))

docs = [cleaner(chapter) for chapter in chapters]
```

Generate Harry Potter Text via GPT-2:

So initially, we only train our model on just one book running 10 epochs and that didn't produced any relevant results related to the Harry Potter text. Instead, it was producing pretrained text from GPT-2 and our reason to believe that is because there just wasn't enough data to process for our Harry Potter text generation. So it would just generate text based on pretrained text. As a result, we included 2 more Harry Potter books into our dataset and see if it yielded better results. We also changed some hyperparameters like increasing epochs from 10 to 100 so we can see any improvement in the model producing more accurate text generation related to a Harry Potter related prompt. Below are some of the results we got from running 10 epochs to 100 epochs:

```
# Text Generation generating n number of times training the model with 10 Epochs

# prompt = "Voldemort is Harry"
# generated_text = generate_multiple_text(prompt, number_of_iterations=8)
img = Image.open('/content/drive/My Drive/10epochs.png')
display(img)
```




```
# Text Generation generating n number of times training the model with 100 Epochs
# prompt = "Voldemort is Harry"
# generated_text = generate_multiple_text(prompt, number_of_iterations=8)  # Change the num
# generated_text
img = Image.open('/content/drive/My Drive/100epochs.png')
display(img)
```

```
"Voldemort is Harry's friend. Even though he could see past his numb disbelief, he couldn't help it ; he felt as though he 'd lost one of his best friend
s. He had a stream of visitors, all intent on cheering him up. Hagrid sent him a bunch of earwiggy flowers that looked like yellow cabbages, and Ginny We
asley, blushing furiously, turned up with a get-well card she had made herself, which sang shrilly unless Harry kept it shut under his bowl of fruit. The
Gryffindor team visited again on Sunday morning, this time accompanied by Wood, who told Harry ( in a hollow, dead sort of voice ) that he didn't blame h
im in the slightest. Ron and Hermione left Harry's bedside only at night- But nothing anyone said or did could make Harry feel any better, because they k
new only half of what was troubling him. He hadn't told anyone about the Grim, not even Ron -and Hermione, because he knew Ron would panic and Hermione w
ould scoff. The fact remained, however, that it had now appeared twice, and both appearances had been followed by near-fatal accidents ; the first time,
he had nearly been run over by the Knight Bus ; the second, fallen fifty feet from his broomstick. Was the Grim going to haunt him until he actually die
d? Was he going to spend the rest of his life looking over his shoulder for the beast? And then there were the dementors. Harry felt sick and humiliated
every time he thought of them. Everyone said the dementors were horrible, but no one else collapsed every time they went near one. No one else heard echo
es in their head of their dying parents. Because Harry knew who that screaming voice belonged to now. He had heard her words, heard them over and over ag
ain during the night hours in the hospital wing while he lay awake, staring at the strips of moonlight on the ceiling. When the dementors approached him,
he heard the last moments of his mother's life, her attempts to protect him, Harry, from Lord Voldemort, and Voldemort's laughter before he murdered her
Harry dozed fitfully, sinking into dreams full of clammy, rotted hands and petrified pleading, jerking awake to dwell again on his mother's voice. It was
```

Evaluation of Results from GPT-2

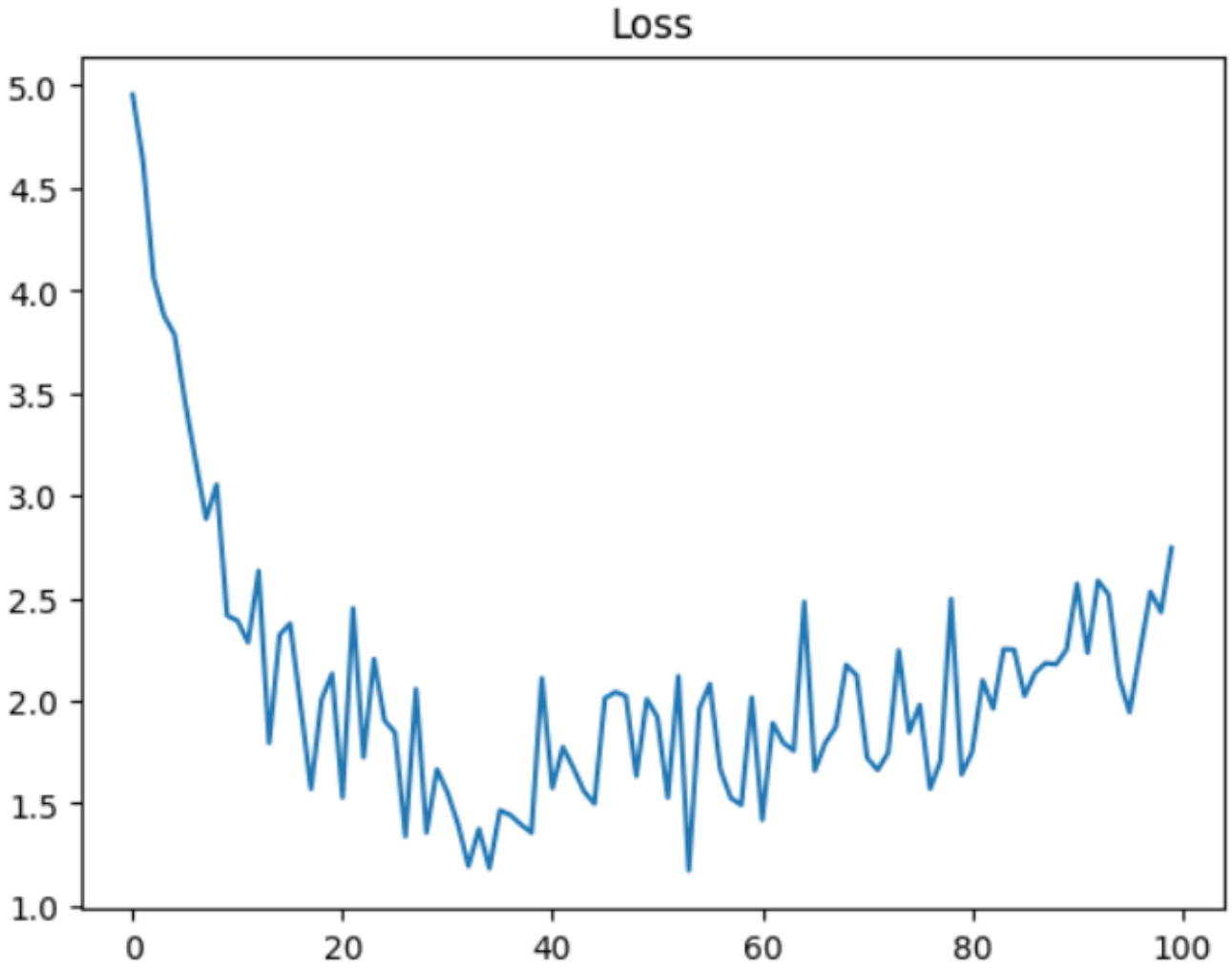
After finetuning the pretrained GPT-2 model with Harry Potter text, it actually did a very good job at generating continuous prompt from the last 100 words. Initially when we had only one book, the results weren't that accurate as it was including the text from pretrained model. And we didn't do a good job at processing the text the first time we plug the dataset in to train which didn't work until we only kept as much word tokens as there can be. The way we evaluated our results was that instead of generating just one prompt of a paragraph text, we wanted to see how well it can train by generating n number of prompts. We wrote a algorithm to calculate by running a for loop that will create a prompt based of the last 100 words produced from the previous prompt. You can see more of the algorithm in our GitHub. Overall, we feel like it was a success on what we needed to accomplish which was to generate quality text from Harry Potter books and not just generating random text that is irrelevant to Harry Potter. These text looks very accurate to what is actually relevant to Harry Potter text. GPT-2 seemes to be a very powerful transformer model that is capable of generating continous text that seems relevant to whatever prompt you inputted to generated.

Generate Harry Potter Text via LSTM:

For LSTM model, we don't plan on this model generating better relevant text compared to GPT-2 transformer model because it's doing sequence prediction which is prediction of the next word in order based on the left context. As a result, it may not produce better text whereas GPT-2 can account for all the elements in the sequence all at the same time by going through different positions within a sequence in order get the best representation of each element. For this part, we reused most of the LSTM model from HW05 when we were performing word generation with the Jane Austen's Persuasion text. You can look at our code on GitHub to see more on the process it took to preprocess each word via converting it from word to indice to indice to word when plugging into the dataloader to train. When training our LSTM model, we had to flattened every chapter of each Harry Potter book into one gigantic list of all tokens as that was a much easier approach to generate each word based on leftmost context. The batch size remained the same with a batch

size of 64 using the Adam optimizer algorithm with learning rate of 0.001. Here is a result of loss along with the text generated from the LSTM model:

```
img = Image.open('/content/drive/My Drive/LSTMloss.png')
display(img)
```




```
# Text Generation generating n number of times training the model with 100 Epochs
# prompt  = 'Harry Potter'
# gen_sentence = generate_sentence(model,prompt,word_to_idx,idx_to_word)
img = Image.open('/content/drive/My Drive/LSTM.png')
display(img)
```

```
"Harry Potter heap upright though they were standing behind her collar , staring down at the swirling white mist before Ludo Bagman in the doorway . Amos
Diggory , what 's fingers ? The other two clunking bottle into fireworks Mmm ! said Ron as she spotted Harry 's trolley and put his broom out of the way
, staring at his watch . From what I had the right person in here he 'd just swallowed a couple of hours ? It 's very easy , had grown at birthday mornin
g and try . Hagrid was right Ron . He was looking at Harry , Ron , and Hermione 's sounded shrill voice echoing across his head . Mom , now , please . Ha
haaa and Told us only when Slytherin is a little time for Christmas ! I did n't give it , said Harry . at all they 'd emerged looking up to the kitchens
. Harry 's eyes filled with tears a sudden burst out of its wits . It 's the best if we ever had been his face He stole livid tonight . This is one of th
e dog ! Quirrell 's , he said in a muffled squeak . No one , said Lupin . Our examination of the castle is they 've done ages into any more . Oh , that
's not true ? said Harry breathlessly . They 'd like to see you ! Ron said angrily , pointing at him , but I think Hagrid did n't people he was . It took
Neville , Harry , come on , I 'll have to speak , said Ron , but Ron pulled back the front steps onto the spiral staircase to their room , still went out
of the tent , and together he had done it , said Harry . At the way you only got inside here . I do n't know the Stone if he 's got it before . That 's w
hat Hagrid might move , please Sirius Lupin and Ron heaved a shining deep mud then a nasty grin on top of a large ginger cushion , his ripped robes off o
```

Evaluation of Results from LSTM

As you can see above, the results aren't that effective at all when it comes to generating Harry Potter text. Just like how we evaluted the results for GPT-2, we wanted to test out the generative effectiveness by using the same prompt or related Harry Potter prompts and GPT-2 clearly outdo LSTM in terms of the quaity of the sentences genrated. The reason for this again could be that LSTM is just going to learn the sequence of the word so it may not always produce accurate results if those words can show up elsewhere in the text. Given LSTM is going sequence by sequence text generation.I felt like LSTM could've done better in this case but it's just not the best text generative model out there. It relies on feeding the text data in and essentially needing to learn through all the text words in order to generate the text. But again, as the prompts get larger, LSTM can't capture it's previous prompt topic which can lead to irrelvant text being generated and some ungrammtical sentences as well.

Future Plans

In the future if we do plan on continue working on this, we'd probably try to incorporate some sentiment anaylsis to this topic or continuing tuning the hyperparameters and see how much better can each model generate texgt. Sentiment Anaylsis was our original project idea that we wanted to do with other text but we just didn't have enough time to create the code and anaylze it. If we didn't have any time constraint, the sentiment anaylsis would be focusing on the mood change of each chapter so we can find out how much the mood changes throughout the chapters of each Harry Potter Text. Or maybe try to to perform sentiment anaylsis on the characters themselves and view the mood change of each character throughout the story. But we do look forward in some time during the future to continue working on this project with the progress we made so far.

Link to GitHub for our code

Here is the link for all of our code for the project! You can look at all the code that we wrote for this project:

https://github.com/KitYan20/CS505Project

Team Contributions

Our team of 2 contributed equal amount of work for this project. Since Tyler was already familiar with the GPT-2 model, he was able to get that all setup and running. Kit was responsible to help cleanup the text for the Harry Potter files, and work on implementing the LSTM model, write up the evaluations of the results for our report. Overall, each of us contributed work equally either by talking through how to approach each problem, identifying bugs to fix, and just talked with each other on what work needed to be done. It was great team effort to get all the code up and running in order to generate the text! This was a really cool project to work on and we would definitely hope to continue working on this future as well!