Convolutional Neural Networks for Fake Tweet Detection

I-Jung Hsu
National Yang Ming Chiao Tung University
tnt.cs07@nycu.edu.tw

1 Introduction

Misinformation is spread through the Internet very quickly. Those fake news may cause the society unsettling. Traditional way to fight against fake news is to examine every text manually. However, this is not enough in the era of information explosion. In order to solve this problem, people use NLP techniques to distinguish fake news.

The topics discussed by the crowd can vary in a short period of time, i.e. they are time-dependent. Therefore, if our model has strong ability to detect fake news but takes a lot of time to train or fine-tune, then it is probably not suitable for this task — by the time we finish training, the contents discussed are already different from what we used to train!

Therefore, we hope to develop a method, which needs few time to train and performs almost as good as those large models.

The model in this work uses simple Convolutional Neural Networks (CNN). It can be trained within 5 minutes, and does well on the Fake-EmoReact 2021 Challenge Round 1 (Student).
(F1 score: 0.8219, 3rd place, part of the scoreboard is on Figure 1.)

On the other hand, since we wish to build a model that updates its dataset frequently. We might not have enough data for training if the period is short. Therefore, we want to explore how much data is enough for our CNN model to achieve acceptable performance. And see if data augmentation methods can be implemented to achieve same accuracy when using less data.

2 Related Work

Fine-tuning a BERT model is much more inexpensive than training one from scratch, but it's still not fast enough and requires high computing power.

#	User	Entries	Date of Last Entry	Team Name	Precision score ▲	Recall score ▲	F1 score ▲	Detailed Results
1	ccc_gogo	8	06/02/21		0.8532 (2)	0.8456 (1)	0.8435 (1)	View
2	ChenMian	8	06/01/21	Team Edward	0.8399 (3)	0.8334 (2)	0.8314 (2)	View
3	Papa	11	06/01/21	Team Papa	0.8300 (4)	0.8239 (3)	0.8219 (3)	View

Figure 1: Scoreboard of Evaluation Phase of Fake-Emo React 2021 Challenge Round 1 (Student).

According to the researchers of BERT (Jacob et al., 2019), fine-tuning BERT needs a few hours on a GPU.

Therefore, an alternative should be found, that is CNN. CNN have been showed to be successful on text classification problem (Yoon, 2014). The experiments tell us simple CNN can achieve high accuracy in SST-2 problem.

Before training, we also implement some data augmentation methods. These methods in NLP are very different from what people often used in computer vision. Since if we flip a sentence, for example, its structure is changed and the meaning may be totally different. This work referenced 2 ways of data augmentation sorted in an article by Shahul, 2021.

3 Datasets

For the experiments on "how much data is enough for our CNN model to achieve acceptable performance" and "whether the 2 data augmentation methods help", use the dataset of Fake-EmoReact 2021 Challenge Round 2. The details of dataset are on the official website.[1]

4 Method

4.1 Data Preprocessing

a. Data Cleaning

All urls, punctuations, not alphabetic tokens and stopwords are removed. GIF data and mp4. files are not used, i.e. only texts and replies are considered.

b. Data Selection and Balancing

Since there might be more than one replies to one text, some actions should be taken to avoid same texts appear in both training and testing set. This work randomly chose one reply for one text, and append it after the text.

The number of "real" texts (23134) are much more than the number of "fake" texts (3218). To avoid the phenomenon that the model having the bias of outputting all "real", simply use same number of real and fake texts for training.

c. Train-Test Split

When doing the experiments on "how much data is enough", change the portion used for training and the rest are for testing.

4.2 Model

The model contains only 6-layers — the embedding layer encodes an input sequence to a vector; the CNN layer extracts features; the max-pooling layer helps to reduce size and extract more important features; the flatten layer converts 2D vector to 1D. The whole structure of the model is shown in Figure 2.

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 82, 100)	2201100
conv1d (Conv1D)	(None, 75, 32)	25632
max_pooling1d (MaxPooling1D)	(None, 37, 32)	0
flatten (Flatten)	(None, 1184)	0
dense (Dense)	(None, 10)	11850
dense_1 (Dense)	(None, 1)	11

Total params: 2,238,593
Trainable params: 2,238,593
Non-trainable params: 0

Figure 2: Model Summary of the Convolutional Neural Networks.

[1] https://sites.google.com/view/covidfake-emoreact-2021/shared-task/datasetregistration?authuser=0

4.3 Data Augmentation Methods

a. Back Translation

In this method, we translate the contents to other language and translate it back to English. By this way, we can get a sentence using different words to express the same things as the original one.

This work utilized the GoogleTrans API and chose German to be the middle language because it is the closest language to English.

b. NLP Aug (Word Substitution)

In this method, we use synonym word augmenter. It replaces n words in our sentences with similar words.

5 Experiment

All of the following experiments are done with the same model and hyperparameters as specified in section 4.2, and all of them are trained for 6 epochs.

During the data preprocessing section, random shuffling has been done and the data has been saved. After that, we use the shuffled data to do the following experiments and never change its order. In this way, we can ensure that the improvement when applying data augmentation is not caused by luckily accessing different and more useful data.

5.1 Relationships Between Number of Data and Accuracy

In this experiment, we want to find out how much data can achieve "not bad" accuracy. Here, "not bad" is defined as the difference between the result's F1-score and the best result's not exceeding 0.05.

All results are shown in Table 1. The best result occurs when using 2500 samples, so a result is "bad" if it's F1-score is lower than 0.7822.

The results when using 125 and 250 samples for training are bad. 500 samples are enough to achieve a not bad result.

num of data	precision	recall	F1-score
3000	0.8283	0.8345	0.8308
2500	**0.8302**	**0.8348**	**0.8322**
2000	0.8181	0.8271	0.8210
1500	0.8266	0.8207	0.8233
1000	0.8164	0.8235	0.8191
500	0.8164	0.8235	0.8191
250	0.7618	0.7717	0.7633
125	0.6799	0.6508	0.5937

Table 1: Results for using 3000, 2500, 2000, 1500, 1000, 500, 250, 125 samples as training set

These experiments, even using the largest scale of data, i.e. 3000 samples, can be finished within 5 minutes.

5.2 Data Augmentation Method - Back Translation

Back translation takes much more time than other normal data augmentation methods because the process sends request to Google Translate and receive the translated result. And if we send requests too frequently, we will get a "HTTP error 429 (too many requests)" and the procedure will be terminated.

Some actions need to be taken to avoid the HTTP problem and also to enhance the efficiency — Separate the data into 6 subsets. Open 6 kernels, run back translation simultaneously and merge them after.

The results are shown in Table 2. We can see that back translation really helps to increase F1-score when using fewer data. However, when having plenty of data, the results of using back translation are slightly lower than those not using this technique. This is probably caused by overfitting.

num of data	precision	recall	F1-score
3000	0.8278	0.8324	0.8298
2500	**0.8279**	**0.8337**	**0.8303**
2000	0.8195	0.8155	0.8173
1500	0.8108	0.8195	0.8136
1000	0.8087	0.8179	0.8115
500	0.7969	0.8008	0.7986
250	0.7752	0.7820	0.7776
125	0.7435	0.7539	0.7393

Table 2: Results for using 3000, 2500, 2000, 1500, 1000, 500, 250, 125 back-translation-augmented samples as training set

5.3 Data Augmentation Method - Word Substitution

Same as in section 5.2, the results in Table 3. shows that word substitution is good for fewer data, but not that necessary if we have plenty of data.

num of data	precision	recall	F1-score
3000	0.8254	0.8359	0.8283
2500	0.8254	0.8359	0.8283
2000	**0.8297**	**0.8389**	**0.8328**
1500	0.8253	0.8360	0.8283
1000	0.8134	0.8258	0.8151
500	0.8007	0.8029	0.8017
250	0.7707	0.7823	0.7682
125	0.7359	0.7378	0.7093

Table 3: Results for using 3000, 2500, 2000, 1500, 1000, 500, 250, 125 word-substitution-augmented samples as training set

5.4 Comparisons and Discussion

For both 125 and 250 samples, using back translation increases more F1-score than using word substitution.

However, for 500+ samples, most F1-score of using back translation also drops more than using word substitution. Although the global best result occurs in 2000 samples using word substitution, we should ignore it.

Overall, to achieve the best result, we should use back translation if we have only 125 or 250 samples. There is no need to use any of the two data augmentation methods when having plenty of data.

6 Conclusion

The work demonstrates that simple Convolutional Neural Networks can perform well on Fake Tweet Detection with merely 500 samples in the data and can be trained within 5 minutes.

For the cases that the amount of data is not enough, NLP data augmentation methods can increase the performance of the CNN model in a large extent.

In the future, we may work on the following experiments:

a. explore more data augmentation methods

b. mix several data augmentation methods

c. adjust the hyperparameters of the Convolutional Neural Networks

d. find some methods that can be applied to increase the accuracy when having plenty of data

Work Division

All of the work is done by 0713309.

Question and Answer

There were no question asked during my presentation in the class.

Reference

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification

Shahul ES. 2021. Data Augmentation in NLP: Best Practices From a Kaggle Master