Rumour Detection and Analysis on Twitter

929715 **760488** **1283915**

Abstract

Rumour detection is the task of identifying rumours, that is, statements whose veracity is not quickly or ever confirmed, in utterances on social media platforms, such as Twitter. Most recent papers attempt to use a combination of feature-based and text-based methods to detect rumours. This report analyses various techniques used for feature engineering and query prediction. We experiment with tokens, sentiment and some Twitter-related features for feature engineering and Logistic Regression, Neural Networks and BERT to infer whether a Twitter thread is a rumour. Our results suggest that the best technique is to use BERTweet tokens as features on a BERTweet classifier, where BERTweet is a BERT model pretrained on a large corpus of English tweets.

1 Introduction

This paper explores the applications of Natural Language Processing and Machine Learning for the task of rumour detection and analysis. We exclusively explore tweets, comprised of both rumours and non-rumours, circulating on the social media platform, Twitter.

In 2019, Twitter had approximately 290 million monthly active users worldwide (Jay, 2022). Because of the platform's unique characteristics such as the 'retweet' feature, specific tweets can spread to millions of people over a relatively short period of time.

Rumours are defined as unverified statements that have the ability to rapidly spread misinformation and cause significant social impact (Pathak et al., 2020). During disasters and pandemics, people's need for information explodes; however, mass media cannot supply enough information to meet the suddenly huge demand. In the absence of formal information about a highly anxiety-inducing situation, people become motivated to share and evaluate information in order to explain the situation (Pathak et al., 2020).

Therefore, it is important to develop a system that detects rumours fast enough in order to prevent the spread of misinformation online. With millions of tweets posted and retweeted every day, it is impossible to assign humans for the task of rumour detection. It is also important to understand the dynamics of rumour discussion to foster the potentially positive uses of rumours and inhibit the negative ones (Pathak et al., 2020).

2 Related Work

In recent years, several papers related to rumour detection have been published. Chen et al. (2017) proposed a CNN model to detect rumours. Their aim was to detect the tweet sentiment and determine the truthfulness of a rumour. Ruchansky et al. (2017) suggest a model focusing primarily on three characteristics: the text content, the responses it receives, and source of the text. They utilise a RNN to capture the temporal user activity of the text and its responses. Another system that leverages BERT to identify the stance of a tweet in relation to its replies was published by Fajcik et al. (2019).

3 Method

3.1 Dataset

We have access to three datasets whereby each instance is a Twitter thread, that is, it consists of a series of tweets ID's, where the first ID is the source tweet and the others are the replies to that tweet, sorted randomly. The training and development datasets also contain the corresponding labels but the test dataset does not.

The training, development and test datasets consist of 1895, 632 and 558 instances respectively. We aim to train the models using the training instances, validate them using the development instances and use our best model to detect rumours on the test dataset.

3.2 Data Collection

We leverage Tweepy (Roesslein, 2020), a Python library that interacts with the Twitter API and can provide tweet objects, including all the textual data and metadata required to engineer features. We use Tweepy to extract the tweet objects for both the training and development datasets. For tweets that

cannot be retrieved by the API, we simply drop them. The same process is applied to the the test dataset, but when the API cannot retrieve a tweet, we extract the corresponding tweet object from the '*tweet-objects*' file provided.

3.3 Exploratory Data Analysis

We first explore the data to identify different patterns in the dataset and analyse the class balance. This process allows us to gain a better understanding of the data before making any assumption.

Upon examination of the language used in each tweet, we find that 91% of the tweets are written in English. As a result, it is reasonable to assume that the provided tweets mostly consist of English characters, grammar and expressions. Therefore, we preprocess the tweets using an English-based tokenizer, English stopwords and an English lemmatizer.

We also inspect the distribution of classes in both the training and development datasets, and find that both have the same ratio of rumour to non-rumour tweets of 1:4. Hence, we make the reasonable assumption that the testing dataset also contains the same ratio of rumour to non-rumour tweets. Consequently, a model trained on the training dataset would create a bias toward the majority class, that is, non-rumour tweets, which is also representative of the class balance in the test dataset.

3.4 Text Preprocessing

We preprocess each tweet through concatenation, tokenization, lowercasing, stopwords removal and lemmatization.

For each thread, the source tweet and all its replies are concatenated together. This is a more intuitive representation of a thread, which makes it easier for models to extract contextual information. We then use the TweetTokenizer from NLTK (Bird et al., 2009) to convert each tweet into a list of tokens. Each token letter is converted to lowercase to achieve a consistent representation. Stopwords from NLTK's English corpus, as well as Tweet-specific stopwords such as '@', 'http', '&amp', 'Q','A' (a list we manually compiled) are removed from the tokens list as these words appear too frequently and are therefore unimportant. Lastly, each token is converted to its lemmatized form according to NLTK's Wordnet's corpus. Lemmatization is performed so that inflected forms of a word can be uniquely represented.

3.5 Feature Engineering

1. Is user verified?: Verified Twitter accounts must be 'authentic, notable and active' (Tian et al., 2020), so we hypothesise that including a feature that shows whether the user of the source tweet is verified or not would be indicative of whether the thread is a rumour.

2. Sentiment score: The sentiment of the text data using Python's TextBlob, which is a library for processing textual data (Loria, 2018). This provides a polarity score ranging from -1 to 1 indicative of how negative or positive the sentiment of a text is. The inclusion of a sentiment-based feature is based on our hypothesis that replies to rumours are more likely to have a negative sentiment as compared to replies to non-rumours.

3. Most frequent tokens: The X most frequently occurring tokens are one-hot encoded. We experiment using with 500 and 1000 as values for X, and find that the top 1000 tokens are the most representative of whether a thread is a rumour or not. We also believe that certain words appear more frequently in rumour threads than in non-rumour threads.

4. BERT tokens: Tweets are fundamentally distinct from formal language texts in terms of length, irregular vocabulary, and use of abbreviations, typographical errors and hashtags (Nguyen et al., 2020). Therefore, using a language model pre-trained on English tweets would lead to a better contextual.

We use the BERTweet Tokenizer, provided by VinAI Research (Nguyen et al., 2020) to get tokens from input tweets in our datasets. This BERTweet Tokenizer has been pre-trained on 850M English Tweets (around 16B word tokens) streamed from 2012 to 2019. Since it was pre-trained on raw tweets, no preprocessing is required. The Tokenizer also translates URL's and emojis into representative word tokens. The output contains *input_ids*, which are the indices of the tokens in the vocabulary and are of size 128, padded and truncated as necessary. It also outputs the corresponding *attention_mask*, which helps the model to attend to the right tokens, and the *token_type_ids*, which indicate which tokens are in the 'question' and 'answer' part for a 'question-answering' setting.

We produce two sets of tokens: one with the tokens from the concatenation of the source and replies, and the other with the tokens from providing input pairs as (source, concatenation of the replies) analogous to a (question, answer) pair. The

intuition behind the first set is that the tokenizer should output an accurate contextual representation of a whole thread when treating the source and replies as one sentence. The motivation for the second set is that we want to treat the replies as an 'answer' to the 'question', that is, the source tweet, so that the tokens represent their relationship.

5. Others: We also consider the number of likes and the number of retweets on the source tweet as features, as well as the number of followers and number of tweets posted by the user of the source tweet.

3.6 Classifiers

Logistic Regression: The Logistic Regression model is a discriminative classifier; it models the conditional probability distribution of the labels given the features. We implement it for rumour detection as it is relatively simple to implement and easily interpretable. We want to assess whether a linear classifier with simple feature engineering performs well on our data. The C-value used is 1.3.

Neural Network: Neural networks model underlying patterns in datasets using hidden layers and non-linear activation functions. For our task of rumour classification, we experiment with a set of neural networks, all following this architecture:

- Input layer: a set of features with values between 0 and 1 (binary or scaled).
- 3 Hidden layers: 20% dropout rate and ReLU activation function.
- Output layer: 1 neuron with a sigmoid activation function, representing the probability that an instance is a rumour.

We experiment with several feature combinations, with aim to find the best feature set, as well as identify the features that are not useful.

The model is trained with early stopping based on the validation accuracy, with the number of epochs set to an arbitrarily large number of 4000. The optimiser used is Adam as it can handle sparse gradients on noisy problems and provides an efficient and stable procedure in updating neuron weights. A batch size of 32 is used on a CPU, and the loss function is Binary Cross Entropy, since our labels are binary.

BERT Classifier: The BERTweet Classifier is a model with a BERT architecture and a classification layer appended on top of the last Transformer layer (Nguyen, Vu and Tuan Nguyen, 2020). It takes as inputs the outputs from the BERTweet Tokenizer: (*input_ids*) and their corresponding *attention_mask*, and additionally the *token_type_ids* for a 'question-answering'-like task.

The main strength of BERT is its ability to account for a word's context, that is, it produces a different embedding for the same word depending on the words around it (Devlin et al., 2018). This is possible since BERT has two training objectives, a Masked Language Model and Next Sentence Prediction, and is based on Transformers, which uses attention to weight each input token to each output token dynamically (Devlin et al., 2018).

Since we are fine-tuning the model for our downstream task of rumour detection, the model is trained for only 10 epochs; we want to fit the weights on the softmax classification layer, while maintaining the parameters in the BERT architecture as those pre-trained parameters are tuned to extract context from the tokens in the form of word embeddings. We optimise the model using AdamW with a fixed learning rate of 5e-5, since it implements weight decay, and use a batch size of 16 across a 'NVIDIA Tesla K80' GPU. The loss function used is Binary Cross Entropy since we are dealing with a binary labels.

The model is trained on both sets of tokens produced by the BERTweet Tokenizer.

4 Results

This section presents the results of our experimental models upon inference on the development set. We define the results in Table 1, which denotes the features and model used, along with their accuracy, precision, recall and F1 scores on the development set. Note that upon evaluation, we drop the instances that the Twitter API could not retrieve.

We first define the following feature sets, which serves as a key for Table 1:

- V: Is user verified?
- S: Sentiment score.
- O: Others (number of likes, retweets, user's followers, user's posts).
- T_{500}: 500 most frequent tokens.
- T_{1000}: 1000 most frequent tokens.
- $T_{BERT_{concat}}$: BERT tokens (source and replies concatenated).
- $T_{BERT_{sep}}$: BERT tokens (source and replies separated).

Observing the results from Table 1, we notice that the BERT Classifier, when used on the BERT tokens on the (source, replies) pair, outperforms all other features and model combination in terms of accuracy, precision, recall and F1.

Features	Model	Accuracy	Precision	Recall	F1
$V + S + O + T_{500}$	Logistic Regression	0.92	0.91	0.92	0.92
$V + S + O + T_{1000}$	Logistic Regression	0.93	0.93	0.93	0.93
$S + T_{500}$	Neural Network	0.89	0.76	0.67	0.71
$S + T_{1000}$	Neural Network	0.91	0.77	0.81	0.79
$V + S + T_{1000}$	Neural Network	0.89	0.69	0.83	0.75
$V + S + O + T_{1000}$	Neural Network	0.89	0.70	0.81	0.75
$T_{BERT_{concat}}$	BERT Classifier	0.96	0.85	0.93	0.90
$T_{BERT_{sep}}$	BERT Classifier	0.98	0.97	0.94	0.96

Table 1: Results

For the Logistic Regression model and the Neural Networks, the feature set used are all dummy representations of the thread, rather than the context of the tweets themselves. The most frequent tokens feature provides a dummy contextual representation in the form of a bag of words. Although this provides a rough portrayal of words within context, it does not capture the semantics and syntactic relationships between the words in the tweets. We also notice that the most frequent tokens is the feature that is most representative of whether a thread is rumour as compared to the other features V, S and O; adding V and O to the neural network actually impairs the evaluation scores. This indicates that those features are not indicative of the of a thread of tweets being a rumour.

Further, Logistic Regression is a linear model. Since it is extremely unlikely that the data is linearly separable, Logistic Regression acts more as a baseline, rather than a quality inference model. Additionally, we notice that all Neural Network results perform worse than the Logistic Regression with the 1000 most frequent tokens. This might be due to the sparse representation of the tokens; as they are one-hot encoded, a lot of noise is introduced into the model, which harms the neural network's ability to memorise training samples because they are changing all of the time, resulting in smaller network weights and a more robust network with a low generalisation error.

In contrast, our BERT tokenizer is pretrained on a large corpus of tweets. This means that it understands the syntax and semantics of tweets and can provide appropriate tokens to be fed to the BERT classifier. The BERT classifier is also pretrained on the same set of tweets. Since BERT uses attention to capture the contextual representation of words, it is able output illustrative word embeddings. Those word embeddings are good features to pass on to the softmax classification layer for rumour inference. We also note that treating the Twitter threads as a (question, answer) pair upon tokenization in the form of (source, concatenation of replies) amounts to better evaluation results than with the concatenation of entire threads. The former allows BERT to get the context of the source tweet and the replies separately, but in relation to one another. This model performs the best since BERT essentially is trained to predict whether the replies are a rumour to the source.

Our final BERT model's predictions performed with an F1-score of 0.91358 and was ranked 11^{th} in the Kaggle Competition. These results are very satisfiable since the high F1-score suggests that the model performs well in terms of both precision and recall; a large proportion of the instances predicted as rumours were actual rumours and only a few rumour labels were missed. However, we jumped from 2^{nd} to 11^{th} in the private leaderboard, which suggests that more thorough evaluation on the development set should be performed to assess which predictions are the best in general.

5 Rumour Analysis on Covid-19 Tweets

In this section, we analyse 15962 Covid-19-related tweets with aim to understand the nature of Covid-19 rumours and how they differ to their non-rumour counterparts. Their labels are infered using our BERTweet Classifier. It classified 4400 tweets as rumours (28%) and 11562 tweets as non-rumours (72%).

Topics: Upon analysing the most frequently occurring tokens in the dataset, we find that tweets involving the words Covid-19 and former US President, Donald Trump, are the most common tokens. Based on the unigram counts of tokens: *coronavirus* (7482), *covid* (6548), *trump* (2284), *test* (2132), *case* (2089), *new* (1910), *death* (1510), *pan-*

demic (1334), *president* (1109), *health* (1007), and *positive* (861), are mentioned 28266 times in the 15962 analysed tweets, at a rate of 1.77 times per tweet. While Covid-19-related tokens appear in both rumour tweets and non-rumour tweets, they are disproportionally more frequent in non-rumour tweets (79%) as compared to the entire dataset. Besides, 75% of the occurrences of *trump* are found in rumour tweets. The bigram token analysis reveals similar insights with additional Trump-related tokens such as: *president trump*, *donald trump*, *trump say*, *white house*, and *trump coronavirus*, frequently appearing in rumour tweets. Similarly, Covid-19-related tokens such as: *positive coronavirus*, *social distance*, *wear mask*, *coronavirus outbreak*, appear frequently in non-rumour tweets.

Hashtags: When analysing the hashtags of the analysed tweets, we find that the most popular hashtags follow a similar pattern to the bigram insights; hashtags related to Donald Trump were frequently used in rumour tweets. Some of those hashtags include *#trumpownseverydeath*, *#stopairingtrump*, *#trumpmeltdown* and *#trumppressconference*, the majority of which seem to have a negative sentiment. In addition, none of the top 10 hashtags in non-rumour tweets are related to Donald Trump. Besides, the top 3 hashtags for both tweet types are *#covid19*, *#coronavirus* and *#breaking*, and the most popular hashtags for non-rumour tweets include other hashtags directly related to Covid-19 such as *#covid*, *#coronaviruspandemic* and *#covid19ph*.

Sentiment: According to the TextBlob polarity, Covid-19 tweets have an average sentiment of 0.06. In contrast, rumour tweets only have an average sentiment of 0.02. This suggests that those tweets generally demonstrate a more negative tone as compared to non-rumour tweets, which had an average sentiment of 0.07. There is also a distinction in the number of retweets for rumour tweets and non-rumour tweets. Rumour tweets have an average of 3918 retweets while non-rumour tweets only have an average of 2367 retweets. This suggests that users are more motivated to spread rumours than non-rumours, which could be dangerous depending on the content of the rumour. We also identified that on average, rumour tweets have 15173 likes while non-rumour tweets accumulate 8615 likes. However, the true cause of such behaviour is hard to discern; since we are not privy to how the Twitter feed's algorithm works, it is hard to identify the impact this could have on users

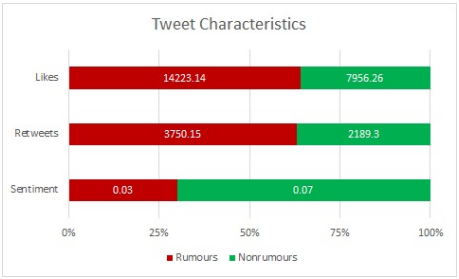

Figure 1: Covid-19 Tweet Characteristics

Users: We also analyse the users who posted the source tweets in terms of number of followers, number of tweets posted, and whether they are verified. 79% of the users are verified and both rumour tweets (77%) and non-rumour tweets (80%) approximately maintain this proportion, which signals that more screening should be performed before setting a user as 'verified'. However, the users posting rumour tweets have approximately 37% less followers than users posting non-rumour tweets. Users posting rumour tweets also posted 17% more statuses than users posting non-rumour tweets. However, 780 of the 6019 unique users in the dataset posted both rumours and non-rumours. On average, the Covid-19 tweets are retweeted 2620 times and have 9684 likes. Rumours have more retweets and likes than the average with 3918 and 15173, respectively. Conversely, non-rumours have less retweets and likes than the average with 2367 and 8615, respectively. This suggests that rumours attract more attention when circulating on social media.

6 Conclusion

In this report, we benchmark and provide a comparative analysis of various feature engineering and inference methods for the task of rumour detection. We also analyse the dynamics of Covid-19 tweets and find that rumours and non-rumours have distinct features. From the evaluation, we find that BERTweet tokens display the best results when paired with the BERTweet Classifier. This model performed with an F1-score of 0.91358 and ranked 11^{th} in the Kaggle Competition.

For future work, including the non-textual data such as the number of likes and the sentiment score, as additional features to the BERTweet tokens would make a more thorough evaluation. Furthermore, increasing the maximum length for the BERTweet tokens might lead to better results since it would capture more context from the tweets.

7 Team Contribution

929715 I have worked mainly on the BERTweet method of the project, which includes engineering the BERTweet tokens and training the BERTweet classifier. I also performed the Exploratory Data Analysis and wrote the Results analysis, Related Work section and the Abstract.

Working with 760488 and 1283915 was very productive. We managed to hold frequent online meetings, which were concise and efficient, and we all respected the deadlines set by the team. It was easy to communicate with them and I believe we all contributed equally to the project.

760488 I mainly worked on the neural networks and the analysis of the Covid-19 dataset. The work on the neural networks involved preprocessing, feature engineering, training, and evaluating. The work on the Covid-19 dataset involved data retrieval and analysis. I believe we all contributed equally to the project.

1283915 I browsed the Tweepy library to find how to access non-truncated text of tweets and explored the data to find useful features outside of the text data. I also implemented some preprocessing and feature engineering for the test and development sets as well as I implemented the logistic regression model. I believe we all contributed equally to the project.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Yi-Chin Chen, Zhao-Yang Liu, and Hung-Yu Kao. 2017. IKM at SemEval-2017 task 8: Convolutional neural networks for stance detection and rumor verification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 465–469, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Martin Fajcik, Pavel Smrz, and Lukas Burget. 2019. BUT-FIT at SemEval-2019 task 7: Determining the rumour stance with pre-trained deep bidirectional transformers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1097–1104, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Allan Jay. 2022. Number of twitter users 2022/2023: Demographics, breakdowns predictions - financesonline.com.

Steven Loria. 2018. textblob documentation. *Release 0.15*, 2.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. Bertweet: A pre-trained language model for english tweets. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.

Ajeet Ram Pathak, Aditee Mahajan, Keshav Singh, Aishwarya Patil, and Anusha Nair. 2020. Analysis of techniques for rumor detection in social media. *Procedia Computer Science*, 167:2286–2296. International Conference on Computational Intelligence and Data Science.

Joshua Roesslein. 2020. Tweepy: Twitter for python! *URL: https://github.com/tweepy/tweepy*.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. CIKM '17, New York, NY, USA. Association for Computing Machinery.

Lin Tian, Xiuzhen Zhang, Yan Wang, and Huan Liu. 2020. Early detection of rumours on twitter via stance transfer learning. In *Advances in Information Retrieval*, pages 575–588, Cham. Springer International Publishing.