Stop PropagHate at SemEval-2019 Tasks 5 and 6: Are abusive language classification results reproducible?

Paula Fortuna[1,2] **Juan Soler-Company**[2] **Sérgio Nunes**[1,3]

(1) INESC TEC and (3) FEUP, University of Porto
Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
paula.fortuna@fe.up.pt, sergio.nunes@fe.up.pt
(2) Pompeu Fabra University
Carrer de Roc Boronat 138, 08018 Barcelona, Spain
juan.soler@upf.edu

Abstract

This paper summarizes the participation of Stop PropagHate team at SemEval 2019. Our approach is based on replicating one of the most relevant works on the literature, using word embeddings and LSTM. After circumventing some of the problems of the original code, we found poor results when applying it to the HatEval contest (F1=0.45). We think this is due mainly to inconsistencies in the data of this contest. Finally, for the OffensEval the classifier performed well (F1=0.74), proving to have a better performance for offense detection than for hate speech.

1 Introduction

In the last few years, several evaluation tasks in the context of hate speech detection and categorization have been created. Some of these tasks include e.g., EVALITA (Bosco et al., 2018) and TRAC-1 (Kumar et al., 2018). These type of initiatives promote the development of different but comparable solutions for the same problem, within a short period of time, which is an interesting contribution for a research field. In this paper, we describe the participation of team "Stop PropagHate" in the HatEval and OffensEval tasks of SemEval 2019.

The main goal of both tasks is to improve the classification of Hate Speech and Offensive Language. Some of the works in the literature achieve a very competitive performance, e.g. Badjatiya et al. (2017) obtain an F1 score of 0.93 when using deep learning for classifying hate speech in one of the most commonly used baseline datasets (e.g. Waseem (2016)). In this context, we have a specific objective with our approach: we aim to reproduce a state-of-the-art classifier as described in the literature of this topic.

We choose to reproduce the study by Badjatiya et al. (2017), not only because of the good performance of the developed models, but also because in this work the authors published their code. Considering the amount of parameters available for definition and tuning in a machine learning classification pipeline, a precise and extensive definition of an experiment's parameters is not simple and is hardly ever provided. Thus, having the code of the experiment is the best way to understand not only which steps were conducted, but also how those steps were indeed executed. This is a highly cited paper, which can be regarded as an indicator of its relevance in the area.

In this paper, we describe our journey in the process of replication and the results achieved when applying this classifier in both shared tasks. The paper is structured as follows: Section 2 briefly reviews the literature, Section 3 presents our methodology, Section 4 describes the tasks and preliminary experiences with the data, Section 5 shows our official results in the shared tasks, and we report the conclusions of our work in Section 6.

2 Related Work

Previous research in the field of automatic detection of hate speech and offensive language can give us insight on how to approach this problem. Two surveys summarize previous research and conclude that the approaches rely frequently on Machine Learning techniques (Schmidt and Wiegand, 2017; Fortuna and Nunes, 2018). Different methods are used, such as word and character n-grams (Liu and Forss, 2014), perpetrator characteristics (Waseem and Hovy, 2016) or "othering language" (Burnap and Williams, 2016). Word embeddings (Djuric et al., 2015) are often used in this field because they can feed deep learning classification algorithms and obtain high performances. Usually, when traditional Machine

Learning classifiers are used, the most frequent algorithms are SVM (Del Vigna et al., 2017) and Random Forests (Burnap and Williams, 2014), but Deep Learning techniques are quickly gaining ground in the area (Yuan et al., 2016; Gambäck and Sikdar, 2017; Park and Fung, 2017). Different studies proved that deep learning algorithms outperform previous approaches (Mehdad and Tetreault, 2016; Park and Fung, 2017; Badjatiya et al., 2017; Del Vigna et al., 2017; Pitsilis et al., 2018; Founta et al., 2018; Zhang et al., 2018; Gambäck and Sikdar, 2017).

Other sources of solutions are previous shared tasks. For EVALITA, the best performing system achieved an F1 score of 0.83 on Facebook data and 0.80 on Twitter data. The best team tested three different classification models: one based on a linear SVM, another one based on a 1-layer BiLSTM and a 2-layer BiLSTM which exploits multi-task learning with additional data. For TRAC-1, the system that achieved the best performance with a F1 value of 0.64 used an LSTM and resorted to translation as a data augmentation strategy.

During the last 2 years, many articles have been published in this area, and one of the main focal points is to find accurate classifiers for the detection and characterization of hate speech. One main dataset is now used (Waseem, 2016), allowing performance comparison between systems. However, it is still not trivial to compare and reproduce the different approaches. Machine learning classification systems involve a long, complex set of steps and parameters and not every paper gives clear and transparent specifications. A precise specification is fundamental for replicating and improving a system.

With this idea in mind, we tried to replicate a paper with promising results as a baseline for our work. We found a paper which describes several classifiers with good performance and that also provides a GitHub repository for the code of the classifiers (as stated before, the work from Badjatiya et al. (2017). In this paper, the authors propose and use different methods. They investigate three neural network architectures applied to the problem of automatic hate speech detection: CNN, LSTM and FastText. In each one of the methods they initialize the weights with either random embeddings or GloVe embeddings. They use a dataset with messages classified as containing sexist hate speech, racist or none (Waseem, 2016). Additionally, they use 10-fold cross validation. The set of experiments achieving better performance consists in using a deep learning architecture, then taking the weights of the last layer and feeding it into a standard machine learning classifier. More particularly, embeddings learned from LSTM model were combined with gradient boosted decision trees and led to the best performance (F1 score of 0.93).

Regarding our specific approach in this shared task, the main research question of our work concerns if it is possible to replicate the results of the aforementioned paper. After trying to replicate their results, we then apply the approach to the two new datasets provided by the shared tasks.

In the next sections, we present our methodology and approach to these shared tasks.

3 Methodology

For conducting this study, we follow a methodology of 10-fold cross-validation with holdout validation (Chollet, 2017). This consists in dividing the data into two sets. One part of the data is used for cross-validation and parameter tuning with grid search on several classification parameters. The second part of the data is used for estimating the performance of this model when applied to classify new data.

In terms of pipeline, we tried to replicate the study by Badjatiya et al. (2017), and we started by downloading the version of the code[1] in December 2018. We then faced some difficulties that we list here:

- Unspecified versions of Python and of some of the used libraries.
- The authors use the fact that they provide the code as a reason not to specify the parameters in detail.
- The code contains only some of the classifiers described in the paper. The set of classifiers using xgBoost together with deep learning as features were not provided. These are the classifiers with the best performance.
- No validation data is hold out for the model to be tested after the tuning during the cross validation.

[1] `https://github.com/pinkeshbadjatiya/twitter-hatespeech`

- In the provided code, the 10-fold cross-validation procedure has a bug. With a more detailed analysis of the code we have found out that the method used for classification is train_on_batch from Keras[2], that runs a single gradient update on a single batch of data. Successive calls to this method are done through the 10 iterations of the cross-validation procedure, without instantiating a new model. This means that, during the 10 iterations, the model will successively update the gradient values without resetting it. The effect of using 10-fold cross-validation is then eliminated because only in the first iteration the testing is conducted in data never seen previously by the model. As a consequence of this problem, we can see that the successive values of F1 score found in the 10 iterations increases every time. See Table 1 for the specific F1 score values per cross-validation phase.

CV Iteration	Macro F1
1	0.76
2	0.78
3	0.8
4	0.83
5	0.87
6	0.88
7	0.89
8	0.91
9	0.92
10	0.89
Average	**0.89**

Table 1: F1 score over the different iterations of the cross-validation procedure. Experiment conducted for replicating the paper from Badjatiya et al. (2017).

In order to overcome these limitations, we provide all the information required to replicate the experiment. We use Python 3.6, Keras (Chollet et al., 2018), Gensim (Řehůřek and Sojka, 2010) and Scikit-learn (Pedregosa et al., 2011) as main libraries, and we make available our project and code[3].

The following subsections describe specific indications on how we implement each step performed by our system.

[2]Line 204 at the file `https://github.com/pinkeshbadjatiya/twitter-hatespeech/blob/master/lstm.py`

[3]`https://github.com/paulafortuna/SemEval_2019_public`

3.1 Text pre-processing

In terms of text pre-processing, we remove stop words using Gensim, and punctuation using the default string library. We transform our tweets to lower case.

3.2 Feature extraction

Regarding the features that we use in our experiment, we extract Glove Twitter word embeddings, sentiment and frequencies of words from Hatebase. The last is a set of features developed in our work. In Table 2 we present an overview of the features.

Used features	Dimensions	Abbreviation
Glove twitter word embeddings	200	glove
Sentiment Vader	4	sentiment
Hatebase	2	hatebase

Table 2: Experiment features.

3.2.1 Word embeddings

Regarding the pre-trained word embeddings, we use Twitter Glove pre-trained word embeddings with 200 dimensions. We then use the methods provided by Keras to map each token in the input to an embedding.

3.2.2 Sentiment Features

Another set of features that we use is the sentiment analysis provided by the Vader library (Hutto and Gilbert, 2014). We extract the negative ('neg'), neutral ('neu'), positive ('pos') and compound ('compound') dimensions. Each text is then represented as a 4-dimensional vector with these values.

3.2.3 Hatebase Features

Finally, we use word frequencies from the Hatebase platform (Hatebase, 2019). This platform provides different data regarding hateful words usually connected to hate speech. For each method, we count two set of words:

- Hateful words - corresponds to one or two words and are defined as "terms" in Hatebase (e.g. bitch);
- Hate topic words - corresponds to a definition of the hateful terms and are defined as "hateful meaning" (e.g. a human female). We excluded the stop words and counted for each message if there would be any reference to

words used to explain hatebase terms, so that we could approximate reference to hate related topics.

For every message, we store the frequencies of total hateful words in the text and also the frequencies of hate topic words.

3.3 Classification

Regarding the classifiers we used LSTM and xgBoost.

3.3.1 Deep Learning

For the deep learning model, we used LSTM as implemented in the code from the paper by Badjatiya et al. (2017). This contains an Embedding Layer with the weights from the word embeddings extraction procedure, an addtional LSTM layer with 50 dimensions, and dropouts at the end of both layers. We used Adam as optimizer, binary cross-entropy as loss function, 10 epochs and 128 for batch size. With this model we classify the data into binary classes and we save the last layer before the classification to extract 50 dimensions for giving it as input to the xgBoost algorithm, in a similar manner as described in the paper we are replicating. Additionally, we tested with higher dimensionality, but we find no improvement when we kept the remaining parameters.

3.3.2 xgBoost

We used the gradient boosting algorithm from the Python library xgboost (Chen and Guestrin, 2016). In terms of parameters, we used the default except for the eta and gamma. In this case we conducted a grid search combining several values of both (eta: 0, 0.3, 1; and gamma: 0.1, 1, 10). Additionally, we ran all the possible combinations of the three available sets of features: hatebase words frequencies, sentiment, and weights extracted from the LSTM model.

4 Tasks, systems and results

We conduct different experiments following the procedure described in Section 3.

4.1 Standard Dataset

In our methodology, we use a standard dataset (Waseem, 2016), so that we could compare our results with the original paper we are replicating.

4.1.1 Data

We randomly divided the data into 90% training and 10% testing datasets, having 15,214 messages for training and 1,691 messages for testing.

4.1.2 Results for Tuning and Validation

In Table 3 we present the results of the experiments with the baseline dataset. Some patterns of the results are in accordance with the original study (Badjatiya et al., 2017). Indeed, classifying the data with xgBoost after extracting 50 dimensions with the LSTM brought improvement when compared to directly classify it with LSTM. However, the results presented here are far from the 0.93 reported in the original paper. We obtained an F1 score of 0.72 using cross validation and 0.78 using the test set, when combining LSTM last layer with sentiment, hatebase, and xgBoost as classifier. One explanation for the differences between our results and the cited paper can be the fact that in this experiment we developed and classified hate speech as binary classes and for that we converted the sexism and racism, to a single hate speech class. On the other hand, the original work was conducted with the three original classes of the dataset. Another possible explanation may be the different problems found in the code, mainly the bug in the cross-validation. The results reported in the original paper may be classification models that were not tested in new data, and can be overfitted.

We can also conclude that the sentiment and hatebase features did not work well for our classification tasks either when used alone or together with the 50 dimensions extracted from the LSTM last layer to feed the xgBoost model.

4.2 HatEval (Task 5)

The proposed task (Basile et al., 2019) consists in the detection of Hate Speech targeting immigrants and women in Twitter, using texts in Spanish and English. There are two different tasks but our team participated only in the first. In Task A, the teams predict whether a tweet is hateful or not hateful as a binary classification task. This task is composed of two different subtasks, one in English and another in Spanish. The systems are evaluated and ranked using macro averaged F1 score.

4.2.1 Data

All the data provided for the competition was collected from Twitter and manually annotated via the Figure Eight crowdsourcing platform. The data is organized and especially released for the competition. More specifically, there are two datasets including tweets about hate against women and immigrants, in English and Spanish. The task dataset contains 9,000 messages for training, 1,000 messages for testing during developing phase and 3,000 messages for final testing and evaluation of the different teams.

4.2.2 Results for Tuning and Validation

Our team participated in the Task A for English and, during the model development phase, achieved the results presented in Table 4. We obtained similar results when applying the classifier to this dataset, when compared to the baseline dataset. Again, using the xgBoost to classify and the 50 dimensions of the last layer of LSTM as features, brought improvement. We achieved an F1 score of 0.75 using cross validation and 0.68 using the test set. We noticed that in the baseline the testing results improve, when compared to the cross-validation while when using the HatEval dataset those results decreased.

4.3 OffensEval (Task 6)

In OffensEval (Zampieri et al., 2019b), there are three sub-tasks and one of the main goals is to take into account the type and target of offenses. Our team participated in the Task A, about Offensive language identification. Again, Classification systems in all tasks are evaluated using the macro-averaged F1 score.

4.3.1 Data

The data used for the contest were previously presented in another work (Zampieri et al., 2019a). Participants were allowed to use external resources and other datasets for this task. Our team received 13,240 messages for training, 320 messages for testing during model development phase, and 860 messages for final testing and ranking of the different teams.

4.3.2 Results for Tuning and Validation

Our team participated in Task A and, during the model development phase, achieved the results presented in Table 5. We obtained similar results to the baseline dataset and HatEval contest. Again, using the xgBoost to classify brought improvement when compared to just use LSTM to directly classify the data (F1 score of 0.78 using cross validation and 0.80 using the test set). Additionally, we can see that the classification of Offensive discourse achieved a better performance, when compared to the classification of hate speech in previous tasks. This may indicate that offensive language is easier to identify when compared with hate speech, which is consistent with previous studies (Kumar et al., 2018).

5 Shared Task Results

In Table 6 we present the results of the team "Stop PropagHate" in the two contests. Regarding the HatEval, we conclude that the results drastically dropped in the F1 metric from 0.77 in the testing phase to 0.45 in the contest. We believe that this result is due to the sampling procedure used for building the datasets, we noticed that the training was conducted with a very balanced dataset which is an uncommon situation for hate speech automatic detection. We also find strange that the winning team only achieved 0.65 which is a result much lower than the current state of art (F1 of around 0.80). The low performances of the systems and our drop in score from the development phase indicates that there should be important differences in the evaluation dataset with respect to the training material. However, we checked the proportion of hate speech in both datasets and it is equivalent (around 42%).

For the OffensEval task, we achieved more consistent results in all the phases, with a F1 of 0.74 more similar to the 0.80 from the testing phases. The consistency in this case might indicate that the evaluation set is similar to the training material. The winner of the competition scored 0.83, so we can see that our approach is not far from that performance.

6 Conclusion

In this paper, we entered a shared task in the field of hate speech detection and characterization. Our approach was based on replicating one of the most relevant works on the state-of-the-art literature. One of our initial conclusions was that it was not possible to replicate the study and the results we aimed at. Our main difficulty was the lack of specification of the method. Additionally, the incomplete available code contained a bug that brought doubt on the validity of the reported results in the original paper. This allowed us to see the importance of sharing code in this field. This is the

Features	**Classifier**	**Number of features**	**Parameters**	**CV training F1 macro**	**testing F1 macro**
glove	LSTM	28	batch: 128, epochs: 10	0.66	-
hatebase	xgBoost	2	eta: 0, gamma: 0.1	0.44	0.45
sentiment	xgBoost	4	eta: 0, gamma: 1	0.50	0.49
hatebase, sentiment	xgBoost	6	eta: 0, gamma: 0.1	0.50	0.51
LSTM layer	xgBoost	50	eta: 0, gamma: 0.1	0.71	0.77
LSTM layer, hatebase	xgBoost	52	eta: 0, gamma: 1	0.71	0.78
LSTM layer, sentiment	xgBoost	54	eta: 0, gamma: 0.1	0.71	0.77
LSTM layer, hatebase, sentiment	xgBoost	56	eta: 0, gamma: 1	0.72	0.78

Table 3: Achieved F1 score during cross validation (CV) and testing for the baseline experiments.

Features	**Classifier**	**Number of features**	**Parameters**	**CV training F1 macro**	**testing F1 macro**
glove	LSTM	52	batch: 128, epochs: 10	0.67	-
hatebase	xgBoost	2	eta: 0, gamma: 10	0.59	0.54
sentiment	xgBoost	4	eta: 0, gamma: 1	0.61	0.53
hatebase, sentiment	xgBoost	6	eta: 0, gamma: 1	0.56	0.36
LSTM layer	xgBoost	50	eta: 0, gamma: 0.1	0.74	0.68
LSTM layer, hatebase	xgBoost	52	eta: 0, gamma: 0.1	0.75	0.68
LSTM layer, sentiment	xgBoost	54	eta: 0, gamma: 0.1	0.74	0.66
LSTM layer, hatebase, sentiment	xgBoost	56	eta: 0, gamma: 0.1	0.74	0.67

Table 4: Achieved F1 score during cross validation (CV) and testing for the HatEval experiments.

Features	**Classifier**	**Number of features**	**Parameters**	**CV training F1 macro**	**testing F1 macro**
glove	LSTM	79	batch: 128, epochs: 10	0.74	-
hatebase	xgBoost	2	eta: 0, gamma: 0.1	0.47	0.50
sentiment	xgBoost	4	eta: 0, gamma: 1	0.65	0.68
hatebase, sentiment	xgBoost	6	eta: 0, gamma: 0.1	0.41	0.47
LSTM layer	xgBoost	50	eta: 0, gamma: 1	0.78	0.80
LSTM layer, hatebase	xgBoost	52	eta: 0, gamma: 0.1	0.78	0.80
LSTM layer, sentiment vader	xgBoost	54	eta: 0, gamma: 0.1	0.78	0.80
LSTM layer, hatebase, sentiment	xgBoost	56	eta: 0, gamma: 0.1	0.78	0.80

Table 5: Achieved F1 score during cross validation (CV) and testing for the OffensEval experiments.

Metrics	**HatEval Task A**	**OffensEval Task A**
Model 1 (A)	0.48	0.82
Model 2 (A)	-	0.82
Model 3 (A)	-	0.82
Model 1 (F1)	0.45	0.74
Model 2 (F1)	-	0.74
Model 3 (F1)	-	0.74
Classification #	35	44
Number of teams	67	103
1st place (F1)	0.65	0.83
last place (F1)	0.35	0.17

Table 6: Achieved performance in the shared tasks. Model 1 corresponds to the original classifier from the replicated paper (LSTM 50d + xgBoost). Model 2 corresponds to the same as Model 1 plus adding hatebase features, and finally, Model 3 corresponds to the same as Model 1 plus adding hatebase and sentiment features.

only way to exactly replicate the reported results to then apply the same approach in other scenarios. A posteriori, we received an answer from the authors explaining that the available GitHub repository does not correspond to the final version of the project. Nevertheless, the it remained not updated at the moment of the submission of this paper. Another work could also not replicate this results (Lee et al., 2018).

After circumventing some of the aforementioned problems of the original code, we explained our specific version and used it to enter the shared task. We show that using the same classifier we found poor results when applied to the HatEval contest. Due to the results achieved on the baseline dataset and testing set before contest, we think this is due to inconsistencies between the characteristics of the training set and the final test set. Finally, for the OffensEval we believe that the classifier performed well, and with a better performance for offense detection than for hate speech.

Acknowledgments

This work was partially funded by the Google DNI grant Stop PropagHate.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. 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 (SemEval-2019)*. Association for Computational Linguistics.

Cristina Bosco, Felice DellOrletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 Hate Speech Detection Task. In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Pete Burnap and Matthew L. Williams. 2016. Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science*, 5(1):11.

Peter Burnap and Matthew L. Williams. 2014. Hate speech, machine classification and statistical modelling of information flows on Twitter: Interpretation and communication for policy decision making. In *Proceedings of Internet, Policy & Politics*, pages 1–18.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Francois Chollet. 2017. *Deep learning with python*. Manning Publications Co.

François Chollet et al. 2018. Keras: The python deep learning library. *Astrophysics Source Code Library*.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity*, pages 86–95.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, pages 29–30. ACM2.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2018. A unified deep learning architecture for abuse detection. *arXiv preprint arXiv:1802.00385*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Hatebase. 2019. Hatebase. Available in `https://www.hatebase.org/`, accessed last time in January 2019.

Clayton J. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Younghun Lee, Seunghyun Yoon, and Kyomin Jung. 2018. Comparative studies of detecting abusive language on twitter. *arXiv preprint arXiv:1808.10245*.

Shuhua Liu and Thomas Forss. 2014. Combining n-gram based similarity analysis with sentiment analysis in web content classification. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 530–537.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the SIGdial 2016 Conference: The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Ji Ho Park and Pascale Fung. 2017. One-step and Two-step Classification for Abusive Language Detection on Twitter. In *Proceedings of the First Workshop on Abusive Language Online*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. *SocialNLP 2017*, page 1.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on Twitter. In *Proceedings of the 1st Workshop on Natural Language Processing and Computational Social Science*, pages 138–142.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of NAACL-HLT*, pages 88–93.

Shuhan Yuan, Xintao Wu, and Yang Xiang. 2016. A two phase deep learning model for identifying discrimination from tweets. In *International Conference on Extending Database Technology*, pages 696–697.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on Twitter using a convolution-gru based deep neural network. In *European Semantic Web Conference*, pages 745–760. Springer.