Rumour Detection using Neural Network

2021 COMP90042 1022880

Abstract

This project proposes a rumour a detection model on the Twitter network to analyse content or tweets related to COVID-19. A rumour detection Natural language Processing model is developed using a general tree structure tweet dataset to find the optimal way of processing the natural language. Moreover, this identifies the best model to predict (predict what?) using the same dataset. Finally, using the developed pipeline, an in-depth analysis is provided using a dataset specific to COVID-19 topics.

1 Introduction

This paper addresses how to build a rumour detection system on a Twitter network and analyses COVID-19 related tweets using the developed model. To build the detection model, a dataset which has a main tweet and a subset of re-tweets was assessed. Each set was then labelled as non-rumour or rumour based on the trained neural network. The pipeline consists of two main parts: a feature processing and a supervised classification model.

2 Data Processing

A. Baseline

TFID Vectorizer[1] with the text of main tweets was adopted as a feature set, and a classic sequential tensorFlow network[2] (“Classic Model”) was adopted as a model for a baseline pipeline. As Table 3-1 shows, the baseline performed at accuracy 0.289 and took 57.99 seconds for training and prediction.

B. Text Data

Considering that the dataset is Twitter data, the data pipeline focused mainly on text features and text pre-processing.

a) Lemmatization

As Stemming is not always interpretable, Lemmatization[3] was used so that a prediction model can focus on the real meaning of the vocabularies using POS-tagging[4]. Lemmatization aims to remove inflectional endings to return a common base which is lemma. As Table B-1 shows, it trimmed the following words:

faces: face, attacks: attack, wounded: wound, shooting: shoot, incidents: incident, investigated: investigate

b) Removing noises

Overfitting[5] can be resolved to some degree by cutting down the size of the feature set. The first attempt after the lemmatization showed overfitted result. Hence, some extra process methods were chosen to refine the word tokens and cut the number of the feature to avoid the overfitting problem.

Stop Words

A stop word[6] is a commonly used word that does not add much meaning to the text analysis (e.g., “the”, “a”, “an”, “in”). Stop word was used to shorten the final tweet vector without losing the meaning of the original tweet.

URLs, @user and hashtags

Regular expression was chosen to get rid of URLs and @user tags. While these could be relevant, they take a high computational cost to extract actual meaningful tokens. However, hashtags were considered as an important feature as it contains a condensed thought of the author.

Punctuation & Special Symbols

Similar to the stopwords, since punctuation and special symbols do not provide much meaning to analyse the text, these were removed from the original text.

Lowercasing

As neural network (NN)[7] cannot treat the inputs 'Low' and 'low' as a same word, lowercasing is a critical method to feed NN with unified word occurrence. Also, this can help with the sparsity issues. Hence, Lowercasing is applied for each token.

After these steps, the tweet tokens decreased its size 51.2% from (4641, 10807) to (4641, 5535) as Figure B-1 shows.

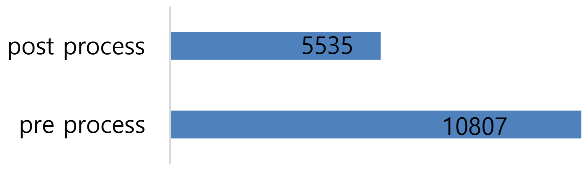

Figure B-1. Decreased size of tokens

As Figure B-2 illustrates, the model had an overfitting problem. After reducing the feature size using noise removing techniques mentioned above, the model showed a stable result without an overfitting problem (Figure B-3).

Table 3-1 shows that under the same classical model, applying the text pre-processing enhanced its accuracy by 20% and reduced its computational time by 46.12%

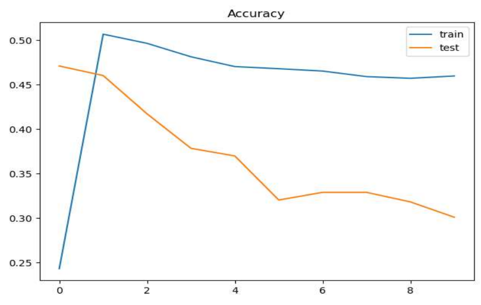

Figure B-2. Accuracy without a pre-processing

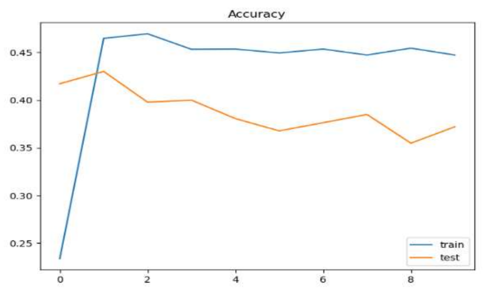

Figure B-3. Accuracy with a pre-processing

Sample 1
'France faces 2 hostage-taking attacks; Paris kosher market attack has 5 hostages, some wounded: http://t.co/dWlaFSDKjL'
'france face 2 hostage-taking attack paris kosher market attack 5 hostage wound'
Sample 2
'UPDATE: Police: "Several shooting incidents" being investigated in downtown Ottawa: http://t.co/9MAAywtu03 http://t.co/wW0aQmzRIb'
'update police several shoot incident investigate downtown ottawa'

Table B-1. Pre and post processed tweets

c) Vectorization

As will be expounded in the next section, Neural Network (NN)[7] easily outperforms other models but it has a downside as it cannot get a dynamic input size. To resolve this, two methods are compared below: .

The CountVectorizer[8] has a sparsity issue and the method only extracts word count and not actual meanings. Conversely, TF-IDF Vectorizer counts word frequency and normalises the count of common words such as "the", "I". Therefore, the TF-IDF Vectorizer was used.

d) Sentiment Analysis

Sentiment analysis reveals the tone of the sentence and the author's characteristics. After the POS-tagging, lemmatisation and noise cancelling, two sentiment results – Positive or Negative – were added to the feature set. Table 3-1 shows that sentiment increased the model accuracy by 8%.

C. Other features

In addition to the text features, the following features were considered:

a) Posting Time

Posting time is assumed as a feature which can distinguish user's posting trend and tendency.

b) Number of Retweets

The volume of retweets was added as a feature as it shows the reach of the main tweet (e.g., is spreadable and gossip-like).

Empirically these features showed slight increases in the final accuracy.

D. Labels and Distribution

Non-Rumour	Rumour
3058	1583
0.66	0.34

Table D-1. Label distribution of Training set

The trainset's label distribution was not even. Random sampling was applied to ease the unfairness of the dataset.

To reduce the cost of computing, both labels were replaced into 1 and 0 respectively.

3 Model Selection & Performance

Classic NN was implemented first as a baseline using two dense layers and one output layer. The task's aim, however, was understanding the context and nuance of serialised text tokens. For the example sentence below, simply counting the occurrence of words generates a result of analysis into something 'Negative' as the sentence contains negative words dominantly. On the contrary, Long Short-Term Memory (LSTM)[9] can derive contextual meaning locally.

'We don't believe our government, but they are not doing bad.'

Therefore, LSTM was selected as a final model as it has the best parameters for low cost of computation without deteriorating the performance of the model?

```
MAX_NB_WORDS = 5000
MAX_SEQUENCE_LENGTH = 250
EMBEDDING_DIM = 10
```

Process	Sentiment	Model	Accuracy	Time
X	X	Classic	0.289	57.99
O	X	Classic	0.345	26.75
O	O	Classic	0.374	26.83
O	O	LSTM	0.678	34.55

Table 3-1. Performance results

Layer (type)	Output Shape	Param #
embedding	(None, 5609, 10)	50000
spatial_dropout1d	(None, 5609, 10)	0
lstm	(None, 10)	840
dense (Dense)	(None, 1)	11
Total params: 50,851		

Table 3-2. LSTM Model Summary

The model performed with an accuracy of 0.678 within 34.55 seconds with the validation dataset. On the Codalab platform, the model resulted in 0.78 Precision and 0.79 F1 value. The 0.81 Recall value shows that the system detects Rumour tweets fairly well.

4 Analysis on Covid-19 Tweets

In this section, a prediction is conducted on a tweet dataset mentioning COVID-19 topics using the model developed previously. Using the predicted result, an analysis of features that distinguish between rumour and non-rumour tweets is conducted.

A. POS-Tagging

Non-Rumour sample
[('president', 'NN'), ('trump', 'NNS'), ('say', 'VBP'), ('beginning', 'VBG'), … ('essentially', 'RB'), ('downplay', 'VB'), ('virus', 'NN'), ('early', 'JJ'), ('say', 'VBP'), ('know', 'NNS'), ('could', 'MD'), ('horrible', 'VB'), ('know', 'VB'), ('could', 'MD'), ('maybe', 'RB'), ('good', 'JJ')]
Rumour sample
[('khive', 'JJ'), ('become', 'NN'), ('toxic', 'JJ'), ('thing', 'NN'), ('twitter', 'NN'), ('absolutely', 'RB'), ('damage', 'VB'), ('general', 'JJ'), … ('check', 'NN'), ('people', 'NNS'), ('apply', 'VBP'), ('so-called',

'JJ'), ('founder', 'NN'), ('wish', 'NN'), ('covid', 'NN'), ('19', 'CD'), ('warren', 'NN')]

Table 4-1. POS-tagging result for Rumour and Non-Rumour

Both rumour and non-rumour data used noun form, followed by adjectives then verbs. However, Rumour data contained more adjectives (37%) and less nouns (47%) compared to non-rumour data. This aligns with the hashtag analysis which demonstrates that the hashtags in non-rumour dataset contained more specific noun words. As Table 4-1 shows, non-rumour sentences had more specific nouns as 'trump' and rumour sentences had more adjectives like 'absolutely', 'toxic'.

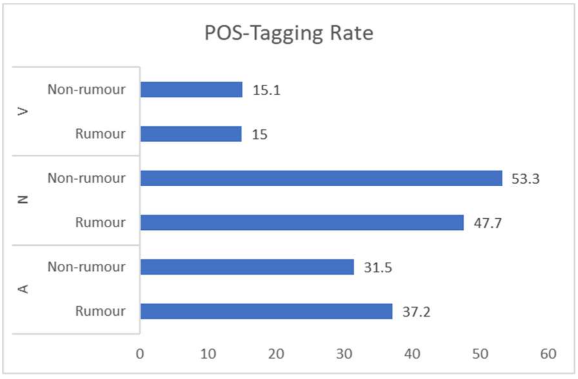

Figure 4-1. Types of tokens used in tweets

B. Hashtags

The number of hashtags in a post showed the biggest difference between rumour and non-rumour posts. One Rumour post used a hashtag by 12.5% when one Non-rumour post used a hashtag by 42.8%. Both tweets used #covid19 and #covid-19 dominantly but unlike the rumour data, non-rumour hashtags contained more specific words as **#tenet, #donald, #udmchiefwhip, #assam,** and **#nationaldayofprayer.**

C. Sentiment

Rumour	Non-Rumour
50%	46.4%

Table 4-2. Positive Sentiment rate of tweets

Intriguingly, the positive and negative sentiment of tweets were evenly distributed between Rumour and Non-Rumour data.

D. Other features

a) Posting Time

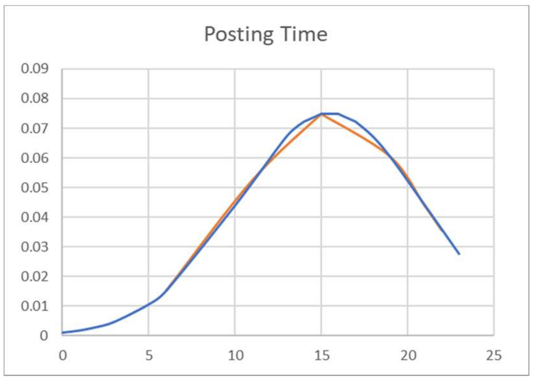

Figure 4-2. Posting Time for Rumour and Non-rumour data

As Figure 4-2 shows, the distribution of posting time showed similar behaviour between both datasets.

b) Retweets

Figure 4-3 shows that Rumour posts got 4.6 retweets per one main post on average. On the contrary, the number of retweets for Non-Rumour was 5 times that of Rumour posts.

From this, it is inferred that users tended to retweet what they actually believed.

	Hashtags(%)	Retweets	Time(24H)
Rumour	12.5	4.62	15.5
Non-rumour	42.8	22.92	14

Table 4-3. Other features for Rumour and Non-rumour data

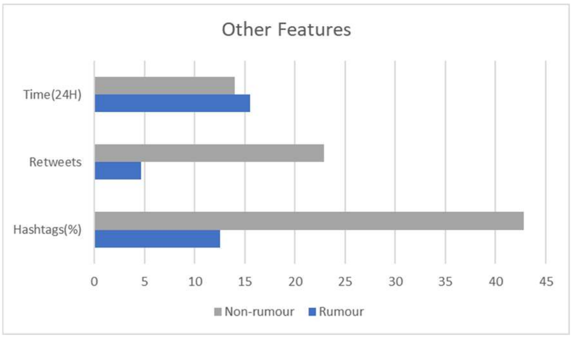

Figure 4-3. Other features for Rumour and Non-rumour data

REFERENCES

[1] Brownlee, J., 2021. *How to Encode Text Data for Machine Learning with scikit-learn*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/>

[2] TensorFlow. 2021. *The Sequential model | TensorFlow Core*. [online] Available at: <https://www.tensorflow.org/guide/keras/sequential_model>

[3] Nlp.stanford.edu. 2021. *Stemming and lemmatization*. [online] Available at: <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>

[4] Nltk.org. 2021. *5. Categorizing and Tagging Words*. [online] Available at: <http://www.nltk.org/book/ch05.html>

[5] Bejani, M.M., Ghatee, M. *A systematic review on overfitting control in shallow and deep neural networks.* Artif Intell Rev (2021). https://doi.org/10.1007/s10462-021-09975-1

[6] Kavita Ganesan, Ph.D. 2021. *What are Stop Words? | Kavita Ganesan, Ph.D*. [online] Available at: <https://kavita-ganesan.com/what-are-stop-words/#.YJvB8bUzaUk>

[7] Schmidhuber, J. (2015). *Deep Learning in Neural Networks: An Overview.* Neural Networks. 61: 85–117. arXiv:1404.7828.

[8] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[9] Graves A. (2012) *Long Short-Term Memory. In: Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence, vol 385. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24797-2_4