Fine-Tuning GPT-2 for a Customized Chatbot

Abstract

The purpose of this project was to fine-tune the GPT-2 language model using a custom dataset to create a chatbot capable of generating coherent and contextually relevant responses. GPT-2, developed by OpenAI, is a transformer-based model well-suited for natural language generation tasks. By training the model on a specific dataset of transcripts, we aimed to adapt its language generation capabilities to reflect the dataset's context. The project employed tools such as Hugging Face Transformers, the datasets library, Gradio, and Google Colab. Ultimately, we successfully trained and deployed a functional chatbot capable of interactive conversations based on the fine-tuned model.

Introduction

Recent advancements in natural language processing (NLP) have enabled the creation of powerful language models like GPT-2, which can generate human-like text. Fine-tuning pre-trained language models has become a popular approach for domain-specific applications, as it leverages the general knowledge of the pre-trained model and adapts it to specific tasks. This project focused on fine-tuning GPT-2 on a custom dataset of transcripts to build a chatbot tailored to the content of the transcripts.

The primary goal was to preprocess the dataset, fine-tune the GPT-2 model, and deploy an interactive chatbot using Gradio. By customizing GPT-2, we sought to explore the model's adaptability to a niche dataset and evaluate its performance in generating contextually relevant responses.

Methodology

The project involved several key steps, starting from data preparation to model fine-tuning and deployment. Each step was crucial in ensuring the chatbot's functionality and relevance to the dataset.

Data Preparation

The dataset for this project consisted of a text file containing raw transcripts. These transcripts included non-verbal annotations such as "[Laughter]" and other extraneous content that needed to be cleaned. A Python script was written to preprocess the dataset. This involved removing non-verbal cues, standardizing spaces and punctuation, and capitalizing the beginning of sentences.

To clean the text effectively, a custom function was implemented to handle tasks such as removing multiple spaces and ensuring proper sentence formatting. For instance, non-verbal annotations were identified using regular expressions and stripped from the text. Sentence boundaries were identified, and missing punctuation was added where appropriate. The cleaned dataset was then split into training and testing subsets, with 90% of the data allocated for training and 10% for testing.

Fine-Tuning GPT-2

The fine-tuning process was conducted using Hugging Face's Transformers library, which simplifies the implementation of state-of-the-art NLP models. The base GPT-2 model, which consists of 117 million parameters, was selected for this project due to its balance of computational efficiency and performance.

Hyperparameters for the training process were carefully chosen to optimize model performance. A batch size of 2 was used to accommodate the limitations of the GPU provided by Google Colab's free tier. The learning rate was set to 5e-5, and the model was trained for three epochs. Weight decay was set to 0.01 to prevent overfitting. The Hugging Face Trainer API was utilized to streamline the training process, handling tasks such as gradient accumulation and evaluation.

During training, the model processed the dataset in batches, adjusting its weights to minimize the loss function. The validation loss was monitored to evaluate the model's performance on unseen data, ensuring that the fine-tuning process did not overfit the training dataset.

Deployment with Gradio

After fine-tuning, the GPT-2 model was deployed using Gradio, a Python library for creating user-friendly interfaces for machine learning models. The chatbot interface allowed users to interact with the fine-tuned model by inputting text prompts and receiving responses generated by the model.

The Gradio interface was designed to be simple yet effective. Users could input their queries in a text box and receive the chatbot's responses in real time. The chatbot leveraged the fine-tuned GPT-2 model's capabilities to generate contextually relevant responses. The interface was deployed in Google Colab, providing a temporary public URL for interaction.

Results

The fine-tuned GPT-2 model successfully generated responses relevant to the context of the training dataset. During the training process, the loss metrics indicated steady improvement, with both training and validation losses decreasing over successive epochs. For example, training and validation loss values started high in the first epoch but reduced significantly by the third epoch, reflecting the model's learning progress.

Sample interactions with the chatbot demonstrated its ability to generate coherent and contextually appropriate responses. For instance, when queried about topics similar to those in the training transcripts, the chatbot produced responses that aligned with the dataset's tone and content. However, some responses lacked coherence when the input prompts deviated significantly from the training data's context.

Challenges and Limitations

Several challenges were encountered during the project. Setting up the environment and resolving dependency issues, such as the WandB API key prompt, required careful troubleshooting. Additionally, the small size of the dataset limited the chatbot's generalization capabilities. Responses were highly specific to the training data, making the chatbot less effective in handling out-of-context queries.

The computational limitations of the free-tier Google Colab environment also posed challenges. A small batch size had to be used to avoid memory issues, which increased training time. Moreover, training for only three epochs may have constrained the model's ability to fully learn the dataset's nuances.

Conclusion

This project successfully demonstrated the process of fine-tuning GPT-2 on a custom dataset and deploying the resulting model as an interactive chatbot. By leveraging tools such as Hugging Face Transformers and Gradio, we were able to create a chatbot capable of generating contextually relevant responses based on the training transcripts.

While the project achieved its primary objectives, there is room for improvement. Future work could involve training on a larger and more diverse dataset to enhance the chatbot's generalization capabilities. Additionally, experimenting with larger GPT-2 models or alternative architectures could further improve response quality. Finally, deploying the chatbot to a production-ready platform such as Hugging Face Spaces or Heroku would make it accessible for broader use.

References

1. Hugging Face Transformers Library: https://huggingface.co/transformers/
2. Gradio Library: https://gradio.app/
3. Python Libraries: transformers, datasets, torch, gradio
4. OpenAI GPT-2: https://openai.com/research/language-unsupervised