**Capstone Project**

Duong Nguyen  
July 26th, 2021

**I. Definition**

**Project Overview**

Twitter, which is one of the most impactful social networks, has provided the facilities for millions of users to share information and news. Tweets reflect personal opinions about different topics. The ability to extract and collect information from the tweets is of great benefits for commercial as well as personal purposes. For example, collecting the tweets related to a product can help the company for the marketing campaign. However, there is a massive amount of data posted daily in the form of short texts or audio. It is nearly impossible to do the analytics manually. Hence, there has been many research trying to do the semantic analysis with tweets automatically.

One of the recent works on the tweet topic is <https://www.hindawi.com/journals/complexity/2020/8892552/>. In the paper, the authors tried to classify and do sentiment analysis on tweets related to healthcare. The aim of the research is to recommend users with personalized tweets. Techniques in Natural language processing (NLP) are used in the paper. NLP is a branch of Artificial Intelligent, which exploits rules, statistic, and neural network techniques to process and analyse large amount of natural language data. NLP has been used to create many useful applications such as spam filter, online translators, text analytics, etc.

Inspired by that, this project also works on NLP to predict if a tweet mentions about a disaster or not. The dataset are random tweets. The datasets are from a competition in Kaggle. <https://www.kaggle.com/c/nlp-getting-started>

**Problem Statement**

The project is going to predict if a tweet mentions about a real disaster or not. The dataset contains of the tweets about disasters and random tweets. The result can be used to detect a surging number of tweets related to disaster and provide on time alerts to government or people in the affected areas. That could help to trigger preventive methods faster and reduce the destructive level. Various NLP techniques are used to extract the features. Classification algorithms such as XGBoosting and Neutral network are applied to build the model.

**Metrics**

F1 is used for evaluation. The formula for F1 score is:

𝐹1=

F1 score can give a balanced score between the recall and precision. In this problem, we want to have a good result for both recall and precision. Failing to provide a disaster notification can be as harmful as giving a wrong alert.

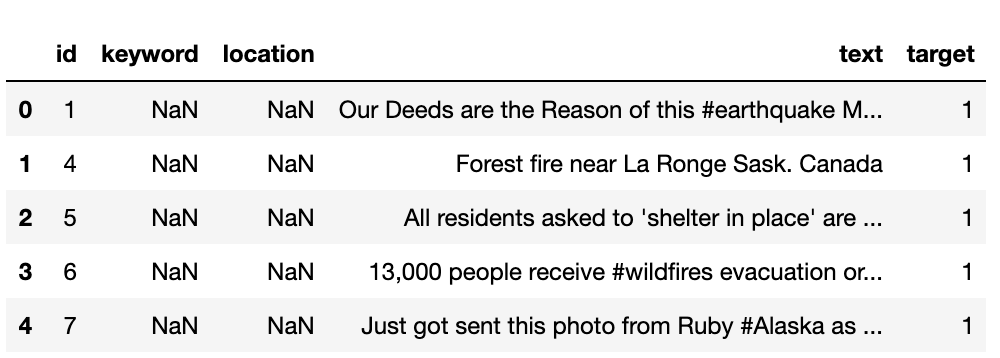
**II. Analysis**

**Data Exploration**

There are two datasets. One is the training set, and the other is the test set. The datasets are given as csv files. The data inside the csv files can be loaded into the dataframe. The training set contains of 7613 samples. The test set contains of 3263 samples. The training set is used to build the models. The test set will be used for prediction and submitted the result to Kaggle for evaluation

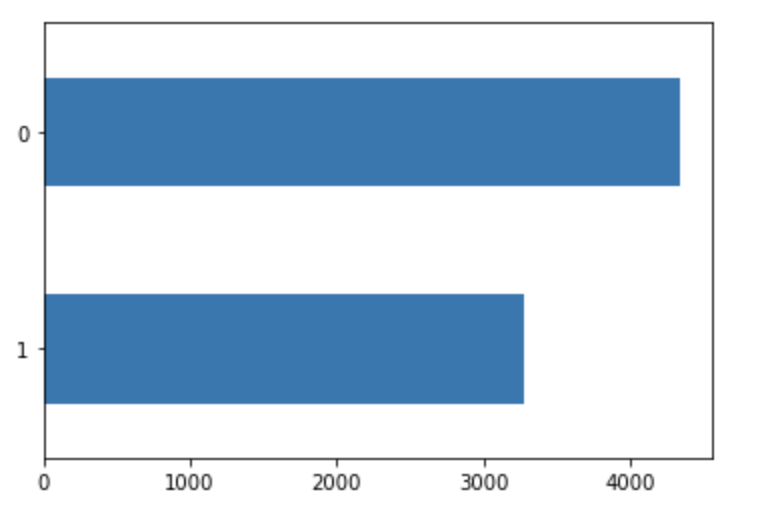
The attributes for each sample are:

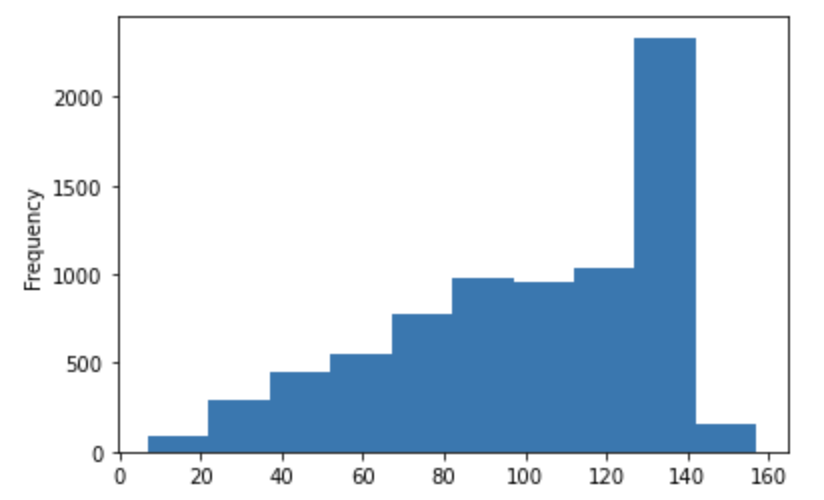
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **Keyword** | **Location** | **Text** | **Target** |
| Id of the row | keyword that determines if the text mentions about the disaster | Location of the tweets | The message of the tweets | 1 if this is a tweet about disaster, 0 if this is not. This attribute is only available in training set. |

****

**Exploratory Visualization**

The train dataset is quite balanced. The number of positive (target=1) and negative (target=0) samples do not differ significantly.



Also, the length of the text ranges from 10 to 160 characters.

**Algorithms and Techniques**

* This is the classification problem. The feature (X) is the text column, and the target variable (Y) is the target column. The feature of each sample (row) is cleaned and convert to an array of number using TfidfTransformer. CountVectorizer is not a good solution to extract the features because the most common words are I, and, the, etc. which cannot be used to predict the tweets.
* Two models are developed to solve this problem. One is the neutral network built with Pytorch, and the other is XGBoosting. Those are the popular models for classification problems.

**Benchmark**

A logistic regression model is picked as the benchmark model: <https://www.kaggle.com/hemanthkumarkarangi/logistic-model-disaster-tweet>.

That is because logistic regression is basic algorithm to train a model. The benchmark solution is able to use a simple algorithm to achieve a public score of 0.79, which is higher than some models built with complex algorithm such as LSTM or Neutral network. In this project, XGBoost and Neural network are used to train the models with the hope that a better result can be achieved with more complex algorithms.

**III. Methodology**

**Data Preprocessing**

The following steps are done to convert the text into features:

1. Data cleaning

* Use BeautifulSoup to extract text from HTML
* Convert text to lowercase
* Clean text: remove unmeaningful characters, such as: ‘ll, !, ?, etc. The code to clean the text is copied from <https://github.com/yoonkim/CNN_sentence/blob/master/process_data.py>
* Remove stop words
* Apply stemming on the words

1. Feature extraction

* Use TfidfVectorizer to convert the cleaned text into a vector of numbers. The maximum number of features is set to 1000.
* The keywords that indication the disaster is rare and do not appear frequently in all the text, so TfidfVectorizer is suitable to be used here to give more score for keywords

1. Create data files

* Applied the method discussed in the previous section, the column “text” in the train set is converted into an array of vector. Those are the X values. The X values and the column “target” (Y) in the train set are combined into one dataframe. The dataframe is saved into train.csv
* Applied the method discussed in the previous section, the column “text” in the provided test dataset is converted into an array of vector. Those are the X values. The X values and the column “target” (Y) in the train dataset are combined into one dataframe. The dataframe is saved into test.csv

1. Upload file to S3

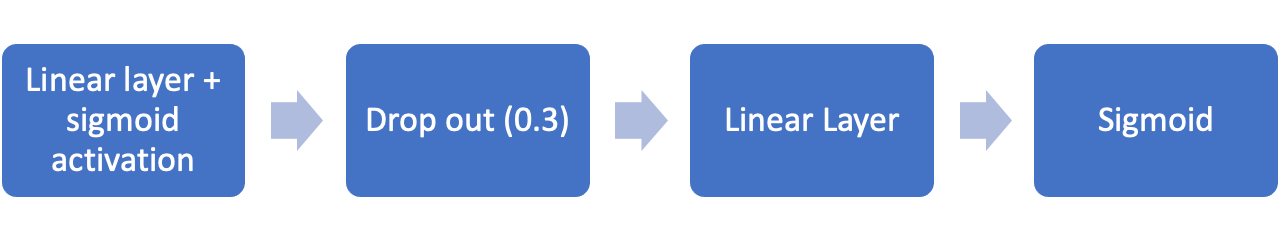
Train.csv and test.csv files created in the previous steps are uploaded to S3

**Implementation**

Two models are trained in this project

1. Neutral Network with Pytorch
2. Model

The architecture for the model is as below:



There are two linear layers in the model. The Activation function for each layer is sigmoid

Model is trained on the train dataset uploaded to S3 in the previous section.

1. Evaluation

The model is deployed and evaluated with the test dataset. The F1 score is: 0.7667

1. XGBoost
2. Model

The model is trained with the provided xgboost image in sagemaker

1. Evaluation

Batch prediction is applied with the evaluation set. The F1 score is: 0.7742

**Refinement**

XGBoosting model is selected for refinement. The following methods are applied to improve to model

1. Cleaning function enhancing

To improve the quality of trained data, nltk tokenize library and stopwords is used to remove stop words. Stop words are common words such as “the”, “a”, “an”

1. Hyperparameter tuning

Hyperparameter tuning is used for the xgboost model. The range of the parameter is:

* 'max\_depth': IntegerParameter(3, 12)
* 'eta' : ContinuousParameter(0.05, 0.5)
* 'min\_child\_weight': IntegerParameter(2, 8)
* 'subsample': ContinuousParameter(0.5, 0.9)
* 'gamma': ContinuousParameter(0, 10),

The provided train dataset is splitted into train set and evaluation set. 80% of the data is for train set and 20% of the data is for evaluation set. The HyperparameterTuner job is fitted with the train set and evaluation set

After the HyperparameterTuner job is completed, the model with the best parameter set is trained with the original dataset to make use of the large data training set. After that, the model is tested on the test set

**IV. Results**

*(approx. 2-3 pages)*

**Model Evaluation and Validation**

The predicted results on test dataset is submitted to Kaggle. Below is the result:

|  |  |
| --- | --- |
| Neural Network with 2 linear layers | 0.767 |
| XGBoosting | 0.775 |
| XGBoosting with enhanced cleaning function | 0.784 |
| XGBoosting with hyperparameter tuning | 0.782 |

The result is not significantly different between different models.

The xgboost model performing better than the neural network might be because our the neural network is too simple. There should be more layers in the neural network in order for the neural network to perform better.

Removing the stop words with the nltk tokenize library and stopwords library helps to improve the result. That can be because removing the stop words help to reduce noises and reducing the number of features.

Applying HyperParameter Tuning does not improve the result. That may be because the grid used to search for the best hyperparameter is small. Another reason can be because the hyperparameter is tuned on the subset of the train data, which is not enough to generalize for the whole dataset.

The final model used in this project is xgboost.

The final model is tested with many manual inputs. A few example is:

* The weather is nice -> predict as negative
* The storm is coming -> predict as positive
* There is no storm -> predict as negative

The model seems to predict correctly in general, but it is not able to hand special case like: negation.

**Justification**

The final model does not outperform the final benchmark solution, but the different is small (78.4 vs 79).

**V. Conclusion**

**Free-Form Visualization**

The wordCloud for the cleaned data is plotted



Figure 1: wordcloud for cleaned data - basic



Figure 2: word cloud for cleaned data – removing stop words

From the wordcloud, it can be seen that by removing stop words, more words relating to the disaster are highlighted, such as: fire, flood, death. That means the model can be able to capture the keywords better.

**Reflection**

This project tried to build a classification model to predict if a tweet is about a disaster or not.

Basic steps to solve a machine learning problem are implemented:

1. Data analysis

A few charts are plotted to analyse the nature of the input data. The dataset given is not imbalanced

1. Data Cleaning

BeautifulSoup and regex are used to remove unmeaningful characters

1. Feature extraction

TFID algorithm is used to convert texts into vectors

1. Model training

Neural network and xgboosting algorithm are used to train the model. XGBoosting performs better than the neural network.

1. Refinement

More data cleaning is applied. Also, hyperparameters are tuned to find the best model.

The final model is trained with xgboosting. It is interesting that xgboost cannot perform better than the benchmark model which was trained with basic Linear Regression. However, the final model performs close to the bench mark model (0.784 vs 0.79) and it is able to predict correctly some random texts found online. In general, the model should add value when it is used to solve the proposed problem.

A lot of time in this project was used to debug the code, especially with the pytorch estimator. An example error happened was: when two dataframe were concated, the result dataframe created more rows than expected due to memory limited. Reset\_index function was used to solve the issue, but then, during the training process, some weird exception was thrown.

**Improvement**

A few improvements can be done in the future for this project

* Clean data: the data should be cleaned more. Some of the words that are not related to disaster like: pronoun can be removed. This will help to reduce the number of features
* Build more complex model for neural networks. Currently, a simple neural network with two layers is used. More complicated models like LSTM can be tested.
* There is a recently popular model called BERT. That model can be tried to solve the problem.