Movie Reviews Sentiment Analysis

Fine-tuing a Pre-trained Model on SST2 Task

Zahraa Noureddine
Department of Physics
University of Michigan
Ann Arbor, Michigan
nzahraa@umich.edu

***Abstract*—In 2019, a new language representation model called BERT, Bidirectional Encoder Representations from Transformers, introduced bidirectional pre-training using masked language modeling by masking random tokens which enabled deep bidirectional representations. BERT improved GLUE score to 80.5%. [1] along with advances in other areas of natural language processing. Shortly after, a distilled version of BERT was released. This version had only 40% of the parameters of BERT but maintained 97% of its language understanding capabilities. [2] This made distilBERT more favorable especially for users having constraints on memory and resources. These models can be fine-tuned to achieve specific natural language processing tasks such as GLUE. In this project, I fine-tune the pre-trained model, distilBERT on sentiment analysis dataset 'sst2'. The accuracy was increased from the poor performance of the general-purpose distilBERT to 89%.**

I. Introduction

Enjoying a good movie is a recharging experience that a lot of us would like to have despite our busy schedules. This makes it important that we get movie recommendations tailored to our unique preferences and state at the time of watching so we get the best out of our invested time. My project is aimed at matching users' input to a list of the top recommended movies for them to watch. To do that, I will use a pre-trained machine learning model that can perform sentiment analysis on text and fine tune it to infer movie quality from peoples' reviews. In the first step of the project, I am fine-tuning the model to get the general sentiment of the movie being either positive/negative. This model can then be used within a movie recommendation app.

II. Methods

A. Model

The **model** used for this project is distilBERT, which is pre-trained on English Wikipedia and BookCorpus (a dataset of over 11,000 books of various genre). Its training enables it to understand general language but it is not trained on a down-stream task. An overview of the moodel (input, output) is shown in Figure1.

Identify applicable funding agency here. If none, delete this.

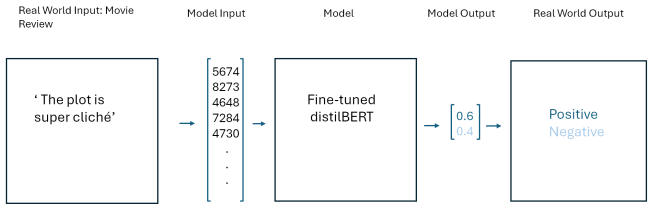

Fig. 1. After supervised fine-tuning, the fine-tuned distilBERT model will be able to distinguish positive and negative sentiments of movie reviews

B. Data

The **dataset** used to train the model is: 'glue'/'sst2', which is a dataset composed of movie reviews along with labels: [positive, negative]. The dataset has 67k data samples.

Quality of the dataset: Upon investigation of the dataset, it was noticed that some reviews were chopped into phrases, some of which ceased to have a strong sentiment (for the human reader). Such data would introduce noise to the learning process. The amount of data flagged as having weak sentimnet in our analysis was about 66%.

III. Results

A. Preprocessing of Data

Generally, for a Natural Language Processing task, data has to be tokenized. Tokenization is a process in which text data is turned into tokens that can be words, subwords, etc and these token would then have a corresponding numerical value that the model has learnt during pre-training.
So, the first step of data preprocessing was tokenization, includding processes like padding and truncation to make sure the data structures were uniform and no runtime errors were raised.

In addition, the data set was 'cleaned' by filtering out the phrases which had weak sentiment. This was done using the TextBlob library which offers tools for a rule-based sentiment analysis and uses a pretrained lexicon. This would scores the sentiment of a text sample with scores ranging from -1 being

the most negative to 1 being the most positive. Phrases with absolute value of sentiment less than 0.3 were filtered out. After filtration, the dataset size was reduced to 20.5k data samples.

B. Training and Regularization

Overfitting was a main problem faced during training. This was indicated by a decrease in training loss, increase in validation loss, and an increase in accuracy. Filtering the data was one method of mitigating this. However, this decreased the dataset size which counteracted its positive effect rendering this method less useful.

Additionally, regularization techniques were applied, like dropping out random neurons from the network and introducing weight decay which discourages the existence of large weights in the network. Both methods serve to prevent the model from heavily relying on certain weights and forces is to train based on more of its weights which prevents memorizing noise from the dataset as pattern and pushes it to recognize the actual patterns.

C. Evaluation

The different effects of these changes to the preprocessing and training process along with their effects on accuracy, training loss, and validation loss are presented in table 2.

The accuracy achieved after the first epoch of training for the model with dropout mechanism within its configuration was 83.4%. Doubling the parameters for dropout (), had negligible effect on the accuracy.

Implementing weight decay helped partially helped reduce overfitting by decreasing the validation loss in the third epoch compared to the training without weight decay. Additionally, the accuracy increased from 83% to 84% in the first training epoch as shown in 2. Despite the negligible effect on mitigating overfitting, this tool helped increase the accuracy. Early stopping can be used in this case after the first epoch.

For the results presented so far, the data was sharded into 7 shards for faster training. It's worth noting that training on the whole dataset on google colab was taking multiple hours when done on CPU. sharding the dataset and using GPUs helped reduce the computation time to 10 min/epoch for the used hyperparameters.

Next, I trained the model on one third the dataset (nb of data shards was modified from being 7 to 3). The accuracy was raised to 88% within the first epoch. Filtering the dataset by removing the examples with neutral sentiment, which reduced the size od the dataset to one third again, raised the accuracy to 89%. The effect that was awaited from this extra data preprocessing step was possibly masked by the fact that filtering improved the data quality while decreasing the dataset size at the same time.

Epoch	Training Loss	Validation Loss	Accuracy
1	0.349300	0.353674	0.832000
2	0.199100	0.424910	0.848000
3	0.130100	0.491055	0.864000
4	0.080000	0.564966	0.864000

Epoch	Training Loss	Validation Loss	Accuracy
1	0.340500	0.337385	0.840000
2	0.197500	0.467447	0.856000
3	0.136400	0.439025	0.880000
4	0.080100	0.487370	0.888000

Fig. 2. Top: Before adding weight decay. Bottom: After adding weight decay, and raising the value gradually from 0.01 to 0.5

D. Hyperparameter Search

A hyperparameter search was conducted over 10 different trials scanning different batch sizes and learning rates. The maximum reached accuracy did not exceed the highest accuracy achieved while doing manual fine-tuning which was 89%.

IV. Conclusion

Training distilBERT on the 'sst2' data was able to raise its accuracy in identofying the sentiment analysis of text to 89%. In this exploration, the effects of changing multiple parameters was studied. Applying tools that are known to mitigate overfitting did not always yield the expected improvement in accuracy whcich shows that the success of these tools is not straightforward and is based on the nature of the model, the layers that it's applied to, the dataset size and quality, and the task being solved.

Acknowledgment

I would like to thank Professor Xian Zhang for her dedication to teaching this course. Her concise lecture slides, simplified explanations, and enthusiam have made the course fun, provided me with a solid base in the the most common data science tools, and intrigued me to dive deeper into machine learning.

Github repository containing the code for this project and a copy of the report can be found at: https://github.com/ZahraaNoureddine/Stat507$_final_project$

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In North American Chapter of the Association for Com- putational Linguistics, 2019.

[2] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. ArXiv, abs/1910.01108, 2019.