A Hybrid Model for Rumor Detection

Student Number: 1058170

Abstract

*In recent years, rumor detection has been a popular natural language inference problem. In this report, we presented an ensemble model that combines random forest, SVM and BERT by majority voting to obtain a more accurate model, each individual model is tuned so that it performs the highest f1-score on validation set. The result shows our model achieves f1-score of **0.847** on validation set and **0.843** on test set. Lastly, we also explored the differences between a rumor and a non-rumor by comparing the number of replies, the count of retweets and favorites, and the content of hashtags. The pattern of the propagation of a rumor is that most of the replies are contributed to the first **2 hours** after the source rumor was published on Twitter.*

1 Introduction

Rumors have a great impact on our society. In recent years, the development of social media, such as Twitter, Facebook, Wechat, etc, facilitates people's life, but these social media also accelerate the transmission of a rumor, which will be harmful for the world.

In this report, we first briefly introduced some state-of-the-art model to detect rumors, and based on their ideas, we presented three different models: random forest, SVM and BERT model, compared their performance on validation set for different hyperparameter setting, and we also combined these models to build an ensemble model by majority voting. Lastly, we use the model to analyze COVID-19 dataset.

2 Related Works

Rumor detection has been a research area in the past 10 years. Many sophisticated models and creative ideas are proposed by researchers. Back in 2015, the main ideas to detect a rumor by feature engineering. In 2011, Castillo purposed a feature engineering method that extracts profile information, patterns of how a rumor is propagated to predict whether the tweet is a rumor [1]. Two years later, Kwon introduced a model that based on the volume of tweets over time [2].

Many models employ kernel-based method alongside with probability graphical models to capture the pattern of propagation of a rumor. Wu designed a hybrid model that combines Gaussian kernel SVM with graph kernel [3]. Later in 2017, tree kernel approach was proposed by Ma. By comparing the similarity of propagation tree structure, the model could detect a rumor more easily [4].

In recent years, recursive neural network (RvNN) has been proposed to handle various tasks, in particular image segmentation and sentiment classification [5][6]. In 2018, Jing, Wei, Kam-Fai proposed an improved version of RvNN that applies both bottom-up and top-down techniques to construct a tree [7].

	rumors	**non-rumors**
training set	1583	3058
test set	187	393

Table 1: positive and negative examples distribution of training set and validation set

3 Data Analysis

The dataset is derived from Twitter raw dataset, with the JSON format. For each part of dataset, it comprises many events, each of which has an original tweet following multiple reply tweets. Statistically, we have 4651 events on the training set, 580 events on the validation set and 581 events on the test set. For each part of the dataset, we count the number of replies for each event and we found most of events have less than 50 replies. The distribution of the reply quantities for each part of

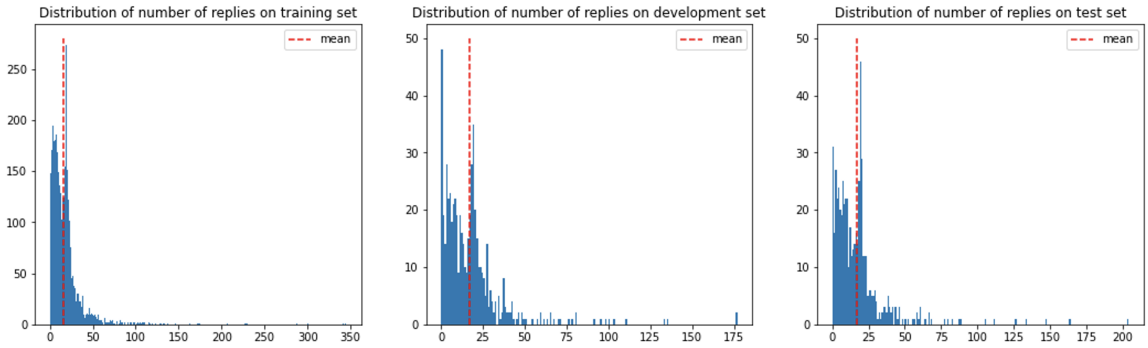

Figure 1. Distribution of replies quantity in training, development and test set

the dataset is shown in Figure 1. For training set and validation set, we also count the number of positive examples and negative examples shown in Table 1.

4 Preprocessing

For each tweet, we extracted the text information from both original tweet and reply tweets. The first removal needed is the URL, since URL contains no information for rumor detection. The second set of words we need to remove are hashtags. Instead of removing the entire hashtag, we kept the word preceding the hash symbol , since we found some meaningful words, such as `#BREAKING`, `#Murder`, etc. After preprocessing the raw text, we tokenized each word for each text in order to build a vocabulary for building a model. The size of vocabulary for training set is 6494.

5 Approaches

In this section, we first presented two different traditional machine learning models (random forest and SVM), shown in **Section 5.1 and 5.2**. Then we finetuned BERT and adapted it for a classification problem (**Section 5.3**). Lastly, we used majority voting to ensemble previous models for better enhancing the overall performance in **Section 5.4**.

5.1 Random Forest

Random forest is an ensemble method that combines multiple weak classifiers, in particular, decision stump, proposed in 2011 [8]. The basic idea of random forest is to build multiple single-level decision stumps and combined them to avoid overfitting.

5.2 SVM

Support Vector Machine (SVM) is a supervised machine learning algorithm originally proposed by Corinna and Vladimr in 1995 [9]. The idea of SVM is to find a hyperplane that will clearly differentiate two classes by maximizing the margin between decision boundary and support vectors. There are two main types of SVM. One is hard-margin SVM which penalizes on-margin or misclassified data points to the infinity scale. Another type of SVM is soft-margin. It tolerates those on-margin or mislabeled instances. The model we used for training is linear SVM without kernel function. The reason for that is we ends up with a very sparse feature vector by tokenization, so it is reasonable to find a linear decision boundary that separates two classes.

5.3 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a self-attention model published by researchers in Google AI in 2018 [10]. The pre-trained model has a wide range of applications, including text classification, name entity recognition, question answering, etc.

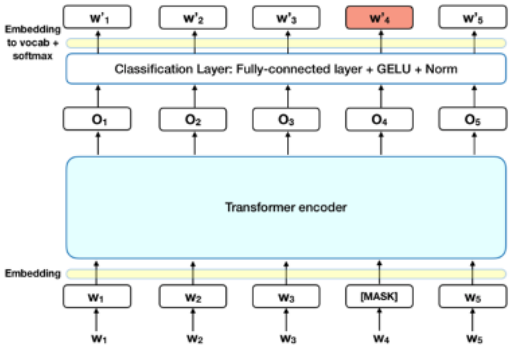

Figure 2. Overall architecture of BERT

The backbone of BERT is based on Transformer,

which is a complicated self-attention model that uses multi-headed attention scheme to enhance model performance. In addition to Transformer, two tasks are used to train a BERT model. One is manually mask 15% of words in a sentence and train a model to predict them. Another is given two sentences, the model will predict whether the second sentence is the next sentence of the first sentence. The illustration of BERT is shown in Figure 2.

6 Experiment and Results

In this section, we will briefly present how to do experiments on different settings on different models. In **Section 6.1**, we will explain how the hyperparameters are tuned and chosen on different models. In **Section 6.2**, we will illustrate the results of each model on its best hyperparameter settings. Many metrices, including accuracy, precision, recall, f1-score and confusion matrix will be used to better reflect the goodness of a model.

6.1 Hyperparameter Tuning

For random forest classifier, we tuned two hyperparameters: criterions and number of estimators. We used two criterions: gini and entropy, and we also use three possible number of estimators, namely 10, 100, 1000.

For SVM classifier, we only tuned one hyperparameter, which is the penalize term C. To find an optimal value, we use hierarchical parameter choice. That is, find a hyperparameter from a setting of $[10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}]$.

After getting the best C in a wide range, the next set of hyperparameters to be tuned are narrowed to small range, which helps us to find a more accurate hyperparameter.

For BERT model, we only tuned three hyperparameters: batch size, maximum length of a sentence and the number of epochs.

6.2 Results

6.2.1 Random Forest

After tuning the hyperparameters, we found the number of estimators has the positive impact on the performance of a model, and Gini index criterion is better than entropy criterion. The model performance on validation set on different hyperparameter setting is shown in Table 2.

	gini	**entropy**
10 estimators	0.7267	0.6899
100 estimators	0.7988	0.7857
1000 estimators	**0.8166**	0.8012

Table 2: F1-score of random forest on different hyperparameter setting

6.2.2 SVM

Since there are two stages of hyperparameter tuning, we demonstrates two tables (Table 3 and Table 4) to show how penalize term C affects the SVM performance. We observed when C = 3, the model achieves the highest F1 score (0.8415) on validation set.

	Val F1-score
$C = 10^{-2}$	0
$C = 10^{-1}$	0.2085
C = 1	0.8114
$C = 10^{1}$	0.8140
$C = 10^{2}$	0.8123

Table 3: F1-score of SVM on different penalize term C (Crude tuning)

	Val F1-score
C = 1	0.8114
C = 2	0.8338
C = 3	**0.8415**
C = 4	0.8302
C = 5	0.8204
C = 6	0.8140
C = 7	0.8162
C = 8	0.8118
C = 9	0.8118
C = 10	0.8140

Table 4: F1-score of SVM on different penalize term C (Fine tuning)

6.2.3 BERT

In BERT, we finetuned the number of epochs, maximum length of a sentence and batch size. In the best setting (6 epochs, max length of 30, batch size of 64), we achieved 0.815 f1 score on the validation set.

6.2.4 Ensemble Learning

After finetuning hyperparameters on all three different models, we combined them into a stronger model by majority voting the labels predicted by

each model. The final results shows the ensemble model achieves 0.847 f1-score on validation set, and if we combined training set and validation set, then ensemble method achieves 0.843 on test set in our best model. The confusion matrix on validation set is shown in Figure 3.

The confusion matrix shows our model achieves 0.898 precision and 0.802 recall for positive instances, which implies our model has a larger generalization ability on negative examples. The reason for that is our dataset is biased, i.e., the number of negative examples is larger than that of positive examples.

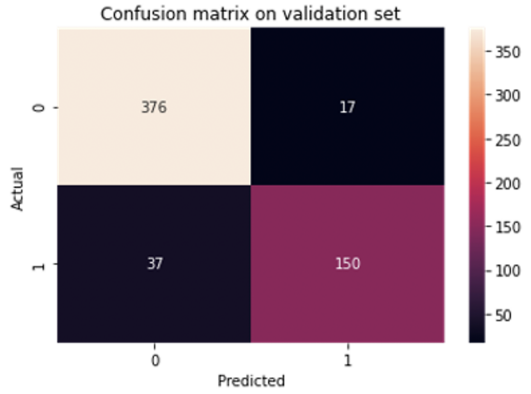

Figure 3. Confusion matrix on validation set

7 COVID-19 Rumors Analysis

Using the model we built previously, we aimed to find the distinguishment between rumors and non-rumors and some patterns that cause a tweet being a rumor.

The first observations are the differences between the number of replies, retweets counts and favorite counts between rumors and non-rumors. Since rumors need to be propagated, it involves people to reply, retweet, and show their favorite to the source tweet. It turns out the number of replies, retweets and favorite counts on rumors should be larger than non-rumors. Our observations support our reasoning. Specifically, all three properties of rumors are at least **50%** larger than non-rumors counterparts. The comparison result is shown in Table 5.

	#replies	**#retweets**	**#favorites**
Rumors	15	3777	14161
Non-rumors	10	2396	7451

Table 5: Comparison of rumors and non-rumors on statistics

The second observation is the topic of rumors. No matter a tweet is a rumor, the hashtags attached to it is the coronavirus related, such as coronavirus, convid19, stayathome etc. However, rumors have some unique keywords. The most eye-catching keyword in a tweet is BREAKING, which is the most common word (except to covid related words) in the predicted rumors set, compared with some other words in non-rumors set.

The last observation is the propagation pattern of a rumor. In our case, we selected a rumor with the largest number of replies, which contains 413 replies. For each reply, we compute the interval between a reply and the source tweet in hours. Then we split different intervals into several slots to observe how a rumor was propagated within a period of time and plot it to a histogram, shown in Figure 4.

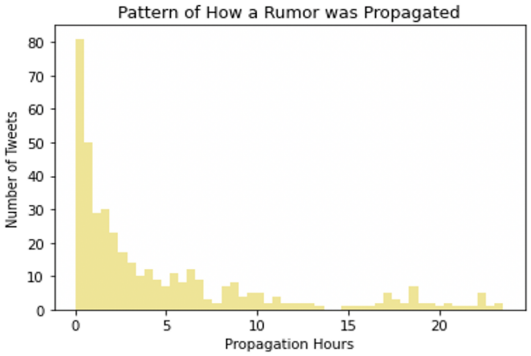

Figure 4. The distribution of the number of replies on a Rumor within a certain period of time after a rumor was published

Our observation is most of replies was sent within 2 hours, which implies a rumor can be spread in a very short period of time.

8 Conclusion

In this report, we briefly introduce the background of how rumor detection has been evolved in the past 10 years, and we designed a model that combines random forest, SVM and BERT, which achieves f1-score of **0.847** on validation set and **0.843** on test set. Lastly, we analyze the feature difference between rumor and non-rumor their unique hashtags, and how a rumor was propagated within a period of time.

References

[1] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In Proceedings of WWW. pages 675–684.

[2] Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In Proceedings of ICDM. pages 1103–1108.

[3] Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In Data Engineering (ICDE), 2015 IEEE 31st International Conference on. IEEE, pages 651–662.

[4] Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). volume 1, pages 708–717.

[5] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In Proceedings of the 28th international conference on machine learning (ICML-11). pages 129–136.

[6] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing. pages 1631–1642.

[7] Ma, J., Gao, W., Wong, K. F. (2018). Rumor detection on twitter with tree-structured recursive neural networks. Association for Computational Linguistics.

[8] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324.

[9] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt and B. Scholkopf, "Support vector machines," in IEEE Intelligent Systems and their Applications, vol. 13, no. 4, pp. 18-28, July-Aug. 1998, doi: 10.1109/5254.708428.

[10] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.