Natural Language Processing with Disaster Tweets

Thanh Son Nghiem & Minh Kien Nguyen

Abstract

As monitoring Twitter for quick real-time emergency responses becomes increasingly important, so does Twitter analytics. One of the major tasks in analyzing Tweets is predicting which Tweets are about real disasters and which ones are not, which is the main goal of our project. This document provides a step-by-step summary of the process leading to that objective, from exploratoty data analysis to data preprocessing to applying models and finally evaluation and conclusion.

1 Introduction

Social networks have evolved beyond their original purpose as a communication platform. They are now often the first sources of information in times of emergency, as their contents are generated by users who underlie no editorial standards or restrictions and may even be observing a disaster in real-time. Twitter shortens the delay between the occurence of an emergency event and its first announcement, which can make a huge contribution to saving lives.

The purpose of this project is to build a Natural Language Processing model which can classify Tweets into two categories: Disaster (1) and Non-Disaster Tweets (0). For that purpose, we used a dataset of roughly 11,000 Tweets, which was generated by the company "figure-eight" and published by Kaggle. The dataset was split into a training set that were already hand classified and a test set that were not. Because of this, we had to personally label the test set.

Each Tweet sample in the dataset contains the following information:

- a unique *ID* for identification
- the *text* of the Tweet
- a *keyword* (optional)
- the *location* the Tweet was sent from (optional)
- the *target* (label) of the Tweet (training set only)

2 Data Analysis

The purpose of performing data analysis on the training dataset is to check its consistency and to identify features of interest which are likely to be potential discriminators for our classification task.

While examining its consistency, it was discovered that there were eighteen Tweets which appear more than once in the training set but were labeled differently in their duplicates. This is understandable, since the training set was classified manually, but not ignorable, as they can affect the training process negatively. These mislabeled Tweets must be relabeled before any training can be done.

The next step is to identify features of interest. The following features have been examined:

- *URL count per Tweet*
- *Punctuation count per Tweet*
- *Hashtag count per Tweet*
- *Mention count per Tweet*
- *Most common hashtags*
- *Most common keywords*
- *Most common locations*
- *Most common named entity types*

A feature is considered a potential discriminator if there is a recognizable pattern in it which can be used to distinguish Disaster Tweets from Non-Disaster ones. Of all those listed above, only the *URL count per Tweet* represents a potential discriminator. Disaster Tweets are likely to contain at least one URL, while Non-Disaster Tweets tend to have none. In the remaining features, no recognizable pattern can be found to tell the difference between the two label classes.

3 Data Preprocessing

3.1 Data Transformation

First, the Tweets in the training set were cleaned by removing URLs, HTML tags, emojis and punctuation marks. Then each Tweet was tokenized using the tokenizer provided by the nltk-package. Subsequently, we transformed all tokens to lower case format and removed all stopwords, which are words that appear frequently but carry little meanings. Part-of-Speech-tagging was then applied to the cleaned tokens before they were finally lemmatized by the WordNet lemmatizer.

3.2 Post-Transformation Analysis

The cleaning and transformation of the training dataset allowed us to examine other features of interest, whose examination could not be done due to the existence of URLs, emojis, stopwords, etc before the transformation step. These are:

- *Character count per Tweet*
- *Word count per Tweet*
- *Unique count per Tweet*
- *Average word length per Tweet*
- *Most common N-grams*

Of those features listed above, *character count per Tweet* can be considered a potential discriminator. Disaster Tweets with between 110 and 115 characters are most common, while Non-Disaster Tweets can have anywhere from as few as 30 to as many as 135 characters. Further analysis shows that Non-Disaster Tweets tend to have more characters than Disaster ones.

Regarding the *most common N-grams* in Disaster Tweets, it is notable that the greater the number N, the more disaster-relevant but less frequent those N-grams become. Now we assumed (based on observations) that starting from N = 4, all N-grams appearing more than once in Disaster Tweets are strictly disaster-relevant and all N-grams appearing at least once in Non-Disaster Tweets are strictly disaster-irrelevant. Because these N-grams appear quite rarely, if we were to classify each Tweet based on whether it contains any of these N-grams, the result would contain a large percentage of false negatives. To solve this, we added new potential discriminators called "disaster N-gram count" and "Non-Disaster N-gram count", which approximate the number of disaster-relevant and –irrelevant N-grams in each Tweet, respectively. We defined disaster-relevant N-grams as those in Disaster Tweets that are also parts (substrings) of any N+2-gram with more than one occurence in Disaster Tweets. An analogous definition was also made for the disaster-irrelevant N-grams. This definition only works with our assumptions when the N value in "disaster N-gram count" is greater than or, as in our implementation, equal to 2.

4 Training

4.1 Stratified Cross Validation

To determine the best model (a combination of embedder and classifier), we used stratified cross validation to split the preprocessed training dataset into four folds, each of which takes turn being used as a validation set to evaluate models that have been trained on the other ones. The ratio of Disaster and Non-Disaster Tweets in the training set also remains the same across all folds.

4.2 Shortlisting Promising Models

Chosen for evaluation were the following classifiers:

- *Logistic Regression*
- *Multiplayer Perceptron*
- *Random Forest*

and embedders:

- *Bag-of-Words*
- *GloVe*[1]
- *Sentence-BERT*[2]
- *TF-IDF*
- *Word2Vec*[3]

Among the embedders, *Bag-of-words* is the simplest representation of sentences in the form of vectors. Each Tweet text is represented as a bag of its words, weighting their importance based only on their frequencies. Because of this, *Bag-of-words* is generally less effective than *TF-IDF*, which not only measures the frequencies of words in a Tweet, but also the informativeness of words by penalizing those which are present in too many Tweets). Both embedders produce sparse vector representations of sentences.

On the other hand, the other three embedders take advantage of publicly accessible pre-trained models. Both *GloVe* and *Word2Vec* are

[1] Pennington/Socher/Manning (2014).
[2] Devlin/Chang/Lee/Toutanova (2019).
[3] Mikolov/Chen/Corrado/Dean (2013).

unsupervised learning algorithms that deliver exactly one vector representation for any single word. They are context independent and disregard word order during their training. *Word2Vec* embeddings predict whether words would appear in similar contexts, while *GloVe* embeddings focus on words co-occurrences over the whole corpus and calculate the probability that two words appear together.[4]

Instead of vectorizing each word in a Tweet separately, *Sentence-BERT* embeds a Tweet directly using a pretraied Sentence Transformer model. It takes word order into account and generates different word embeddings of a word for each different context it appears in. The relative distances between the resulting sentence vectors inside the vector space reflect the similarities (or differences) in their meanings.

4.3 Model Evaluation

Each listed embedder was then combined with each listed classifier, resulting in a total of 15 models. The metric chosen for the evaluation of these models was the F-Beta-Score, which weighs recall more than precision by a factor of Beta. Because the predictions of our final model will be used for classifying Tweets for disaster phase, precision is much more important than recall. That is why we set Beta = 0.5.

With an F0.5-Score of 97.49%, the best performance was delivered by the combination of *Sentence-BERT* and *Logistic Regression*. This model was thus selected to be the result model of our project. Finally, before the final testing, the *Logistic Regression* classifier was trained on the whole training set.

5 Testing & Conclusion

Applying the model on the test set delivers an F0.5 -Score of 65.43%, which was lower than our expectation. The reason for this may be our own manual classification of the test set. Since each of the two team members hand classified one half of the test set, our different views on the disaster-relevance of a Tweet most likely have led to data inconsistency, which affects the score negatively.

Our submission of the result model onto Kaggle confirmed our expectation. There our combination of *Sentence-BERT* and *Logistic Regression* achieved a F1-Score of 79.19%. Our result model was applied to the exact same test dataset, with the only difference being that the test set used to evaluate our model was classified by its creators themselves.

Other possible reasons for the low performance include no hyperparameter tuning for the Logistic Regression Classifier, only one training epoch for the BERT Embedder with inaccurate and incomplete training samples. There is also a possiblity data being overfitted or inadequately cleaned Tweets. Therefore, we suggest using Random and Grid Search to optimize hyperparameters, increasing the number of training epochs as well as preprocessing samples before training the Sentence-BERT embedder, and finally applying stronger regularization and stricter data cleaning.

References

Böhm, T. (2018, December 30). *The General Ideas of Word Embeddings.* Retrieved June 15, 2021, from Towards Data Science: https://towardsdatascience.com/the-three-main-branches-of-word-embeddings-7b90fa36dfb9

Breiman, L. (2001). Random Forests. *Machine Learning*(45), 5-32. Retrieved from https://doi.org/10.1023/A:1010933404324

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL 2019*. Retrieved from https://arxiv.org/pdf/1810.04805v2.pdf

Dumais, S. (2005). Latent Semantic Analysis. *Annual Review of Information Science and Technology*(38), pp. 188-230. Retrieved from https://asistdl.onlinelibrary.wiley.com/doi/10.1002/aris.1440380105

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*(9), 1871-1874. Retrieved from https://www.csie.ntu.edu.tw/~cjlin/liblinear/#document

Hinton, G. (1989). Connectionist learning procedures. *Artificial intelligence*, 185-234. Retrieved from https://www.cs.toronto.edu/~hinton/absps/clp.pdf

Kaggle. (n.d.). *Natural Language Processing with Disaster Tweets*. Retrieved June 1, 2021, from

[4] Böhm (2018).

Kaggle: https://www.kaggle.com/c/nlp-getting-started/overview

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at ICLR 2013*.

Pennington, J., Socher, R., & Manning, C. (2014). *GloVe: Global Vectors for Word Representation.* Computer Science Department. Stanford, CA: Stanford University. Retrieved from https://nlp.stanford.edu/projects/glove/

Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.* Department of Computer Science. Darmstadt, Germany: Technische Universitaet Darmstadt. Retrieved from https://arxiv.org/pdf/1908.10084.pdf