Report week 3

Sreshtha Lahiri

24b0614

My code in this sentiment pipeline file is using Hugging Face's datasets and transformers libraries to fine-tune an already trained BERT model (on the imdb dataset). This dataset that I'm using consists of movie reviews which have to be categorized as either positive (1) or negative (0), making this a binary classification. I'm using the datasets library to load and then make subsets of the imdb data, then convert it to a pandas dataframe (this I did to avoid compatibility issues with NumPy 2.0 during label extraction and I'm using this point in the challenges I faced section as well).

The berttokenizer is used to preprocess the text data. It transforms raw reviews into tokenized input (like in sequences). I then added a classification head to the BERT model using bertforsequenceclassification with the code – "num_labels=2 ". Then this model is fine-tuned using a pytorch training loop and after that is is evaluated using accuracy and F1-score to assess classification performance.

I chose this design for its modularity, and scalability. Some challenges I faced included managing issues between Hugging Face datasets and NumPy 2.0, which I solved by converting datasets to pandas dataframes before preprocessing , this I did because the list and tolist attributes did not work. Also I had problems loading the dataset of imdb but I solved it later on.

Some challenges that might occur and how I'll fix those :

1. Fine-tuning a very large pre-trained model (like bert-base-uncased in this case) is very time consuming , especially on this type of imdb dataset (that has around 50,000 samples). Running this code on a CPU can lead to long training times and subsequently memory issues. So my solution for this is to use google colab with a gpu runtime (runtime > change runtime type > gpu). We can also train on a small subset (like around 1000 samples), then scale up.
2. A known issue with Hugging Face datasets and NumPy 2.0 causes crashes when coding on dataset fields like label. This is due due to copy=False. My solution to this problem which is something I used here as well is to convert datasets to pandas dataframes using .to_pandas() before extracting text and labels (this avoids the array and numpy conversion problems).
3. Reviews of this dataset vary in length. So without proper truncation, inputs may exceed BERT's maximum input size which is 512 tokens. So what we can do is apply tokenizer(..., truncation=True, padding=True, max_length=256) to standardize input shapes.
4. BERT is large (approximately 420MB), which can be challenging for deployment in resource-constrained environments. So we can distill the model using distilbert for lighter inference if needed.