In-Class Quiz: Advanced Text Generation Application

Introduction

The **Advanced Text Generation Application** is an innovative tool that utilizes the Hugging Face transformers library to generate coherent text based on user-input prompts. This enhanced version includes features for **Sentiment Analysis**, **Batch Processing**, and an **Interactive Mode**, allowing users to engage with the application dynamically and efficiently.

Note:
The quiz is worth a total of 8 points. You will receive full credit if you complete and submit it by 11:59 PM on April 21. Additionally, if you finish and submit it by 3:50 PM on April 21, you will receive 3 extra credit points. We will upload a sample solution to the same directory on Canvas at 3:55 PM on April 21, which you can refer to.

Objectives

This project provides an engaging opportunity for students to learn about the concepts of large language models (LLMs) operating on a cloud machine, with a specific focus on text generation. By completing the required tasks, you will gain practical experience with modern cloud LLM tools and will be able to write your own code to enable the following functions of text generation:

1. **Sentiment Analysis**:
    - Assess the emotional tone of the generated text, categorizing it as positive, negative, or neutral.
2. **Batch Processing**:
    - Allow users to input multiple prompts simultaneously, generating results efficiently for each prompt in a single operation.
3. **Interactive Mode**:
    - Enable continuous user interaction, allowing multiple prompts to be processed during a single session without restarting the application.
4. **Save Results to File**:
    - Store the generated text and sentiment analysis results in a text file for easy reference.

5. **User-Defined Parameters**:
    - Allow users to specify temperature, maximum length, and number of paragraphs for text generation.

The program you created will have the following key features:

- **Coherent Text Generation**: Leverage the GPT-2 model to produce human-like text based on user inputs.
- **Emotional Insights**: Offer sentiment evaluations alongside generated content for understanding tone.
- **Efficiency in Processing**: Support batch processing to handle multiple prompts in one go.
- **Interactive Experience**: Allow users to generate text interactively without restarting the application.
- **Results Documentation**: Save all generated outputs and analyses in a structured format.

Example Code

Below is the foundational code for the **Advanced Text Generation Application**. It includes necessary setups for text generation and outlines the framework for implementing sentiment analysis, batch processing, and interactive mode functionalities.

File: advanced_text_generation.py

You can download the file from the same folder of this introduction.

```
from transformers import pipeline

# Load the text generation pipeline using GPT-2
generator = pipeline('text-generation', model='gpt2', device=-1)  # Use CPU

# Load the sentiment analysis model

### TO DO ###

# Hint: You can refer to the instructions in the introduction file for more
information about the function to use, as well as its input and output.

def generate_text(prompts, max_length=100, temperature=0.7,
num_return_sequences=1):
```

```
    """Generate text using the GPT-2 model based on a list of prompts."""

    ### TO DO ###

    # Hint: You can use a for loop to generate each prompt or set the input
directly as a list of prompts.
    # Hint: To retrieve the output, you may use `result['generated_text']`. You
can also refer to the code from the last LLM Quiz for guidance on the output
format.

def save_to_file(results, filename='generated_results.txt'):
    """Save generated text and sentiment results to a file."""
    with open(filename, 'w', encoding='utf-8') as file:
        for result in results:
            file.write(f"Prompt: {result['prompt']}\n")
            file.write(f"Generated Text: {result['generated_text']}\n")
            file.write(f"Sentiment: {result['sentiment']['label']} (Score:
{result['sentiment']['score']:.2f})\n")
            file.write("-" * 40 + "\n")
    print(f"Results saved to {filename}")

def main():
    print("Welcome to the Advanced Text Generation Application!")

    # Get user-defined parameters
    max_length = int(input("Enter the maximum length of generated text
(recommended 30-100): "))
    temperature = float(input("Enter temperature (recommended range 0.5-1.0): "))
    num_paragraphs = int(input("Enter the number of paragraphs to generate for
each prompt: "))

    results_to_save = []

    # Interactive Mode: Allow users to input multiple prompts

    ### TO DO ###

    # Hint: You can use a while loop to keep generating text based on the input
prompts until you enter 'exit' or 'quit' to terminate the interactive mode.

        user_input = input("Enter your prompts separated by commas (or type
'exit' or 'quit' to quit): ")

        # Clean the generated text for current batch
        results = []
```

```
        # Split the input into a list of prompts (separated by commas)
        prompts = [prompt.strip() for prompt in user_input.split(',')]

        # Generate text for a batch of prompts
        generated_texts = generate_text(prompts, max_length, temperature,
num_paragraphs)

        # Analyze sentiment for each generated text and store results

        ### TO DO ###

        # Hint: You can call the loaded sentiment analysis model you created
before and refer to "Display Results Clearly" below for more information on the
format of the results.

        # Display results clearly
        for result in results:
            print(f"Prompt: {result['prompt']}")
            print(f"Generated Text: {result['generated_text']}")
            print(f"Sentiment: {result['sentiment']['label']} (Score:
{result['sentiment']['score']:.2f})")
            print("-" * 40)

        # Save results to a file after each batch
        results_to_save.extend(results)
        save_to_file(results_to_save)

if __name__ == "__main__":
    main()
```

Instructions for Students

In the end of this file you will see an example output of this program, which helps you better understand the instruction and write the code.

1. **Implement Sentiment Analysis**:
   - Use the Hugging Face sentiment analysis pipeline as follows:

     ```
     sentiment_analyzer = pipeline('sentiment-analysis')
     ```

   - **Input Format**:
     1. The input can be a single string or a list of strings representing the text(s) you want to analyze.
     2. For example:

        ```
        text = "I love this product!"  # Single input
        ```

```
texts = ["I love this product!", "This is the worst experience I've ever had."]  # Batch input
```

- **Output Format**:
  1. The output will be a list of dictionaries, with each dictionary containing:
     1) `label`: Indicates the predicted sentiment (e.g., "POSITIVE", "NEGATIVE").
     2) `score`: A float representing the confidence level of the sentiment prediction (which ranges from 0 to 1).
  2. Example output for a single input:

```
output = sentiment_analyzer(text)
# Output: [{'label': 'POSITIVE', 'score': 0.99}]
```

  3. Example output for batch input:

```
output = sentiment_analyzer(texts)
# Output: [{'label': 'POSITIVE', 'score': 0.99}, {'label': 'NEGATIVE', 'score': 0.85}]
```

2. **Implement Batch Processing**:
   - Write your own code to generate text for each prompt in the generate_text function.
   - Hint: You can use a for loop to generate each prompt or set the input directly as a list of prompts.
   - Hint: To retrieve the output, you may use `result['generated_text']`. You can also refer to the code from the last LLM Quiz for guidance on the output format.

3. **Create Interactive Mode**:
   - Implement a continuous loop in the `main` function that allows users to input prompts repeatedly.
   - Check for commands like "exit" or "quit" to allow graceful termination.
   - Hint: You can use a while loop to keep generating text based on the input prompts until you enter 'exit' or 'quit' to terminate the interactive mode.
   - For each valid prompt, generate the corresponding text using the `generate_text` function and analyze its sentiment.
   - Finally, you need to store the result in the `result` list.
   - Hint: You can call the loaded sentiment analysis model you created before and refer to "Display Results Clearly" in the code for more information on the format of the results.

Feel free to explore and customize the application further based on your interests. If you have any questions during your implementation, don't hesitate to ask!

We will upload a sample solution to the same folder on Canvas at 3:55 PM on April 21.

Submission

Please save your output to a file and upload it along with your code to the quiz.

Example outputs of the code after completing all TODOs:

Welcome to the Advanced Text Generation Application!

Enter the maximum length of generated text (recommended 30-100): 30

Enter temperature (recommended range 0.5-1.0): 0.5

Enter the number of paragraphs to generate for each prompt: 2

Enter your prompts separated by commas (or type 'exit' or 'quit' to quit): Monday is, Tuesday is

.....

Prompt: Monday is

Generated Text: Monday is the first day of the new year.

In the past, the company has been known to offer free shipping and to charge customers for

Sentiment: POSITIVE (Score: 0.60)

Prompt: Monday is

Generated Text: Monday is a good time to look at the past.

As a former NFL player, I'm not a fan of the current NFL. I

Sentiment: POSITIVE (Score: 1.00)

Prompt: Tuesday is

Generated Text: Tuesday is the day that the final round of the playoffs begins, and the final round of the playoffs ends.

If you want to know more

Sentiment: POSITIVE (Score: 0.72)

Prompt: Tuesday is

Generated Text: Tuesday is the first day of the NFL Draft, and while it's not the first time the draft has been held during the offseason, it will be

Sentiment: POSITIVE (Score: 1.00)

Results saved to generated_results.txt

Enter your prompts separated by commas (or type 'exit' or 'quit' to quit): Wednesday is, Thursday is

.....

Prompt: Wednesday is

Generated Text: Wednesday is the deadline for the first of six teams to sign a one-year, $5 million deal with the Eagles. The Eagles will have to

Sentiment: NEGATIVE (Score: 0.98)

Prompt: Wednesday is

Generated Text: Wednesday is the 10th anniversary of the launch of the first Galaxy S4 smartphone.

The first Galaxy S4 was released in April 2013.

Sentiment: POSITIVE (Score: 0.99)

Prompt: Thursday is

Generated Text: Thursday is the last day of the season.

The Packers have won three straight games and are looking to extend their winning streak to five games.

Sentiment: POSITIVE (Score: 0.99)

Prompt: Thursday is

Generated Text: Thursday is the first time that a major-league team will play in the American League East. The Astros will play the Dodgers in the American League East

Sentiment: POSITIVE (Score: 1.00)

Results saved to generated_results.txt

Enter your prompts separated by commas (or type 'exit' or 'quit' to quit): exit

Exiting the application. Goodbye!