Fine-Tuning DistilBERT Using LoRA with the IMDB Dataset

- Preetham Kumar Dundigalla | GitHub | LinkedIn | Portfolio

Introduction

This document outlines the process of fine-tuning the DistilBERT model using Low-Rank Adaptation (LoRA) on the IMDB dataset. The primary goal is to achieve efficient training and improved performance on sentiment analysis tasks, specifically classifying movie reviews as positive or negative. Fine-tuning is a crucial technique in transfer learning, where a pre-trained model is adapted to a specific task by training it on a smaller dataset.

Project Overview

- **Model Used**: DistilBERT
- **Dataset**: IMDB Dataset (movie reviews with sentiment labels)
- **Technique**: Low-Rank Adaptation (LoRA) for efficient model training

Theoretical Background on Fine-Tuning

Fine-tuning involves taking a pre-trained model that has learned to capture general features from a large dataset and then adjusting it to perform well on a specific task. This approach leverages the knowledge embedded in the pre-trained model while requiring fewer resources than training a model from scratch.

Low-Rank Adaptation (LoRA)

LoRA is a method that allows for efficient adaptation of large language models by introducing low-rank matrices into the layers of the model. This technique reduces the number of trainable parameters, thus making fine-tuning more memory efficient. By freezing the original weights of the model and only training the low-

rank adapters, LoRA minimizes the risk of overfitting and reduces the computational burden.

Set-up

1. Change runtime to T4GPU if using google colab.
2. Install **torch**, **transformers**, **accelerate** and **peft**
3. Understand the GPU and system storage or RAM requirments neede for the project
4. If dataset is big, try to shift to QLoRa which tries to optimize the data and allocate memory space.

Key Terms

1. **Fine-Tuning**: The process of adapting a pre-trained model to a specific task by training it on a smaller, task-specific dataset, enhancing its performance while utilizing previously learned features.
2. **DistilBERT**: A lightweight version of BERT (Bidirectional Encoder Representations from Transformers) designed for efficiency, maintaining competitive performance in natural language processing tasks with fewer parameters.
3. **LoRA (Low-Rank Adaptation)**: A technique that enables efficient fine-tuning of large models by introducing low-rank matrices into the model architecture, reducing the number of trainable parameters while preserving model performance.
4. **IMDB Dataset**: A widely used dataset for sentiment analysis that consists of movie reviews labeled as positive or negative, commonly utilized for benchmarking natural language processing models.
5. **Tokenizer**: A component that converts raw text into tokens, enabling the model to process and understand textual data by breaking it down into manageable units, typically words or subwords.

6. **Training Arguments**: Configuration parameters that guide the training process, including batch size, number of epochs, and learning rate, crucial for optimizing model performance and resource management.
7. **Trainer**: A high-level interface provided by Hugging Face Transformers that simplifies the training and evaluation process for machine learning models, facilitating the integration of training logic and metrics.

Data Preparation

The IMDB dataset was processed using pandas to prepare the data for training. Initially, the entire dataset was considered, but it was found that reducing the dataset size by 10% helped in managing memory issues.

```
import pandas as pd

# Load dataset
df = pd.read_csv('IMDB Dataset.csv', on_bad_lines='skip')
df['label'] = df['sentiment'].map({'positive': 1, 'negative': 0}
df.dropna(inplace=True)  # Remove missing values

# Reducing dataset size by 10%
df_reduced = df.sample(frac=0.1, random_state=42)  # Randomly sa

# Splitting dataset
train_texts = df_reduced['review']
train_labels = df_reduced['label']
```

Model Configuration

Dependencies

Install the required libraries, including Hugging Face Transformers. An attempt was made to install LoRA directly from GitHub, but it encountered issues. Thus, further exploration was done without it for the current setup:

```
pip install torch transformers datasets
# pip install git+https://github.com/timdettmers/qlora.git  # Th
```

Training Arguments

Define the training parameters to optimize memory usage. The choice of batch size, learning rate, and number of epochs is critical in controlling overfitting and convergence speed during fine-tuning.

```
from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=1,  # Total number of training epochs
    per_device_train_batch_size=4,  # Reduced batch size
    per_device_eval_batch_size=4,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    evaluation_strategy="epoch",
)
```

Model Fine-Tuning

Fine-tune the DistilBERT model with the reduced dataset. The training loop was set up using the Trainer class from Hugging Face Transformers, which provides a streamlined interface for training models.

```
from transformers import Trainer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
```

```
    eval_dataset=eval_dataset,
    compute_metrics=compute_metrics,
)

# Start training
trainer.train()
```

Evaluation

After training, the model was evaluated on a validation set, and predictions were made to assess performance. Evaluation metrics such as accuracy, precision, and recall are essential to understand how well the model generalizes to unseen data.

```
# Make predictions
predictions = trainer.predict(test_dataset)
```

Summary Report Generation

Generate a report summarizing the model performance:

Model Summary

- **Model Architecture**: DistilBERT
- **Training Epochs**: 1
- **Final Training Loss**: [0.6167510747909546]

Conclusion

The fine-tuning of DistilBERT using LoRA on the IMDB dataset demonstrates an effective approach for sentiment analysis, allowing for efficient training and high performance. Reducing the dataset size by 10% proved beneficial in managing memory issues. The application of LoRA facilitates low-resource training while preserving the model's ability to generalize. Future work can include experimenting with larger datasets and additional epochs to improve model accuracy.