Sentiment Analysis of COVID-19 Tweets Using BERT-RNN and CNN

EECS498-004 Intro to NLP Final Project Report

Weiji Li
University of Michigan
weijili@umich.edu

Xiaopan Zhang
University of Michigan
xiaopanz@umich.edu

Yutong Xu
University of Michigan
ytxu@umich.edu

1 Problem Description

COVID-19 has impacted all of us in some way. We wanted to apply what we have learned in this course to analyze how the public has progressed in its sentiment towards the pandemic. In order to get the public's opinion on this topic, we decided to build models to perform sentiment analysis on one of the most popular microblogging platforms, Twitter. Each of the models we explored receives raw text of Tweets as input and outputs the label (either positive or negative). After determining the model with the best performance, we used it to label Tweets from February to December, 2020 and produced an overall sentiment trend throughout this past year. From a broader view, once determined, the best suitable model could also be used to produce sentiment trends and offer insights into important and/or popular topics.

Throughout the project, we encountered some limitations and challenges. First and foremost, it was hard to find pre-labeled Tweets for training, we eventually used the Sentiment140 dataset, but it still has the limitation of data being labeled using the simple rule of whether there is a smiley face or sad face present. For the same reason of limited data resources, we had to resort to binary labeling (labeling each input as either positive or negative), which might not be the most realistic labeling scheme, but the only we could work with. Finally, capturing Twitter sentiment is just a hard task—Tweets often include sarcasm, spelling errors, and new vocabulary with non-traditional meanings—and therefore most models done by others using Sentiment140 have accuracy under 80%.

Despite these challenges, we compared between quite a few models with preprocessed data and used different types of embeddings to find the ones that best capture Twitter sentiment, CNN with 0.74 accuracy on human-labeled test dataset and RNN+BERT with 0.74 accuracy on validation set. We used both models to produce a sentiment trend on COVID-19 related Tweets and observed similar fluctuations between the two.

2 Related Work

The first reference we made before starting the project was the research that our training dataset came from. In their paper, (Go et al., 2009) used the Naive Bayesian approach along with Max Entropy as well as Support Vector Machine to label Tweets as either positive or negative, which inspired us to use Naive Bayes as our baseline model. Moreover, we wanted to explore different types of neural models and therefore looked to our second reference. In this paper, the research team discussed several types of neural networks that are suitable for sentiment analysis, including Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) (Das and Chakraborty, 2018). We adapted the approach in this paper by implementing both CNN and RNN, which are also different from their work because we tried using BERT embeddings in addition to the GloVe embeddings mentioned in the paper. We chose to experiment with BERT because it is currently the most popular embedding used in NLP and were interested to see whether it could give us better results. Finally, we also combined BERT with a simple logistic regression model in order to make further comparisons between supervised and unsupervised models for the sentiment analyzing task.

3 Methodology

3.1 Dataset

For training and validation purposes, we used Sentiment140, which has 1.6M real Tweets, half labeled as positive and half labeled as negative (Go et al.,

2009). This dataset is suitable for our task because it has been pre-labeled and has a big enough size to fully train our models. For testing and the final trend-producing task, we acquired access to the Twitter Decahouse provided by University of Michigan and pulled around 2000 Tweets for each month from February to December of 2020. Among the total 14,000+ Tweets, we hand-labeled around 700 Tweets with obvious positive or negative sentiments and used them for testing, and we used all 14,000+ Tweets to produce the trend at the end. The hand-labeled Tweets are different from our training and validation dataset in that they are much more recent and are actually related to the topic of our interest, COVID-19. Sentiment140 was acquired from a research done in 2009, which obviously did not contain any mention of COVID-19. Therefore, we decided to hand label Tweets in order to obtain results that are more closely related to our intended goal.

For each of our models, we did some preprocessing on the data. Tweets are characterized by having username-tagging, hashtagging, and linking features. We took care of these by turning all instances of "@..." into "__USERNAME" and all instances of "http://..." into "URL" and removing the hashtag sign from all instances of "#...". For further feature reduction, we also processed words that have repetitive letters by removing the repetitive letters.

3.2 Models

3.2.1 Naive Bayes

We have applied the Naive Bayes model as the baseline. Naive Bayes is a simple but powerful machine learning model for the text classification task; sentiment analysis is basically binary text classification. The reason we chose this method to begin with is because it's easy to build, perform, and evaluate. Furthermore, we have found much related work on using the Bayesian approach as the baseline and wanted to replicate their results before moving onto more complex models.

We used the NLTK library to remove stopwords as well as perform other steps of data preprocessing as mentioned above. In terms of model implementation, we utilized the scikit-learn library and applied both Gaussian Naive Bayes and the Multinomial Naive Bayes models to the data.

We start with the Gaussian Naive Bayes model. The likelihood of the features is assumed to be Gaussian:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\omega_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\mu_y^2})$$

After finding that the Gaussian Naive Bayers model does not behave so well, we switch to the Multinomial Naive Bayes model. The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, ..., \theta_{yn})$ for each class y, where n is the number of features (the size of the vocabulary) and θ_{yi} is the probability $P(x_i|y)$ of feature i appearing in a sample belonging to class y.

The parameters θ_y are estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class y. We use Laplace smoothing by setting $\alpha = 1$.

3.2.2 CNN

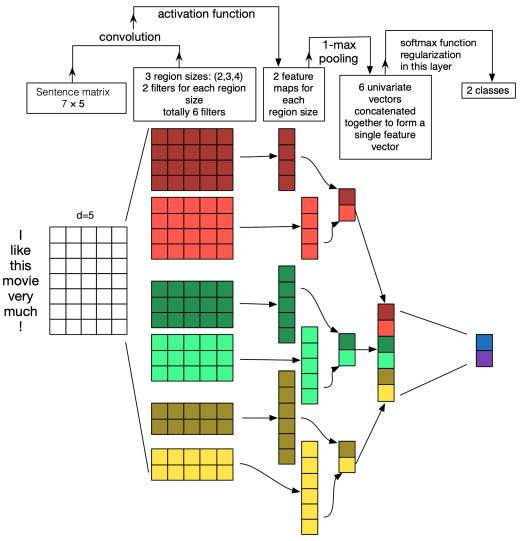

Figure 1: CNN model(Zhang and Wallace, 2016)

We use CNN with GloVe embeddings (Pennington et al., 2014). Specifically, we use GloVe Wikipedia 2014, which contains 6 billion tokens and 400k vocabulary words. We use the 50 dimensional vectors due to limited computing resources and in order to increase efficiency. We implement

the model in (Zhang and Wallace, 2016) with Pytorch and fine-tune it. Figure 1 demonstrates the model structure.

The first layer embeds words into low-dimensional vectors. All input sentences are padded to the same length. The next layer performs convolutions over the embedded word vectors using multiple filter sizes. In our project, it slides over 3, 4 or 5 words at a time. Next, we max-pool the result of the convolutional layer into a long feature vector, add dropout regularization, and classify the result using a softmax layer.

3.2.3 BERT

The third and fourth model both make use of Bidirectional Encoder Representations from Transformers (BERT) (Wolf et al., 2020). We use the encoding part of the BERT, and we use its representation for a whole sentence. Due to the computing resource limitation, we use the "Distilled BERT" version (Sanh et al., 2020) which could save a lot of computing resources while not decreasing the performance too much. For a given sentence, we tokenize it at first. We break words into tokens, add [CLS], [SEP] and [EMPTY] tokens and then substitute tokens with their ids. Then we run the tokenized input through the Distill BERT. Then we would use the [CLS] output from the model, which is also its first dimension. [CLS] represents the features of the whole sentence so we just need this as our feature vector for sentiment analysis. All sentences have 768 dimensions.

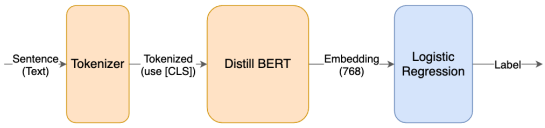

Figure 2: BERT + Logistic Regression model

For the Logistic Regression model, we simply use the linear_model.LogisticRegression model from sklearn [1]. This model doesn't involve any complicated setup on its own. The purpose of this model is just to set up a baseline model for BERT encoding. With it, we could better compare with the BERT + RNN model later.

For RNN, we build a simple RNN model with nn.GRU. In the GRU, there are 2 hidden layers. The hidden output would then be inputed into a linear layer, which would generate output. Then sigmoid function would be applied to generate predicted label.

[1] https://scikit-learn.org/stable/index.html

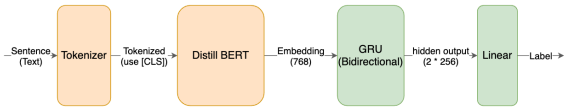

Figure 3: BERT + RNN model

4 Experiments

4.1 Data & Training

For the Naive Bayes model, we randomly choose 100,000 Tweets from the sentiment140 dataset with half positive and half negative as our training set and another 20,000 Tweets as the validation set. Then we use the TF-IDF vectorizer to extract feature vectors and train them by the model.

For the CNN + GloVe model, we utilize the entire Sentiment140 dataset of 1.6M Tweets and split them into train and validation sets randomly using a 90/10 cut. The test set contains 723 COVID-related Tweets labeled by ourselves. For the hyperparameters settings, the detailed parameters are listed here:

- Learning Rate: 1e-4
- Epoch: 40
- Loss: CrossEntroyLoss
- Optim: Adam
- Batch Size: 128

For the BERT + Logistic Regression and BERT + RNN models, we didn't use the full dataset. The BERT encoder takes a long time to run on Google Colab GPUs so we couldn't run our full dataset on it. Instead, we use a portion of it. We split the dataset using a 80/20 test-validation split and used human-labeled Tweets as our test set. For Logistic Regression, no more parameters are needed. The detailed parameters for RNN training are listed here:

- Learning Rate: 1e-5
- Epoch: 100
- Loss: BCEWithLogitsLoss
- Optim: Adam
- Batch Size: 32

	Validation Accuracy	Validation F1-score (Macro)	Human-labeled Accuracy	Human-labeled F1-score (Macro)
Naive Bayes	0.7218	0.7209	0.6421	0.6094
CNN	0.7780	0.7679	0.7449	0.6749
BERT + LR	0.7347	0.7346	0.7018	0.6680
BERT + RNN	0.7447	0.7446	0.6851	0.6600
Transformers Sentiment Analysis (SOTA)	0.6993	0.6964	0.8058	0.7379

Figure 4: Performance

4.2 Performance

Performance comparison is shown in figure 4.

4.3 Analysis

Here is the comparison of performance across 5 different models. The first 4 models are implemented by us and the last model is the state-of-the-art sentiment analysis model from transformers (Wolf et al., 2020).

On the validation set, we could see that our CNN model works the best. It could get around 0.78 accuracy while the best model from other researchers could get around 0.79. We think it performs well because it makes use of all the data and because the model is suitable for the task. In contrast, our BERT+LR and BERT+RNN models only use $\frac{1}{16}$ of whole dataset due to computing resource limitation. Therefore those models didn't perform as well as CNN models. For the transformer model, it is not trained on the Sentiment140 dataset so it doesn't get a good results on it.

On the human-labeled test set, we see that the performance of our model slightly decreases. We also see that our model does not compare with the transformers. We think the reason might be that our own method of labeling is not the same as Sentiment140. Furthermore, Sentiment140 comes from quite a few years ago, therefore it might not work well on recent Tweets, such as on those with new words like "COVID" and "coronavirus".

5 Result

Using our best and second best model (CNN, Bert + RNN), we produce a sentiment trend (Figure 5) on COVID-19 related Tweets from Feb 2020 to Dec 2020. We make use of the 14327 COVID-19 related Tweets we collected across the time periods. And we generate a percentage of positive Tweets for each 10 days.

From the trend, we could see that among most time periods, there is no large peak or trough on the trend. The reason might be that the overall percentage of positive Tweets is consistent across all the time, while some breaking news would cause some variations and fluctuations on the trend.

From the trend, we also could see that when COVID-19 first spreads in the US, there is a huge drop on the percentage of positive Tweets. Also, during summer 2020, the cases reported each day largely increase, generally decreasing the percentage of positive Tweets. Lastly, during October and November, where the election happened, we could see large fluctuations which represent that people are changing attitudes towards the issue quickly.

6 Future Work

We have two major components for improvements. The first is to create larger and more updated datasets using Twitter API and human evaluation. Currently, we don't have any COVID-related Tweets in our training set, which potentially causes our models to underperform on the test set rather than the validation set. In the future, we can

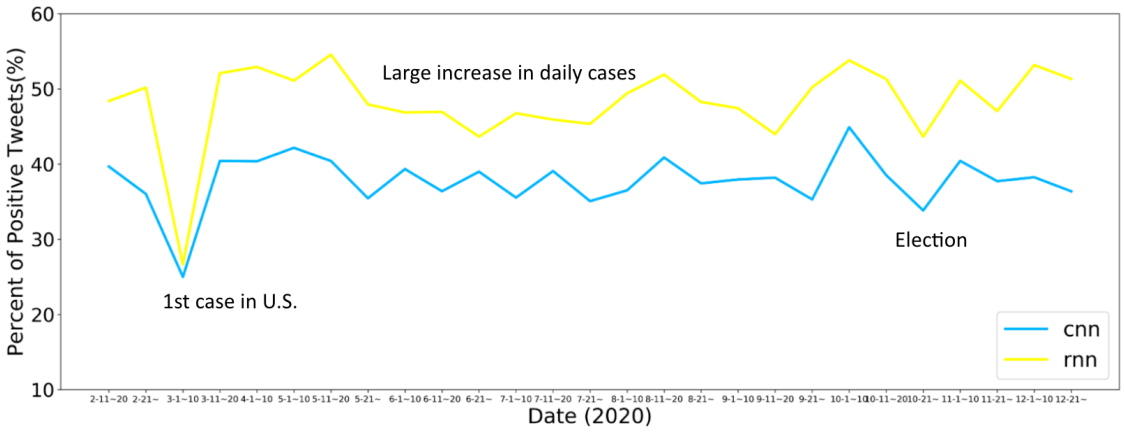

Figure 5: Predicted Trend

increase the size of the COVID-related Twitter dataset and train our model on this updated dataset.

Another improvement is about data preprocessing and feature extraction. We can include additional information (e.g. emojis, hashtags) in feature vectors. Also, it is possible to use GloVe Embeddings with a higher number of dimensions and a larger vocabulary (Twitter), which require higher memory and GPUs with higher computing power.

7 Conclusion

In the present work, we have described a series of experiments with CNN built on top of GloVe embeddings and RNN built on BERT encoding. With a little tuning of hyperparameters, a simple CNN with one layer of convolution and two linear layers performs remarkably well on the validation set. For the test set of COVID-19 related Tweets, a lot of room remains for improvement and future work.

References

Bijoyan Das and Sarit Chakraborty. 2018. An improved text sentiment classification model using tf-idf and next word negation.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *Processing*, 150.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ye Zhang and Byron Wallace. 2016. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification.