INDRAPRASTHA INSTITUTE of INFORMATION TECHNOLOGY DELHI

CSE508 Information Retrieval

Assignment-4

Shrugal Tayal

shrugal20408@iiitd.ac.in

20th April, 2024

Review Summarization using GPT2

Overview:

This notebook guides through fine-tuning Hugging Face's GPT-2 model on Amazon Fine Food Reviews dataset for summarization.

Through this exploration, we seek to improve retrieval accuracy and user experience in information retrieval applications.

Certainly! Here's the updated explanation with the code snippets included:

1. Importing Libraries

The code begins by importing necessary libraries for the task, including the GPT-2 model and tokenizer from the Transformers library and the Rouge library for computing ROUGE scores.

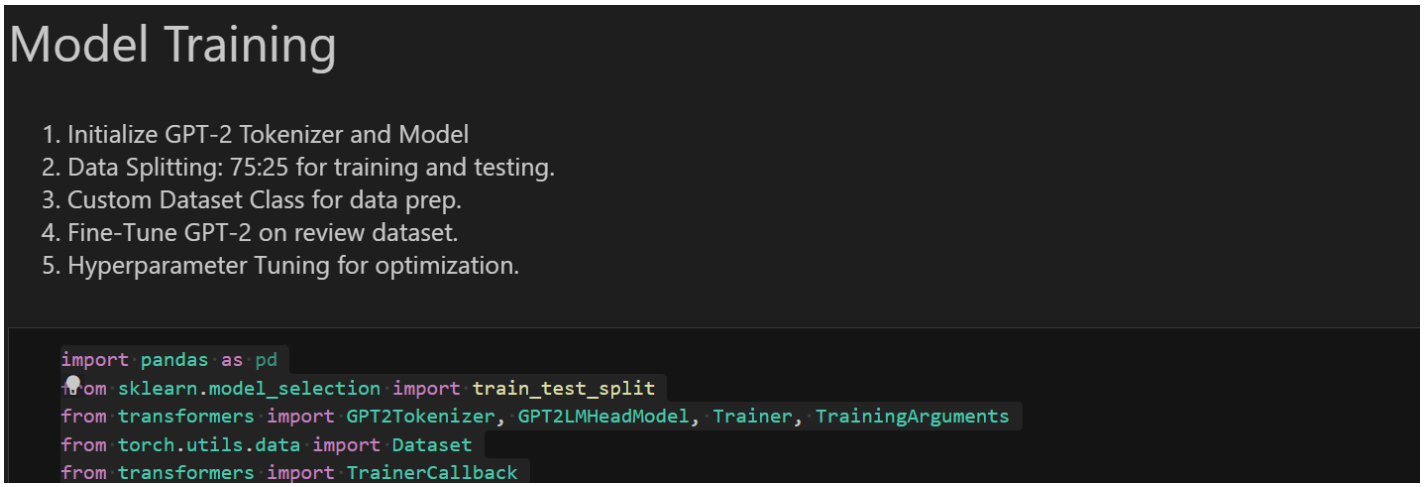
Load the Amazon Fine Food Reviews dataset

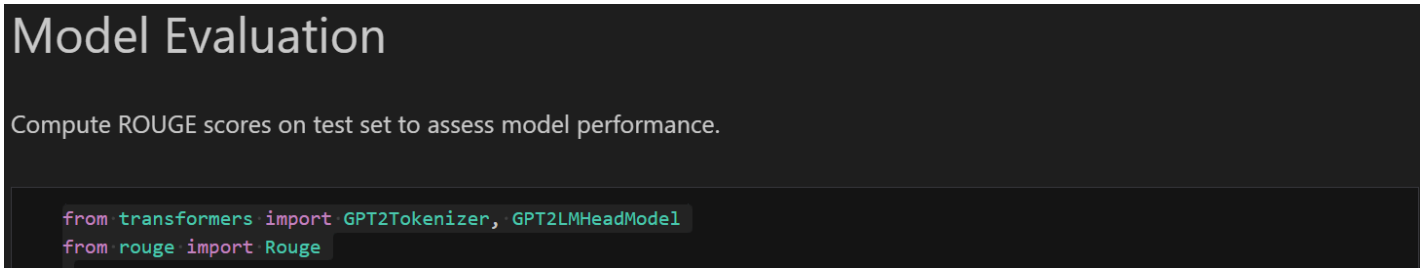
```python
import pandas as pd
```

Text Preprocessing on the 'Text' and 'Summary' column from the dataset

```python
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import re

# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```





2. Text Preprocessing

Before generating a summary, the provided review text undergoes preprocessing. This step ensures that the input text is clean and suitable for generating accurate summaries. Preprocessing typically involves converting text to lowercase, removing special characters, numbers, and stopwords, tokenizing the text, and performing lemmatization to reduce words to their base form.

```python
# Define a function for text preprocessing
def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()

    # Remove special characters and numbers using regex
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenize the text
    tokens = word_tokenize(text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    # Join tokens back into text
    cleaned_text = ' '.join(tokens)

    return cleaned_text
```

3. Loading Fine-Tuned Model and Tokenizer

Next, a pre-trained GPT-2 model and tokenizer are loaded. These models have been fine-tuned specifically for text summarization tasks, ensuring optimized performance in generating concise summaries from input text.

```
C:\Users\HP\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\si
dataloader_config = DataLoaderConfiguration(dispatch_batches=None, split_batches=False, even_batches=True, use_seedable_samp
  warnings.warn(

  0%|          | 0/2665 [00:00<?, ?it/s]

  0%|          | 0/889 [00:00<?, ?it/s]

{'eval_loss': 2.584134101867676, 'eval_runtime': 1297.9282, 'eval_samples_per_second': 2.737, 'eval_steps_per_second': 0.685
{'train_runtime': 21533.9564, 'train_samples_per_second': 0.495, 'train_steps_per_second': 0.124, 'train_loss': 2.5553141123

('fine_tuned_gpt2_model\\tokenizer_config.json',
 'fine_tuned_gpt2_model\\special_tokens_map.json',
 'fine_tuned_gpt2_model\\vocab.json',
 'fine_tuned_gpt2_model\\merges.txt',
 'fine_tuned_gpt2_model\\added_tokens.json')
```

4. Summary Generation

Once the model and tokenizer are loaded, the review text is tokenized using the tokenizer. The tokenized input is then fed into the GPT-2 model to generate a summary. During the generation process, the model

predicts the most probable next token at each step based on the input tokens, producing a sequence of tokens that represents the summary.

```
from transformers import GPT2Tokenizer, GPT2LMHeadModel
from rouge import Rouge

# Load tokenizer
tokenizer = GPT2Tokenizer.from_pretrained("fine_tuned_gpt2_model")

# Load model
model = GPT2LMHeadModel.from_pretrained("fine_tuned_gpt2_model")

# Define review text
review_text = "The Fender CD-60S Dreadnought Acoustic Guitar is a great instrument for beginners. It has a solid construc
# review_text = input("Given Review Text: ")

# Tokenize the review text
inputs = tokenizer(review_text, return_tensors="pt")

# Generate summary
generated_summary_ids = model.generate(inputs.input_ids, max_length=50, num_beams=4, early_stopping=True)
generated_summary = tokenizer.decode(generated_summary_ids[0], skip_special_tokens=True)

# Print generated summary
print("Generated Summary:", generated_summary)

# Define actual summary
actual_summary = "Good for beginners but has tuning stability issues."
# actual_summary = input("Given Summary :")
```

5. Evaluation Using ROUGE Scores

After generating the summary, it's compared against the actual summary using the ROUGE metric. ROUGE evaluates the overlap between the generated and actual summaries by measuring precision, recall, and F1-score for unigram, bigram, and Longest Common Subsequence (LCS) matches. This provides a quantitative measure of how well the generated summary captures the essence of the original text.

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass you
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated Summary: The Fender CD-60S Dreadnought Acoustic Guitar is a great instrument for beginners. It has a solid constru
ROUGE-1: Precision: 0.17647058823529413 Recall: 0.75 F1-Score: 0.2857142826303855
ROUGE-2: Precision: 0.05714285714285714 Recall: 0.2857142857142857 F1-Score: 0.09523809246031753
ROUGE-L: Precision: 0.14705882352941177 Recall: 0.625 F1-Score: 0.2380952350113379
```

Github

https://github.com/ShrugalTayal/CSE508_Winter2024_A4_2020408