NLP with Disaster Tweets

Yogita Mallikarjun Pattan
Master of Computer Science
University of California, Irvine
pattany@uci.edu

***Abstract*— Twitter has become a crucial communication platform during emergencies, allowing individuals to report and share information in real time. In this paper, we explore Natural Language Processing (NLP) approaches for predicting whether a tweet is related to a real disaster or not. We explore a range of NLP techniques, including feature engineering, word embeddings, and machine learning algorithms such as Support Vector Machines (SVM), Random Forests, and LSTM Neural Networks. We evaluate the performance of our approach using accuracy metrics on a dataset of 7600 hand-classified tweets. We also analyze the most informative features for the classification task and provide insights into the types of words and phrases that are most indicative of disaster-related tweets.**

Keywords*— SVM, Random Forests, GloVe Word Embedding, Stopwords, ngrams, TF-IDF, LSTM.

I. Introduction

Natural language processing (NLP) is an area of study that focuses on enabling computers to understand, interpret, and generate human language. One of the most popular applications of NLP is sentiment analysis, which involves analyzing text data to determine the emotional tone behind it. In recent years, the use of NLP techniques for sentiment analysis has become increasingly important in various domains, including social media, politics, and business.

In this paper, we focus on the application of NLP techniques for the analysis of Twitter data in the context of disaster response. During natural disasters, Twitter is often used as a primary means of communication to share information and updates. Our goal is to explore various models and their performance on how they can accurately identify tweets related to disasters and those that are not, as well as classify them based on their sentiment (positive or negative).

II. Dataset

The dataset that is being used is provided by the company figure-eight. It consists of 7600 hand-classified tweets. It includes an id unique to each tweet, the text of the tweet, the keyword and location of the tweet, and a target specifying if is a disaster tweet or non-disaster tweet.

III. Data Exploration

In this section, the dataset was explored to gain insights and identify any potential issues that might affect the analysis.

A. Columns

The columns in the dataset were examined and checked for any null values. The dataset had three columns: ID, keyword, location, text, and target. The ID, text, and target columns did not have any null values, whereas the location had around 2500 null values and the keyword had around 60 null values. These two columns did not add any significant value to the data as location and keyword values were similar in both disaster and non-disaster tweets. The text column was then examined to identify any potential issues such as non-alphanumeric characters. It was found that there were some non-alphanumeric characters such as hashtags, mentions, and URLs in the tweets, which might require some cleaning before further analysis.

B. Class Distribution

Next, the dataset was checked if it was balanced or unbalanced based on the count of tweets for each class by plotting the class distribution (Fig 1). It was found that the dataset was almost balanced, with 57% of the tweets labeled as non-disaster tweets and 43% of the tweets labeled as disaster tweets. This was done to check if it may impact the performance of the model and it was noted that it will need to be accounted for during modeling.

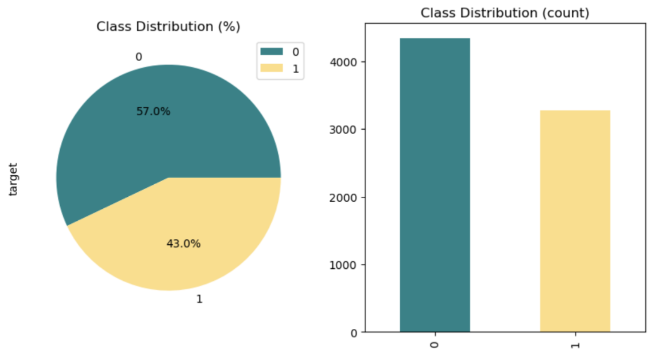


Fig. 1. *Class Distribution*

C. Number of words

The number of words in each tweet was calculated and compared if they were disaster-related or not. From the data, it looked like disaster tweets were no longer than non-disaster tweets. It was found that the average length of a tweet in the dataset was 15 words. The distribution of the number of words in the tweets was also observed in the plotted histogram (Fig 2).

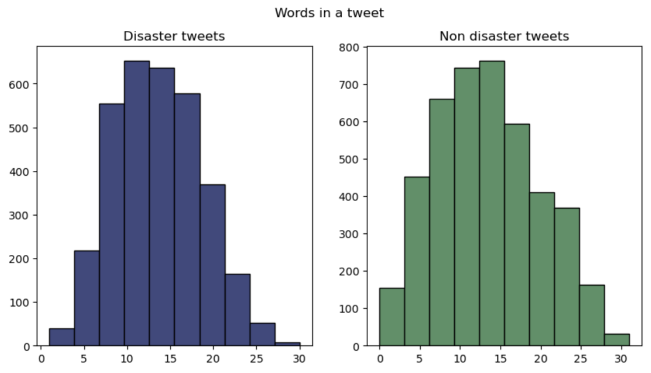


Fig. 2. *Number of Words In Tweets*

D. Most frequent words:

The most frequent words in both categories of tweets were identified. These could be seen in the plot below (Fig 3). These observations suggested that disaster tweets had words like fire, killed, and storm indicating disasters, while non-disaster tweets contained positive and neutral language.

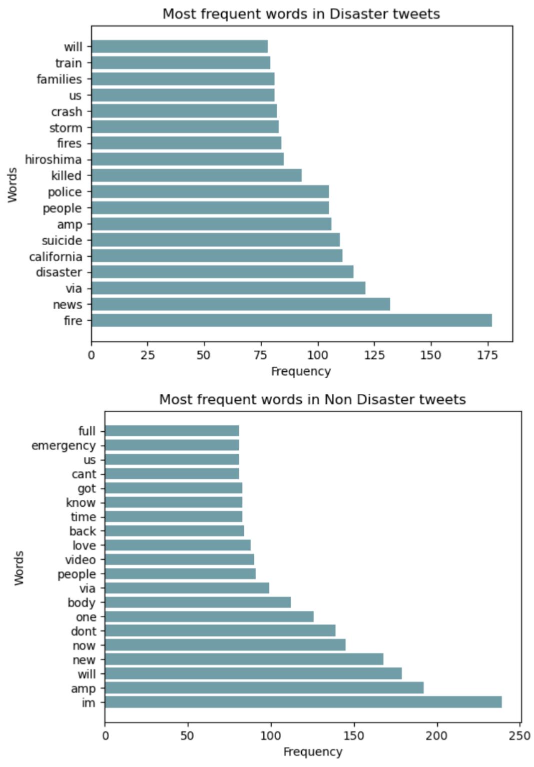


Fig. 3. *Most Frequent Words*

E. *Most common trigrams:*

The most frequent trigrams in both categories of tweets were analyzed. In the disaster category, the most frequent trigrams were "northern california wildfire", "more homes razed", "suicide bomber who", and "families affected by". In the non-disaster category, the most frequent trigrams were "liked youtube video", "reddit will now", "cross body bag", and "now quarantine offensive". These insights helped in understanding that contextual information would help in better classification of the tweets.

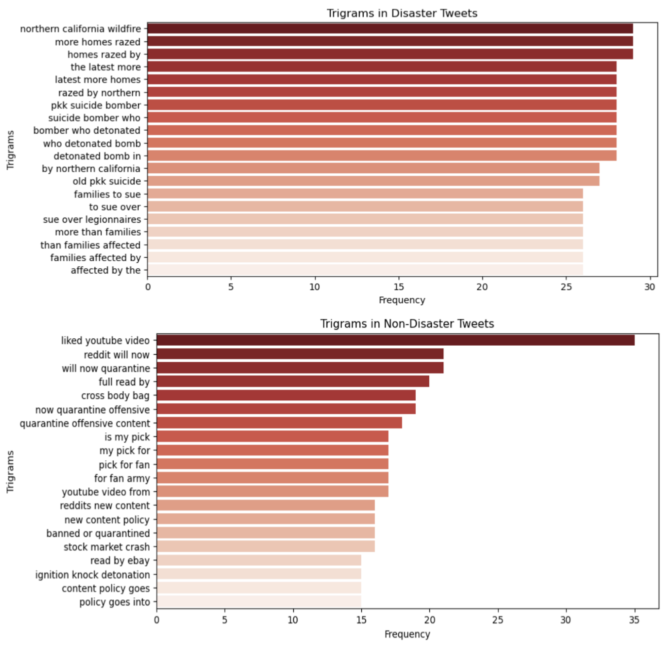


Fig. 4. *Most Common Trigrams*

F. *Word clouds:*

Finally, the most frequent words were visualized using word clouds (Fig 5). This visualization reinforced the earlier observation that disaster tweets tended to contain more words related to the disaster, while non-disaster tweets tended to contain positive and neutral language.

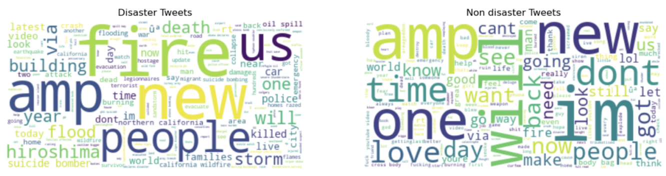


Fig. 5. *Word Clouds*

IV. Data Preprocessing and Feature Design

In this section, the data preprocessing steps taken to clean the data and prepare it for modeling are described.

- Only the text column was selected, as it contains the most relevant information for the classification of tweets. The other columns - keyword and location - were dropped as they do not contribute significantly.
- Any non-alphanumeric characters and punctuations were removed from the tweets to avoid any bias in the analysis.
- Any URLs in the tweets were also removed as they do not provide any useful information for modeling.

Once the noisy text was removed, there was a drop in the vocabulary size which indicated the amount of noise the data contained.

Stopwords such as "the", "is", and "and" were removed as they were deemed to not add significant value to the analysis. However, during a comparison of the model's performance before and after stopword removal, it was observed that the model performed better when the stopwords were not removed. As a result, this preprocessing step was later discarded as it was found to remove contextual information from the tweets.

The preprocessing performed on data required for specific models is described below.

Preprocessing for SVM and Random Forest

- TF-IDF Vectorizer was used to weigh the words in the tweet based on their importance in the corpus.
- TF-IDF stands for Term Frequency-Inverse Document Frequency. It is a numerical statistic that reflects the importance of a word in a document, relative to its frequency in the entire corpus of documents. The term frequency (TF) measures the number of times a word appears in a document, while the inverse document frequency (IDF) measures how much information the word provides, i.e., whether the word is common or rare across all documents in the corpus.

Overall, these preprocessing and feature design steps help create a clean and meaningful dataset that can be used for the classification of tweets into disaster and non-disaster categories.

Preprocessing for LSTM-based Neural Networks

- After cleaning the data, the text was tokenized using the Word Tokenizer.
- Padding was added to ensure that all the tweets were of the same length.
- The GloVe word vectors were then used to represent each word in the tweets as a vector of numerical values.
- GloVe (Global Vectors for Word Representation) is a type of word embedding technique that maps words to high-dimensional vectors of real numbers, with the goal of capturing their semantic relationships. GloVe word vectors are pre-trained using a large corpus of text. GloVe is one of the most widely used techniques and has been shown to perform well on a variety of NLP tasks.

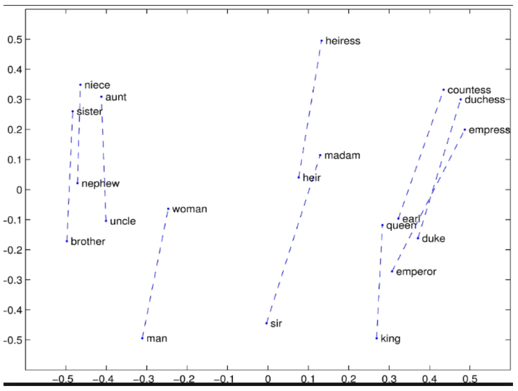


Fig. 6. *Example of GloVe Word Vector Representation in 2D*

- By tokenizing the text data, each tweet was converted into a sequence of integers, with each integer representing a unique word in the tweet. The resulting feature matrix was then used as input to the classification model.

Overall, this preprocessing and feature design step allowed the text data to be prepared for use in a machine learning model, by transforming the raw text into a format that can be understood by the model. By using Glove word vectors, the meaning of each word in the tweets was captured, which is an important factor in distinguishing between disaster and non-disaster tweets.

V. Model Exploration And Performance Validation

In this section, three models, namely SVM, Random Forest Classifier, and LSTM Neural Network are explored to predict if a tweet is disaster-related or not.

A. *Support Vector Machine (SVM)*

A Support Vector Machine (SVM) model was explored, which is a popular algorithm for classification tasks. The hyperparameters of the SVM model, namely the kernel function and the regularization parameter (C), were experimented with for tuning. To find the best combination of hyperparameters, GridSearchCV was used. A parameter grid was defined containing different values for the kernel function ('linear', 'poly', and 'rbf') and the regularization parameter C (0.1, 1, 10, 100).

GridSearchCV exhaustively searches over a specified parameter grid to find the best combination of hyperparameters that optimize the model's performance. GridSearchCV performs cross-validation by iterating over all possible combinations of hyperparameters and training a model on each combination of k-folds to ensure that the hyperparameters generalize well to unseen data and prevent overfitting.

It was observed that the ‘linear’ kernel function performed the best, with the best value of C being 1. The SVM model achieved a maximum validation accuracy of 81.61%. From the plot (Fig 7), it can also be observed that ‘rbf’ kernel function provides good accuracy for C values 1, 10, and 100 whereas, for the ‘linear’ kernel, the accuracy drops for higher values of C.

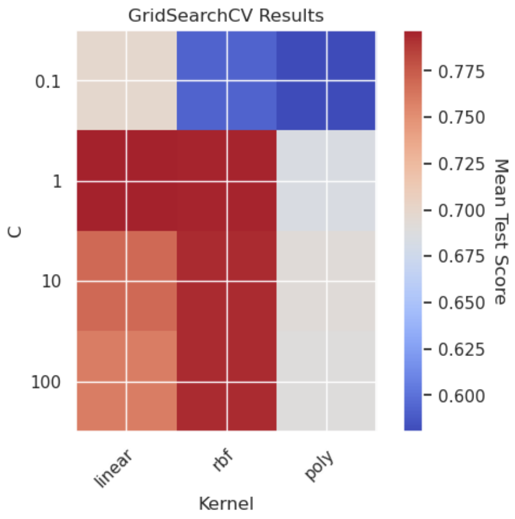


Fig. 7. *SVM GridSearchCV Results for hyperparameters Kernel and C*

B. *Random Forest Classifier*

Random Forest Classifier works by constructing multiple decision trees and combining them into a single model to improve accuracy and reduce the risk of overfitting.

While experimenting, the number of decision trees was varied from 10 to 100, and two different split criteria, gini, and entropy were tested. For each combination of n_estimators and criterion, the model was fitted on the training set and its accuracy was tested on the validation set. It was found that the performance of the model generally improves with increasing the number of decision trees, but the improvement began to plateau after a certain number of trees. It was observed that using gini as the split criterion performed slightly better than using entropy, with the maximum validation accuracy achieved at 100 decision trees and gini criterion. The best validation accuracy achieved by Random Forest Classifier was 79%.

Furthermore, the max_depth parameter value was also modified from 10 to 80, but it was observed that limiting the max depth did not lead to good performance. The Random Forest Classifier worked best on training and validation data when the max_depth was set to none.

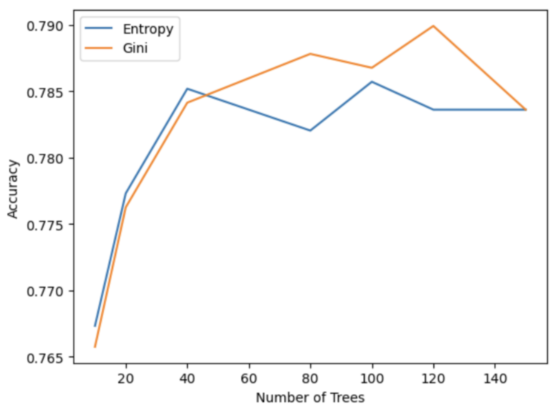


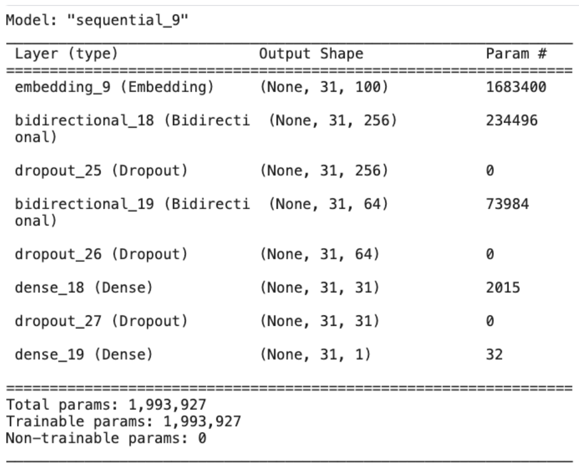
Fig. 8. *Accuracies for different no. of trees and criterion*

C. *LSTM-Based Neural Network*

Long Short-Term Memory (LSTM), based neural networks have gained popularity in recent years due to their ability to capture long-term dependencies in sequential data.

For the disaster tweet classification task, an LSTM-based model was explored. Pre-trained GloVe word vectors are taken as input by the model and it consists of two layers of LSTM units. Initially, only when LSTM layers were used, they did not capture the contextual information as much and the model was underfitting. It was then replaced with Bidirectional LSTM layers. One unique aspect of Bidirectional LSTMs is that it processes the input sequence both forward and backward, allowing the network to capture contextual information from both directions. This is particularly useful for identifying the type of tweet where the context of a word heavily depends on the surrounding words.

A dense layer with rectified linear activation was also added to map the output of the LSTM layers to a one-dimensional vector, which is then fed into the final output layer with a sigmoid activation to produce the binary classification output. The model was compiled using the RMSprop optimizer and binary cross-entropy loss function.



```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_9 (Embedding)     (None, 31, 100)           1683400

 bidirectional_18 (Bidirecti  (None, 31, 256)          234496
 onal)

 dropout_25 (Dropout)        (None, 31, 256)           0

 bidirectional_19 (Bidirecti  (None, 31, 64)           73984
 onal)

 dropout_26 (Dropout)        (None, 31, 64)            0

 dense_18 (Dense)            (None, 31, 31)            2015

 dropout_27 (Dropout)        (None, 31, 31)            0

 dense_19 (Dense)            (None, 31, 1)             32

=================================================================
Total params: 1,993,927
Trainable params: 1,993,927
Non-trainable params: 0
_________________________________________________________________
```

Fig. 9. *Model Summary*

Overfitting of Model

During the training process, it was observed that the model started to overfit as the validation loss began to increase drastically. Even though the training accuracy increased drastically, the validation accuracy continued to remain the same.

To prevent overfitting, dropout layers were included between the LSTM layers, which randomly drop out a fraction of the LSTM units during training. On adding the dropout layers, the overall training and validation accuracy increased.

Due to the smaller size of the dataset, the model started to overfit even after the inclusion of dropout layers. The model was then trained for 8 epochs with a batch size of 32, and the loss and accuracy graphs were plotted (Fig 10). From the curves, it was observed that the highest accuracy was obtained at the 5th epoch, and this epoch size was chosen for the final model training. Despite the overfitting issue, a maximum validation accuracy of 81.7% was achieved by the model.

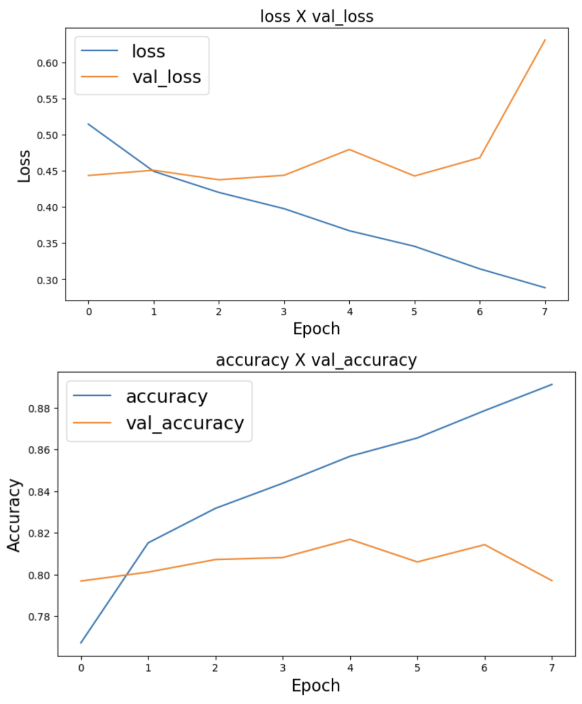


Fig. 10. *Loss and accuracy plots for training and validation data*

VI. Conclusion

In this study, different machine learning algorithms like Support Vector Machines and Random Forest Classifiers were explored for disaster tweet classification. Finally, an LSTM-based neural network was explored to capture long-term dependencies in sequential data. The performance results are summarised in Table 1.

Model	Accuracy (%)
Random Forests	79
SVM	81.61
LSTM Neural Network	81.70

Table 1: A summary of the results of explored model

Despite the overfitting issue in the LSTM model, it achieved competitive performance. In conclusion, our study demonstrated that machine learning algorithms can be effective for disaster tweet classification, and the

LSTM-based model with bidirectional LSTM layers shows promising results. Further research could explore the use of pre-training techniques, ensembles, and other advanced neural network architectures to improve performance.

VII. Acknowledgement

This study was conducted by me alone, who was solely responsible for the data exploration and visualization, data preprocessing, feature design, model exploration and analysis, and interpretation of the results.

I would like to acknowledge the following sources for their contributions to the visualizations in our project:

Source 1: I referred to the code in this source to create count and frequency graphs. The code was modified to fit our data and analysis needs.
Source 2: The word cloud plot in Figure 5 was created using the code available in this source. I adapted the code to include our own data points and labels. This code also helped me understand how to approach an NLP problem statement.

I would like to thank the authors of these sources for making their code publicly available, which helped us create effective visualizations for our project.