6893 Project Progess Report

Tiancheng Shi
UNI:
ts3474

Anne Wei
UNI:
yw3939

Abstract

Bidirectional Encoder Representations from Transformers (BERT) has become the state-of-the-art baseline for Natural Language Processing (NLP). Despite the sheer advantages of $BERT_{LARGE}$ in model performance, $BERT_{BASE}$ or even smaller BERT models are widely applied in the industry, likely due to their less strict hardware requirements, especially GPU Memory.

In this project, the team aims at using distributed training strategy on local devices to train a $BERT_{LARGE}$ model for sentiment analysis tasks. After comparing with various other classification models, we propose to deploy the best model into PySpark to predict the sentiment implied in Twitter real-time streaming data.

1. Introduction

Twitter is a popular online social media platform for users to share their immediate feelings and thoughts within a comparably shorter length of paragraphs. With the explosion of available information in the Big Data era, it is a general trend that understanding the opinion – or specifically, the emotions of the public – is gradually gaining its significance. The project team is primarily motivated to use modern Natural Language Processing (NLP) techniques to gain insights of English Twitter users' response regarding the ongoing 2022 FIFA World Cup.

In this work, our team first employs traditional NLP methods (e.g., word frequency, Python NLTK Vader Sentiment Analyzer, etc.) on a static, labeled dataset of Twitter users' message posts (i.e., tweets) and their corresponding sentiment of either positive (1) or negative (0). We then train multiple classification models on the static data to predict the sentiment of tweets, among which the pre-trained BERT model achieves the best performance on validation set, after using Multi-Worker Mirrored Strategy training. We finally broadcast the weights of the saved, suitable BERT model onto PySpark, in order to predict the real-time text data streamed through Tweepy.

To deploy and evaluate our model, we choose to focus on the analysis on the trending topic 2022 FIFA World Cup. Twitter API is used to extract both streaming and history tweets, and hashtag (e.g. #Argentina, #Neymar) and keywords filters are applied to see people's sentiments on a specific country or player. The results should not only provide insights on people's reactions to unexpected events but also reveal the performance of the model based on the events happened during the FIFA tournament. We also manually label **xxx** data as positive and **xxx** data as negative to further analyze the model performance, specifically robustness to the negation and

The source code of this project will be available upon the submission of the final report.

2. Related Work

2.1. Related Sentiment Analysis with Streaming Data

Studies similar to ours [5] have investigated real-time sentiment analysis on politics-related tweets mainly using Spark streaming, while also introducing Kafka to arrange the streaming data from twitter API before sending to Spark Stream which could solve the bottleneck of processing the text analysis on a high-velocity information. It would be a good method if we want to pursue an efficient analysis framework. On the other hand, the proposed NLP method to do sentiment analysis is somewhat old-fashioned. Basically, the model classified the tweets into positive or negative by summing up the positiveness of each word in the tweet, and there are positive and negative dictionaries where all the words were stored. This rule-based analysis could be misleading due to the fact that no negation handling and modifier management is adapted to the sentiment classification [1].

2.2. NLP in Deep Learning Era

Witnessing the emergence of modern Artificial Intelligence, Recurrent Neural Network had long been the state-of-the-art method for NLP tasks, because of its memory property when dealing with sequential data. Long Short-Term Memory (Hochreiter et al., 1997) model, by ad-

dressing RNN's gradient vanishing issue in long sequences through forget gates, further improved its capability, especially in the sequence-to-sequence tasks like machine translation.

Yet researchers later discovered that embedding words in a sequential, left-to-right order does not reasonably represent the inherent structure of human languages. In addition, RNN-based models work poorly in parallelization, as the input of later cells is dependent on the previous cells in a layer. Thus, the Self-Attention mechanism is proposed with the Transformer (Vaswani et al., 2017) model. It is worth admitting that Transformers, by every means, marked a cutting-edge milestone for NLP; meanwhile, its transductive, encoder-decoder architecture – originally designed for translation tasks – encountered drawbacks when generalizing to text classification tasks like sentiment analysis.

Based on self-attention in the previous approach, the direct descendant Bidirectional Encoder Representations from Transformers (BERT [2]) forwards both within-sentence word attentions and cross-sentence correlations, in a broader scope compared with traditional Transformers. Even better for our specific task, BERT also introduced a self-supervised learning strategy that makes transfer learning applicable – the pre-trained parameters are label-independent and thus more generalizable.

2.2.1 $\text{BERT}_{\text{BASE}}$

Devlin et al. [2] initially proposed the model with 2 different levels of complexity, among which $\text{BERT}_{\text{BASE}}$ has $L = 12$ layers, hidden size of $H = 768$, and $A = 12$ self-attention heads (approximately 110 million parameters in total). These structural hyperparameters were chosen to mimic the model size of, and thus compare with Generative Pre-trained Transformers (OpenAI GPT; Radford et al., 2018). The $\text{BERT}_{\text{BASE}}$ model, as well as other smaller variants like $\text{BERT}_{\text{MEDIUM}}$ or $\text{BERT}_{\text{SMALL}}$, has been widely adopted in various text classification tasks, especially sentiment analysis ones [4] (Xu et al., 2019).

2.2.2 $\text{BERT}_{\text{LARGE}}$

The primary purpose of $\text{BERT}_{\text{LARGE}}$ is to challenge the highest potential performance of this model. The authors set hyperparameters of $L = 24$ layers, $H = 1024$ hidden size, and $A = 12$ self-attention heads, resulting in 3 times more parameters, compared with $\text{BERT}_{\text{BASE}}$.

Such increase in performance comes with cost – in practice, training a $\text{BERT}_{\text{LARGE}}$ model typically requires at least 16 GB of GPU Memory, which is oftentimes beyond the capacity of a single GPU worker. In the industry, researchers (Xu et al., 2019, etc.) expecting higher performance of $\text{BERT}_{\text{LARGE}}$ models may compromise to use more concise ones, due to the limitation of computational power.

3. Progress

3.1. Data

3.1.1 Static Data

The static dataset we use is *Stanford Twitter Sentiment (STS)* which includes 1.6 million tweets and their corresponding sentiments. Other trivial information like User ID, Tweet ID, and post date are also included in the dataset. Initial explorations show that each tweet has 74.09 alphanumeric characters, from which 16.41 words are formed, on average. Further, instead of manually labeling data, the sentiments are generated in a seemingly silly but efficient and highly accurate way – researchers streamed through all tweets in a certain period of time, filtered out all tweets with special characters ":)" and ":(", and labeled them as positive and negative, respectively.

One of the big challenges of the dataset is that BERT requires a large scale data due to its gigantic model size. This is the main reason we choose to use STS instead of fine-grained dataset and emotion detection dataset as both of them needs to be manually labeled to ensure their accuracy. If the label is automatically generated by models, it is still not feasible to fit BERT on the model-based dataset. Thus, the safest way is to use binary classification data which could acheive both high accuracy and efficiency. And since it is hard to extract the streaming data from twitter API within limited time, using STS is the best choice for our project, which has also been broadly used in the research such as twitter sentiment analysis and subjectivity analysis [3].

3.2. Data Preprocessing and Exploration

Due to the sparse nature of human natural languages, data preprocessing procedures can be challenging. The project team first drops trivial columns, and excludes all information irrelevant to sentiments through Regular Expression. For example, mentioning other users in a tweet, in the format of "@SomeUserID", as well as URLs starting with "http://" and "https://", are all removed, with the belief that the characters in User IDs and URLs will not influence the sentiment when composing the tweet. In the raw dataset, sentiments are assigned labels of 0 (negative) and 4 (positive), which are transformed to 0 and 1 in the cleaned dataset.

Next, we apply word tokenizer in Python NLTK to split each text into words, bigrams, and trigrams, and count their frequency of occurrences in both positive and negative tweets separately. The odds of a certain element (a single word, 2-word, or 3-word phrase) belonging to a positive tweet against that belonging to a negative one are visualized in the scatterplot below. The diagonal represents the line on which elements have equal probability of belonging to pos-

itive and negative, and the more an element deviates from the diagonal, the more extreme in sentiment it can be interpreted. Result shows that words generally have the most stark positive-negative contrast.

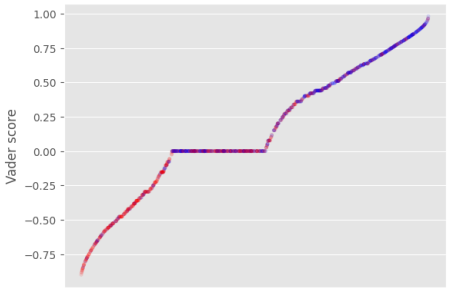

Figure 1. Vadar Score

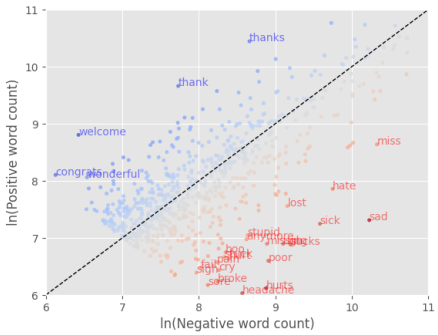

Figure 2. Word Frequency with positive and negative analysis

As a reference, the Vader Sentiment Intensity Analyzer is also applied to the texts in the dataset to visualize the distribution of positive and negative emotions. The model generates a continuous, context-independent Vader Score between -1 (negative) and 1 (positive). The dataset is then sorted in the ascending order of Vader Score, with each red and blue point representing tweets with negative and positive labels. It is worth emphasizing that the Vader model tends to make 0 predictions when the text seems to be neutral, while the labels in the dataset are dichotomous.

3.3. Methods

3.3.1 LSTM

Though not implemented yet, we plan to train an LSTM as a baseline model. Considering the relatively long input sequences of strings, it is reasonable that the long-term memory feature enabled by Forget Gates are less capable than the attention mechanisms. Thus, we do not expect perfect performance of such model.

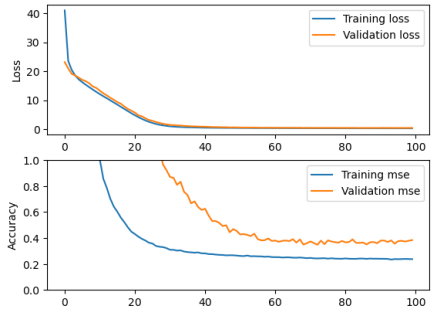

Figure 3. LSTM Accuracy and Loss Comparison (To be completed)

3.3.2 $\text{BERT}_{\text{BASE}}$

Upon studying multiple Deep Learning models in NLP, the project team decides to apply a pre-trained, uncased $\text{BERT}_{\text{BASE}}$ model obtained through TensorFlow Hub. To perform transfer learning, another dense layer is added on top of BERT, followed by an output layer of 1 neuron with Sigmoid activation function.

During our experiment on the team's local devices, a 5-epoch training session of $\text{BERT}_{\text{BASE}}$ takes less than 17 hours on a single GeForce RTX 3080 GPU with 10 GB memory. With proper hyperparameter tuning, this model achieves a Training Accuracy of 94.16% and Validation Accuracy of 85.63%. (We are currently improving the model performance. The temporary best set of hyperparameters are: batch size = 32, learning rate $= 2 \times 10^{-5}$ for Adam optimizer. Regardless, we tend to conclude that the result is promising, given the possibility of automatically generated labelled data error from being ground truth.)

In order to further accelerate the training and hyperparameter tuning process, we may employ Mirrored distributed training strategy on another GeForce RTX 2070 Super with 8 GB memory in the future. During the training session, a copy of the full model parameters is kept in both workers, and they mirrored each other when performing gradient descent.

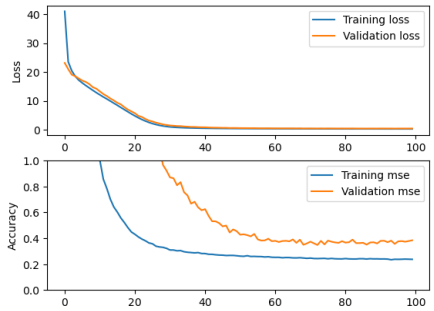

Figure 4. BERT-base Accuracy and Loss Comparison (To be completed)

3.3.3 BERT$_{\text{LARGE}}$

The project team further proposes to experiment with more complex models, specifically the BERT$_{\text{LARGE}}$ model with 340 million parameters. Considering that the model size (16GB) already exceeds the GPU memory of every available hardware (10 GB or 8 GB) alone, we plan to employ a Multi-Worker Mirrored Strategy for distributed training – i.e., only load part of the full model on each local worker, and communicate the model parameters in every step of the training session. We initiate a local server to connect our machines, and we expect to utilize Nvidia NCCL implementations (through TensorFlow Distributed Strategy methods) to achieve Collective Communication between the GPUs.

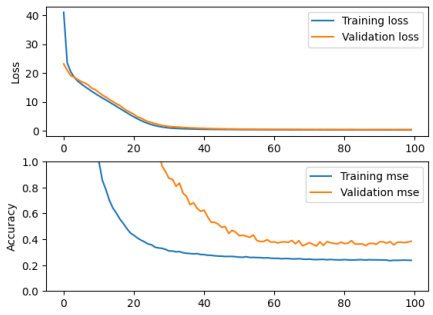

Figure 5. BERT-large Accuracy and Loss Comparison (To be completed)

3.4. Systems

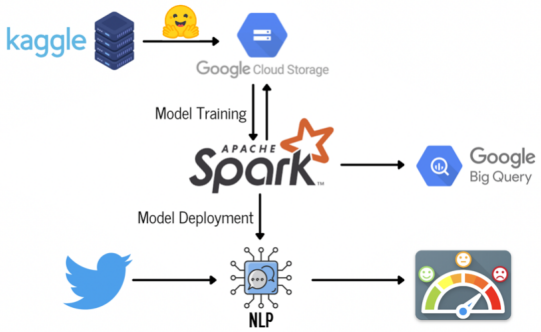

Figure 6. Systems Overview

The project's body is mainly constructed by *Stanford Twitter Sentiment (STS)*, Apache Spark, and Twitter API. The model is first trained on STS. Once a satisfying model for NLP tasks is trained on local machine, we then applied the model on the streaming data extracted from Twitter API. After sentiment analysis is finished, we will load the data into big query and do further visualization.

4. Experiments

4.1. Model Evaluation Metrics

Besides overall accuracy, we would also like to know other relevant evaluation metrics, including F1-Score and Area Under Receiver Operating Characteristic (AUROC). Further, the Precision-Recall curve can be plotted using the training history data obtained above.

4.2. Performance Experiments

Besides evaluating on validation set, the robustness of the selected model can be further tested in a self-supervised manner, by intentionally manipulating the text data and comparing its new prediction with the original one.

One common experiment for robustness is the negation test, during which a strong negation word is manually inserted into each sentence in the test set. It is expected that the model should generate an opposite outcome. For example, since "The food is good" is considered as positive, the model should, if well mimicking human intuition, assign a negative label to the opposite sentence "The food is not good."

On the other hand, it is also considered a desired behavior for a model if, without explicitly adding strong positive or negative words or phrases, the prediction of sentiment should be robust to the minor changes in irrelevant information in the text. For example, "Issac Newton likes the mild weather in London" should reasonably has the same label with "Albert Einstein likes the mild weather in New Jersey", under the common consensus that people's names and locations are oftentimes irrelevant to the sentiment of a sentence.

4.3. Deployment on PySpark

The selected model will not be limited to run locally on static data. A TensorFlow model of the same structure will be created on Google Cloud Platform. The trained model's corresponding weights and parameters stored in local directory will be pushed to the cloud through SparkFiles, and broadcast to the new-initiated model on Cloud. Upon successfully building the same model in PySpark, it will be tested on streaming data obtained from Twitter Tweepy API. The testing procedure needs to be implemented in Pandas UDF function before applying transformations on Spark RDD. We will also manually label, say, 100 tweets to evaluate its behavior.

4.4. Future Work

In the future, we are going to adapt our model to the Twitter streaming data to test the performance of the models. The topic we are interested in is the ongoing FIFA World Cup. The first analysis task is to compare the sentiments among top 10 popular players in the FIFA

World Cup. The list contains Messi, Neymar, Mbappe, Cristano Ronaldo, Moderic, Kevin De Bruyne, Antoine Griezmann, Harry Kane, Robert Lewandowski, Son Heungmin. We want to extract the tweets about those players using the hashtags or keyword search, compare the total tweets amount and the percentage of positive and negative sentiments among them. We are potentially going to use a bar chart with R to do the statistics analysis.

The second task is a sentiment analysis task on how sentiments are affected by the unexpected events. We are going to fetch tweets about a specific country for three to four days. Those days contain some important games which will affect the team's qualification or elimination. A line chart will record the trend of the sentiments as the tournament goes on. Finally, in order to check the accuracy and robustness performance of the model, we are going to manually label 100 tweets from the streaming data. We might change some key words or add negation words to the sentence according the the robustness experiments rules mentioned in Secion 4.2.

References

[1] Muhammad Zubair Asghar, Aurangzeb Khan, Shakeel Ahmad, Maria Qasim, and Imran Ali Khan. Lexicon-enhanced sentiment analysis framework using rule-based classification scheme. *PLOS ONE*, 12(2):1–22, 02 2017. 1

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. 2

[3] Anastasia Giachanou and Fabio Crestani. Like it or not: A survey of twitter sentiment analysis methods. *ACM Comput. Surv.*, 49(2), jun 2016. 2

[4] Manish Munikar, Sushil Shakya, and Aakash Shrestha. Fine-grained sentiment classification using bert. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–5, 2019. 2

[5] Nashwan Zaki, Nada Hashim, Yasmin Mohialden, Mostafa Mohammed, Tole Sutikno, and Ahmed Ali. A real-time big data sentiment analysis for iraqi tweets using spark streaming. *Bulletin of Electrical Engineering and Informatics*, 9(4):1411–1419, 2020. 1