Building and Training Large Language Model

Dialogue Data Preparation
The datasets contain 10 entries, each presenting a conversation dialogue between two people. I have formatted each entry in a prompt-style format.

To create a batching process for training, I implemented a custom collate function to handle the specific requirements and formatting of the dialogue dataset. At a high level, this custom collate function performs the following actions:

1. It formats each data entry using a prompt-style format.
2. It tokenizes the formatted data.
3. It pads the training examples within each batch to the same length, while allowing different batches to have different lengths. This approach minimizes unnecessary padding by extending sequences to match the longest one in each batch, rather than the entire dataset.
4. It creates batches with target token IDs that correspond to the batch of input IDs. These target IDs are crucial because they represent what we want the model to generate. The target token IDs match the input token IDs but are shifted one position forward.
5. It assigns a placeholder value of -100 to all padded tokens to exclude them from contributing to the training cross-entropy calculation. This ensures that only meaningful data influences the model training.

I divided the dialogue dataset into training and validation sets, using a 70-30 split. Subsequently, I implemented a DialogueDataset class and applied data loaders to both the training and validation sets, which are necessary for large language model (LLM) fine-tuning and evaluation.

Model Selection
I chose the GPT-2 model due to its relatively simple architecture compared to other transformer architectures. As a decoder-style autoregressive model, it incorporates its previous outputs as inputs to predict the next word sequentially, making it ideal for text generation and conversational tasks. Specifically, I selected a pretrained GPT-2 LLM model over GPT-3. While GPT-3 shares the same fundamental model architecture as GPT-2, it scales to 1.5 billion parameters. GPT-2 is a more suitable choice for training on a single laptop, whereas GPT-3 necessitates GPU clusters.

I loaded the medium-sized model, which has 355 million parameters, instead of the smallest model with 124 million parameters. I chose the medium-sized model for finetuning on my personal laptop because its size is manageable enough to load and train effectively. I didn't select the smallest model because its limited capacity would prevent it from learning and retaining the complex patterns and nuanced behaviors necessary for the conversation task.

Training
I fine-tuned the LLM model on the dialogue dataset for 10 epochs. For training and validation losses, I used cross-entropy loss to compare the predicted distribution from the model (token probabilities) with the true distribution of labels (tokens in the dialogue dataset). I chose the

AdamW optimizer to update the weights because it is a more sophisticated optimizer that adaptively sets the learning rate and momentum and correctly applies weight decay.

Based on empirical observation of a small dataset, I set the hyperparameters for batch size to 2 and the number of epochs to 10.

Evaluation and Result

Based on the hyperparameter settings mentioned above, the training output demonstrates effective learning by the model, as indicated by the consistently decreasing training and validation loss values observed over 10 epochs. Specifically, for the dialogue dataset, the training loss decreased significantly from 3.45 to 0.040, while the validation loss also decreased substantially from 3.36 to 0.545.

I used the first part of Alex's dialogue as input for the model and, after fine-tuning, generated the remainder of the conversation between two people. The results of this generated conversation, along with the true conversation from the validation set (three cases), are provided in JSON format. While the model closely captures the dialogue between the two people, it is not yet perfect.

Analysis

To further improve the performance various strategies can be explore various:

- Adjust the hyperparameters during finetuning such as the learning rate, batch size, and number of epochs.
- Increase the size of training dataset or diversify the dialogue examples to cover a broad range of topics and styles.
- Experiment with different prompt and instruction formats to target IDs and track the model response.
- By using a larger pretrained model, we could further decrease validation loss, and the model would be better able to capture complex patterns and generate more accurate and elaborate conversations.