Real or Not? NLP with Disaster Tweets

COLLEGE: Nangyang Business School

Major: Business Analytics

Group: 4

INSTRUCTOR: Teoh Teik Toe

Date: 2020.10.25

Summary

Twitter is an instantaneous mode of communication that nowadays anyone can report an emergency they are observing at present in real-time. However, we must differentiate between what tweets indicate a real disaster and what tweets do not before action. Based on a Kaggle dataset, our project would build up NLP models to predict real disaster from tweets and evaluate which model performs the best.

Firstly, we did exploratory data analysis. We create seven new derived dimensions (columns) to better describe the data sets and found the train set and test set should be from the same data source. Leveraging on grouping by “target” variable, we get the underlying sentiment in real disaster-related text is more negative than the non-disaster-related tweets. Moreover, the real disaster tweets generally include more punctuations, hashtags, and less mention than the fake ones.

Then we did the text cleaning. We remove URLs, HTML tags, punctuations and perform a spelling correction on the strings of text. Once this is done, we tokenize every tweet into a processed corpus that is in a list of lists format where various vectorization methods for machine learning can subsequently be employed.

Logistic regression, multinomial naïve bayes and recurrent neural network are established in this project. We use CountVector, TF-IDF and to vectorize the text data for word embedding, then applied machine learning models to the tokenized data. However, it is noticed that the prediction models are not 100% correct. Considering the high cost of neglecting a true alarm, we consider recall value for comparing the model as we consider that predicting actual disaster as correct is more important.

For comparison of three models with various vectorization methods, RNN is the best model with a 72.23% recall rate. Recurrent neural network can retain information of previous state to predict the outcome of the input. Since, the input is a long sequence of words, recurrent neural network is trained instead of feed forward network on the trainset. LSTM is a variant of recurrent neural network which performs better than recurrent neural network in condition where there is long sequence of input and is designed to hold important information for long time.

For application, our models can be applied in three aspects. Firstly, many relevant agencies are utilizing twitter to gain timely development of disasters. Our models can be used in such scenario. Additionally, our models can filter out the noise and avoid fake news about disasters. When an emergency occurs, our models will recognize it as soon as possible and alarm organizations to take measures and post it to citizens. Thirdly, our models can also be used in post-disaster analysis.

For future study, it is possible to improve our findings in several aspects. In terms of accuracy, we can improve our models to achieve higher accuracy and recall value. Also, we want to try to use other advanced models like SVM. Lastly, deeper insights can be obtained from our analysis results, like identify types and features of disasters.

Contents

1. Motivation..1
    1.1 Introduction..1
    1.2 Project Objectives..1
2. Literature Review..1
3. Data Overview..2
    3.1 Data Description..2
    3.2 Data Preparation..3
        3.2.1 Create Calculation fields (new columns)..3
        3.2.2 Summary Table..3
    3.3 Data Visualization..4
        3.3.1 Text..4
        3.3.2 Punctuation in text..5
        3.3.3 Keyword..5
        3.3.4 location..5
    3.4 Correlation..7
        3.4.1 Correlation Matrix..7
        3.4.2 Correlations between calculation fields with DV..7
    3.5 Data Cleaning..8
        3.5.1 Pre-Data Cleaning..8
        3.5.2 Data Cleaning..9
4. Modelling..10
    4.1 Model Configuration..10
        4.4.1 Vectorization Methods..10
        4.4.2 Models..11
        4.4.3 Evaluation criteria..11
    4.2 Logistic Regression..11
    4.3 Naïve Bayes – Multinomial Model..12
    4.4 Recurrent Neural Network (RNN)..12
        4.4.1 Loading of GloVe file to dictionary in python..13
        4.4.2 Creating Sequence of Words..13
        4.4.3 Padding..13
        4.4.4 Creating Embedding Matrix..14
        4.4.5 Adding layers and parameters for Training..14
        4.4.6 Training of Model..15
    4.5 Model Comparison..15
5. Model application..16
    5.1 Aid in relevant agency..16
    5.2 Enhance crisis awareness..16
    5.3 Post-disaster analysis..17
6. Conclusion..17
7. Future Study..17

1. Motivation

1.1 Introduction

Twitter is an almost instantaneous mode of communication and has not only become an important means of communication in the event of an emergency but also a useful means of almost instantaneous emergency detection. The accessibility and commonality of smartphones nowadays have enabled anyone to report an emergency they are observing at present in real-time. As a result, disaster response organizations are increasingly interested in monitoring social media tweets, allowing them to reduce lag time and swiftly act in the event of a disaster.

However, it is not always clear whether a person's sentence indicates a disaster. For instance, a person's use of the word "on fire" could be used to refer to something metaphorically rather than literally. Hence, there is a need for natural language processing techniques to differentiate between what tweets indicate a real disaster and what tweets do not. In this project, we explore a Kaggle competition dataset of tweets with real and fake disaster labelling to examine after data visualization and cleaning, several machine learning techniques applied to the natural language processing of tweets to distinguish between tweets that indicate the occurrence of a disaster and those that do not.

Our models can be of benefit to various aid agencies in disaster forecasting and relief deployment, enhancement of crisis awareness among the general public and research in post-disaster inquiry and analysis.

1.2 Project Objectives

There are several key objectives this paper aiming at:

- Introduce AI tech in the scanning of social media posts, in particular Twitter tweets to aid relevant agency monitoring for potential disasters and emergency response in the event of a disaster.
- Develop a Natural Language Processing (NLP) machine learning model that is able to predict if a tweet is about a real disaster or not.
- The model should not fail to detect an emergency when it happens. However, it should also not give off too many false alarms which can be very disruptive. Hence, it should strive for a high prediction accuracy score.
- Having said the above, the consequences of failing to detect an emergency when it happens can be very dire. It must be ensured that the model has a sufficiently high prediction sensitivity score.
- A dashboard to accompany the predictions and vital statistics of the model so that it is easy for any relevant agency monitoring staff to interpret the results, detect an emergency fast if it happens and alert emergency response.

2. Literature Review

Singh J P, Dwivedi Y K, Rana N P, et al(2019) built a classification system, which helps identify flood related tweets into high and low priority to distinguish those for urgent attention from general information. Six features are extracted from the textual part of the tweet and put into a classifier, then they applied 3 popular classification algorithms, namely (i) support vector machine (SVM), (ii)gradient boosting and(iii)random forest. After finding out users of high priority, they localized users according to the tweet text, and in absence of location information, they used Markov chain to predict user’s probable location according to historical locations, which is one of the highlights of the current research. Finally, they use precision, recall and F1 scores to evaluate the algorithms, in which random forest classifier offers best results, and the system is capable of identifying 81% users tweeting for help.

Viridiana Romero Martinez(2019) introduces some methods and models and applies these methods to dealing with text analytics problems about identifying disaster-related tweets. The first step of preparing data in this experiment is to change the raw text to a list of words or tokens, this is called ‘tokenization’ and transform those tokens into numbers, which is “numericalization”. Then, the second step is to apply the fastai module into getting the data ready for a language model (TextLMDataBunch) and for a text classifier (TextClassDataBunch). This experiment then uses a pre-trained model with the AWD-LSTM architecture, which can identify context and predict next word, to generate encoder for the classification model. The classification utilizes the former created text classifier (TextClassDataBunch) with the fine-tuned encoder. Finally, this classification got 80% accuracy.

Sakaki T, Okazaki M, Matsuo Y.(2010) proposes a method to detect events from Tweet messages and group the group into positive or negative class. They prepare three groups of features for each tweet.

- Features A (statistical features) the number of words in a tweet message, and the position of the query word within a tweet.
- Features B (keyword features) the words in a tweet.
- Features C (word context features) the words before and after the query word.

To process Japanese texts, morphological analysis is conducted using Mecab, which separates sentences into a set of words. In the case of English, standard stopword elimination and stemming is applied. Using the Temporal model, they classify whether a new tweet corresponds to a positive class or a negative class.

Alseadi et al (2017) performed disruptive event detection using Twitter by classification and online clustering. In classification step event-related tweets were separated from non-event tweets, while in clustering step the topic of an event could be identified. Specifically, the contents of posts were transformed into feature vectors and their corresponding category (event or non-event) were provided when being classified. The author evaluated 3 different classification algorithms on accuracy, precision, and other aspects. The algorithms were Naïve Bayes, Logistic Regression and Support Vector Machines (SVM). Among them, Naïve Bayes performed best with 86.13% accuracy and less overfitting. But all 3 methods produced over 80% accuracy in detecting expected events in twitter posts.

3. Data Overview

3.1 Data Description

The link of the dataset is: https://www.kaggle.com/c/nlp-getting-started/data. This project offers two csv datasets, train and test. The train contains all columns and has 7613 rows, while the test set has 3263 rows and no target column, which will be predicted after the analysis.

There are 6 columns in the dataset:

- id - a unique identifier for each tweet
- text - the content of the tweet
- location - the location the tweet was sent from (can be null)
- keyword - a particular keyword in the tweet (can be null)
- target - this denotes whether a tweet is about a real disaster (1) or not (0), this information is only available in the train set.

3.2 Data Preparation

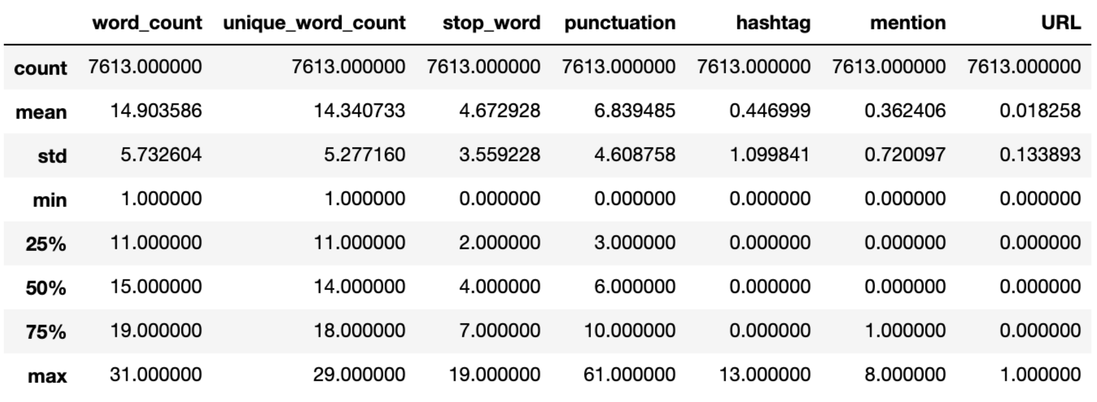
3.2.1 Create Calculation fields (new columns)

In order to describe the datasets thoroughly, we create seven new dimensions to record these details.

- word_count: number of words in text (numeric)
- unique_word_count: number of unique words in text (numeric)
- stop_word_count: number of stop words in text (numeric)
- url_count: contain urls in text or not (categorical)
- punctuation_count: number of punctuations in text (numeric)
- hashtag_count: number of hashtags (#) in text (numeric)
- mention_count: number of mentions(@) in text (numeric)

3.2.2 Summary Table

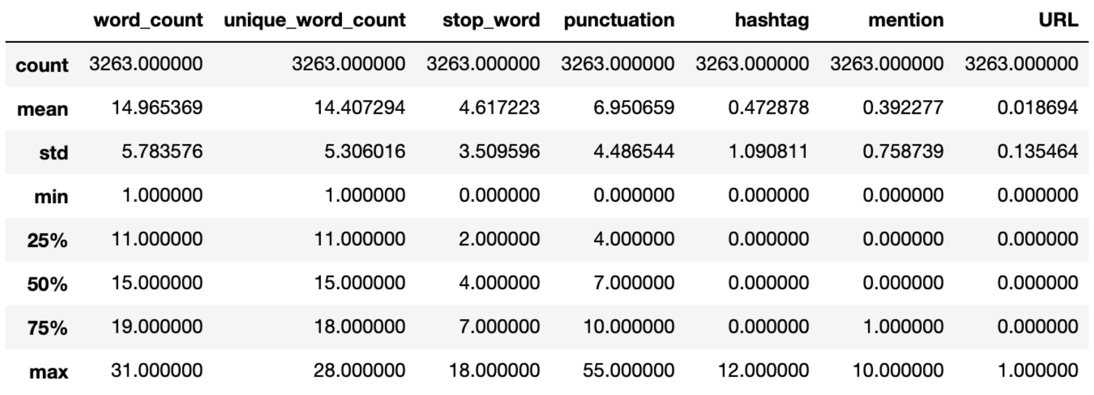
The below table describes the basic information in trainset. There are 7613 entries in the train set. A total of 4342 fake disaster tweet records in the train set and 3271 are real ones. The general information about each tweet is about 15 words with 6 punctuations and no hashtag, no mention.



	word_count	unique_word_count	stop_word	punctuation	hashtag	mention	URL
count	7613.000000	7613.000000	7613.000000	7613.000000	7613.000000	7613.000000	7613.000000
mean	14.903586	14.340733	4.672928	6.839485	0.446999	0.362406	0.018258
std	5.732604	5.277160	3.559228	4.608758	1.099841	0.720097	0.133893
min	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	11.000000	11.000000	2.000000	3.000000	0.000000	0.000000	0.000000
50%	15.000000	14.000000	4.000000	6.000000	0.000000	0.000000	0.000000
75%	19.000000	18.000000	7.000000	10.000000	0.000000	1.000000	0.000000
max	31.000000	29.000000	19.000000	61.000000	13.000000	8.000000	1.000000

Figure 1 summary table (trainset)

The below table summarizes the basic information in test set. There are 3263 records in the test set. The general information about each tweet in this set is around 15 words without hashtag, mention and URL, and 7 punctuations.



	word_count	unique_word_count	stop_word	punctuation	hashtag	mention	URL
count	3263.000000	3263.000000	3263.000000	3263.000000	3263.000000	3263.000000	3263.000000
mean	14.965369	14.407294	4.617223	6.950659	0.472878	0.392277	0.018694
std	5.783576	5.306016	3.509596	4.486544	1.090811	0.758739	0.135464
min	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	11.000000	11.000000	2.000000	4.000000	0.000000	0.000000	0.000000
50%	15.000000	15.000000	4.000000	7.000000	0.000000	0.000000	0.000000
75%	19.000000	18.000000	7.000000	10.000000	0.000000	1.000000	0.000000
max	31.000000	28.000000	18.000000	55.000000	12.000000	10.000000	1.000000

Figure 2 summary table (testset)

We can see that the basic information in both two sets are almost same, which means the two sets are from the same data source.

3.3 Data Visualization

3.3.1 Text

In this part, we divide the trainset text into two groups based on the “target” equal to zero or one. The left chart is the real disaster tweet word cloud and the right is fake disaster tweet word cloud.

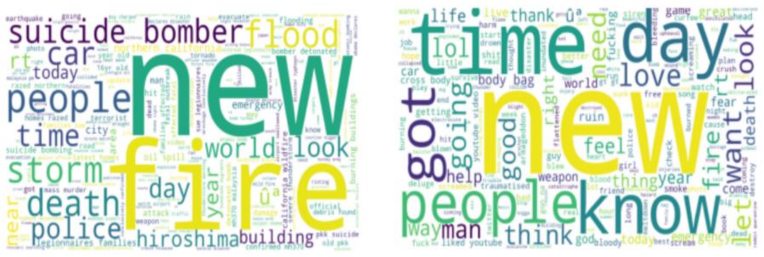


Figure 3 trainset(target=1&0)

The sentiment of the words in the left chart is more negative and the right chart stands on the opposite.

Due to no “target” variable in the test set, we calculate the word frequency and show the ranking.

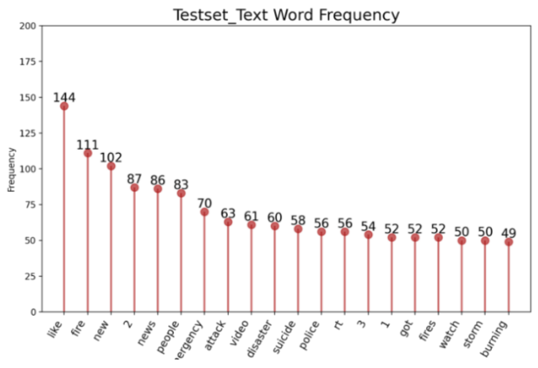


Figure 4 testset word frequency

The top 3 words are like, fire and new.

3.3.2 Punctuation in text

We plot the number of each punctuation in trainset' two groups (target=1 or 0) respectively. The left bar chart is True and the right is False.

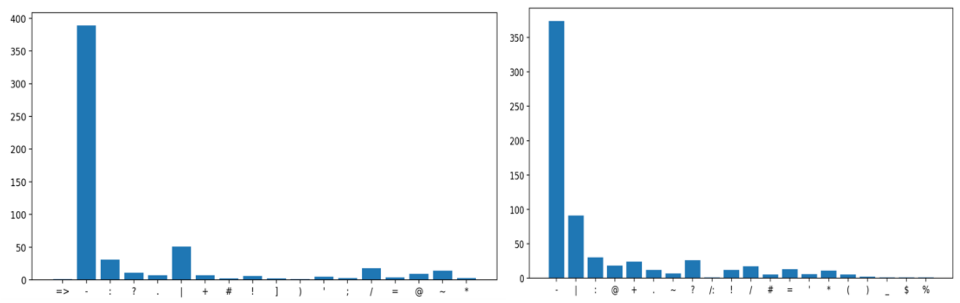


Figure 5 trainset punctuation (target=1&0)

There is no difference in two groups' punctuation ranking.

3.3.3 Keyword

Also, we plot the keyword frequency in both trainset and testset. As for the trainset, the two groups of keywords are drawn on different charts.

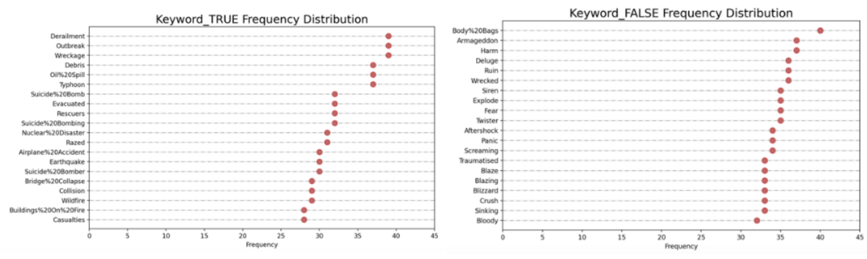


Figure 6 keyword frequency (target=1&0)

The sentiments of the top 20 used keywords in two groups are more oriented toward negative way.

The below chart shows the most mentioned keywords in the testset.

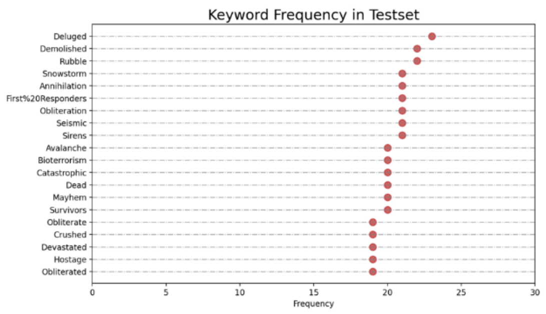


Figure 7 keyword frequency (test set)

3.3.4 location

Also, we plot the locations in two sets in the same way as we had done in keyword. In addition, we use maps to reflect the distributions. The below map shows the top mentioned locations in trainset under the "target equal to 1" condition.

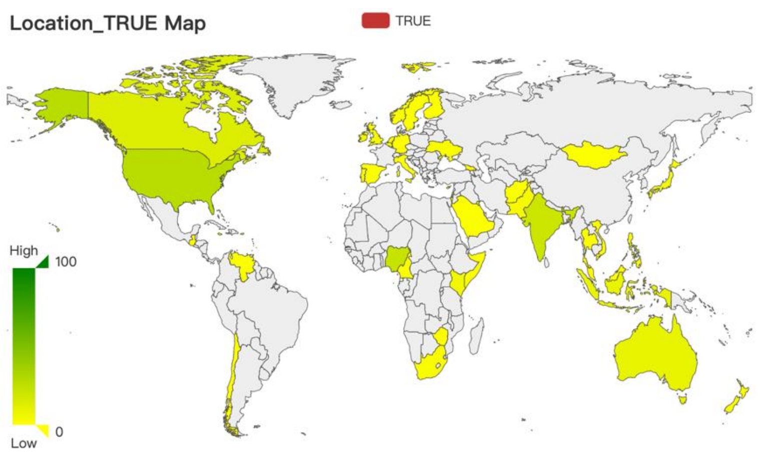


Figure 8 top mentioned locations(target=1)

We can see the real disaster locations mainly happened on U.S., Canada, India and Australia.

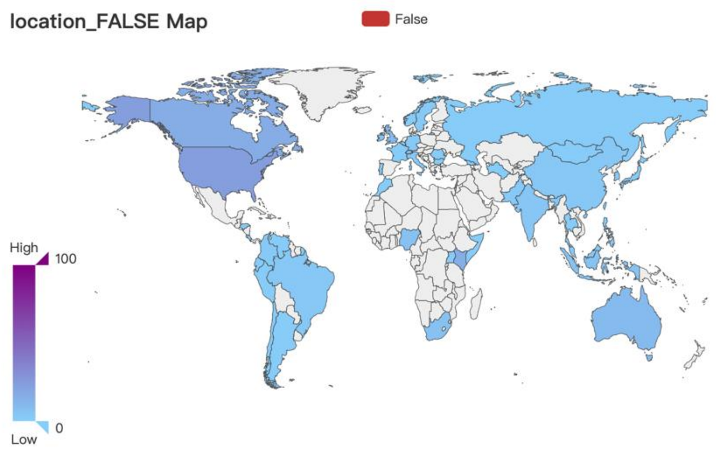


Figure 9 top mentioned locations(target=0)

According the above map, we know the U.S. and Canada, Australia are also the most-used locations in fake tweets.

The below map shows the locations in testset.

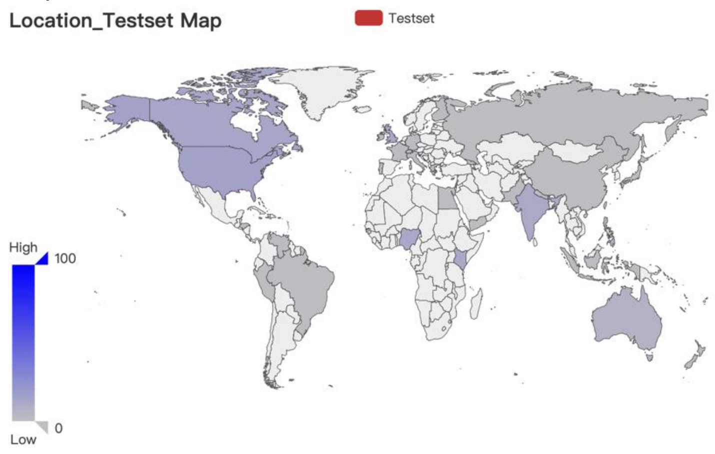


Figure 10 top mentioned locations (test set)

3.4 Correlation

3.4.1 Correlation Matrix

In this part, we focus on train set as the real value of dependent variable only exists in train set. So we explore the correlation between dependent variable and created calculation fields.

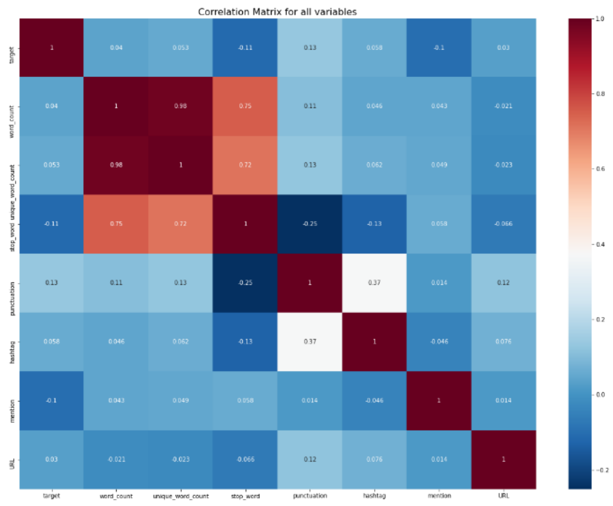


Figure 11 Correlation Matrix (train set)

The punctuation count is the most highly correlated factor with dependent variable. We will dive into the correlations in the next part.

3.4.2 Correlations between calculation fields with DV

We plot the number of punctuations, the number of hashtags in two target groups respectively.

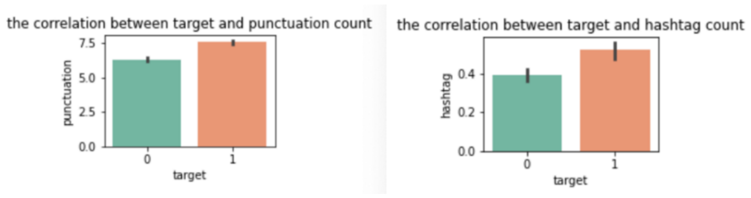


Figure 12 punctuations and hashtags (target=1&0)

The real disaster tweets contain more punctuations and hashtags than the fake ones.

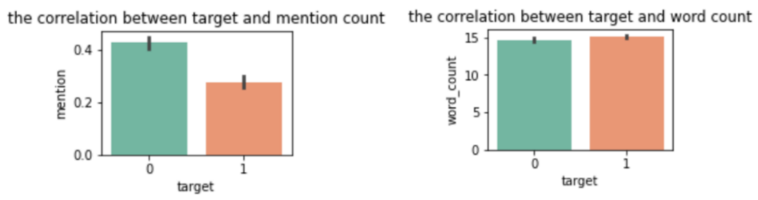


Figure 13 mentions and words (target=1&0)

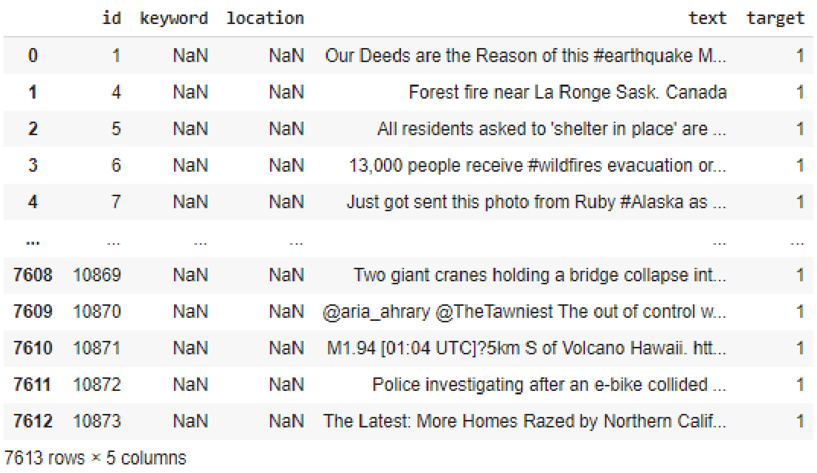
While as the above charts shown, the real disaster tweets include less mentions (@) than the other group. Moreover, the number of words in two groups are mostly equal.

3.5 Data Cleaning

Then we lay out a procedure to clean the data and pre-process it before it can be in a suitable format that is ready to be input into a machine learning model for text analytics. Concurrently, the pre-processing process can be done so as to provide intuitive insights on the texts being examined.

The relevant columns of interest are the text and target columns. The values in the text column are essentially whole strings of tweets as tweeted by users. They are typically messy and contain unwanted symbols, URLs, HTML tags, punctuations, misspelt words etc. which need to be cleaned. We use the nltk package in Python to aid in the text cleaning. For preliminary data exploration, we investigate the type of stop words, split the strings into elementary strings to create a preliminary corpus to get a sense of the types of unwanted elementary strings and also count the number of occurrences of punctuations and unwanted symbols. Next, we proceed with the data cleaning by removing URLs, HTML tags, punctuations and unwanted symbols as well as performing spell correction for the strings of text in every row of the text column of the train set data frame. Tokenization and creation of the corpus of words is subsequently performed.

TFIDF method is used. Term frequency–inverse document frequency, or TFIDF, presents a mathematical method to weight text information based on probability, and thus measure the importance of words in documents or corpus based on an information-theoretic view of retrieval events (Aizawa A, 2003). In this project, TFIDF will be applied to different models to remove the less important words, and see whether or not the results can be benefited comparing to the original version.



	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1
...	...	...	...	...	...
7608	10869	NaN	NaN	Two giant cranes holding a bridge collapse int...	1
7609	10870	NaN	NaN	@aria_ahrary @TheTawniest The out of control w...	1
7610	10871	NaN	NaN	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	1
7611	10872	NaN	NaN	Police investigating after an e-bike collided ...	1
7612	10873	NaN	NaN	The Latest: More Homes Razed by Northern Calif...	1

7613 rows × 5 columns

Figure 14 original dataset

3.5.1 Pre-Data Cleaning

In the train set text column, for every row, there is a string of text containing a tweet. A simple text corpus is created in the form of a list of individual texts obtained via splitting the strings of texts into individual strings that are separated by a space. Each individual string may or may not be a proper word. A sample snapshot of the text corpus is shown below.

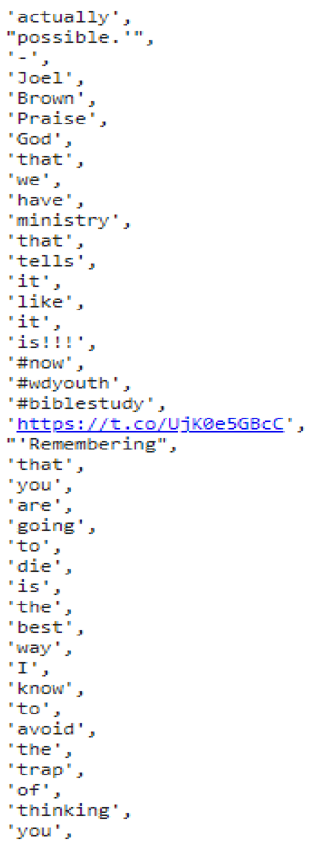

```
'actually',
"possible.'",
'-',
'Joel',
'Brown',
'Praise',
'God',
'that',
'we',
'have',
'ministry',
'that',
'tells',
'it',
'like',
'it',
'is!!!',
'#now',
'#wdyouth',
'#biblestudy',
'https://t.co/UjK0e5GBcC',
"'Remembering",
'that',
'you',
'are',
'going',
'to',
'die',
'is',
'the',
'best',
'way',
'I',
'know',
'to',
'avoid',
'the',
'trap',
'of',
'thinking',
'you',
```

Figure 15 text corpus

As we can see in the sample snapshot above, there are unwanted special symbols called punctuations and URLs. In addition, we note that there are also other issues such as HTML tags and misspelt words and the text data has to be cleaned before any meaningful text analytics can be performed. We subsequently investigate the type of punctuations that are present in the text corpus and the corresponding frequency of which they occur in the train set as shown in the bar chart below.

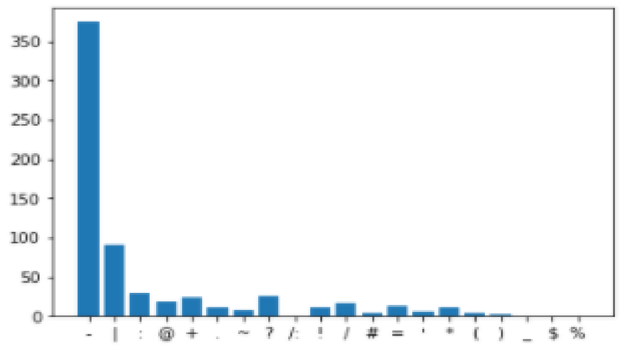
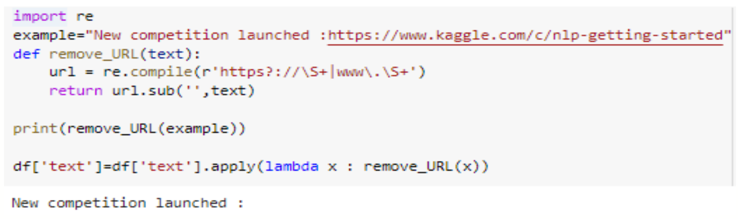


Figure 16 punctuations frequency

3.5.2 Data Cleaning

We start off by removing URLs from the texts in the original train set dataframe. The URL removal code is tested on an example to ensure that it works as expected. An example string "New competition launched: https://www.kaggle.com/c/nlp-getting-started" should be transformed to "New competition launched :" after we perform the URL removal and it works as expected as shown below.

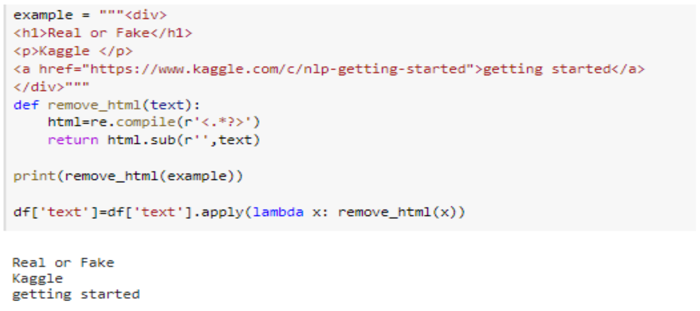


```python
import re
example="New competition launched :https://www.kaggle.com/c/nlp-getting-started"
def remove_URL(text):
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub('',text)

print(remove_URL(example))

df['text']=df['text'].apply(lambda x : remove_URL(x))
```

```
New competition launched :
```

Figure 17 URL Removal Code

We subsequently remove HTML tags from the texts. The HTML tag removal code is tested on an example to ensure that it works as expected. The string containing various URL tag types as shown below should have all its HTML rags removed after we perform the HTML tag removal and it works as expected as shown below.

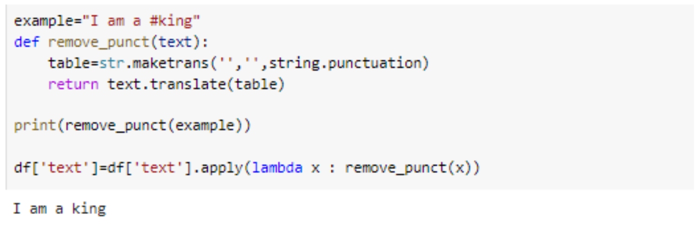


```python
example = """<div>
<h1>Real or Fake</h1>
<p>Kaggle </p>
<a href="https://www.kaggle.com/c/nlp-getting-started">getting started</a>
</div>"""
def remove_html(text):
    html=re.compile(r'<.*?>')
    return html.sub(r'',text)

print(remove_html(example))

df['text']=df['text'].apply(lambda x: remove_html(x))
```

```
Real or Fake
Kaggle
getting started
```

Figure 18 HTML Tags Removal Code

Moving on, we remove punctuations from the texts. The punctuation removal code is tested on an example to ensure that it works as expected. The string containing a "#" symbol as shown below should have the "#" symbol removed after we perform the punctuation removal and it works as expected.



```python
example="I am a #king"
def remove_punct(text):
    table=str.maketrans('','',string.punctuation)
    return text.translate(table)

print(remove_punct(example))

df['text']=df['text'].apply(lambda x : remove_punct(x))
```

```
I am a king
```

Figure 19 Punctuations Removal Code

We are at this stage ready to perform a spell correction to correct any misspelt words that may be present in the texts before we finally tokenize our texts into the eventual corpus in the form of a list of lists, we wish to use in our machine learning model input. The corpus is saved in a txt file and acts as our cleaned train set, ready to be uploaded for any subsequent model training.

4. Modelling

4.1 Model Configuration

4.4.1 Vectorization Methods

CountVector: CountVector will generate a matrix of the number of split words in each line. It is a basic method for text vectorization.

TF_IDF: Term frequency–inverse document frequency, or TFIDF, presents a mathematical method to weight text information based on probability, and thus measure the importance of words in documents or corpus based on an information-theoretic view of retrieval events (Aizawa A, 2003).

GloVe: Unlike TF-IDF which is based on sparse vector representation, GloVe (Global Vectors for Word Representation) relies on the dense vector representation (Céline Van den Rul, 2019). GloVe is a neural embedding algorithm often applied to unsupervised machine learning in neural network. Its way of vectorization is not based on the frequency, but word-word co-occurrences. For example, the word 'hot' would happen more alongside 'summer'.

4.4.2 Models

In this project, three models including logistic regression, naïve bayes multinomial, and neural network are established to fit the dataset and conduct prediction. Because GloVe will generate negative values (which is not allowed in naïve bayes) and GloVe is considered more suitable for neural network, logistic regression and naïve bayes will use CountVector and TF-IDF method, while neural network will use GloVe.

4.4.3 Evaluation criteria

• Accuracy: the number of the condition of being predicted correctly divided by the total number of predictions. The train-test split is set at 0.2.

• Recall Rate: Among those texts who reported actual disasters, the number of the condition of being predicted correctly divided by the total amount of real texts.

• Average f1 score: f1 score is calculated in the cross validation which takes both accuracy and recall rate into consideration. This is used to test whether the result from the one-time train-test split is stable.

4.2 Logistic Regression

Logistic regression, one of the popular machine learning models, can handle nonlinear and nonparametric data. It relies on the linearity assumption between the dependent variables and the independent ones. LR can incorporate ordinal information and generate a basic understanding.

The result for test set is as below:

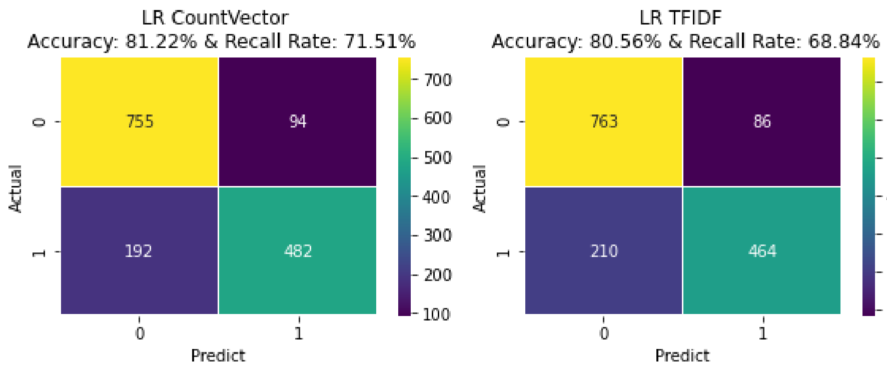


Figure 20 Confusion Matrix for LR (CountVector &TFIDF)

In order to avoid the particularity of the train-test split, a 3-fold cross-validation has also been conducted. The average f1 score for "LR CountVector" is 63.36%, for "LR TFIDF" is 62.37%. So, it's safe to draw the conclusion that the CountVector is better than TF-IDF, especially when it comes to the recall rate.

Note that this situation is abnormal. TF-IDF should improve the model by adjusting the weight based on the frequency of words. A reasonable explanation may be that in this case, stop words, which are often considered repeatable and meaningless, have already been removed. Therefore, the words in the rest of the content, even with a higher frequency, can also contribute to the final prediction. In this project, obviously the lower weights of some frequently-used words cause a remarkable loss of predicting a true disaster. It is well-acknowledged by the psychologists that people's words and behaviour can imply their minds. This might also apply to disaster posting tweets.

4.3 Naïve Bayes – Multinomial Model

Naïve bayes classification is one of the basic probabilistic models. Among text classification, most-used two naïve bayes models are "multinomial" and "multi-variate Bernoulli". In this project, the multinomial model is selected because it is found to be generally better in error reducing (McCallum A, Nigam K., 1998).

The confusion matrices of naïve bayes for both count vector and TF-IDF is as below:

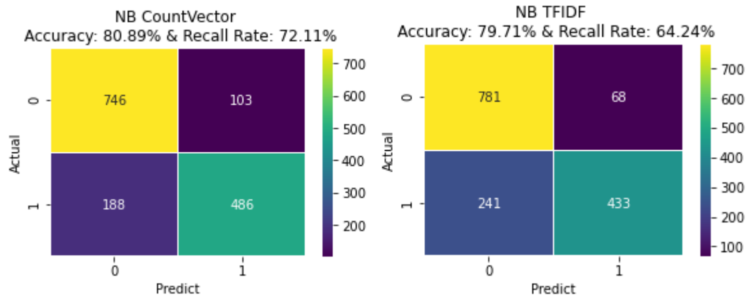


Figure 21 Confusion Matrix for Naïve Bayes (CountVector &TFIDF)

The average f1 score for "NB CountVector" is 68.59%, for "NB TFIDF" is 64.49%. This shows consistency with the LR performance above, which means TF-IDF is not helpful in the project. And based on the cross-validation score, multinomial naïve bayes is a better predictor.

4.4 Recurrent Neural Network (RNN)

Neural network is one of the models we explored to solve the problem, where each tweet will be classified into disaster or not a disaster. There are variety of machine learning models which can solve this problem, and each has its advantages and disadvantages. The model which is being discussed below is the LSTM Model which is a type of Recurrent Neural Network and can be used where it requires to grasp long term dependencies.

We have feed forward networks where inputs are multiplied by weights, bias is added, and output is passed through activation function to predict the output. But the problem with such type of network is that they do not store the previous inputs in memory while training the model. It would be difficult to achieve good accuracies using such type of model for problems where there is dependency on sequence of data. Recurrent Neural Network were introduced to solve such type of sequence problem. In RNN, we consider the weights associated with output of previous states along with weights associated with inputs of previous state to get an output after passing it through activation function. In this way, we consider the previous state to solve the problem associated with feed forward network for sequence problems. But RNN suffers from vanishing gradient problem that is very significant change in weights do not help to learn the model. LSTM is a variant of Recurrent Neural Network. It helps to store important information for a long time which is required to predict based on complete sequence. LSTM fits the current problem and was used for training the model for this problem.

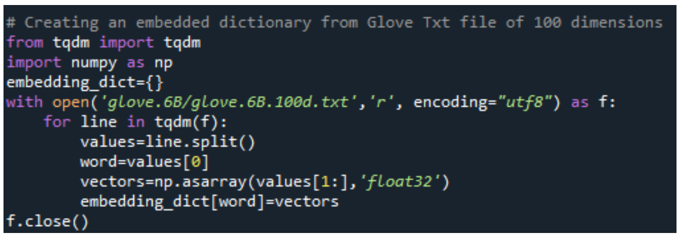
As mentioned during Data Exploration and Analysis, data is completely processed and a corpus is created after removing punctuations, removing stop words, correcting the spellings, removing hyperlinks and HTML tags. Each sentence was tokenized and made suitable for running the models for analytics.

For training Neural Network model for analysis of Tweet, a pre-existing embedding of word vectors i.e. GloVe was used. GloVe is a pre-trained vector of words, which is used to transform each word in a new corpus to replace it with corresponding numeric vector. For current modelling, 100-dimension vector of GloVe was used.

Following are the steps involved in using GloVe and training the model for classification of tweets:

4.4.1 Loading of GloVe file to dictionary in python

An empty dictionary was created to load each word and its corresponding vector from GloVe File to this dictionary.

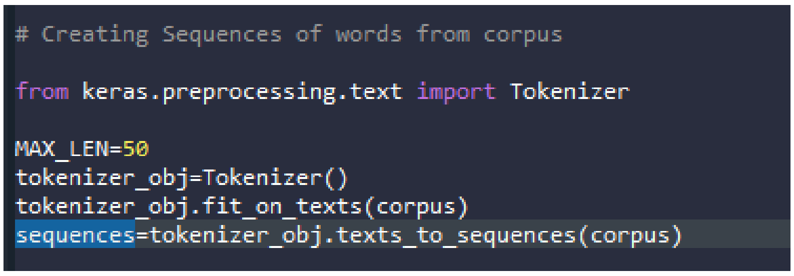


```
# Creating an embedded dictionary from Glove Txt file of 100 dimensions
from tqdm import tqdm
import numpy as np
embedding_dict={}
with open('glove.6B/glove.6B.100d.txt','r', encoding="utf8") as f:
    for line in tqdm(f):
        values=line.split()
        word=values[0]
        vectors=np.asarray(values[1:],'float32')
        embedding_dict[word]=vectors
f.close()
```

Figure 22 Empty Dictionary Creation

4.4.2 Creating Sequence of Words

A sequence of words was created from the Corpus. That is each word was assigned a number and saved in a list for each corresponding sentence. This was done to change each word to a numeric value which makes it easy for neural network to train on this data.

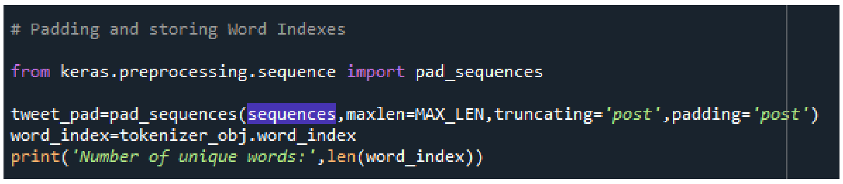


```
# Creating Sequences of words from corpus

from keras.preprocessing.text import Tokenizer

MAX_LEN=50
tokenizer_obj=Tokenizer()
tokenizer_obj.fit_on_texts(corpus)
sequences=tokenizer_obj.texts_to_sequences(corpus)
```

Figure 23 Creating Sequence of Words

4.4.3 Padding

To make each list corresponding to each sentence of equal size, each list was padded to size of 50 which is set as MAX_LEN. The post padding was done to set places after the generated sequence for each sentence to 0’s.

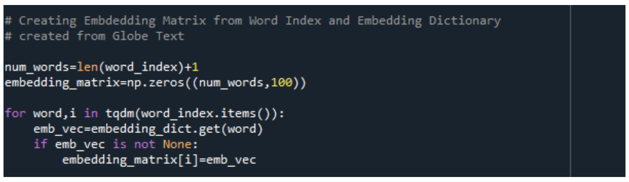


```
# Padding and storing Word Indexes

from keras.preprocessing.sequence import pad_sequences

tweet_pad=pad_sequences(sequences,maxlen=MAX_LEN,truncating='post',padding='post')
word_index=tokenizer_obj.word_index
print('Number of unique words:',len(word_index))
```

Figure 24 Padding

4.4.4 Creating Embedding Matrix

Next step to before training neural network model is to create an embedding matrix from Word Index. Here is when previously created embedding dictionary from GloVe is used to create embedding matrix. Each word is replaced with its corresponding vector from Embedding Dictionary. The list of the list of these 100-dimension vectors create a matrix.



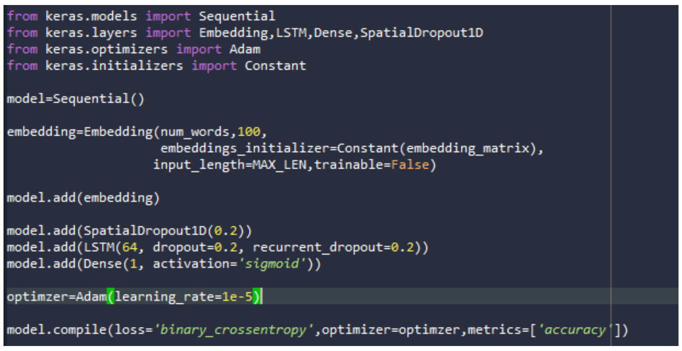
```
# Creating Embdedding Matrix from Word Index and Embedding Dictionary
# created from Globe Text

num_words=len(word_index)+1
embedding_matrix=np.zeros((num_words,100))

for word,i in tqdm(word_index.items()):
    emb_vec=embedding_dict.get(word)
    if emb_vec is not None:
        embedding_matrix[i]=emb_vec
```

Figure 25 Creating Embedding Matrix

4.4.5 Adding layers and parameters for Training

Following parameters were added to neural network for training the model. There is an Embedding layer with parameters as shown in following figure. Number of words is the total number of words in the training set. The embedding is initialized with the embedding matrix formed in previous step. Input length is set to 50 which is Max_Len set for each sentence.

Further Spatial dropout layer is added to reduce the chances of overfitting of trainset. LSTM model is preferred as it helps to store important information for a long time which is required to predict based on complete sequence of text. Finally, a Dense Layer is added with sigmoid activation function to generate the final output. Adam optimizer is added to reduce the losses using learning rate as mentioned in figure below.

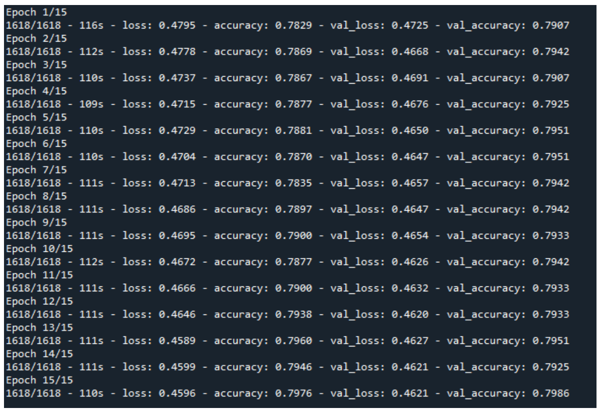


```
from keras.models import Sequential
from keras.layers import Embedding,LSTM,Dense,SpatialDropout1D
from keras.optimizers import Adam
from keras.initializers import Constant

model=Sequential()

embedding=Embedding(num_words,100,
                    embeddings_initializer=Constant(embedding_matrix),
                   input_length=MAX_LEN,trainable=False)

model.add(embedding)

model.add(SpatialDropout1D(0.2))
model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

optimzer=Adam(learning_rate=1e-5)

model.compile(loss='binary_crossentropy',optimizer=optimzer,metrics=['accuracy'])
```

Figure 26 Add layers and parameters

4.4.6 Training of Model

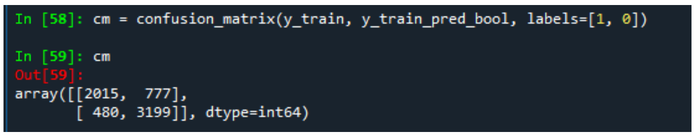
Model was trained for 15 epochs and following accuracy was achieved for Train and Validation set. Initially the model was trained after lemmatization process but the final validation accuracy reduced to 76% on apply lemmatization. Hence the Lemmatization process was omitted while training the model and the accuracy achieved is around 79.8% on validation set. Accuracy for final trained model is as below.



```
Epoch 1/15
1618/1618 - 116s - loss: 0.4795 - accuracy: 0.7829 - val_loss: 0.4725 - val_accuracy: 0.7907
Epoch 2/15
1618/1618 - 112s - loss: 0.4778 - accuracy: 0.7869 - val_loss: 0.4668 - val_accuracy: 0.7942
Epoch 3/15
1618/1618 - 110s - loss: 0.4737 - accuracy: 0.7867 - val_loss: 0.4691 - val_accuracy: 0.7907
Epoch 4/15
1618/1618 - 109s - loss: 0.4715 - accuracy: 0.7877 - val_loss: 0.4676 - val_accuracy: 0.7925
Epoch 5/15
1618/1618 - 110s - loss: 0.4729 - accuracy: 0.7881 - val_loss: 0.4650 - val_accuracy: 0.7951
Epoch 6/15
1618/1618 - 110s - loss: 0.4704 - accuracy: 0.7870 - val_loss: 0.4647 - val_accuracy: 0.7951
Epoch 7/15
1618/1618 - 111s - loss: 0.4713 - accuracy: 0.7835 - val_loss: 0.4657 - val_accuracy: 0.7942
Epoch 8/15
1618/1618 - 111s - loss: 0.4686 - accuracy: 0.7897 - val_loss: 0.4647 - val_accuracy: 0.7942
Epoch 9/15
1618/1618 - 111s - loss: 0.4695 - accuracy: 0.7900 - val_loss: 0.4654 - val_accuracy: 0.7933
Epoch 10/15
1618/1618 - 112s - loss: 0.4672 - accuracy: 0.7877 - val_loss: 0.4626 - val_accuracy: 0.7942
Epoch 11/15
1618/1618 - 111s - loss: 0.4666 - accuracy: 0.7900 - val_loss: 0.4632 - val_accuracy: 0.7933
Epoch 12/15
1618/1618 - 111s - loss: 0.4646 - accuracy: 0.7938 - val_loss: 0.4620 - val_accuracy: 0.7933
Epoch 13/15
1618/1618 - 111s - loss: 0.4589 - accuracy: 0.7960 - val_loss: 0.4627 - val_accuracy: 0.7951
Epoch 14/15
1618/1618 - 111s - loss: 0.4599 - accuracy: 0.7946 - val_loss: 0.4621 - val_accuracy: 0.7925
Epoch 15/15
1618/1618 - 110s - loss: 0.4596 - accuracy: 0.7976 - val_loss: 0.4621 - val_accuracy: 0.7986
```

Figure 27 RNN Training Process

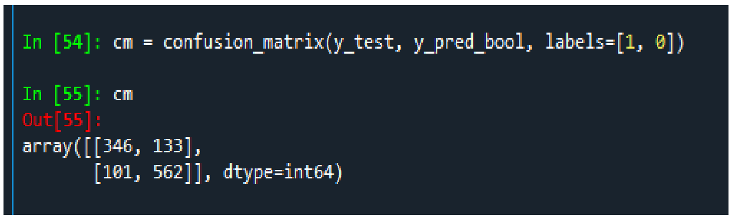
Trainset Confusion Matrix:



```
In [58]: cm = confusion_matrix(y_train, y_train_pred_bool, labels=[1, 0])

In [59]: cm
Out[59]:
array([[2015,  777],
       [ 480, 3199]], dtype=int64)
```

Figure 28 Trainset Confusion Matrix

Test Set Confusion Matrix:



```
In [54]: cm = confusion_matrix(y_test, y_pred_bool, labels=[1, 0])

In [55]: cm
Out[55]:
array([[346, 133],
       [101, 562]], dtype=int64)
```

Figure 29 Test set Confusion Matrix

Based on the confusion matrix, if we calculate the accuracy, recall, and precision of the model, we could get the following result:

Data	Accuracy	Recall	Precision
Train set	79.76	72.17	80.76
Validation set	79.86	72.23	77.4

Table 1 RNN Result

4.5 Model Comparison

Models	Accuracy	Recall
Logistic Regression	81.22	71.51
Naïve Bayes	80.89	72.11

Recurrent Neural Network	79.86	72.23

Table 2 Model Comparison

Looking at the comparison table for all the models, it’s clearly evident that all models perform comparatively similar. We think that recall value is the better parameter for the comparison of all the models as it would be dangerous if an actual disaster is predicted as non-disaster from the tweet. This might lead to negligence of the tweet and may result in not taking appropriate action time. The recall value is also comparatively equal from all the three models. RNN outperforms the other two models with a minor difference in recall value. RNN model is trained by keeping previous state information of input to predict the next outcome. The sequence of information stored in RNN results in better performance than other two model or a feed forward model.

5. Model application

Social media has become such an important communication tool that in times of emergency people always seek help or share information on these platforms. Among them, twitter is one of the most frequently used instant communication applications. Because of this, more and more agencies are interested in monitoring live tweets and get concrete information about a disaster or an emergency through analysis by machine learning models. There are several potential applications of our work and we will elaborate in detail.

5.1 Aid in relevant agency

Social media is the most immediately accessible source of information. Therefore, many relevant agencies are utilizing it to gain timely development of disasters, from natural disasters to man-made accidents, and even pandemics. At the same time, twitter allows news to spread very rapidly and as a result, it has great potential usage in disaster detection. For example, the terrible Hurricane Sandy in 2012 was extensively reported by twitter users. Also, in the outbreak of the Zika virus in 2015, World Health Organization (WHO) utilized twitter to gather information and post latest updates. If our models are used in such disaster forecasting process, those organizations are able to perform with higher efficiency and accuracy, which will reduce the loss of economy and society.

5.2 Enhance crisis awareness

Twitter as a form of social media is a platform for each individual to express themselves and capture timely news. But it’s not easy for citizens to figure out whether a disaster is happening or not. Sometimes the tweets are not true although they seem like to describe a disaster, while sometimes the emergent tweets gain less attention from viewers. With our predictive models, we can filter out the noise and avoid fake news about disasters. This could prevent unnecessary panic among people. More importantly, when an emergency occurs, our models will recognize it as soon as possible and alarm organizations to take measures and post it to citizens. In this way, during a crisis, people are more aware of current situation, staying calm and cooperate with government to control this disaster.

5.3 Post-disaster analysis

Besides the occurrence of an emergency, there is a lot of valuable information that can be obtained by our models as well, and it is useful for post-disaster analysis, including:

- The reasons why the emergency happens;
- The spread process of this event;
- Public reactions (panic, anxiety, depression and so on);
- Citizen needs and requirements.

With these insights, it is possible for relative organizations to begin re-construction process, reassure the public, and prevent similar events from happen again.

6. Conclusion

In the paper, we used tweet messages to predict disaster by NLP. Firstly, we did data exploration, then data processing like removing punctuations, URLS, and spell correction are conducted. Then we applied regression, naïve bayes and RNN to trainset and validated using testset. The result shows RNN model performs best, which achieves recall value of 72.23% as compared to other two models. The reason is might be RNN allows us to consider the previous state to solve the problem associated with feed forward network for sequence problems. The model result shows that by omitting the Lemmatization process, the model has better performance. We think the model would be helpful for aid in relevant agency, crisis awareness enhancement, and post-disaster analysis.

7. Future Study

A significant improvement can be expected using larger-scale dataset for training and testing. The data sources we used have limited location and keywords that features of tweets can be added to provide more information, like time, geolocation, environmental conditions, individual information etc. In this study, we use logistic, naïve bayes, and RNN, which are traditional methods used in twitter text analysis. The overall accuracy is… , which still can be improved. Another commonly used model SVM can be explored to achieve better performance. What's more, after disaster classification, we can further try to identify types of disaster, research on features of most efficient emergency calls to guide people ask for help properly via tweets.

Reference

[1] Singh J P, Dwivedi Y K, Rana N P, et al. Event classification and location prediction from tweets during disasters[J]. Annals of Operations Research, 2019, 283(1): 737-757.

[2]Viridiana Romero Martinez, Identifying disaster-related tweets using deep learning and natural language processing with Fast Ai,2019, Link:https://medium.com/datadriveninvestor/identifying-disaster-related-tweets-using-deep-learning-and-natural-language-processing-with-fast-e0dfb790b57a

[3] Sakaki T, Okazaki M, Matsuo Y. Earthquake shakes Twitter users: real-time event detection by social sensors[C]//Proceedings of the 19th international conference on World wide web. 2010: 851-860.

[4] Alsaedi N, Burnap P, Rana O. Can we predict a riot? Disruptive event detection using Twitter[J]. ACM Transactions on Internet Technology (TOIT), 2017, 17(2): 1-26.

[5] McCallum A, Nigam K. A comparison of event models for naive bayes text classification[C]//AAAI-98 workshop on learning for text categorization. 1998, 752(1): 41-48.

[6] Aizawa A. An information-theoretic perspective of tf–idf measures[J]. Information Processing & Management, 2003, 39(1): 45-65.

[7] Littlepage, G., & Pineault, T. (1978). Verbal, facial, and paralinguistic cues to the detection of truth and lying. Personality and Social Psychology Bulletin, 4(3), 461-464.

[8] Pranckevičius T, Marcinkevičius V. Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. Baltic Journal of Modern Computing. 2017;5(2):221.

[9] Céline Van den Rul (2019). Understanding Word Embeddings with TF-IDF and GloVe, https://towardsdatascience.com/understanding-word-embeddings-with-tf-idf-and-glove-8acb63892032