Profiling Hate Speech Spreaders on Twitter: PAN Shared Task

Javier Martínez Bernia, Jaime Ferrando Huertas

May 2021 @ MIARFID, UPV

Contents

1 Introduction

In this document, we present our participation on the PAN shared task of 'Profiling Hate Speech Spreaders on Twitter'. This task consists in determining if an author spreads hate speech on social media given a twitter feed. We built different models based on machine learning and Natural Language Processing techniques in order to solve the problem.

In the next sections, we will discuss about what experiments we carried out and how we arrived to the final solution.

2 Data acquisition and preprocessing

The data is organized in two folders, one for tweets in Spanish and one for tweets in English. In each of the folders, we have 200 XML files, one per author with 200 tweets each file. We also have a truth.txt file with the labels for each author: 0 for not hate spreader and 1 for hate spreader. This will be our training set. With this data, we will perform a 10-fold cross-validation when we train our models in order to calculate the accuracy.

We built a parser that converted all the XML files and returned a CSV file, which is easier to manipulate. Before building the models, we applied some processing techniques to the text. The data has been prepared in two separate ways. The first one is to join all the tweets from an author and use them as one single input for the models. The second point of view is to take tweet by tweet as input. For both approaches, we applied the same techniques: Twitter tags removal, replace of emojis by their description, stopwords removal and stemming.

2.1 Twitter tags removal

After analyzing the text, we found some twitter tags like '#USER#' or '#HASHTAG#'. We just processed the text and removed them, because these words don't provide any meaning to the tweet.

2.2 Emojis replacement

In this technique we used two python libraries (emoji and demoji) in order to replace every emoji to a textual description. We used one library for Spanish and the other one for English. This seems that helps the models to better performance.

2.3 Stopwords removal

One of the most popular techniques in Natural Language Processing is stopwords removal. This words are the most common words in a language and don't have any meaning by themselves. With this, we can compact our data and make it more meaningful.

2.4 Stemming

One of the techniques that we tried is stemming. It reduces the words to their base form or root. It's a very well-known technique for processing natural language.

3 Solution proposal

We propose different models based on machine learning. We used the Scikit-learn library in order to build and experiment with most of the models. In addition, we used a Deep Learning model from HuggingFace, which is pretrained for detecting hate speech.

3.1 Scikit-learn

For the models used with Scikit-learn we first need to vectorize the data. In order to do that, we used two types of vectorizer which are implemented in the library. These are CountVectorizer and TfidfVectorizer. They extract some features from the text and return a matrix of data, which we can give to the models as input.

Once we have the data in a vector, we can train the models. The models we have tested with are the following ones:

1. Gradient Boosting Classifier (GradBoost)
2. Linear Support Vector Machine (L-SVM)
3. Non-Linear Support Vector Machine (SVM)
4. Stochastic Gradient Descent Classifier (SGD)
5. Multi-Layer Perceptron Classifier (MLP)
6. K-Nearest Neighbors Classifier (KNN)
7. Random Forest Classifier (RandForest)
8. Gaussian Naive-Bayes (GaussianNB)
9. Decision Tree Classifier (DecisionTree)

3.2 Hugging Face

We also tried a model from Hugging Face, twitter-roberta-base. This model is a BERT based model, specifically a Bi-directional transformer already trained by Hugging Face in enormous datasets. Using this model requires no training from our side and allows us to use a model trained on more generic data, something that can be highly valuable when having low size datasets. This model was trained only for English language.

4 Experiments

After preparing the data and defining the models, we experimented with them. We carried out many experiments doing a 10-fold cross-validation to our training dataset. In the case of the pretrained model from HuggingFace we used our dataset only for validating it. Now, we present the different experiments which have been done and the metrics we used in both approaches: joining all tweets from every author into one whole piece of text and processing tweet by tweet.

4.1 Joined tweets

In this approach the metrics are very easy to define because we have one sample and one label per author. We trained all the models that we proposed from the scikit-learn library and used accuracy as metric. We performed a grid search in order to get the best parameters for every model. The results we obtained are summarized in the next table:

Model	**Text preprocess**	**Vectorizer**	**Acc. (es)**	**Acc. (en)**
GradBoost	Full	Count	0.75	0.69
L-SVM	Full	Count	0.79	0.70
SVM	Full	Count	0.81	0.72
SGD *	Full	Count	**0.83**	0.71
MLP *	Full	Count	0.82	**0.75**
KNN	Full	Count	0.76	0.62
RandForest	Full	Count	0.79	0.72
GaussianNB	Full	Count	0.75	0.61
DecisionTree	Full	Count	0.68	0.61
GradBoost	Full	TF-IDF	0.73	0.59
L-SVM	Full	TF-IDF	0.78	0.70
SVM	Full	TF-IDF	0.77	0.70
SGD	Full	TF-IDF	0.76	0.67
MLP	Full	TF-IDF	0.78	0.68
KNN	Full	TF-IDF	0.77	0.62
RandForest	Full	TF-IDF	0.73	0.64
GaussianNB	Full	TF-IDF	0.74	0.53
DecisionTree	Full	TF-IDF	0.63	0.56
MLP	No-Stemming	Count	0.66	0.65
MLP	No-Stemming	TF-IDF	0.51	0.56

Table 1: 10-fold cross-validation scores on the training set.

As we see in table 1, the best results for Spanish are achieved with a SGD classifier. For English, we achieved an accuracy of 0.75 with a Multi-Layer Perceptron. This model performs very well for Spanish too (0.82 accuracy),

that is why we tested it without applying Stemming to the text, but the results did not improve.

With respect to the HuggingFace pre-trained model we could not experiment with all the tweets in one single string, because the size of the input data was too large and transformers models can not fit those sizes yet.

4.2 Tweet by tweet

As we could not experiment with the pretrained model giving all the tweets in the same string as input, we tried to find a way to process them one by one. In this approach, we processed every tweet by itself. We used the same label as the author for every tweet from this author on the training set. We knew there would be some errors on the labeling but at least all the hateful tweets would be labeled as hateful.

After labeling tweet by tweet, we defined how we were going to label each author. After classifying all the tweets from an author, we used a function that given the number of tweets classified on each class and a ratio as a parameter, it returns the label for the author. This function can be written as:

$$Label(n_{pos}, n_{neg}, ratio) = \begin{cases} 1 & \text{if } n_{neg} > ratio \cdot n_{pos} \\ 0 & \text{other cases} \end{cases}$$

being n_{neg} the number of tweets from the author classified as hateful, n_{pos} the number of tweets classified as non hateful and $ratio \in \mathbb{R}$ a parameter for balancing.

For this approach, we only tested the data in English because the pretrained model had been trained with English text. After experimenting with a model from Scikit-learn and with the model from HuggingFace, we present our results in the next table:

Model	**Text preprocess**	**Ratio**	**Accuracy (en)**
twitter-roberta-base	None	0.2	0.50
twitter-roberta-base	None	0.4	0.54
twitter-roberta-base	None	0.6	0.58
twitter-roberta-base	None	0.8	0.63
twitter-roberta-base	None	1.0	**0.66**
twitter-roberta-base	None	1.2	0.57
twitter-roberta-base	None	1.4	0.57
MLP	Full	0.5	0.50

As we can see in the table, we couldn't improve the better results achieved by the models in the other approach, but we got better results than some of them.

5 Selected models for testing

For the competition we have selected the SGD for Spanish and MLP for English. We sent our predictions under the team of jaiferhu@inf.upv.es and jamarbe2@inf.upv.es and achieved the following results:

Task	Accuracy
EN	61%
ES	82%
AVG	71.5%

Our Spanish results are in line with what our cross validation test reported, but English is 14% behind from our results. This could be due to the English test set being significantly different from the training that we aggressively overfitted in the training steps. Another reason could be that we have missed preprocessing techniques for the English language, we are quite excited to see what other teams achieved and learn from their reports.

6 Conclusion

With this shared task we have expanded on the knowledge of hate speech detection from this lecture second lab session. We had some basic knowledge of what models to use but in this task we have taken it a step further with better text processing and even trying pre-trained models. Text-processing has shown itself to be a good improvement on performance. We are happy with our Spanish results but a bit down due to English results being quite different from our testing results.

We would like to highlight the higher performance of the twitter-roberta-base over our MLP when running the experiment *tweet by tweet*. If the transformer based model could accept higher input sizes we think it could have beat our MLP. Worth keeping in mind this model for future use cases when we have the tweet by tweet scenario.