Disaster Twitter Prediction Using Different Machine Learning and Deep Learning Techniques

1st Md. Rakibul Hasan Shezan
department of Computer Science and Engineering
East West University
Dhaka-1212, Bangladesh
2017-2-60-036@std.ewubd.edu

2nd Mohammed Nasif Zawad
department of Computer Science and Engineering
East West University
Dhaka-1212, Bangladesh
2017-2-65-001@std.ewubd.edu

3rd Yeasir Arafat Shahed
department of Computer Science and Engineering
East West University
Dhaka-1212, Bangladesh
2017-2-60-089@std.ewubd.edu

***Abstract*—Twitter has become one of the most frequently used media for both communication and getting daily information. Tweets can be a useful medium for everyone to be alert when a disaster occurs on a place. So, in this paper we used some state of the art machine learning and deep learning techniques with information retrieval technique to create models that can be used to classify disaster tweets from random tweets. We used a labeled tweeter dataset with 7613 disaster and random tweets. Logistic Regression with TF-IDF technique gave us the best result as it's accuracy was 80% in the dataset. The other techniques that we used were Naive Bayes, Random Forest and Long Short Term Memory(LSTM).**

***Index Terms*—Twitter Disaster, Machine Learning, Deep Learning**

I. Introduction

In this modern-day, communication has become so much easier and faster that it is just a matter of a couple of seconds to propagate news after the occurrence of it. Internet is the thing that has brought the whole world to a network and made this communication happen. Twitter is a place where any individual can post anything at any time and recently it has become one of the most frequently used medium for users to spread the news.

The ubiquitous of smartphones enables people to announce an emergency they're observing in real-time. Different type of disaster is occurring in someplace in the world may be in every second. So if someone can know about a disaster quickly then he or she can be aware of it and take proper actions or steps. As Twitter is where people share tweets about different information, making a machine learning model to classify disaster tweets can be a really helpful mechanism for the users.

A. Definitions

1) TF-IDF: It is an information retrieval technique that weighs a term's frequency (TF) and its inverse document frequency (IDF). In this work we focused on TF-IDF vectorizer amongst TF-IDF, CountVectorizer, HashVectorizer etc. as the frequency-based word embedder. TF stands for Term Frequency and IDF stands for Inverse Document Frequency. It is used in text mining as a weighting factor for features. The equation representing the TF-IDF weight of a term t fora particular document d (given the whole dataset D and the number of documents in the dataset N) is

$$tf - idf(t,d) = tf(t,d) \times idf(t,d)$$

$$here,$$

$$tf(t,d) = Frequency\ of\ t\ in\ d$$

$$idf(t,d) = \log(N/documents\ with\ term\ t)$$

TF is upweighted by the number of times a term occurs in an article. And IDF is down-weighted by the number of times a term occurs in the whole dataset/corpus. So TF-IDF assigns less significant values to words that generally occur in most documents such as is, are, be, to, on ... etc. We used the ngram_range = 1 meaning we took one word at a time and considered it as one term and our max features were 2000 as we took only the top 2000 words whose TF-IDF values are higher.

II. Related Works

A study [1] used Convolutional Neural Networks with 'bag of words' and n-grams techniques and applied to flooding events dataset. They got around 83% accuracy on their dataset by applying CNN.

Another study [2] proposed a domain adaptation approach that can learn classifiers from unlabelled target data as well as source labelled data. They used the Naive Bayes classifier and they found better results for the domain adaption classifiers. Their version used a self-training strategy with hard labels

instead of EM with soft-labels to identify disaster-related tweets.

Similar to our data preprocessing technique, one study [3] used TF-IDF vectorization to convert text data into feature vectors. They applied and compared eight machine learning models to analyse the public sentiment on Twitter.

Sequence models is a very popular technique to find the relation and importance of different text structures for sentiment analysis. One study [4] used one of the most popular deep learning technique- Long Short Term Memory (LSTM) to identify hashtags in tweets. Their model gave 92.22% accuracy in their dataset.

As different study used different techniques, our study adds up another concept of using TF-IDF vectorization techniques rather than word2vec for creating feature vectors and applying on both machine learning and deep learning algorithms on a small twitter dataset.

III. Proposed Method

First of all, we cleaned the dataset to remove unnecessary words and symbols. Then we applied TF-IDF algorithm to extract feature verctors from the given cleaned text corpus. FInally we applied some machine learning techniques: Logistic Regression, Naive Bayes and Random Forest to create models. We also applied one sequence model which is LSTM to see the performance.

A. Dataset

We used a labeled disaster tweets dataset that is publically available. The dataset contains 7503 unique tweets and the outputs are 1 that indicates that the tweet is a disaster tweet and 0 to indicate it's not.

TABLE I
TWITTER DATASET

Dataset Statistics	Size
Training Set Size	6090
Test Set Size	1523

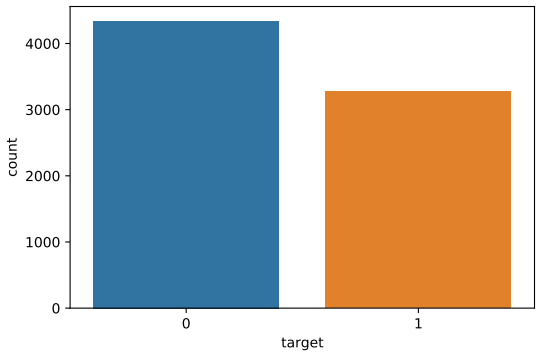

Fig. 1. Number of tweets based on the target value(0: Random Tweets, 1: Disaster Tweets)

B. Text Preprocessing

The collected dataset is mixed with some noisy and unnecessary data. So, we had to get rid of them through a bit of preprocessing. Our preprocessing consisted of the following steps:

1) Ignoring stopwords and punctuations: We removed the stopwords from each of the tweets. Stopwords are English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. (eg: "the", "he", "have").

2) Replacing Emojis and Urls: The tweets had many emojis and urls in it's body. So we had to remove them to improve the performance of the model. We created a dictionary containing all the emojis and their textual meaning like: :) stands for smile, :(means sad etc and then used this dictionary to replace all the emojis in our dataset. We replaced: "Thank You :)" with "Thank You EMOJI smile".

C. Corpus to TF-IDF Vectors

After cleaning the whole dataset by removing the stopwords, urls, removing emojis and making lower case texts, we used this cleaned corpus of texts to calculate the term frequency-inverse document frequency of the top 2000 words. We implemented Tfidfvectorizer to calculate the TF-IDF values of these words and created a ndarray with the word vectors. We used the unigram concept and considered each word as one term.

D. Structure of the LSTM model

- Embedding layer with the max input features of 2000 and the embedding dimension is 128.
- One hidden LSTM Layer with 128 units.
- Single output dense layer with the sigmoid activation function.
- Optimizer used is 'adam' and the loss is calculated with binary crossentropy .

IV. Results and Analysis

The dataset was randomly split to two parts. 80% of the data were used as training dataset. The rest 20% were used to test the performance of our model. We got different accuracy for the test sets using different machine learning techniques.

A. Result using Logistic Regression

Logistic regression turned out to give the best accuracy result among all the other models. We got 80% accuracy in our test model by using Logistic Regression model.

We can look at the classification report Tab 2. of this model after applying the logistic regression.The f1-scores for both of the labels are good as it is almost close to 1.F1-score combines precision and recall relative to a specific positive class. The F1-score can be interpreted as a weighted average of the precision and recall, where an F1-score reaches its best value at 1 and worst at 0.

The confusion matrix for logistic regression in Figure 2, shows that it predicted 788 values from test set correctly 0 and

TABLE II
CLASSIFICATION REPORT OF LOGISTIC REGRESSION

Label	Report		
	precision	***recall***	***f1-score***
0	0.79	0.89	0.84
1	0.81	0.68	0.74

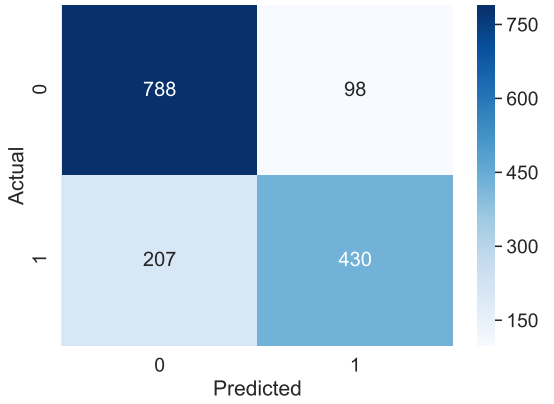

Fig. 2. Confusion matrix for Logistic Regression

predicted 430 values correctly as 1. The confusion matrix, is a table with two dimensions ("Actual" and "Predicted"), and sets of "classes" in both dimensions.

B. Result using Naive Bayes

The naive bayes classification technique also gave a good accuracy value as it was the 2nd best after the logistic regression model. The accuracy for Naive Bayes Classifier was 78.1%.

TABLE III
CLASSIFICATION REPORT OF NAIVE BAYES CLASSIFIER

Label	Report		
	precision	***recall***	***f1-score***
0	0.78	0.88	0.82
1	0.79	0.65	0.71

C. Result using Random Forest

We lastly applied the random forest classifier from ml algorithms as it is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier. It runs efficiently on large databases. We got 77% accuracy on our dataset by applying the Random Forest classifier.

D. Result using LSTM

We applied one of the algorithms from deep learning to see how it performs on our dataset. We used sequence model in our case as we are dealing with text data. We applied long short term memory(LSTM) because it can explicitly remember the words special features to decide the relation of it with the other words in a sentence.

We used the tf-idf word vectors and used it in the embedding layer of the LSTM. The accuracy of our model in this case was moderate as it gave 72.7% accuracy.

TABLE IV
CLASSIFICATION REPORT OF RANDOM FOREST CLASSIFIER

Label	Report		
	precision	***recall***	***f1-score***
0	0.78	0.85	0.81
1	0.76	0.66	0.70

TABLE V
ACCURACY TABLE

Algorithms	Accuracy
Logistic Regression	80%
Naive Bayes	78.1%
Random Forest	77%
Long Short Term Memory(LSTM)	72.7%

V. DISCUSSION AND FUTURE WORK

We applied our state-of-the model machine learning and deep learning algorithms to a Twitter dataset. As we know someone cannot create a post with more than 280 words, so each tweet was pretty small. Many of them were too small and filled with noisy values(like- URLs, emojis, special characters). When we were working with the lstm model, we tried to create our embedding layer but giving an embedded matrix created with the word2vec feature. But the dataset was not that big and many tuples with irrelevant words, the embedding vectors were not worth using and it was not able to create a good relationship with having similar words with closer values. Whereas our TF-IDF technique gave a good combination of vectors that we fed into the input layer of LSTM as well as to the other models.

In future, if we can use a large dataset for these models consist of thousands of tweets, we might be able to create a much more accurate and efficient model that can be used to detect disaster tweets automatically.

REFERENCES

[1] C. Caragea, A. Silvescu, and A. H. Tapia, "Identifying informative messages in disaster events using convolutional neural networks," in *International Conference on Information Systems for Crisis Response and Management*, pp. 137–147, 2016.

[2] H. Li, D. Caragea, C. Caragea, and N. Herndon, "Disaster response aided by tweet classification with a domain adaptation approach," *Journal of Contingencies and Crisis Management*, vol. 26, no. 1, pp. 16–27, 2018.

[3] L. Meng, Z. S. Dong, L. Christenson, and L. Fulton, "Mining public opinion on twitter about natural disaster response using machine learning techniques," *arXiv preprint arXiv:2005.07019*, 2020.

[4] J. R. Chowdhury, C. Caragea, and D. Caragea, "On identifying hashtags in disaster twitter data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 498–506, 2020.