Fake-EmoReact 2021 Challenge

Team Bravo
309551177 / Chia-Wei Hsu
309551179 / Ting-Xin Lin
309551038 / Yu-Lin Lu

1 Introduction

We are now in an informative world. We can easily get a lot of information from any platforms whether the information is real or not. Misinformation may have a bad influence on our life. To mitigate the misinformation problem, we participate in the Fake EmoReact 2021 Challenge(Student) to test whether we can predict the label of the Twitter news correctly.

Firstly, we preprocess the Twitter data. Secondly, we use the tf-idf word embedding technique to obtain some important words and word vectors in documents. Thirdly, we take advantage of Scikit-Learn to predict the label of data. Lastly, we also try the Bert technique to predict but the result is not so satisfying.

Moreover, we not only review our results but also try other machine learning models.

2 Related Work

Originally, it is hard for us to figure out a feasible approach to tell whether the Twitter news is real or not. Fortunately, we find Huang (2018) using the tf-idf word embedding technique and machine learning model reference Granik and Mesyura (2017) Dadgar et al. (2016) to distinguish between fake news and real news. Therefore, based on the similar bi-partitional concept, we use the same approach to distinguish between two different types of news and analyze the results Udomsak (2015).

Later, we do research on some word embedding techniques like the Bag-of-words, the tf-idf, the Word2Vec, and the GloVe Polamuri (2020). And we will discuss the four different techniques in a later section. We also do research on some fake news detection models, but it is hard for us to implement them and there is no result at last.

Additionally, we also find Jwa et al. (2019) using the Bert technique to transform words in documents to word embedding vectors and to solve the predicting problem. Therefore, we also try the Bert technique to predict the label of the Twitter news.

It should be noted that we notice a data imbalance problem in the data provided by the organizer. As a result, we also pay attention to the issue of data imbalance and figure it out by sampling the original data.

3 Method

3.1 BERT

We have tried to use BERT to do the classification task. We will describe how we do the data preprocess and some hyper-parameter of the model.

3.1.1 Preprocess

We will describe how we deal with the data in this section. After take the label and the text from the data frame. We first remove the symbol, tags and links. Then we simply use the encode function in huggingface's Transformer API to encode the texts to the format that can feed in BERT model. We just do some adjust including word to vector, padding the sentence length to 256 and add special token.

3.1.2 Hyper-parameter

There are some hyper-parameter we can choose in BERT. The following is the value we used for fine tune BRET model.

Learning rate: 0.00005
Batch size: 16
Epoch: 5
Loss function: binary cross entropy loss

We follow the normal fine tune setting, except the batch size. We can only afford the batch size to be 16 because the memory is limited.

3.2 ML

Due to the failure of the Bert model which will be described in the later section, we turn our attention

to the classical machine learning model and use TF-IDF to extract the fake news' features. The major steps in this part involve preprocessing data, feature engineering, and fitting classical machine learning models. In the preprocessing step, we delete some useless words by regular expression and turn the words back to their base forms to make the data more meaningful and easier to analyze. In the feature engineering step, we calculate the scores of each Twitter text which represent the possibility of news being fake using TF-IDF and convert the words from strings to numerical vectors using the scores' information to make the vectors helpful toward our target. In the fitting classical machine learning models step, we use scikit-learn to implement naive Bayes classifiers Granik and Mesyura (2017) and SVM model Dadgar et al. (2016) to determine whether each Twitter is fake or not.

3.2.1 Preprocess

Rather than the Bert model which has the neural network that can automatically learn the features by minimizing the cost function, the classical machine learning model requires defining the feature vectors by the researchers first. It's important for us to fully understand our data and find the key information within the data that can help us detect fake news. To make the featuring process smoother, we need to separate the wheat from the chaff of the data before we use it to construct the feature vectors that contain the critical elements manifesting the wrong pieces.

Besides, that unnecessary information usually has some kind of rule or some common sense of pointless words. For example, personal pronoun(I, you, they, it, he, she, me, them, her, him, my, your, etc.), auxiliary verb(can, may, should, do, will, etc.), preposition(about, above, to, from, up ,down, etc.), grammatical tense(the simple past tense concates 'ed' at the end of the verb in most of the cases, the simple future tense insert 'will' in front of the verb, etc.), and so on. Fortunately, there is an existing package doing this thing, nltk, so we simply use nltk package to remove stopwords and lemmatized each noun and verb to restore them to the original state.

Furthermore, the regular expression is another convenient method that can search some particular patterns and is widely use for text processing utilities in Unix. We reuse the concept to clean our data. That is, we capture some rules like URL, tagging, special symbols by regular expression and remove these redundant words from data. After that, we turn all the words into a small case. Throughout this preprocessing step, we get clear data and can move on to the next step.

3.2.2 Feature Engineering

TF-IDF assesses how important a word is to a document. It can scale up the importance of the word frequently appearing in each document, and scale down the importance of the word infrequently appearing in each document while appearing frequently in a single document. The former case is more likely to be a functional word or the commonly used word in all the documents which gives no information for determining whether the word is relevant to a single document. The latter case is more likely to be a keyword to the particular document which usually reflects the core concept of that document. We use the above property of TF-IDF to determine whether a word is more likely for the fake document or the real document. To achieve that, we first separate the tweets with a fake label from the tweets with a real label i.e. combine all the fake tweets to a document, and all the real tweets to another document, because each tweet with the same idx has the same label. Also, the same tweet's (with the same idx) text and replies are grouped as one piece of a document. Then we calculate the TF-IDF score of each word in the collection of the fake document and the real document.

$$\begin{array}{lcc} \text{TF-IDF} & \text{Fake doc.} & \text{Real doc.} \\ \text{word}_1 & \left(\text{score}_{\text{word}_1,\text{fake}} \right. & \left. \text{score}_{\text{word}_1,\text{real}} \right) \\ \text{word}_2 & \text{score}_{\text{word}_2,\text{fake}} & \text{score}_{\text{word}_2,\text{real}} \\ \text{word}_3 & \text{score}_{\text{word}_3,\text{fake}} & \text{score}_{\text{word}_3,\text{real}} \\ \vdots & \vdots & \vdots \end{array}$$

Since the score reflects how relevant the word in either type of document, we subtract the word's score of the real document, $\text{score}_{\text{word,real}}$ from word's score of fake document, $\text{score}_{\text{word,fake}}$ to get the score of how likely the word belongs to fake documents.

$$\begin{array}{lc} & \text{Fake possibility} \\ \text{word}_1 & \left(\text{score}_{\text{word}_1,\text{fake}} - \text{score}_{\text{word}_1,\text{real}} \right) \\ \text{word}_2 & \text{score}_{\text{word}_2,\text{fake}} - \text{score}_{\text{word}_2,\text{real}} \\ \text{word}_3 & \text{score}_{\text{word}_3,\text{fake}} - \text{score}_{\text{word}_3,\text{real}} \\ \vdots & \vdots \end{array}$$

We define different amounts x of important fake words to build different size of keyword dictionary

for fake news i.e. we sort the fake possibility series from largest to smallest and pick x words with top x large fake possibility. Also, the size of the dictionary represents the size of feature vectors that we are going to construct.

After getting the keyword dictionary for fake news, we convert each tweet (similarly, we treat the text and all the replies with the same idx as the same tweet.) into a numerical vector. Each vector is an array recording whether the tweet contains the important fake words or not, and the array's index represents the location of the word in the dictionary. If a tweet contains a word in the dictionary, we set the corresponding array's element to 1. Otherwise, we set it to 0.

For example, suppose we set the dictionary size to 5, and the keyword dictionary is ['news', 'fake', 'trump', 'media', 'believe']. Given a tweet with the text "Trump just recognized Bitcoin as legal currency.", the first reply "Empty head lied pathetic wonder called fake news earned title lies.", and the second reply "I can't believe this news.". We can set this tweet's feature vector as [1,1,1,0,1].

Finally, we replace string labels with numerical digits. The 'fake' label transfers to number 0 and the real 'label' transfers to number 1.

3.2.3 Classical Machine Learning Models

The last step is much simpler. Since we have gotten the feature vector of each tweet, we just need to fit them into a classical machine learning model and predict the test data's labels to calculate the performance. We use a naive Bayes classifier and SVM model to do the classification. Naive Bayes classifier assumes that each feature is independent of any other feature, e.g. fake keywords of 'fake' and 'trump' have independently contributed to the fake label, and there is no correlation between them. It's a straightforward model and can capture non-linear decision boundaries. However, the naive assumption of conditional independence may penalize the performance, because fake keywords exist or not may affect the other fake keyword's appearance, especially in this large-scale complicated textual dataset. Hence, we use the SVM model to tackle this model's drawback. An SVM model is likewise capable of capturing the non-linear relationship using kernel function which projects data onto high dimensional feature space but conversely can handle the dependencies existing between features. As mentioned above, there is an existing package in python named scikit-learn for us to implement these models, so we simply use them to do the classification.

4 Experiment

4.1 BERT

After the first time we try to train the BERT model by using the hold data we get a bad result. We think that might be cause by data imbalance so we do some experiment to test it. We test two training data setting witch are imbalance data(normal setting) and balance data(sample from origin data) the train and test split rate is 4:1. Noted that we only use about 1/5 of the train data to do the training for saving time. The following figure is the training loss of both setting.

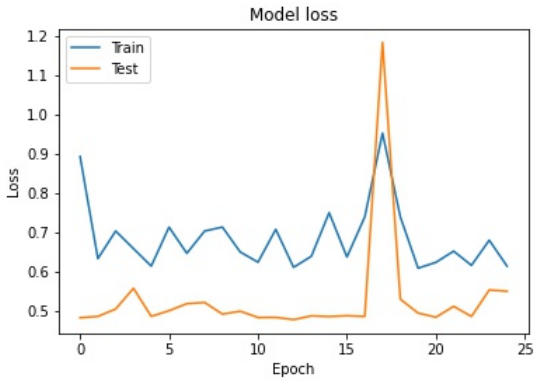

Figure 1: Trainning loss with imbalance data

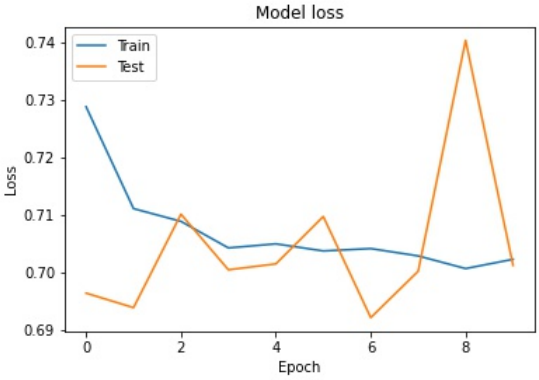

Figure 2: Trainning loss with balance data

We can see that there is no improvement of the training loss. So the bad performance is not cuase by data imbalance problem.

Doing experiment on BERT model is time consuming so we didn't do more experiment to find out the reason of the failure of BRET. We try to focus on other method.

4.2 ML

Here are the results of two ML models in the development phase. The performance is calculated by F1-score. As we mentioned in the method section, the SVM model is more powerful than the naive Bayes classifier for detecting fake news in general.

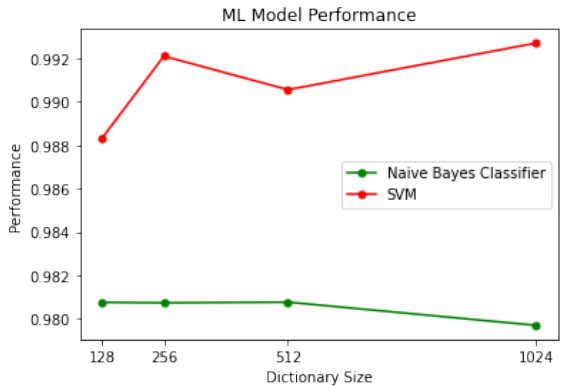

Figure 3: The ML model performances in dev. phase

In addition, we use different amounts (128, 256, 512, 1024) of fake keywords to see whether a more number of keywords are more helpful for improving the performance. The results suggest that the naive Bayes classifier has better performance using fewer keywords, and SVM has better performance using more keywords. We think that the reason behind this is because the naive Bayes classifier doesn't allow the dependency between each feature. Remember that we use whether each keyword existing or not as our feature. That is, keywords are the features, and more keywords mean more features. This increases the chance of covariation. Hence, an excessive amount of keywords make this model, not affordable to handle the complicated relations and cause ineffective performance. In opposite, given more keywords, SVM makes use of the kernel function to work better to deal with the correlation between them. The kernel function of the SVM model is the RBF kernel function with gamma equal to 1/ (number of features * variance of features). The SVM model with 1024 keywords has the best performance. It seems that we get a good performance in the development phase. However, our performance in the evaluation phase is not that good. Our f1-score is 0.5657, and the ranking is 19-th.

5 Discussion

The reason we choose the models of Scikit-Learn is that the Python provides convenient packages for us to use. Compared to some machine learning models, there is no need to code by ourselves.

We mention embedding techniques in the related work section. The following are four mentioned techniques, which are respectively the Bag-of-words, the tf-idf, the Word2Vec, and the GloVe. Speaking of the Bag-of-words, if we chose it as our solution, the more words in the corpus, the higher the dimension. We do experiments with the Colab, and the memory is not enough to store the whole corpus in the Twitter news when our program is running. As a result, we finally choose the other embedding technique. As for tf-idf, we referred to the reason why we opt for it earlier. It is worth a mention that tf-idf is widely used in document classification. After obtaining important words, we can dramatically save the memory in later steps. At last, we still have the remaining two techniques. It seems that we should also try the Word2Vec and the GloVe because they consider the semantic meanings of a word. Besides, the previous two, Bag-of-words and the tf-idf, don't get any semantic meaning from words. Therefore, we can also give them a try and conduct experiments with them in the future.

We note that we also attempt the Bert technique but there is no satisfying result. We try to figure out the reason for the failure but in vain. At first, we discover that there is a data imbalance problem between the labels of real news and fake news. Hence, we suspect that it is one of the reasons for failure in the beginning. Nevertheless, after sampling the original data and conducting experiments with the Bert again, we consider the failure is not related to the data imbalance problem. Later, we suspect that there is a domain gap problem in the data provided by the organizer due to the decline in performance in phrase II, and it may be one of the reasons for failure.

6 Conclusion and Future Work

We have tried Bert model and classical machine learning model, and finally find that the SVM model with 1024 keywords has the best performance in development phase. Nevertheless, this model doesn't have a good result in evaluation phase.

Besides the effort we have made in this project, there are many other techniques that may have the chance to help us further enhancement our performance.

Firstly, as mentioned before, we have tried our best to figure out why our Bert does not perform well, and we can further improve it. Secondly, Scikit-Learn not only provides the Naive Bayes and the SVM but also others like the KMeans and the EM, so we can give them a chance to try and maybe they can help us get better performance.

From the experiments of other participants in the Fake EmoReact 2021 Challenge(Student), it seems that other machine learning models like CNN and LSTM are worth research.

7 Task Allocation

7.1 309551177 Chia-Wei Hsu

1. Team leader: registering the competition, figuring out the competition rules, communicating with the competition holder
2. Literature search: searching for codes or papers related to fake news detection
3. Coding ML method and experiment: preprocessing, feature engineering, naive Bayes classifier and SVM model implementation
4. Writting the report: Method section's ML, Experiment section's ML, part of the conclusion

7.2 309551179 Ting-Xin Lin

1. Presenting the report: Making slides and a presentation in class
2. Writing the report : Including Introduction, Related Work, Discussion, and Conclusion and Future Work

7.3 309551038 Yu-Lin Lu

1. Coding BERT and doing experiment with it
2. Doing some data analysis
3. Writing the report: Including BERT related part

8 Question and Answer

We don't get any question.

References

S. M. Dadgar, Mohammad Shirzad Araghi, and M. M. Farahani. 2016. A novel text mining approach based on tf-idf and support vector machine for news classification. *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, pages 112–116.

Mykhailo Granik and Volodymyr Mesyura. 2017. Fake news detection using naive bayes classifier. In *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, pages 900–903.

Tommy Huang. 2018. Machine learning example for "sms spam collection dataset".

Heejung Jwa, Dongsuk Oh, Kinam Park, Jang Mook Kang, and Heuiseok Lim. 2019. exbake: Automatic fake news detection model based on bidirectional encoder representations from transformers (bert). *Applied Sciences*, 9(19).

Sharmila Polamuri. 2020. Most popular word embedding techniques in nlp.

Napas Udomsak. 2015. How do the naive bayes classifier and the support vector machine compare in their ability to forecast the stock exchange of thailand? *arXiv preprint arXiv:1511.08987*.