Fine-Tuning BERT LLM model for Question Answering on PDF Documents

Overview

This document provides a detailed explanation of the process to fine-tune a BERT model for question answering (QA) tasks and the subsequent application of this fine-tuned model to extract answers from PDF documents. The workflow involves the following steps:

1. Installing and importing necessary libraries.
2. Loading and preprocessing the SQuAD dataset.
3. Tokenizing the dataset using the BERT tokenizer.
4. Fine-tuning the BERT model on the QA dataset.
5. Evaluating the fine-tuned model.
6. Extracting text from PDF documents and applying the model to answer questions.

1. Installing Required Libraries

First, install the necessary libraries using the following commands:

```
!pip install accelerate -U
!pip install transformers[torch]
!pip install transformers PyMuPDF
!pip install datasets
```

```
Requirement already satisfied: accelerate in /usr/local/lib/python3.10/dist-packages (0.32.1)
Requirement already satisfied: numpy<2.0.0,>=1.17 in /usr/local/lib/python3.10/dist-packages (from accelerate) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from accelerate) (24.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from accelerate) (6.0.1)
Requirement already satisfied: torch>=1.10.0 in /usr/local/lib/python3.10/dist-packages (from accelerate) (2.3.0+cu121)
Requirement already satisfied: huggingface-hub in /usr/local/lib/python3.10/dist-packages (from accelerate) (0.23.4)
```

2. Importing Necessary Modules

The following imports are essential for the code to function:

```
[ ]  import json
     import torch
     from transformers import BertTokenizerFast, BertForQuestionAnswering, Trainer, TrainingArguments, pipeline
     from datasets import Dataset
     import numpy as np
     import fitz  # PyMuPDF
     from sklearn.model_selection import train_test_split
     import cProfile
     import pstats
```

3. Loading and Preprocessing the SQuAD Dataset – Kaggle data

The function read_squad is defined to read the SQuAD JSON file and return a dataset structured as dictionaries containing contexts, questions, and answers.

```python
# Function to read the JSON file and return a dataset
def read_squad(path):
    with open(path, 'r') as f:
        squad_dict = json.load(f)

    contexts = []
    questions = []
    answers = []
    for group in squad_dict['data']:
        for passage in group['paragraphs']:
            context = passage['context']
            for qa in passage['qas']:
                question = qa['question']
                for answer in qa['answers']:
                    contexts.append(context)
                    questions.append(question)
                    answers.append(answer)

    return {
        'context': contexts,
        'question': questions,
        'answers': answers
    }
```

4. Splitting the Dataset

The dataset is divided into training and validation sets using the train_test_split function from sklearn. So that we get to train the model on a comparitivly large data scale.

5. Tokenizing the Dataset

Initialized the BERT tokenizer.

```
Train Dataset: Dataset({
    features: ['context', 'question', 'answers'],
    num_rows: 70079
})
Validation Dataset: Dataset({
    features: ['context', 'question', 'answers'],
    num_rows: 17520
})
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
Map: 100% 70079/70079 [01:57<00:00, 1060.95 examples/s]
Map: 100% 17520/17520 [00:21<00:00, 920.85 examples/s]
Tokenized Train Dataset: Dataset({
    features: ['context', 'question', 'answers', 'input_ids', 'token_type_ids', 'attention_mask', 'start_positions', 'end_positi
    num_rows: 70079
})
Tokenized Validation Dataset: Dataset({
    features: ['context', 'question', 'answers', 'input_ids', 'token_type_ids', 'attention_mask', 'start_positions', 'end_positi
    num_rows: 17520
})
```

6. Profiling and Preprocessing the Dataset

To profile the preprocessing function and identify potential bottlenecks, use cProfile and pstats

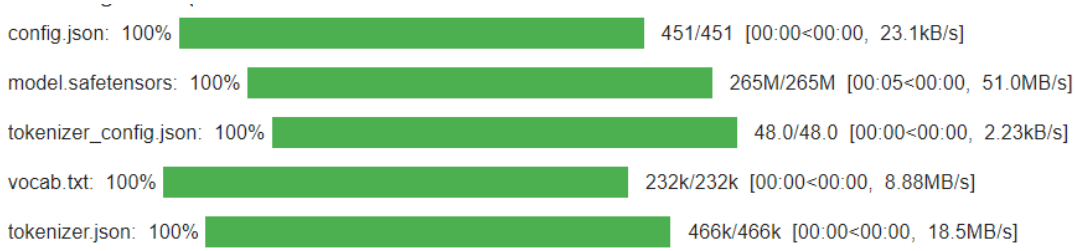
7. Fine-Tuning the Model

After Loading the pre-trained BERT model, Define a function to compute evaluation metrics, specifically the exact match between the predicted and actual start and end positions of answers.

8. Extracting Text from PDF and Answering Questions

Then, we Define a function to extract text from PDF documents using PyMuPDF. Then we Define a function to get answers from the extracted text using the fine-tuned model.

Example usage:




```
Question: What is the main topic of the document?
Answer: Global Warming
Question: Who is the author?
Answer: Umair Shahzad
University of Sunderland
Question: What are the key findings?
Answer: Climate, fossil fuels, deforestation, global warming, alternative energy sources
```

Future Aspects

The methodology described above for fine-tuning BERT for question answering tasks can be extended and improved by leveraging more advanced techniques and larger datasets. Here are a few future directions to consider:

Creating a Large Language Model (LLM) from Scratch

1. **Data Collection and Preparation**: Gather extensive and diverse datasets to cover a wide range of topics and contexts. This data should be cleaned, preprocessed, and structured appropriately.
2. **Model Architecture**: While BERT is highly effective, experimenting with more recent architectures such as customising transformer models tailored to specific tasks could yield better results.