NLP Homework 3: Pre-trained Transformer-based Models

Emma Ozias

1. Dataset

My task was to fine-tune two pre-trained models using a dataset. Once, I fine-tuned them, I wanted to evaluate how accurate they are at predicting the label of an item in my dataset. I used the Wikipedia toxic comment dataset. The dataset is comprised of comments that have been posted on Wikipedia. Each comment is assigned a label: 0 – not offensive, or 1 – offensive. The input for my task was the comment, and the output was the label. I chose to use accuracy as my evaluation metric. It compares the predicted label to the actual label in the dataset. A higher accuracy score means that the model is doing a good job predicating whether a comment is offensive or not offensive. This dataset has a train set, a balanced train set, a validation set, and a test set. I chose to use the balanced train split to fine tune my models because the regular train split is very unbalanced, it is made up of mostly not offensive comments. Then, I used the test split to evaluate my models.

	Balanced Train	Train	Test	Validation
Size (comments)	25,900	128,000	64,000	31,900

2. Fine-tuned Models

I used the google-bert/bert-base-cased model and the distilbert/distilbert-base-multilingual-cased model.

My first model (google-bert/bert-base-cased) has 109 million parameters. This is how the model was trained: “BERT is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labelling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts.”

(Hugging Face model card). The training datasets used were BookCorpus, a dataset consisting of 11,038 unpublished books and English Wikipedia (excluding lists, tables and headers). The compute requirements for pretraining were "4 cloud TPUs in Pod configuration (16 TPU chips total) for one million steps with a batch size of 256. The sequence length was limited to 128 tokens for 90% of the steps and 512 for the remaining 10%. The optimizer used is Adam with a learning rate of 1e-4, $\beta 1=0.9$ and $\beta 2=0.999$, a weight decay of 0.01, learning rate warmup for 10,000 steps and linear decay of the learning rate after." (Hugging Face model card).

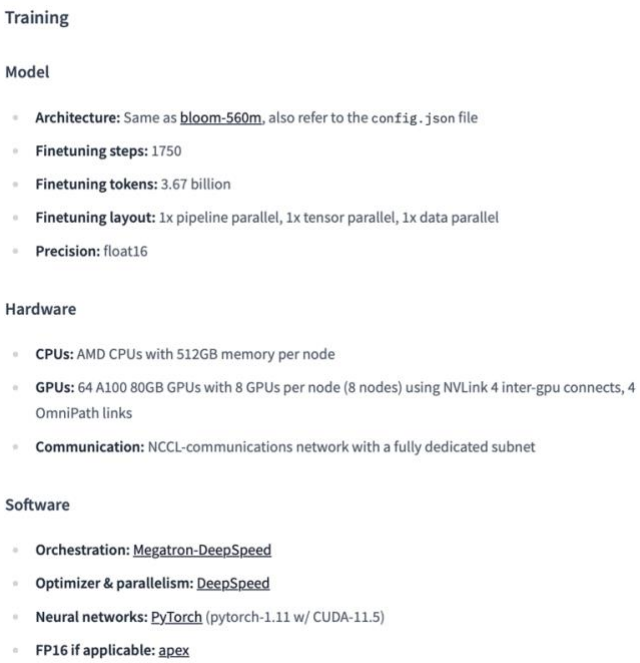
My second model (distilbert/distilbert-base-multilingual-cased) has 135 million parameters. This is how the model was trained: "The model is trained on the concatenation of Wikipedia in 104 different languages listed here. The model has 6 layers, 768 dimension and 12 heads, totalizing 134M parameters (compared to 177M parameters for mBERT-base)." (Hugging Face model card). However, I could not find any information on the compute requirements for this model.

3. Zero-shot Classification

The two models that I used for zero-shot classification were mistralai/Mistral-7B-Instruct-v0.1 and bigscience/bloomz-560m.

My first model (mistralai/Mistral-7B-Instruct-v0.1) has 7.24 billion parameters. It is a instruct fine-tuned version of the Mistral-7B-v0.1 model. The original model was trained on publicly available conversation datasets. The instruct version of the model was fine-tuned on instruction datasets publicly available on HuggingFace. I could not find compute requirements for this model.

My second model (bigscience/bloomz-560m) has 559 million parameters. This model has been trained and fine-tuned on multilingual datasets. This model is meant to translate text into another language. The compute requirements found on Hugging Face are below:



Training

Model

- **Architecture:** Same as bloom-560m, also refer to the `config.json` file
- **Finetuning steps:** 1750
- **Finetuning tokens:** 3.67 billion
- **Finetuning layout:** 1x pipeline parallel, 1x tensor parallel, 1x data parallel
- **Precision:** float16

Hardware

- **CPUs:** AMD CPUs with 512GB memory per node
- **GPUs:** 64 A100 80GB GPUs with 8 GPUs per node (8 nodes) using NVLink 4 inter-gpu connects, 4 OmniPath links
- **Communication:** NCCL-communications network with a fully dedicated subnet

Software

- **Orchestration:** Megatron-DeepSpeed
- **Optimizer & parallelism:** DeepSpeed
- **Neural networks:** PyTorch (pytorch-1.11 w/ CUDA-11.5)
- **FP16 if applicable:** apex

The formula that I used to prompt both models: comment from Wikipedia dataset + “Is this text offensive? Answer only with the tokens Yes or No. Answer: ”. Originally, I did not have “Answer: ” at the end of my prompt. I had to add that in because the model that I was originally using (GPT-2) was not answering with yes or no. Instead, it was writing a sentence about the comment and what does the word offensive mean. I think that adding in the “Answer: ” part of the prompt made it easier for the model to respond in the correct way.

4. Baselines

I did not use code to calculate my baselines. I met with Professor Wilson for office hours, and he told me that both of my baselines are 50%. These are my baseline values because I am using the balanced train split of my dataset. Due to my training set being balanced, it is reasonable to assume that there is a 50% chance that a comment is offensive and a 50% chance that a comment is not offensive when choosing randomly. Also, there is not one majority or target class. Instead, there are two classes: offensive and not offensive which each make up 50% of the training data.

5. Results

	Accuracy
google-bert/bert-base-cased	0.818
distilbert/distilbert-base-multilingual-cased	0.776
mistralai/Mistral-7B-Instruct-v0.1	0.788
bigscience/bloomz-560m	0.895
Multinomial Naïve Bayes Classifier (BOW baseline)	0.843
Random baseline	0.50
Majority class baseline	0.50

All of my models are performing with high accuracy values. This means that for toxic comment detection, a user has multiple options: fine-tuning a pre-trained model, using zero-shot classification, or using a multinomial naïve bayes classifier. By implementing any of these options, the user can expect to have a high accuracy value for toxic comment detection. Based on the accuracy values, it seems that each option is just as good as the other options. There is not one choice that has a significantly lower accuracy value than the rest. However, it is important to note that the zero-shot classifiers were tested on a smaller test dataset. I used the entire test dataset (64,000 comments) to evaluate my fine-tuned models and my multinomial naïve bayes classifier. However, I was only able to use 1000 comments to evaluate the performance of zero-shot classification because of hardware limitations. Due to this, it may be best to use a fine-tuned pre-trained model or a multinomial naïve bayes classifier for predicting whether or not a comment is offensive.

6. Reflection

I learned how to use Google Colab while completing this assignment. I have never written code on that platform before, but I had to use it due to hardware limitations. I think that I could end up using Google Colab again in the future. Also, I learned more about prompt engineering. I did not realize that you have to format each prompt in a specific way for the model to be able to understand the prompt. Additionally, I learned

how to fine-tune a pre-trained model. This could be helpful for my final project, or my upcoming machine learning course that I am taking this summer.

The only thing that was unexpected was how well zero-shot classification works. I thought that there would be a significant gap in the performance of a model using zero-shot classification and a model that was fine-tuned for this task. However, there was not a gap as I expected.

I faced many challenges with this assignment. At first, I was using the go_emotions dataset, but I kept getting an error when trying to fine-tune my model. As soon as I switched to the Wikipedia Toxic Comment dataset, my code started working. My next challenge was with section 2.3 of the assignment. I could not get any models to run on my computer. After talking with Professor Wilson and our TA, I was able to switch to Google Colab. On Google Colab, I was able to get two models working, so I could perform the zero-shot classification. However, due to limitations on Google Colab, I could only perform zero-shot classification on 1000 comments in the test dataset as opposed to the whole datatset. Finally, my last challenge was understanding how to do section 2.4 of the assignment. I originally was using my naïve bayes code from a previous homework assignment, but I switched to use the multinomial naïve bayes function from the sklearn package. Also, I did not understand how to calculate my baselines. Professor Wilson explained that my baselines are all 50% because I am using the balanced train split of my dataset.