Detection of Hate Speech Spreaders with BERT

Notebook for PAN at CLEF 2021

David Dukić[1], Ana Sović Kržić[1]

[1]University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb, Croatia

Abstract

As social media grows, more and more users are disseminating hate speech through their posts. This often comes as a consequence of feeling a false security and anonymity in virtual environment. To stop hate speech spreaders, researchers started developing machine learning systems that automatically detect spreaders of hate speech based on the contents of their posts. This paper describes one such system which was trained on a corpus of English *Twitter* posts with a goal to predict if author of the given posts spreads hate speech or not. The features were crafted using fine-tuned BERT contextualized embeddings summed over the last 12 hidden states corresponding to the classification token, concatenated with the three binary variables called *indicators*. Binary variables were indicating whether hashtag, retweet or url were present in author's tweet posts, respectively. Feature vectors were then fed into a Logistic Regression classifier. Described model achieved 75% of accuracy score on the test set.

Keywords

BERT, fine-tuning, indicators, logistic regression

1. Introduction

In the last decade, social networks started to attract a vast number of newcomers. Users can now share everything they desire with their followers and express their opinions in an instant. While the freedom of speech is important and generally should not be restricted, not all kinds of free speech should be tolerated. This brings us to the subject of *hate speech*. Freedom of speech in virtual environment gave users a false sense of security, a sense that even hate speech could be shared without repercussions. Since the network is huge, it is almost impossible to stop hate speech spreaders using only human resources. Therefore, numerous computational methods are being developed to enable automated detection of hate spreaders on social networks.

To support and encourage creation of machine learning systems equipped to detect users that spread hate speech, PAN[1] organized a shared task called *Profiling Hate Speech Spreaders on Twitter*[2] [1]. This task was one of the shared tasks in this year's PAN at CLEF 2021 [2]. Developed models were deployed on the TIRA platform [3].

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

david.dukic@fer.hr (D. Dukić); ana.sovic.krzic@fer.hr (A. S. Kržić)

https://gitlab.com/ddaviddukic (D. Dukić)

0000-0003-2219-5294 (D. Dukić)

© 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)

[1]https://pan.webis.de/

[2]https://pan.webis.de/clef21/pan21-web/author-profiling.html

CEUR-WS.org/Vol-2936/paper-164.pdf

This paper describes how given task can be solved using a combination of fine-tuned Bidirectional Encoder Representations from Transformers (BERT) [4], binary variables (`hashtag`, `retweet` (*rt*), and `url` occurrence indicators), and Logistic Regression model. We give a brief related work overview in Section 2. Further, a data set description is presented in Section 3 followed by a detailed description of used features and models in Section 4. Implementation details can be located in Section 5 while evaluation results are displayed in Section 6 alongside an ablation study of different types of features. Lastly, we conclude the paper in Section 7.

2. Related Work

The task of automatic hate speech detection from an author profiling viewpoint is relatively new. Therefore, there were not many proposed solutions.

In the past few years PAN organized multiple author profiling shared tasks including bots and gender profiling in Twitter [5], profiling fake news spreaders on Twitter [6], gender and language variety identification in Twitter [7], and multimodal gender identification in Twitter [8]. There were also tasks closely related to the task solved in this paper. Some refer to the hate speech directed towards specific groups of people. Examples are two tasks that deal with misogyny identification [9, 10], and hate speech task against immigrants and women in Twitter [11].

The most similar task to the task at hand is hate speech detection task [12] given in Italian language. Two data sets were used. One was obtained from Facebook comments and the other from Twitter posts. Highest performing system had 82.88% of macro F1-score on the Facebook data set and 79.93% on the Twitter data set.

Some authors approached hate speech detection problem from text using deep learning [13, 14]. Others conducted detailed research of existing methods for automatic detection of hate speech [15, 16]. These surveys show that researchers also used methods from the traditional natural language processing toolbox.

3. Data Set

The data set for this task was given in English and Spanish language. We only used the English part of the corpus and did not develop model for Spanish language. English language corpus had data from 200 authors in the train set and 100 authors in the test set. Half of all the authors in both sets were labeled with 1 indicating that the author spreads hate speech, while the other half in both sets was labeled with 0 to indicate that the author does not spread hate speech. Data was collected from users that posted on the *Twitter* social network. Each of the total 300 authors had 200 unique tweet posts. This sums up to 60000 tweet posts which seems as a reasonable number. However, if we consider that the task is to predict on author-level rather than on tweet-level, and also that we cannot use test set for creating the model, we end up with only 200 training examples. Moreover, it is possible that not all posts, from an author that spreads hate speech, are genuine hate speech. Hence, this brings additional noise into the data set and aggravates prediction on author-level.

In order to tackle the small data set problem and also the noisy data set problem, we incorporated two approaches. To enlarge the training corpus, we concatenated every 20 tweet posts of each

author into a separate training example. Thus, we got 10 training examples for each author and labeled them with original label that was assigned to author whose tweets we concatenated. This produced 2000 training examples and enabled creation of the model that took 20 concatenated tweets, as a single data point, at its input. Test set was likewise restructured by concatenating every 20 tweet posts grouped by author into a new training example. However, that did not result in larger test corpus since we still predicted on author-level. More details on our prediction idea comes in Section 6. It is worth mentioning that we tried concatenating other number of tweet posts e.g. 5, 10, 25, but 20 worked the best for our final model performance. To mitigate the noise effect, we conducted an analysis of what *indicators* we could use to better discriminate between hate speech spreaders and the non hate speech spreaders. The indicators are variables that occur in most tweet posts and their presence or absence could indicate to which class author belongs.

Furthermore, some tweet posts had the following html entities: `&`, `&quot;`, `&apos;`, `>`, and `<`. They were replaced with appropriate punctuation that they represent. Hashtag, mention, and url occurrences were already substituted with *#hashtag#*, *#user#*, and *#url#* respectively. Afterwards, three distinct cleaning techniques were applied:

1. All the text in tweet post was lowercased, unicode errors were fixed, weird characters were transliterated to the closest ASCII representation, and emojis were removed
2. The same as previous cleaning method with additional removal of all the punctuation
3. The same as previous cleaning method with additional removal of the lowercased *hashtag*, *user*, *rt*, and *url* occurrences.

It is important to note that all three cleaning techniques were applied separately and tested separately. Surprisingly, the first cleaning method always gave best results independent of the model we tried when we used BERT for feature extraction. It seems that BERT works better with punctuation, as if interpunction provides extra context. Hence, we used data cleaned with first cleaning method for the models that we describe in this paper.

Indicators analysis is presented in Figure 1. Analysis was done on separate tweet posts where each tweet was labeled with *hate/no hate* depending on the corresponding author's class. We estimated the probabilities $P(Class|Indicator)$ for each of the five indicators: *hashtag*, *user*, *rt*, *url*, and *emoji* using the maximum likelihood estimation (MLE) with Laplacian smoothing:

$$P(Class = c|Indicator = i) = \frac{C(c, i) + 1}{C(i) + 2}$$

where C denotes count value i.e. frequency. Probabilities sum to 1.0 inside each indicator variable by classes. We can observe that there exists significant difference between the two classes for indicators *hashtag*, *rt*, *url*, and *emoji*. Indicator *user* did not turn out as indicating for discrimination between the classes and therefore was not used for feature creation. On the other hand, all the other indicators were utilized for crafting the features. *hashtag*, *rt*, and *url* were used as binary variables (verbatim *indicators*) while *emoji* were utilized with the help of emoji2vec embeddings [17] for some alternative models we developed.

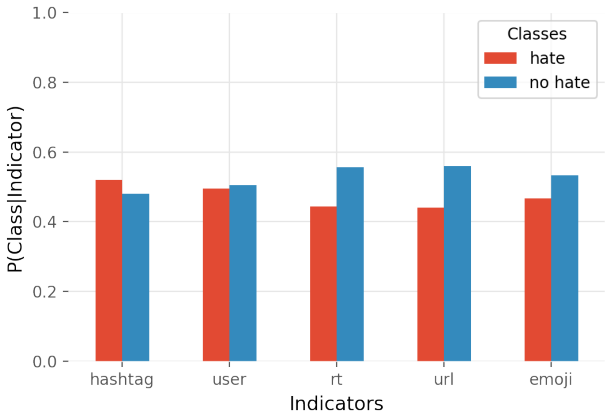

Figure 1: Estimated probability distributions for each possible indicator on the entire train set on a tweet-level. Probabilities were estimated using MLE with Laplacian smoothing.

4. Model and Feature Engineering

Models that achieved best results on the test set were essentially Logistic Regression classifiers with specific features. Generally speaking, feature vector for each training example was created using a concatenation of the following three different types of feature vectors:

1. Fine-tuned BERT embedding
2. emoji2vec embedding
3. Three binary indicator variables.

Some models used all three feature vector types, while others used only a concatenated subset.

4.1. Fine-tuned BERT Embeddings

For the first part, $\textbf{BERT}_{\textbf{BASE}}$ (uncased) model was used. To be specific, a Hugging Face[3] implementation called `BertForSequenceClassification` was used which is in fact BERT model with a linear layer on top of the model to enable classification and fine-tuning. Since the data set was enlarged by merging every 20 tweets into a new training example, the average length of each train example (after split over whitespaces was applied) came to be around 250 tokens and 75% train examples had length less than or equal to 270 tokens. Hence, maximum number of tokens for BERT model was set to 300. Tokens can be either words, characters or subwords. Training examples were tokenized using `BertTokenizer`. Special classification token $[\text{CLS}]$ was added at the beginning of the each training example, while special separation token $[\text{SEP}]$ was added at the end of each training example. Longer tweets than 300 tokens were cut off, and shorter tweets were padded with zeros until there were 300 tokens present in the training example.

[3] https://huggingface.co/transformers/

Fine-tuning details can be found in Section 5. It is important to note that a part of training set was held out as a validation set for hyperparameters optimization when we fine-tuned the model on the training set. The model with highest accuracy score on validation set was used as a final fine-tuned BERT model.

From the fine-tuned `BertForSequenceClassification`, the 768-dimensional embeddings corresponding to the [CLS] token were extracted. We experimented with different embeddings from different hidden states. Since $\mathbf{BERT_{BASE}}$ is composed of 12 stacked encoders, it is possible to extract embeddings from hidden states corresponding to each of the 12 encoders. Thus, we experimented with the following embedding combination options (numbers in brackets refer to the length of each feature vector):

1. Only embedding from the last hidden state (768)
2. Sum of embedding vectors from the 12 hidden states (768)
3. Average over the embedding vectors from the 12 hidden states (768)
4. Sum of embedding vector from the last 4 hidden states (768)
5. Average over the embedding vectors from the last 4 hidden states (768)
6. Concatenated embeddings from the last 4 hidden states (3072).

4.2. emoji2vec Embedding

Regarding the second part, 300-dimensional emoji2vec embeddings were exploited. As mentioned before, every 20 tweet posts were concatenated into a new training example. Ergo, the emojis from these posts were concatenated. Although each BERT embedding was extracted from the training example that was stripped of emojis, they were not discarded. Only the unique emojis from the concatenated string were used (if some emoji occured more than once, that information was ignored). Each of the emojis had unique emoji2vec embedding and was summed cumulatively with starting 300 dimensional embedding of zeros. Consequently, if the training example before cleaning contained no emojis, its emoji embedding was a 300-dimensional vector of zeros.

4.3. Binary Indicator Variables

Lastly, the three binary indicator variables were indicating whether the training example contained at least one *hashtag*, *rt* or *url* occurrence. Therefore, each of the three dimensions was either 1 if the occurrence was present or 0 otherwise.

The model that achieved highest accuracy score on the test set used a sum of fine-tuned BERT embeddings from the 12 hidden states concatenated with three indicator variables in combination with Logistic Regression.

5. Implementation Details

To be consistent and train all models one the same part of the train set, tune hyperparameters on the same validation set, and rate prediction success on the same test set, 50 authors and their tweet posts were held out from the train set to act as a validation set. Therefore, train set consisted of tweet posts from 150 authors, validation set of 50 authors, and test set of 100 authors. After

tuning the hyperparameters on the validation set, each model was retrained on the original train set and then the predictions were obtained on the test set.

5.1. Fine-tuning BERT

`BertForSequenceClassification` model was fine-tuned on the tweet posts of 150 authors from the train set. Number of epochs was set a priori to 200. Moreover, early stopping method was implemented that stopped the fine-tuning process if there was no improvement in terms of accuracy score on the validation set for more than 20 epochs. Because of that, the fine-tuning process never lasted longer than 50 epochs. During the fine-tuning we employed Adam optimizer, cross-entropy loss as a loss function, and a linear learning rate scheduler with zero warmup steps. Model was fine-tuned using mini batches with size equal to 5 data points in a single mini batch. Prediction on validation set was done on author-level using a threshold method that we introduce in the next section. Best learning rate was 1×10^{-7}, while best threshold was the one equal to 50%. Mentioned hyperparameters were optimized with randomized search. This fine-tuned model was trained only on 75% of the train set and achieved 72% of accuracy score on the test set.

5.2. Hyperparameter Optimization for Traditional Models

We tried three different traditional classifiers: Logistic Regression (LR), Support Vector Machine (SVM) with radial basis function (RBF) kernel, and Random Forest (RF) classifier with no bootstrap samples used for building decision trees.

Hyperparameters for these models were optimized using a grid search method with same fixed validation set for each classifier. Values of hyperparameters optimized for each of the classifiers are presented in Table 1. Hyperparameter C represents inverse of regularization strength while γ refers to the RBF kernel coefficient. For RF, max_depth defines the maximum depth of each decision tree. $None$ value specifies that nodes are being expanded while it is possible to make splits. n_estimators is the parameter that determines how many decision tree classifiers are in the random forest.

Table 1
Different traditional machine learning classifiers and their hyperparameters which were optimized using a grid search method in order to maximize the prediction success on the validation set.

Classifier	Hyperparameters	Values
LR	C	$\{2^{-6}, 2^{-5}, ..., 2^{0}, 2^{1}, 3, 4, 5, 6, 7, 2^{3}, 2^{4}, 2^{5}, 2^{6}\}$
SVM	C	$\{2^{-8}, 2^{-7}, ..., 2^{7}, 2^{8}\}$
	γ	$\{2^{-8}, 2^{-7}, ..., 2^{7}, 2^{8}\}$
RF	max_depth	$\{10, 50, 100, 200, None\}$
	n_estimators	$\{128, 512, 1024, 2048\}$

6. Evaluation

Before presenting our evaluation results and the ablation study, it is necessary to clarify the concept behind model prediction. Namely, the idea was to predict with a threshold. The prediction for single author happens in three steps:

1. 200 tweet posts for each author are split into 10 tweet posts by concatenating every 20 tweet posts into a single data point
2. For each of the created 10 data points the prediction is obtained (either 0 or 1)
3. If we set the threshold to e.g. 50% and less than 50% of predictions are zeros, then the final prediction for that author is 1, videlicet author spreads hate speech. Otherwise, the final prediction for that author is 0 which means that he does not in fact share hate speech.

We experimented with prediction thresholds set to 30%, 50%, and 70%. Surprisingly enough, the threshold of 50% always gave the best results and was tuned on the validation set.

6.1. Results on the Test Set

Table 2 displays evaluation results on the test set. Moreover, careful reader will notice that this table also shows a simple ablation study, that is *what happens when we remove some type of features from the model and how does the removal affect model performance?*. Although the specified evaluation metric for the shared task was accuracy, we present results through three additional standard evaluation metrics: precision, recall, and F1-score. First column shows which combinations of features and classifier were tried out. Regarding the features, fine-tuned BERT embeddings were tested solely or in combination with emoji2vec and indicator features. Additionally, we experimented with the 6 BERT embedding combination options. Three traditional classifiers were used: LR, SVM, and RF classifier. We can observe how, for most feature combinations and evaluation metrics, Logistic Regression outperformed the other two classifiers. The model that achieved highest accuracy and F1-score was the one that used fine-tuned BERT embeddings summed over the last 12 hidden states combined with indicator features and LR classifier. High recall of 86% indicates minimization of false negatives. This is useful from the interpretation perspective because only in 14% of cases the hate speech spreaders detection system would fail to detect them.

Table 2
Evaluation results on the test set for all tried combinations of features and traditional classifiers. Four evaluation metrics were used: precision (P), recall (R), F1-score (F1), and accuracy (Acc). **BERT** refers to BERT embedding extracted after the fine-tuning phase was finished. Superscripts denote specific embedding combination option introduced in Section 4. **emoji** represents emoji2vec features while **indicators** describe the three aforementioned binary variables. Bold scores denote the features and classifier combination that gave the proposed model with highest F1-score and accuracy on the test set.

Features + Classifier	P	R
BERT + emoji + indicators + LR	66.102^{1}, 71.186^{2}, 64.407^{3}, 67.692^{4}, 65.517^{5}, 69.355^{6}	78.000^{1}, 84.000^{2}, 75.000^{3}, 88.000^{4}, 75.000^{5}, 86.000^{6}
BERT + emoji + LR	67.213^{1}, 72.414^{2}, 63.793^{3}, 67.188^{4}, 62.500^{5}, 69.841^{6}	82.000^{1}, 84.000^{2}, 74.000^{3}, 86.000^{4}, 70.000^{5}, 88.000^{6}
BERT + indicators + LR	66.129^{1}, $\mathbf{71.667^{2}}$, 65.517^{3}, 67.742^{4}, 63.158^{5}, 69.841^{6}	82.000^{1}, $\mathbf{86.000^{2}}$, 75.000^{3}, 84.000^{4}, 72.000^{5}, 88.000^{6}
BERT + LR	65.079^{1}, 69.355^{2}, 63.158^{3}, 67.742^{4}, 63.158^{5}, 68.750^{6}	82.000^{1}, 86.000^{2}, 72.000^{3}, 84.000^{4}, 72.000^{5}, 88.000^{6}
BERT + emoji + indicators + SVM	58.182^{1}, 64.706^{2}, 65.000^{3}, 61.404^{4}, 59.259^{5}, 63.043^{6}	64.000^{1}, 66.000^{2}, 78.000^{3}, 70.000^{4}, 64.000^{5}, 58.000^{6}
BERT + emoji + SVM	58.182^{1}, 64.706^{2}, 68.333^{3}, 61.404^{4}, 58.929^{5}, 63.043^{6}	64.000^{1}, 66.000^{2}, 82.000^{3}, 70.000^{4}, 66.000^{5}, 58.000^{6}
BERT + indicators + SVM	62.745^{1}, 64.815^{2}, 66.038^{3}, 63.793^{4}, 63.158^{5}, 62.963^{6}	64.000^{1}, 70.000^{2}, 70.000^{3}, 74.000^{4}, 72.000^{5}, 68.000^{6}
BERT + SVM	62.745^{1}, 64.815^{2}, 63.462^{3}, 63.793^{4}, 62.264^{5}, 62.963^{6}	64.000^{1}, 70.000^{2}, 66.000^{3}, 74.000^{4}, 66.000^{5}, 68.000^{6}
BERT + emoji + indicators + RF	60.714^{1}, 62.005^{2}, 61.404^{3}, 60.345^{4}, 60.714^{5}, 60.000^{6}	68.000^{1}, 70.000^{2}, 70.000^{3}, 70.000^{4}, 68.000^{5}, 72.000^{6}
BERT + emoji + RF	63.793^{1}, 63.636^{2}, 60.714^{3}, 61.017^{4}, 63.158^{5}, 68.000^{6}	74.000^{1}, 70.000^{2}, 68.000^{3}, 72.000^{4}, 72.000^{5}, 68.000^{6}
BERT + indicators + RF	61.111^{1}, 63.158^{2}, 63.636^{3}, 62.264^{4}, 60.345^{5}, 65.517^{6}	66.000^{1}, 72.000^{2}, 70.000^{3}, 66.000^{4}, 70.000^{5}, 75.000^{6}
BERT + RF	63.793^{1}, 62.069^{2}, 59.615^{3}, 60.000^{4}, 60.714^{5}, 60.000^{6}	74.000^{1}, 72.000^{2}, 62.000^{3}, 66.000^{4}, 68.000^{5}, 72.000^{6}

Features + Classifier	F1	Acc
BERT + emoji + indicators + LR	71.560^{1}, 77.064^{2}, 69.725^{3}, 76.522^{4}, 70.370^{5}, 76.786^{6}	69.000^{1}, 75.000^{2}, 67.000^{3}, 73.000^{4}, 68.000^{5}, 74.000^{6}
BERT + emoji + LR	73.874^{1}, 77.778^{2}, 68.519^{3}, 75.439^{4}, 66.038^{5}, 77.876^{6}	71.000^{1}, 75.000^{2}, 66.000^{3}, 72.000^{4}, 64.000^{5}, 75.000^{6}
BERT + indicators + LR	73.214^{1}, $\mathbf{78.182^{2}}$, 70.370^{3}, 75.000^{4}, 67.290^{5}, 77.876^{6}	70.000^{1}, $\mathbf{75.000^{2}}$, 68.000^{3}, 72.000^{4}, 65.000^{5}, 75.000^{6}
BERT + LR	72.566^{1}, 76.786^{2}, 67.290^{3}, 75.000^{4}, 67.290^{5}, 77.193^{6}	69.000^{1}, 74.000^{2}, 65.000^{3}, 72.000^{4}, 65.000^{5}, 74.000^{6}
BERT + emoji + indicators + SVM	60.952^{1}, 65.347^{2}, 70.909^{3}, 65.421^{4}, 61.538^{5}, 60.417^{6}	59.000^{1}, 65.000^{2}, 68.000^{3}, 63.000^{4}, 60.000^{5}, 62.000^{6}
BERT + emoji + SVM	60.952^{1}, 65.347^{2}, 74.545^{3}, 65.421^{4}, 62.264^{5}, 60.417^{6}	59.000^{1}, 65.000^{2}, 72.000^{3}, 63.000^{4}, 60.000^{5}, 62.000^{6}
BERT + indicators + SVM	63.366^{1}, 67.308^{2}, 67.961^{3}, 68.519^{4}, 67.290^{5}, 65.385^{6}	63.000^{1}, 66.000^{2}, 67.000^{3}, 66.000^{4}, 65.000^{5}, 64.000^{6}
BERT + SVM	63.366^{1}, 67.308^{2}, 64.706^{3}, 68.519^{4}, 64.078^{5}, 65.385^{6}	63.000^{1}, 66.000^{2}, 64.000^{3}, 66.000^{4}, 63.000^{5}, 64.000^{6}
BERT + emoji + indicators + RF	64.151^{1}, 66.038^{2}, 65.421^{3}, 64.815^{4}, 64.151^{5}, 65.455^{6}	62.000^{1}, 64.000^{2}, 63.000^{3}, 62.000^{4}, 62.000^{5}, 62.000^{6}
BERT + emoji + RF	68.519^{1}, 66.667^{2}, 64.151^{3}, 66.055^{4}, 67.290^{5}, 68.000^{6}	66.000^{1}, 65.000^{2}, 62.000^{3}, 63.000^{4}, 65.000^{5}, 68.000^{6}
BERT + indicators + RF	63.462^{1}, 67.290^{2}, 66.667^{3}, 64.078^{4}, 64.815^{5}, 70.370^{6}	62.000^{1}, 65.000^{2}, 65.000^{3}, 63.000^{4}, 62.000^{5}, 68.000^{6}
BERT + RF	68.519^{1}, 66.667^{2}, 60.784^{3}, 62.857^{4}, 64.151^{5}, 65.455^{6}	66.000^{1}, 64.000^{2}, 60.000^{3}, 61.000^{4}, 62.000^{5}, 62.000^{6}

7. Conclusion

Automated hate speech detection is already an important area of research among computer scientists. This paper's contribution to the community lies in the development of a machine learning model that successfully detects English hate speech spreading authors with 75% of accuracy score. Various models were tried out. The one that worked best used a combination of fine-tuned BERT embeddings, indicator binary variables, and LR classifier.

References

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

[2] 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.

[3] 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.

[4] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: http://arxiv.org/abs/1810.04805. arXiv:1810.04805.

[5] F. Rangel, P. Rosso, Overview of the 7th Author Profiling Task at PAN 2019: Bots and Gender Profiling, in: L. Cappellato, N. Ferro, D. Losada, H. Müller (Eds.), CLEF 2019 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2019, pp. 1–36. URL: http://ceur-ws.org/Vol-2380/.

[6] F. Rangel, A. Giachanou, B. Ghanem, P. Rosso, Overview of the 8th Author Profiling Task at PAN 2020: Profiling Fake News Spreaders on Twitter, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), CLEF 2020 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2020, pp. 1–18. URL: http://ceur-ws.org/Vol-2696/.

[7] F. Rangel, P. Rosso, M. Potthast, B. Stein, Overview of the 5th Author Profiling Task at PAN 2017: Gender and Language Variety Identification in Twitter, in: L. Cappellato, N. Ferro, L. Goeuriot, T. Mandl (Eds.), CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers, 11-14 September, Dublin, Ireland, CEUR-WS.org, 2017, pp. 1–26. URL: http://ceur-ws.org/Vol-1866/.

[8] F. Rangel, M. Montes-y-Gómez, M. Potthast, B. Stein, Overview of the 6th Author Profiling Task at PAN 2018: Cross-domain Authorship Attribution and Style Change Detection, in: L. Cappellato, N. Ferro, J.-Y. Nie, L. Soulier (Eds.), CLEF 2018 Evaluation Labs and Workshop – Working Notes Papers, 10-14 September, Avignon, France, CEUR-WS.org, 2018, pp. 1–25. URL: http://ceur-ws.org/Vol-2125/.

[9] E. Fersini, D. Nozza, P. Rosso, Overview of the evalita 2018 task on automatic misogyny identification (ami), EVALITA Evaluation of NLP and Speech Tools for Italian 12 (2018) 59.

[10] E. Fersini, P. Rosso, M. Anzovino, Overview of the task on automatic misogyny identification at ibereval 2018., IberEval@ SEPLN 2150 (2018) 214–228.

[11] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. Rangel, P. Rosso, M. Sanguinetti, SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA, 2019, pp. 54–63.

doi:10.18653/v1/S19-2007.

[12] C. Bosco, D. Felice, F. Poletto, M. Sanguinetti, T. Maurizio, Overview of the evalita 2018 hate speech detection task, in: EVALITA 2018-Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, volume 2263, CEUR, 2018, pp. 1–9.

[13] S. Caetano da Silva, T. Castro Ferreira, R. M. Silva Ramos, I. Paraboni, Data driven and psycholinguistics motivated approaches to hate speech detection, Computación y Sistemas 24 (2020).

[14] 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, pp. 1–8.

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

[16] 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.

[17] B. Eisner, T. Rocktäschel, I. Augenstein, M. Bosnjak, S. Riedel, emoji2vec: Learning emoji representations from their description, CoRR abs/1609.08359 (2016). URL: http://arxiv.org/abs/1609.08359. arXiv:1609.08359.