**DISASTER DETECTION SYSTEM**

*USING REAL TIME ANALYSIS OF TWITTER DATA*





*Applied Data Science*

*CUSP New York University*

# Table of Contents

[Table of Contents 2](#_heading=h.x0rkm26mrjco)

[List of Figures 2](#_heading=h.1t3h5sf)

[List of Tables 3](#_heading=h.4d34og8)

[1.](#_heading=h.26in1rg) Abstract 4

[2.](#_heading=h.gq574pkriwss) Problem Statement 4

[3.](#_heading=h.lnxbz9) Introduction 4

[4.](#_heading=h.35nkun2) Background 4

[5.](#_heading=h.1ksv4uv) Data 5

[5.1.](#_heading=h.44sinio) Data collection 5

[5.2.](#_heading=h.z337ya) Data processing 5

[5.3.](#_heading=h.3j2qqm3) Data challenges 6

[6.](#_heading=h.4i7ojhp) Methodology 6

[6.1.](#_heading=h.3whwml4) TF-IDF 8

[6.2.](#_heading=h.qsh70q) Bag-of-Words 9

[6.3.](#_heading=h.3as4poj) Word2Vec 9

[6.4.](#_heading=h.49x2ik5) Seq2Seq 9

[6.5.](#_heading=h.147n2zr) Transformer XL 10

[6.6.](#_heading=h.ihv636) BERT (Bidirectional Encoder Representations for Transformers) 11

[7.](#_heading=h.agememrpvz1n) Results 12

[7.1.](#_heading=h.41mghml) TFIDF + Classifier 12

[7.2.](#_heading=h.vx1227) Bag of Words 13

[7.3.](#_heading=h.1v1yuxt) Word2Vec 13

[7.4.](#_heading=h.2u6wntf) Transformer-XL 13

[7.5.](#_heading=h.3tbugp1) BERT 14

[8.](#_heading=h.v8esro3qa83k) Conclusion 14

[9.](#_heading=h.kle3ozg7i9qa) Future Work 15

[10.](#_heading=h.37m2jsg) References 15

[11.](#_heading=h.1mrcu09) Team Contribution 16

# List of Figures

[Figure 1 Using Tweepy to access twitter data 5](#_heading=h.2jxsxqh)

[Figure 2 Examples of tweets containing information on disaster 6](#_heading=h.1y810tw)

[Figure 3 Project Methodology 7](#_heading=h.2xcytpi)

[Figure 4 NLP methods to capture disaster detection 7](#_heading=h.1ci93xb)

[Figure 5 TF-IDF model architecture](about:blank) 8

[Figure 6 Importance of words with weights in TFIDF 8](#_heading=h.2bn6wsx)

[Figure 7 Continuous Bag of Words (CBOW) and Skip-gram model architecture. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word 9](#_heading=h.1pxezwc)

[Figure 8 Seq2Seq Model Architecture 10](#_heading=h.2p2csry)

[Figure 9 Example of word training and evaluation in Vanilla Transfomer 10](#_heading=h.3o7alnk)

[Figure 10 Illustration of model evaluation beyond fixed length in transformer XL model 11](#_heading=h.23ckvvd)

[Figure 11 BERT Model Architecture](about:blank) 11

[Figure 12 BERT model architecture in tweet data 12](#_heading=h.32hioqz)

[Figure 13 Confusion Matrix TFIDF model (a) Gaussian Naive Bayes Classifier (left) (b) RF classifier (right) 12](#_heading=h.2grqrue)

[Figure 14 Bag of words Confusion Matrix ((a) LogReg Classifier (left) (b) RF classifier (right)) 13](#_heading=h.3fwokq0)

[Figure 15 Word2Vec Confusion Matrix (a) LogReg Classifier (left) (b) RF classifier (right) 13](#_heading=h.4f1mdlm)

[Figure 16 Tranformer XL model confusion matrix (a) SVC Classifier (left) (b) RF Classifier (right) 13](#_heading=h.19c6y18)

[Figure 17 BERT model confusion Matrix 14](#_heading=h.28h4qwu)

[Figure 18 Visualization for identified disaster using BERT model 14](#_heading=h.nmf14n)

# List of Tables

[Table 1 Accuracy scores using different models 12](#_heading=h.1hmsyys)

# Abstract

Natural disasters or crises are unexpected and come with little warning. They are capable of causing notable destruction causing injuries and death– acting promptly is crucial for the protection of people and their properties. This paper is aggregating more than 50,000 tweets to further train and validate the model [16], comparing different machine learning approaches to find the most accurate results for real-time prediction: TF-IDF, Seq2Seq, BERT, Word2Vec, and Transformer-XL. Each of the models are utilized to compute the target variable (1 for disaster and 0 for non-disaster) on the test set based on the training set, each portraying accuracy of results. The BERT model for example, has been able to achieve 93% classification accuracy on predicting the target variable from our dataset. This paper will include multiple forms of visualizations and graphs such as the loss and accuracy curve and flow charts to further aid in understanding the concepts of models and results.

Keywords: Disaster Detection; Deep Learning; Twitter Monitoring; Sentiment Analysis (NLP)

# Problem Statement

A data science approach to build a machine learning classifier model to predict which tweets are about ‘real disasters’ and which tweets are not.

# Introduction

Social media 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 over social media. Apps such as twitter are particularly helpful because people can easily share short [6], real time text statements about ongoing situations. Because of this, disaster relief organizations and news agencies are increasingly interested in programatically monitoring Twitter. However, simply finding keywords in the twitter text is not always enough to properly identify disasters because the full meaning of a word often depends on its relationship with other words in the sentence.

We seek to solve this problem by using machine learning on tweeted text to quickly identify and respond to disasters. While there are limits to this approach, such as the tendency for overreliance on a tool like this to prioritize those that use the internet over those that do not, it is still a valuable tool as both an early warning system and a way to find the scope of a disaster. By having a reliable alert system processed from social media data, the government can move quicker to allocate their resources to the area that needs it the most, reducing casualties and saving lives.

# Background

Natural disasters can often overload, and sometimes even break down traditional means of contacting first responders. A 2018 research paper found that even in disaster zones, the internet was not only still available, but also used by people being directly affected by the disaster seeking help [1]. There are several opportunities to use technology to improve how we currently respond to disasters, however currently researchers are still looking for the correct model to most accurately and efficiently parse available data.[2] With the right models, it could be possible to significantly improve feature extraction from tweets.[3]

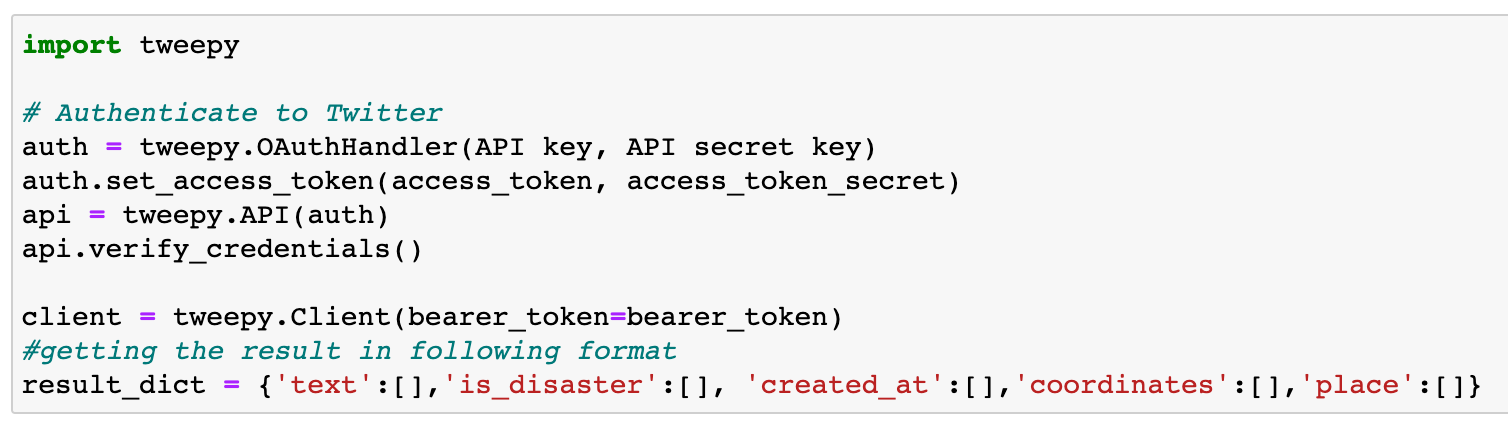
Our paper hopes to contribute to the literature on this topic and provide an exploration of different methods using text analysis to use social media for good.

# Data

## Data collection

For this project, we have analyzed text from tweets to understand if the content reflects an active disaster event. To train the model, we need a supervised dataset with each tweet labeled [5] as disaster (1) or not disaster (0). Deep learning models require a large amount of training data which makes manual tagging of every tweet by the research team infeasible. Kaggle has a training dataset from a playground competition with ~7600 labeled tweets for disaster content. In addition, we manually tagged about 2500 tweets to create a train set of 10,000 tweets [15]. We used 75% of the dataset for training and 25% for model validation. For testing dataset we tapped into live twitter streams for the past month (March-April 2022) and sampled 10,000 tweets (out of >1 million) catered to disaster related hashtags. The positively identified tweets were visualized in the dashboard as output. Disaster relief organizations, news agencies, fire department, police department etc. would be the main users of this kind of application

For scraping the data from twitter, we used the Tweepy python library. Tweepy is a Python library for accessing the Twitter API [6]. It is great for simple automation and creating twitter bots. It helps us to smoothly and transparently access Twitter's endpoints such as likes, retweets, tweets, etc. The following code snippet is used to scrape the data from twitter:



*Figure 1 Using Tweetpy to access twitter data*

## Data processing

Before starting any NLP project, text data needs to be preprocessed to convert it into a consistent format. Text will be cleaned, tokenized and converted into a matrix [9]. Here are the steps that were followed for data cleaning and formatting:

* Text all lower or uppercase
* Removing Noise
* Tokenization
* Stopword Removal
* Stemming
* Lemmatization

## Data challenges

Context of the tweet is very important for text classification. Often the presence of important words can influence the predictions of a machine learning model, but the context in which those words were used might be completely unrelated [7]. Below are a few examples of tweets containing disaster words, but used in different context:

| Graphical user interface, text, application  Description automatically generated |  |
| --- | --- |
|  | |

*Figure 2 Examples of tweets containing information on disaster*

The challenge in this situation is the possibility of high “False Positives” which often trigger false alarms for disaster relief agencies. The goal is to reduce the false positives and get a higher accuracy in identifying an actual disaster. The crowdsourced verification of identified signals based on a consensus of people within similar geographic locations can help reduce false alarms.

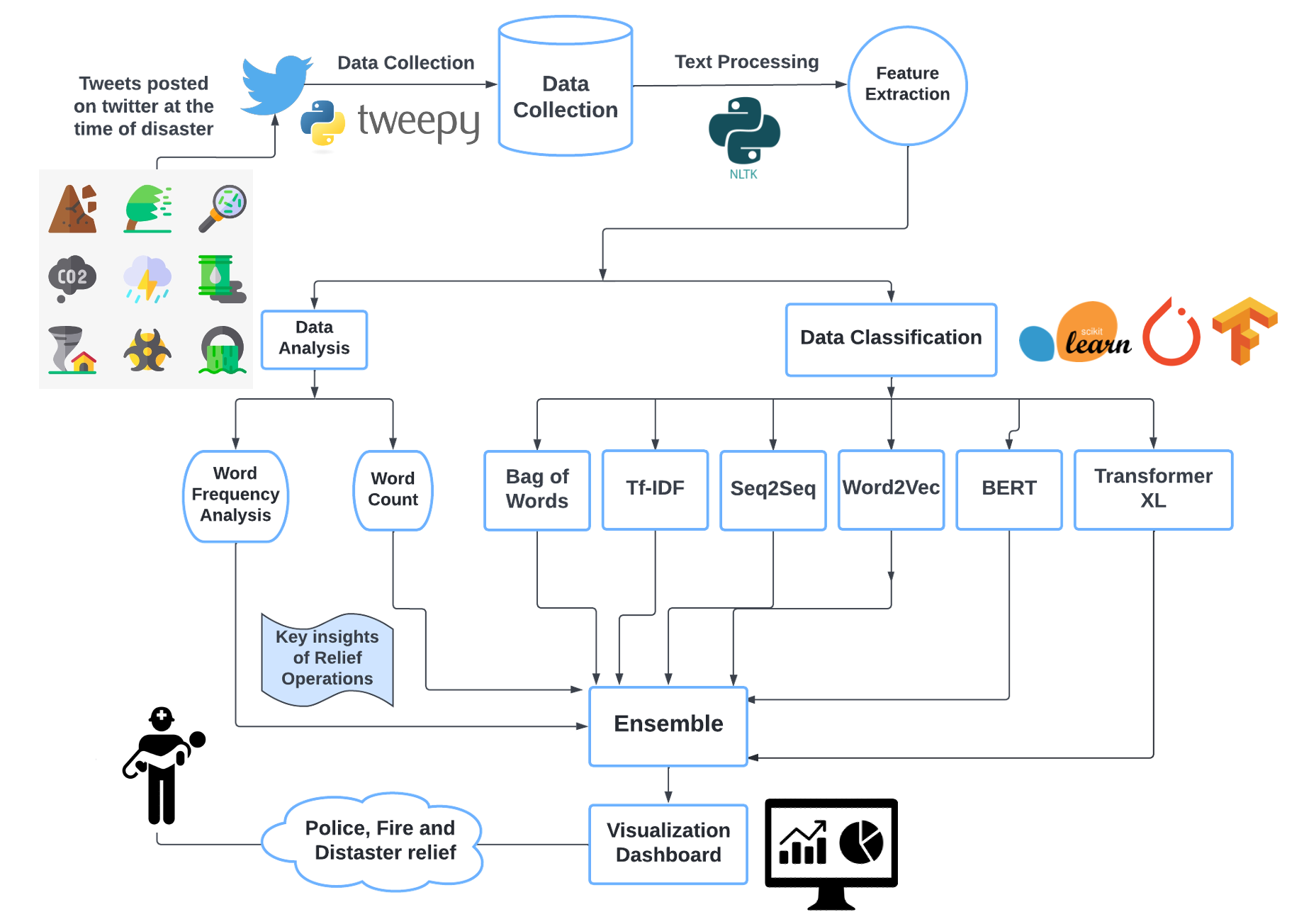
# Methodology

Our broad methodology can be divided into three steps:

* Data collection
* Data modeling
* Visualization

We have already covered the data collection, cleaning, and preprocessing steps in above section. Different text classification models were tested to get the model with best performance and ensembling the predictions of the tested models was performed to get higher accuracy.

Lastly, the trained model was used to make predictions on the test tweets and identify disaster alerting tweets[14]. Using the geo-tag of tweets and intensity of identified disaster based on number of people/tweets, an intensity hotspot map was visualized in a KeplerGL dashboard. This dashboard can be monitored by disaster relief, fire and police departments for quicker response. The process flow is outlined in figure below.



*Figure 3 Project Methodology*

Text classification has been studied thoroughly in literature and the goal was to capture the wide spectrum of different methodologies. The chosen models belong to three categories based on type of modeling technique:

1. Classical models (Word frequency based)

2. Embedding models

3. Advanced models (Transformers based)

Diagram

Description automatically generated with medium confidence

*Figure 4 NLP methods to capture disaster detection*

Based on research, 2 models were picked for experimentation from each of the above three categories. The figure below depicts the model hierarchy and chosen models for this exercise.

Next subsection outlines the details, differences, and nuances of each text classification methodology.

## TF-IDF

TF-IDF (Term-Frequency – Inverse Document Frequency) is a technique to quantify words in a set of documents. It evaluates and calculates how relevant each word or term that occurs in text, used for information retrieval and text mining. The product of how many times a word appears in a document (TF) and the inverse document frequency of the word across documents (IDF) make up the TF-IDF weight of the term(s).

In order to take a closer look at what TF and IDF looks like in the training portion, we will provide examples of IDF weight and TF-IDF transformation. IDF is a weight indicating how commonly a word is used, and TF-IDF transformation represents the importance of the word in a particular document (Train.csv in our example here).

|  |  |
| --- | --- |

*Figure 6 Importance of words with weights in TFIDF*

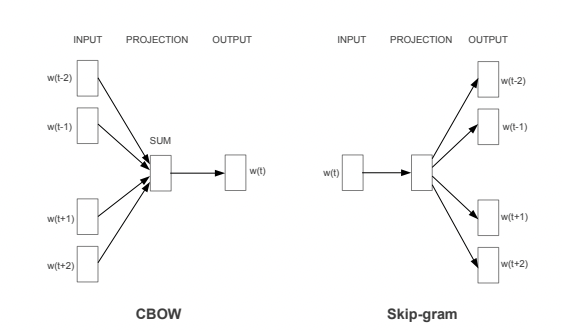
## Bag-of-Words

A model that represents text in a multiset of its words, and it is mainly used as a tool for feature generation. The most common feature calculated from the Bag-of-words model is term frequency, which calculates the number of words appearing in a text in the process referred to as vectorization[8].

The steps for the vectorization process is determining the vocabulary from a text document. Then, to vectorize the document we need to count how many times each word appears in the document based on the vocabulary that has been made before.

## Word2Vec

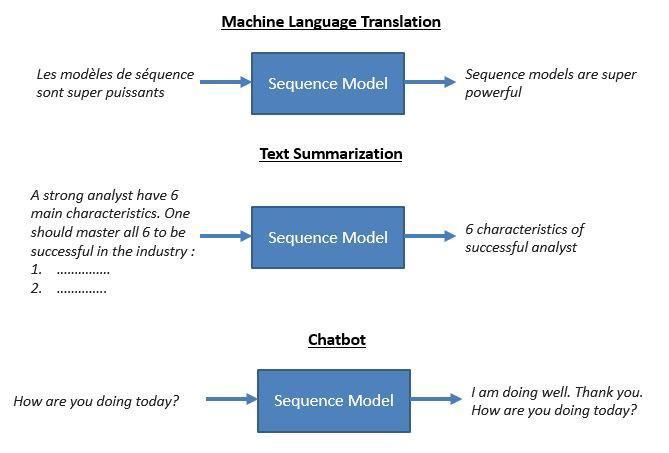
A technique for natural language processing that uses a neural network model to learn word associations from a corpus of text. It combines a continuous bag of words and a skip-gram model [3]. The model can detect synonymous terms and represent each word with a specific list of vectors. Which later also indicates the semantic similarity between the words represented by those vectors.



*Figure 7 Continuous Bag of Words (CBOW) and Skip-gram model architecture. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word*

## Seq2Seq

The Sequence-to-sequence model uses recurrent neural networks to take an input and generate a matching output. A common use case is with automated chatbots, such as those used by companies to answer their customer’s frequently asked questions in real time.



*Figure 8 Seq2Seq Model Architecture*

## Transformer XL

The TransformerXL model was proposed as an ‘Attentive’ language model beyond fixed-length context like one used in vanilla transformers. TransformerXL with relative positioning uses embeddings which are previously computed hidden-states to attend to longer context. The Transformer XL is a new architecture that enables language understanding beyond a fixed length context. Unlike the other transformer model, it can capture long-term dependencies beyond fix length which are the main limitations of vanilla and BERT transformers.

Chart, scatter chart

Description automatically generated

*Figure 9 Example of word training and evaluation in Vanilla Transfomer*

Transformers language processing was introduced with a fixed length text. It splits the inputs into segments or sentences and trains the model within each segment. Therefore, they fail to capture long-term dependencies beyond a specified fixed length. The evaluation makes one prediction at one position at a time and shifts one position at each segment and the segment is processed from scratch. TransformerXL addresses this limitation by improving the computational time.

There are two methods to train a transformerXL model namely segment-level recurrence and relative positional encoding. Segment-level recurrence uses the hidden-state sequence computed in the previous segment fixed and cached to be reused in the training. This segment-level recurrence mechanism improves the evaluation speed because it can advance by an entire long segment without recomputation [10].

Relative Positional encoding employs novel relative positional encoding which is different from other approaches that incorporate bias in initial embedding. The use of fixed embeddings which are learnable transformations makes it more intuitive and generalizable to learn longer sequences. Till date the transformer XL model has been trained on WikiText-103, text8, enwik8, One Billion Word and Penn Treebank datasets. The WikiText-103 and one Billion word model outperform other models in terms of learning long term dependencies [11]. The WikiText-103 TransformerXL training model is used to make the disaster predictions in the dataset.

Chart, scatter chart

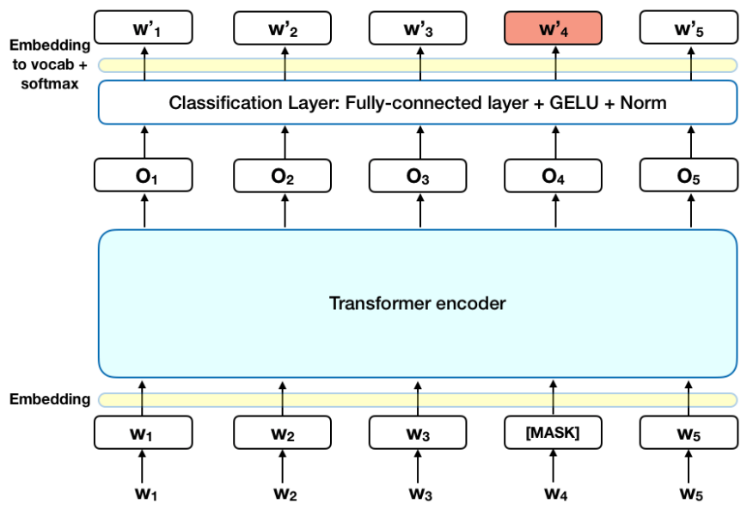
Description automatically generated

*Figure 10 Illustration of model evaluation beyond fixed length in transformer XL model*

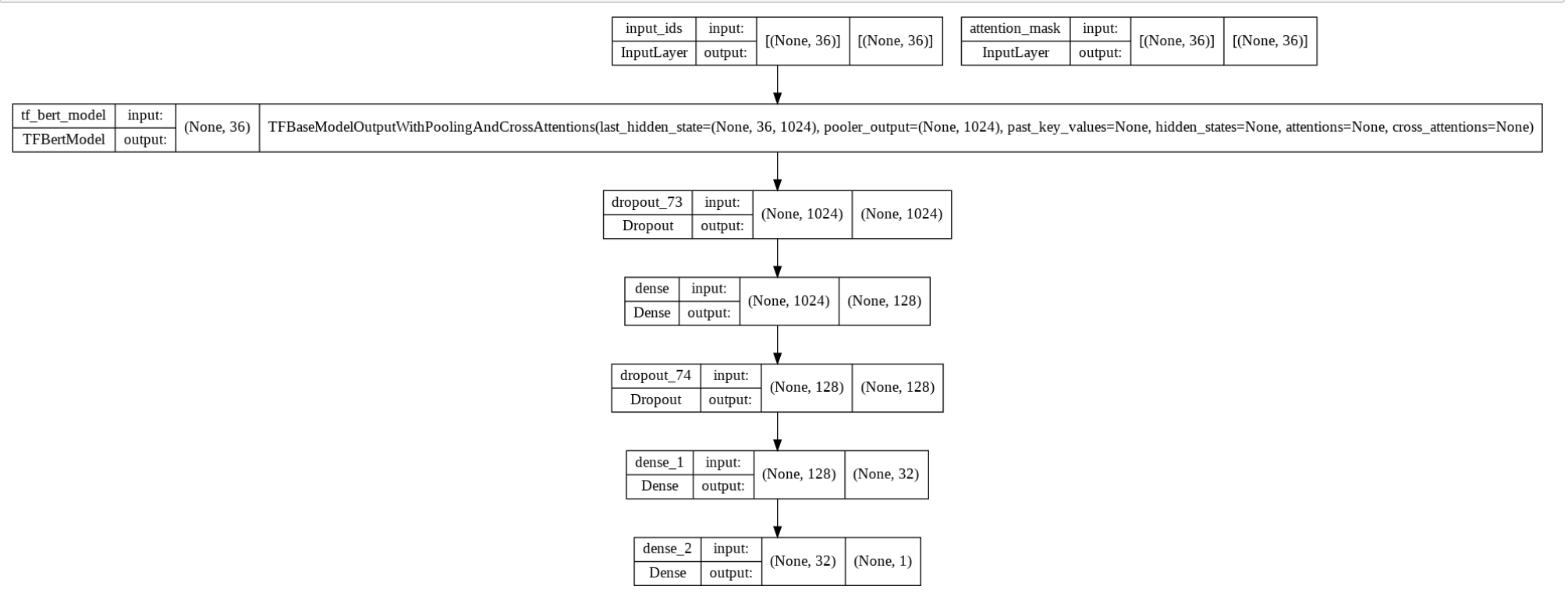
## BERT (Bidirectional Encoder Representations for Transformers)

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. The model has achieved state-of-the-art results in a variety of NLP tasks such as Question Answering, Natural Language Inference and others. The technical innovation in this model is the application of bidirectional training of Transformer to language modeling. This is new in comparison to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training.

The model architecture is shown below:

BERT models are usually pre-trained on a large corpus of text, then fine-tuned for specific tasks [9]. The application of BERT to our use case requires text preprocessing as an initial step.

The preprocessing task results in 3 outputs: *(input\_words\_id, input\_mask and input\_type\_ids)*. After passing the pre-processed data through the BERT model, we get a 1024-dimension vector for each input text. We build a classifier network with dense and dropout layers to process the BERT outputs and perform the classification task. Our model architecture can be seen below:

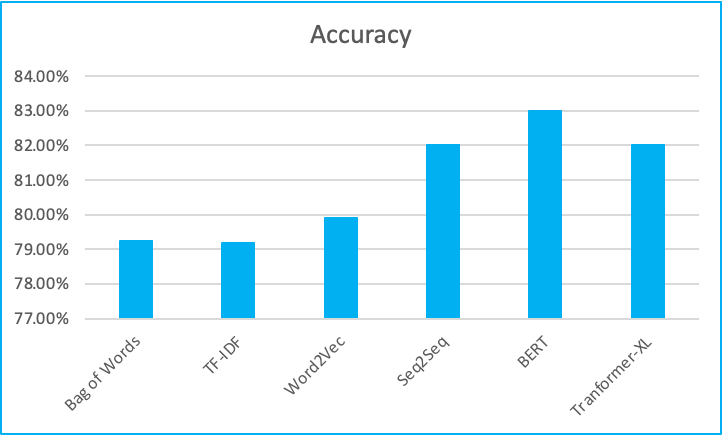
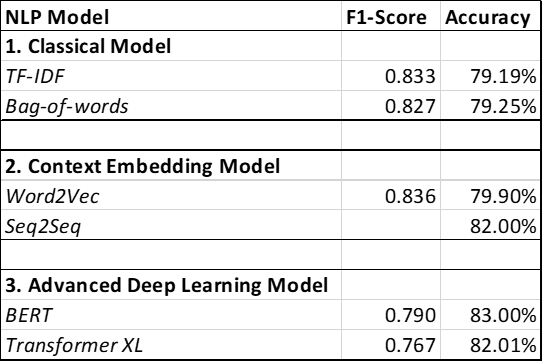


*Figure 12 BERT model architecture in tweet data*

# Results

This section outlines the results on the hold out test set achieved by each model. For this task, accuracy and F1 score have been used for reporting model performance. We also plot the confusion matrix for each model to understand the false positive rate (FPR) and False negative rate (FNR).

*Table 1 Accuracy scores using different models*



Confusion matrix for each model have been plotted below:

## TFIDF + Classifier

|  |  |
| --- | --- |

*Figure 13 Confusion Matrix TFIDF model (a) Gaussian Naive Bayes Classifier (left) (b) RF classifier (right)*

## Bag of Words

|  |  |
| --- | --- |

*Figure 14 Bag of words Confusion Matrix ((a) LogReg Classifier (left) (b) RF classifier (right))*

## Word2Vec

|  |  |
| --- | --- |

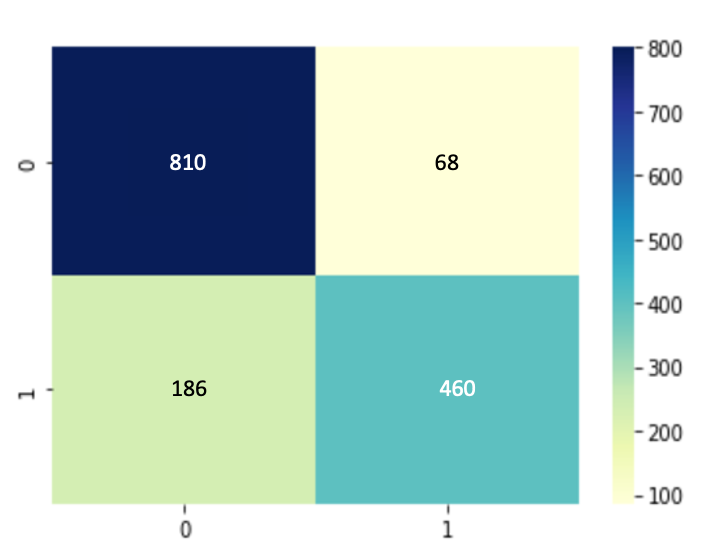
*Figure 15 Word2Vec Confusion Matrix (a) LogReg Classifier (left) (b) RF classifier (right)*

## Transformer-XL

| Chart, treemap chart  Description automatically generated | Chart, treemap chart  Description automatically generated |
| --- | --- |

*Figure 16 Tranformer XL model confusion matrix (a) SVC Classifier (left) (b) RF Classifier (right)*

## BERT



*Figure 17 BERT model confusion Matrix*

Based on the above results, BERT model has the highest accuracy at 83.3% for the task of disaster identification from tweet text. Below is the dashboard visualization for identified disasters.

****

*Figure 18 Visualization for identified disaster using BERT model*

The visualization dashboard helps understand location and type of disaster along with severity (a) Earthquakes (*top-left)* (b) Explosions *(top-right)* (c) Icequakes *(bottom-left)* (d) Quarry blasts *(bottom-right)*

# Conclusion

After performing the analysis using different types of models, like traditional word frequency-based models, context embedding models, and advanced deep learning models we can conclude that the BERT model outperformed other models with the highest accuracy score compared to another model that we’ve been tested with 83 % accuracy, followed by the Transformer XL and Seq2Seq with 82% accuracy. By using this model we are able to map the area in time of crisis, know where to prioritize the limited resources of first responders, and evaluate rescue performance by comparing the disaster tweet post-event.

# Future Work

We recommend adding more data sources such as other social media platforms, local channels, and news outlets to get more accurate data. Improving the methodology by exploring learning models other than classical, embedding, and deep learning models already covered in this study. Other learning models such as Convolutional Neural Network (CNN), Recurrent Neural Networks (RNN), and other advanced models for sentiment analysis. We also suggest collaborating with local governments and civil organizations to improve the real-time feed.

# References

1. Jahanian, Mohammad, et al. "The evolving nature of disaster management in the internet and social media era." 2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN). IEEE, 2018.
2. Roth, Florian, and Tim Prior. Volunteerism in Disaster Management: Opportunities, Challenges and Instruments for Improvement. ETH Zurich, 2019.
3. Sentiment Analysis and Ontology Engineering, Springer (2016), pp. 313-340
4. Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).
5. Dhillon, H.S.; Huang, H.; Viswanathan, H. Wide-area wireless communication challenges for the Internet of Things. IEEE Commun. Mag. 2017, 55, 168–174.
6. Hayashi, N.J.; Ott, E.S., IV; Tsang, A.Y.; Fukuda, M.; Wascovich, D.; Quoc, M. Multimedia Sharing in Social Networks for Mobile Devices. U.S. Patent No. 8,046,411, 25 October 2011.
7. Kaplan, A.M.; Haenlein, M. Users of the world, unite! The challenges and opportunities of Social Media. Bus. Horiz. 2010, 53, 59–68.
8. García-Palomares, J.C.; Salas-Olmedo, M.H.; Moya-Gómez, B.; Condeco-Melhorado, A.; Gutierrez, J. City dynamics through Twitter: Relationships between land use and spatiotemporal demographics. Cities 2018, 72, 310–319.
9. Sagl, G.; Resch, B.; Hawelka, B.; Beinat, E. From social sensor data to collective human behavior patterns: Analyzing and visualizing spatio-temporal dynamics in urban environments. In Proceedings of the GI\_Forum 2012: Geovisualization, Society and LearningGI-Forum, Salzburg, Austria, 3–6 July 2012; Herbert Wichmann Verlag: Berlin, Germany, 2012; pp. 54–63.
10. Aggarwal, C.C.; Abdelzaher, T. Social sensing. In Managing and Mining Sensor Data; Springer: Boston, MA, USA, 2013; pp. 237–297.
11. Aggarwal, C.C. (Ed.) Managing and Mining Sensor Data; Springer Science & Business Media: Berlin, Germany, 2013.
12. Abdelzaher, T.; Anokwa, Y.; Boda, P.; Burke, J.; Estrin, D.; Guibas, L.; Kansal, A.; Madden, S.; Reich, J. Mobiscopes for human spaces. IEEE Pervasive Comput. 2007, 6, 20–29.
13. Xu, Z.; Mei, L.; Choo, K.K.R.; Lv, Z.; Hu, C.; Luo, X.; Liu, Y. Mobile crowd sensing of human-like intelligence using social sensors: A survey. Neurocomputing 2018, 279, 3–10.
14. Wang, R.Q.; Mao, H.; Wang, Y.; Rae, C.; Shaw, W. Hyper-resolution monitoring of urban flooding with social media and crowdsourcing data. Comput. Geosci. 2018, 111, 139–147.
15. Kursuncu, U.; Gaur, M.; Lokala, U.; Thirunarayan, K.; Sheth, A.; Arpinar, I.B. Predictive Analysis on Twitter:Techniques and Applications. In Emerging Research Challenges and Opportunities in Computational Social Network Analysis and Mining; Springer: Cham, Switzerland, 2019; pp. 67–104.
16. Gaber, I. Twitter: A useful tool for studying elections? Convergence 2017, 23, 603–626.

# Team Contribution

1. FNU Sonam (ss15624) – Data collection (twitter API streaming), cleaning and processing, Training BERT model, building visualization dashboard in Kepler GL. Completed following sections of report - Data, Methodology for BERT, Results & References.
2. Max Magid (mmm9940) – Training Seq2Seq Model, Completed Background, Introduction, Methodology for Seq2Seq and confusion matrix.
3. Wonchan Lee (wcl311)– Training TF-IDF Model, Completed Abstract section and Methodology for TF-IDF and confusion matrix.
4. Hanfie Vandanu (hv480) – Training BoW and Word2Vec model, completed conclusion and future works section and confusion matrix.
5. Suraj Sunil (ss14449) – Training TransformerXL Model, Completed the formatting and peer reviewed all the sections and Methodology for TransformerXL model and confusion matrix.