# Natural Language Processing with Disaster Tweets

Akshay Jain
10087792
akshayj2@uci.edu

Sahil Jain
16683795
sahilj2@uci.edu

Viraj Shah
79085971
shahvp@uci.edu

***Abstract*—Natural Language Processing, which is a sub field of Artificial Intelligence, helps us to understand verbal and written content. It comprises of various tasks such as Speech Recognition, Sentiment Analysis as well as Language Generation. It has applications in several use cases like chat-bots, machine translation, spam content detection and virtual assistants. Our project is based on sentiment analysis to extract meaningful subjective qualities like emotions and attitude from the provided data. Our objective is to identify if a specific tweet is a real disaster or not. This project is built for a Kaggle competition and utilizes a data set that has over 10,000 tweets that were hand classified. We tested a variety of methods, including TF-IDF features, Count Vectorization, nGrams and pre-trained ALBERT model to determine whether a tweet is related to a tragedy or not. We discovered that the ALBERT performed the best for this dataset, with an accuracy score of 81%**

***Keywords*—** Natural Language Processing , ALBERT , Disaster Tweets, Logistic Regression

## I. Introduction

With the growth in micro-blogging, an announcement which allows users to share short messages to the internet community, even in a crisis, social media delivers a plethora of data, including details on the nature of the event, the feelings of those impacted, and relief activities. Taking advantage of one micro-blogging website- Twitter, we propose a natural-disaster analysis in this project that simply relies on tweets made by Twitter users during natural disasters.

Users are mostly using Twitter, the most popular of micro-blogging platform, to send news, photographs, and views from the incident site throughout the world. Twitter is mainly unfiltered, widely available, and user centric and it is real-time data created by the user community. One problem to the above advantage just mentioned is that users across the world have a different way of communication and a different style to express their feelings towards the people involved in the disaster. Therefore it is tough to classify them as real or fake and perform sentiment analysis on it.

The major motivation behind this project is that once the information about people's emotions before, after and during a disaster is precisely understood, it will be a great tool for handling different aftermath of a disaster.

## II. HISTORY

Data from social media is regarded to be crucial for determining popular emotions and moods. Finding postings that express irritation, danger, or anxiety is crucial, especially when it comes to disaster management. As a result, the processing of such posts is aided by a systematic classification that is divided into feeling intensity and aspect-based categories, which will enable many institutions, including non-governmental organizations and governments, in managing such situations. For this purpose extensive research has been done in the field of NLP. Natural Language Processing (NLP) is a field of study that brings together artificial intelligence, computer science, and linguistics. Sentiment analysis using product review by Xing Fang and Justin Zhan[1] is a good example of how text data can be used to understand the response of people towards the product.

We will be building upon ideas discussed and already present in research work like, Identifying and Categorizing Disaster-Related Tweets by Kevin Stowe, Michael Paul, Martha Palmer, Leysia Palen, Ken Anderson[2]. Another closely related work is the flu classification system Lamb et al. (2013), which is classifying tweets for relevance and then the research work applied finer grained classifier.

## III. Analysis of Data

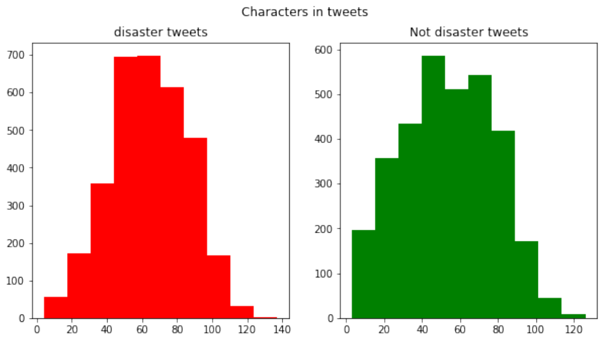
Number of researchers have used social media to either do sentiment analysis on comments for an e-commerce website, to detect fake news from Facebook posts or sentiment analysis on twitter tweets. Therefore there is a plethora of organised data available on the internet.

### A. About the Data

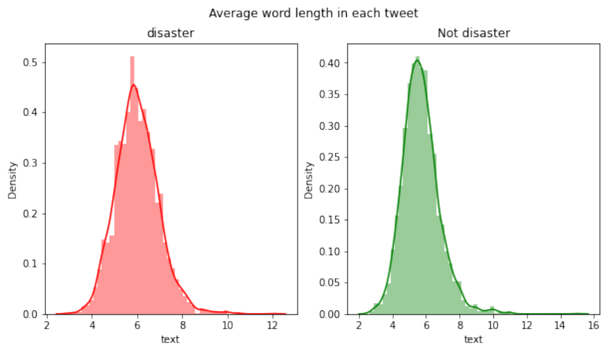
This data set was created by the company figure-eight and originally shared on their 'Data For Everyone' website. Each data sample in the train data has five columns (id, text, location, keyword, target). Out of these five columns we have text, which is the text of the tweet. Keyword has important keywords from that tweet and finally location column tells you the location from which this tweet was sent out from. We are predicting whether a given tweet is about a real disaster or not. If so, predict a 1. If not, predict a 0. Therefore in our target value we can see values (1,0) indicating real or fake tweet.

### B. Analytical Viewpoint of Data

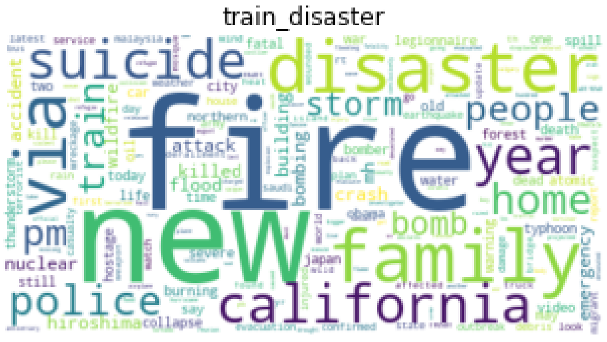
The train data that was provided to us contains 7613 rows of information along with five columns. This reduces to a 6542 rows after balancing the data by output label. To understand the number of characters in both categories of tweets we plot a graph of number of characters vs frequency. Our findings suggest that both type of tweets had 120-140 character per tweet.



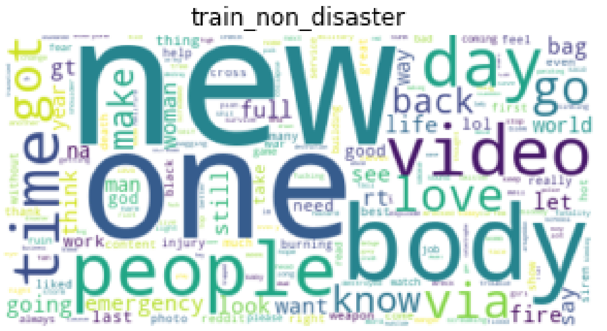
To understand the data further a plot of average word length in each tweet is plotted.



To analysis the frequency of words we plotted a word cloud for the train data in both disastrous and non disastrous category. A word cloud is a collection, of words depicted in different sizes. The bigger and bolder the word appears, the more often it's mentioned within a given text.



Word cloud for train data that are labelled as disaster tweets



Word cloud for train data that are labelled as non-disaster tweets

We can see that words like fire, disaster, police, suicide, killed have a higher frequency in disastrous tweets while words like people, love, body, day can be found in non disastrous tweets.

### C. Data Cleaning and Prepossessing

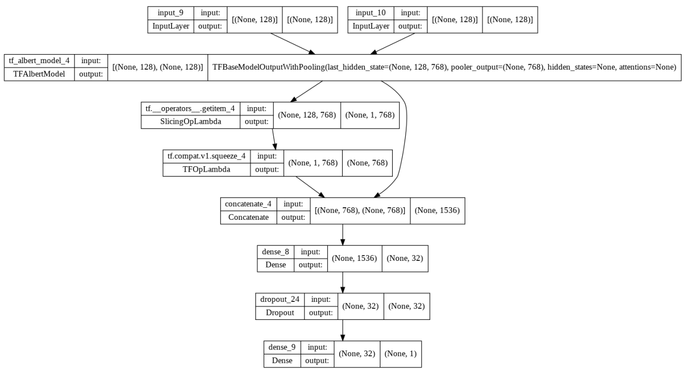
For processing of data and cleaning it to make it usable we have taken the following steps:

1) Remove any person or page tagged by the user using @ operator
2) Remove any URLs present in the tweet
3) Remove emoji added by the user
4) Remove @amp occurences
5) Remove all non-alpha characters
6) Remove all stopwords
7) Perform stemming and lemmatization
8) Covert text to lowercase

## IV. Model Selection

After prepossessing and analyzing of data we trained our data in various models to experiment and understand which model would be a better fit for the given dataset.

1) Count Vectorization: It is the process of counting how many times each word appears in a text. To do this, Python's Sci-kit learn package includes a utility called CountVectorizer. Setting this parameter can be incredibly useful when dealing with a large amount of document.
2) TF-IDF: It can be broken down into two parts TF (term frequency) and IDF (inverse document frequency). Term frequency works by looking at the frequency of a particular term and Inverse document frequency looks at how common (or uncommon) a word is amongst the corpus.
3) N-Grams: In case of n-grams the recurrence of a word is predicted using an N-gram model based on the occurrence of its N – 1 preceding words.
4) Logistic Regression: Logistic Regression is a special type of linear regression algorithm that is used for classification task. Here, the final layer uses a sigmoid activation and a binary cross-entropy loss. Any value more than 0.5 is classified as positive and others as negative.
5) ALBERT model: Transformers are a special category of deep neural networks specially useful for sequential data which is the case with natural language processing. It encodes each time-step (in out case, word) of the sequence. Now, each time-step is compared with every other time-step in the sequence for context using a mechanism known as "attention". BERT is one of the type of transformers, while ALBERT is a light-weight implementation of BERT with approximately 11M parameters. We will use a pre-trained version of ALBERT model as provided by HuggingFace library and append some custom fully-connected layers at the end of it.


input_9 | InputLayer | input: | output: | [(None, 128)] | [(None, 128)]
input_10 | InputLayer | input: | output: | [(None, 128)] | [(None, 128)]
tf_albert_model_4 | TFAlbertModel | input: | output: | [(None, 128), (None, 128)] | TFBaseModelOutputWithPooling(last_hidden_state=(None, 128, 768), pooler_output=(None, 768), hidden_states=None, attentions=None)
tf.__operators__.getitem_4 | SlicingOpLambda | input: | output: | (None, 128, 768) | (None, 1, 768)
tf.compat.v1.squeeze_4 | TFOpLambda | input: | output: | (None, 1, 768) | (None, 768)
concatenate_4 | Concatenate | input: | output: | [(None, 768), (None, 768)] | (None, 1536)
dense_8 | Dense | input: | output: | (None, 1536) | (None, 32)
dropout_24 | Dropout | input: | output: | (None, 32) | (None, 32)
dense_9 | Dense | input: | output: | (None, 32) | (None, 1)

## V. METHODOLOGY

### A. Feature Extraction

For regular machine learning model like SVMs, Logistic Regression, there are various ways of extracting features from text data like CountVectorization, TF-IDF and N-grams. We use each of these approaches to find out what works best for our data. In case for ALBERT transformer model the input sequence must be converted to a pair of ids and masks, the process of generating these features is very specific to the ALBERT model which is described in the respective research article[5]. So we use an implementation from HuggingFace library to generate these features.

### B. Parameter Selection

For the Logistic Regression model we used a default implementation provided by the Scikit library and train it on the data after performing feature extraction as described earlier. Since the

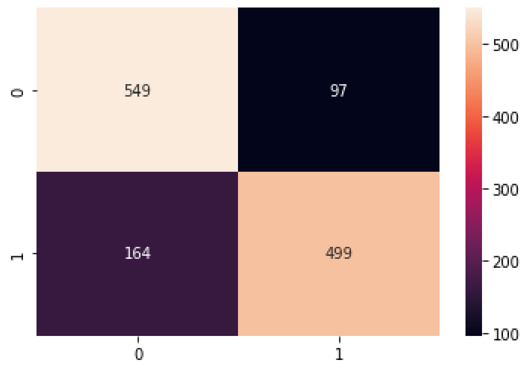
logistic regression model has a simple configuration and doesn't have to many hyper parameters that can be tweaked we use the default parameters. While training the model on Count Vectorization and TF-IDF we observed that tweets like, ”Flood” and ”No flood” produced same outputs. Therefore we moved to N-grams so as to get a better coherence on the tweets we used N-grams, where N was experimented for 2,3 and 4. Finally we settled for n=2 based on results.
For classification using ALBERT Model we use a pre-trained ALBERT transformer backbone provided by the huggingface librrary, we append fully-connected layers after this backbone to transform the output into desired shape, we also added a dropout layer to add regularization effect. We are using the Adam optimizer, we begin with an initial learning rate = 0.0002 (as optimally specified in[5]) We also use methods such as learning rate decay and early stopping to monitor the training process. This ensures that the model leverages transfer learning along with fined tuned fully connected layers and is trained efficiently using the aforementioned callbacks.

## VI. Results

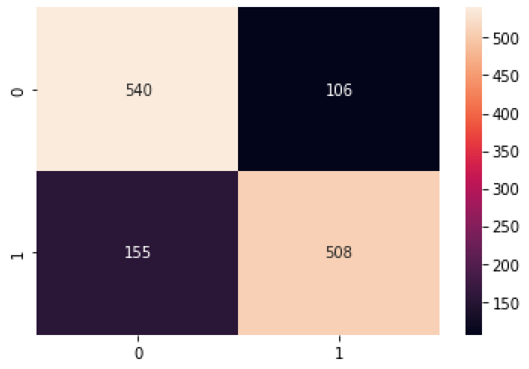
Following results are obtained by the above discussed methodology

| Method | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Count-Vectorization | 0.80 | 0.84 | 0.75 | 0.79 |
| TF-IDF | 0.80 | 0.83 | 0.77 | 0.80 |
| N-Grams | 0.79 | 0.82 | 0.75 | 0.79 |
| ALBERT | 0.82 | 0.85 | 0.78 | 0.81 |

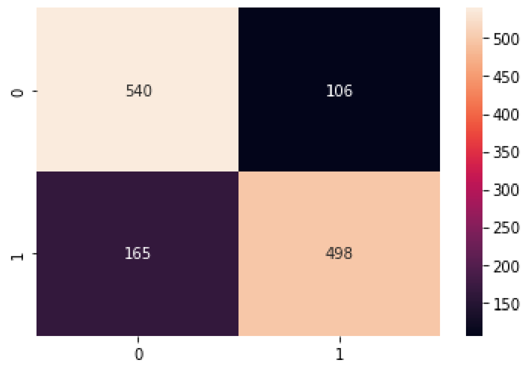
- Confusion Matrix: Count Vectorizer



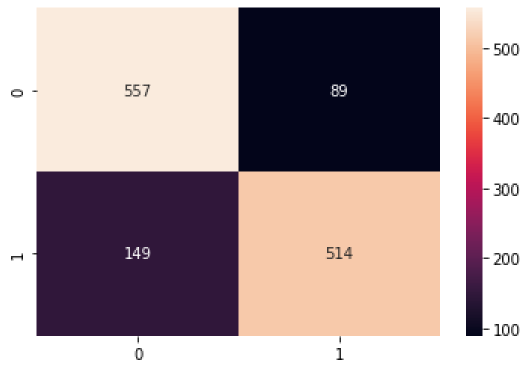
- Confusion Matrix: TF-IDF



- Confusion Matrix: N-Grams



- Confusion Matrix: ALBERT



## VII. Conclusion

Our experimental results from the above models prove that it is possible to categorize relevant tweets with high precision while maintaining fairly high recall. The ALBERT model provided us with an accuracy of nearly 0.82. While all different techniques in which we used Count Vectorization, TF-IDF and N-grams along with Logistic Regression gave us an accuracy of approximately 0.79. Even though all these approaches are fairly equal in terms of accuracy we will still prefer to say that ALBERT outperforms others, since we need to take into account all the factors like Precision, Recall and F1-Score for the comparison.

## Contributions

The contributions from each member of the group have been provided below -

- Akshay Jain - Contributed significantly in terms of decision regarding feature selection, Exploratory Data Analysis , designed and trained the model on TF-IDF and n-grams.
- Sahil Jain - Designed, developed, and trained the ALBERT deep learning model on the dataset. Implemented the script to clean text data. Contributed in the process of feature selection and EDA.
- Viraj Shah - Contributed in decision making for feature selection and designed and trained the model for CountVectorization.

## References

[1] X. Fang, J. Zhan, “Sentiment analysis using product review data,” Journal of Big Data volume 2, Article number: 5 ,2015.
[2] K. Stowe, M. Paul, M. Palmer, L. Palen, K. Anderson, “Identifying and Categorizing Disaster-Related Tweets“, University of Colorado, Boulder, 2022.
[3] “Natural Language Processing with Disaster Tweets“ , Kaggle, 2014. https://www.kaggle.com/c/nlp-getting-started/overview

[4] A K Ningsih and A I Hadiana, “Disaster Tweets Classification in Disaster Response using Bidirectional Encoder Representations from Transformer (BERT)“ ,2021 IOP Conf. Ser.: Mater. Sci. Eng. 1115 012032.

[5] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R., 2019. Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.

[6] HuggingFace Transformer Models https://huggingface.co/

[7] Scikit-Learn machine learning models for python: https://scikit-learn.org/stable/

[8] Understanding TF-IDF for Machine Learning : https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

[9] An Introduction to N-grams: What Are They and Why Do We Need Them : https://blog.xrds.acm.org/2017/10/introduction-n-grams-need/