# Detection of Real Disasters from Tweets

Group 1: Ankit Malhotra (A0232322X), Cristian Bojaca (A0231999L), Liu Yishun (A0231849X), Ma Yuankai (A0231868W), Yang Yuchen (A0119430N)

GitHub link: https://github.com/RobinNeverBow/BT5153-Group-Project

## Abstract

Twitter has become an important communication channel in times of emergency. The ubiquitousness of smartphones enables people to announce an emergency they are observing in real-time. Because of this, more agencies are interested in programmatically monitoring Twitter. However, it is not always clear whether a person's words are actually announcing a disaster. In this project, we aimed to implement various natural language processing (NLP) techniques to classify the tweets from the users on disasters into real or non-real. Utilizing the balanced data set, after the data preprocessing procedure, we experimented with BOW models, BERT models, Text CNN and GNN models and evaluated their model performance based on accuracy, F-1 score and ROC-AUC scores. Finally, we suggested the potential application of the models for emergency responses.

## 1. Introduction

### 1.1 Background

Nowadays, social media offers a realm of information on varied topics ranging from news, politics, entertainment, healthcare to recent trends, emotions, and opinions of people worldwide. With the omnipresence of smartphones and their easy accessibility by people of nearly all age groups, it becomes considerably easy to disseminate ideas, viewpoints, sentiments, and different schools of thought with just a mere few clicks. This level of convenience in reaching out to numerous people through online channels becomes particularly essential in crises such as natural disasters.

In such scenarios, social media platforms can play a pivotal role in providing critical information about the mishappening, including the type of disaster, its intensity and precise location of occurrence, the problems people face, and people's emotions and reactions, to name a few.

While there are different types of social networking platforms available at our disposal, Twitter has turned out to be probably one of the best mediums to find out real-time information on what's happening around. Different stakeholders involving national disaster relief organizations, governments, media, volunteers, public, etc., can use Twitter to collaborate swiftly and effectively.

However, the quality of information being posted by the users often lacks authenticity, which can lead to miscommunication and unbefitting actions by the relevant parties concerned. This motivated us to address this issue by filtering out the tweets which pertains to a real disaster, based on natural language processing (NLP) techniques and machine learning (ML) algorithms. In the forthcoming sections, we will define the problem statement, dataset, and the models that we implement to solve this problem.

### 1.2 Problem Statement

Our primary objective in this project is to classify the tweets from the users on disasters into real (1) or non-real (0). To clarify, for instance a user tweets some information relating to a scenic view, and the tweet contains the keyword *'apocalypse'*. Here, the word does not correspond to a real disaster but is just used metaphorically. Thus, it will not be classified as a disaster and will accordingly be assigned a value of 0. Therefore, the objective of the study is to build a classification model to accurately identify tweets related to real disasters. Deployment of such a model would enable governments or other relevant agencies to monitor information on Twitter more efficiently in order to rapidly respond to the emergencies.

## 2. Dataset Description

The main source of information is Kaggle's dataset on "Natural Language Processing with Disaster Tweets" (Kaggle, 2022). This dataset was originally created by the company Figure Eight (now known as Appen) (Appen, 2021). It provides an adequate amount of data to deploy NLP models (~ 11,000 tweets), which are ML techniques trained from unstructured, specifically text data.

The dataset obtained from Kaggle contains the following columns:

a) ***text:*** Contains the raw text extracted from the tweet to be further analyzed
b) ***location:*** Establishes the location at which the user was while sending the tweet

c) ***keyword:*** Provides the tweet's most relevant word, however, does not establish the criteria used to obtain this information
d) ***target:*** Whether a given tweet is about a real disaster or not (1 if yes and 0 if no)

For our study, we will mainly focus on the text feature in the dataset, as the keyword information are contained in the text and the locations are not the emphasis of this NLP study.

## 3. Exploratory Data Analysis

To verify the class imbalance of the target variable, we examine the label distribution of the target column and the results are shown in Figure 1.

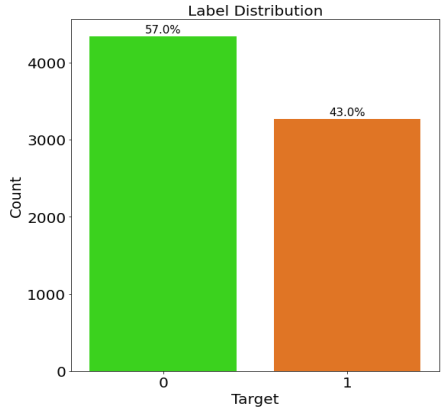


*Figure 1*. Class distribution of the dataset

From the distribution, we can see that the split between the two labels was around 57% to 43%, thus we would consider it as a balanced data set and no data sampling techniques are required before the modelling.

The distribution of the word counts per tweet is shown in Figure 2. It can be observed that the word counts of real and non-real disaster tweets are of a similar distribution. The median number of words is around 10 and the maximum number of words are around 25. The real disaster tweets have less occurrences of length shorter than 5. This observation could indicate that tweets of very short length, thus having limited information, are likely to be non-real disaster tweets.

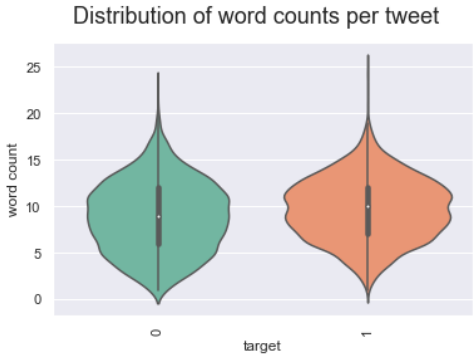


*Figure 2*. Distribution of word counts in tweets

We conduct a Part of Speech (POS) tagging to examine the grammatical composition of the tweets. Syntactic components are tagged using the spaCy pipline (spaCy, 2022), including the verbs, adjectives, nouns and proper nouns. The counts of the various components are shown in Figure 3. It can be observed that real and non-real disaster tweets have similar number of verbs and adjectives. However, interestingly, the real disaster tweets have on average slightly more nouns and proper nouns. It is another indication that real disaster tweets tend to have more information.

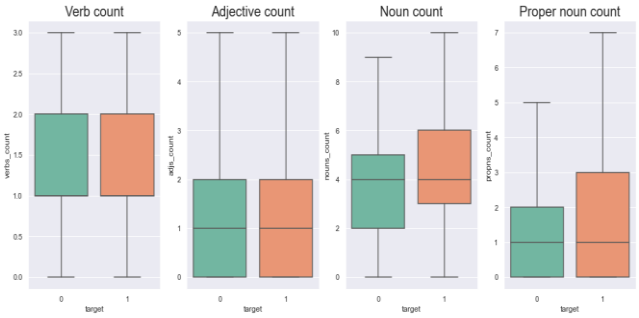


*Figure 3*. Distribution of the counts of grammatical components

To have a rough understanding on the differences between real and non-real tweets, in terms of the contents, we take a look at the word clouds, as shown in Figure 4 and Figure 5. We find that some frequent words in real disaster tweets, such as 'fire', 'storm', and 'Hiroshima', are able to pinpoint the type or location of the disasters and indicate that its content might be relevant to real disasters. Meanwhile, the frequent words in non-real disaster tweets, such as 'new' and 'time', are not referring to disasters. Therefore, the two types of tweets differ from each other in the lexicons used. We would expect a reasonable classification accuracy even with simple bag-of-word models.

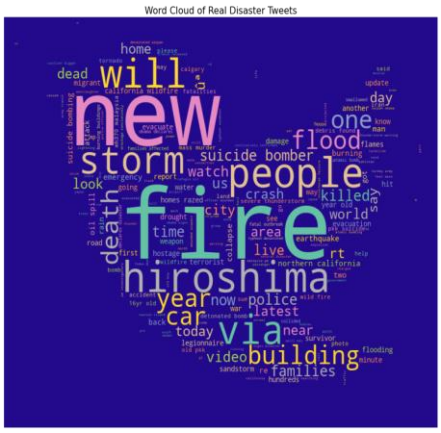


*Figure 4*. Word cloud of real disaster tweets

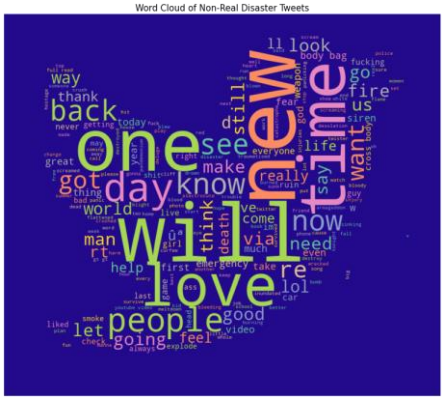


*Figure 5*. Word count of non-real disaster tweets

## 4. Data Pre-processing

### 4.1 Data Cleaning

This part is the preliminary data cleaning for Tweets content, and the operations to be carried out included:

- Remove URLs, special characters, digits, underline, and white spaces
- Make text lowercase
- Remove stopwords
- Correct the typos
- Remove the single letters

### 4.2 Word Lemmatization

Lemmatization here refers to, with the use of a vocabulary and morphological analysis of words, remove inflectional endings only and to return the base and dictionary form of a word. The WordNetLemmatizer in nltk library is utilized.

## 5. Modeling

### 5.1 Bag of Word (BOW) Models

#### 5.1.1 Motivation

Given the problem statement, it is not always clear whether the individual announcing the disaster through a mere tweet is real or not. The tweets contain a mix of spam and non-spam content, the automated filtering of which makes it an important application. The kind of application mentioned above urged us to conduct a binary text classification problem (i.e. there would be two outcomes of an event) starting with simple bag of word models as the baseline.

#### 5.1.2 Methodology

The data pre-processing step generated the cleaned version of the text taking into consideration removal of stopwords, digits, special characters, urls etc. Post this, we proceeded to convert the cleaned text into vector form through bag of word (BoW) representations, more specifically by the term frequency-inverse document frequency (TF-IDF) (Ultraviolet Analytics, 2018), which represents the score of the words in each tweet. The vector form of the text, thus, renders it suitable for further analysis and machine learning modeling.

The TF-IDF sparse vectors are used as the independent variables in the classification models. The candidate models chosen to be implemented were as follows:

- Logistic Regression (LR)
- Multinomial Naive Bayes (NB)
- Random Forest (RF)
- Support Vector Classifier (SVC)
- Light Gradient Boosting Machine (LGBM)

We selected three evaluation methods for each of the models used. The metrics were: accuracy, ROC_AUC score, and F1 score. ROC_AUC score provides the tradeoff between true positive rate and false positive rate, which would be relevant to the given problem as we are interested in finding how many real disaster tweets were actually classified as disasters by the prediction models. Additionally, F1-score which is based on the combination of two metrics, namely precision and recall, is particularly well-suited for the binary classification. The detailed rationale on the choice of metrics will be discussed in Section 5.5.

We used these metrics to compare the optimal performance of the machine learning classification models with one another.

#### 5.1.3 Results

A comparative analysis of the accuracy, AUC score and F1-score across five models used for text classification has been shown in Table 1 below:

*Table 1:* Test Accuracy, AUC & F1-Score of ML classification models

| MODEL | ROC AUC | ACCURACY | F1 | BEST |
|---|---|---|---|---|
| LR | 0.81 | 0.81 | 0.75 | x |
| NB | 0.80 | 0.80 | 0.74 | x |
| RF | 0.79 | 0.79 | 0.74 | x |
| SVC | 0.82 | 0.81 | 0.75 | ✓ |
| LGBM | 0.77 | 0.77 | 0.71 | x |

For the given binary classification problem, all the models have a fair performance, however Logistic Regression, Naive Bayes and SVC models perform relatively better in terms of accuracy, AUC score and F1 score. Ensemble models such as Random Forest and LightGBM have a slightly lower accuracy after tuning than the LR, NB and SVC models. This could be due to that the feature space is approximately linearly separable and thus the non-linear models do not offer a performance edge. Overall, the SVC performs the best and its confusion matrix is shown in Table 4.

*Table 4:* Confusion matrix of SVC model

| | PREDICT 0 | PREDICT 1 |
|---|---|---|
| TRUE 0 | 994 | 267 |
| TRUE 1 | 97 | 546 |

To understand which words in the corpus contribute the most to the classification, we plot the feature importance of the top 10 words from the RF results, as shown in Figure 6. We observe that words indicating the type of disasters are important, such as 'fire', 'bombing' and 'flood'. It is consistent with our observations in the EDA that the real disaster tweets can provide concrete information on the type of a disaster. We also note that the two location words 'Hiroshima' and 'California' are important features. That may be due to the two large disasters that are frequently mentioned in the tweets, the Hiroshima earthquake and the California wildfire.

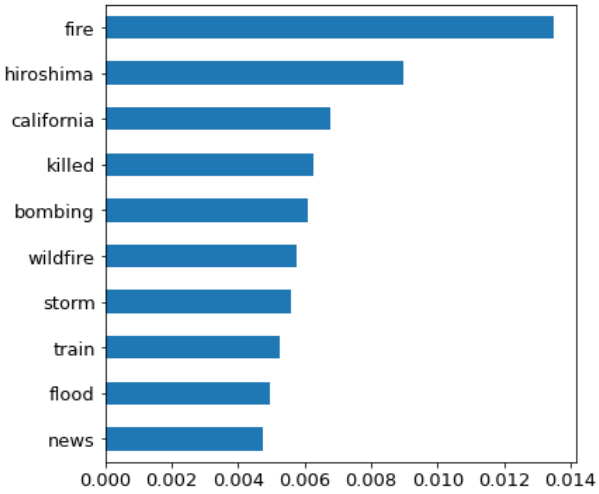


*Figure 6.* Feature importance of RF model

#### 5.1.4 STRENGTHS AND LIMITATIONS

We obtained a baseline accuracy of over 80% for three classification models, namely LR, SVC and NB, laying a solid foundation for this problem. While these models give a fairly decent accuracy, they are based on the BoW assumption that words are independent and no sequence information is capture. Therefore, we could strive to improve the prediction performance by using some state-of the art text classification techniques, namely BERT, TextCNN and GNN models, which will be discussed in the forthcoming sections.

### 5.2 BERT + Ensemble models.

#### 5.2.1 MODEL DESCRIPTION

The Bidirectional Encoder Representations from Transformers (BERT) model generates contextually based embeddings using bidirectional encoders from the transformer's NN architecture (Khalid, 2019). BERT models are particularly suitable for our disaster tweets classification, as it is able to capture the multiple sense of a word. For example, the word "fire" has different semantic meanings in a real disaster tweet "Fire in Jurong East" and a non-disaster tweet "I couldn't fire up my car". A static embedding will not differentiate the two interpretations. Thus, we would like to experiment and evaluate the performance of BERT embedding.

Another critical feature of why BERT outperforms other NLP techniques such as Bag of Words (BoW) is that it is pre-trained using a large corpus. As a second step, it can be fine-tuned to a specific dataset and a specific task such as classification or text creation.

#### 5.2.2 MODEL PRETRAINING

The model performs masking language model (MLM) and next sentence prediction (NSP) to understand the context of the text. Masking is a technique that trains the model to guess the right word in the blank, and the next sentence prediction technique tries to teach the model to recognize the context of the text and think about what sentence makes sense next.

The masking technique trains the model to use a bidirectional context-based approach, as opposed to other neural networks architectures that are unidirectional or traditional NLP techniques (BoW) that are context-free.

#### 5.2.3 Model Deployment

The models' input is the text tokenized after adding particular token embeddings such as CLS, a unique embedding for classification tasks, and SEP that helps understand the model at the end of each sentence.

Our team explores the BERT model as a preprocessing step to combine it with some ensemble models such as Light GBM, Gradient boosting, and Random forest. We use just CLS embeddings as an input for the models, but we set the max sentence length to 20 words due to computational limitations.

#### 5.2.4 Randomized Search Cross Validation

For the hyper-parameter tuning, we used Randomized Search CV to find the best possible model in contrast to grid search; this approach improves the efficiency by training just a sample of the possible combinations of the hyper-parameters.

*Table 2:* Best hyper-parameters by model

| Model | L. Rate | Max Depth | # Estimators | Best |
|---|---|---|---|---|
| RF | NA | 20 | 2500 | ✓ |
| LGBM | 0.3 | 12 | 2500 | x |
| GB | 0.3 | 12 | 1500 | x |

BERT combined with ensemble models improved some of the metrics, such as AUC, but the results were similar to the traditional models for some others, such as accuracy and F1 score.

*Table 3:* Test Accuracy, AUC & F1-Score of BERT classification models

| Model | ROC AUC | Accuracy | F1 | Best |
|---|---|---|---|---|
| RF | 0.81 | 0.76 | 0.70 | ✓ |
| LGBM | 0.80 | 0.74 | 0.68 | x |
| GB | 0.80 | 0.75 | 0.69 | x |

Among the models explored, the random forest has the highest performance in all the metrics. The confusion matrix of the RF model is shown in Table 4, where we can observe other metrics such as sensitivity and recall in more details.

*Table 4:* Confusion matrix of BERT+RF model

| | Predict 0 | Predict 1 |
|---|---|---|
| True 0 | 939 | 152 |
| True 1 | 299 | 514 |

### 5.3 BERT + Text CNN

#### 5.3.1 Text Convolutional Neural Network (CNN)

Text can be seen as a one-dimensional image, so that we can use one-dimensional convolutional neural networks to capture associations between adjacent words.

One of the important parts of Text CNN is the one-dimensional convolutional layer. Like a two-dimensional convolutional layer, a one-dimensional convolutional layer uses a one-dimensional cross-correlation operation. In this operation, the convolution window starts from the leftmost side of the input array and slides on the input array from left to right successively. When the convolution window slides to a certain position, the input subarray in the window and kernel array are multiplied and summed by element to get the element at the corresponding location in the output array, as illustrated in Figure 7.

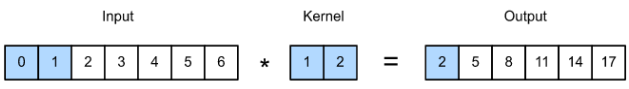


*Figure 7*. Convolution of Text CNN

Similarly, we have a one-dimensional pooling layer. The max-over-time pooling layer used in Text CNN actually corresponds to a one-dimensional global maximum

pooling layer. Assuming that the input contains multiple channels, and each channel consists of values on different time steps, the output of each channel will be the largest value of all time steps in the channel. Therefore, the input of the max-over-time pooling layer can have different time steps on each channel.

#### 5.3.2 TEXT CNN MODELING

We implement the previously discussed Text CNN on our disaster tweet classification problem. Before we feed the data to the text CNN model, we use a BERT transformer to convert our data to a 3D tensor. Different from Section 5.2, here we will use all the BERT last layer hidden outputs of each word.

Due to the computational power limitation, we have to limit the max sentence length, in this case, we choose the average sentence length, which is 32. We do padding for each text and feed it to the BERT transformer to get the output. Still because of the memory size issue, we cannot save all 768 dimensions of each word. So, we choose the last 64 dimensions and each word is converted to a 64-dimension embedding vector.

Now, we construct a text CNN model. First connect the input to four 1D convolutional layers with filter size of 128, with kernel size of 2, 3, 5 and 7 respectively. Then connect all the output to max pooling layers and concatenate them together. The output are then connected to a six-layer fully-connected neural network with drop out and batch normalization. The structure of this model is shown in the Appendix.

#### 5.3.3 TEXT CNN RESULTS

At about 150 epochs, the train and test accuracies are stable. The train/test accuracy at every epoch is plotted in Figure 8.

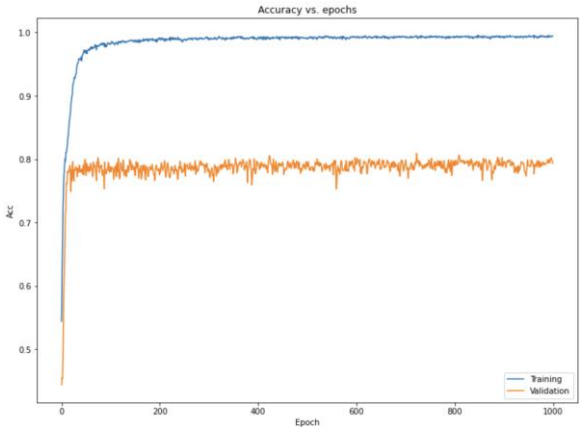


*Figure 8*. Train/test accuracy of BERT+Text CNN model

We use the best Text CNN model to get the test performance, which achieves an accuracy of 0.7957, ROC AUC score of 0.8500, and a f1 score of 0.7482. The confusion matrix is shown in Table 5.

*Table 5:* Confusion matrix of BERT+Text CNN model

| | PREDICT 0 | PREDICT 1 |
|---|---|---|
| TRUE 0 | 937 | 154 |
| TRUE 1 | 235 | 578 |

### 5.4 Graph Convolution Network (GCN)

Graph neural network (GNN) has attracted increasing attention as a method of graph analysis in many domains such as social network, knowledge graph etc. GNN is able to capture the dependencies between graph nodes, and also preserve the global structure information of a graph in the embeddings. As the state of art, GNN has been applied to text classification problems, where the training corpus is used to build graph representations of vocabulary and documents. In this study, we explore this novel method for the binary disaster tweet classification problem.

#### 5.4.1 Graph

For our study, we implemented the graph convolution network (GCN) text classification method proposed by Yao et al. (2019). The graph is built using the entire corpus where the nodes are the unique words and the documents with the training documents labeled and test documents unlabelled, as illustrated in Figure 8. The document-word edges represent the word occurrence in the documents, and its weight is calculated using the term frequency-inverse document frequency (TF-IDF) of a word in a document. These edges capture the semantics of the documents i.e., the tweets. The word-word edges represent the co-occurrence of words, where the weights are calculated by the pointwise mutual information (PMI) with a sliding window size 10. Words that are highly correlated in semantics will have a higher weight on the edge between them. The PMI value of a word pair *i,j* is computed as

$$PMI(i,j) = log\frac{p(i,j)}{p(i)p(j)}$$

$$p(i,j) = \frac{\#W(i,j)}{\#W}, p(i) = \frac{\#W(i)}{\#W}$$

where *#W(i,j)* is the number of windows containing word *i* and word *j*, *#W(i)* is the number of windows containing word *i*, *#W* is the total number of windows in the entire corpus.

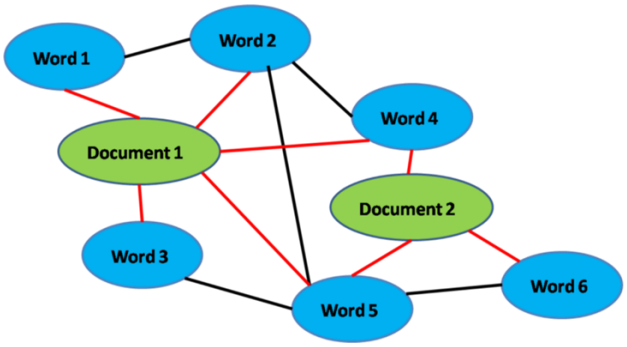


*Figure 8.* Graph built on documents and vocabulary with document-word edges (red) and word-word edges (black)

#### 5.4.2 Convolution Neural Network

Unlike the pixels in image data, graph nodes do not have a structured spatial relationship. To perform convolutions on the graph, we implement the two-layer GCN proposed by Kipf and Welling (2016), which generates, directly from the graph, embedding vectors for the nodes based on their neighborhoods. The embeddings are then fed into a softmax classifier for the text classification. The embeddings output from the second layer of the GCN is

$$L^{(2)} = \tilde{A}ReLU(\tilde{A}XW_0)W_1$$

$$\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

where $A$ is the adjacency matrix of the graph, $D$ is the degree matrix of $A$ ($D_{ii} = \sum_j A_{ij}$), $X$ is the diagonal square matrix of ones of the dimension of the number of nodes, $W_0$ and $W_1$ are the weight matrix of the first and second layer respectively.

Yao et al. (2019) suggest that two hidden layers GCN allows the information to be passed to nodes that are two steps away, which enable the information exchange between documents even though there are no document-document edges in the graph. It is also reported that more layers beyond two does not improve the classification performance. Therefore, we set a two-convolution-layer structure, with the layer size of 330 and 130 respectively.

#### 5.4.3 GCN Results

The previously mentioned GCN is implemented for the disaster tweet classification in this study. One important observation is that the graph size is huge if built on the entire dataset. The number of edges is roughly of the order of *[#doc*#vocab+0.5*(#vocab)$^2$]*. Due to the memory constraint, we use a random sample of 1000 data points from the dataset. And 75% of documents are used as the training nodes, 25% as the testing nodes.

We train the model for 1000 epochs, and the train/test accuracy is plotted below in Figure 9. The best accuracy, ROC_AUC and F1 score of the model are 0.61, 0.63 and 0.58 respectively. The accuracy of the model is not as good as the other models we examined previously. It could be due to the small fraction of the data we used for training.

The graph needs to be generated every time new test data are fed in for classification. This process takes long and would result in high latency if deployed to real applications. Together with the large memory requirement for the graph, we may conclude that the GCN method is suitable for the text classification tasks where the available training data are limited. If resources allow, further experiments with the whole dataset shall be conducted to gauge its performance against other models.

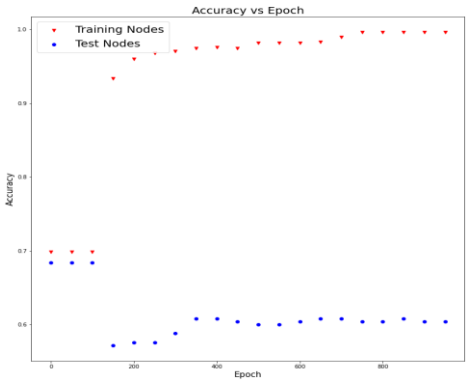


*Figure 9.* Cross entropy loss vs. Epoch (above), Train/Test accuracy vs. Epoch (below)

### 5.5 Model evaluation

For this study, we evaluate the model performance based on the accuracy, ROC_AUC score and F1 score. Accuracy provides a simple and classic measurement of the binary classifiers' overall performance. However, it is biased towards the majority class. ROC_AUC offers an evaluation of the model independent of the class distribution and is thus chosen as another metric.

Normally, the choice of emphasis on the model precision or recall will be based on the application and the tradeoff between Type I and Type II error cost. For example, in our case, if the purpose of the application is to identify as many potential disasters as possible and the manual screening of false positive tweets cost little, the model evaluation shall focus on the recall. On the contrary, if the model is deployed with minimum human screening, it would be beneficial to have a high-precision model to reduce the cost of the false alarms. In this study, as the model could be potentially applied to both scenarios, we use F1 score, the harmonic mean of precision and recall, as the metric to achieve a balance.

The performance of the models examined are summarized in Table 5.

*Table 5*. Summary of model performance.

| MODEL | ROC AUC | ACCURACY | F1 |
|---|---|---|---|
| TF-IDF+LR | 0.81 | 0.81 | 0.75 |
| TF-IDF+NB | 0.80 | 0.80 | 0.74 |
| TF-IDF+RF | 0.79 | 0.79 | 0.74 |
| TF-IDF+SVC | 0.82 | 0.81 | 0.75 |
| TF-IDF+LGBM | 0.77 | 0.77 | 0.71 |
| BERT+RF | 0.81 | 0.76 | 0.70 |
| BERT+LGBM | 0.80 | 0.74 | 0.68 |
| BERT+GB | 0.80 | 0.75 | 0.69 |
| BERT+TEXT CNN | 0.85 | 0.80 | 0.75 |
| GCN* | 0.63 | 0.61 | 0.58 |

*Built on an undersampled dataset

Overall, when we look at the prior-independent metric ROC_AUC, the BERT+Text CNN model performs best as we expected, due to its contextual embedding. However, in terms of accuracy and F1 score, the BOW models outperform BERT marginally. That could potentially be due to the slight class imbalance in the dataset.

With adequate resources, the BERT models could be further improved by fine tuning the parameters instead of using the fixed pre-trained ones. Additionally, the performance may improve if we use the complete embedding vectors instead of the truncated ones.

## 6. Application

The deployment of the model could potentially help emergency response agencies to monitor social media to identify disasters and collect anecdotal reports, which could contain crucial insights for planning the response strategies. The detected tweets are likely to offer temporal and spatial details about a tragedy and may be used to guide aids, rescues, and restorations.

Additionally, the model might create business insights as well. For example, news companies could potentially utilize it to obtain first-hand information about disasters to stay ahead of competitors. Also, the classification models that we explored can be combined with text generator models, such as GPT3, to inform the people on social media without any human supervision, increasing the efficiency of the communication industry.

Lastly, since the police, firefighters, and government agencies have limited staff, we can combine in a first step to identify real disaster information and then automatically perform sentiment analysis to allocate the human resources as efficiently as possible.

## 7. Conclusion

We started defining a baseline performance using traditional machine learning models and BoW embeddings, which achieves a decent performance. In the second stage, we explored the BERT model with contextual information that improved some of the metrics, such as ROC_AUC, but the results are similar to the traditional models for some others, such as accuracy.

Another novel technique we explored is Graph Convolution Network (GNC). This model also underperforms compared to the baseline. Still, it is worth noting that although the GNC and BERT models can capture more sophisticated relationships in the text, we face computational limitations that explain the underperformance of the models. The GNC use a small portion of the dataset for training, and with the BERT models, we have to limit the size of the embeddings considerably and cannot fine-tune the model due to memory constraints.

The explored models provide feasible solutions for disaster tweets identification on social networks, which see promising applications for emergency responses.

## References

*Datasets resource center*. Appen. (2021, March). Retrieved April 24, 2022, from https://appen.com/datasets-resource-center/

*English · spacy models documentation*. spaCy. (2022). Retrieved April 24, 2022, from https://spacy.io/models/en

Khalid, S. (2019, November). *Bert explained: A Complete Guide with Theory and tutorial*. Medium. Retrieved April 24, 2022, from https://medium.com/@samia.khalid/bert-explained-a-complete-guide-with-theory-and-tutorial-3ac9ebc8fa7c

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

*Natural language processing with disaster tweets*. Kaggle. (2022). Retrieved April 24, 2022, from https://www.kaggle.com/c/nlp-getting-started/overview

*TF-IDF Basics with pandas and Scikit-Learn*. Ultraviolet Analytics. (2018). Retrieved April 24, 2022, from http://www.ultravioletanalytics.com/blog/tf-idf-basics-with-pandas-scikit-learn

Yao, L., Mao, C., & Luo, Y. (2019). Graph convolutional networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence, 33*, 7370–7377. https://doi.org/10.1609/aaai.v33i01.33017370

## **Appendix –** BERT+Text CNN Structure