# NATURAL LANGUAGE PROCESSING WITH DISASTER TWEETS

**Fadila Hamid Abdulai**: fhamid@aimsammi.org
**Bonaventure Fonteh**: kfonteh@aimsammi.org
**Idriss Nguepi Nguefack**: inguepi@aimsammi.org
**Sakayo Toadoum Sari**: tsakayo@aimsammi.org

## Introduction

Natural language processing (NLP) helps computers communicate with humans in their own language and scales other language-related tasks. For example, NLP makes it possible for computers to read text, hear speech, interpret it, measure sentiment and determine which parts are important.Today's machines can analyze more language-based data than humans, without fatigue and in a consistent, unbiased way. Considering the staggering amount of unstructured data that's generated every day, from medical records to social media like twitter, automation will be critical to fully analyze text and speech data efficiently [Ins]. Text classification is a basic problem in the field of natural language processing and different machine learning approaches have been studied successively throughout history. The rapid growth and impact of social network has produce enormous data that can be used as a valuable source of information.Twitter has become an important communication channel in times of emergency. The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programatically monitoring Twitter (i.e. disaster relief organizations and news agencies). But, it's not always clear whether a person's words are actually announcing a disaster.. In this work, we will to build machine learning models that predicts which Tweets are about real disasters and which one's aren't. Hence our objective is to;

- Perform Exploratory Data Analysis (EDA) of the dataset;
- Explore different machine learning algorithms and choose the appropriate one (the one which yields the best performance on unseen data based on the accuracy score).

## 1 Exploratory Data Analysis (EDA)

The main objective of this section is to explore the dataset we are provided to get key insights from the data. To achieve this, we made use of histograms and word clouds to have an idea of the balance between the classes (0 for non-disaster tweets and 1 for disaster tweets).

### 1.1 Classes

The dataset provided has two main classes. The non disaster tweets represented by a target value of 0 and the disaster tweets represented by a target value of 1. Figure 2 presents the number of data points we had in each class of our dataset. It can be seen that the dataset has 4342 non disaster data points and 3271 disaster data points. Hence, the dataset is unbalanced because we have 1071 more data points in the non disaster tweets class than in the disaster tweets class. This suggests that we should explore ways of handling unbalanced datasets for optimal model performance.

### 1.2 Most common words In classes

To present the most frequent words used in both classes, we made use of the world cloud, which is an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance.

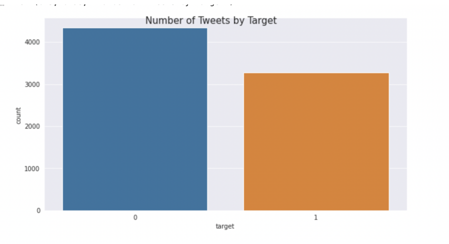

Figure 1: Dataset classes

### 1.2.1 Non Disaster Tweet

Figure 1 presents the word cloud of non-disaster tweets word cloud. It can be observed that the letter 't' occurs many times and same frequency like 'CO', followed by https. Again, the least frequent words are put, gt and book.

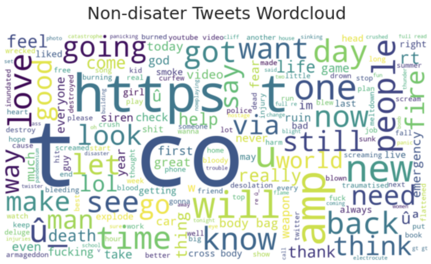

Figure 2: Non-Disaster Tweets Word cloud

### 1.2.2 Disaster Tweet

Same like above figure 3 shows the word cloud for disaster tweets. From this figure one can see that most of the disasters are fires, followed by floods and storms at the same level, while wildfires are the least occurring amongst the disasters.

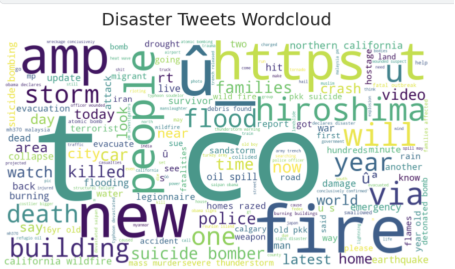

Figure 3: Disaster Tweets Word cloud

## 2 Methods

There are many machine learning techniques when it comes to classification tasks. These ML models operate on vectors, hence we need vectorized versions of dataset to work with because our dataset is in text format. In this challenge, we emplored the use of count vectorizer with Logistic Regression and Naive Bayes [Kha20], TF-IDF with logistic regression and Naive Bayes and, BERT [BOUdf],[Dev+18]. In the case of Logistic Regression and Naive Bayes, we used cross-validation for more accurate validation scores. We also decided to use k-fold cross-validation (with k = 5) on the labelled dataset and used accuracy as our evaluation criterion.

Count vectoriser is a basic vectoriser which takes every token (in this case a word) from our data and turn it into a feature. As a whole it converts a collection of text documents to a sparse matrix of token counts. TF-IDF (term frequency–inverse document frequency) vectorizes words by taking into account the frequency of a word in a given document and the frequency between documents.Mathematically, the importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus [Jon81].

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text[Dev+18]. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction in a sequential manner. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.

To measure our models' performance before submitting, we first performed training-validation splits on the training dataset with 0.2 as a validation percentage.

## 3 Results

We implemented Logistic Regression and Naive Bayes using a count vectorizer. The count vectorizer + Logistic regression gives a validation accuracy of 73% and Naive Bayes have an average 50%. When we use the two models with Tf-idf, Logistic regression yields an accuracy of 73% like the one using count vectorizer and Naive Bayes have an accuracy of 47%. These poor accuracies can be explained by the fact that both count vectorizer and tf-idf do not take into account the order of words and also words' meaning, contextual information, and similarity. Finally, we trained our BERT with (epochs=5,batch_size=32). This model outperforms the two other models with a validation accuracy of 83% because it produces embedding by using a word meaning, contextual information, or similarity of words and has an attention mechanism.

## 4 Conclusion

In this project, our main goal was to predict whether a given tweet announces a disaster or not using a dataset of tweets. To achieve this goal, we started off by performing exploratory data analysis (EDA) for a better understanding of the dataset and finally we explored different machine learning and deep learning algorithms including; logistic regression, naive bayes, and the BERT model. To apply the classical machine learning we implemented a count vectorizer which allowed us to achieve an accuracy of 73% for the logistic regression model and 50% for the naive bayes model.Using the TF-IDF embedding technique, we achieved same results for the logistics model with a decrease in the performance of the naive base from 50% to 47%. We can say that the encoder method had no effect on the logistic regression model. We pushed the research further by using the BERT model as an encoder then applied a linear layer to classify the tweets and this enabled us to achieve an accuracy of 83%. The reason for this improvement which are the limitations of the count vectorizer and TF-IDF are that the encoding includes words meaning, contextual information, or similarity of words and has an attention mechanism. Obtaining this results, we are happy to have taken this research to completion and we hope if we had more time, we would explore better options of fine-tuning in a bid to achieve an optimal performance. We could also explore classical models after using the BERT as an encoder.

## References

[Dev+18] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[Kha20] Harsh Khandewal. "Sentiment Analysis of a Tweet With Naive Bayes". In: *towardsdatascience* (2020).

[Ins] SAS Insights. "Natural Language Processing". In: $https://www.sas.com/en_us/insights/analytics/what - is - natural - language - processing - nlp.html$ ().

[BOUdf] XAVIER Cedric BOUSBIB Ruben. "Sentiment Analysis of a Tweet With Naive Bayes". In: *github.io* (https://rubenbsb.github.io/pdfs/nlp-project-mva.pdf).

[Jon81] Anna Bianca Jones. “Sentiment analysis on reviews: Feature Extraction and Logistic Regression”. In: *Medium* (https://medium.com/@annabiancajones/sentiment-analysis-on-reviews-feature-extraction-and-logistic-regression-43a29635cc81).