CSCI 297 - Generative AI Project 4 Report

Malachi Eberly, Bennett Ehret, Armando Mendez, Micah Tongen, Barrett Bourgeois

Methodology

Data Collection

The first step of this project was the data collection and preparation. Upon doing research into web scraping, we found an XML document called a “site map”, that contains all of the sub URLs under the wlu.edu domain. From this point, we wrote a simple Python script that went through each URL in the sitemap and extracted the raw text components.

The raw text was then split into sentences using regex sentence matching. The data was then filtered for length, removing outliers with a minimum length threshold of 3 words and maximum of 40 words. Each sentence was paired with its respective website of origin and formatted as follows: “The following is a response from the Washington and Lee website: {note} To learn more, visit {url}”. The data was then split using a <> training/testing ratio.

Model Fine-tuning

The model fine-tuning process took place with help from the example notebook. We decided to also use a brev.dev instance for tuning to save time and improve results. As per common practice, we used an 80/20 split for training and testing the data. After a few hours of training, the model had a 100-checkpoint and was ready for integration as a chatbot.

App Development

To interface with the chatbot and use the tuned Phi2 model, we created a Gradio app. The process of app development was straight-forward and required relatively few lines of code. Using the Phi2 and Gradio tutorials, we designed the interface script to import and load the model and tokenizer, set up the pipeline for generating text, create a function to generate text in the proper format based on the incoming and previous messages, and finally display the chatbot window. Gradio also allows for useful features such as example question prompts, which we implemented for users to click. All of the responses were set to have a token limit of 64 to properly balance processing time and response quality.

Challenges Faced

One of the main challenges that we faced was in the training of the model. After our first few training runs, we found that the model was not accurate, giving us incorrect information in response to our prompts. After a few days of trying different training methods, we discovered that this issue may have been caused by our training data instead of the actual fine-tuning methods. It was then discovered that we were mistakenly training on a different file than intended, accidentally using a previous version that was not fully cleaned and prepared for fine-tuning. We re-trained on the correct data and found better results

Another challenge was the specific dependencies needed to run the tuned model on our local machines. We were able to run the Gradio app with the tuned model on the brev.dev instance that we used for training, but there were some issues running on our local machines with the peft and bitsandbytes modules. However, we expect that on a more powerful machine with the right modules installed, our training checkpoint will work with the model to provide improved responses. More improvement is needed, as can be seen from the demo results, but the pipeline of data, tuning, and interfacing runs smoothly, and results could be improved with more training.

Instructions for Running App

1. Clone the WLU-GPT GitHub Repository onto your local machine.
2. Use pip to install any missing modules from the **interface.py** file. Also run ‘pip install bitsandbytes’ if it is not already installed.
3. Run the **interface.py** document from the root level of the repository.
4. Wait for all dependencies to download. This may take a few minutes (subsequent runs will be much quicker because the model won’t need to be re-downloaded).
5. Once “Running on local URL: http://localhost:7860/” appears in the terminal, click the link. This will open the application and is ready to use.