Natural Language Processing with Disaster Tweets.

Abhiteja Mandava, Yoonjung Choi, Ishaan Bhalla, Sakruthi Avirineni

Abstract— Social Network Services (SNS) have become not only an important source of emergency information during disasters but also a medium for expressing immediate responses of warning, evacuation or rescue. Because smartphones are so common, people can use them to broadcast an emergency in real time. As a result, more organizations (such as disaster relief organizations and news companies) are interested in programmatically monitoring tweets, however it's not always clear whether a person's statements are announcing a calamity. By analyzing context, we can utilize this study to track, monitor and predict disasters from the real time data and this study would help make prediction models.

I. INTRODUCTION

Natural Language Processing (NLP) is a branch of artificial intelligence that helps computers understand and interpret human language and give computers the ability to understand written and spoken words. NLP is one of the most promising avenues for social media data processing and NLP has been widely used to analyze SNS and extract potential patterns.

Twitter is one of the popular SNS platforms and many tweets have been delivered in emergencies. Since there are demands for companies to utilize these tweets, we investigate natural language processes and develop prediction models having better performance in this paper.

The problem can be viewed as a binary classification problem and this project's goal is to figure out how to tell which tweets are about "genuine disasters" and which aren't. This project will involve experimentation on various machine learning models that will predict which tweets are about "actual disasters" and which aren't.

II. METHODS

To develop the models, we have designed a sequence of steps. The steps involved are:

1. Data Collection
2. Exploratory Data Analysis
3. Data Preprocessing
4. Feature Selection
5. Extraction
6. Data Split
7. Model Training, Parameter tuning
8. Model Evaluation
9. Best Fit Analysis

Workflow:

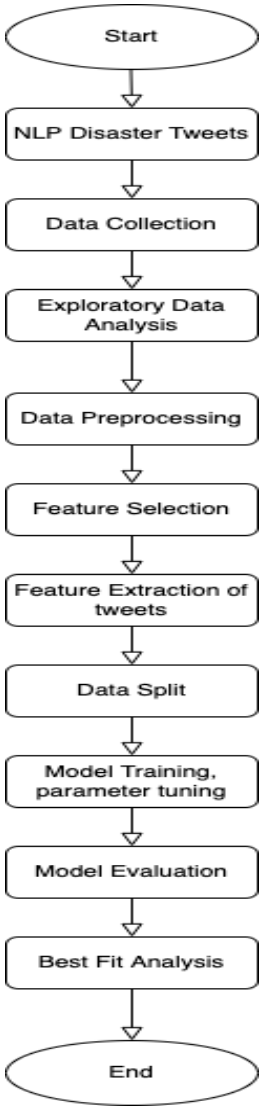


Figure 1. Sequence of tasks.

Data Collection: The data set has been collected from the company figure-eight and originally shared on their 'Data for Everyone' website [2]. We found the data set from Kaggle Competition [3]. It contains 7613 tweets data with the following features:

Exploratory Data Analysis, Data Preprocessing and Feature Selection.

Feature	Dtype	Descrition
id	int64	
keyword	object	39 null-values
location.	object	2533 null-valus
text	object	tweetter content
target	int64	0:non-disaster, 1:disaster

Figure 2. Description of data set.

Observation: We observed that the data set has four features and one label.

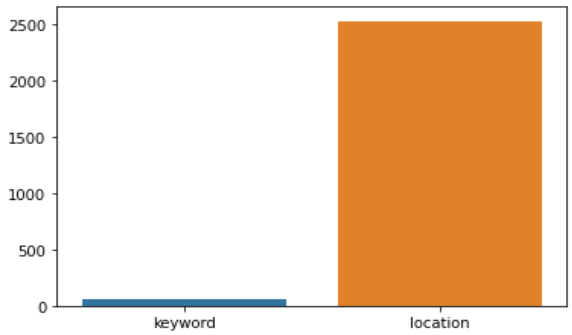


Figure 3. The number of missing values in the data set.

Observation: We observed that the 'location' feature has many null values (2533) and the 'keyword' feature has null values (39).

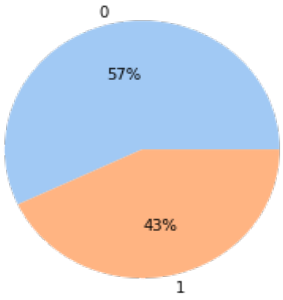


Figure 4. Pie chart of 'target' label.

Observation: We cannot say that it has a perfectly balanced dataset, but slightly it is a balanced data set.

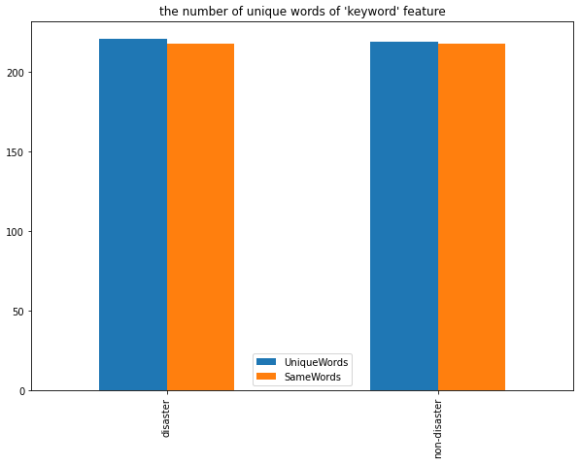


Figure 5. The number of unique words and common words at the respective label (disaster and non-disaster).

Observation:

1. The 'keyword' feature does not have many null values (39), but there are many common words. It has 221 words of disaster, and 219 words of non-disaster.
2. There are 218 intersection words.
3. The difference is that only 'disaster' tweets have been {'debris', 'wreckage', 'derailment'}.
4. The difference is that only 'non-disaster' tweets have been {'aftershock'}.
5. Thus, we do not use keyword features.

location
Karachi
Happily Married with 2 kids
Happily Married with 2 kids
Loveland Colorado
|| c h i c a g o ||

Figure 6. Partial samples of 'location' data set.

Observation:

1. The 'location' feature has many null values (2533) and does not have a format and is not generated automatically.

2. This feature has invalid data such as 'Happily Married with 2 kids, 'have car; will travel', 'peekskill. new york, 10566', or 'milky way'.
3. We do not use 'location' as a feature.

Concluded Observation of Exploratory Data Analysis and Data preprocessing: The 'id' feature is nominal data, which means that there is no meaningful information. Finally, it was decided to drop 'id', 'keyword', 'location' features and used only 'text' feature and 'target' label.

Implementation: For cleaning text, we changed all words to lowercase, and removed URL, HTML tags, Emojis, punctuation, and ASCII codes [4],[5].

Sample of the original 'text' feature can be viewed below.

text
Crying out for more! Set me ablaze
On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE http://t.co/qqsmshaJ3N
@PhDSquares #mufc they've built so much hype around new acquisitions but I doubt they will set the EPL ablaze this season.
INEC Office in Abia Set Ablaze - http://t.co/3lmaomknnA
Barbados #Bridgetown JAMAICA ‰ÛÒ Two cars set ablaze: SANTA CRUZ ‰ÛÓ Head of the St Elizabeth Police Superintende... http://t.co/wDUEaj8Q4J

Figure 7. Partial data set of 'text' feature.

Observation: We observed each sample has mixed data such as upper/lower cases, URL, and emojis.

Changes after Data Cleansing can be viewed below.

CleanText
crying out for more set me ablaze
on plus side look at the sky last night it was ablaze
phdsquares mufc theyve built so much hype around new acquisitions but i doubt they will set the epl ablaze this season
inec office in abia set ablaze
barbados bridgetown jamaica two cars set ablaze santa cruz head of the st elizabeth police superintende

Figure 8. Partial data set after cleaning meaningless data.

Observation: Incomplete, duplicated, incorrect, and irrelevant data from the dataset is removed.

However, there is still an additional implementation that we can do to extract meaningful information from the cleaned data. We apply stemming or lemmatization to get normalized words. Natural Language Toolkit (NLTK)[6] provides easy-to-use interfaces for natural language processing.

Tokenization: Tokenization divides strings into lists of substrings. We used the library to find the words and punctuation in sentences.

Changes after applying tokenization can be viewed below.

TokenizedText
['crying', 'out', 'for', 'more', 'set', 'me', 'ablaze']
['on', 'plus', 'side', 'look', 'at', 'the', 'sky', 'last', 'night', 'it', 'was', 'ablaze']
['phdsquares', 'mufc', 'theyve', 'built', 'so', 'much', 'hype', 'around', 'new', 'acquisitions', 'but', 'i', 'doubt', 'they', 'will', 'set', 'the', 'epl', 'ablaze', 'this', 'season']
['inec', 'office', 'in', 'abia', 'set', 'ablaze']
['barbados', 'bridgetown', 'jamaica', 'two', 'cars', 'set', 'ablaze', 'santa', 'cruz', 'head', 'of', 'the', 'st', 'elizabeth', 'police', 'superintende']

Figure 9. Partial data set after applying tokenization on previous cleaned data.

Observation: We observed the separated words in each sample.

Stop words: We removed commonly used words, such as "the", "a", "is", and "in".

Changes after applying stop words can be viewed below.

RemoveStopWords
['crying', 'set', 'ablaze']
['plus', 'side', 'look', 'sky', 'last', 'night', 'ablaze']
['phdsquares', 'mufc', 'theyve', 'built', 'much', 'hype', 'around', 'new', 'acquisitions', 'doubt', 'set', 'epl', 'ablaze', 'season']
['inec', 'office', 'abia', 'set', 'ablaze']
['barbados', 'bridgetown', 'jamaica', 'two', 'cars', 'set', 'ablaze', 'santa', 'cruz', 'head', 'st', 'elizabeth', 'police', 'superintende']

Figure 10. Partial data set after removing stop words on tokenized data.

Observation: In the first sample, 'out', 'for', 'more', 'set', 'me' are removed.

Stemming: Stemming is the process of producing morphological variants of a root/base word. For example, words such as “Likes”,” liked”,” likely” and “liking” will be reduced to “like” after stemming. There are different algorithms for stemming. Porter Stemmer, one of them, is a basic stemmer and it is straightforward and fast to run.

Changes after applying stemming can be viewed below.

PorterStemmer
['cri', 'set', 'ablaz']
['plu', 'side', 'look', 'sky', 'last', 'night', 'ablaz']
['phdsquar', 'mufc', 'theyv', 'built', 'much', 'hype', 'around', 'new', 'acquisit', 'doubt', 'set', 'epl', 'ablaz', 'season']
['inec', 'offic', 'abia', 'set', 'ablaz']
['barbado', 'bridgetown', 'jamaica', 'two', 'car', 'set', 'ablaz', 'santa', 'cruz', 'head', 'st', 'elizabeth', 'polic', 'superintend']

Figure 11. Partial data set after applying stemming on data without stop words.

Observation: We observed some changes in words. The 'crying' changed to 'cri' or 'acquisitions' changed to 'acquisit’.

Lemmatization: Lemmatization is the process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form. Both stemming and lemmatization are word normalization techniques, but we can find the word in the dictionary in the case of lemmatization.

For example, the original words 'populated' changed to 'popul' in Stemming, but it is not changed in lemmatization. Lemmatization has better performance than Stemming [7].

Changes after applying lemmatization can be viewed below.

LemmatizedText
['cry', 'set', 'ablaze']
['plus', 'side', 'look', 'sky', 'last', 'night', 'ablaze']
['phdsquares', 'mufc', 'theyve', 'built', 'much', 'hype', 'around', 'new', 'acquisition', 'doubt', 'set', 'epl', 'ablaze', 'season']
['inec', 'office', 'abia', 'set', 'ablaze']
['barbados', 'bridgetown', 'jamaica', 'two', 'car', 'set', 'ablaze', 'santa', 'cruz', 'head', 'st', 'elizabeth', 'police', 'superintende']

Figure 12. Partial data set after applying lemmatization on data without stop words.

Observation: We observed some changes of words. Unlike stemming, lemmatization made that 'crying' changed to 'cry' or 'acquisitions' changed to 'acquisition'.

Data Visualization:

After performing the aforementioned steps on the feature 'text', we made data visualization using word cloud [8].

Word cloud on disaster can be viewed below.

Figure 13. Word Cloud on 'target' labeled as a disaster.

Observation: We discovered disaster tweets' related words: suicide, police, news, kill, attack, death, California, storm, flood.

Word cloud on non-disaster can be viewed below.

Figure 14. Word Cloud on 'target' label as non-disaster.

Observation: Non disaster tweets show words unrelated of disasters: time, want, great, feel, read. Also, injury or emergency.

III. Feature Extraction

Word Embedding to transform data into numerical feature vectors.

One of the biggest problems with text is that it is messy and unstructured, and machine learning algorithms need structured, properly defined fixed-length inputs. To train text on machine learning models, we need to transform 'text' features (words or sentences) into fixed-length numerical feature vectors.

Word embedding is one of the most popular representations of document vocabulary. It can capture the context of a word in a document, semantic and syntactic similarity, and relation with other words. There are a few methods we can use to transform the text into numerical feature vectors.

Count Vectorizer: The Bag-of-Words model is a simplified representation used in NLP. A text is represented as the bag of its words, disregarding grammar, and describes the occurrences of words within a document. Count Vectorizer can be used for a bag of words model.

Count Vectorizer can be used for a bag of words model. This converts a collection of text documents to a matrix of token counts and transforms the text into fixed-length vectors.

Occurrences of words by Count Vectorizer can be viewed below.

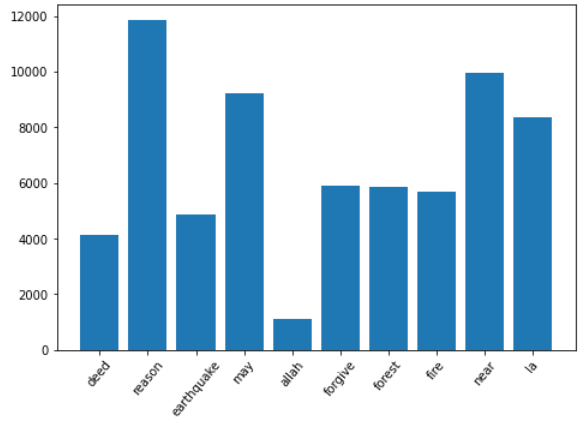


Figure 15. Occurrences of words by Counter Vectorizer

Observation: We observed the frequencies of words. The word 'deed' was present in 4000 odd occurrences. The word 'reason' was present in 10 odd occurrences.

Term Frequency Inverse Document Frequency (Tf-Idf): TF-IDF is better than Count vectorizer because it not only focuses on the frequency of words present in the corpus but also provides the importance of the words. Hence, we removed the words that are less important for analysis making the model building less complex.

Occurrences of words by Tf-Idf can be viewed below.

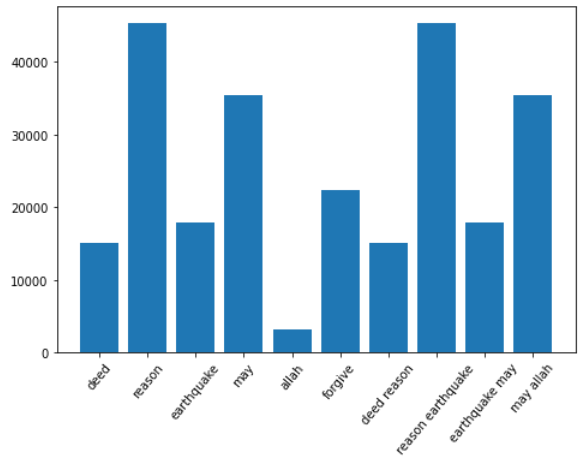


Figure 16. Occurrences of words by Tf-Idf.

Observation: We observed weighted frequencies of words. The word 'deed' looks more weighted, and the occurrences changed to around 15000. The words

'reason' and 'reason earthquake' have appeared to weigh more as well.

Word2Vec: The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence [9]. Word2Vec represents words in vector space in a way of similar meaning: words are positioned in close locations, but dissimilar words are placed far away. We followed the genism tutorial [10] to visualize the relationship with words.

Observation: Blue represents most similar words associated with the 'Wildfire' and green represents most unrelated words associated with 'Wildfire'.

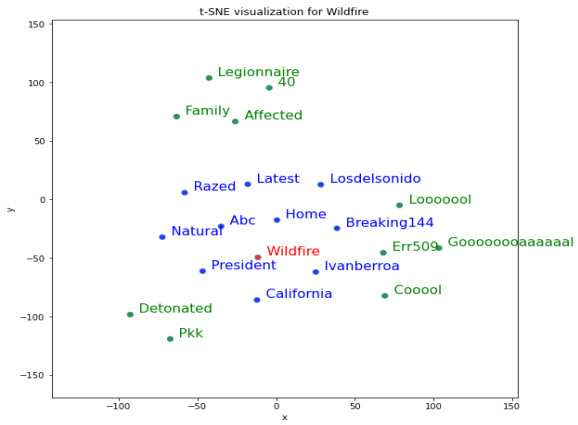


Figure 17: Visualization of words using Word2Vec

Word2Vec with PCA applied: Principal Component Analysis is a strategy to reduce dimensionality and identify important relationships in data and extract new features. According to the Genism guide, Word2Vec's default dimensionality is 100, so we can have direct 100 dimensions data set without PCA. However, we used PCA to reduce 100 components from Word2Vec with a dimensionality of 300 since PCA not only reduces dimensionality but also extracts new features.

We have noticed that the implementation of PCA has better performance than the word2vec feature vector set with 100 dimensionalities without PCA.

Glove: Glove stands for global vectors for word representation. It is an unsupervised learning algorithm developed by Stanford for generating word embeddings by aggregating global word-word co-occurrence matrices from a corpus [11]. It was decided to use the glove.6B.100d.txt file containing the glove vectors trained on the Wikipedia and Giga Word datasets.

The difference between Word2Vec and Glove is the way of training. The glove is based on the global word to word co-occurrence counts utilizing the entire corpus, on the other hand, Word2Vec uses co-occurrence within local neighboring words.

We build four feature vectors from Count Vectorizer, Tf-Idf, Word2vec, and Word2Vec with PCA applied. Glove and Word2Vec embedding are used for the LSTM model. We are not able to apply PCA on Bag of Words feature vector sets because of its sparsity. Transformed feature vector sets have respective shapes (7613,16270) from Count Vectorizer, shapes (7613, 63245) from Tf-Idf, shape (7613,300) from Word2Vec, and shape (7613,100) from Word2Vec applied PCA.

IV Models

1. Logistic Regression (LR): Logistic Regression is a supervised machine learning algorithm that can be used to model the probability of a certain class or event. It is used when the data is linearly separable, and the outcome is binary or dichotomous in nature [12]. That is the reason we have used LR to solve a binary classification problem in our case.

Performance on Logistic Regression without modifying parameters can be viewed below.

LR	Count Vector	Tf-Idf	Word2Vec	Word2Vec + PCA
Accuracy	0.797	0.776	0.666	0.761

Table 1. Logistic Regression's accuracies with respective to feature vector sets.

Results and confusion matrix of the model can be viewed below.

LR	Accuracy	Recall	Precision	F1Score
	0.801	0.687	0.826	0.750

Table 2. Logistic Regression's performance with Count Vectorizer feature vectors set.

Observation: We observed that the accuracy is improved (0.801). However, it's not significant.

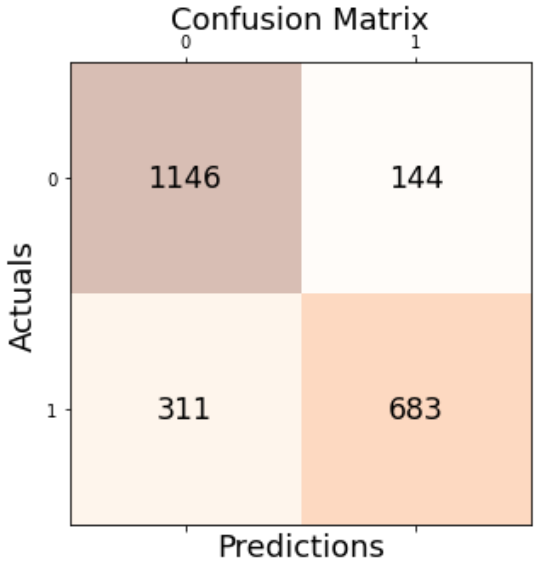


Figure 18. Confusion Matrix of Logistic Regression with Count Vectorizer feature vectors set.

Observation: This confusion matrix shows the number of samples between prediction and actuals. This Logistic Regression model predicts 683 true positive (disaster) and 1146 true negative(non-disaster) samples.

2. Support Vector Machine: Support Vector Machine is a supervised learning model used for classification and regression problems. SVM can be used when data has exactly two classes. SVM classifies data by finding the best hyperplane that separates all data points of one class from those of the other class. The best hyperplane for an SVM means the one with the largest margin between the two classes [16]. That is the reason we have used SVM to solve a binary classification problem in our case.
Performance on SVM without modifying parameters can be viewed below

SVM	Count Vector	TF-IDF	Word2Vec	Word2Vec +PCA
Accuracy	0.799	0.761	0.627	0.712

Table 3. SVM's accuracies with respective to feature vector sets.

Results and confusion matrix of the model can be viewed below.

SVM	Accuracy	Recall	Precision	F1Score
	0.800	0.688	0.839	0.744

Table 4. SVM's performance with Count Vectorizer feature vectors set.

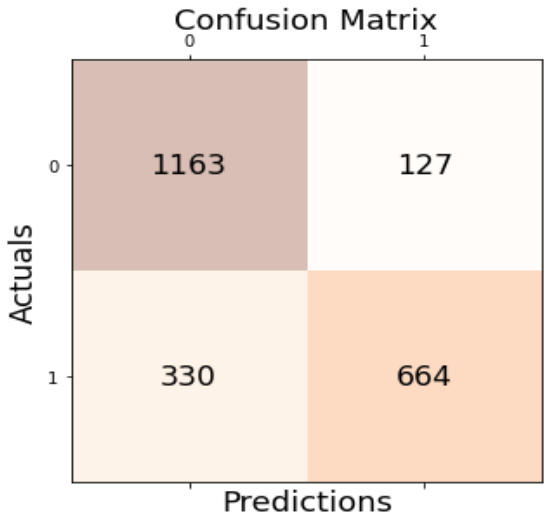


Figure 19: SVM confusion Matrix

3. Decision Tree: A decision tree can be used for either regression or classification. Decision Tree uses 'entropy' or 'Gini' to calculate impurity of split and obtains information gain, and then decides which node splits in a way of having as much as possible higher information gain. Advantages of classification with Decision Trees are inexpensive to construct, extremely fast at classifying unknown records, easy to interpret for small-sized trees, accuracy comparable to other classification techniques for many simple data sets and excludes unimportant features. Thus, we try to train data on decision trees as well. Table 5 shows performance on the Decision Tree without modifying parameters.

DT	Count Vector	TF-IDF	Word2 Vec	Word2Vec +PCA
Accuracy	0.746	0.749	0.650	0.664

Table 5. DT's accuracies with respective to feature vector sets.

Results and confusion matrix of the model can be viewed below.

DT	Accuracy	Recall	Precision	F1Score
	0.756	0.681	0.737	0.708

Table 6. DT's performance with TF-IDF feature vectors set.

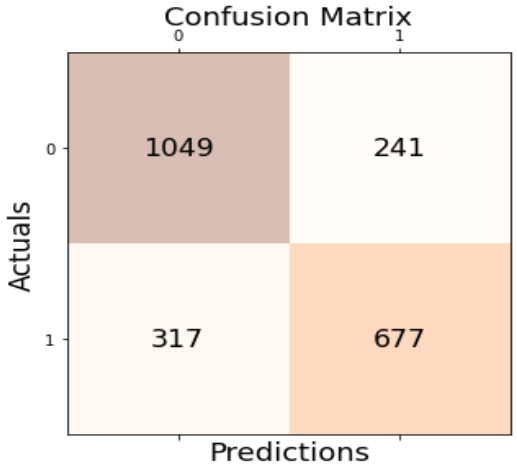


Figure 20: Confusion Matrix of Decision Tree with Tf-Idf feature vectors set.

4. Random Forest: Random Forest is a supervised learning algorithm. It can be used for both classification and regression. However, it's mainly used for classification problems. A forest comprises trees and it's said that the more trees it has, the more robust the forest is. Random Forest is a set of multiple decision trees. Random Forest creates decision trees on randomly selected data samples, gets predictions from each tree, and selects the best solution by means of voting. Decision trees may suffer from overfitting, but random forest prevents overfitting by creating trees on random subsets. Decision trees are computationally faster.
The results and confusion matrix of the model can be viewed below.

Random Forest	Count Vectorizer	Tf-Idf	Word2 Vec	Word2 Vec+PCA
Accuracy	0.785	0.768	0.733	0.771

Table 7. Random Forest's accuracies with respective feature vector sets.

Random Forest has better accuracy with the Count vectorizer feature vector set.

The results and confusion matrix of the model can be viewed below.

RF+CV	Accuracy	Recall	Precision	F1 Score
	0.799	0.704	0.809	0.753

Table 8. Random Forest's performance with Count vectorizer feature vectors set.

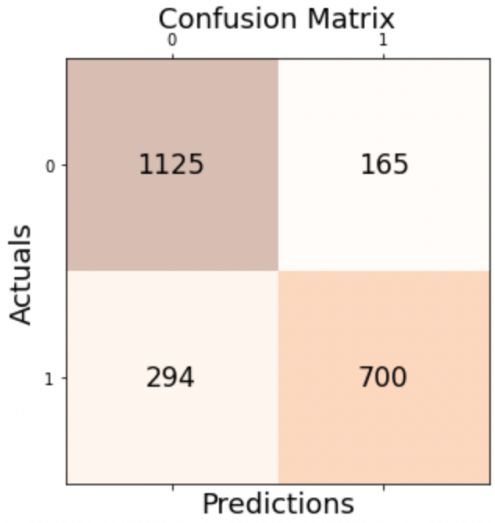


Figure 21. Confusion Matrix of Random Forest with CV feature vectors set.

Observation: We observed the RF + CV resulted in better accuracy (0.799) than other feature vector sets.

5. XGBoost: Extreme Gradient Boosting (XGBoost) is a distributed gradient-boosted decision tree (GBDT) machine learning toolkit that is scalable. Supervised machine learning, decision trees, ensemble learning, and gradient boosting are all used in

XGBoost. We were keen to use this approach and notice the results.

The results and confusion matrix of the model can be viewed below.

XGB	Count Vectorizer	Tf-Idf	Word2 Vec	Word2 Vec + PCA
Accuracy	0.713	0.729	0.729	0.753

Table 9. Xgboost's accuracies with respective feature vector sets.

Xgboost has better accuracy with Word2Vec+PCA feature vector set.

The results and confusion matrix of the model can be viewed below.

XGB	Accuracy	Recall	Precision	F1 score
	0.774	0.659	0.786	0.717

Table 10. XGBoost's performance with Word2Vec + PCA feature vectors set.

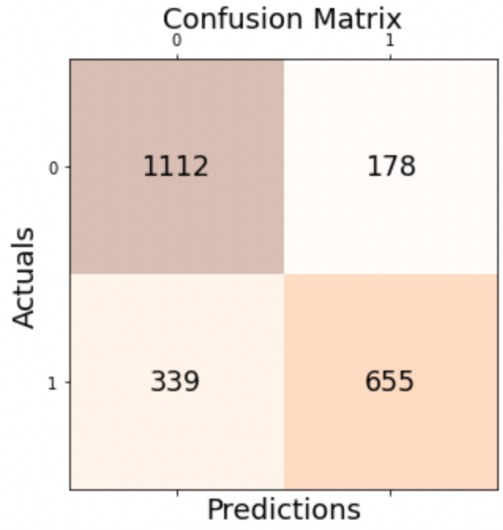


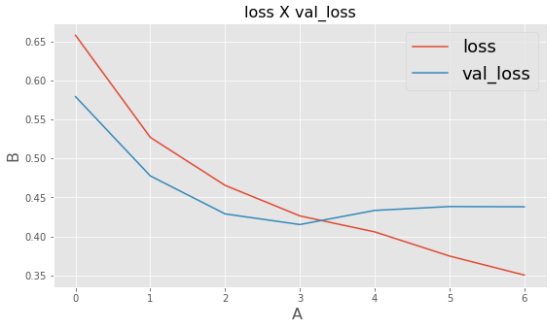
Figure 22. Confusion Matrix of XGBoost with PCA and Word2Vec feature vectors set.

Observation: We observed the XGB+PCA+W2V resulted in better accuracy (0.774) than other feature vector sets.

6. LSTM: Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning. The Long Short Term Memory architecture was motivated by an analysis of error flow in existing RNNs which found that long time lags were inaccessible to existing architectures because backpropagated error either blows up or decays exponentially [20].

An LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. These blocks can be thought of as a differentiable version of the memory chips in a digital computer. Each one contains one or more recurrently connected memory cells and three multiplicative units – the input, output and forget gates – that provide continuous analogs of write, read and reset operations for the cells. The net can only interact with the cells via the gates.

An LSTM has four "gates": forget, remember, learn and use (or output). It also has three inputs: long-term memory, short-term memory, and E. (E is some training example/new data). Step 1: When the 3 inputs enter the LSTM, they go into either the forget gate, or learn gate. The long-term info goes into the forget gate, where, shocker, some of it is forgotten (the irrelevant parts). The short-term info and "E" go into the learn gate. This gate decides what info will be learned. Step 2: information that passes the forget gate (it is not forgotten, forgotten info stays at the gate) and info that passes learn gate (it is learned) will go to the remember gate (which makes up the new long-term memory) and the use gate (which updates short term memory +is the outcome of the network).



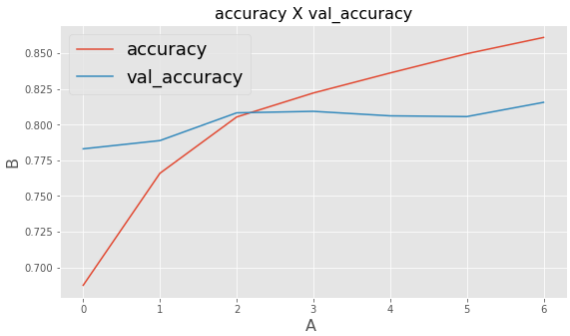


Figure 23: LSTM results

Observation:

We tried to set a random seed to make reproducible results, but still LSTM model fluctuates its accuracy score from 0.803 to 0.815.

LSTM + Glove	Accuracy	Recall	Precision	F1 Score
	0.809	0.793	0.759	0.775

Table 11. LSTM with Glove model's performance

Observation: LSTM has a higher accuracy rather than other models that we tried so far.

We trained the same LSTM model with Word2Vec for comparison. Results are shown in the below Table.

LSTM +W2V	Accuracy	Precision	Recall	F1 Score
	0.635	0.548	0.927	0.688

Table 12. LSTM with word2vec model's performance

Observation: In this case, the model with Word2Vec has a worst performance rather than the model with Glove. However, this model has many possibilities to improve considering that we do not use optimization. We will apply optimization in the further study since we found that there are many parameters to improve performance [21].

7. Ensemble:

Ensemble methods are techniques that create multiple models and then combine them to produce improved results. Ensemble methods usually produce more accurate solutions than a single model would. We have four different feature sets and random state parameters enable split feature vector sets in the same way, which means we can use ensemble models on our own. Based on the voting way, first ensemble model consisted of non-sequential models; Logistic Regression with Count vectorizer, SVM with Counter vectorizer, Decision Tree with Tf-Idf, Random Forest with counter vectorizer, Xgboost with word2vec applied PCA.

The table shows the accuracy, recall, precision, and f1 score of the ensemble model.

Ensemble	Accuracy	Recall	Precision	F1 score
	0.812	0.708	0.835	0.766

Table 13. Performance of ensemble model without LSTM

Observation: When we use the ensemble model, accuracy is better than accuracy from respective nonsequential models.

The next ensemble model is the first ensemble model adding the deep learning model, LSTM with Glove. This ensemble model also fluctuates performance depending on the performance of the LSTM model. The below table shows the accuracy, recall, precision, and f1 score when LSTM with Glove model has 0.809 accuracies.

Ensemble	Accuracy	Recall	Precision	F1 score
	0.821	0.697	0.865	0.772

Table 14. Performance of Ensemble model

Observation: When we use the ensemble model, accuracy is better than accuracy from respective models.

V Comparisons.

Performance Metrics

Accuracy: Accuracy is a metric that generally describes how the model performs across all classes. It is calculated as the ratio between the number of correct predictions to the total number of predictions.

Precision: The precision is calculated as the ratio between the number of Positive samples correctly classified to the total number of samples classified as Positive (either correctly or incorrectly).

Recall: The recall is calculated as the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples.

F1 Score: The F1 score combines the precision and recall of a classifier into a single metric.

ROC Curve: ROC curve is a graphical plot that illustrates recall(x-axis) and precision(y-axis) for different thresholds.

Comparison:

From non-sequential modeling on one classifier and four-word embeddings, we expected to find the best combination having better accuracy. We found that when Logistic Regression, SVM, Random Tree, XGBoost trained on Count Vectorizer feature vectors set, Decision Tree trained on Tf-Idf feature vectors set, and XGBoost trained on Word2Vec applied PCA feature vectors set, they resulted in the highest accuracy in each of the experiments. In addition, although we were not able to acquire much improvement by optimizing parameters, we obtained a little improvement we learned that sometimes default parameters have good accuracy because they are supposed to work for general purposes. On the ensemble model of non-sequential models, we found that the ensemble model has better performance than cases using a single model.

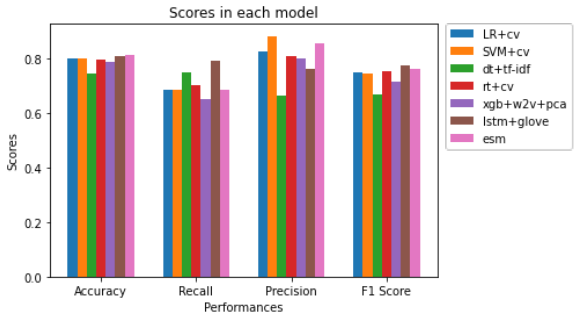


Figure 24: Scores In each Model

From deep learning modeling on LSTM and two-word embeddings; word2vec and glove, LSTM trained on the Glove feature vectors set obtained the highest accuracy. LSTM has better accuracy than other models. However, we also noticed that the difference is not significant and non-sequential models had similar performance to LSTM model.

ROC CURVE:

We obtained ROC Curve and AUC (Area under the ROC Curve) of respective combinations of models and word embeddings. Since we used the voting way of ensemble, we were unable to make ROC curve and AUC of ensemble model. As we can see in Figure and Table, LSTM with Glove has the highest area under the curve (0.881).

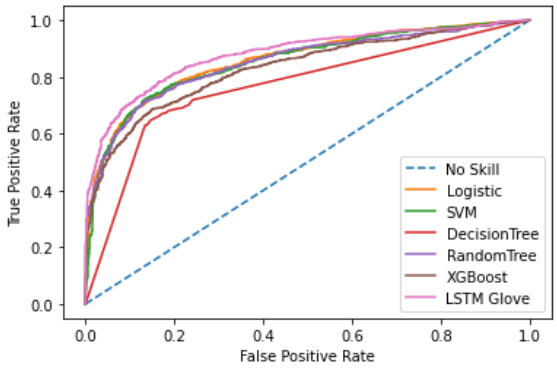


Figure 25: ROC curves comparison

Comparison Table:

Models	AUC
LR + CV	0.859
SVM + CV	0.854
DT + TF-IDF	0.767
RT + CV	0.852
XGB + W2v + PCA	0.831
LSTM + Glove	0.881

Market Comparison:

Other submissions on Kaggle have done similar steps for preprocessing and applying models. They tried to train a single model or even in case of ensemble, they trained ensemble classifiers with the same data set. However, we had taken a different direction expecting that there would be a suitable combination of feature vector sets and models.

VI. Conclusions

In this analysis, we experienced four prominent word embeddings and seven classification techniques using a Figure-Eight Company data set. LSTM with glove has good performance among individual models and the ensemble model with combined different feature vectors and these classifiers have outperformance the other classifiers on each data set.

VII. Limitations and Future Research

We obtained the qualified data set from the company, so we assumed that the data is reliable. However, the fact that data could be not truthful is the main limitation of this study. Overcoming these limitations can be done in future research. By distinguishing the reliability of data first, we can analyze and predict emergencies properly. Additional studies on deep learning algorithms should be continued. When we made models on LSTM with word embeddings, we faced many difficulties in understanding complicated algorithms themselves and choosing diverse optimization options. In a further study, we will continue dealing with our concerns.

VIII. References

[1] NLP Market Research, https://www.statista.com/statistics/607891/worldwide-natural-language-processing-market-revenues/

[2] Data For Everyone' website, https://appen.com/datasets-resource-center

[3] Kaggle Competition, https://www.kaggle.com/competitions/nlp-getting-started/data

[4] Text Preprocessing for NLP (Natural Language Processing),Beginners to Master, https://medium.com/analytics-vidhya/text-preprocessing-for-nlp-natural-language-processing-beginners-to-master-fd82dfecf95

[5] Text Preprocessing in NLP, https://towardsdatascience.com/text-preprocessing-in-natural-language-processing-using-python-6113ff5decd8

[6] Natural Language Toolkit, https://www.nltk.org/index.html

[7] Natural Stemming-vs-lemmatization, https://www.baeldung.com/cs/stemming-vs-lemmatization

[8] WordCloud, https://kavita-ganesan.com/python-word-cloud/#.YnnPcPPMIeU

[9] Word2Vec Wikipedia, https://en.wikipedia.org/wiki/Word2vec

[10] Word2Vec, gensim-word2vec-tutorial, https://www.kaggle.com/code/pierremegret/gensim-word2vec-tutorial/notebook

[11] What is Glove?, https://medium.com/analytics-vidhya/word-vectorization-using-glove-76919685ee0b

[12] Glove File from Kaggle, https://www.kaggle.com/datasets/danielwillgeorge/glove6b100dtxt

[13] Logistic Regression, https://www.sciencedirect.com/topics/computer-science/logistic-regression

[14] Logistic Regression API, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[15] Logistic Regression Sparsity, https://scikit-learn.org/stable/auto_examples/linear_model/plot_logistic_l1_l2_sparsity.html

[16] SVM How it works, https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[17] SVM Kernel Trick, https://datamites.com/blog/support-vector-machine-algorithm-svm-understanding-kernel-trick/

[18] Decision Tree, https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[19] XGBoost, https://www.nvidia.com/en-us/glossary/data-science/xgboost/

[20] LSTM, https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

[21] Kera's Optimizers API, https://keras.io/api/optimizers/