Disaster Tweet Analysis

Himanshu Tiwari[1] and Gourav Beura[2]

[1]Khoury College of Computer Sciences, Northeastern University

December 15, 2021

Abstract

The current innovations are revolutionizing the entire world. The Internet has become a daily necessity for everyone as its being utilized in every field all over the world. With the swift increase in social network applications, people are utilizing these platforms to express their thoughts and opinions with regard to regular day issues or a sudden crisis. Analyzing and collecting this information has become crucial. An important use case could be classification of the tweets, that can help in case of emergency and calamity. In our research, we aim to explore tweets related to disaster. We employ natural language processing to predict whether a text is referring to a natural disaster or not. We are using dataset from Kaggle with nearly 10000 data points to train and test our models. We have used traditional Machine learning models and deep leaning techniques to perform classification on the dataset. Through the research work, we are comparing performance of different classification methods. The results concludes that deep learning architectures have accomplished high metrics scores as compared to traditional machine learning models.

1 Introduction

Twitter was launched as a messaging service which allowed its users to exchange messages by forming small groups. Its first prototype was released on 15th July, 2006. It briskly gained popularity around the year 2007 and evolved into a micro-blogging and social networking portal which enables users to post small write-ups, reviews and messages popularly known as “tweets”. The site provides features for sending, liking and repeating tweets to its registered users, while unregistered users can only view tweets.

In 2010, Twitter revamped its entire user experience by adding support to share pictures and videos in tweets. As of October, 2019, around 6000 tweets are produced every second on an average globally. This figure is 350,000 per minute or 500 million tweets per day or 200 billion tweets per year. This produces and generates millions of petabytes (210 terabytes) of data worldwide.

As there is no prescribed format for the tweets, this makes the data unstructured and challenging for analysis. Every post is rich in content as it includes essays, poems, data and information. In order to pull out the useful data from these text content we use Natural Language Processing principles. The tweets are cleaned and tokenized into words. Then analysis is performed on the tokenized text data using several machine learning techniques.

On a day to day basis these microblogs are commonly used to share information, express general public opinion during any event such as natural disasters, movie promotion, news, sports matches, political elections etc.In recent times, the social media has played a crucial role to educate its users about event of a natural disaster which usually occurs without any warning. Therefore, it has gained a lot of attention as an additional medium for crisis communication apart from news broadcast on TV and Radio. Twitter has become the most widely used mode of communication in times of emergencies and disasters. Multiple data science oriented companies are investing heavily on monitoring such information and designing solutions analyze twitter data. Specifically, news agencies and disaster relief organizations are trying to analyze tweets in real time to identify the occurrence of disasters from tweets. Agencies can understand the trend and make predictions about the impact these disaster which would help millions of take precautionary measures before hand. Not just the users and common people, Government agencies can proactively initiate evacuation process before the situation goes beyond control.

The data used for the analysis is from Kaggle which has a total of 7613 tweets with labels 0 and 1 standing for "Disaster Tweets" and "Not Disaster Tweets" respectively. These data points are used for training the model. There is a test set with 3263 unlabeled tweets to evaluate the performance of the model. For validation of the model while training, we have split the tweets in Trainset and Validation-set with 80:20 ratio. The total training cases for each class are almost balanced with 57:43 ratio of class 0:1 respectively.

The basic objective is to classify the tweets into 2 categories, belonging to "disaster related tweets" or otherwise.

We have used traditional machine learning algorithms such as naive bayes, logistic regression, random forest and support vector machine and compared the results with advanced deep learning methods such as Long Short-Term Memory, Recurrent Neural Networks, Gated Recurrent Unit, Bidirectional LSTM and Bidirectional GRU. The results metrics include Accuracy across both the classes, Precision, Recall and F1 score.

Preprocessing the test data and accurately representing it as a multidimensional vector is a challenging task for traditional Machine learning algorithms. The performance of the model varies based on the quality of the preprocessed data. Therefore, we have used Stemming, lemmatization, Punctuation removal and Stop word removal to extract valuable information from the tweets. We have experimented with 4 different vectorization techniques such as Count Vectorizer, TF-IDF, Word2Vec (pretrained) and Glove (pretrained) to come up with the best metric results for each model. For Deep learning models, embedding/vectorization can be learnt as part of the architecture but requires large dataset to train the model as there are close to 10000 weights and parameters to train and it keeps increasing with more complicated architecture. Hence, the data size is a big issue for deep learning models. To overcome this issue, we are not training the embedding and are using Glove pretrained vector representations for each word in the corpus.

2 Background

In this section, we very briefly study different blocks that were used to build our models in this paper.

2.1 Traditional Machine Learning Models

- **Logistic Regression**
  Logistic regression is a statistical model which tries to find best line that will accurately split data into two classes of target. The target variable is called response and there one or more independent variables known as predictor variables.

  It uses are sigmoid function to map the predicted values to probabilities.

$$Logit(P(x)) = w_1x_1 + w_2x_2 + w_3x_3$$

$$Logit((P(y = 1|x)) = \frac{1}{1 - e^{-(w^t x)}}$$

- **Naive Bayes**
  Naïve Bayes is a traditional machine learning model used for classification of data into different classes/labels. It is a probabilistic model which uses Bayes Theorem on strongly independent events.

  As per Bayes theorem,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|A')P(A')}$$

where A: Hypothesis Variable, B: Evidence Variable When we are provided independent evidence variables(features) $X = (x_1, x_2, x_3, ..., x_n)$, and we have to predict the probability of response y.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

then

$$P(y|x_1, x_2, .., x_n) = \frac{P(y)P(x_1|y)P(x_2|y)..P(x_n|y)}{P(x_1, x_2, ..., x_n)}$$

- **Support Vector Machines(SVM)**
  SVM is a pattern classification technique based on statistical learning theory. It finds a hyper-plane in the N-dimensional space to classify the required data points, N is the number of given features. SVM classifies the data into two classes by picking a hyperplane from set of hyperplanes using different kernels. The aim is to maximize distance between data points of both classes and the hyperplane.

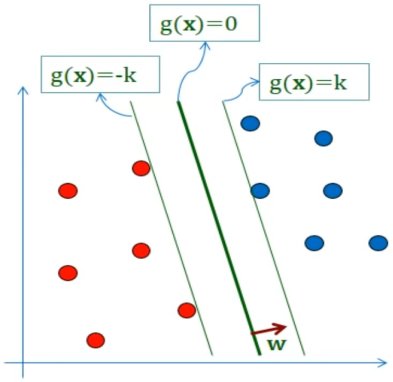


Figure 1: Graphical representation of SVM

- **Random Forest Classifier**
  Random Forest Classifier is a classification technique that creates multiple decision trees of dataset and takes average of result of the sub-samples to classify the data. This technique has better accuracy and controls overfitting of data. Each tree will give an output prediction and the class with highest number of votes is considered as prediction result. The critical part is maintain the minimal correlation between the trees.

2.2 Deep Learning Models

- **Recurrent Neural Network (RNN)**
  Traditional neural network architecture lacks the ability to store the information computed and use it to update the training weights. Recurrent neural networks address this issue. Each neuron in an RNN loops back to itself to persist the information.

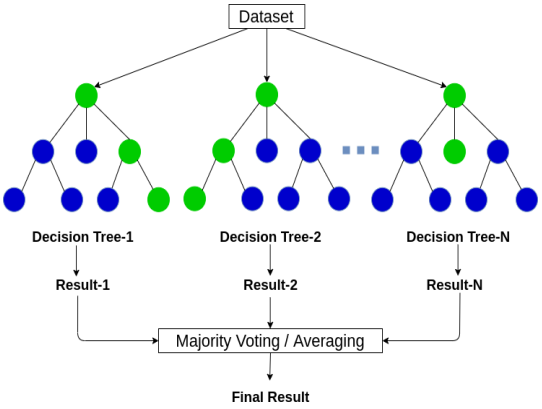


Figure 2: Random Forest Classifier Example

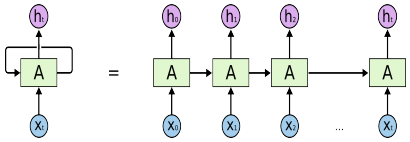


Figure 3: Recurrent Neural Network cell

The figure 3 represents a RNN neuron with x_t as an input and h_t as an output. A loop allows the information calculated so far to be utilized as a memory from previous steps. It allows information to flow from one step of the network to next. A RNN can be thought as a multiple copy of the same neuron passing information to its successor. The RNN forward pass is represented by the following equations:

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\ \hat{\boldsymbol{y}}^{(t)} &= \text{softmax}(\boldsymbol{o}^{(t)}) \end{aligned}$$

Figure 4: RNN forward pass equations

- **Long Short Term Memory (LSTM)**
  Simple RNN's have a drawback when it comes to "long – term dependencies" in the input data set. When the sequence of time steps is too large to capture, RNN's don't seem to be able to learn them. Hence, we use Long Short Term Memory (LSTM) networks which are a special kind of RNN that are capable of capturing the long-term dependencies. LSTM's have the same chain like structure as well but with different repeating neuron architecture.
  Cell states and various available gates are the core concept of LSTMs. They have the ability to remove or add information to the cell states with the help of regulated structures called gates. LSTM has 3 gates to control the cell states. The first gate is called Forget gate layer which works on what information we need to through away from the cell state. The second gate is called Input gate layer which works on what information new information we are going to store in the cell state. The third gate is called output gate layer which decides on what we are going to output. The LSTM forward pass is represented by the following equations:

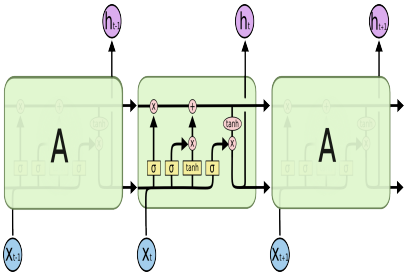
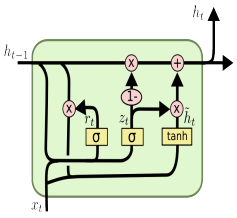


Figure 5: Long Short Term Memory model

$$\begin{aligned} f_t &= \sigma_g\left(W_f \times x_t + U_f \times h_{t-1} + b_f\right) \\ i_t &= \sigma_g\left(W_i \times x_t + U_i \times h_{t-1} + b_i\right) \\ o_t &= \sigma_g\left(W_o \times x_t + U_o \times h_{t-1} + b_o\right) \\ c'_t &= \sigma_c\left(W_c \times x_t + U_c \times h_{t-1} + b_c\right) \\ c_t &= f_t \cdot c_{t-1} + i_t \cdot c'_t \\ h_t &= o_t \cdot \sigma_c(c_t) \end{aligned}$$

Figure 6: LSTM forward pass equations

- **Gated Recurrent Unit (GRU)**
  Gated Recurrent Unit is a variition of LSTM network with the forget gate and input date combined together into a single update gate. It merges the cell states and hidden states resulting in a simpler model than LSTM. The GRU achitecture and forward pass equation is is as follows:



$$\begin{aligned} z_t &= \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \\ r_t &= \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \\ \tilde{h}_t &= \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \end{aligned}$$

Figure 7: Gated Recurrent Unit and Forward pass Equations

- **Bidirectional LSTM**
  Bidirectional LSTM consists of 2 LSTM layers with one layer processing the input sequence in forward direction and another layer in backward direction. This results in increase in amount of information

available to the network, improving the context available to the algorithm.

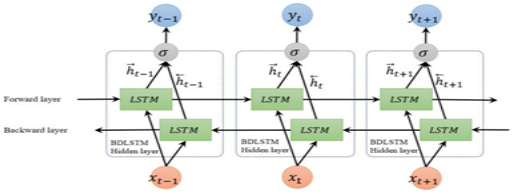


Figure 8: Bidirectional LSTM

- **Bidirectional GRU**
  Similar to Bidirectional LSTM, Bidirectional GRU consists of 2 GRU layers with one layer processing the input sequence in forward direction and another layer in backward direction. This results in increase in amount of information available to the network, improving the context available to the algorithm.

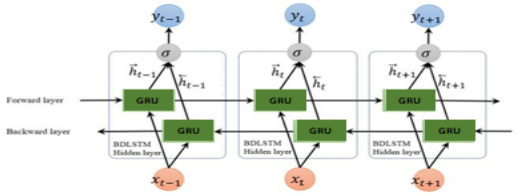


Figure 9: Bidirectional GRU

3 Related Work

There have been several significant and remarkable research work in the field of text analysis using Natural language processing when it comes to twitter. The authors Sana Moin, Ali Hasan, Shahaboddin Shamshirband, and Ahmad Karim had collected tweets that are related to certain political topics emerging in Pakistan and performed sentiment analysis of the tweets using Naive Bayes and SVM classifier (Hasan A , Moin S, Karim A , Shamshirband S , 2018) [1]. A predictive model for sentiment analysis of twitter (Bagheri H , Islam Md J, 2017)[2] has been doveloped by authors Md Johirul Islam and Hamid Bagheri. They used NLTK for Natural language Processing and Textblob python library for text processing. They classified tweets based on politics, movies, fake news, justice, fashion, humanity and justice into 3 categories of tweets – neutral, positive and negative.

4 Method

Given a tweet, along with associated attributes, we want to classify the data. If this analysis had to be done manually by a human, it would be highly time consuming. Therefore, using the traditional classification models we can utilize the feature attributes to get predictions quickly with better accuracy.

Dataset

We use the Kaggle Dataset of Disaster tweets for developing our model. This dataset contains both train and test data:
In train dataset, we have 7613 samples with following features:

- **ID**: A unique identifier for each tweet.
- **Keyword**: identified keyword related to disaster
- **Location**: the location of post
- **Text**: the tweet content
- **Target**: Represents tweet is Disaster (1) or non-disaster tweet (0)

The dataset has the following description: In Test dataset,

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7613 entries, 0 to 7612
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        7613 non-null   int64
 1   keyword   7552 non-null   object
 2   location  5080 non-null   object
 3   text      7613 non-null   object
 4   target    7613 non-null   int64
dtypes: int64(2), object(3)
memory usage: 297.5+ KB
```

Figure 10: Dataset Information

we have 3263 samples with same features except the target attribute.The Distribution of train data is shown below:

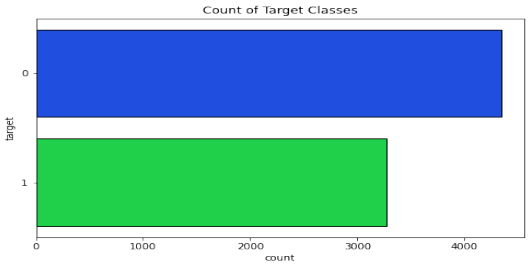


Figure 11: Analysis of target classes

From the above plot, we can clearly see that samples in Target Class 1 (disaster tweet) are around 3100 while

in Target Class 0(non-disaster tweet), it is approximately 4500. Also, the classes seem to be in balanced state.

Prior to running the ML models, we analyze the provided data to understand the relationship between attributes. We get insights about data by figuring out the distribution of words in text attribute, checking for null data, average word length in tweets, distribution over location, etc.
Analysis of training dataset:

- Analyzing average number of words in text

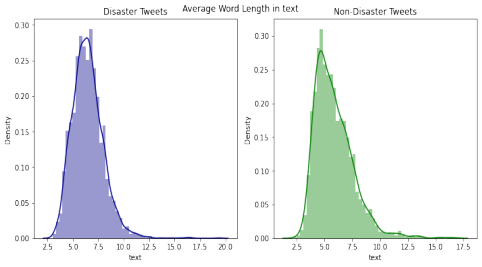


Figure 12: average number of words in tweets

From the above distributions, we can observe that the average word count is between 7 to 8 words for disaster tweets whereas for non-disaster tweets it ranges between 4.5 to 5 words.

- Analyzing Null Values in Data set

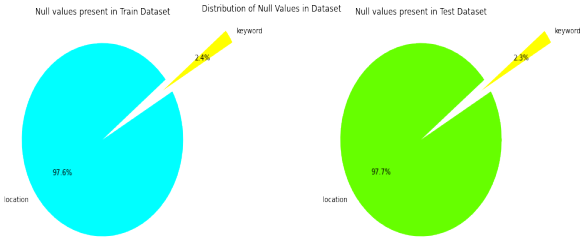


Figure 13: Missing content distribution in dataset

From the distribution, we observed that attributes Keyword and Location contains null values. For training data, 97.6% of "location" data are missing and 24% of "keywords" are missing. For testing data, 97.7% of "location" data is missing and 23% of keywords are missing.

- Analyzing Top Disaster Keywords
From the bar chart, we can observe that the most occurring keywords are derailment, wreckage, outbreak while the least occurring are sandstorm and evacuation.

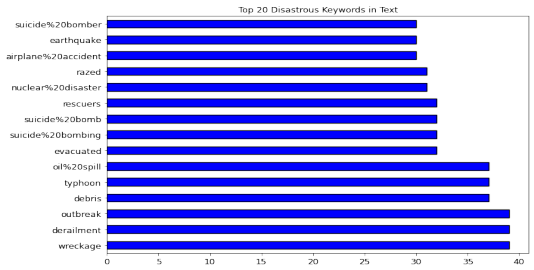


Figure 14: Top 20 Disastrous keywords

5 Project Description

The project is broken down into four parts which are as follows:

- Data Exploration
- Corpus Generation and Vectorization
- Training Model
- Result Metrics Generation

5.1 Data Exploration

In order to classify the text content into particular classes, we need to understand the relationship datapoints available in dataset, so that we can feed correct data to the models to get results with proper accuracy.

The moment we start exploring this data, we started by investigating the structure of it. Most eminent observation was that there were plenty of patterns to differentiate the tweets into two categories. Many tweets labelled as a disaster contained words indicating some type of serious accident, weather, or criminal activity, riots such as “killed.”, “crash”, “storm”, and "fire". The non-disaster tweets were comprised of words and phrases such as “love”,“one”,"go", etc. We can’t rely just on word frequency alone, so we applied different feature engineering methods to the all texts and found the exact same trends. We also decided to generate a couple of word clouds to get a better view of the word frequencies in our data.

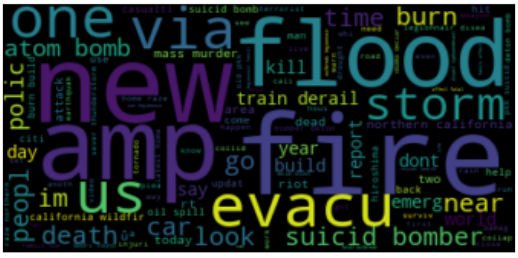


Figure 15: Word Map of Disaster Tweets

Figure 16: Word Map of Non-Disaster Tweets

The word clouds represent the frequencies of the words for disasters(Fig. 15) and non-disasters(Fig. 16) in our dataset. They give us a view of what our dataset contains.

On further exploration we find out the following results:

- Frequency of Stopwords in dataset:

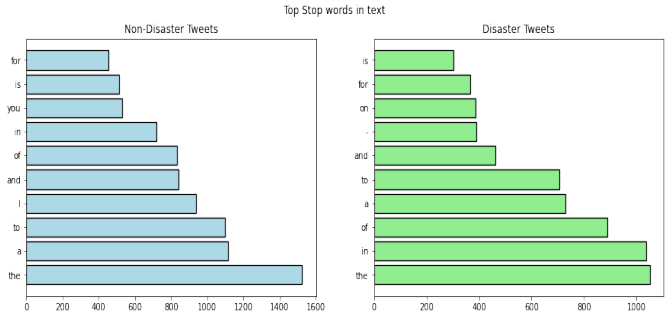


Figure 17: Frequent Stopwords

- Punctuations used in dataset:

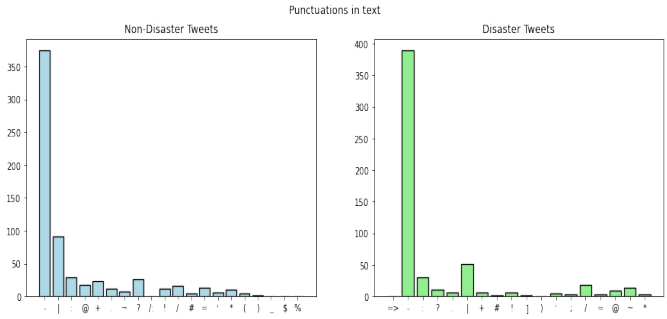


Figure 18: Frequency Distribution of punctuation in text

As the dataset has a a lot of garbage texts and characters, therefore, we need to remove those. The verbose data comprises URL's, html tags, emojis and miscellaneous text. We remove words of length of length less than 2 and performing Stemming and lemmatization.

5.2 Corpus Generation and Vectorization

The processed tweets are represented as a multidimensional vector / embedding. The embeddings are the source of information for a machine learning model to learn the weights and parameters to predict the desirable results. We have experimented with 4 word embedding methods which are Count Vectorizer, TF-IDF, Word2Vec (pretrained) and Glove (pretrained). Count Vectorization is a method of converting text data into numerical values based on the number of occurrences of each word in the document. TF-IDF stands for Term Frequency – Inverse Document Frequency, is a numerical statistic that reflects how important a word is to the document based on the probability distribution of the word occurring in a document by the probability distribution of the word occurring across all the documents. Word2Vec and Glove uses neural network models to learn the embeddings for each word from a large corpus of text such that words that share common contexts in the corpus are located close to one another in the space.

5.3 Training Models

Once the data has been pre-processed and vectorized we start training different ML models to classify the tweets. Based on architecture and performance we split our implementation into two categories i.e. traditional machine learning methods and deep learning methods.

Traditional Machine Learning Models

1. **Logistic Regression**:

   Logistic regression produces feature weights to make decision. We have chosen the default *lbfgs* solver for our implementation, as it picks the best parameters with least error while predicting the output and saves a lot of memory.

2. **Naive Bayes**:

   We chose multinomial naive bayes classifier as it has good performance for this type of text classification.

3. **Support Vector Machine**:

   We decided use SVM classifier because it works well in non-linear problem and tries to find a hyperplane that splits the data accurately in one of the two categories.

   As the encoded data is large sparse vector, we trained the model with polynomial, Gaussian Radial Basis Function (RBF) and Sigmoid kernel and default value of gamma parameter. Based on accuracy of model with different kernels, we picked the result obtained from rbf kernel.

4. **Random Forest**:

   For Random Forest, we generate the results by training model with the number of decision tree in range 1 to 100, and max_feature as ["sqrt", "log2"].

Deep Learning Models

1. **Simple RNN**:

   The architecture of the Recurrent Neural Network is described in figure 20. We used an embedding layer with weight matrix assigned as glove pretrained vectors followed by RNN layer with output dimension of 64. Further, we passed the results though a dense layer to output single value of prediction representing probability of the input text to be a disaster tweet. We have used Binary-Crossentropy to compute loss for the architecture which compares each of the predicted probabilities to actual class output which can be either 0 or 1.

$$\text{Loss} = -\frac{1}{\underset{\text{size}}{\text{output}}} \sum_{i=1}^{\underset{\text{size}}{\text{output}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Figure 19: Binary Cross-Entropy Loss

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 40, 100)           1655600

 simple_rnn (SimpleRNN)      (None, 64)                10560

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 1,666,225
Trainable params: 10,625
Non-trainable params: 1,655,600
_________________________________________________________________
```

Figure 20: RNN Model Architecture

2. **LSTM**:

   The architecture of the Long Short Term Memory is described in figure 21. It is similar to RNN architecture.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 40, 100)           1655600

 lstm_1 (LSTM)               (None, 64)                42240

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 1,697,905
Trainable params: 42,305
Non-trainable params: 1,655,600
_________________________________________________________________
```

Figure 21: LSTM Model Architecture

3. **Bidirectional LSTM**:

   The architecture of the Bidirectional LSTM is described in figure 22. We used an embedding layer with weight matrix assigned as glove pretrained vectors followed by Bidirectional LSTM layer with output dimension of 128. Further, we passed the results though a dense layer to output single value of prediction representing probability of the input text to be a disaster tweet. We have used Binary-Crossentropy to compute loss for the architecture.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 40, 100)           1655600

 bidirectional (Bidirectiona  (None, 128)              84480
 l)

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 1,740,209
Trainable params: 84,609
Non-trainable params: 1,655,600
_________________________________________________________________
```

Figure 22: Biidirectional LSTM Model Architecture

4. **Bidirectional GRU**:

   The architecture of the Bidirectional GRU is described in figure 23. It is similar to Bidirectional LSTM Architecture.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 40, 100)           1655600

 bidirectional (Bidirectiona  (None, 128)              63744
 l)

 dense (Dense)               (None, 1)                 129

=================================================================
Total params: 1,719,473
Trainable params: 63,873
Non-trainable params: 1,655,600
_________________________________________________________________
```

Figure 23: Bidirectional GRU Model Architecture

5.4 Result Metric Generation

We validate the results using metrics such as Accuracy, Precision, Recall and F1 Score. We calculate these values as follows:

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

TP: predicted a positive and that's true.

FP predicted a positive, and that's false.

FN predicted a negative, and that's false.

F_1 is calculated as follows:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

$$Accuracy = \frac{TP + TN}{Total} \tag{4}$$

F_1 has been widely used in NLP literature and will provide us with the most accurate comparison for the above experiments.

6 Experiment & Results

We discuss the results for our experiments in this section. The training graph for each deep learning model is plotted in the figures(24-27) for roughly 8-15 epochs. We observe that the validation accuracy increases with increase in the complexity of the deep learning architecture from Simple RNN to Bidirectional architectures as the number of trainable parameters increases. The next plots(shown in fig. 28-31) compares the result metrics for Traditional Machine learning models and deep learning models in terms of Accuracy, Precision, Recall and F1 Score. We can clearly observe that the results are in favour of Deep learning models. Based on the results, bidirectional GRU has performed the best in terms of Accuracy, Recall and F1 score. Bidirectional LSTM has performed the best in terms of Precision.

If we only observe the results for traditional machine learning models, random forest classifier has outperforms SVM, Naive Bayes and Logistic Regression models in all metrics because it uses decision trees to calculate the result whereas the other algorithms use probabilistic and hyperplane based classification approach which are slightly inefficient in capturing the essence of the sequential data points.

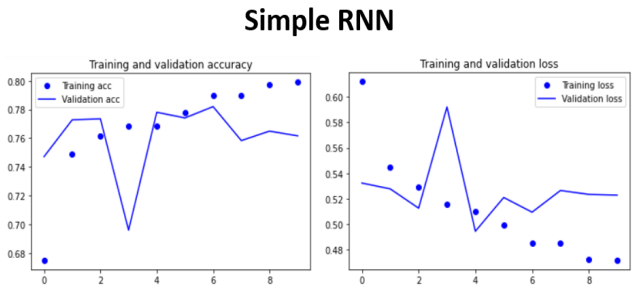


Figure 24: RNN accuracy and loss for train and validation set

Conclusions

In this Project, we classified tweets as "Disaster Tweets" or "Not a Disaster Tweets" by experimenting with multiple classification models. We observed that Deep learning

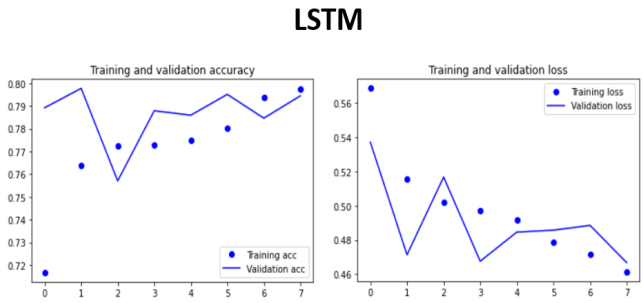


Figure 25: LSTM accuracy and loss for train and validation set

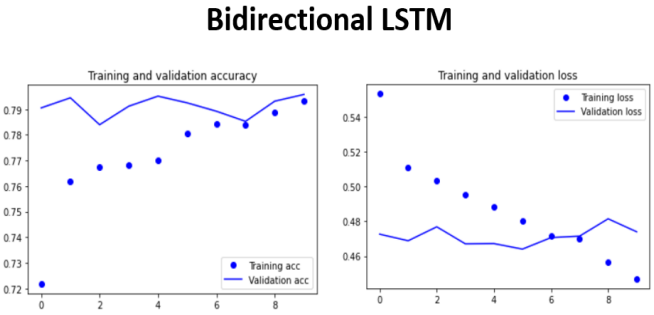


Figure 26: Bidirectional LSTM accuracy and loss for train and validation set

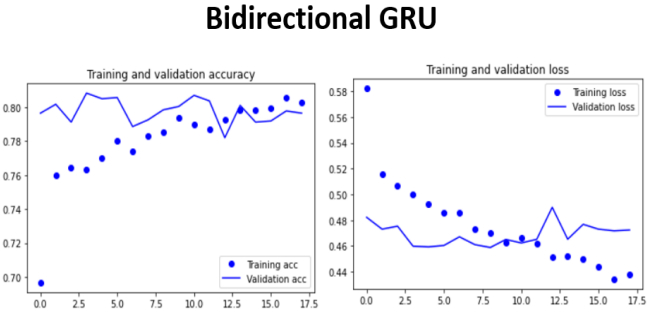


Figure 27: Bidirectional GRU accuracy and loss for train and validation set

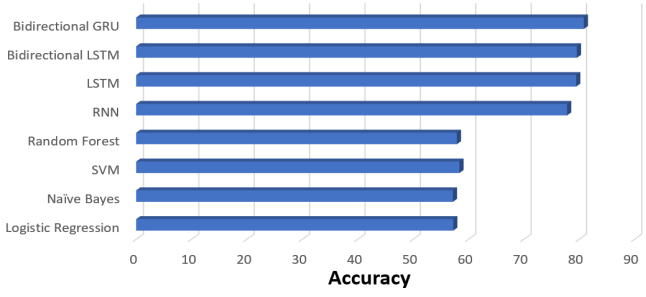


Figure 28: Accuracy Comparison of all models

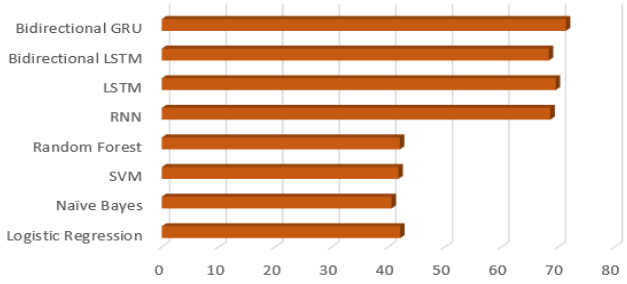


Figure 29: F1 Score Comparison of all models

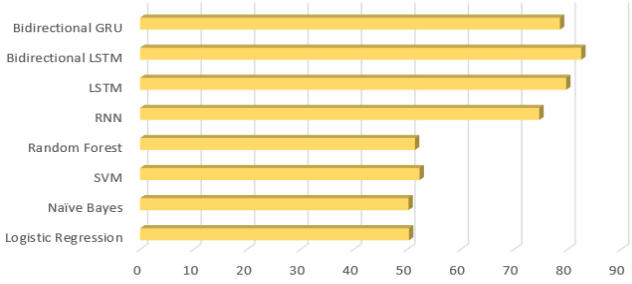


Figure 30: Precision Score Comparison of all models

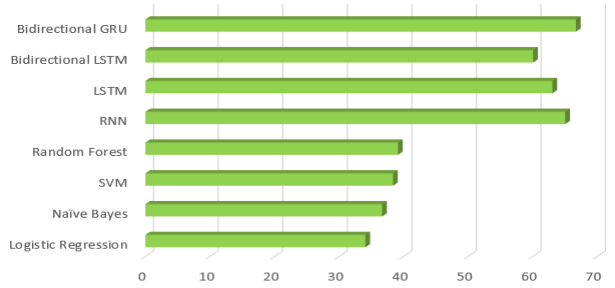


Figure 31: Recall Score Comparison of all models

models were able to achieve higher results in terms of Accuracy, Precision, Recall and F1-Score. The traditional models didn't perform up to the mark, because the data is sequential. As encoding methods generate huge sparse vectors which has arbitrary representation of data therefore does not provide useful information to the model regarding the interactions or similarities that may exist between individual symbols. As a result it gets difficult for model because it won't be able to leverage much of what it has learned about word x when it is processing the data regarding word y. Where as the Deep learning models have advance gated architecture to preserve the observations made so far in the sequence as a memory and pass it on to the next time steps to accurately predict the desired results. Although the results are still not upto the mark as the size of the data is too small to train deep learning models. The number of trainable parameters are more the 80k for bidirectional LSTM which is 10 times the training dataset available. Based on the complexity of the problem, if we had more labeled dataset, the metric scores of the model would be much higher than the current obtained results. Therefore, to conclude we have observed that Deep learning models perform better as compared to traditional machine learning models for sequential data and the accuracy scores can improve provided additional data points. We tested our best performing model on the kaggle test set which achieved the accuracy of 78.4%.

Acknowledgements

We would like to thank Professor Chris Amato for providing valuable suggestions and feedback.

References

[1] Bagheri H , Islam Md J (2017). *Sentiment analysis of twitter data.*Computer Science Department Iowa State University,United States of America.

[2] Hasan A , Moin S, Karim A , Shamshirband S (2018). *Machine learning based sentiment analysis for twitter accounts.* Mathematical and Computational Applications.

[3] Kristopher Flint & Zachary Kellerman(2020) *Real or Not? NLP with Disaster Tweets*,Tennessee Tech

[4] Shriya Goswami & Debaditya Raychaudhuri(2020) *Identification of Disaster-related tweets using Natural Language Processing* ,Computer Science and Engineering , Techno India University

[5] Bharti O, Malhotra M (2016).*Sentiment analysis on twitter data.* International Journal of Computer Science and Mobile Computing, 5(6): 601 – 609.

[6] Ikonomakis Emmanouil K , Kotsiantis S , Tampakas V (2005). *Text classification using machine learning techniques .Wseas Transactions on Computers*, 4(8): 966-974

[7] Wei Di, Anurag Bhardwaj & Jianing Wei(2018) *Why Deep Learning is perfect for NLP*

[8] Khan A , Baharudin B, Lee LH, Khan K (2010). *A review of machine learning algorithms for text-documents classification.* Journal of Advances in Information Technology, 1(1): 4-20.

[9] Sharma H, Kumar S (2016). *A survey on decision tree algorithms of classification in data mining.* International Journal of Science and Research, 5(4): 2094-2097

[10] Mashat A F, Fouad Mohammed M , S Yu P , Gharib Tarek F (2012). *A decision tree classification model for university admission system.* International Journal of Advanced Computer Science and Applications(IJACSA), 3(10): 17-21

[11] Kristopher Flint, Zachary Kellerman*NLP Disaster Tweets Classification*, 2020

[12] J. B. Lee, M. Ybanez, M. M. De Leon, and M. R. E. Estuar,2014 *"Under- ˜ standing the behavior of filipino twitter users during disaster," Journal on Computing (JoC), vol. 3, no. 2,*

[13] *Twitter Wikipedia*

[14] Jeff Hale(2019)*Don't Sweat the Solver Stuff*

[15] Oinkina(2015)*Understanding LSTM Networks*