Jordyn Brooks
CPSC 8430 - Deep Learning
Homework 3
Github link: https://github.com/brooksjk/Deep_Learning - under folder HW3

The purpose of this homework is to create a question-answering model that uses the BERT (Bidirectional Encoder Representations from Transformers) architecture and is trained on the SQuAD (Stanford Question Answering Dataset) to extract answers from contexts depending on given questions.

Bert Model:

BERT is a transformer-based model that achieves cutting-edge performance on a variety of natural language processing tasks, including question answering. Unlike traditional models, BERT processes words in relation to all other words in a phrase rather than individually, allowing it to better grasp context.

SQuAD Dataset:

The SQuAD dataset consists of questions based on a set of Wikipedia articles. Each question is accompanied by a background paragraph that offers the solution. The objective is to correctly guess the answer's start and end places inside the context.

***Methodology*:**

Pre-trained model: bert-base-multilingual-uncased

- Preprocessing

  The preprocessing phase involves several key steps:

  1. Loading the SQuAD Dataset: The SQuAD dataset is loaded and parsed to extract contexts, questions, and answers.

2. Answer Adjustment: End indices of the answers are adjusted to ensure proper alignment with the text, which is essential for accurate answer span predictions.

- Tokenization

  For tokenization, the BertTokenizerFast class is employed. This tokenizer efficiently converts input text into the appropriate format for the BERT model.

  The tokenization process includes:

    - *Truncation*: Input texts that exceed the maximum model length are truncated.
    - *Padding*: Shorter texts are padded for consistent tensor shapes during training.
    - *Encoding*: Texts are converted into input IDs, attention masks, and token type IDs required by BERT.

- Data Preparation

  - Answer start and end positions are mapped to token IDs. This is done to allow the model to predict answer spans effectively. Additionally, any unmatched tokens are adjusted using a shifting approach to ensure they are within the context bounds.

- Transformation

  BertForQuestionAnswering is specifically designed to handle question-answering tasks, providing mechanisms to compute loss for both start and end position predictions.

  The architecture of this class includes:

  - *Input Embeddings*: Combines token, segment, and position embeddings.
  - *Transformer Layers*: A sequence of transformer blocks that capture contextual relationships.
  - *Output Layer*: A final layer that predicts the start and end positions of the answer within the context.

- Training

  The training process follows these key components:

  - Optimizer: The AdamW optimizer, which includes weight decay regularization, helps improve the model's generalization.
  - Learning Rate Schedule: A learning rate scheduler with a warm-up phase is used to stabilize the training initially. Here, a starting learning rate of 1e-5 was chosen, and the model is trained for 5 epochs.
  - Training Loop
    - Within each epoch:
      - Batch Processing: The model computes the loss based on predicted versus true start and end positions.

        - Backpropagation: Losses are backpropagated to adjust model parameters.

    - Evaluation: To monitor performance, evaluation is conducted periodically on a validation set, with metrics such as F1 score and exact match ensuring the model is not overfitting.

Evaluation and Testing

Evaluation is crucial to measure the model's question-answering accuracy:

- *Exact Match*: Measures the proportion of exact matches between predictions and ground truth.
- *F1 Score*: A balance between precision and recall, computed based on overlapping words between predicted and ground-truth answers.
- *Accuracy Calculation*: Predictions are evaluated by comparing the start and end token positions to the true answer locations.

Evaluation Functions:

- *Normalization*: To standardize the evaluation, a normalization function removes punctuation, lowercases text, and handles spacing.
- *Prediction and Reference Extraction*: Helper functions extract predictions and references by decoding tokenized IDs to readable text, providing accurate benchmarks for performance metrics.

Evaluation Results:

The primary evaluation metric for this model is the F1 Score, as specified in the project requirements. The F1 score balances precision and recall, reflecting the model's ability to capture partial overlaps between predicted answers and true answers.

Obtained Results:

- F1 Score: 54.67

To gain additional insight, an Exact Match score was also calculated, yielding a score of 35.42. This supplemental metric provides the proportion of answers that exactly match the ground truth, offering an additional perspective on the model's performance.

Model Testing and Custom Functions

To validate the model on unseen data (squad_files/spoken_test-v1.1.json), a test loop evaluates start and end position predictions, comparing them to actual values. The get_predictions and get_references functions gather predictions and references for F1 and exact match calculations.

Execution Environment

The model training and evaluation were executed using an accelerated GPU environment through torch.device, and the Accelerator from the Hugging Face accelerate library was used to streamline distributed training.