

struggle at SemEval-2019 Task 5: An Ensemble Approach to Hate Speech Detection

Aria Nourbakhsh, Frida Vermeer, Gijs Wiltvank, Rob van der Goot

University of Groningen
Groningen, the Netherlands

{a.nourbakhsh, f.h.vermeer, g.g.wiltvank}@student.rug.nl
r.van.der.goot@rug.nl

Abstract

In this paper, we present our approach to detection of hate speech against women and immigrants in tweets for our participation in the SemEval-2019 Task 5. We trained an SVM and an RF classifier using character bi- and trigram features and a BiLSTM pre-initialized with external word embeddings. We combined the predictions of the SVM, RF and BiLSTM in two different ensemble models. The first was a majority vote of the binary values, and the second used the average of the confidence scores. For development, we got the highest accuracy (75%) by the final ensemble model with majority voting. For testing, all models scored substantially lower and the scores between the classifiers varied more. We believe that these large differences between the higher accuracies in the development phase and the lower accuracies we obtained in the testing phase have partly to do with differences between the training, development and testing data.

1 Introduction

An unwanted phenomenon that can be found across social media, is the publication of texts with hateful content. In SemEval-2019 Task 5 (Basile et al., 2019), it is defined as any communication that disparages a person or group. We focused on immigrants and women, which are two of the most targeted groups of people who are victims to this kind of discourse.

The micro-blogging service Twitter is a medium on which posts containing hateful content can be found in abundance. In order to filter out tweets with such content, machine learning and neural network techniques can be used to discriminate tweets which do contain hate speech from tweets which do not. Among the characteristics of Twitter data we can point out its noisiness and the use

of emojis and hashtags, which can be taken into account for the classification task.

In this paper, we tried to solve this problem for English, by incorporating a variety of classification algorithms including Support Vector Machine (SVM), Random Forest (RF), and Bidirectional Long Short-Term Memory (BiLSTM). For the classical machine learning classification algorithms (SVM and RF) we used character n-grams and for the BiLSTM we used word embeddings trained on a huge amount of Twitter data. By combining these three models into an ensemble learning model, we achieved our best results on the development data, so we submitted this model for this shared task.

We start the paper by discussing some earlier work done in this field. Then, we describe the dataset we used for this task. In chapter 4, we present our approach. In chapter 5, we continue with the results of our methods and the discussion. Finally, we end with a conclusion and future work in chapter 7.

2 Related work

There has been done a lot of research regarding the automatic detection of hate speech on social media, in particular Twitter. A great deal of different approaches to solve this task had been implemented in different works. The majority of these studies was done on English texts. It is clear that there is quite some overlap between each of these approaches. However, direct comparison of previous approaches is not straightforward, as different datasets were used.

Most papers tried one or multiple different classifiers, albeit with different features, but in general SVM classifiers usually achieve the best performance (Saleem et al., 2016; Davidson et al., 2017).

Some papers divided the ‘hate’ class into two

classes. For example, Watanabe et al. (2018) and Davidson et al. (2017) used the classes ‘offensive’ and ‘hate’, and Del Vigna et al. (2017) classified comments as ‘weak hate’ and ‘strong hate’.

Both the j48graft algorithm in Watanabe et al. (2018), and the SVM and LSTM in Del Vigna et al. (2017) performed better on a binary classification rather than a multiclass classification. Davidson et al. (2017) also tried different classifiers including Naive Bayes, decision trees, SVM and logistic regression. Their logistic regression and SVM classifiers achieved the best results. Waseem and Hovy (2016) tried different features for a logistic regression classifier, among which the character n-grams up to length of four in combination with the user’s gender information performed the best.

Other approaches are based on neural networks, like Zhang and Luo (2018). Their base convolutional neural network with a gap window (skipped CNN) had higher results than their SVM.

3 Data

The data provided by the organizers were collected from Twitter and manually annotated via the Figure Eight¹ crowdsourcing platform. All tweets contain a numeric ID, text of the tweet, and three labels with binary values (0 or 1). The first label indicates whether it is hate speech or not, the second if the target is a generic group of people or a specific individual and the third whether the tweeter is aggressive or not. The second and third labels can only be 1 if the first tag (hate speech or not) is 1 as well. However, we only used the first tag, since we only participated in task A, which is detecting hate speech.

The dataset for English contains 9,000 tweets for training, 1,000 for development and 3,000 for testing. Some tweets in the testing dataset were duplicates and removed from the dataset, resulting in 2,971 tweets. For each dataset, 57% of the tweets was labeled as non-hate speech and the rest as hate speech. For the testing phase, both training and development data were used for training the models.

¹<https://www.figure-eight.com/platform/>

4 Method

4.1 Preprocessing

Before training the classifiers, we did some preprocessing steps over the data.

One of the reasons we did these preprocessing steps was that many words were not available in our word embeddings for the BiLSTM. Also, we could reduce the dimensionality of the character n-gram features for the RF and SVM by the following deletions and changes:

- Lower casing text
- Removing usernames
- Removing punctuation (except ‘#’)
- Replacing each URL by ‘URL’
- Replacing each number by ‘0’

Tokenization was done based on whitespace, as tokenization on tweets is non-trivial and wrong tokenization might actually hurt performance.

Different machine learning models were evaluated and compared, among which Naive Bayes, k-nearest neighbours, RF, SVM, bagging and boosting models, and BiLSTM. Out of these models, the SVM, RF and BiLSTM proved to perform the best. The SVM and RF were implemented using Python’s scikit-learn library (Pedregosa et al., 2011) and the BiLSTM was implemented using Python’s Keras² library.

Finally, these models were combined in a majority voting ensemble model. The models are explained in more detail in the next sections, as well as the baseline models.

The architecture of our approach is shown in figure 1.

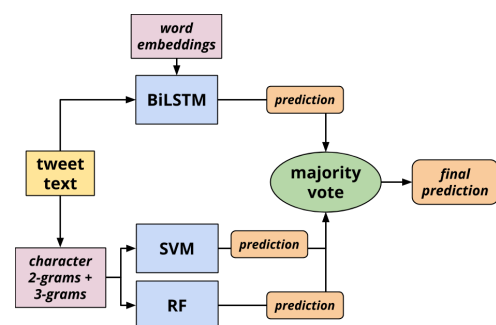


Figure 1: Architecture of our final approach.

²<http://keras.io/>

4.1.1 Baseline

We compared our results to two different baselines provided by the shared task organizers: a linear SVM with default parameters based on a tf-idf representation, and a classifier which assigns the most frequent label in the training set (MFC).

4.1.2 SVM

For the SVM, we tried different types of n-grams as features, including character and word n-grams. In the development phase, the combination of character bi- and trigrams gave the best results. These bi- and trigrams were represented as a tf-idf vector as input for the classifier.

During development, we also fine tuned the hyperparameters of the SVM classifier. The parameter values we changed for the final model were:

- $C = 100$
- kernel = 'linear'
- probability = True

4.1.3 RF

Just like the SVM classifier, we tried several different features during training for the RF, and again, the character level bi- and trigrams performed the best.

While developing, we fine tuned the parameters of the RF classifier. We experimented with different values and ratios for the number of trees and the maximum depth of the trees. Finally, we found that the following combination of parameter values led to the best performance:

- number of trees = 100
- max tree depth = 49

4.1.4 BiLSTM

In the recent years, neural network algorithms and its variants, proved to give excellent results for many NLP tasks including classification problems. Considering this, we tried a BiLSTM classifier and we used word embeddings taken from [van der Goot and van Noord \(2017\)](#) that were specifically trained on Twitter data. The embeddings were trained with Google's word2vec³ ([Mikolov et al., 2013](#)) tool with 100 dimensions. In the process of training the model, we updated the initial values. The words that were not in the word embeddings were assigned values of 0 for

³<https://code.google.com/archive/p/word2vec/>

their vector. Moreover, 6,515 words of all 26,026 unique tokens in the dataset (including the test set) were not included in the word embeddings. Among these words one can find unusual hashtags with CamelCasing (e.g. *#SendAllIllegalsHome*), use of repeated emojis and uncanonical usage of words of which some of them can be related to spelling errors.

Finally, we trained the model using a 3 layer BiLSTM model ran with 8 epochs and the following settings:

- 50 hidden units
- batch size = 300
- *adam* optimizer
- dropout rate = 0.2

4.1.5 Ensemble model

The predictions of the RF, SVM and BiLSTM were used in an ensemble model. For each final prediction, the majority vote (MV) of these predictions was taken. This means that the prediction (0 or 1) with the most votes is chosen as final prediction.

In addition to the previously described MV ensemble model, another one was made which uses the confidence scores of the SVM and RF, indicating the chance of being a 0 or 1. Finally, the average of these confidence values and the binary value of the BiLSTM was used to determine the final prediction.

5 Results and Discussion

Table 1 shows the accuracy, precision, recall and F-score for task A (hate speech detection) of the MFC, SVM (baseline), RF, SVM (character n-grams) and BiLSTM separately, as well as the final MV ensemble models and our official results. The official results differ from the other MV because after submission we did small changes in preprocessing and aggregated the development and training data for training the models.

57.3% of the data in each of the datasets (training, development and testing), was annotated as non-hate speech. Therefore, the accuracy of the MFC baseline was 57.3% as well. However, only one of the participants in this shared task managed to beat this baseline.

The accuracies and F-scores were substantially lower on the testing data than on the development data and the scores between the classifiers varied

Table 1: Evaluation measures (in percentage) of all models for development and testing on task A (hate speech detection). Precision and recall are averaged over the two classes.

Model	Development				Testing			
	Accuracy	Precision	Recall	F-score	Accuracy	Precision	Recall	F-score
MFC baseline*	57.2	32.8	57.3	41.7	57.9	28.9	50.0	36.7
SVM baseline*	72.0	71.9	72.1	71.9	49.2	59.5	54.9	45.1
RF	73.6	73.8	73.6	72.9	42.7	50.0	42.7	29.9
SVM (char. n-grams)	74.0	73.9	74.0	73.9	47.0	63.3	47.0	37.3
BiLSTM	72.2	72.6	72.2	72.3	49.7	65.1	49.7	42.5
Ensemble (MV)	75.4	75.3	75.4	75.2	45.6	64.2	45.6	33.7
Ensemble (MV conf. scores)	74.0	74.2	74.0	74.1	48.2	65.8	48.2	39.2
Submitted ensemble (MV)	-	-	-	-	46.0	58.1	52.6	39.2

*The baselines for development are re-implemented, so they could be differently compared to the organizer’s baseline.

more. This phenomenon can be explained by the fact that the testing data differed a lot from the training data. The organizers stated that before splitting the entire dataset, the data were not shuffled, which can be an explanation for these differences. These differences between the training and test data can lead to overfitting during training and parameter optimization.

During the development phase, character n-grams were the best features for the SVM, but in the testing phase it scored lower than the baseline SVM with a tf-idf representation of word unigrams. Furthermore, the MV ensemble model, combining the binary predictions of the three classifiers, got the highest scores on the development set. As a result, we submitted this MV ensemble model, without confidence scores. In contrast to the development phase, both ensemble models did not perform better on the testing data than the BiLSTM model alone. The BiLSTM performed the best (50% accuracy and 43% F-score) on the test data, but still below the baseline. Furthermore, all models had a higher precision than recall for testing, which also can be attributed to the imbalanced distribution of the data.

Finally, there were some issues with the annotation of the training and testing data. One could find instances of tweets that contain hate speech but were annotated incorrectly in our opinion. Moreover, as it was stated before, only one of the participants outperformed the MFC baseline.

6 Conclusion and future work

For the task of hate speech detection, we incorporated a variety of classifiers (SVM, RF and BiLSTM) and experimented with a range of different features and parameters. At the end we combined the predictions of these models in ensemble mod-

els. For development, the SVM, RF and BiLSTM reached similar performances and the ensemble model performed slightly better than these models individually. However, all models scored substantially lower on the test data than on the development data. As a result, we think that our approach led to overfitting on the development set without being able to generalize on the test set.

For future work, there are some ways to improve the results besides experimenting with different classification algorithms. Firstly, Twitter data is noisy and there are many uncanonical words and emojis. We tried to tackle this problem by using word embeddings that were trained on Twitter data for the BiLSTM and character n-grams for the RF and SVM. Another strategy to try is to normalize the data into a more canonical form and feed it to the classifiers. Furthermore, more experiments could be done by incorporating different features and exploiting other information that is available in the data. For example, RF and SVM classifiers trained on word embeddings and the use of certain punctuation marks, emojis and hashtags as separate features could be tried.

Finally, as it was stated in the previous section, we believe there were some issues with the dataset. Moreover, hate speech is hard to define and there is no clear agreement on the definition. This is why we can be skeptical about the annotation procedure. Therefore, we believe better datasets are necessary for the hate speech detection task.

References

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 *Pro-*

ceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019). Association for Computational Linguistics.

Thomas Davidson, Dana Warmusley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *ICWSM*, pages 512–515.

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.

Rob van der Goot and Gertjan van Noord. 2017. MoNoise: Modeling noise using a modular normalization system. *Computational Linguistics in the Netherlands Journal*, 7:129–144.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Haji Mohammad Saleem, Kelly P. Dillon, Susan Benesch, and Derek Ruths. 2016. A web of hate: Tackling hateful speech in online social spaces. *Proceedings of the 1st Workshop on Text Analytics for Cybersecurity and Online Safety*.

Zeera Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Hajime Watanabe, Mondher Bouazizi, and Tomoaki Ohtsuki. 2018. Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE Access*, 6:13825–13835.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *CoRR*, abs/1803.03662.