Sentiment analysis

Hate Speech Detection: Creation of a dictionary of the bully language

Martina Corsini ID 944506
Camilla Gotta ID 945522

MSc Data science and economics
Università degli studi di Milano

June 2021

Abstract. The aim of this research is to create a dictionary of the bully language so as to be able to distinguish offensive/abusive tweets from non-offensive ones, in order to suggest a possible solution to the propagation of hate speech on social media. In order to perform this task, the research exploits the Local Interpretable Model-agnostic Explanations(LIME), a Phyton library that explains how the classifiers perform on data, and Latent Dirichlet Allocation(LDA), an unsupervised method based on topic modeling. The resulting dictionaries contain the most used words in the dataset to insult someone or something. We then combine the dictionaries obtained from these two methods, considering both the union and the intersection of the two sets. The last step is the validation of the union and intersection dictionaries.
Keywords: hate speech, classification, supervised learning, LIME, unsupervised learning, LDA, dictionary.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, the increasing propagation of hate speech on social media has had great resonance not only among computer science's experts but also in the public sphere: this is the case, for example, of the debate around the permanent suspension of Donald Trump's Twitter account "due to the risk of further incitement of violence"[1]. It is therefore crucial to develop effective countermeasures against the spiral of violence on social media.
In the last years, large number of methods has been developed for automated hate speech detection online[2].The aim of this research is to create a dictionary of the bully language so as to be able to distinguish offensive/abusive tweets from non-offensive ones. In order to perform this task, the research exploits the Local Interpretable Model-agnostic Explanations(LIME), a Phyton library that explains how the classifiers perform on data, and Latent Dirichlet Allocation(LDA), an unsupervised method based on topic modeling. The resulting dictionaries contain the most used words in the dataset to insult someone or something. We then combine the dictionaries obtained from these two methods, considering both the union and the intersection of the two sets. The last step is the validation of the union and intersection dictionaries.

1.1 Dataset

The dataset used in this analysis is the Bully Messages dataset[3] taken from Kaggle. There are 1065 tweets tagged as normal text or bully text, respectively 638 and 427 records.

2 Research question and methodology

The aim of the project is to create a dictionary useful in the detection of offensive language. In the next paragraphs, we will take into consideration two different models in order to reach this goal. Firstly, we will use Local Interpretable Model agnostic Explanations(LIME), a Phyton library that explains how the classifiers perform on data, to retrieve the most incisive words in the choice of the logistic regression classifier- the one with the highest performance among those we have considered- of labelling a tweet as abusive or normal. Subsequently, Latent Dirichlet Allocation(LDA), an unsupervised method based on topic modeling, will be exploited in order to find the unigrams and bigrams with the highest frequencies in the tweets labelled as bully.

2.1 Data cleaning and preprocessing

The kind of data one gets from social media is usually unstructured. It contains, for example, unusual text and symbols that need to be cleaned up so that a machine learning model can grasp their meaning, but also typos and abbreviations. The reliability of the model is highly dependent upon the quality of the data. It may be helpful to get rid of unhelpful parts of the data, or noise, by converting all characters

to lowercase, removing stop words and punctuation marks. In particular, removing stopwords allows the model to consider only key features because these words typically don't contain much information: there are many variations of those words that do not bring any new information and create redundancy, ultimately bringing ambiguity when training machine learning models for predictions. Finally, text must be tokenized and vectorized, transforming it into a vector of numbers that a machine can understand. In order to clean and preprocess the dataset, Scikit-learn provides the sklearn.pipeline module, that assembles several steps that can be cross-validated together while setting different parameters.

2.2 Local Interpretable Model-agnostic Explanations

2.2.1 Choice of the classifier

Classification algorithms are supervised Learning techniques: therefore, after the splitting of the dataset in train and test set, respectively 80% and 20% of the total observations, the machine uses the training data to build the model which aims to assign the correct label to the new observations based on past examples. In order to perform classification and compare the results, five different classifiers have been implemented.
The first classifier we took into consideration is the **random forest**, an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.
Then, **logistic regression** has been applied, a method that uses a logistic function to model a binary dependent variable: it considers $P(y^* = 0|x^*)$ and $P(y^* = 1|x^*)$ and it selects the label corresponding to the class with the higher probability.
The Naive Bayes algorithms are a class of classifiers which classifies a document with terms from the most likely class, given the corresponding set of features (assuming independence). More in detail:
In **Multinomial Naïve Bayes** feature vectors represent the frequencies with which certain terms have been generated by a multinomial classificator with parameters $(p_{1,c}, \dots, p_{i,c}, \dots)$ where $p_{i,c}$ is the probability that term d_i occurs in class c.
In **Bernoulli Naïve Bayes** features are binary variables (term is present / absent). It is useful for classifying short texts.
Complement Naïve Bayes is an adaptation of the standard Multinomial Naive Bayes algorithm that is particularly suited for imbalanced data sets.
We subsequentely evaluated the performance of the models, retrieving the few metrics: as shown in Table 1, the logistic regression classifier exhibits the best levels in terms of accuracy ($\frac{TP+TN}{TP+FP+TN+FN}$) and precision ($\frac{TP}{TP+FP}$) among those we have considered. Therefore, we will use this classifier in the implementation of LIME.

Model	Precision	Accuracy
Random Forest	0.65	0.65
Logistic Regression	0.71	0.70
Multinomial Naïve Bayes	0.67	0.68
Bernoulli Naïve Bayes	0.71	0.69
Complement Naïve Bayes	0.67	0.68

Table 1: Classification results

2.2.2 Application of LIME

Local Interpretable Model-agnostic Explanations (LIME) is a useful library that explains individual predictions for text classifiers: it requires the classifier to implement a function that takes in raw text and outputs a probability for each class. This tool takes any black box model, such as a classifier, and explains how it performs on data, so the reasons why the classifier labels the data in a certain way. The output of LIME, as shown in Fig. 1, is a list of explanations, reflecting the contribution of each feature to the prediction of a data sample. This provides local interpretability, and it also allows to determine which features have most impact on the prediction[4].

Figure 1: Functioning of LIME

In the case of a dataset containing texts, this method makes it possible to find the most relevant words to explain why the classifier chose to label a single text in a certain way. Specifically, in our case it makes it possible to find the most incisive words in the choice of labelling a tweet as abusive or normal. Unfortunately, at the moment, LIME restricts the explanation of terms to unigrams, so we were not able to retrieve the bigrams [5]. We applied this method on each tweet contained in the test set and then we retrieved a dictionary containing the most important words for the explanation of the functioning of the classifier on the entire test set, with the associated weights, namely the sum of the contribution of each word to the prediction of a single tweet. We took into consideration only the words with a negative impact. Finally, in order to construct the dictionary based on LIME words, only nouns, adjectives and verbs have been selected through Spacy, a library that tags up each of the tokens in a document with a part of speech. The Fig. 2 shows the most important words retrieved with

LIME in a intuitive manner (the biggest words are the one with the highest associated weights).

Figure 2: Bully Word Cloud from LIME

2.3 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative statistical model that generates topics based on word frequency from a set of documents. It's one of the most popular topic modeling methods. Topic models are unsupervised machine learning methods that help discover hidden semantic structures in a collection of texts. Clusters of similar words are called topics. We chose the number of topics at the beginning of our process. In the application of this method, it is important not to be limited to single terms, due to the fact that there are terms that, taken individually, are not connoted with hatred or bully language but, taken in combination, end up being offensive. After a good cleaning up of the text data, we therefore proceeded with the creation of the skipgrams. The skipgrams of a sentence $w_1, ..., w_m$ can be described as:

$$w_{i1}, w_{i2}, \ldots, w_{in} | \sum_{i=1}^{n} i_j - i_{j-1} < k$$

where parameter k refers to the max skip-distance and n to the "grams" or subsequence length. We set k equal to the length of the sentence and n equal to 2, in order to obtain the bigrams. Each topic is a combination of keywords, and each keyword contributes with a certain weight to the topic. We applied this method on the part of dataset labelled as bully, in order to find the words with the highest frequencies in the offensive tweets. Once again, we considered only nouns, adjectives and verbs, the most important parts of speech for the detection of hateful language. Fig. 3 shows the results.

Figure 3: Bully Word Cloud from LDA

3 Experimental results

The last step is the aggregation of the results obtained from the application of LDA and LIME. In order to make these results comparable, it is necessary to normalize the weigths associated to the words, so to have values ranging from 0 to 1. We then plotted the scatterplot (Fig. 4) in order to visualize which words or bigrams have a greater impact on either the dictionary created with the LDA method, the one resulting from the exploitation of the LIME library or on both of the dictionaries.

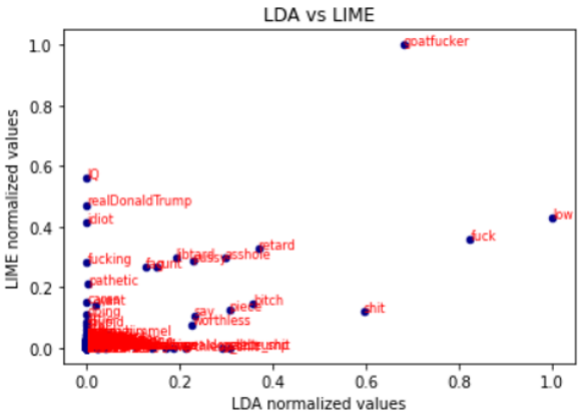

Figure 4: Scatterplot of bullism words

We will consider both the union and the intersection of the two sets: when validating them, we expect that the dictionary resulting from the union will have higher recall level but lower precision compared with the intersection dictionary.

3.1 Intersection Dictionary

Starting from the words obtained by the two previous methods, we firstly created the intersection dictionary, considering the words contained in both the LDA and the LIME dictionary and the mean of the corresponding weights. The resulting words are shown in the Fig 5. In order to validate the obtained dictionary, it was applied on our original dataset: a value of offensiveness for each tweet has been retrieved by summing up the weights of the offensive words that appear in the text. This dictionary is more robust compared to the union one: the selected words, in fact, are retrieved both by the LIME and the LDA methods, and they are therefore more likely to be actually offensive words. On the other hand, however, this dictionary will give an higher percentage of false negative, due to its small size: in fact, a bigger set of tweets will be wrongly classified as normal texts, due to the fact that they will not include the words contained in the intersection dictionary, although they are offensive texts. The confusion matrix of Fig. 6, compared to the one of the union, shows this theoretical point. A function was created in order to evaluate the level of accuracy, precision, recall and F1_score, respectively 0.62, 0.52, 0.86, 0.65.

Figure 5: Dictionary from intersection

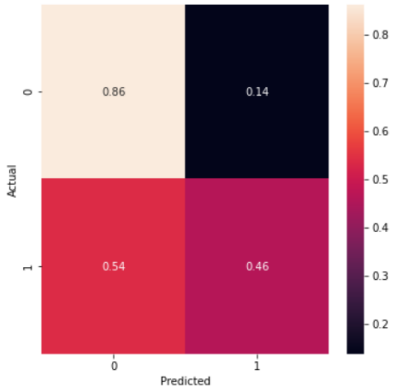

Figure 6: Confusion Matrix form intersection

3.2 Union Dictionary

The same analysis was done considering the union of the words contained in at least one among the LDA and the LIME dictionary (Fig. 7). In this case, as expected, the number of false negatives is lower than in the case of the intersection (Fig. 8), but the price is of lower precision: indeed, accuracy, precision, recall and F1_score are respectively 0.53, 0.46, 0.97, 0.62. This is the result of having a larger word sample.

Figure 7: Dictionary from Union

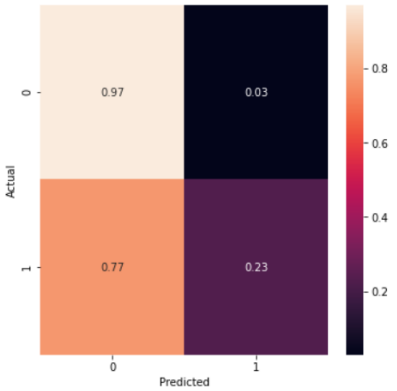

Figure 8: Confusion Matrix form union

4 Concluding remarks

The initial goal was to create a dictionary able to detect if a corpus could be a bully text or not. Both the intersection dictionary and the union one perform quite well in the classification of the bully tweets, but they exhibit lower performance in terms of identification of the non-offensive texts. This result can be attributed to two different issues: the LIME library, since it restricts the explanation of terms to unigrams, makes it impossible to analyze the word in its context and it therefore retrieves a lot of non-offensive words; in the spoken language, a lot of dirty words are used to emphasize the concepts expressed rather then to insult. It could be useful to identify a threshold in order to retrieve only the tweets that have high values of offensiveness in order to mitigate this problem. Moreover, it is also possible to increase the size of the two dictionaries by adding synonyms of the offensive words we found, making them more complete. WordNet can be used in order to perform this task, as the following code explains.

```
for syn in wn.synsets("idiot"):
        for name in syn.lemma_names():
            print(name)

[output] idiot
         imbecile
         cretin
         moron
         changeling
         half-wit
         retard
```

Listing 1: WordNet synonyms example

References

[1] "Permanent suspension of @realDonaldTrump", Twitter Inc.,Friday, 8 January 2021

[2] https://ieeexplore.ieee.org/document/9320755 by Rini Rini, Ema Utami, Anggit Dwi Hartanto

[3] Kaggle.com, *Bully Messages dataset*. Available at: https://www.kaggle.com/imranzaman5202/bully-messages-detection

[4] https://towardsdatascience.com/understanding-model-predictions-with-lime

[5] https://github.com/marcotcr/lime/issues/7