Parameter-Efficient Fine-Tuning of RoBERTa on AGNEWS using LoRA

Utkarsh Mittal(um2113), Rushabh Bhatt(rb5726), Aryan Ajmera(aa1290)
GitHub Repository: https://github.com/AryanAjmera18/proj

Abstract

This paper presents our work on fine-tuning RoBERTa using Low-Rank Adaptation (LoRA) for the AGNEWS text classification task under a constraint of fewer than 1 million trainable parameters. Our final model achieved a private leaderboard score of 0.8500 and a public score of 0.85475 with approximately 980K trainable parameters. We outline our methodology, architectural choices, experimentation with QLoRA, Pros and Cons of our approach and lessons learned.

Introduction

Fine-tuning large language models can be resource-intensive. LoRA offers an efficient alternative, introducing trainable low-rank matrices into frozen pretrained models. In this project, we fine-tuned a ‘roberta-base‘ model on the AGNEWS dataset using LoRA while ensuring the total trainable parameters remained under one million.

Methodology

Dataset and Preprocessing

We used the AGNEWS dataset from Hugging Face. Tokenization was performed using the RoBERTa-base tokenizer, truncating sequences to 256 tokens. Outliers beyond the 1st and 99th percentile in token length were filtered to reduce noise.

Model Architecture and LoRA Design

We adapted the **RobertaForSequenceClassification** model with the PEFT library to include LoRA layers. Our configuration:

- Rank (r): 7
- Alpha: 77
- Target Modules: **query**, **key**, **value**
- LoRA Dropout: 0.01

Training Strategy

We trained the model using the following hyperparameters:

- Optimizer: AdamW
- Learning Rate: 3e-5
- Epochs: 6
- Batch Size: 64
- Weight Decay: 0.1
- Scheduler: Cosine Annealing

Evaluation was performed at step intervals with **load_best_model_at_end=True**. Accuracy was the primary metric, measured using Hugging Face’s **evaluate** library.

Attempted Use of QLoRA

We experimented with QLoRA for 4-bit quantization to reduce memory usage. However, compatibility issues on NYU HPC (bitsandbytes-related) prevented successful deployment, so we proceeded with standard LoRA.

Results

- Validation Accuracy: 0.938817
- Private Test Score: 0.8500
- Public Test Score: 0.85475
- Trainable Parameters: 980,740

Pros and Cons of Our Approach

Pros

- LoRA allowed us to train a performant model with only 980,740 trainable parameters.
- The PEFT framework made it simple to integrate LoRA into existing Transformer models.
- Despite strict constraints, the model achieved over 93% accuracy on validation set and 85% on test set.
- Extensive community and library support via Hugging Face and PEFT accelerated our development.

Cons

- Only a small subset of parameters were updated, potentially limiting expressiveness.
- Significant tuning was required to stay under 1M parameters while achieving good accuracy.

- We could not take advantage of QLoRA due to system compatibility problems.
- Without quantization, we may have used more memory than ideal for deployment.

Reproducibility

Our GitHub repository contains training scripts, evaluation notebooks, logs, and instructions for reproducing our experiments.

Lessons Learned

- LoRA enables competitive performance under tight parameter budgets.
- Data filtering (based on token length) can significantly improve generalization.
- QLoRA is promising, but platform compatibility remains a barrier.
- Hyperparameter tuning, especially of LoRA-specific parameters, is critical.
- Robust validation strategy and logging are essential for reproducibility.

References

1. Hugging Face Transformers: https://huggingface.co/transformers/
2. AGNEWS Dataset: https://huggingface.co/datasets/ag_news
3. LoRA Paper: https://arxiv.org/abs/2106.09685
4. PEFT Library: https://github.com/huggingface/peft