Finetuning Task Report

This report covers the steps and challenges encountered while fine-tuning the Gemma model using datasets and techniques suitable for limited-resource environments like Google Colab.

Dataset Preparation

Three different datasets were loaded from Hugging Face using Colab. The datasets were converted from Parquet format to pandas DataFrames. For each dataset, 5,000 samples were used for training and 2,000 for testing. These subsets were saved in .jsonl format.

Since Parquet files required special handling, a custom script was developed to properly parse and format the data. This script has been included in the GitHub repository. The datasets were then transformed into a format suitable for instruction tuning, using the input-instruction-response structure for both training and testing sets.

Model Fine-Tuning with QLoRA

The fine-tuning was performed using the Gemma model with the QLoRA PEFT technique. However, several issues were encountered, primarily due to Colab's environment. In particular, the bitsandbytes library required for quantized models was incompatible with the CUDA 12.4 runtime currently used in Colab notebooks. This limitation does not exist on standalone servers.

As a workaround, fine-tuning was conducted using the non-quantized version of LoRA. However, Colab's limited GPU resources made it difficult to use larger batch sizes or run extended training. Despite this, the main objective was to demonstrate the fine-tuning process and technical proficiency, which was achieved successfully.

Alternative: DORA PEFT

Since Galore format was not supported, I experimented with an alternative PEFT method: DORA. This method is similar to LoRA but introduces a dropout layer and a configuration flag use_dora=True. This allowed for a comparative study of two distinct fine-tuning strategies under constrained resources.

Training Optimization Techniques

Several strategies were employed to maximize training efficiency:

- Data Caching in RAM: Reduced IO bottlenecks by avoiding repeated file reads from local storage.
- Gradient Accumulation: Simulated larger batch sizes by using a batch size of 4 with gradient_accumulation_steps=2, effectively emulating a batch size of 8.
- Tokenizer Configuration: After experimenting with different values, max_length=256 was chosen to balance GPU memory use.
- Mixed Precision: Training used float16, which is more efficient on Colab's T4 GPUs.

These optimizations reduced the training time significantly. While the initial plan would take 9 hours for 2 epochs, the optimized setup completed the same in 1 hour and 20 minutes - all while making training feasible on Colab.

Evaluation

Evaluation was conducted using BLEU and ROUGE metrics. Results, along with the fine-tuning code and evaluation notebooks, have been shared on GitHub.