Profiling Hate Spreaders using word N-grams

Notebook for PAN at CLEF 2021

Jorge Alcañiz[1], José Andrés[1]

[1]*Universitat Politècnica de València*

Abstract
With the rise of social media over the last decade, the amount of content that is published every day on the internet has become huge. Unfortunately, as the amount of published content grows, the amount of hate speech that can be found on social media also grows. This fact motivates the creation of systems that could automatically detect this undesired behaviors in order to report them to the competent authorities. With this purpose, we have developed a system that detect users which could be considered as hate spreaders employing a TF-IDF vectorizer in combination with an SVM, achieving an accuracy of 81% over the Spanish dataset and 69% over the English dataset.

1. Introduction

Hate speech is commonly defined as a propaganda of ideas based on the superiority of a group of people because of their race, color or ethnic origin. This problem is not novel, as it has been present in our society during centuries, but due to the rise of social media it has reached unprecedented levels. Given the huge amount of content generated by the users and the impossibility to manually check all the content, the automatic detection of hate speech has become a relevant task.

With this purpose, the aim of this competition is to automatically detect hate speech, but employing an author profiling perspective instead of a tweet perspective. Therefore, we are interested in detecting which users could be considered as hate spreaders.

This paper presents our participation in the Author Profiling task at PAN [1] for detecting hate spreaders [2]. Our method follows the ideas presented in [3], which were focused on employing n-grams of chars and words as features and an SVM as classifier. Moreover, we have tried different classifiers and we have compared the obtained accuracies between them. The rest of this paper is structured as follows: Section 2 describes the dataset used for this shared task, Section 3 presents the preprocessing that we have applied for each language, Section 4 presents our approach to the problem, the results obtained per each model and a discussion of them and finally Section 5 summarizes the paper and proposes possible future works.

CEUR-WS.org/Vol-2936/paper-150.pdf

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania

joralvi1@inf.upv.es (J. Alcañiz); joanmo2@inf.upv.es (J. Andrés)

© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

2. Corpus

The dataset of this competition was composed by 200 authors for each language, where each author was composed by 200 tweets. From the 200 authors, 100 were hate spreaders and the other 100 were not. Moreover, we would like to remark that all the urls, links, hashtags and user mentions on the corpus were masked by a unique token for each type.

3. Preprocessing

The objective of this step is to reduce the vocabulary size by merging different token occurrences that are referring to the same concept. For the preprocessing of the dataset we have followed these steps.

In first place, as we are interested in detecting hate speech at author level, we have concatenated all the tweets of each author into one single string. Then, we have converted the text to lowercase and we have replaced all the emojis and emoticons by the token "emoji" employing a regular expression. After that, we have applied a different linguistic preprocessing for each language: in the English dataset, we have replaced some contractions by their expanded form, (for example, the token: "you'd" has been replaced by "you would", the token "it's" has been replaced by "it is", etc), meanwhile in the Spanish dataset we have replaced some words by their homologous colloquial tokens (for instance, the token "por" has been replaced by "x" and the token "que" has been replaced by "k"). Then, we have removed all the punctuation signs such as points, commas, exclamation signs, etc. Moreover, we have reduced different forms of expressing laugh such as "hahahah", "ahahha","jajajaja", "lol","lmao" to the token "haha". Finally, we have removed the stopwords and performed stemming on both datasets.

4. Our Approach

In this task we have considered the feature extraction process and the machine learning model estimation as a combined optimization process. Therefore, we have performed an extensive grid search to choose the best combination of hyper-parameters for the tfidf vectorizer and the different machine learning models employed. To assess the performance of our classifier, we have performed a 10-fold cross validation over the training dataset.

For feature extraction, we have employed a TF-IDF [4] vectorizer, which allows us to quantify the importance of every sequence of terms present in the corpus, multiplying the term frequency in the text by the inverse document frequency of the term in the corpus. The hyper-parameters of the Vectorizer are the following: "analyzer", which denotes the level at which the feature extraction is performed (either word level or character level), "ngram_range", which denotes the order of the employed language model and "min_df", which removes those n-grams whose document frequencies are lower than a given threshold.

Before starting to discuss the selected classifiers, we would like to remark that we have decided to avoid employing deep learning for this task. This is due to the fact that we only have 200 samples for each author, and given that deep learning is usually data-hungry, it could easily turn into overfit. Therefore, among all the possible machine learning models, we have chosen the

Table 1
Hyper-parameters tested during grid search for Tfidf Vectorizer.

Vectorization	Hyper-parameters
Analyzer	{'word','char','char_wb'}
Ngrams	{(1,1), (1,2), (2,2)}
min_df	{1,2,3,4,5,6,7,8,9,10,11}

following: logistic regression (LR) [5], Naive Bayes (NB)[6], Support Vector Machine (SVM) [7], Random Forest (RF) [8], multiple linear models trained with Stochastic Gradient Descent (SGD) [9] and K-nearest neighbor (KNN) [10]:

- **Logistic regression:** Basic algorithm for binary classification, equivalent to "Linear regression" but taking a logit function. The hyper-parameters taken for this model are, "penalty" of L2, with a "solver" liblinear and the regularization coefficient "C".

- **Naive Bayes:** A well-known technique which has been employed for tackling many information retrieval problems. For this model, the hyper-parameters used are the smoothing parameter "alpha" and "fit_prior", which denotes if the model wants to learn the prior of every class in the model.

- **Support Vector Machine:** A linear classification model that employs as decision boundary the maximal margin hyperplane. This fact is relevant to our task, given that due to the moderate size of the dataset and the large number of extracted features, many possible decision boundaries exist. Moreover, this model also allows solving non-linear problems by applying the appropriate kernel function. The tuned hyper-parameters are the regularization coefficient "C" and the employed kernel.

- **Random Forest:** An ensemble model of multiple decision trees. The tuned hyper-parameters are "criterion", to measure the quality of every split, and "min_samples_leaf", which denotes the number of samples required to be a leaf node.

- **Stochastic Gradient Descent classifier:** An optimization technique that allows us to fit linear classifiers employing gradient descent. The hyper-parameters selected for this model are the following: "'loss" criteria, where each one of the losses represent a linear classifier (for example, having as loss "log" will result in a logistic regression model with SGD training). In this task we have used an L2 "penalty" and an "'alpha" to regularize terms.

- **K-Nearest Neighbor:** A well known non-parametric classifier where the class for each test sample is computed from a simple majority vote of the K nearest neighbors of each point. The tuned hyper-parameters for this model are the “weights”, which could be uniform (each point is weighted equally) or proportional to the distance from their neighbors. Among all the possible distance metrics, we have tried the following: Euclidean distance, Manhattan distance and Minkowski distance. Finally, we have also tuned the number of neighbors “n_neighbors” to consider during the voting and the “leaf_size” of each branch.

Finally, we would like to remark that we have used scikit-learn [11] as toolkit for the employed machine learning models.

Table 2
Best hyper-parameters and accuracy obtained performing a 10-fold cross validation.

Model	Language	Model Hyper-params	Tf-idfVect Hyper-params	Acc. (%)
LR	EN	C: 100	word, unigram and bigrams, min_df: 11	68.50
	ES	C: 100		80.50
NB	EN	alpha: 0.25, prior: False	word, unigram and bigrams, min_df: 8	65.55
	ES	alpha: 0.25, prior: True		79.00
RF	EN	criterion:“gini”, depth: 4, min_samples: 10	word, unigram and bigrams, min_df: 8	67.00
	ES	criterion:“gini”, depth: 4, min_samples: 8		80.50
KNN	EN	weights:“distance”, metric: “euclidean”, neighbors: 5, leaf_size=20	word, unigram and bigrams, min_df: 8	65.00
	ES	weights:“distance”, metric: “euclidean”, neighbors: 10, leaf_size=20	word, bigrams, min_df: 9	77.00
SGD	EN	alpha:0.01, loss:“perceptron”	word, unigram and bigrams, min_df: 9	70.00
	ES	alpha:0.001, loss:“hinge”		80.00
SVM	EN	C: 0.1, kernel: “linear”	word unigrams, min_df: 4	70.50
	ES	C: 1, kernel: “linear”	word unigrams, min_df: 10	80.90

If we take a look at Tab. 2, it can be seen that linear models such as SVM, logistic regression and SGD classifier have performed particularly well at this task. This is due to the fact that the number of features is much larger than the number of samples, making it feasible to separate the two classes linearly. From these models, the best performing one has been the SVM, achieving an 80.90% of accuracy for the Spanish dataset and a 70.50% of accuracy for the English dataset.

Again, this is motivated by the fact that the SVM chooses the separating hyperplane with the maximal margin from among all the possible separating hyperplanes. Other techniques such as Random Forest, Naïve Bayes classifier and K-NN also performed well, but they did not achieve the results obtained by the linear models.

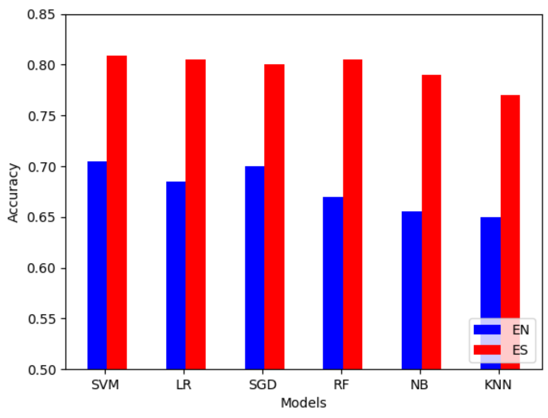

Figure 1: Accuracies obtained by each model during 10-fold CV.

As final model, we have chosen an SVM for both datasets, given that it is the classifier which has achieved the highest estimated accuracy in both languages. Finally, we have trained an SVM for each language, employing the full train dataset and the hyper-parameters described in Tab. 2. The results obtained in the competition with these models are the following:

Table 3
Train and test obtained accuracies employing an SVM.

Language	Estimated acc. (Train 10 - CV)	Test Acc.
ES	80.90%	81%
EN	70.50%	69%

It can be seen that our estimation of the performance of the system has worked reasonably well, matching the test accuracy very similar levels to the ones predicted during training. Test accuracy results have been provided by the TIRA evaluation platform [12].

5. Conclusion

To sum up, we have described the methods employed for this task. We have detailed how our whole system works, from the preprocessing step to the estimation of the best hyper-parameters for the feature extractor and the machine learning models. We have also seen that our estimation of the accuracy employing a 10 fold CV is consistent with the test results. As future works, we would like to test an ensemble model of different classifiers to see if it can beat the performance achieved by our SVM.

References

[1] J. Bevendorff, B. Chulvi, G. L. D. L. P. Sarracén, M. Kestemont, E. Manjavacas, I. Markov, M. Mayerl, M. Potthast, F. Rangel, P. Rosso, E. Stamatatos, B. Stein, M. Wiegmann, M. Wolska, , E. Zangerle, Overview of PAN 2021: Authorship Verification,Profiling Hate Speech Spreaders on Twitter,and Style Change Detection, in: 12th International Conference of the CLEF Association (CLEF 2021), Springer, 2021.

[2] F. Rangel, G. L. D. L. P. Sarracén, B. Chulvi, E. Fersini, P. Rosso, Profiling Hate Speech Spreaders on Twitter Task at PAN 2021, in: CLEF 2021 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2021.

[3] J. Pizarro, Using n-grams to detect fake news spreaders on twitter, in: CLEF, 2020.

[4] K. S. Jones, A statistical interpretation of term specificity and its application in retrieval, Journal of documentation (1972).

[5] R. Pearl, L. J. Reed, On the rate of growth of the population of the united states since 1790 and its mathematical representation, Proceedings of the National Academy of Sciences of the United States of America 6 (1920) 275.

[6] M. E. Maron, J. L. Kuhns, On relevance, probabilistic indexing and information retrieval, Journal of the ACM (JACM) 7 (1960) 216–244.

[7] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (1995) 273–297.

[8] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[9] L. Bottou, Online learning and stochastic approximations, On-line learning in neural networks 17 (1998).

[10] B. W. Silverman, M. C. Jones, E. fix and jl hodges (1951): An important contribution to nonparametric discriminant analysis and density estimation: Commentary on fix and hodges (1951), International Statistical Review/Revue Internationale de Statistique (1989) 233–238.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.

[12] M. Potthast, T. Gollub, M. Wiegmann, B. Stein, TIRA Integrated Research Architecture, in: N. Ferro, C. Peters (Eds.), Information Retrieval Evaluation in a Changing World, The Information Retrieval Series, Springer, Berlin Heidelberg New York, 2019. doi:10.1007/978-3-030-22948-1_5.