Fall 2020

Real or Not? NLP with Disaster Tweets

EE285 Project 2

Rojin Zandi

Partner: Shifa Shaikh

1. Dataset Description

Twitter, with 330 million users from all over the world, is one of the most popular social media. Our dataset is a collection of tweets that contain words about disasters. Our goal is to distinguish whether there has been a disaster, or the tweet is metaphorically mentioning words related to a disaster, so this is a binary classification problem. Considering the limitation of 280 characters in each tweet, the dataset does not need high computation power or large memory. The dataset contains the Training set and Test set with 7613 and 3263 data points, respectively, and each data point contains some information about the location, ID, keyword, and text of the tweet.

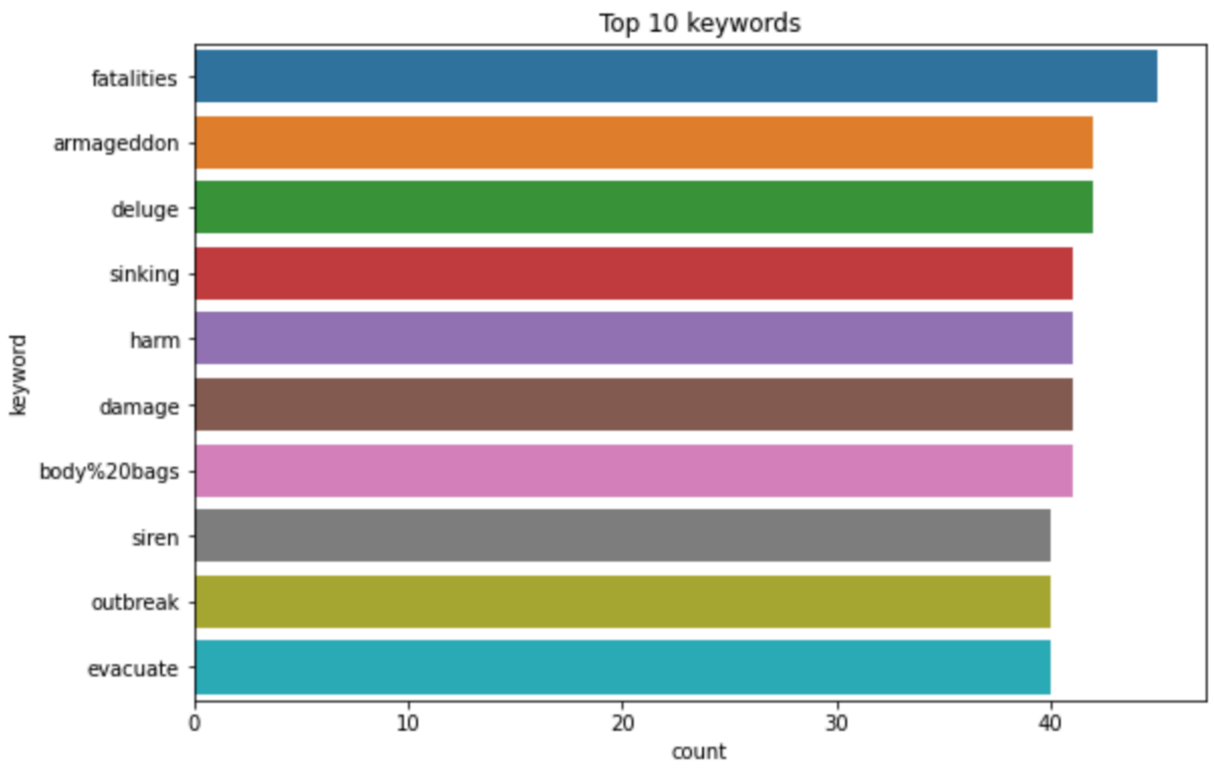


Fig. 1: Top 10 keywords in training set.

As it can be seen in Fig.1, ***Fatalities*** with more than 40 times repetition is most common keyword in the training set tweets. In Fig.2, we have studied top ten keywords in disaster and non-disaster tweets, separately. The word ***Outbreak*** is the most repeated keyword in the disaster tweets, while it ranked 9th in the whole training set.

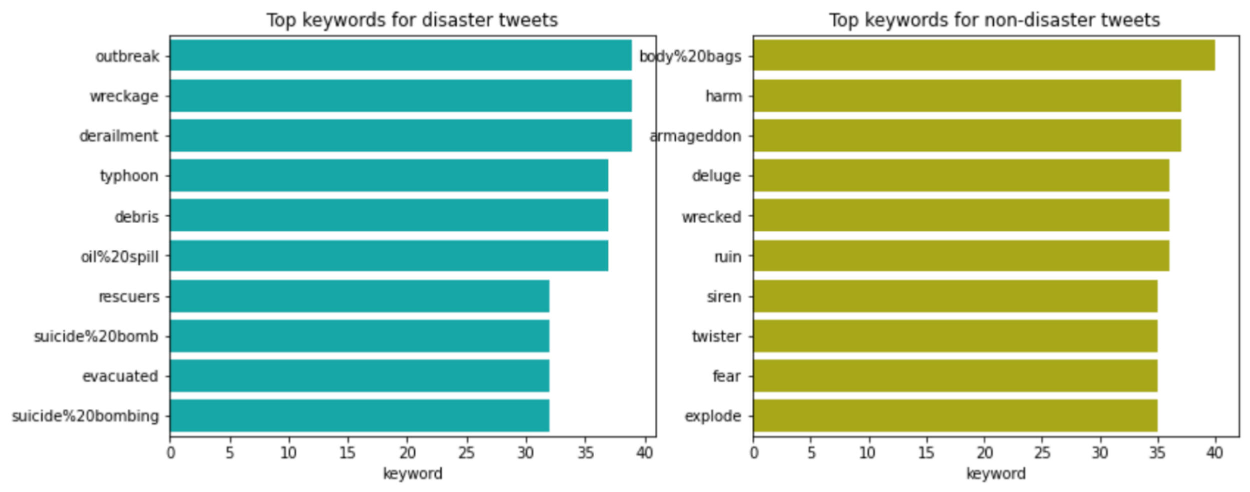


Fig. 2: Top 10 keywords in disaster (left) and non-disaster (right) tweets.

In Fig.3 we have investigated the most and least probable keywords for the disaster tweets. For example, if a tweet includes the word ***Derailment***, it has a high probability to be a disaster tweet, and vice versa, the word ***Aftershock*** has the least probability.

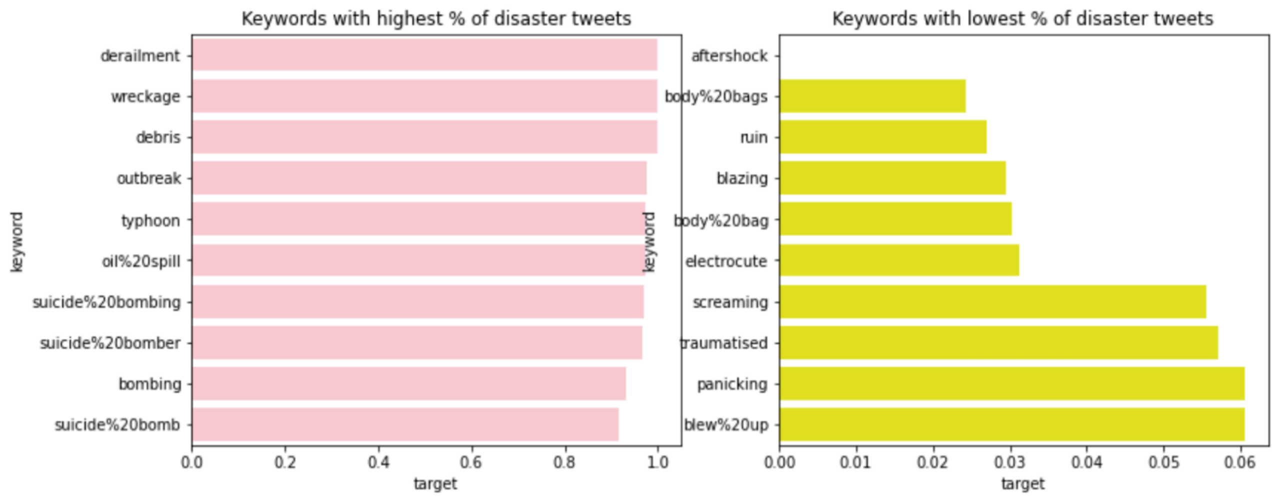


Fig. 3: The most and least probable keywords in disaster tweets.

Another information that we can visualize is the location of the tweets, and in Fig.4 the top 15 common locations are shown. But the *Tag Location* in twitter suggests different tags for a same location. As it can be seen in Fig.4, ***USA*** and ***United States*** are considered as two different locations, so this is not a proper comparison. So, we fixed this problem in Fig.5.

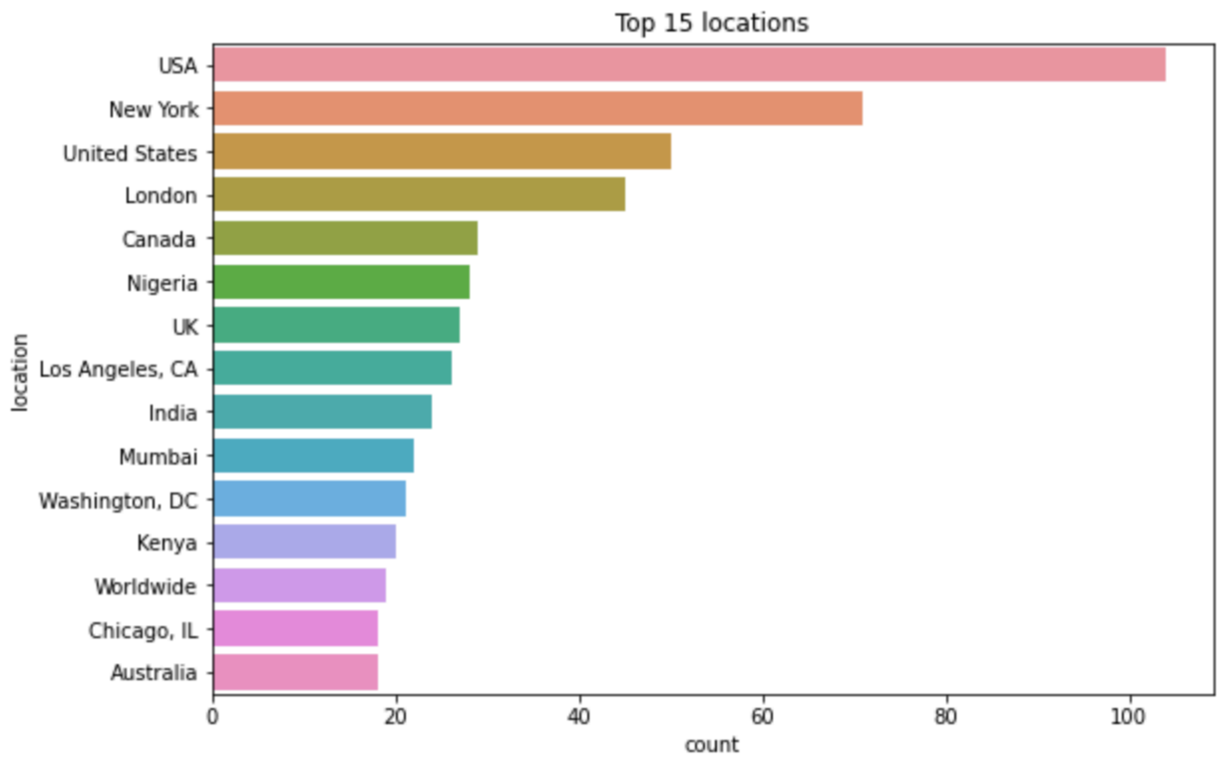


Fig. 4: The top 15 location of the training set tweets.

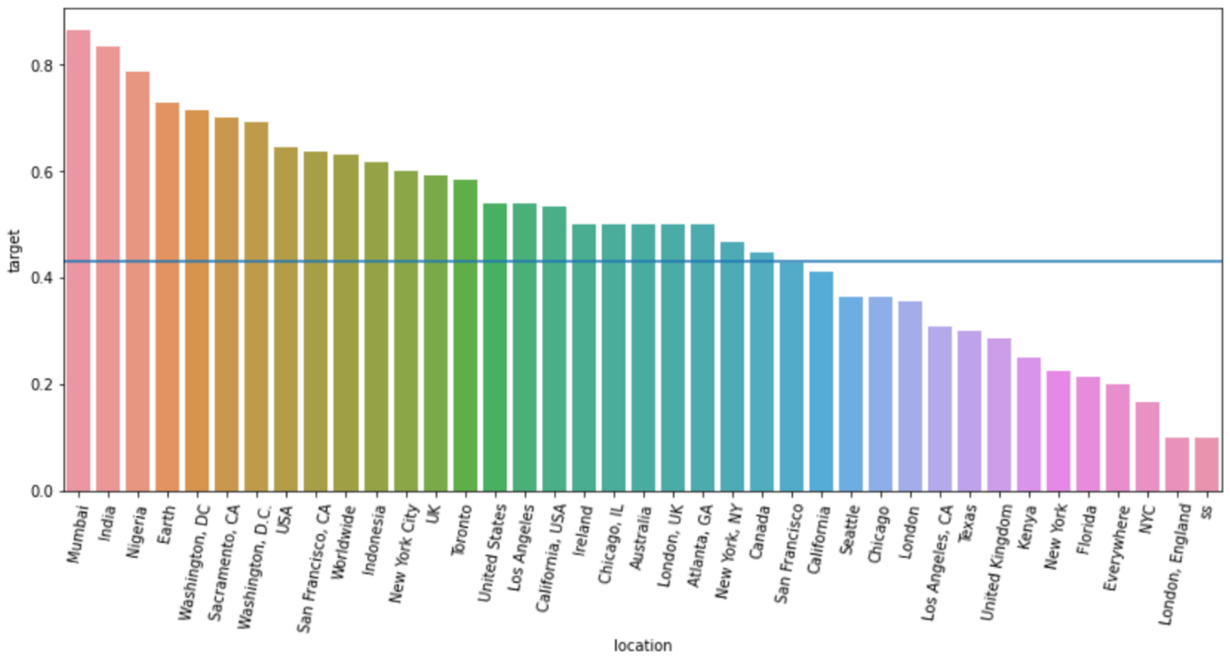


Fig. 5: The most common locations of the training set tweets.

2. Preprocessing

Preprocessing in machine leaning is transforming and cleaning the raw data to a proper form in order to feed ML algorithms. It includes data cleaning, transformation, reduction, and integration which depends on our data and application. In this project we are dealing with text data, that needs to be transformed to vectors of numbers.
As mentioned before, each tweet can contain 280 characters that these characters include words, punctuations, emojis, URLs, HTML tags, symbols and some other type of data. Considering our goal, we need to remove most of them in order to improve the accuracy of our model. In other words, we only need to keep the words. In the preprocessing part we have removed and cleaned our dataset from the unnecessary data types and also, we have made a misspell dictionary to replace the misspell written words. To preprocess the text datasets there are many Natural Language Processing libraries that help us such as NLTK (Natural Language Toolkit) and we have used it to remove the stop-words. [1]
Next step in preprocessing is breaking the texts to words or characters also known as Tokens. Each token is represented by a number, for example the word ***Rain*** can be represented by 0001. After splitting the text, we have to create a sequence of tokens that represent each tweet. Notice that different tweets can have different number of tokens and it affects our sequence. To solve this issue, we have use post padding, that puts zeros after the sentence. Also, you can control the length of sequence by defining a maximum length. If we have a tweet which is longer than the defined maximum length, we can do the truncating, that we have applied post truncating. Finally, we have applied the mentioned preprocessing methods on the training and test set.

3. Models Utilized

In this project, we have implemented four Recurrent Neural Networks. As can be seen in Table 1, in each model we have changed at least one parameter which will affect the efficiency and may decrease or increase model accuracy and loss. In the provided codes, in order to create our models layer by layer, we have used Sequential API. As I can be seen in the codes, we have done Word Embedding which is an unsupervised method of representing words that have similar meaning or representation. Before applying word embedding, for every token we had a large dimension sparse vectors and it was difficult to find any relation between the words with same meaning. [2] By applying word embedding, we will have an analogy between the words that can improve the accuracy of our model to classify the disaster and non-disaster tweets.

Baseline Model:

Our baseline model, was implemented with six layers, including: Word Embedding, Dropout, 1-D Convolution, Max Pooling, LSTM (Long Short-Term Memory Network), and Dense layer. The activations functions that we used are ReLU (Rectified Linear Unit) and Sigmoid, for the convolution layer and output layer respectively. The optimization algorithm applied in all the models is Adam algorithm with 0.0001 as learning rate.

	# epochs	Regularization technique	Batch Size	Training Accuracy	Val. Accuracy
BM	4	Dropout	64	57.11%	57.64%
1	15	Dropout	15	99.33%	75.77%
2.1	15	None	12	86.12%	81.81%
2.2	Changes	Early Stopping	12	89.95	81.22%

Table 1: Parameters and accuracy of utilized models

4. Model Improvements

After implementing the baseline model and trying to improve it, we understood that we have a problem in our preprocessing part. So, we changed the preprocessing method and our model.

- **Model 1**

  In model 1, we have an RNN model with 1978101 parameters in four layers, including: Word Embedding, Bidirectional, and Dense layer. In comparison with baseline model, we have reduced number of layers and also added Bidirectional layer which is generative method and improves the accuracy of our model. This layer allows our model to have access to past and future state simultaneously. In other words, we can run the network backward and forward in each time step. As you can see, the training and the validation accuracy have increased by 42.22% and 18.13%, respectively, but unfortunately, we are facing the overfitting problem which can be seen in Fig.7.

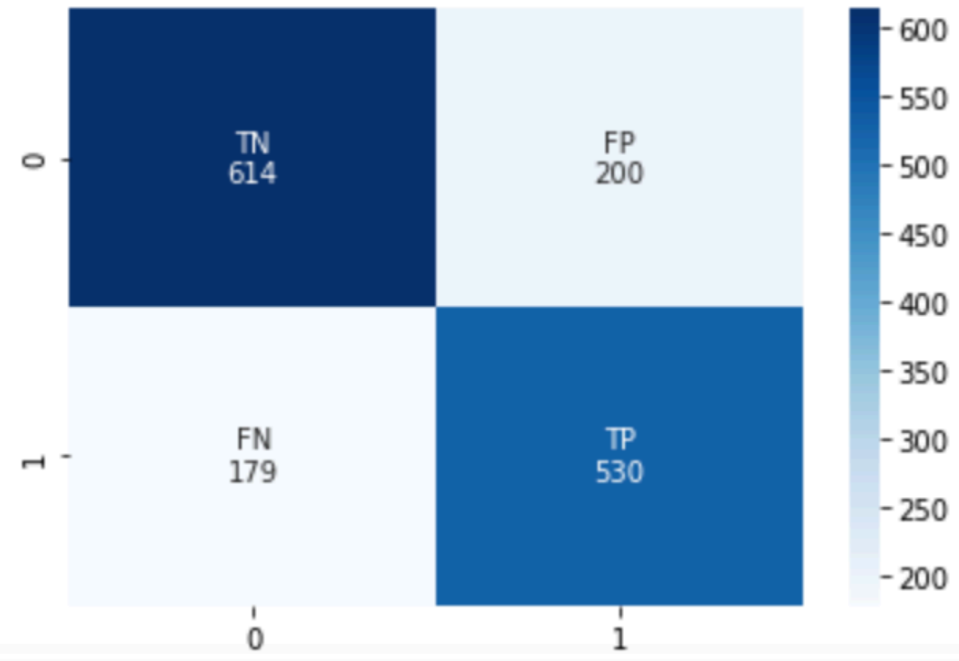


Fig. 6: Confusion matrix of validation set in model 1

	Precision	Recall	Accuracy	F1-Score
Validation set	77.42%	85.42%	75.11%	76.41%

Table 2: Metrics of model 1

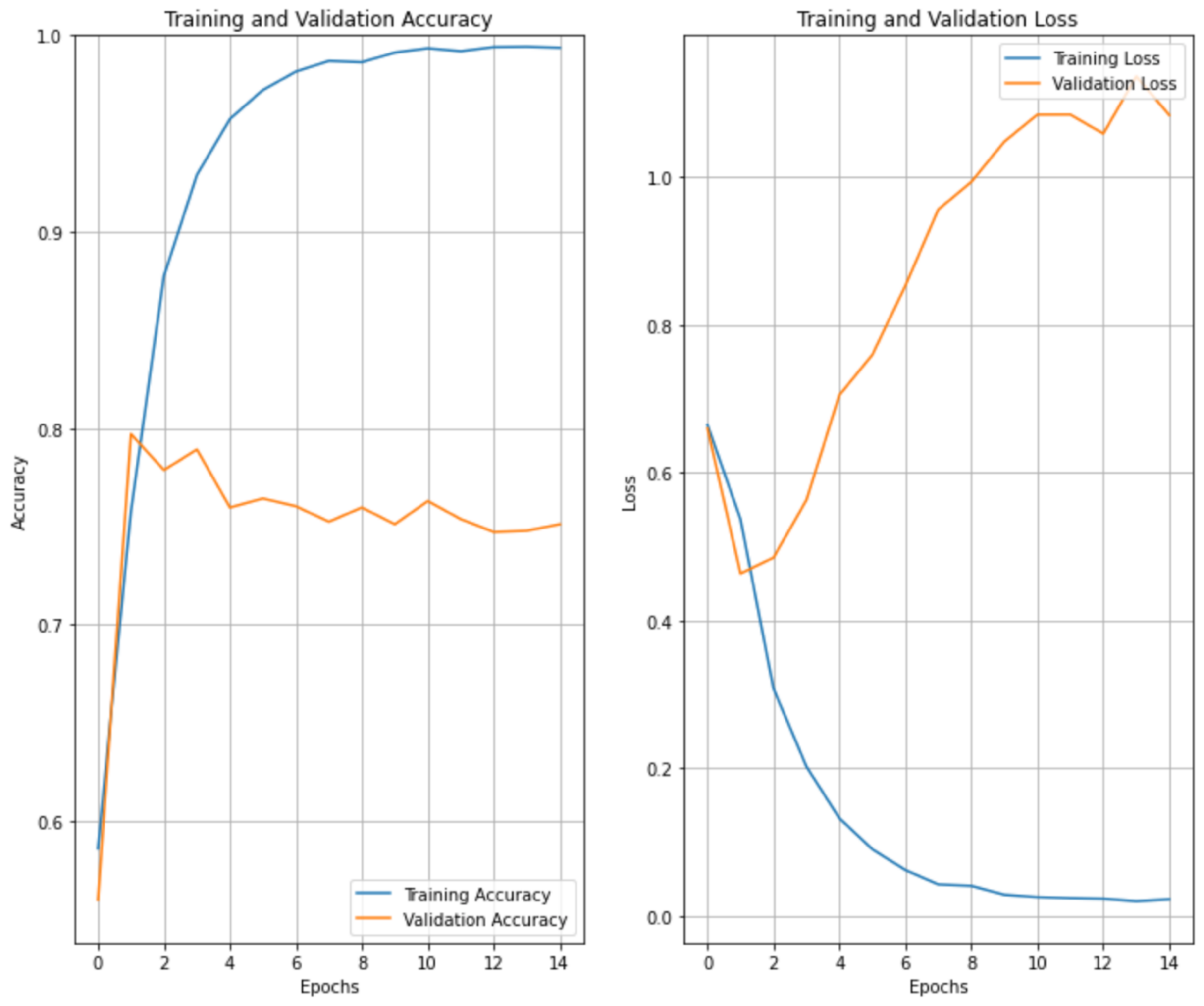


Fig. 7: The accuracy and loss of the training and validation set in model 1.

- **Model 2.1**

As shown in Fig. 7, there is a high overfitting in our model. To tackle this issue, we have applied GloVe (Global Vector for Word Representation), which is an unsupervised algorithm that computes the co-occurrence of two different tokens, and it helps to find a relation between them. It produces a matrix of word-word co-occurrence matrix and uses the non-zero elements of this vector for training. GloVe helps the model in time and computation, and also outperforms comparing to the last model. [3] [4] The advantage of applying GloVe is benefits from local and also global statistics.

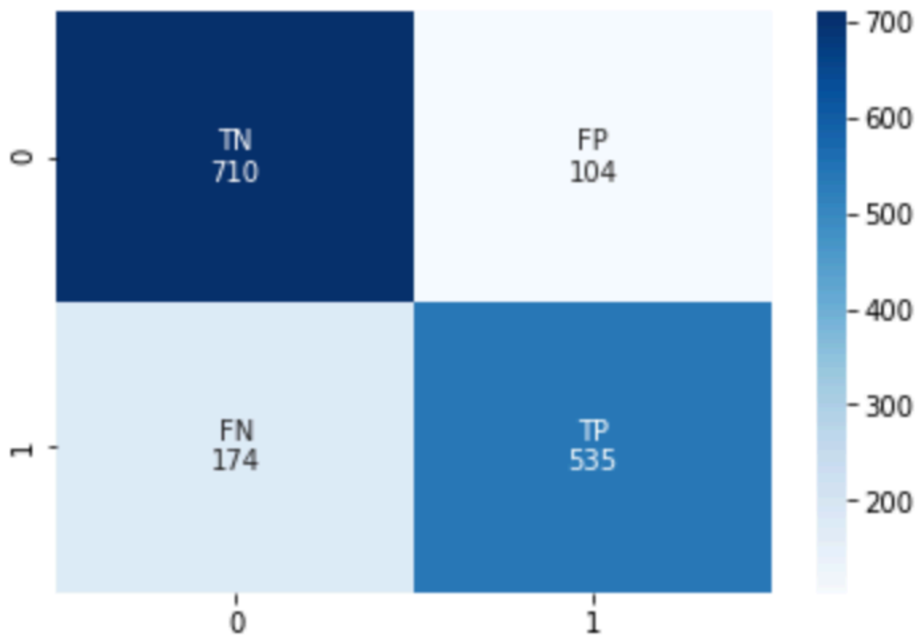


Fig. 8: Confusion matrix of validation set in model 2.1

	Precision	Recall	Accuracy	F1-Score
Validation set	77.42%	85.42%	75.11%	76.41%

Table 3: Metrics of model 2.1

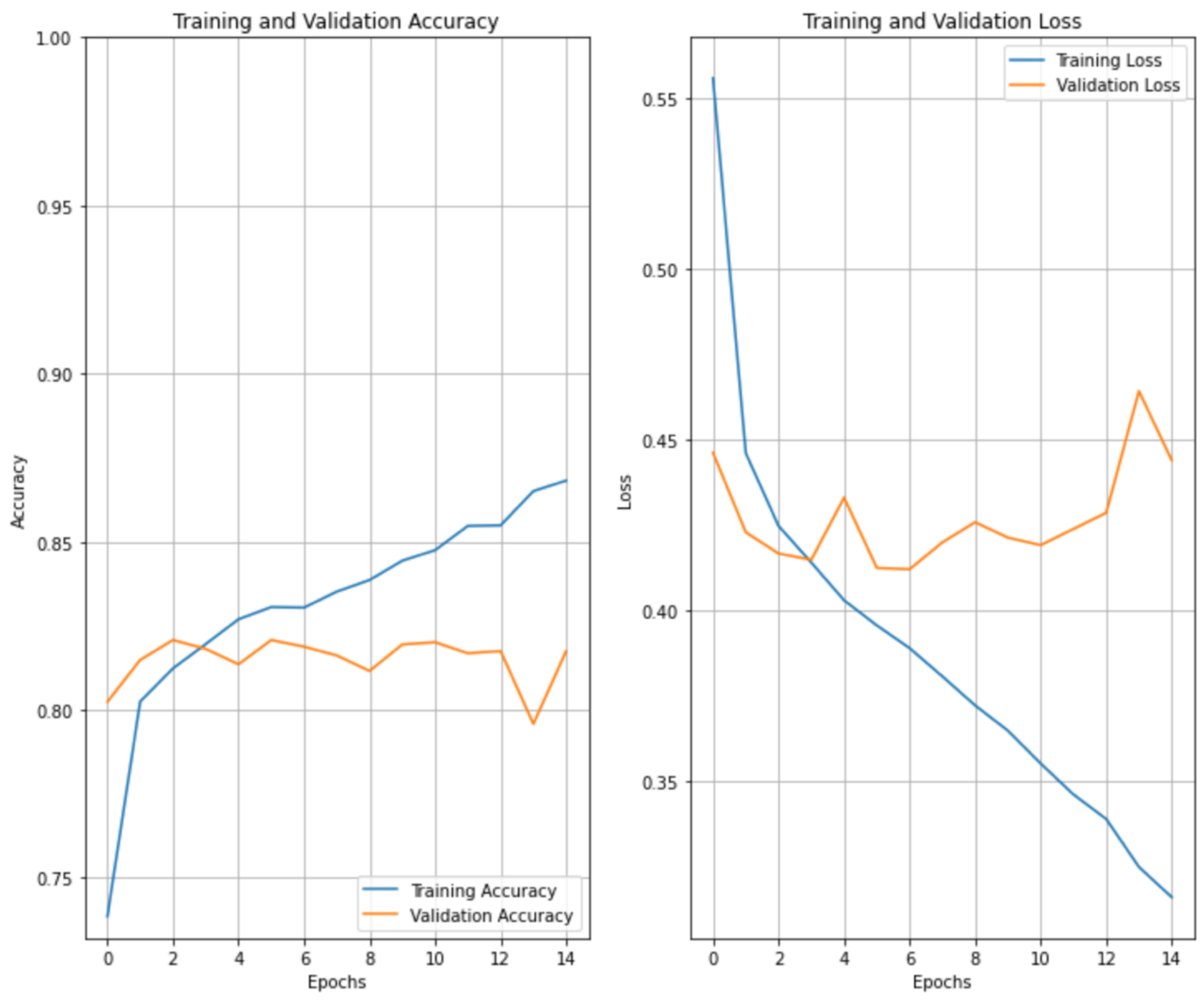


Fig. 9: The accuracy and loss of the training and validation set in model 2.1.

- **Model 2.2**

In model 2.2, we have tried to apply early stopping method to prevent the overfitting of our previous model. But unfortunately, it does not improve the model in this case and even increases the overfitting (Fig.11).

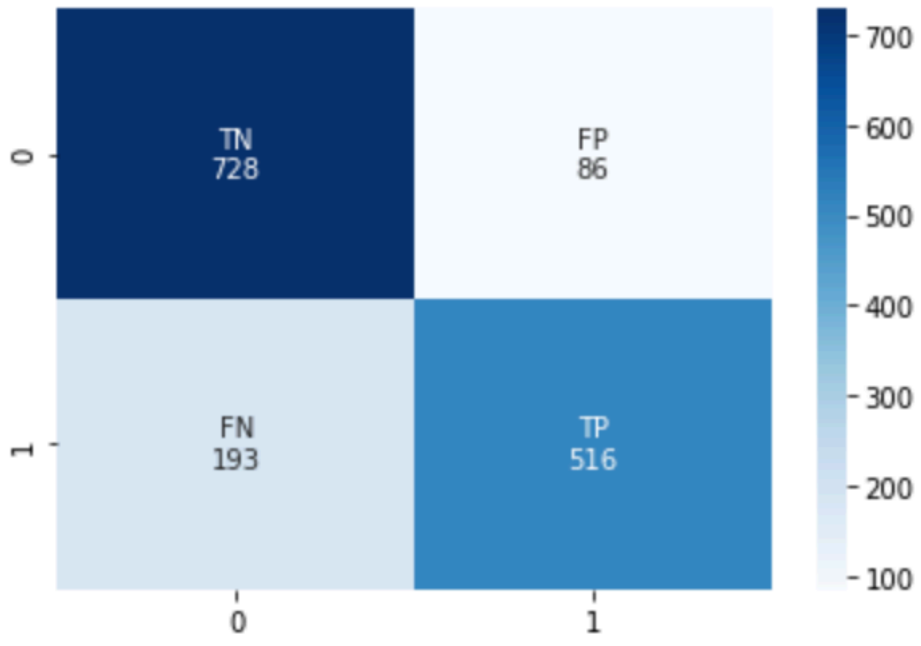


Fig. 10: Confusion matrix of validation set in model 2.2

	Precision	Recall	Accuracy	F1-Score
Validation set	77.42%	%	81.68%	83.91%

Table 4: Metrics of model 2.1

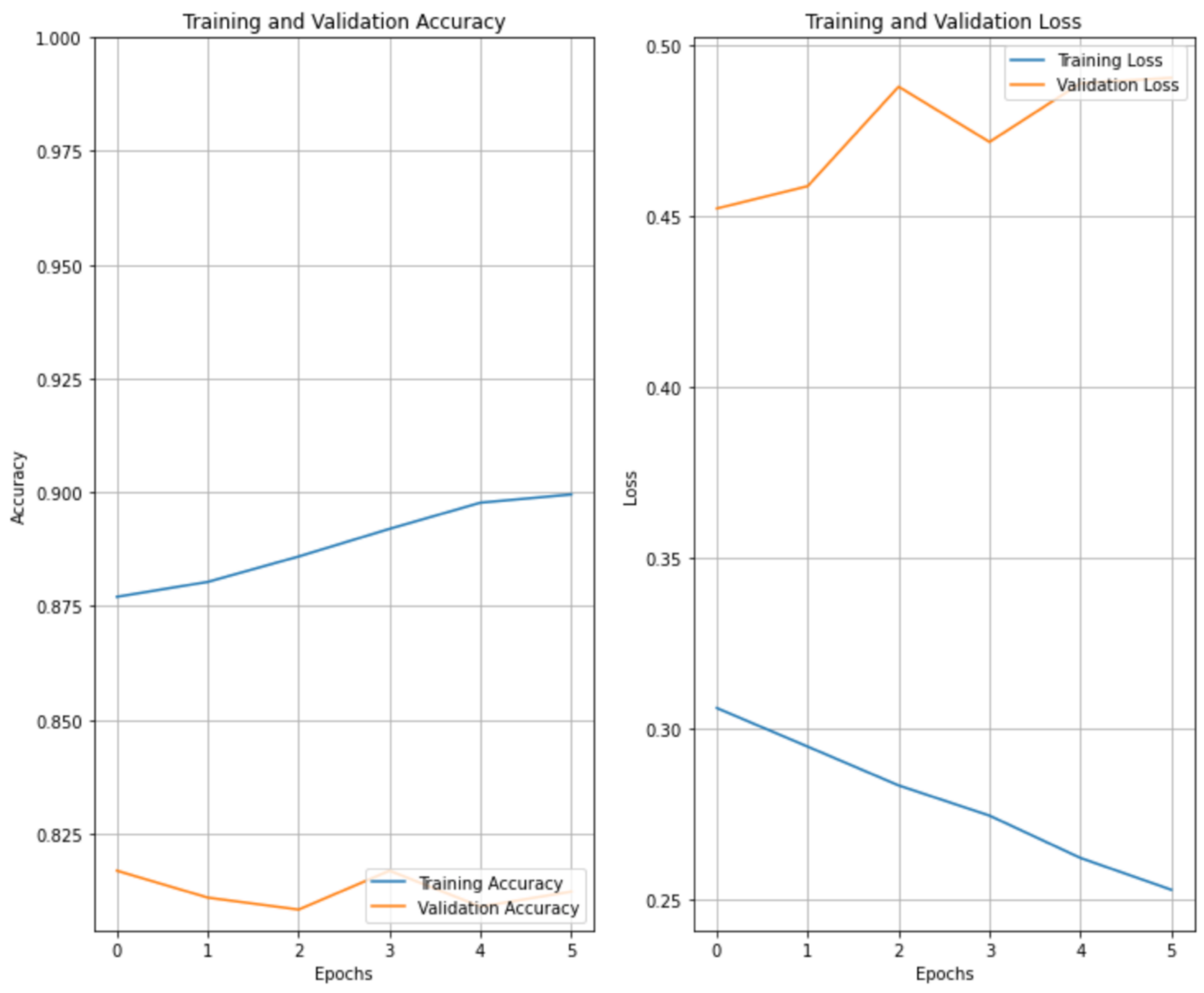


Fig. 11: The accuracy and loss of the training and validation set in model 2.1

5. Conclusions

Finally, after comparing different implemented models in section 4 we have observed:

1. In Natural Language Processing problems, the Preprocessing is one of the most important parts of solution.
2. In feature extraction, Word Embedding helps the model to achieve higher accuracies.
3. Generative methods, such as BI-LSTM, by allowing the model to have access to past and future state simultaneously, improve the model efficiency.
4. GloVe is an unsupervised algorithm that finds words with similar meaning. This method outperforms other embedding techniques and saves time and computational power.

5. In this problem and our model, early stopping regularization method is not suggested and does not improve the model. Notice that it changes in different models and problems.

We suggest applying TF-IDF preprocessing method and also fine tuning the model with BERT (Bidirectional Encoder Representations from Transformers) on this dataset. Also, the regularization methods can be improved, and they can help to obtain a better model.

References

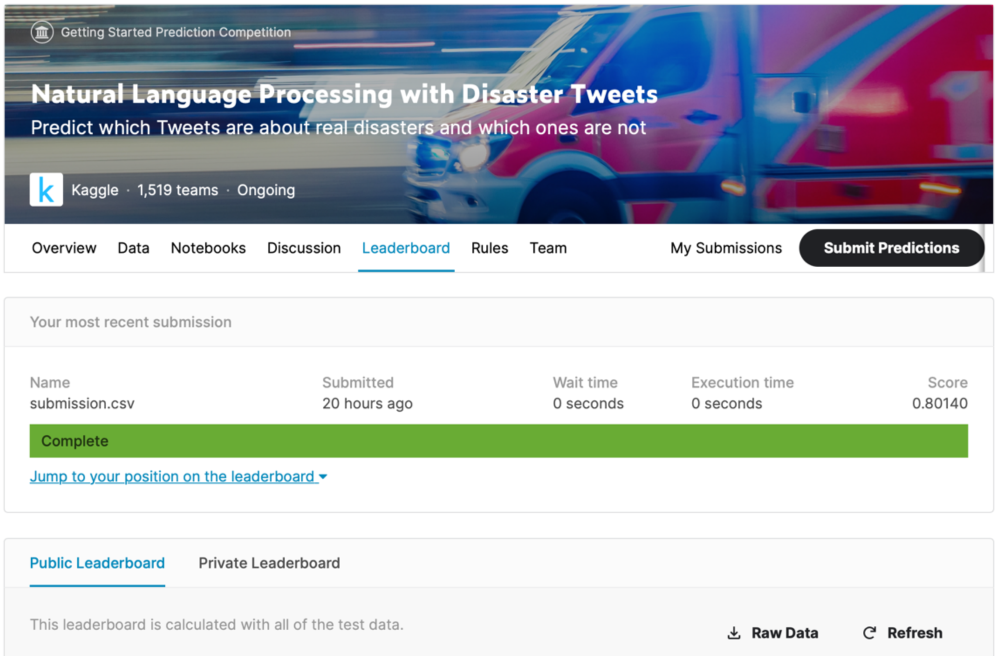
[1] [Online]. Available: https://www.kaggle.com/theoviel/improve-your-score-with-some-text-preprocessing .

[2] Mahnoosh Kholghi, Lance De Vine, Laurianne Sitbon, Guido Zuccon, "The Benefits of Word Embeddings Features for Active Learning in Clinical Information Extraction," in *Proceedings of Australasian Language Technology Association Workshop*, 2016.

[3] Jeffrey Pennington, Richard Socher, Christopher D. Manning, "GloVe: Global Vectors for Word Representation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014.

[4] [Online]. Available: https://www.kaggle.com/danielwillgeorge/glove6b100dtxt.

KAGGLE SUBMISSION


Getting Started Prediction Competition
Natural Language Processing with Disaster Tweets
Predict which Tweets are about real disasters and which ones are not
Kaggle · 1,519 teams · Ongoing
Overview
Data
Notebooks
Discussion
Leaderboard
Rules
Team
My Submissions
Submit Predictions
Jump to your position on the leaderboard
Public Leaderboard
Private Leaderboard
Raw Data
Refresh

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission.csv	20 hours ago	0 seconds	0 seconds	0.80140

Complete

This leaderboard is calculated with all of the test data.