Detecting disasters from Twitter data

Use of modern Natural Language Processing to classify Tweets

Ernesto Monroy
CID: 01010397

Word Count 4,670

Imperial College Business School
Business Analytics Online
31 August 2020

Detecting disasters from Twitter data:
Use of modern Natural Language Processing to classify Tweets

August 31, 2020

Abstract

Twitter has become an important channel for users to publish data about their experiences, this presents an opportunity for emergency response teams and disaster managers to detect events relevant to them. In this study a series of statistical and neural language models that detect the mention of a disaster in tweets are proposed. The models were trained and tested on a public dataset from Kaggle. The best model was a pre-trained BERT network that achieved a 90.87% F1 score and 88.94% accuracy, highlighting that current open source technology is able to successfully detect emergency situation in social media. In addition, this study concluded that other less computationally intensive methods such as Support Vector Machines can also achieve decent scores between 79% and 80%. This report was also intended to serve as a guide and review of the literature on the subject of classification tasks on short text data.

1 Introduction

Smart phones and social media are ubiquitous, they have changed the way we share information and have created a constant and public live stream of text, images and videos that report what people see and experience around them. During emergency situations, these live streams have the potential of providing valuable information to emergency response teams (Diaz et al. 2020). However, the large scale automatic classification of textual data remains a challenge for most organisations looking to leverage social media data (Altnel & Ganiz 2018)

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) that focuses on enabling machines to understand human language (Fang et al. 2020). Although the field of NLP started gaining popularity in the 1990s with statistical methods, it was not until the last decade that advances in Deep Learning allowed the creation of models that can surpass human intelligence in language processing tasks such as translation, question-answering, information retrieval and text generation (Zhou et al. 2020). Both neural and statistical NLP models have become widely available to data scientists thanks to libraries such as NLTK, SciKit Learn, Tensorflow as well as pre-trained models such as GPT and BERT.

The present study uses modern NLP methods to detect Twitter posts that contain disaster information. This document is also intended to serve as a guide on text classification by reviewing and comparing 10 of the most common language models available to data scientists. Its important to mention that the study does not make a comparative study of the data pre-processing tasks since this is a large field of research that would require its own separate review.

The problem is approached in two steps, in the first, a literature review is performed to identify the models that have been applied to text classification tasks as well as the pre-processing steps needed to make the models work. The second step is to use the Kaggle Disaster Detection dataset to train and test these models and compare them using F1 scoring.

2 Literature Review

The use of NLP for the detection of disasters in text is a concept that is currently receiving significant attention from researchers and has proven useful to society. Diaz et al. (2020) mention that emergency services in the United States were seeking to use Twitter data to classify users according to their likelihood to provide disaster information. Abburu & Golla (2017) found that there was a need to build a standardised disasters database, where the current challenge is to be able to detect these events from news feeds and Wikipedia articles. Fang et al. (2020) worked with a construction company to detect near misses from safety reports with the intention of informing management and operators of potentially unsafe areas. In addition to classification, other research has started to look at information extraction tasks, one example is To et al. (2017) who proposed a methodology for mining disaster details (such as service disruptions, number of affected people, etc.) from social media. The need to extend to non-physical disasters is also documented, Behzadan et al. (2018) found that social media post classification could also benefit areas such cybersecurity and threat detection.

Before narrowing the research to individual models a broader understanding of the data pipeline of NLP projects was needed. Martinez (2010) gives an overview of the field and highlights that there are generally two parts to an NLP task, the pre-processing of the data, where text is converted into meaningful numerical representations (vectors), and the language models, that use these vectors to understand a language and perform tasks. Gomez-Perez (2020) gives an in depth overview of the most popular language models and divides them into 2 categories: Statistical and Neural.

Due to the scope of this study, only the most basic pre-processing tasks we researched. Assery et al. (2019) suggests a cleaning procedure for Twitter data involving the removal of punctuation, HTMLs, URLs and Emojis. To et al. (2017) suggests using a spell checker and a slang dictionary given that non-english words are commonly used in tweets. After the data is cleaned, Kamath (2019) uses Tokenisation to separate phrases into words and Part of Speech (POS) to tag each word with their role in the phrase (verb, noun, adjective, etc.). Martinez (2010) mention that the size of the lexicon (i.e. the unique words in the dataset) is a common issue that can be tackled with either lemmatising (grouping similar words under a common word) or stemming (extracting the root of the word). Strong evidence was found in support of lemmatising over stemming (Sit et al. (2019), Rostami & Mumivand (2014), Billal et al. (2016)), this is because since stems are not necessarily valid words, they are incompatible with further pre-processing techniques and vectorisation that relies on correct spelling.

After the cleaning procedure was established, the review of the literature focused on finding statistical language models. In their study Nazura & Muralidhara (2017) found that a Naive Bayes classifier achieved a high accuracy in determining the semantic classification of tweets. Both Wang & Manning (2012) and Phand & Phand (2017) performed a study to find a model for sentiment analysis on twitter and agreed that Support Vector Machines achieved the best results when compared to other statistical methods. In a similar study, Kanakaraj & Guddeti (2015) found that the most effective methods were the ensemble models that combined the output of other classifiers. Finally, in a study dedicated to finding hurricane information on Twitter, Assery et al. (2019) point out that the best model is largely dependent on the dataset being used. They indicate however, that Decision Trees and Linear Regression rank consistently amongst the top performers and K-Nearest Neighbours (KNN) amongst the worst.

Apart from the traditional statistical classifiers, this study aims to explore more modern approaches. In the last decade, following the success of deep learning, NLP has shifted from statistical methods to Neural Network methods (Zhou et al. 2020). Overall, we found three types of Neural Networks in the NLP domain: traditional feed-forward networks, Long Short-Term Memory (LSTM) and pre-trained models. Thanks to internal states, LSTM layers excel at tasks that require "memory" across a data-point (Hochreiter & Schmidhuber 1997). In their speech recognition experiment, Sundermeyer et al. (2015) found that LSTMs performed much better than traditional feed forward networks. Sit et al. (2019) successfully employed an LSTM model to extract disaster information from tweets. Pre-trained models, which are created using massive text volumes, are often overlooked and can provide a significant improvement from custom built networks (Turc et al. 2019). Amongst the best known pre-trained models is BERT, which Fang et al. (2020) successfully applied to the task of finding near misses in safety reports and Zhang et al. (2019) used for the evaluation of purchase intent on flight reservations.

3 The Dataset

The main dataset used in this study is a series of tweets obtained from the Kaggle competition "Real or not? NLP with Disaster Tweets". The features of this set consist of an index, a geolocation, the text of the tweet and the target class describing if the data-point refers to a disaster or not. The table below provides a top-level analysis of the distribution of the main properties of the tweets. Here we can observe that the disaster and no disaster populations are very similar, which means we need to analyse the individual texts in order to find their classification.

Table 1: Distribution of tweets

	Disaster	No Disaster	Total
	3271 (43%)	4324 (57%)	7613 (100%)
No Hashtag	2396 (73%)	3456 (80%)	5852 (77%)
Hashtag	875 (27%)	886 (20%)	1761 (23%)
No URL	1099 (34%)	2543 (59%)	3642 (48%)
URL	2172 (66%)	1799 (41%)	3971 (52%)
No Location	2196 (67%)	2884 (66%)	5080 (67%)
Location	1075 (33%)	1458 (34%)	2533 (33%)
No Mention	2605 (80%)	2999 (69%)	5604 (74%)
Mention	666 (20%)	1343 (31%)	2009 (26%)
Rare word average	75.68%	77.66%	76.53%

We consider "Rare words" those that are observed in less than 1% of the tweets. A surprising discovery was that the majority of the words in a tweet are rare, 76.52% across all tweets as shown above. This means that our problem is very sparse, a factor that will be significant to the performance of different models.

4 Methodology

This section highlights the different technical aspects of the study. The scoring techniques used in the model comparison and the final pre-processing methodology are described. As well, a brief overview of the statistical and neural language models are given.

4.1 Scoring

Based on observations from the literature (Hand & Christen 2018), and in order adhere to be able to compare the results with other Kaggle submissions, F1 was selected as the main scoring methods. This measure is the harmonic of the precision and recall scores, where precision measures the accuracy of positive predictions and recall the ratio of positive to negative samples (Beitzel 2006). In addition to the F1 score, the regular classification accuracy is also reported for convenience.

Since the aim of the paper is to also to serve as a practical guide on text classification, two additional measures were included. The first, the computational cost, is a discrete scored based on the server requirements needed to run the model: low means the code can be executed on computer with 4 GB RAM and 2 CPUs or less; medium requires up to 8 GB RAM 2 CPUs and high require up to 32 GB RAM and 8 CPUs. The other measure

included is the percentile ranking in the Kaggle competition (taken on the 10th of August 2020), this is intended to give an external benchmark on how good the model is compared to submissions from other members of the Kaggle community.

4.2 Feature Engineering

The literature review revealed that NLP models are heavily influenced by the feature engineering, making it a research topic in itself. It was also discovered that the order of the pre-processing tasks in NLP is important, as an example, punctuation removal in large text analysis would reduce the quality of Part of Speech tagging. Our final pre-processing pipeline, illustrated in Figure 1, is split into four parts: text cleaning, dimension reduction, test-train splitting and vectorising. In the text cleaning part we perform the following operations intended to yield a grammatically correct text that can be parsed in subsequent steps:

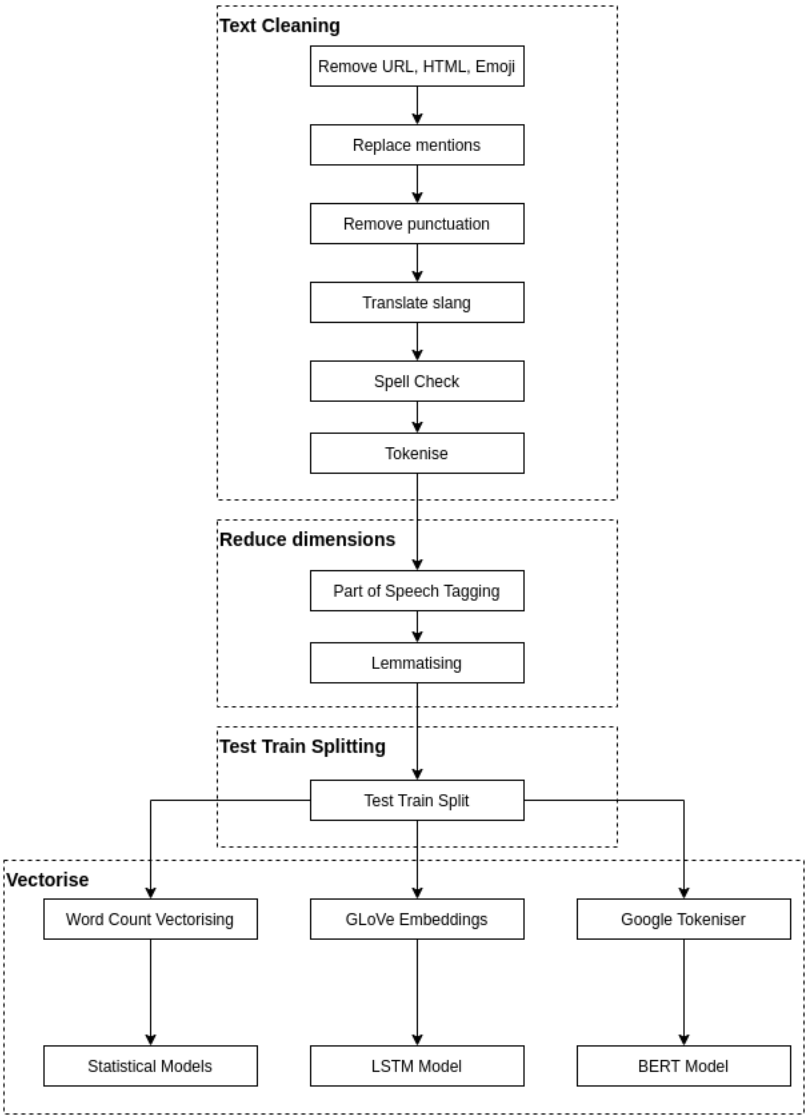
- Removing URLs, HTML, emojis and encoded special characters
- Replacing mentions with the word "someone"
- Removing punctuation
- Removing the hashtag symbol
- Slang translation
- Dictionary based spell checking
- Tokenisation: splitting the phrase into words

In addition to the above steps, we considered the use of Hashtag splitting and repeat letter correction as suggested by Assery et al. (2019). However, it was found that the occurrence of these errors were not common (only 0.5% and 0.8% in disaster and non-disaster tweets respectively) and thus the steps were not included.

In the dimension reduction part of the process, we took the correctly spelled arranged tokens obtained in the cleaning part and enriched them with their role in the phrase. This technique known as Part of Speech tagging (POS) is typically used to disambiguate words (for example the difference of the word "refuse" in the phrases "They may refuse you" and "Throw it in the refuse"). The next step was lemmatisation, which uses both the word and the POS tag to obtain the canonical form of each token (known as lemma) Green et al. (2009). As mentioned in the literature review, an alternative to lemmatisation is stemming, and although this process tends to reduce dimensions further (Manning et al. 2010), it would have yielded a set of data incompatible with LSTM and BERT models.

Before applying model specific pre-processing techniques we split the data into the test and train subsets, which made sure results from the different models were comparable. Vectorisation for the statistical models consisted in converting the set of tokens into a vector containing the count of each word. For the LSTM, an embedding transformation was used, this converts each word into a spatial vector that represents the meaning of the word, where similar words point in similar directions. The specific embedding dictionary used is the Stanford Global Vector for Word Representation (GloVe) as suggested by Pennington et al. (2014). For BERT, given it is a pre-trained model, it requires the input to be re-tokenised using the algorithm supplied by Google.

Figure 1: Feature engineering framework



4.3 Statistical Language Models

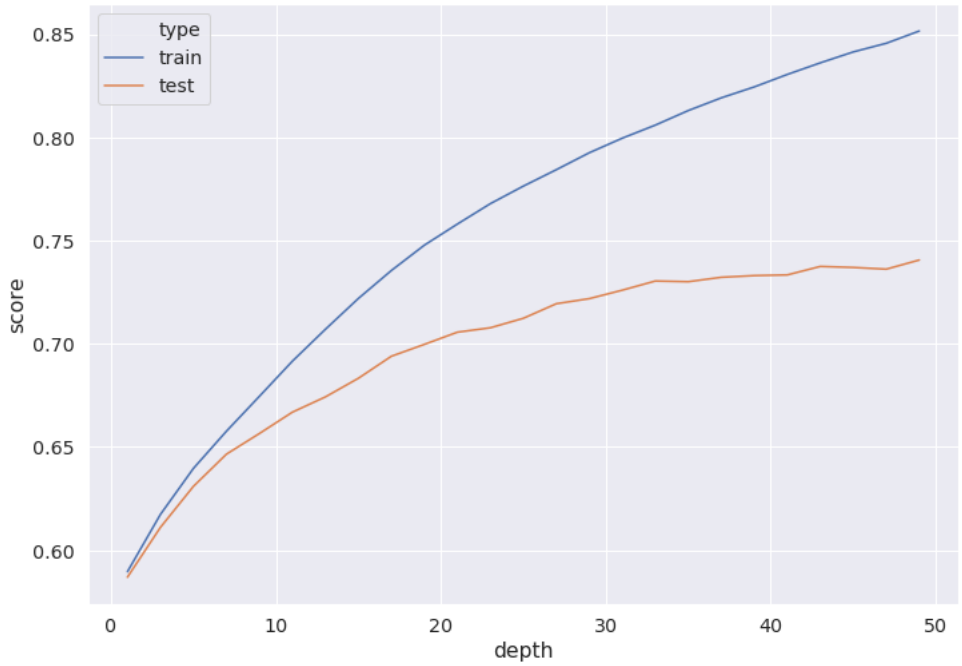
In the following paragraphs a brief description of the statistical models is provided. The implementation of all of these models was done using the scikit-learn library on a Python 3 Jupyter notebook hosted on an AWS SageMaker instance with 4 GB RAM and 2 CPUs.

Naive Bayes Two different classifiers were used: Bernoulli and Multinomial. Both models use conditional probability to understand the likelihood that a text belongs to a certain class given the individual words present in the text. It was decided not to include the Gaussian variant of the Naive Bayes classifier given it requires the frequencies to display a Gaussian distribution, which is not possible in short texts such as tweets.

Decision Tree Decision trees work by navigating a network of nodes, where the next node to visit is decided based on features of the data-point. The label is obtained once the navigation reaches an end node (leaf). This methodology is commonly used in Natural Language Processing in tasks where rigid rules can be applied, such as Part of Speech tagging and lemmatization and have the advantage of being faster than other statistical methods (Boros et al. 2017).

In order to optimise our Decision Tree algorithm, we performed a hyper-parameter grid search for tree depth. This obtained the scores for a range of tree-depths and yielded the setting that obtained the highest cross-validation accuracy. In our case, this was a depth of 45. Although this might seem large, its important to consider that the sparsity of the data is very high (Total words in lexicon 13,649) A sample of the learning curve for number of leaves is shown in Figure 2.

Figure 2: Decision Tree learning curve for depth



Support Vector Machine A Support Vector Machine (SVM) works by finding a set of hyperplanes that can separate the data points into its classes in a space where the dimensions are derived from the features of the data. This is similar to the perceptron algorithm with the exception that the SVM tries to maximise the separation distance. Friedman et al. (2001). Both SVMs and Naive Bayes Classifiers are often used in NLP tasks because of their ability to deal with high dimensional problems (Wang & Manning 2012). In terms of parameter tuning, it was found that the linear kernel performed better than the polynomial one.

KNN Although the literature review suggest this is the worst performing algorithm, it was decided to include K-Nearest Neighbours as a benchmark. KNN interprets each feature

as a space dimension, and assigns unlabelled data-points the same class as the majority of its K closest neighbours (based on Euclidean distance). Typically, this algorithm requires the features to be scaled in order to produce the best results, otherwise there would be a bias based on features that have higher numerical values (Trstenjak et al. 2014), however, since most of our vector values are 0 or 1 we disregarded the normalisation step. Hyper-parameter tuning similar to the one performed on the Decision Tree was used to find the ideal number of neighbours to use (7).

Linear Regression During the literature review it was discovered that despite not being commonly used in NLP, OLS regressions can provide decent results in the classification of text. Sinthupuan (2020) found that a multinomial logistic regression was able to predict disease classes based on short text descriptions from doctors. Given that we have a single class to predict (Disaster or Not), it was decided to use a simple logit model, where the label was assigned based on the predicted probability. Sinthupuan (2020) also recommends the use of Stochastic Average Gradient solver for dealing with sparse datasets.

Voting The Voting algorithm is the last of the statistical methods evaluated in this study. It works by taking the predictions generated by a set of classifiers, and taking the most frequent outcome as its own prediction. The models that formed this classifiers are those mentioned in this subsection, this is: Naive Bayes, Decision Tree, SVM, KNN and Linear Regression.

4.4 Neural Language Models

In the following paragraphs we describe the three Neural Language Models selected for this study. All of the models were built using the Keras and Tensorflow libraries on a Jupyter Notebook hosted on an AWS SageMaker instance. As suggested by Zhou et al. (2020), the ADAM optimiser was used on both the LSTM and Feed Forward Network. Given the BERT model required a larger computer, its code is provided in a separate Jupyter Notebook to the rest of the models.

Feed Forward Network This is the most basic type of network and takes as input the same frequency vector as the one used on the statistical models. Since the data has spatial relevance (like images or word sequences), the use of pooling and convolutional layers was discarded, in fact, the only type of layer used in this architecture were fully connected. Similar to the grid-search method, several combinations of number of layers and filter sizes were evaluated to find the best scoring model. The final design consists of an input layer, followed by two fully connected layers of 16 units and an active layer with a sigmoid activation. The fitting of the data was performed over 500 epochs, however, it was observed that 100 epochs was enough to reach convergence. This model was able to run on a computer with low capacity (4 GB RAM and 2 CPUs)

LSTM LSTMs are a special type of recurrent neural networks, this means that in contrast to feed-forward networks, the sequence in which words appear in the phrase is meaningful (Sundermeyer et al. 2015). Although the use of LSTM layers is widespread, the specific architecture of the network depends on the task and dataset being solved. Furthermore, Merity et al. (2017) mentions that in order to avoid the common over-fitting issue in LSTM, dropout layers should be introduced.

The architecture of the LSTM model in this study was derived by taking the suggested network layouts proposed by Gomez-Perez (2020) and Kamath (2019). The layer properties as well as the order and quantity of layers were modified in a stepwise fashion, looking for the model that yielded the best F1 score. The final architecture is as follows:

- Input Layer
- Dropout Layer (0.2 rate)
- 4xLSTM Layers (128 units)
- Dropout Layer (0.8 rate)
- Dense (256 units)
- Dense (128 units)
- Dense Output (softmax activation)

Similar to the simple Neural Network, it was found that 100 epochs were enough to reach convergence of the validation score. In terms of computing power the LSTM model required a medium sized machine to run (8 GB RAM 2 CPUs)

BERT BERT is a pre-trained neural network model created by Google Research and trained on Wikipeida and BookCorpus (Gomez-Perez 2020). Whilst LSTM uses parameters to pass information forward or backward on the phrase, BERT allows direct word interactions across the sentence (Zhou et al. 2020). Similar to LSTM, it uses word embeddings to transform words into directed vectors, however, this embeddings dictionary is provided as par of the BERT model and should not be modified (Turc et al. 2019).

Several versions of BERT trained on different types and sizes of copra are available from the Tensorflow Hub API, for this experiment, the original BERT-Large model was used. After downloading and compiling the model, only 4 epochs were necessary to achieve a satisfactory convergence of the scores. Due to its size, this model was run on a large SageMaker instance with 32 GB of RAM and 8 CPUs.

5 Results and Analysis

As described in the previous section, all models were trained and evaluated with the same subsets of data to ensure results are comparable. The final results are displayed in Table 2. Here we can observe that in line with other studies (Nazura & Muralidhara (2017), Phand & Phand (2017)) SVM is the best statistical model providing an F1 score close to 80%. However, despite its success in this experiment, this model only managed to outperform 60% of the submissions in Kaggle, showcasing the widespread ability to create accurate classifiers.

Disappointingly, the Naive Bayes method, which Wang & Manning (2012) regards amongst the best for NLP tasks, was not amongst the top 3 statistical models, and was ranked in the bottom 30^{th} percentile amongst the Kaggle community. However, this result is in line with Nazura & Muralidhara (2017), who reported that NB methods were susceptible to performance deterioration when the dataset is too small (number of data points) or too sparse (number of features). Billal et al. (2016) mention that in order to improve on the sparsity issue experienced on Twitter datasets, an aggressive stemming method and trimming the size of the input vector to include the most frequent words should be applied.

Table 2: Model comparison results

	Classifier	Accuracy	F1	Kaggle Percentile	Computing Effort
Statistical					
	Bernoulli NB	72.87%	77.61%	28	Low
	Multinomial NB	72.23%	76.89%	25	Low
	Decision Tree	69.60%	75.51%	21	Low
	SVM	73.03%	79.78%	61	Low
	KNN	54.03%	71.50%	15	Low
	Regression	74.25%	79.64%	56	Low
	Voting	74.17%	79.51%	53	Low
Neural Network					
	Feed-forward NN	72.28%	77.34%	27	Low
	LSTM	75.86%	80.69%	73	Medium
	BERT	88.94%	90.87%	98	High

Despite its lack of popularity, the regression model achieved a decent accuracy and ranked above average amongst other community submissions. In addition, the wide availability of this method amongst software packages and its low computational requirements make it an excellent choice for those looking for a simple and quick implementation.

Similar to the findings from Assery et al. (2019), the feed-forward neural network was outperformed by both statistical and neural models, however, LSTM was able to outperform all statistical methods. By comparing LSTM to SVM we can observe that despite the fact that the former has and F1 improvement of less than 1%, this equates to an improvement in Kaggle ranking of more than 10 points, suggesting that these type of high performing models are not common in the community. This may be due to the fact that LSTMs are difficult to design and require the use of large embedding dictionaries. Due to this, we conclude that LSTMs are appropriate only for users that have very specific problems and require marginal improvement over regular methods.

In line with findings from Fang et al. (2020) and Zhou et al. (2020), BERT provide a significant improvement over LSTMs, in this case over 10%. The 91% F1 score for BERT places it amongst the top 2% of submissions in the Kaggle competition. In addition to its accuracy, BERT was surprisingly easy to implement, requiring very little configuration or running However, the high computational requirements make this model suitable only for enterprises that can capitalise from the high running costs that this model will generate.

6 Conclusions

Summary In this paper, we compared 10 different statistical and neural language models in the task of detecting disasters in Twitter data. We concluded that both types of models are able to perform above 75% F1 score in this classification task. The best model, a pre-trained BERT neural network was able to achieve an accuracy of 91%, placing it in the top 2% of submissions in the Kaggle competition where the dataset was derived from. In addition, this study proposed a simple pre-processing pipeline and highlighted the most popular models used in the literature today.

A key conclusion is that although pre-trained models such as BERT and GPT2 are becoming increasingly available, the high computational requirements makes them difficult to be widely adopted. This is why there is still a place in the field of NLP for simpler statistical based models such as SVM, which we found to be able to beat the majority of submissions in Kaggle.

Another important conclusion was that NLP models are very sensitive to the pre-processing methodology. In particular, when dealing with sparse or small datasets, like those from Twitter, it is in the best interest of the data scientist to try to minimise the dimensions of the input vector.

Significance Overall, these results highlight that as pointed out by Zhou et al. (2020), current text classification technology has reached human level. This means that the technology is at a stage where it can begin to be implemented in areas such as emergency response, disaster management and research. Furthermore, its important to highlight that this study was carried out using only open source data, and aside from the computational requirements, all of the components to create an efficient tweet classifier are already available to the general public.

Limitations The main limitation of this study was that it was focused solely in the model selection and the pre-processing part of the study was left unoptimised. This means that there is a bias in the result for the methods that are less sensitive to bad input data. In particular, Naive Bayes was significantly affected and it is expected that a better pre-processing method would help those models improve their results.

Another limitation was the dimensions of data that were used. Because of convenience, the pre-processing part discarded information relating to the URL, the mentioned user, the geolocation and the significance of the hashtag.

It is also important to mention that the dataset was not derived directly from the Twitter API. In the description of the dataset it was highlighted that the distribution of targets was 50:50, which intuitively suggests the data has not been randomly selected. This emphasises the point that the classifier is only a piece in a larger application that should include some filtering criteria such as keyword search.

Further research This study would like to propose three areas where further research could be applied: pre-processing, data enriching and model selection.

Within pre-processing, an interesting question would be: "How do different pre-processing methods affect the most important NLP models?" A study in this area should cover the traditional steps such as POS, Stemming, Lemmatising but also other methods such as PCA dimensionality reduction and Named Entity Recognition. In addition, a comparative study of the different vectorisers would be insightful, particularly a comparison of the most popular ones, word2vec and GloVe.

As hinted by Diaz et al. (2020), future models don't need to be confined only to the analysis of one source or type of data. An interesting are of research on Twitter data would be on how to make use of image, video and location information. Going beyond the content of the tweet, crawling URLs and analysing the user profile have the potential of revealing more information that may help in task such as classification or information mining.

The final improvement to this study would be to expand the neural language models evaluated. As Zhou et al. (2020) mentions, there are many other pre-trained models avail-

able such as ELMo, GPT2, XLNet that would be useful to evaluate and compare. Given these models are already highly accurate, the aim of further research should be to find models and methods to reduce the computational requirements of these models in order to encourage the further adoption of NLP.

References

Abburu, S. & Golla, S. B. (2017), Ontology and nlp support for building disaster knowledge base, *in* '2017 2nd International Conference on Communication and Electronics Systems (ICCES)', Vol. 2018-, IEEE, pp. 98–103.

Altnel, B. & Ganiz, M. C. (2018), 'Semantic text classification: A survey of past and recent advances', *Information Processing and Management* **54**(6), 1129–1153.

Assery, N., Xiaohong, Y., Almalki, S., Kaushik, R. & Xiuli, Q. (2019), Comparing learning-based methods for identifying disaster-related tweets, IEEE, pp. 1829–1836.

Behzadan, V., Aguirre, C., Bose, A. & Hsu, W. (2018), Corpus and deep learning classifier for collection of cyber threat indicators in twitter stream, *in* '2018 IEEE International Conference on Big Data (Big Data)', pp. 5002–5007.

Beitzel, S. M. (2006), *On understanding and classifying web queries*, Citeseer.

Billal, B., Fonseca, A. & Sadat, F. (2016), Efficient natural language pre-processing for analyzing large data sets, *in* '2016 IEEE International Conference on Big Data (Big Data)', IEEE, pp. 3864–3871.

Boros, T., Dumitrescu, S. D. & Pipa, S. (2017), Fast and accurate decision trees for natural language processing tasks, *in* 'Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017', INCOMA Ltd., Varna, Bulgaria, pp. 103–110.

Diaz, J., St Denis, L., Joseph, M. B., Solvik, K. & Balch, J. K. (2020), Classifying twitter users for disaster response: A highly multimodal or simple approach?, *in* 'Proceedings of the Information Systems for Crisis Response and Management Conference (ISCRAM 2020)'.

Fang, W., Luo, H., Xu, S., Love, P. E., Lu, Z. & Ye, C. (2020), 'Automated text classification of near-misses from safety reports: An improved deep learning approach', *Advanced engineering informatics* **44**.

Friedman, J., Hastie, T. & Tibshirani, R. (2001), *The elements of statistical learning*, Vol. 1, Springer series in statistics New York.

Gomez-Perez, J. M. (2020), 'A practical guide to hybrid natural language processing combining neural models and knowledge graphs for nlp'.

Green, N., Breimyer, P., Kumar, V. & Samatova, N. (2009), Webbanc: Building semantically-rich annotated corpora from web user annotations of minority languages, *in* 'Proceedings of the 17th Nordic Conference of Computational Linguistics (NODALIDA 2009)', pp. 48–56.

Hand, D. & Christen, P. (2018), 'A note on using the f-measure for evaluating record linkage algorithms', *Statistics and Computing* **28**(3), 539–547.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Kamath, U. (2019), 'Deep learning for nlp and speech recognition'.

Kanakaraj, M. & Guddeti, R. M. R. (2015), Nlp based sentiment analysis on twitter data using ensemble classifiers, *in* '2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)', pp. 1–5.

Manning, C., Raghavan, P. & Schütze, H. (2010), 'Introduction to information retrieval', *Natural Language Engineering* **16**(1), 100–103.

Martinez, A. R. (2010), 'Natural language processing', *Wiley Interdisciplinary Reviews: Computational Statistics* **2**(3), 352–357.

Merity, S., Keskar, N. S. & Socher, R. (2017), 'Regularizing and optimizing lstm language models', *arXiv preprint arXiv:1708.02182* .

Nazura, J. & Muralidhara, B. L. (2017), Semantic classification of tweets: A contextual knowledge based approach for tweet classification, IEEE, pp. 1–6.

Pennington, J., Socher, R. & Manning, C. (2014), Glove: Global vectors for word representation, Vol. 14, pp. 1532–1543.

Phand, S. A. & Phand, J. A. (2017), Twitter sentiment classification using stanford nlp, *in* '2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)', pp. 1–5.

Rostami, M. & Mumivand, H. (2014), 'Automatic text classification using machine learning algorithm of k-nearest neighbor (k-nn) and n-gram indexing', *Majlesi Journal of Multimedia Processing* **3**(2), 7–12.

Sinthupuan, S. (2020), Disease classification with lstm and logistic regression models using pos tagging categories, *in* '2020 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)', IEEE, pp. 101–105.

Sit, M. A., Koylu, C. & Demir, I. (2019), 'Identifying disaster-related tweets and their semantic, spatial and temporal context using deep learning, natural language processing and spatial analysis: a case study of hurricane irma', *International Journal of Digital Earth* **12**(11), 1205–1229.
URL: *https://doi.org/10.1080/17538947.2018.1563219*

Sundermeyer, M., Ney, H. & Schlter, R. (2015), 'From feedforward to recurrent lstm neural networks for language modeling', *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* **23**(3), 517–529.

To, H., Agrawal, S., Kim, S. H. & Shahabi, C. (2017), On identifying disaster-related tweets: Matching-based or learning-based?, *in* '2017 IEEE Third International Conference on Multimedia Big Data (BigMM)', pp. 330–337.

Trstenjak, B., Mikac, S. & Donko, D. (2014), 'Knn with tf-idf based framework for text categorization', *Procedia Engineering* **69**, 1356 – 1364. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013.
URL: *http://www.sciencedirect.com/science/article/pii/S1877705814003750*

Turc, I., Chang, M.-W., Lee, K. & Toutanova, K. (2019), 'Well-read students learn better: On the importance of pre-training compact models', *arXiv preprint arXiv:1908.08962* .

Wang, S. & Manning, C. (2012), Baselines and bigrams: Simple, good sentiment and topic classification, *in* 'Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)', Association for Computational Linguistics, Jeju Island, Korea, pp. 90–94.
URL: *https://www.aclweb.org/anthology/P12-2018*

Zhang, Z., Zhang, Z., Chen, H. & Zhang, Z. (2019), 'A joint learning framework with bert for spoken language understanding', *IEEE Access* **7**, 168849–168858.

Zhou, M., Duan, N., Liu, S. & Shum, H.-Y. (2020), 'Progress in neural nlp: Modeling, learning, and reasoning', *Engineering* **6**(3), 275–290.