Profiling Hate Speech Spreaders on Twitter using stylistic features and word embeddings

Notebook for PAN at CLEF 2021

Lucía Gómez-Zaragozá [1] and Sara Hinojosa Pinto [2]

[1] *Instituto de Investigación e Innovación en Bioingeniería, Universitat Politècnica de València, Valencia, Spain*
[2] *Multiscan Technologies S.L., Universitat Politècnica de València, Valencia, Spain*

Abstract

This paper presents the different solutions proposed for the Profiling Hate Speech Spreaders on Twitter task at PAN 2021, which consists of classifying each author as hater or no hater from a set of tweets, for Spanish and English languages. The given approaches are different for each language. For Spanish, an ensemble of LSTM and a Logistic Regression model trained with stylistic features is used. For English, an ensemble of SVC and Random Forest model, also with stylistic features, is proposed. Our solutions achieved an accuracy of 83% in Spanish and 58% in English, resulting in an overall accuracy of 70.5% in the task ranking.

Keywords

Hate speech, author profiling, natural language processing, NLP, embeddings, LSTM, Twitter, machine learning

1. Introduction

Automatic hate speech detection on social media has become a topic of growing interest in the artificial intelligence community and particularly, in the area of Natural Language Processing [1]. Although different definitions can be found in the literature, hate speech is commonly described as language that attacks or disparages a person or a group based on specific characteristics that include, among others, physical appearance, nationality, religion or sexual orientation [2]. Given the huge amount of user-generated content and the rapid dissemination of information these days, being able to identify not isolated hate speech comments but hate speech spreaders is a key first step in trying to prevent hate speech from spreading in online communications.

This paper describes the proposed models for the PAN 2021 Profiling Hate Speech Spreaders on Twitter [3], which is one of the three proposed tasks at CLEF 2021 [4] deployed on TIRA platform [5]. The dataset provided in the shared task consisted of a balanced set of users that have shared some hate speech tweets, labeled as haters and non-haters otherwise. It was provided in two languages, namely Spanish and English. For each of them, the dataset it included 200 different users and 200 tweets per user. As recommended by the shared task, we presented a different solution for each language. For the Spanish dataset, an ensemble of LSTM and a logistic regression model trained with stylistic features is proposed, which achieved 83% of accuracy in the provided test set. For the English dataset, an ensemble of Support Vector Classification and Random Forest both based on stylistic features is presented, which achieved 58% of accuracy in the provided test set.

In Section 2 we present some related work on profiling hate speech spreaders. In Section 3 we describe the two approaches proposed, including the description of the features used and the

[1]CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania
EMAIL: lugoza@i3b.upv.es (A. 1); shinojosa@multiscan.eu (A. 2)
ORCID: 0000-0001-9885-2559 (A. 1); 0000-0002-8166-8138 (A. 2)

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

CEUR Workshop Proceedings (CEUR-WS.org)

implemented machine learning models. In Section 4 we present the experimental results achieved for both languages independently. Finally, in Section 5, we present the conclusions and future work.

2. Related work

Generic text mining features are commonly used for hate speech detection [2]. These include several types of characteristics, such as those obtained from dictionaries, bag-of-words (BOW), N-grams, TF-IDF, Part-of-speech (POS) or word embeddings. There are also specific features for hate speech detection, but in some cases, they require additional user information (like gender, age or geographic localization), or they focus on specific stereotypes. Regarding the algorithms used for hate speech detection, which is typically considered as a binary classification (hate vs not-hate), the most common are Support Vector Machines, followed by Random Forest, Decision Trees and Logistic Regression [2]. More recent approaches use deep learning techniques, such as attention-based neural networks [6] or an ensemble of neural networks [7], obtaining good performance results.

In addition, the aim of this shared task is not only to detect hateful content, but profile hate speech spreaders. In this sense, common features used in the field of author profiling are stylistic features (such as frequency of punctuation marks, capital letters, word frequency), content features (such as BOW, TF-IDF or N-grams), POS tags, readability features or emotional features (emotion words and emoticons) [8,9]. Other message features such as retweets, hashtags, URLs and mentions are also considered in this area and, recently, the word and character embeddings are also applied. Regarding the algorithms used for author profiling, the traditional machine learning models are widely used, but in the last few years, deep learning approaches such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) have gained attention [10].

3. Methodology

The proposed models aim to discriminate hate speech spreaders from those who have never shared hate speech content on Twitter. They were built as an ensemble of classifiers, using two different approaches. On the one hand, stylistic features were extracted for each tweet and statistics per author were obtained from them in order to apply classic machine learning algorithms. On the other hand, a neural network with word embeddings was trained with the groups of tweets. Since no development set was provided in the shared task, it was decided to randomly split the training set into two partitions: 90% of the users for the development set and 10% for the test set, each containing 180 and 20 users respectively. These data partitions were used to evaluate the models using the official metric for this task, the accuracy, and to compare their performance on the same unseen data. The best models were then applied to the test data provided in the shared task, whose results were used to rank the performance of our system. The following sub-sections describe the two different approaches.

3.1. Word embeddings and LSTM

In this approach, an aggregation of everyone's set of tweets was first performed in order to obtain one text per subject. A preprocessing step was also applied in order to remove accents, capital letters, double spaces and stop-words. Then, the development set was divided into two partitions: 60% of the users for the training set and 40% of the users for the validation set, with 108 and 72 users respectively.

First, a tokenizer with a selected number of maximum words was adjusted to the training set, so that only the top words remained in the vocabulary, and the less used words were eliminated. The next step was to convert texts into sequences, meaning that each word of the tweet was translated into the index of that word in the vocabulary. The last step was to pad the sequences, so that they all had the same length regardless of the number of words they originally had. Specifically, a maximum sequence length of 1000 words was set, with the sequences being the collection of tweets from one subject, so that none of them would be trimmed. The training of the word embeddings with the configured dimension is performed simultaneously with the rest of the neural network parameters.

Once the above steps have been completed, the training of the neural network can be performed, setting the maximum number of words to be considered in the tokenizer and the word embedding dimensions. The neural network architecture is based on an LSTM, as shown in Figure 1, and it was trained using the categorial cross-entropy as the loss function.

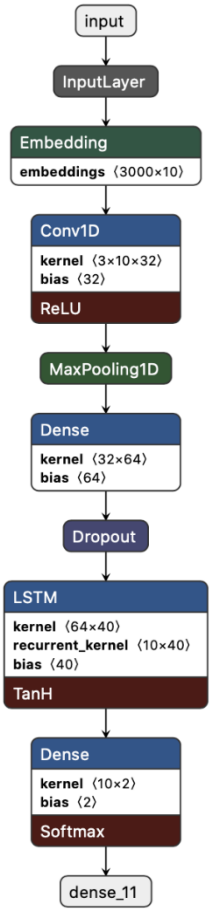

Figure 1: Neural network architecture based on word embeddings and LSTM.

3.2. Stylistic features and classical machine learning algorithms

To obtain the model that discriminates between a hater and a no hater, a set of stylistic features was calculated for each tweet independently. These characteristics have been divided into three groups: pattern-related, word-related and emoji-related features. The first group include, among others, the number of occurrences of certain patterns in the texts (such as hashtags, URLs or retweets) and the number of certain characters (such as symbols or letters). Word-related features include counts of particular words, such as nouns, verbs or adjectives. Both feature sets were calculated using regular expressions with the RegEx Python module [11] and the English model "en_core_web_sm" for de English dataset and "es_core_news_sm" for the Spanish dataset from spaCy Python library [12] for lemmatization and identification of word categories. Regarding the emojis, they were analyzed and grouped following different categories from the advertools Python library [13]. The rate between the unique emojis and the total emojis in the tweet was also included. The total set of 37 characteristics, referred to here as handcrafted features, is shown in Table 1.

Table 1
Stylistic features extracted per tweet, divided into three groups: pattern, word and emoji related features.

Pattern-related features	
• Retweets	• Symbols
• Mentioned users	• Arousal symbols [¿?¡!]
• URLs	• Capital letters
• Hashtags	• Total letters
• Laugh expressions	
Word-related features	
• Stopwords	• Verbs
• Adjectives	• Repeated words
• Nouns	• Total words
• Proper nouns	• Letters/words
Emojis-related features	
• Ratio unique / total emojis	• Face-tongue
• Face-affection	• Face-unwell
• Face-concerned	• Body-parts
• Face-costume	• Emotion
• Face-glasses	• Gender
• Face-hand	• Hand-fingers-closed
• Face-negative	• Hand-fingers-partial
• Face-neutral-skeptical	• Hand-single-finger
• Face-sleepy	• Hands
• Face-smiling	• Person-gesture

Once the stylistic features were calculated for each tweet, four statistics (mean, standard deviation, minimum and maximum) were computed for all the tweets of the same user. As result, a vector of 148 stylistic features was obtained for each author. Features were then standardized by subtracting the mean and dividing by the standard deviation for the development set, and these values were then applied to the test.

As there were only 200 different users in the dataset, a feature reduction method was applied to reduce the number of characteristics. First, the Pearson's correlation matrix was calculated, and high-correlated features ($p > 0.95$) were eliminated. Then, a filter method was implemented to avoid overfitting. It consisted of calculating the area under the ROC Curve for each characteristic and removing those with values close to 0.5, which mean that they were not relevant for the classification task. With this method 50% of the features were eliminated, remaining the features with more information. Finally, sequential backward selection was applied to determine the optimal combination of N features for classification in the range 10 to a threshold (T) of the maximum allowed characteristics, which could be 15, 20 or 30 items, respectively. This selection method iteratively computes a criterion function for a given machine learning classification algorithm using a cross-validation strategy. In each iteration, one feature is removed at a time to create n-1 subsets of features. For each of them, a machine learning model is trained, and the criterion function for cross-validation is recalculated. Based on these results, the feature associated with the best performing model is removed, since removing it yielded the best result and therefore, is the one that helps the least in the classification. This process, called feature ablation, is repeated until 10 features are left. In this work, we used accuracy as the criterion function and the stratified K fold cross-validation with five folds as cross-validation strategy.

Regarding the machine learning classification algorithms, the following were chosen, Support Vector Classification (SVC), K-Nearest Neighbors (KNN), Logistic Regression (LR), Random Forest (RF) and Decision Tree (DT). Each of these algorithms was applied sequentially in both sequential backward selection with default hyperparameters and hyperparameter tuning. In the latter step, the same cross-validation strategy was used as in the feature selection method, for the different hyperparameter

combinations shown in Table 2. Finally, the test set was transformed by keeping only the selected features and applying the standardization with the training set statistics. The machine learning model was then applied with the chosen hyperparameters and the predictions were obtained.

Table 2
Hyperparameter sets for the implemented machine learning models, with the default values used in cross-validation indicated with "default" in brackets.

Model	Hyperparameters
SVC	Kernel: Radial Basis Function (default), Sigmoid Gamma: 0.001, 0.01, 0.1, 1, 'auto', 'scale' (default) C: 1 (default), 10, 100, 1000
KNN	Number of neighbours: 1, 3, 5 (default), 7 Weights: Uniform (default), Distance Metric: Euclidean, Manhattan, Minkowski (default)
LR	Penalty: l2 C: 1 (default), 20 logarithmically scaled values between -4 and 4 Solver: liblinear, lbfgs (default)
RF	Number of trees: 100 (default), 200, 300, 400, 500 Maximum depth of the tree: 2, 4, 6, 8, 10, unlimited (default) Number of features considered for the best split: sqrt(N) (default), log2(N), where N is the total number of features
DT	Maximum depth of the tree: 2, 4, 6, 8, 10, unlimited (default) Function to measure the quality of a split: entropy, Gini impurity (default).

4. Experimental results

The following sections summarize the results obtained with the different datasets, in Spanish and English, and detail the final models chosen for each of them.

4.1. Spanish dataset

As mentioned above, two approaches were evaluated for the dataset. Firstly, the word embedding described in Section 3.1 was trained using different combinations of parameters to obtain the best configuration. Table 3 shows the accuracy results obtained in the test set by modifying the maximum number of dictionary words to be tokenized between 1000, 2000, 3000 and 4000, keeping the embedding dimensions constant.

Table 3
Neural network results varying the maximum number of vocabulary words for the Spanish dataset.

Max number words	Embedding dimension	Test-accuracy
1000	10	0.65
2000	10	0.70
3000	**10**	**0.80**
4000	10	0.70

Based on the results of Table 3, the maximum number of words was set at 3000. Then, the embedding dimensions were modified between 5, 10 and 15. The results are shown on Table 4.

Table 4
Neural network results varying the word embedding dimensions for the Spanish dataset.

Max number words	Embedding dimension	Test-accuracy
3000	5	0.55
3000	**10**	**0.80**
3000	15	0.70

The experimentation conducted showed that the best performing network configuration consisted of a maximum of 3000 words considered in the tokenizer and a 10-dimensional embedding, achieving 80% accuracy.

Despite the good results, the methodology described in Section 3.2 was used to obtain a new hater versus non-hater classifier based on stylistic features. The results are shown in Table 5, where the machine learning model and the number of features used by each model (N-features) are indicated. It also includes the following evaluation metrics: the cross-validation accuracy (CV-acc), the test accuracy (Test-acc), the true positive rate and the true negative rate in the test set (Test-TPR and Test-TNR, respectively). Only the models with the feature selection and hyperparameters that provided the best results have been included, rather than all combinations tested.

Table 5
Results of the classical machine learning models for the Spanish dataset.

Model	N-features	CV-accuracy	Test-acc	Test-TPR	Test-TNR
SVC	15	0.80 ± 0.09	0.70	0.90	0.50
KNN	12	0.80 ± 0.05	0.70	0.80	0.60
LR	**18**	**0.80 ± 0.07**	**0.80**	**0.90**	**0.70**
RF	14	0.79 ± 0.06	0.75	0.90	0.60
DT	15	0.71 ± 0.05	0.70	0.70	0.70

The highest accuracy was 80%, as in word embedding. This score was achieved with the logistic regression, both in the development test and in the test set, using the features listed in Table 6.

Table 6
Selected features in the LR model for the Spanish dataset.

Pattern-related features	
• Mean mentioned users	• Std hashtags
• Std mentioned users	• Mean arousal symbols
• Mean URLs	• Mean symbols
• Mean hashtags	• Mean capital letters
Word-related features	
• Std nouns	• Mean letters/words
• Max verbs	• Std letters/words
• Std total letters	
Emojis-related features	
• Mean emoji face-affection	• Std emoji face-concerned
• Std emoji face-affection	• Mean emoji face-smiling
• Mean emoji face-concerned	

As a last step, since both approaches achieved high accuracies, an ensemble of the two best models was built. The logistic regression and the word embedding scores were combined using the sum rule with an alpha weight associated with the score of each approach. It is shown in the equation (1), where

sc_c is the combined score, sc_{lr} is the score from the logistic regression, sc_{we} is the score from the word embedding and alpha is the weight in the range [0,1].

$$sc_c = \alpha \cdot sc_{we} + (1 - \alpha) \cdot sc_{lr} \quad (1)$$

To find the best alpha estimate, values between 0 and 1 were tested in increments of 0.05 for the development set. The alpha value that achieved the highest accuracy was 0.85, which reached 86% accuracy on the development set.

The ensemble of the regression model and word embedding was finally applied to the test set provided in the task, achieving 83% accuracy.

4.2. English dataset

As with the Spanish tweets, word embeddings were first tested to solve the classification task for the English dataset. The neural network was adapted to the dataset by modifying the maximum number words to be considered in the tokenizer between 1000, 2000, 3000 and 4000, keeping the embedding dimensions constant. The results are shown in Table 7.

Table 7
Neural network results varying the maximum number of vocabulary words for the English dataset.

Max number words	Embedding dimension	Test-accuracy
1000	10	0.45
2000	10	0.50
3000	10	0.40
4000	**10**	**0.55**

Although the results in Table 7 were not as expected, additional experiments were carried out by varying the embedding dimensions for the best of the configuration found. The results are shown in Table 8.

Table 8
Neural network results varying the word embedding dimensions for the English dataset.

Max number words	Embedding dimension	Test-accuracy
4000	5	0.45
4000	**10**	**0.55**
4000	15	0.50

The variation of the embedding dimensions also did not provide better results. Therefore, it was decided not to continue in this direction and to focus on the second approach based on classical machine learning classifiers.

Following the pipeline described in Section 3.2, the results showed in Table 9 were achieved. The table shows the machine learning model and the number of features used by each model (N-features). It also includes the following evaluation metrics: the cross-validation accuracy (CV-acc), the test accuracy (Test-acc), the true positive rate and the true negative rate in the test set (Test-TPR and Test-TNR, respectively).

Table 9
Results of the classical machine learning models for the English dataset.

Model	N-features	CV-accuracy	Test-acc	Test-TPR	Test-TNR
SVC	**13**	**0.72 ± 0.06**	**0.70**	**0.50**	**0.90**
KNN	19	0.69 ± 0.08	0.55	0.80	0.30
LR	26	0.71 ± 0.05	0.55	0.30	0.80
RF	**12**	**0.68 ± 0.08**	**0.65**	**0.60**	**0.70**
DT	29	0.67 ± 0.04	0.65	0.60	0.70

According to the results, the best models were the SVC and the RF, which obtained 70% and 65% test accuracy, respectively. The characteristics used in each model are listed in Table 10 for the SVC and Table 11 for the RF.

Table 10
Selected features in the SVC model for the English dataset.

Pattern-related features	
• Mean retweet	
Word-related features	
• Std repeated words	• Max total words
• Std total words	• Std letters / words ratio
Emojis-related features	
• Std unique emojis	• Max emoji face-hand
• Mean emoji face-affection	• Mean emoji face-sleepy
• Max emoji face-affection	• Std emoji face-unwell
• Mean emoji face-hand	• Std emoji-hands

Table 11
Selected features in the RF model for the English dataset.

Pattern-related features	
• Mean retweets	• Mean URLs
• Std mentioned users	
Word-related features	
• Std stopwords	• Max proper nouns
• Max nouns	• Max verbs
Emojis-related features	
• Mean emoji face-costume	• Std emoji face-unwell
• Mean emoji face-neutral-skeptical	• Std emoji-hands
• Std emoji face-sleepy	

Since the classical machine learning models based on stylistic features obtained better results, it was decided to create an ensemble of the two best models obtained. The final prediction was obtained by combining the SVC and RF scores using the sum rule with an alpha weight associated with the score of each model, as previously performed in the Spanish ensemble. The combination is shown in the equation (2), where sc_c is the combined score, sc_{svc} is the score from the SVC, sc_{rf} is the score from the RF and alpha is the weight in the range [0,1].

$$sc_c = \alpha \cdot sc_{rf} + (1 - \alpha) \cdot sc_{svc} \quad (2)$$

To find the best alpha estimate, values between 0 and 1 were tested in increments of 0.05 for the development set. The alpha value that achieved the highest accuracy was 0.65, which reached 56% accuracy on the development set.

The ensemble of the SVC and RF was finally applied to the test set provided in the task, achieving 58% accuracy.

5. Conclusions and future work

This paper presented the proposed ensemble models for the PAN 2021 Profiling Hate Speech Spreaders on Twitter shared task at CLEF 2021. The problem was addressed in two languages, namely Spanish and English, and two approaches were presented for each of them, whose evaluations in the task ranking are summarized in Table 12. For the Spanish dataset, an ensemble was created from a neural network with word embeddings and a logistic regression. The first one was created with all the tweets grouped by subject, whereas the second was based on statistic obtained from stylistic features computed for each user's tweet. This approach achieved 83% accuracy on the provided test set. Regarding English dataset, an ensemble of a support vector classifier and a random forest, both based on statistics of stylistic features, achieved 58% accuracy on the provided test set.

Table 12
Accuracy in test data provided in the shared task for the English and Spanish models, and the mean of both used for the task ranking.

Approach	Accuracy (%)
English ensemble	58.0
Spanish ensemble	83.0
Average	70.5

Overall, the results showed that stylistic characteristics are important features to consider when identifying hate speech spreaders, as they helped to improve the results of the word embeddings in Spanish, and they obtained better results than word embedding for the English dataset. However, the task of detecting hate speech spreaders turned out to be very difficult for the English dataset. The best accuracy result was only 70% in our test partition, which accounted for 58% in the test provided in the shared task. Word embeddings were investigated for this language, but they were not included because they showed not accurate results, contrary to Spanish. The difference in accuracy between English and Spanish may indicate that users have different hate-spreading behaviors in different cultures. Future work will include adding more features, such as TF-IDF based n-grams for both words and characters.

6. References

[1] F. Poletto, V. Basile, M. Sanguinetti, C. Bosco, V. Patti, Resources and benchmark corpora for hate speech detection: a systematic review, Language Resources and Evaluation (2020) 1–47.

[2] P. Fortuna, S. Nunes, A survey on automatic detection of hate speech in text, ACM Computing Surveys (CSUR) 51 (2018) 1–30.

[3] 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.

[4] 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.

[5] 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.
[6] H. J. Jarquín-Vásquez, M. Montes-y Gómez, L. Villaseñor-Pineda, Not all swear words are used equal: Attention over word n-grams for abusive language identification, in: Mexican Conference on Pattern Recognition, Springer, 2020, pp. 282–292.
[7] S. Zimmerman, U. Kruschwitz, C. Fox, Improving hate speech detection with deep learning ensembles, in: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 2018.
[8] F. Rangel, P. Rosso, M. Koppel, E. Stamatatos, G. Inches, Overview of the author profiling task at pan 2013, in: CLEF Conference on Multilingual and Multimodal Information Access Evaluation, CELCT, 2013, pp. 352–365.
[9] F. Rangel, P. Rosso, M. Potthast, M. Trenkmann, B. Stein, B. Verhoeven, W. Daelemans, et al., Overview of the 2nd author profiling task at pan 2014, in: CEUR Workshop Proceedings, volume 1180, CEUR Workshop Proceedings, 2014, pp. 898–927.
[10] F. Rangel, P. Rosso, M. Potthast, B. Stein, Overview of the 5th author profiling task at pan2017: Gender and language variety identification in twitter, Working notes papers of the CLEF (2017) 1613–0073.
[11] G. Van Rossum, The python library reference, release 3.8. 2, Python Software Foundation (2020) 36.
[12] M. Honnibal, I. Montani, spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing, To appear 7 (2017) 411–420.
[13] Elias Dabbas, advertools: productivity and analysis tools to scale your online marketing, 2021.URL: https://advertools.readthedocs.io/en/master/readme.html.