Analysis of Machine Learning Algorithms for the Identification of Disaster Related Tweets

Allison Fleming, Seoyeon Lee, Nihal Hossam, Irina Shcherbakova, Ruixin Tang

March 2020

Abstract. In recent years, Twitter has become a major channel for communication during disasters. We were presented with a binary classification problem concerning the detection of disaster related tweets within a large pool of tweets. We evaluated the ability of different machine learning models to automatically distinguish between the classifications and compared their performance based on several metrics. Experimental results showed that Logistic Regression has the best results compared to the other classifiers, obtaining a F1 score of 0.740, precision value of 0.820, and recall value of 0.680. We conclude that the heterogeneity of the positive class of the disaster tweet dataset causes noisiness in the data which favors a model that gives more attention to instances near the class boundary, such as the Logistic Regression Classifier.

1 Introduction

Disasters often happen suddenly and without warning. Early notification to first responders can aid in the mitigation of loss of life and damage to infrastructure. Due to the ubiquity of mobile phone use, disaster information is often available in real-time from eye witnesses. Twitter is a common application used by both official agencies as well as individuals to disseminate information to a wide audience. A large corpus of research has been conducted to these ends.

A problem emerges in respect to the identification of tweets related to a real disaster as opposed to those which are conversational. Keywords alone are not adequate as such words are often used conversationally as well. For example, "Crying out for more! Set me ablaze" represents a conversational use of the word ablaze while, "huge fire at Wholesale markets ablaze" concerns a real disaster. This complexity presents a good opportunity for the use of machine learning techniques. Therefore, this research is aimed to evaluate binary classification models for identifying disaster related tweets.

2 Related Work

A substantial amount of research has been conducted on the use of Natural Language Processing (NLP) with Machine Learning applications for the classification of text as well as the specific task of the identification of disaster related tweets. Approaches differ depending on the intended use of the information. We identified three major categories of research: 1. the identification of tweets which can help first responders and other aid agencies to identify and prioritize those needing help, 2. creating a model which is adequately generalizable to be extrapolated to non-specific disasters, and 3. disseminating useful information during a disaster.

Aiding first responders in disaster situations is a primary objective for much of the research in this area. Several approaches were identified to solve this problem with no real consensus on the best model. Singh et al. found that a Random Forest classifier was the best model for classifying tweets containing people asking for help and assigning low vs high priority to them [18]. However, Kabir and Madria's research suggested a Convolutional Neural Network model in which they augmented the dataset with semantic information is the best model for the same task [9]. Ashkatorab et al. performed a similar binary classification to identify tweets related to infrastructure damage and human casualty information but found Logistic Regression the best predictor [3]. Bai and Yu successfully identified Weibo users who needed help in a post-disaster sit-

uation by performing a sentiment analysis using an SVM model [4].

The second major division in approach concerns generalizing the classifier. Most classifiers are trained using labeled data from specific events. However, to be realistically useful in an emergency situation, the classifier must be adequately generalizable to new situations. This problem is exacerbated when considering that not only may the topic of the emergency situation change from one event to another (for example an earthquake versus a flood), but also the language itself as expressed in the tweets can vary between locations due to regional dialects, use of local place names, and local differences in the use of Twitter. For example, Pekar et al found a 40% decrease in accuracy of label predictions when training and testing on a heterogeneous dataset of incident types versus a homogeneous one [14]. Schulz, Guckelsberger, and Jannsen used features of Linked Open Data (LOD) to incorporate semantic abstraction for local place references to increase the accuracy of their model for generalized use [16]. Kabir and Madria established a similar approach to identify semantic meaning using global vectors for word representation (GLoVe) and pre-trained disaster word vectors [9]. Li et al. employed an approach that combines a Naive Bayes classifier which was pre-trained on an existing labeled data set. To increase accuracy for a specific event, they used iterative self-training to add new labels derived from incoming tweets whose label predictions are above a threshold considered "confident". This resulted in an increase in accuracy of the model when applied to new, different disaster events [12]. Palshikar, Apte, and Pandita proposed a similar method, which used news articles on disasters to create the initial word vectors (bag of words) which were subsequently adapted using an online method which adjusts the weights attributed to words depending on new incoming tweet information [13].

Finally, the dissemination of important information to those affected by a disaster comprises the last category of research in this area. Twitter poses a potentially useful source of information during disasters, however the sheer volume of information and noisiness of the data is a major drawback to its use for this purpose. This area of research tried to identify helpful information from reliable sources. One method was to consider the formality versus informality of tweets as an aid in classification [1,19]. In many of these studies, additional features were added using both manual and derived methods to give the classifier more information for the analysis. For example, Truong et al. reasoned that important information is more likely to be related in more formal tweets. To do this, they used a Naive Bayes model and combined the bag of words technique with a feature vector containing added information such as "informative URL," "has emoticon," and "has curse word" meant to distinguish between these categories [20]. Gul et al. performed a sentiment analysis of the tweets sent during the Kashmir flood of 2014. They found that informational tweets were more likely to be liked and shared [8]. Each of these methods attempted to augment the normal bag of words approach to increase the discriminatory ability of the models.

3 Methods

The purpose of this project is to identify the best model that can be used for retrieving disaster-related tweets from a large pool of tweets. The pipeline in Figure 1 outlines the major steps of the classification process. Data preprocessing steps were similar for all models that took part in the experiment. On the contrary, tuning hyper parameters was not possible for all models since some models lack them (see section Naive Bayes). From each type of classifier, we selected the best performing model based on the accuracy it demonstrated on the validation set. Finally, we ran all selected models independently on the test set. The results are analyzed in section 6.

3.1 Data Inspection and Preparation

The training dataset consisted of a total of 7613 instances and 5 features: "id", "keyword", "location", "text", and "target". As can be seen from Figure 2, the dataset was not perfectly balanced but the target class (disaster-related tweets) was not significantly underrepresented. Therefore we did not apply additional steps for balancing the dataset.

Due to the informal nature of tweets and the usage of jargon, abbreviations, links, and non-alphabetic symbols, we applied extensive preprocessing steps to the training and test set: cleaning tweet text of non-alphabetic characters such

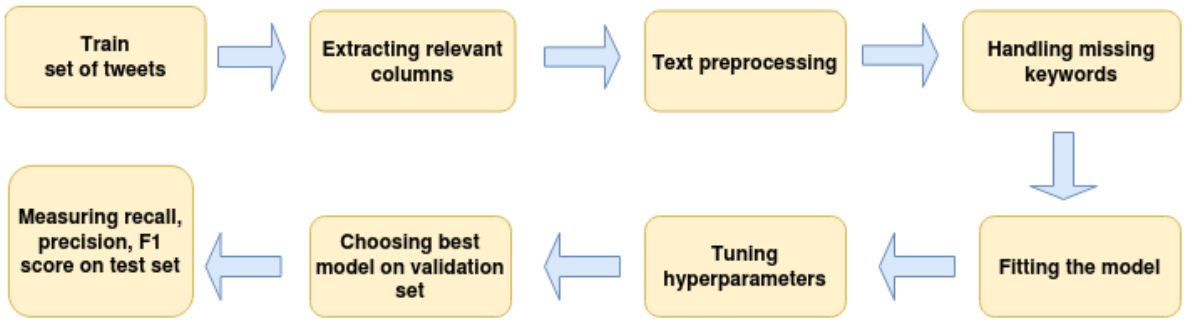

Figure 1: Pipeline of the classification process

as punctuation marks, numbers, and other symbols; removing http links from the text; making all tweets lower case; removing line breaks and all extra spaces; such as leading and trailing spaces; splitting text into tokens; applying grammar correction rules to tokens; removing stop word tokens such as articles and prepositions; and stemming the remaining tokens to their root form.

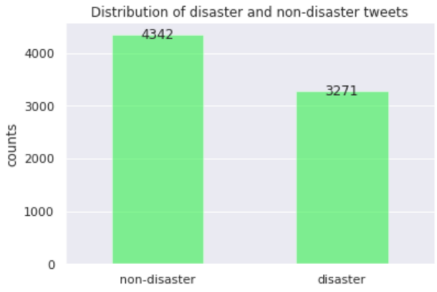

Figure 2: Distribution of tweets in the training set

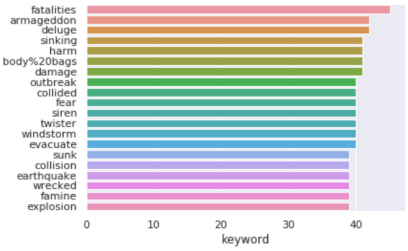

Figure 3: Top 20 keywords in the training set

The inspection of the data revealed that both training and test datasets contained missing values for the features "keyword" and "location". In particular, the training set lacked 61 values for the "keyword" feature and 2533 values for the "location" feature. As for the test set, 26 keywords and 1105 locations were missing among all of its instances. The "location" feature had only 5 unique values in total: USA, UK, Africa, India, and worldwide. Due to its general character and uselessness for the classification task, we removed it from the dataset. On the contrary, the "keyword" feature contained much more valuable information. There were in total 222 unique keywords (see Figure 3 of the top 20 keywords) all of which were somehow related to a disaster topic. Therefore additional steps were taken aimed to fill in missing values for this feature.

The first step consisted of extracting all unique keywords and applying usual preprocessing techniques to them: converting words to lowercase, splitting them into tokens, removing non-alphabetic characters and stop words, and stemming of the tokens.

During the second step, we extracted all instances with the missing value for the "keyword" feature. We compared each tweet to the set of unique keywords. If the tweet's text contained one of the keywords from this set, then this word was used as its keyword. If none of the unique keywords matched the tweet's text, the tweet was assigned a special non-disaster-related keyword - "other".

In the production environment this modification of the data could be also realistically applied. For every incoming tweet we could perform the same keyword extraction procedure that takes just a small constant amount of time. Therefore, we also

modified our test set and filled in missing values for the "keyword" feature.

In the last preprocessing step we transformed all tokens into a vector of numbers. The primary technique that we employed was term frequency-inverse document frequency (TF-IDF). TF-IDF prioritizes words that occur often in the particular document and at the same time penalizes words that occur often in the entire corpus. In this way noisy and irrelevant words have a higher chance of being disregarded from consideration. Therefore this technique helped to clean insignificant and unrelated words from the data that appear frequently in many tweets.

3.2 Models used in the experiment

3.2.1 Gradient Boosting Classifier

Gradient boosting is one of the most used classifiers in Kaggle data science competitions. This classifier builds a decision tree from many and takes several hyperparameters such as learning rate, maximum number of features, maximum depth of the tree, and maximum number of leaves [5,7].

Due to the large number of various hyper parameters, we applied a trial and error method in order to detect the best performing model on the validation set. In particular, we tested different values for the maximum number of features, maximum tree depth, and learning rates (see Table 1). The model with 100 features, a tree depth of 25, and a learning rate of 0.075 demonstrated the best accuracy on the validation set and therefore was chosen as the best model. Having the area under the curve of 0.836, this model puts a randomly drawn pair of disaster and non-disaster tweets in the correct order with the probability of 83.6 percent. This model achieved 0.78016 score on the Kaggle leader board.

3.2.2 Random Forest Classifier

The Random Forest classifier is an ensemble model that trains many individual decision trees and combines them into a single model [10]. Single decision trees are rather weak models that tend to have high variance and suffer from over-fitting. A trial and error approach on the hyper parameters revealed that the best model has 80 features. According to Table 2, this model showed a 0.791 accuracy score on the validation set and its area under the curve was

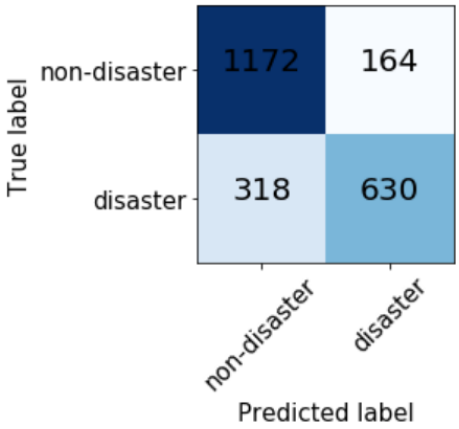

Figure 4: Confusion matrix for the best model of Gradient Boosting

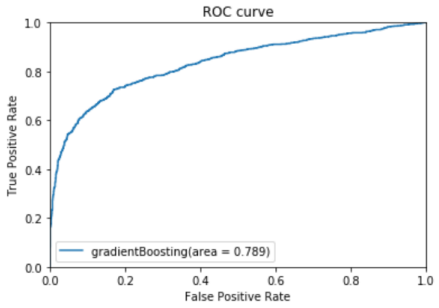

Figure 5: ROC curve for the best model of Gradient Boosting

0.835. This means that the possibility of selecting a randomly chosen pair of disaster and non-disaster tweets correctly is 83.5 percent using this model. It performs relatively the same as the Gradient Boosting Classifier.

3.2.3 Logistic Regression Classifier

The Logistic Regression Classifier is a statistical model that calculates the probability of being a class by applying the logistic sigmoid function. We applied grid search to select the best regularization strength. The parameter C represents the inverse of the regularization strength for the model. Therefore, a smaller C indicates a stronger regularization. Regularization helps models to generalize for unknown data rather than paying attention to specific data points during training. The grid search

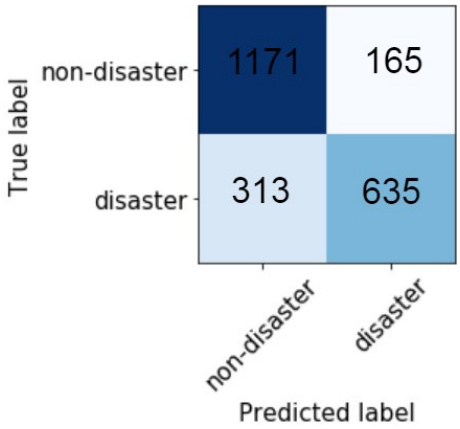

Figure 6: Confusion matrix Random Forest

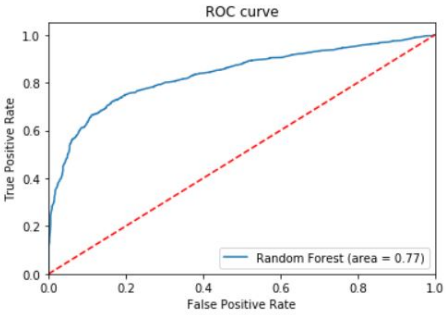

Figure 7: ROC curve Random Forest

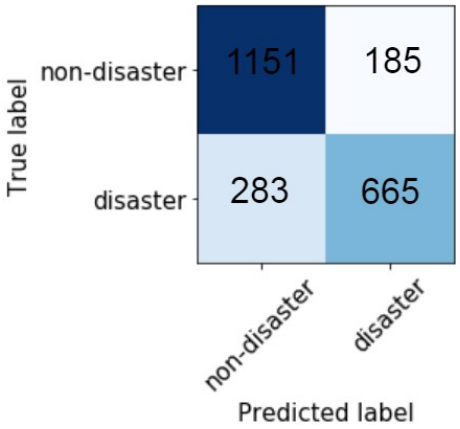

Figure 8: Confusion matrix Logistic Regression

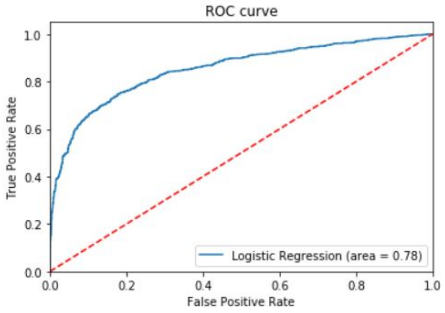

Figure 9: ROC curve Logistic Regression

showed a 0.795 accuracy on the validation set with $C = 1$. (see Table 3) Although models where C was greater than 1 had better accuracy (larger than 97%) on the training set, the models suffered from overfitting compared to the accuracy result shown in the validation set.

3.2.4 Naive Bayes Classifier

The Naive Bayes classifier is a probabilistic classifier based on the assumption that all features are independent given the value of the class. The various Naive Bayes classifiers which were used in this project differ mainly by the assumptions regarding the probability distribution of a particular feature given the class. The Gaussian Naive Bayes Classifier assumes that the likelihood of the features follow a Gaussian distribution. The Multinomial Naive Bayes Classifier assumes that the data is multinomially distributed, whereas the Complement Naive Bayes Classifier (a variation of Multinomial NB) uses statistics from the complement of each class to compute the model's weights. Bernoulli Naive Bayes Classifier assumes that the data is distributed according to a multivariate Bernoulli distribution. We tested all of the above mentioned models on the validation set and the Bernoulli Naive Bayes Classifier model showed the best results with an accuracy of 0.808. (Table 4 for specific result)

3.2.5 Neural Networks and LSTM

Neural Networks are complex models which try to mimic the way that the human brains learns. They consist of different layers of neurons, each layer receiving inputs and passing outputs depending on the weight given to that specific link. We used three different types of Neural Networks in the evaluation: a Recurrent Neural Network (RNN), a Bidirectional Neural Network, and a Long Short-term

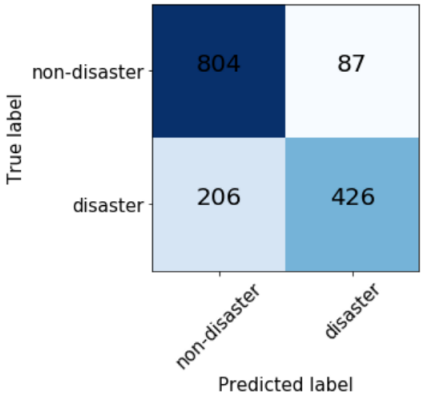

Figure 10: Confusion matrix Bernoulli Naive Bayes

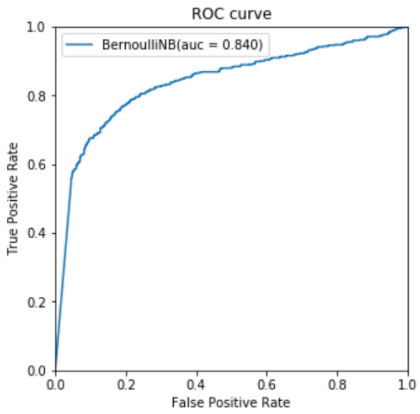

Figure 11: ROC curve Bernoulli Naive Bayes

Memory recurrent neural network (LSTM).

Recurrent Neural Networks are a type of neural network which contains cycles in which previous hidden layers are "remembered" by concatenation into later layers [15]. They are often used for sequence data. We chose to iterate the RNN though 10 epochs, after which it reached a loss of 0,0501 and accuracy of 0,9806 on the training set and an accuracy of 0,7774 on the validation set.

LSTM's are a subset of Recurrent Neural Networks that are particularly good for sequence data such as language models [11]. They use a series of input, output, and forget gates to regulate memory for each cell in the data. After 5 epochs, the LSTM model reached an accuracy of 0.9647 on the training set and 0.7551 on the validation set.

Finally, Bi-directional Recurrent Neural Networks are another type of RNN that are commonly used for sequential data. This model differs from ordinary RNN's because it combines two RNN's, one in which is fed in the usual way from the beginning of the sequence to the end and a second that runs the opposite way from the end to the beginning [17]. This gives the model a "look ahead" ability. The BRNN model achieved a f1 score of 0.722, with a precision of 0.753 and recall of 0.690.

3.2.6 Support Vector Machine

First developed in 1995, the Support Vector Machine classifier tries to maximize the margin between negative and positive points that lie closest to the decision boundary between classes [6]. It uses the kernel trick that allows expansion of the feature space and makes the linear models more powerful. We tested several kernels against the accuracy of the classifier on the validation set. Based on Table 5, the linear kernel produced the best result of 0.76.

3.2.7 K-Nearest Neighbours

K-Nearest Neighbours (KNN) is a lazy classifier as it does not learn a discriminative function from the training data but memorizes the training dataset instead. First described by Naomi Altman in 1992, the kNN classifier takes the k nearest points from the training data and assigns the class most prevalent among those points [2]. We chose the hyperparameter K, which is the number of neighbors to query in order to make a prediction, by the trial and error method and selected the model with the best performance. All values were odd in order to avoid confusion between two classes of data.

As shown in Table 6, high values of K produced a slightly worse performance compared to lower values of K. A K value of 3 outperformed the other values giving an accuracy of 0.830 for the training set and 0.746 for the validation set. This is because picking a low K tends to overfit the model while picking a high K increases bias and lowers variance.

4 Results

From all classes of models described above we chose the best candidate which showed the highest accuracy on the validation set. In particular, from the Gradient Boosting Classifiers we chose the model

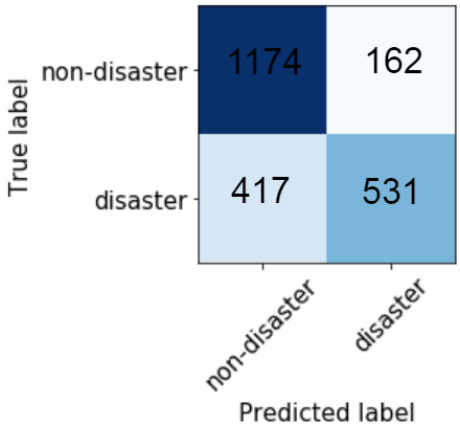

Figure 12: Confusion matrix KNN

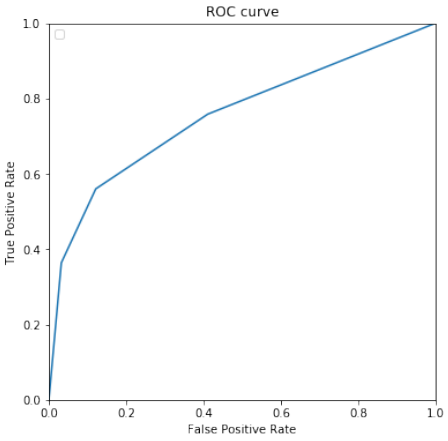

Figure 13: ROC curve KNN

with 100 features and a tree depth of 25. The Random Forest Classifier model with 80 features was chosen. Logistic regression with regularization parameter 1 was chosen by highest accuracy score on validation set. From the Naive Bayes set of classifiers, the Bernoulli classifier achieved the highest accuracy of 0.808. From the SVM class of classifiers the linear kernel demonstrated the best accuracy of 0.76 and therefore was selected for a final evaluation. For K-Nearest Neighbor, the model with 3 neighbors was chosen. For Recurrent Neural Networks, Bi-directional Neural Networks, and LSTM, the models using a sigmoid function for activation and Adam as an optimizer, with 5 epochs were chosen.

We evaluated all selected classifiers on the test set which consisted of 3263 instances and was withheld from the training set at the beginning of the project. In this way none of the models could see it before the final evaluation. In the final evaluation three metrics have been tested: the precision which shows the proportion of instances classified as positives that are actually positive; recall which shows the proportion of actual positives that have been found by the classifier; and f1 score which is the weighted average of the previous two.

We plotted ROC curves of different classifiers in order to recognize the most useful classifiers. In general, classifiers with a larger area under the curve have a higher probability of ranking an arbitrary pair of points correctly. The results can be seen in Figure 10 and Figure 11.

Model	F1 score	Precision	Recall
Gradient Boosting	0.725	0.784	0.675
SVM	0.726	0.751	0.703
Random Forest	0.730	0.797	0.674
Logistic Regression	0.740	0.820	0.680
LSTM	0.716	0.719	0.713
Bidirectional NN	0.722	0.753	0.690
Recurrent NN	0.716	0.693	0.741
Naive Bayes	0.733	0.828	0.657
KNN	0.638	0.762	0.549

Table 7: Evaluations of best models of its type on the test set

5 Discussion

Logistic Regression was our top performing classifier based on f1 score (0.740) and ROC curve (0.859). We believe this is due to the noisiness of the positive class in our dataset. The relatively low accuracy of our models overall can be explained by the somewhat complicated task of trying to identify heterogeneous types of disasters using a binary classifier. The tweets express different needs and responses depending on the type of disaster and this results in very different language used. Lumping all types of disasters into one category degrades the predictive ability of the models. Logistic Regression Classifiers use a maximum likelihood estimator combined with a sigmoid function. The least confident points near the boundary between classes

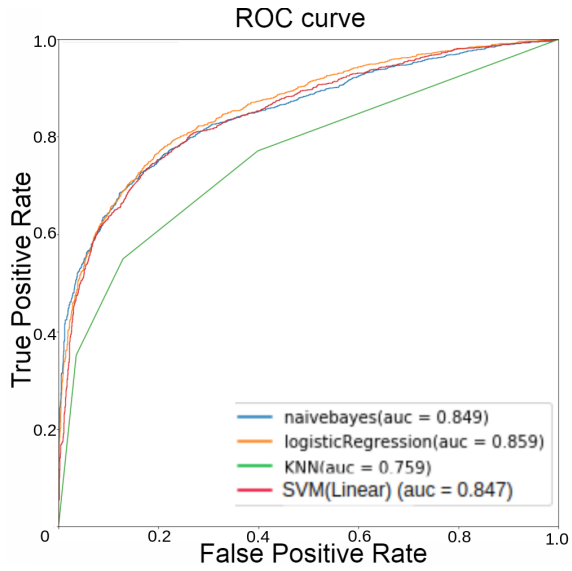

Figure 14: ROC curve results

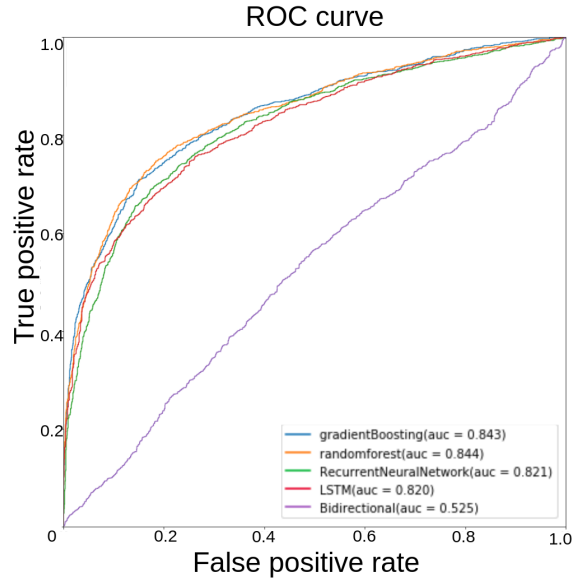

Figure 15: ROC curve results

are given more influence in the division than those with more confidence. In our case, this allows Logistic Regression to do a better job of differentiating between classes because they do not have a clear boundary.

Logistic Regression was followed by Naive Bayes (f1: 0.733, ROC: 0.849) and Random Forest (f1: 0.730, ROC: 0.844). However, in all three cases there is a bias from high precision scores (0.820, 0.828, and 0.797 respectively) while recall scores are relatively low (0.680, 0.657, and 0.674 respectively). In a disaster related situation, we determined that the cost of misidentifying disaster related tweets, associated with a higher recall, was higher than the cost of including some non-disaster related tweets, associated with higher precision, in the final classification. This is because it would be worse for first responders to miss early notification of a disaster situation or someone in need, while it is more acceptable for a few non-disaster related tweets to be included in the final results. The mixed results we obtained from our experiments were reflected in our background research which found no specific model better overall for identifying disaster related tweets.

If we prioritize recall, the best performing classifier was the Recurrent Neural Network with a recall of 0.741 and ROC of 0.821. LSTM was next with a recall of 0.713 and ROC of 0.821. This result makes sense because RNNs and their subset, LSTMs, commonly perform well on language models. The low precision of these classifiers could be explained because these models work by analyzing sequences of words and tweets have less of a dependence on the order of words due to the informal use of language and lack of formal sentence structure.

The Random Forest and Gradient Boosting classifiers also performed quite well as they are ensemble methods that combine several models which allow the production of better predictive performance compared to a single model. They resulted in F1 score values of 0.730 and 0.725 respectively.

The KNN classifier demonstrated the worst performance among all other classifiers. This phenomenon could be explained by two factors. First, the input given to models is high-dimensional since all words are converted to numeric representation and KNN does not deal efficiently with high-dimensional data. Second, since KNN treats highly predictive and noisy features equally, it is sensitive

to irrelevant features. Tweets contain many noisy words, and so this might influence the performance of this classifier dramatically.

For future research, we suggest two possible solutions for the problem of noisy class boundaries between positive and negative classes. First, we would separate out types of disaster events and make sure they are equally represented by keywords within the dataset. It makes sense that underrepresented classes will contain more errors. We might also consider a multi-class classifier which would allow us to see classification errors by disaster type and understand better why some are not as easy to identify as others. Secondly, we propose that adding information derived from the tweets beyond the tf-idf technique could improve classification. For example, links to official emergency websites would be a good indicator of a disaster related tweet.

6 Conclusion

For our research, we applied various machine learning models with the aim of retrieving disaster-related tweets from a predetermined pool of tweets. For this purpose we first modified all instances in the dataset with natural language processing techniques and replaced missing values of the "keyword" feature with real values. Second, we transformed all features into numeric representations and then fitted different types of models to the input. We used a validation set for hyperparameter tuning and choosing the best performing models of each type of classifier. Finally, we evaluated all of the best models on the test set. Our results showed that a Logistic Regression classifier was the top performing classifier for distinguishing between disaster and non-disaster related tweets, followed by Naive Bayes and Random Forest, while the KNN classifier demonstrated relatively poor results.

7 Bibliography

[1] Akilandeswari J. and Jothi G. (2018). Sentiment classification of tweets with non-language features. Procedia Computer Science, Vol 143: 426–433. DOI: 10.1016/j.procs.2018.10.414

[2] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 46(3): 175–185. DOI: 10.1080/00031305

[3] Ashktorab, Z.; Brown, C.; Nandi, M.; and Culotta, A. (2014). Tweedr: Mining twitter to inform disaster response. Proceedings of the 11th International ISCRAM Conference. DOI: 10.1.1.676.2784

[4] Bai, H. and Yu, G. A Weibo-based approach to disaster informatics: incidents monitor in post-disaster situation via Weibo text negative sentiment analysis. Natural Hazards, Vol 83:1177–1196. DOI 10.1007/s11069-016-2370-5

[5] Breiman, L. (1997). Arcing the edge. DOI: 10.1.1.48.4816

[6] Cortes, C. and Vapnik, V. N. (1995). Support-vector networks. Machine Learning, 20(3): 273–297. DOI: 10.1007/BF00994018.

[7] Friedman, J. H. (1999). Greedy function approximation: A gradient boosting machine. The Annals of Statistics, Vol 29(5): 1189-1232. DOI: 10.1.1.29.9093

[8] Gul, S.; Shah, T.; Ahad, M.; Mubashir, M.; Ahmad, S.; Gul, M. and Sheikh, S. (2017). Twitter sentiments related to natural calamities: Analysing tweets related to the Jammu and Kashmir floods of 2014. The Electronic Library, Vol 36(1): 38-54. DOI: 10.1108/EL-12-2015-0244

[9] Kabir, M.Y. and Madria, S. (2019). A deep learning approach for tweet classification and rescue scheduling for effective disaster management. SIGSPATIAL '19: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp 269–278. DOI: 10.1145/3347146.3359097

[10] Ho, T. K. (1995). Random decision forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16: 278–282. DOI: 10.5555/844379.844681

[11] Hochreiter, S. and Schmidhuber J. (1997). Long short-term memory. Neural Computation, 9(8): 1735–1780. DOI: 10.1162/neco.1997.9.8.1735

[12] Li, H.; Caragea, D.; Caragea, C.; and Herndon, N. (2017). Disaster response aided by tweet classification with a domain adaptation approach. Journal of Contingencies and Crisis Management, Vol 6(1): 16-27. DOI: 10.1111/1468-5973.12194

[13] Palshikar, G.K.; Apte, M.; and Pandita, D. (2018). Weakly supervised and online learning of word models for classification to detect disaster reporting tweets. Information Systems Frontiers, Vol 20: 949–959. DOI: 10.1007/s10796-018-9830-2

[14] Pekar, V.; Binner, J.; Najafi, H.; Hale, C.; and Schmidt, V. (2020). Early detection of heterogeneous disaster events using social media. Journal of the Association for Information Science and Technology, Vol 71(1): 43-54. DOI: 10.1002/asi.24208

[15] Rumelhart, D. E. (1985). Learning Internal Representations by Error Propagation. San Diego (CA): Institute for Cognitive Science, University of California. DOI: 10.5555/104279.104293

[16] Schulz, A.; Guckelsberger, C.; and Janssen, F. (2017). Semantic abstraction for generalization of tweet classification: An evaluation of incident-related tweets. Semantic Web, vol 8: 353–372. DOI: 10.3233/SW-150188

[17] Schuster, M., and Paliwal, K.K. (1997). Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11): 2673-2681.2 DOI: 10.1.1.331.9441

[18] Singh, J.P.; Dwivedi, Y.K.; Rana, N.P.; Kumar, A.; and Kapoor, K.K. (2019). Event classification and location prediction from tweets during disasters. Annals of Operation Research, Vol 283:737–757. DOI: 10.1007/s10479-017-2522-3

[19] Stavrianou, A.; Brun, C.; Silander, T.; and Roux, C. (2014). NLP-based feature extraction for automated tweet classification. DMNLP'14: Proceedings of the 1st International Conference on Interactions between Data Mining and Natural Language Processing. vol 1202: 145–146. DOI: 10.1109/ICSCN.2015.7219856

[20] Truong, B.; Caragea, C.; Squicciarini, A; and Tapia, A.H. (2014). Identifying valuable information from twitter during natural disasters. Proceedings of the American Society for Information Science and Technology, vol 51(1): 1-4. DOI: 10.1002/meet.2014.14505101162

[21] Getting started with Natural Language Processing - A general Introduction https://www.kaggle.com/philculliton/nlp-getting-started-tutorial

[22] Basic NLP with TensorFlow and WordCloud https://www.kaggle.com/marcovasquez/basic-nlp-with-tensorflow-and-wordcloud

[23] Prediction on Disaster tweets using LSTM https://www.kaggle.com/ryanchan911/prediction-on-disaster-tweets-using-lstm

[24] RNN for Disaster tweets https://www.kaggle.com/manjeetsingh/rnn-for-disaster-tweets

[25] NLP with Disaster using Neural Network https://www.kaggle.com/terminate9298/nlp-with-disaster-using-neural-network

8 Appendix

Features	Depth	LR	Accuracy train set	Accuracy validation set
20	depth 5	0.05	0.609	0.629
		0.075	0.636	0.650
		0.25	0.739	0.721
		0.5	0.767	0.734
		0.75	0.766	0.728
		1	0.769	0.718
70	depth 20	0.05	0.954	0.793
		0.075	0.976	0.790
		0.25	0.989	0.789
		0.5	0.989	0.782
		0.75	0.989	0.773
		1	0.988	0.763
100	depth 25	0.05	0.982	0.789
		0.075	0.989	0.797
		0.25	0.989	0.775
		0.5	0.989	0.777
		0.75	0.989	0.761
		1	0.989	0.753

Table 1: Hyperparameter tuning for gradient boosting classifier

Features	Accuracy train set	Accuracy validation set
50	0.936	0.789
65	0.989	0.790
80	0.989	0.791
100	0.989	0.786 8
200	0.984	0.790

Table 2: Hyperparameter tuning for Random Forest Classifier

C	Accuracy train set	Accuracy validation set
0.001	0.564	0.585
0.01	0.583	0.600
0.1	0.831	0.786
1	0.897	0.795
10	0.977	0.788
100	0.989	0.773
1000	0.989	0.768

Table 3: Hyperparameter tuning of Logistic Regression

NB Classifier	Accuracy train set	Accuracy validation set
Multinomial	0.849	0.791
Gaussian	0.929	0.758
Complement	0.850	0.790
Bernoulli	0.859	0.808

Table 4: Performance of different types of Naive Bayes classifiers

Kernel	Accuracy train set	Accuracy validation set
Linear	0.989	0.760
Polynomial	0.876	0.670
Gaussian	0.769	0.580
Sigmoid	0.770	0.580

Table 5: Performance of different types of SVM classifiers

K Value	Accuracy train set	Accuracy validation set
3	0.830	0.746
5	0.796	0.733
11	0.726	0.710
21	0.666	0.675
87	0.567	0.586

Table 6: Hyperparameter tuning for KNN classifier