UNIVERSIDAD
POLITÉCNICA
DE MADRID

INTELLIGENT SYSTEMS: NLP

NATURAL LANGUAGE PROCESSING WITH DISASTER TWEETS
ANALYSIS AND CLASIFICATION PREDICTIONS

Author:

CARLOS MOROTE GARCÍA (49312756C)

UPM

2021/2022

Madrid, January 30th.

1. Introduction

Since the emergence of social networks, people have been breaking down the barrier of their private lives little by little sharing more and more information through these networks where, normally, it is of public content. (Whiting & Williams, 2013). Thanks to the theory of gratification (Lariscy, y otros, 2011) by which these networks were sustained and makes users continue to use these networks daily we are faced with an inexhaustible well of information.

Social networks support multiple multimedia formats such as videos, images, texts, etc. From all of them can be extracted countless information that can be used in multiple ways.

Twitter is a social network that promotes constant posting, so many people end up posting anything they see. Based on this, we propose to carry out a study on these tweets using natural language processing in order to know if a disaster has occurred before the local authorities notify it. With this analysis we will generate a predictive model capable of knowing this very thing, if the tweet corresponds to a disaster or just someone sharing anything else.

2. Data

Dataset

For the execution of this project we will need some data, as for any data science project. For this we have turned to the Kaggle platform. This repository of datasets also organizes competitions based on these. Currently there is a competition in which they are looking for the best predictive model that is able to predict the goal of this project, that is, to predict based on a Tweet a disaster is occurring. In order to be able to make use of the information we have had to register for the competition. (appen, 2022)

The dataset is divided into a training set, which we know the classification and a test set, which we do not know the classification and only by means of a delivery to the competition we will know the quality of that classification. Additionally, the dataset has the location of the tweet and keywords (if available). Since in this work we intend to delve into the advantages of NLP these fields will be omitted and only the raw Tweet will be used.

Finally, we checked the variance of the problem, i.e., of the 7613 records in the dataset, 4342 correspond to cases where the Tweet did not correspond to a disaster. Therefore, the remaining 3271 did. So, without being a balanced problem, it cannot be considered to be highly unbalanced either.

Corpus

After this brief introduction to the problem and a brief description of the data available to us, it is time to analyze the texts. First, we will make use of a library (*'hunspell'*) that informs us of the grammatical errors that can be found in the tweets. We do this because due to the limit of characters per Tweet it is very common to make use of contractions so that everything fits

into one. It is also the case that people on the Internet do not write well. Finally, this method informs us that 21% of the words present in one way or another some grammatical error in both dataset, train and test.

Next, we will generate the **Corpus** and proceed to apply a series of transformations to it, both generic ones that are widely used in practically any project dealing with texts. We will also perform specific transformations based on the premise that we are dealing with Tweets, which may contain mentions, hashtags, images, web links, etc. For the Corpus creation and transformations applied we used the package '*tm*'.

The first transformation we will perform will be a first step to normalize the texts. We will perform more methods with the same object, but as a starting point we will only convert all characters to **lowercase**.

After this basic transformation we will proceed to apply the more specific ones. First, we will start by dealing with **hashtags**. Twitter has a tag system to allow quick searches by tags, as well as to allow grouping tweets by the same topic. These are the hashtags, which are identified with the hash symbol (#). These tags may contain useful information, but by their nature they tend to group multiple words without spaces, causing the algorithm to detect them as a single one. To extract the maximum knowledge from these tags we have made use of regular expressions by which it will detect the different words within a Hashtag, as long as they are differentiated with the first letter of each word capitalized. Therefore, the hashtag *#SpainOnFire* would be transformed into the three words that compose them: *Spain*, *On*, *Fire*. Additionally, they will be transformed into lowercase letters to match the transformation previously made.

Once we have dealt with hashtags, we will deal with **mentions**. These can be easily identified thanks to the at symbol (@). Since mentions are made with the username and this is unique and without following any logical structure like hashtags, we will proceed to delete them.

We proceed by dealing with the non-text elements of tweets and continue with **URLs** and images. Both are modeled by means of URLs. The URLs as a text source do not provide any useful information since they could be considered a succession of random characters that only have in common the beginning (http...). If one wanted to go deeper into this problem, these links are reverencing an image on the web, therefore, they could be extracted and analyzed with other algorithms in order to generate derived variables. As this is not the object of this work, the latter will not be implemented. Therefore, the urls will be eliminated.

The transformations applied to hashtags, mentions and URLs have been detected and treated by means of regular expressions. To see the expression built for each specific case, consult the notebook attached to this work where this expression is implemented.

After this we continue searching and eliminating the emoticons. We will also look for contractions between the different tokens. After identifying

them we will proceed to undo the contraction. This transformation is necessary in the normalization process.

To finish with the data transformations, we will proceed to apply the typical and basic transformations. These transformations are the elimination of numbers, the elimination of stop words, the elimination of punctuation marks, the elimination of consecutively repeated blank characters and stem the document.

Once all the Corpus it is preprocess we will expose an example of the resulting transformations. For instance, the Tweet "*@bbcmtd Wholesale Markets ablaze http://t.co/lHYXEOHY6C*" will be casted into "*wholesal market ablaz*". In this case we observe that the URL and the user mentioned were deleted. Also, the words where stemmed and lowercased.

Term Document Matrix

In this subsection we will analyze the tokens resulting from the Corpus modifications by means of a Term Document Matrix. We will study the most frequent tokens and eliminate those that are not so frequent (given a threshold). To do this we will make use of the *'tm'* library as well.

After generating the TDM with the whole Corpus we want to reduce its dimensionality (sparseness). Reducing the sparse means that sparsity refers to the threshold of relative document frequency for a term, above which the term will be removed. Relative document frequency here means a proportion. For instance, in this case we set *sparse* to 0.99 as the argument to *removeSparseTerms*() method, then this will remove only terms that are more sparse than 0.99. This leads to 103 terms as opposed to the initial 11552 terms.

We then calculate those terms that are most frequent. We obtain the graphs found in Figure A, which shows in a scatter plot all the terms ordered from most frequent to least frequent. We also see in Figure B a word cloud that shows exactly what those most common words are.

The Figure A figure shows that the most frequent terms are easily distinguished as they are much more frequent than the rest and there is a clear separation between them and the rest of the terms. On the other hand, looking at the word cloud we can study that those most frequent terms are correlated (but not all of them) with disasters, such as fire, bomb, evacuation, etc.

Modeling

At the end, and after processing and analyzing the Corpus, we started to experiment with machine learning models that will be in charge of making the predictions. We will experiment with support vector machine models, both linear and nonlinear, and with Naive Bayes.

In order to obtain evaluations of the models and to be able to compare them empirically we will first divide the Corpus for which we know the classification of its data. In this way we will use 70% of the training set for this very thing, training. While the remaining 30% will be used for evaluation.

The results obtained from these experiments are quite interesting. The worst model among these three is the linear SVM (f1: 0.6265), followed by the nonlinear SVM (f1: 0.7497) and finally the Naive Bayes (f1: 0.7718). We can observe that the difference between models is substantial. Especially between linear and non-linear SVM.

Multiple metrics have also been obtained from these models such as recall, accuracy, sensitivity and specificity. However, we have focused on the metric (F1) because it will be used to evaluate the test set internally in the Kaggle platform, i.e., data for which we do not know their classification.

Lastly, we generate a new model using all the training data. With this new model we will perform the prediction of the test dataset and the corresponding deliveries will be made in Kaggle. The results obtained are quite close to the values obtained in the testing phase explained above. The final results obtained with an unpublished dataset for our model is:

- **Naïve Bayes**: 0.78915
- **SVM**: 0.77597
- **SVM Linear**: 0.62212

The results of these models are quite promising given the superficial exploration and transformation that has been made of the information. Moreover, we have only made use of the text data when we still had two other variables that may (or may not) have provided information.

3. Conclusions

As conclusions of this work, we can state that natural language processing is a technology that is an excellent fit for the type of information we generate today. Although it is true that it has a higher level of complexity than an ordinary classification problem, we have shown in this short paper that good results can be obtained thanks to this technique.

In addition, if this work continues to be refined and new ways of aggregating knowledge are explored, it is more than likely that a perfectly functional model could be created to help identify those tweets that correspond to catastrophes. Thus helping to identify or track them.

All the code performed for this assigment can be found here.

4. References

appen. (2022). *Kaggle*. Obtenido de Natural Language Processing with Disaster Tweets. Predict which Tweets are about real disasters and which ones are not.: https://www.kaggle.com/c/nlp-getting-started/overview

Lariscy, W., Ruthann, Tinkham, F, S., Sweetser, & D, K. (2011). Kids these days: Examining differences in political uses and gratifications, internet political participation, political information efficacy, and cynicism on the basis of age. *American Behavioral Scientist, 55*(6), 749-764.

Whiting, A., & Williams, D. (2013). Why people use social media: a uses and gratifications approach. *Emerald Group Publishing Limited*.

Appendix

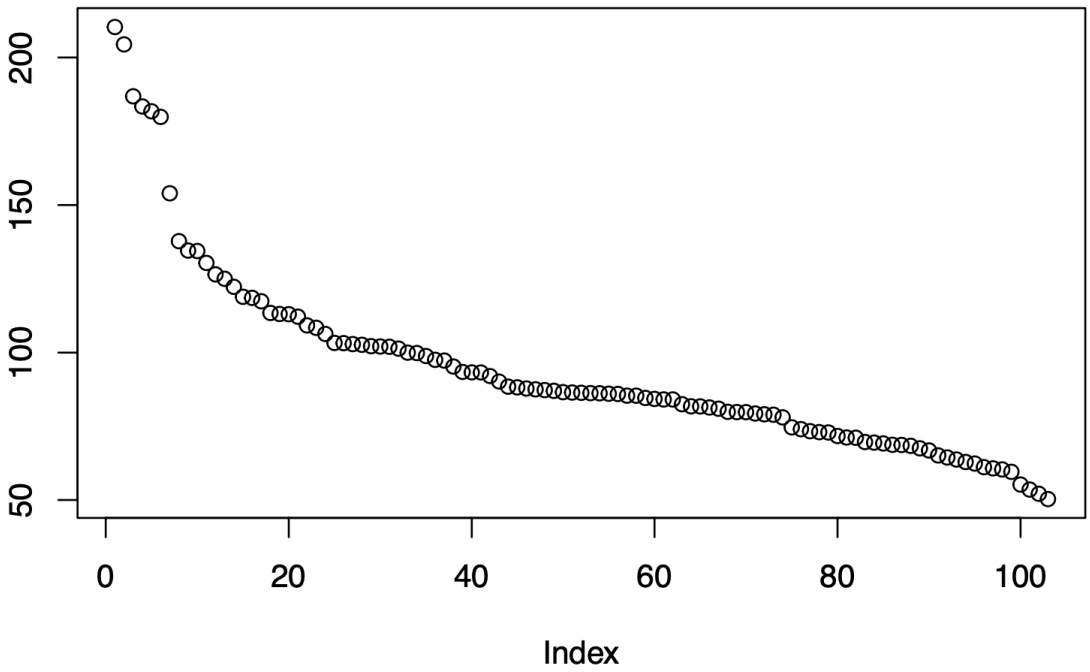

Figure A. Frequency of all terms sorted.

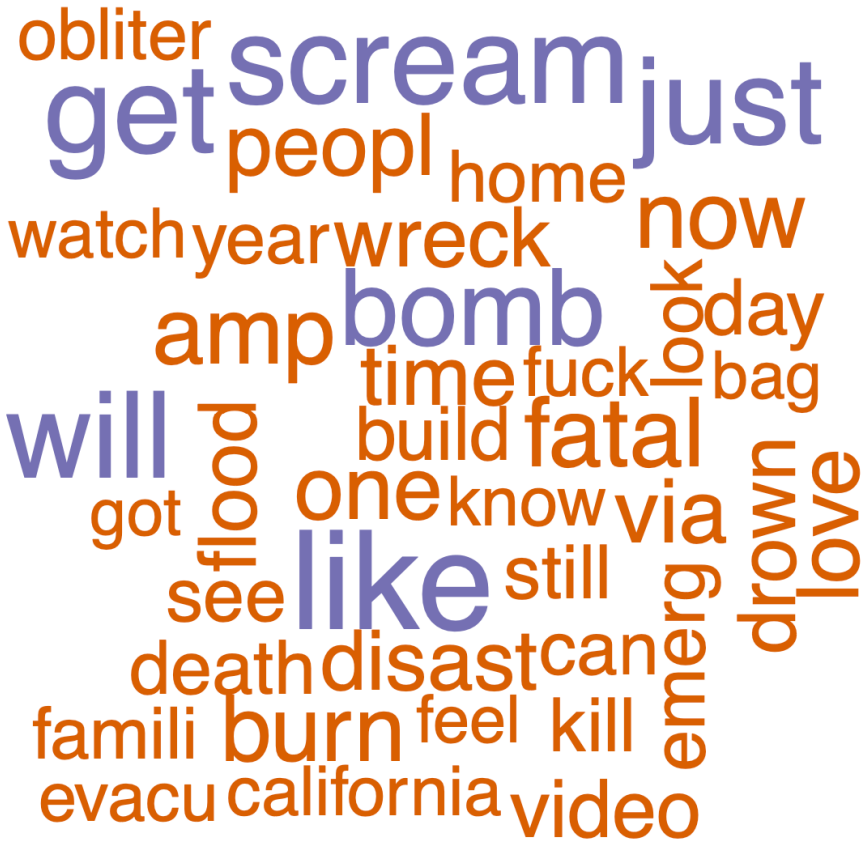

Figure B. Wordcloud of the most frequent terms.