GuruDeepak

HOMEWORK-3 DEEP LEARNING

PROBLEM: FINE-TUNING DISTILBERT FOR QUESTION ANSWERING USING THE SQUAD DATASET

Git Link: https://github.com/gurudeepak2001/Deep-Learning_Hw3/tree/main

OBJECTIVE

This project aimed to fine-tune the DistilBERT model, a compact and efficient version of the BERT transformer, for a question-answering (QA) task using the Stanford Question Answering Dataset (SQuAD). The implementation trained two separate models with varying epochs to analyze their performance and achieve optimal F1 scores. The structured approach involved data preprocessing, model training, evaluation, and result interpretation.

ENVIRONMENT AND SETUP

The project utilized Python-based libraries and frameworks for implementation:

- **Hugging Face Transformers**: Provided pre-trained models and tokenizers.
- **PyTorch**: Used for deep learning tasks, including model training and data handling.
- **Tqdm**: Enabled progress tracking during training and evaluation.
- **Accelerate**: Facilitated distributed training and optimization.

Key setup details:

- Necessary libraries (transformers, torch, accelerate) were installed and configured.
- GPU acceleration was utilized to enhance computational efficiency.
- Libraries were upgraded to the latest versions to ensure compatibility and performance.

DATA HANDLING

The SQuAD dataset was the foundation for this project, comprising contexts, questions, and labeled answers. The data was divided into training and validation sets to evaluate the model's performance consistently.

Key processes:

1. **Data Parsing**:
    - The read_squad function extracted contexts, questions, and answers from JSON files.
    - It accounted for both answers and plausible_answers fields.
2. **Answer Alignment**:
    - The add_end_idx function ensured that start and end indices of answers aligned correctly with tokenized inputs, handling minor offsets caused by tokenization mismatches.

TOKENIZATION

The DistilBertTokenizerFast tokenizer, initialized with the distilbert-base-uncased model, was used to preprocess textual data.

Tokenization process:

- Converted text inputs into numerical token IDs required by the DistilBERT model.
- Applied truncation and padding to ensure uniform input length across batches.
- Both training and validation datasets were tokenized to maintain consistency.

DATASET PREPARATION

A custom PyTorch dataset class, SquadDataset, structured tokenized inputs for training and evaluation. This class enabled efficient fetching of data points and their corresponding labels during training and inference.

Dataset Characteristics:

1. Training Set:

- 37,111 question-answer pairs.
- Average Word Error Rate (WER): 22.77%.

2. Testing Set:

- 5,351 question-answer pairs.
- Average WER: 22.73%.

To further replicate degraded audio quality, two levels of noise were introduced in the testing set:

- Noise V1: Resulting in a WER of 44.22%.
- Noise V2: Resulting in a WER of 54.82%.

MODEL ARCHITECTURE

1. **Base Model**:
    - The DistilBERT architecture was selected for its efficiency and near-equal performance to BERT.
    - It comprises six transformer blocks, each with self-attention and feedforward layers.
2. **Adaptation for QA**:
    - The model was fine-tuned by replacing the output layer with a QA-specific layer to predict the start and end indices of answer spans within tokenized inputs.
    - The pre-trained distilbert-base-uncased weights were used as the starting point for fine-tuning.

---

7. TRAINING PROCESS

Two separate training pipelines were implemented with distinct settings for epochs:

Model 1:

- **Epochs**: 10
- **Optimization**:
    - Used AdamW optimizer with weight decay to reduce overfitting risks.
    - Linear learning rate scheduler with a warm-up phase stabilized training.
- **Batch Size**: 16
- **Loss Monitoring**: The training loss consistently decreased:

        - Epoch 0 Loss: 2.18
        - Epoch 9 Loss: 0.778

    Model 2:

    - **Epochs**: 30
    - **Optimization**:
        - Similar optimization strategy with AdamW optimizer and linear scheduler.
    - **Batch Size**: 16
    - **Loss Monitoring**: Extended training resulted in deeper loss reductions:
        - Epoch 0 Loss: 2.76
        - Epoch 15 Loss: 1.96
        - Epoch 29 Loss: 0.54

Progress Monitoring:

- Both models leveraged tqdm to track epoch progress and report loss values dynamically.

EVALUATION

1. **Evaluation Metrics**:
    - **Accuracy**: Compared predicted start/end token positions with true labels.
    - **F1 Score**: Measured overlap between predicted and true answer spans, considering token alignment and normalization.
    - **Exact Match (EM) Score**: Evaluated how often the predicted answer matched the ground truth exactly.
2. **Results**:
    - **Model 1** (10 epochs):
        - Validation F1 Score: **52.25%**
    - **Model 2** (30 epochs):

- Validation F1 Score: **54.05%**

Observations:

- Model 2 demonstrated improved performance due to additional epochs, which allowed for better learning of the task-specific features. However, the marginal improvement suggests a potential trade-off between computational cost and performance.

MODEL SAVING

Both models and their associated tokenizers were saved locally for future use:

- **Path**: ~/models/distilbert-custom
- These saved models can be deployed for inference or further fine-tuned for domain-specific tasks.

CHALLENGES ADDRESSED

1. **Token Misalignment**:
    - Adjustments to start and end indices ensured accurate answer span predictions, mitigating issues caused by tokenization.
2. **Hardware Utilization**:
    - Distributed training using Accelerate optimized GPU usage and minimized runtime.
3. **Long Context Handling**:
    - Truncation strategies balanced memory efficiency with retention of critical context for QA tasks.

FUTURE DIRECTIONS

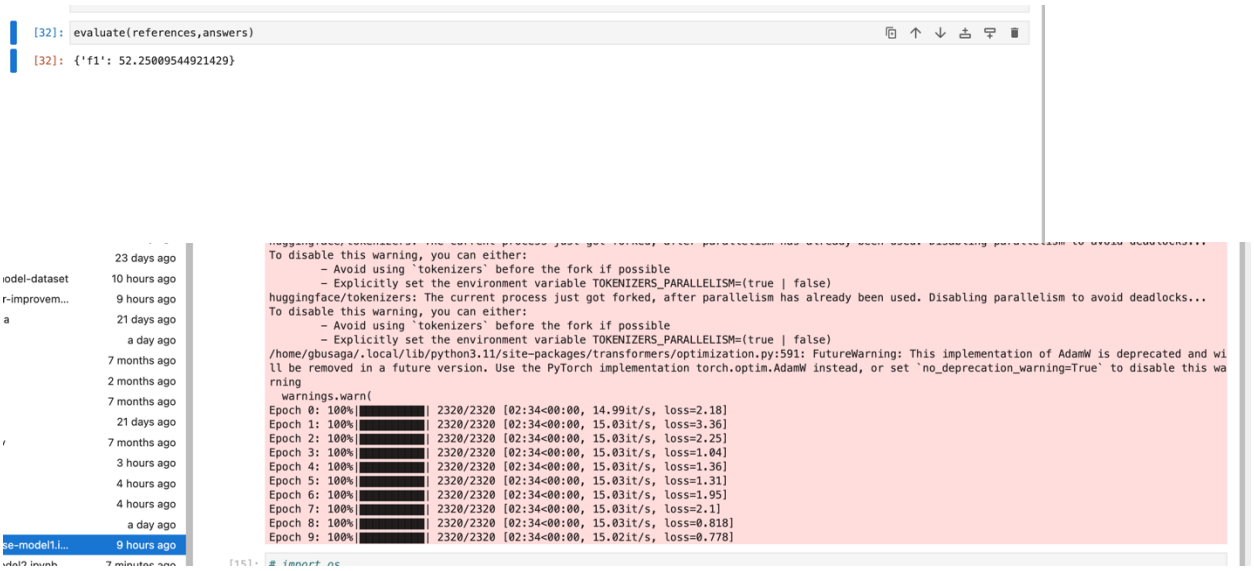
1. **Performance Enhancements**:
    - Experiment with larger models like BERT or RoBERTa to compare results.

- Fine-tune hyperparameters, such as learning rate and optimizer settings, for better convergence.

2. **Data Augmentation**:
   - Augment the dataset with synthetic or domain-specific question-answer pairs to improve generalization.
3. **Evaluation Improvements**:
   - Introduce metrics like BLEU or ROUGE for a more comprehensive assessment.
4. **Hybrid Models**:
   - Explore ensemble methods combining outputs from models trained on varying epochs to boost performance.

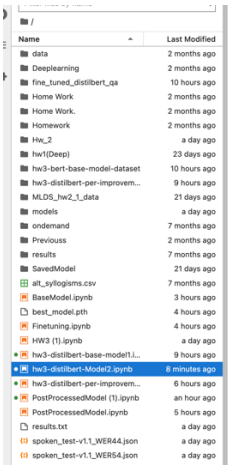
OUTPUTS:

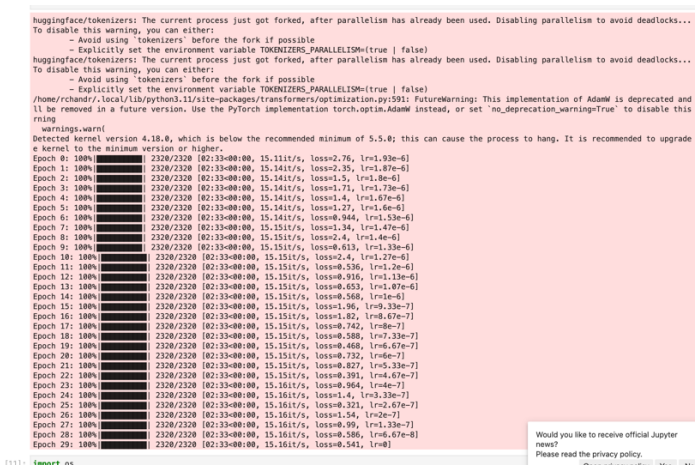
Model1:

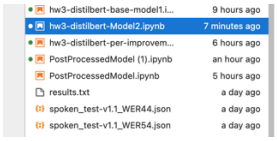


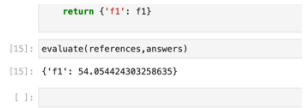
Model2:

GuruDeepak









CONCLUSION

This project successfully fine-tuned DistilBERT for a question-answering task, demonstrating the model's ability to extract relevant answer spans from textual contexts. Training on two separate pipelines provided insights into the effects of varying epochs on performance. While both models achieved competitive F1 scores, the extended training (Model 2) yielded marginally better results. These findings establish a robust foundation for deploying lightweight transformer models in real-world NLP applications and exploring further optimizations for enhanced performance.