Natural Language Processing for Disaster Tweets Classification

Subham Gupta[1], Aby Kuruvila[2], Ajay Pallivathukkal Manoj[3]

1 Masters of Applied Computing, University of Windsor, 110093436, gupta2d@uwindsor.ca
2 Masters of Applied Computing, University of Windsor, 110089795, kuruvila@uwindsor.ca
3 Masters of Applied Computing, University of Windsor, 110089283, palliva@uwindsor.ca

***Abstract*—** A system for predicting tweets about real disasters is presented in this article. A new model will be developed and compared with the existing baseline model. Social media has become increasingly popular among netizens these days. Social media text can be categorized and used to predict disasters. This article describes how we classify tweets containing keywords related to disaster, using classification models.

Keywords- NLP, disaster, classification, twitter, SVM, naïve Bayes, LSTM, GloVe.

I. Introduction

The Twitter platform is a microblogging service. Tweets are short messages that users post and interact with. Users post about all kinds of topics on Twitter. Almost 500 million tweets are posted every day on Twitter, according to a recent survey. As a result, it has a large amount of uncategorized data and hence it is possible to find valuable information from this large data set.

This article classifies real or fake tweets about disasters. We used the Kaggle twitter dataset to analyze the raw text and we started with the pre-processing step. In order to apply it to classification models, it must be pre-processed. In pre-processing tokenization, normalization, stemming, lemmatization, stop words, and noise removal are done. The data was then subjected to feature extraction. The features were extracted using TF-IDF. Once we had the pre-processed data we trained the Multinomial Naïve Bayes and Support Vector Machine algorithm on our data set.

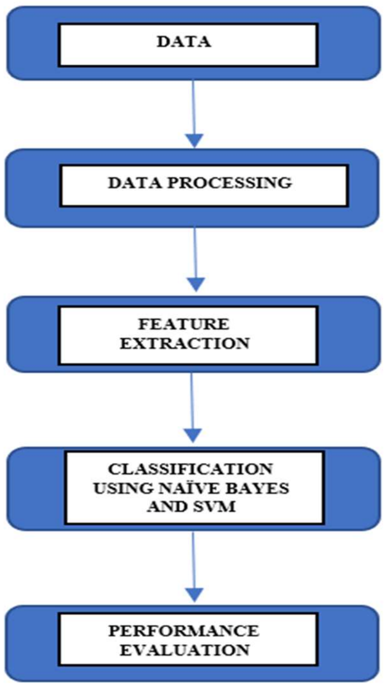


Figure 1: Pipeline for training baseline model

II. Dataset

Kaggle was used to collect disaster-related tweets. The number of tweets is 7613 and the number of columns is 5. Identifier, keyword, location, text, and target are the 5 columns. There are 7521 tweets left after removing duplicates. There are 4315 incidents unrelated to disasters and 3206 incidents related to disasters.

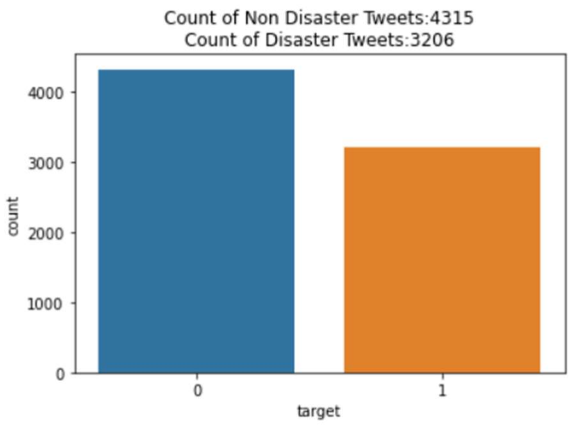


Figure 2: Count of disaster and non disaster based tweets.

III. Text pre-processing

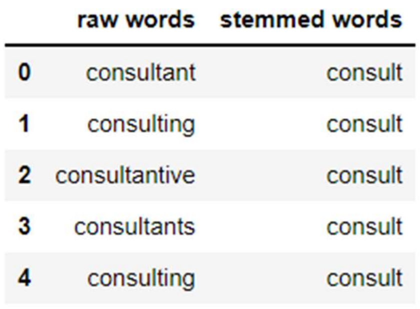
Our training set is pre-processed before we use it directly. Whenever possible, we try to normalize datasetse. To begin the normalization process, we used the tokenization technique to break up the sentences into smaller chunks. We also converted all text to lowercase, removed any stop words from the English language that had no discernible meaning, and reduced the amount of punctuation that was present throughout all texts. Internet-related phrases like "http://, @, #, www, com," which don't provide any information about the text's substance, have been removed. As we reasoned that these parameters wouldn't be of much use to us in the modeling process, we deleted the location and keywords parameters that had excessive amounts of NaN and inconsistent values. Using the stemming technique, we reduced the words to their fundamental roots, stopping the words with suffixes from repeatedly being processed as a separate term. We aimed to boost accuracy by condensing many words with the same meaning into a single word using the lemmatization technique.

A. Tokenization

The process of tokenization involves breaking up large text into smaller pieces.

B. Stemming

In order to reduce the possibility of overprocessing for words with the same tenses, the stemming process involves replacing all of a word's tenses with their base.

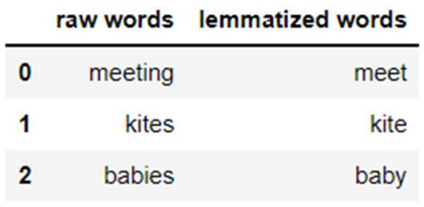


	raw words	stemmed words
0	consultant	consult
1	consulting	consult
2	consultantive	consult
3	consultants	consult
4	consulting	consult

Figure 4: Stemming

C. Lemmatization

Lemmatization is a technique used to combine similar-sounding words into a single word. Additionally, it completes the process of changing the words' endings or restoring them to their original forms, such as came to come.



	raw words	lemmatized words
0	meeting	meet
1	kites	kite
2	babies	baby

Figure 5: Lemmatization

D. Stop Word Removal

In English, stop words like am and is are the most commonly used words. Our classification model will be more accurate if these words are removed.

IV. Feature extraction

A. TF-IDF

A TF-IDF is an acronym for Term Frequency - Inverse Document Frequency. In text mining, it is widely used to illustrate the importance of a term to a document. The term is weighted based on how frequently it is used. The importance of the terms is ranked to produce healthier results. Additionally, the frequency of use of these phrases in various texts is examined. If it appears frequently, the term's weight suffers. On the other hand, if it seems too little, weight is raised. The following describes the TF-IDF's mathematical foundation. In the presence of a document collection D, a word w, and a single document d ϵ D, we determine:

wd = fw, d ∗ log (|D|/fw, D)

where |D| is the size of the corpus, fw,d is the number of times w appears in d, and fw,d is the number of documents where w appears in d.
(Salton & Buckley, 1988, Berger, et al, 2000).

V. BASELINE CLASSIFICATION MODEL

SVM and Nave Bias are two classification algorithms we used.

A. *Naïve Bayes:*

Naive Bayes, a well-known and frequently applied model for unstructured text, served as our foundation. It operates effectively with little data.

The method is criticised for being "naive" since it assumes two things about the world around us: first, that each variable is statistically independent of the others, and second, that each feature is of equal weight. The underlying idea of Bayesian decision making and conditional proximities allow us to infer the likelihood of a specific event despite the fact that these two conditions are extremely uncommon in the real world.

With our data ready, we can invoke an SKLearn class to create a Naive Bayes classifier while initially using default parameters to build a baseline performance. We will use this baseline performance to determine whether to continue exploring the Naive Bayes parameter space or to take another model for our situation into consideration.

B. *Support Vector Machines:*

SVM is our next consideration for tweet classification. It is well known that Support Vector Machines are effective at binary categorization of unstructured data. In order to serve as our decision boundary for incoming, unobserved data, the model seeks to provide a separating hyperplane between the classes in the data.

We want to maximise the distance between the support vectors (for each class) and the separating hyperplane, which are the data points that are on the threshold of this hyperplane. The dot product of our weights and the instance inputs is fed to a straightforward activation function (like sigmoid) by the algorithm, which outputs a class label.

Support Vector Machines determines the ideal separation hyperplane by selecting the appropriate weights and slope factor for our data.

Using default parameters, we have fitted a support vector machine model.

VI. RESULT: The classifier's accuracy has been calculated. The results are displayed in the table below.

Table 1: Accuracy table

Baseline Model Accuracy on Disaster Tweet Dataset		
MODEL	Naïve Bayes	SVM
ACCURACY	78.19%	78.98%

VII. ADVANCED CLASSIFICATION MODEL

A. *LSTM Neural Network*

RNNs are improved by Long-Short Term Memory Neural Networks. Sepp Hochreiter and Jurgen Schmidhuber introduced it in 1995. In order to solve the problem of long-term dependency, LSTM was developed. In LSTM, the flow of information is controlled by sigmoid functions. The basic model began with a single layer of LSTM. We used the vectorized text from the training dataset as our input. E1 Layer embeds vectorized text. Input vectors are run through LSTM1 gates to produce 64 feature maps. In addition, these feature maps are passed to D1 for generation of 256x1 feature maps. To reduce noise and overfitting, D2 uses a dropout value of 0.2. The D3 output layer takes 256 attributes as input and outputs whether the text is about a true disaster or not. In terms of training accuracy and validation accuracy, this model showed 96.99% and 76.63% accuracy, respectively. Model has training loss of 6.86% and validation loss of 140.47%. when tested with the test data model gave the accuracy of 77.65% and loss of 122.66%.

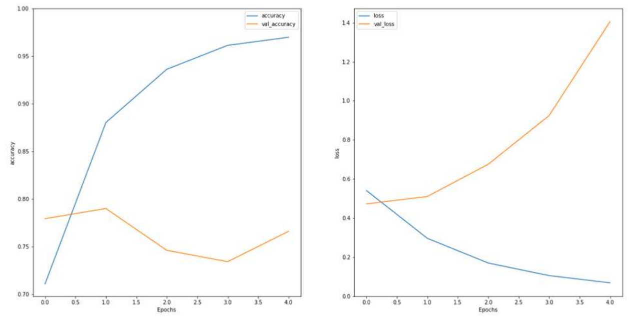


Figure 6: Training Accuracy and Validation Loss

Bidirectional LSTM was implemented due to the higher loss of the model. Bidirectional model had higher validation loss and test loss of 114.47% and 103.19%, respectively and training and validation accuracy of 97.19% and 78.19%

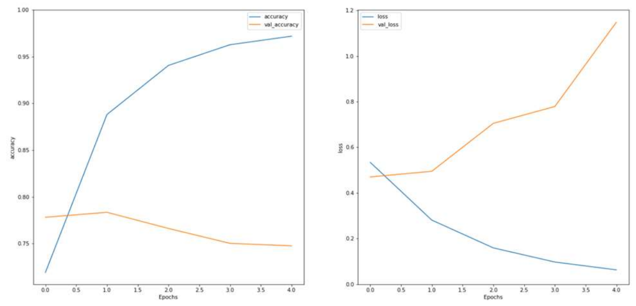


Figure 7: Training Accuracy and Validation Loss

B. *LSTM with Attention*

Since the LSTM neural network had to condense all the data into a fixed length vector, it was designed to address the long-term dependency problem. Long sentences become challenging for the neural network as a result. Badhdanau proposed an attention mechanism [[1409.0473] Neural Machine Translation by Jointly Learning to Align and Translate (arxiv.org)] in 2015. All the hidden states can be seen with the help of the attention mechanism. Although a new attention layer has been introduced, the LSTM architecture is still the same. In order to converge the graph after the LSTM1 layer, the Attention layer was added. This model provides a train accuracy of 97.24% and validation accuracy of 74.77%

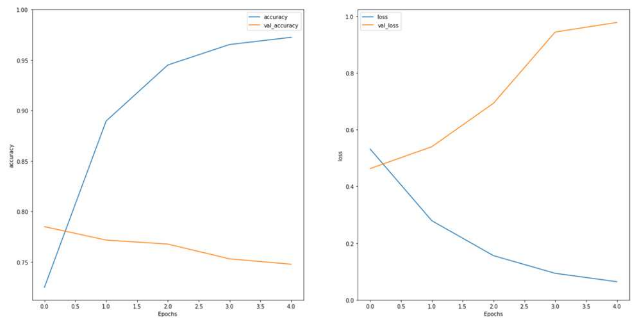


Figure 8: Training Accuracy and Validation Loss

C. *LSTM with GloVe*

Due to the non-converging graph, we used a GloVe model that had been pre-trained. The GloVe algorithm focuses on the co-occurrence of words across the corpus as a whole. Based on its embedding, two words were more likely to appear together than on their own. There are 400,000 words in the GloVe Model we used. We converted the word vectors to an embedding matrix and inserted zeros for every word we missed. A total of 9576 words were converted, and 3043 were missed. We just changed the input data of the neural network in the previous model. With the help of glove models, we vectorize text.

This model produced some respectable outcomes. Model's training accuracy was 83.11% and loss was 37.91%, validation loss was 45.29% and accuracy was 81.52% compared to previous model this model has better validation curve.

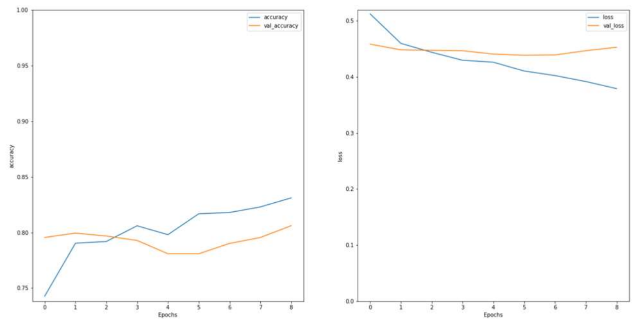


Figure 9: Training Accuarcy and Validation Lloss

Also, the results of the test data were superior to the previous models that were implemented. Test accuracy was around 81.51% and loss was 42.78% which is a significant improvement when compared with LSTM and LSTM with Attention.

VIII. CONCLUSION

The best learning algorithm is undoubtedly more advantageous for the deep learning models. An overview of model learning and observation can be found in the following table.

Table 2: Accuracy table

Deep Learning Model Accuracy on Disaster Tweet Dataset				
MODEL	**LSTM Model**	**Bi-LSTM Model**	**LSTM with Attention**	**LSTM with Glove + Attention**
TRAIN ACCURACY	96.99%	97.19%	97.24%	83.11%
VAL ACCURACY	76.63%	74.77%	74.77%	80.61%
TEST ACCURACY	77.65%	78.19%	78.05%	81.51%
OBSERVATION	Overfitting	Overfitting	Overfitting	Properly Fitted

We note that the deep learning model was highly susceptible to overfitting. As the word embeddings enhance their semantic representations through the use of several word2vec pretrained models, we also noticed a significant improvement in the model test accuracy.