CS5984 - HW1

Abhinav Sethi (PID: abhinavsethi)

October 3, 2021

1 Introduction

This document provides a brief overview of the approaches used in the Sentiment Analysis of IMDB dataset task as part of homework 1 for CS 5984 - Natural Language Processing.

2 Data

The problem statement requires building a model for classification of IMDB reviews as either Negative or Positive. The dataset consists of 50K labeled records of movie reviews from the IMDB website. This dataset is further divided into Training (40K), Validation (5K), and Testing (5K) datasets.

The dataset is fairly balanced as shown in table 1. Hence, there is no need to apply any resampling technique and the data can be used as is. Next, we need to find a good balance in the max length as with a large value we can run into memory issues and with low value a lot of information will be truncated. To estimate the max length for the tokenizer we plotted the token counts from the training data as given in figure 1. From the figure it is evident that the peak density is 200 words length so we have picked 300 as the max length as it seems to be a decent estimate.

3 Methodology

For the classification task we have used the state-of-the-art model BERT (Bidirectional Encoder Representations from Transformers). Its key innovation is

Dataset Type	Positives	Negatives	Total
Train	19089	20019	40000
Validation	2514	2486	5000
Test	2505	2495	5000

Table 1: IMDB dataset

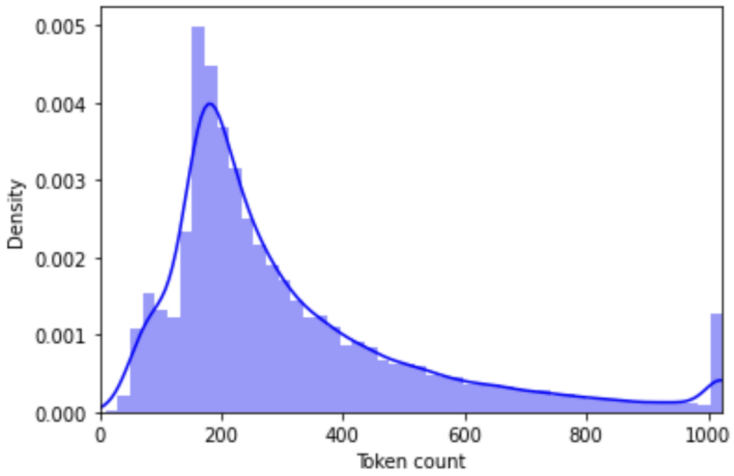

Figure 1: Encoded tokens count vs density plot

that it applies bidirectional training of Transformers, to language modelling [1]. Which is different from the previous models which looked at a text sequence either from left to right or both left-to-right and right-to-left training. The model showed that a bidirectionally trained model can have a deeper sense of language context and flow than a directional language model. This bidirectional property is particularly useful for application in sentiment analysis tasks as it requires a good understanding of the importance of words in relation to other words in the text.

Specifically, we have used 'bert-base-cased' pre-trained model that was trained using a stack of 12 transformer encoders. Also, the model was trained by taking the casing of words into account. Intuitively, it makes sense to use a cased model for sentiment analysis, example: reviews can have all caps words like 'BAD' which can convey more sentiment than simply 'bad'.

4 Pre-processing

The training for BERT model requires tokenization of the input text such that it has special tokens for unknown words, sentence separation, and attention masks. Also, the input tokens need to be of fixed length (we have set 300) thus requires padding or truncation. All this processing is done using the encode_plus method from the BERTTokenizer class. For batching of the dataset we have used the torch Dataset and DataLoader class by passing in the data along with a batch size. For this model, the batch size is set as 16. Any further increase can lead to memory issues.

Parameter	Current model's value
Batch Size	16
Number of Warm-up steps	500
Drop out probability	0.3
Learning rate	$2e^{-5}$
Max Length	300

Table 2: Model Parameters Used

5 Model

The core model of our classifier is built on the basic pre-trained BertModel 'bert-base-cased' from the hugging face repository. The training approach used in the current model is primarily inspired from the original paper [1].

The BERT model takes in the encoded token ids and the attention mask tokens as the model input. The model does all the heavy lifting and outputs a 'summary' of the content in the pooled output parameter with a size of 768, which is the default size of the hidden layer in the Bert model configuration.

This output is then flowed through a drop out layer for regularization that essentially zeros out the input values with a probability of p. Here p is a hyper-parameter its value was selected as 0.3, however it can be further tuned.

After the drop out layer, the output is then pushed through a fully connected linear layer. This layer essentially applies a linear transformation to the incoming data such that $y = xA^T + b$, essentially reducing the output size from 768 down to 2. The final output is of size 2 where the higher value represents the output label as being positive or negative sentiment.

6 Optimization

For optimization, Adam optimizer with weight decaying (AdamW) is used. It has been shown experimentally that AdamW yields better training loss and that the models generalize much better than models trained with Adam [2]. The learning rate is a hyper-parameter and was set to be $2e^{-5}$, however further tuning may be required to improve performance as suggested by the authors (among $5e^{-5}$, $4e^{-5}$, $3e^{-5}$, and $2e^{-5}$) in [1].

We have also incorporated a linear scheduler to further manipulate the learning rates during the batch training process. The learning rate selected above is the initial rate for a linear scheduler with warm-up that essentially first increases the learning rate and then starts decreasing linearly. The number of warm up steps was set to 500 as it equals to about 20% of the training steps (as with 16 batch size we have total 2500 steps per epoch). Here, the number of warm up steps can also be further tuned. Also, for the loss function we have used CrossEntropyLoss. Table 2 shows a list of all the hyperparameter values.

As an outline of the training process, for each batch, first all gradients are

Dataset Type	Total Samples	F1 Score
Train	40000	96%
Validation	5000	92%
Test	5000	92%

Table 3: Results after final epoch

set to zero. Then the batch data is fed to the model then the current gradients are stored using the loss.backward() method call. Following this the gradients are clipped to the max value of 1 to prevent exploding gradients using clip_norm method from the pytorch utils. After the optimizer and the scheduler steps are iterated over before the next batch is trained.

7 Results

We ran the model for 3 epochs separately (google collab requires a consistent connection, otherwise it automatically disconnects) as each epoch requires a training time of 65 minutes (approximately).

The performance statistics were produced using sklearn library's classification_report method. The F1 score on the test set was 89%, 91%, 92% for the three epochs respectively. The overall summary of the results after the final epoch is shown in table 3. As the F1 score has only dropped 4 points when compared with the training accuracy (96%), it shows that the model has generalized quite well.

The total training time for the current model is $3 * 65 = 195$ minutes. Which has essentially hindered the proper tuning of the model. The F1 score can further be improved by running the model for more number of epochs. Apart from this, hyper-parameters such as number of warm up steps, max length, drop out probability, and learning rate can further be tuned to improve testing accuracy.

Although we only used a generic BERT base model for our task, yet it has proved to be highly effective and has performed quite well in this Sentiment Analysis task leading to good generalization for prediction on new texts.

References

[1] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2019), NAACL, Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina Toutanova [arXiv:1810.04805v2]

[2] Decoupled Weight Decay Regularization (2019), ICLR, Ilya Loshchilov, Frank Hutter [arXiv:1711.05101v3]