Comparing Different Supervised Approaches to Hate Speech Detection

Michele Corazza[†], **Stefano Menini**[‡], **Pinar Arslan**[†], **Rachele Sprugnoli**[‡]
Elena Cabrio[†], **Sara Tonelli**[‡], **Serena Villata**[†]
[†]Université Côte d'Azur, CNRS, Inria, I3S, France
[‡]Fondazione Bruno Kessler, Trento, Italy
{michele.corazza,pinar.arslan}@inria.fr
{menini,sprugnoli,satonelli}@fbk.eu
{elena.cabrio,serena.villata}@unice.fr

Abstract

English. This paper reports on the systems the InriaFBK Team submitted to the EVALITA 2018 - Shared Task on Hate Speech Detection in Italian Twitter and Facebook posts (HaSpeeDe). Our submissions were based on three separate classes of models: a model using a recurrent layer, an ngram-based neural network and a LinearSVC. For the Facebook task and the two cross-domain tasks we used the recurrent model and obtained promising results, especially in the cross-domain setting. For Twitter, we used an ngram-based neural network and the LinearSVC-based model.

Italiano. *Questo articolo descrive i modelli del team InriaFBK per lo Shared Task on Hate Speech Detection in Italian Twitter and Facebook posts (HaSpeeDe) di EVALITA 2018. Tre classi di modelli differenti sono state utilizzate: un modello che usa un livello ricorrente, una rete neurale basata su ngrammi e un modello basato su LinearSVC. Per Facebook e i due task cross-domain, si è scelto un modello ricorrente che ha ottenuto buoni risultati, specialmente per quanto riguarda i task cross-domain. Per Twitter, sono stati utilizzati la rete neurale basata su ngrammi e il modello basato su LinearSVC.*

1 Introduction

In this paper, we describe the submitted systems for each of the four subtasks organized within the HaSpeeDe evaluation exercise at EVALITA 2018 (Bosco et al., 2018): Hate speech detection on Facebook comments (Task 1: HaSpeeDe-FB), Hate speech detection on tweets (Task 2: HaSpeeDe-TW), Cross-domain task hate speech detection from Facebook to Twitter posts (Task 3.1: Cross-HaSpeeDe_FB) and Cross-domain task hate speech detection from Twitter to Facebook posts (Task 3.2: Cross-HaSpeeDe_TW). We build our models for these binary classification subtasks testing recurrent neural networks, ngram-based neural networks[1] and a LinearSVC (Support Vector Machine) approach[2]. In HaSpeeDe-TW, which has comparatively short sequences with respect to HaSpeeDe-FB, an ngram-based neural network and a LinearSVC model were used, while for HaSpeeDe-FB and the two cross-domain tasks recurrent models were used.

2 System Description

We adopt a supervised approach and, to select the best model for each task, we perform grid search over different machine learning classifiers such as Neural Networks (NN), Support Vector Machines (SVM) and Logistic Regression (LR). Both ngram-based (unigram and bigram) and recurrent models using embeddings were tested, but only the ones that were submitted for the tasks will be described. A LinearSVC model from scikit-learn (Pedregosa et al., 2011a) was also tested, and it showed good performance on the Twitter dataset. In order to perform a grid search over the parameters and models, the training set released by the task organisers was partitioned in three: 60% of it was used for training, 20% for validation and 20% for testing.[3]

2.1 Preprocessing

Since misspellings, neologisms, acronyms and jargon are common in social media interactions, it was necessary to carefully preprocess the data, in

[1] https://gitlab.com/ashmikuz/creep-cyberbullying-classifier

[2] https://github.com/0707pinar/Hate-Speech-Detection/

[3] To split the data we use the scikit-learn train_test_split function, using 42 as seed value.

order to normalize it without losing information. For this reason, we first replace URLs with the word "url" and "@" user mentions with "username" by using regular expressions.

Since hashtags often provide important semantic content, they are normalized by splitting them into the words composing them. To this end, we adapted to Italian the Ekphrasis tool (Baziotis et al., 2017), using as ngram model the Italian Google ngrams starting from year 2000. In addition to the aforementioned normalizations, for the LinearSVC model we also stemmed Italian words via the Snowball Stemmer (Bird and Loper, 2004) and we removed stopwords.

2.2 Feature Description

We used the following text-derived features:

- **Word Embeddings**: Italian fastText embeddings (Bojanowski et al., 2016)[4] employed in the recurrent models (Section 2.3);
- **Ngrams**: unigrams and bigrams, used for the ngram-based neural network and the linearSVC (Sections 2.4, 2.5);
- **Social-network specific features**: the number of hashtags and mentions, the number of exclamation and question marks, the number of emojis, the number of words that are written in uppercase.
- **Sentiment and Emotion features**: the word-level emotion and sentiment tags for Italian words extracted from the EmoLex (Mohammad and Turney, 2013; Mohammad and Turney, 2010) resource.

2.3 Recurrent Neural Network Model

In order to classify hate speech in social media interactions, we believe that recurrent neural networks are a useful tool, given their ability to remember the sequence of inputs while considering their order, differently from the feed-forward models. In the context of our classifier, this allows the model to remember the whole sequence of words in the order they appear in.

More specifically, our recurrent models, implemented using Keras (Chollet and others, 2015), combine both sequences of word embeddings and social media features. In order to achieve that, an asymmetric topology is used for the neural network: the sequences of word embeddings are fed to a recurrent layer, whose output is then concatenated with the social features. The concatenated vector is then fed to one or two feed forward fully connected layers that use the Rectified Linear Unit (ReLU) as their activation function. The output layer is a single neuron with a sigmoid activation, while binary cross-entropy is used as the loss function for the model.

Batch normalization and various kinds of dropout have been tested to reduce the variance of the models. Experimental results suggested that applying the former to the output of the recurrent layer had a negative effect on performance. For this reason, batch normalization was applied only to the output of the hidden layers. As for dropout, we tried three different mechanisms. A simple dropout layer (Srivastava et al., 2014) is applied to the output of the hidden layers, as applying dropout to the output of the recurrent layer introduces too much noise and does not improve performance. We also tested a dropout on the embeddings (Gal and Ghahramani, 2016) that effectively skips some of the word embeddings in the sequence, as dropping part of the embedding vector causes a loss of information, while dropping entire words can help reduce overfitting. In addition, a recurrent dropout (Gal and Ghahramani, 2016) was also tested. While evaluating the models, we tested both a Long Short Term Memory (LSTM) (Gers et al., 1999) and a Gated Recurrent Unit (GRU) (Cho et al., 2014) as recurrent layers. The latter is functionally very similar to an LSTM but by using less weights it can sometimes reduce the variance of the model, improving its performance.

2.4 Ngram-based Neural Networks

Ngram-based neural networks are structurally similar to the recurrent models. We first compute the unigrams and bigrams over the lemmatized social media posts. The resulting vector is then normalized by using tf-idf from scikit-learn and concatenated to the social-specific features. One or two hidden feed-forward layers are then used, and the same output layer as in the recurrent models is used. The same dropout and batch normalization techniques used in the recurrent models have been tested for the ngram-based neural networks as well. For the first submitted run of Task

[4] https://github.com/facebookresearch/fastText

2: HaSpeeDe-TW, we used unigrams and bigrams along with the required preprocessing steps based on tf-idf model.

2.5 Linear SVC System

We implemented a Linear Support Vector Classification system (i.e., LinearSVC) (Fan et al., 2008) based on bag-of-words (i.e., unigrams), using scikit-learn (Pedregosa et al., 2011b) for the first submitted run in Task 2: HaSpeeDe-TW. We chose this system as it scales well for large-scale samples, and it is efficient to solve text classification problems. To deal with imbalanced labels, we set the `class_weight` parameter as "balanced". To mitigate overfitting, penalty parameter `C` was scaled as 0.7.

3 Submitted Runs and Results

In this Section we describe the single runs submitted for each task and we present the results. The official ranking reported for each run is given in terms of macro-average F-score.

3.1 Task 1: HaSpeeDe-FB

For Task 1: HaSpeeDe-FB, two recurrent models were used. The first submitted run used a single fully connected layer of size 200 and a GRU of size 100 as the recurrent layer. Recurrent dropout was applied to the GRU with value 0.2. The second submitted run used two fully connected layers of size 500 and a GRU of size 300 as the recurrent layer. Simple dropout was applied to the output of the feed-forward layers with value 0.5. The first run ranked third and the second ranked fourth out of 18 submissions (Table 1). As shown in Table 1, both runs yield a better performance on the hate speech class.

First Run				
Category	**P**	**R**	**F1**	**Instances**
Non Hate	0.763	0.687	0.723	323
Hate	0.858	0.898	0.877	677
Macro AVG	0.810	0.793	**0.800**	1000
Second Run				
Non Hate	0.716	0.703	0.709	323
Hate	0.859	0.867	0.863	677
Macro AVG	0.788	0.785	0.786	1000

Table 1: Results on HaSpeeDe-FB

3.2 Task 2: HaSpeeDe-TW

In the first submitted run for Task 2: HaSpeeDe-TW, we used the LinearSVC-based model described in subsection 2.5. This run was ranked sixth out of 19 submissions. As our second run on the Task 2: HaSpeeDe-TW, an ngram-based neural network was used having a single fully connected hidden layer with size 200. Simple dropout was applied to the hidden layer with value 0.5. This run ranked fourth. Both runs show better performance when classifying the non hate speech class as displayed in Table 2.

First Run				
Category	**P**	**R**	**F1**	**Instances**
Non Hate	0.873	0.827	0.850	676
Hate	0.675	0.750	0.711	324
Macro AVG	0.774	0.788	0.780	1000
Second Run				
Non Hate	0.842	0.899	0.870	676
Hate	0.755	0.648	0.698	324
Macro AVG	0.799	0.774	**0.784**	1000

Table 2: Results on HaSpeeDe-TW

3.3 Task 3.1: Cross-HaSpeeDe_FB

For Task 3.1: Cross-HaSpeeDe_FB two recurrent models were used. In the first submitted run, two hidden layers of size 500 were used. An LSTM of size 200 was adopted as the recurrent layer. Embeddings dropout was applied with value 0.5 and a simple dropout was applied to the output of the feed-forward layers with value 0.5. The recurrent model for the second run had one hidden layer of size 500. A GRU of size 200 was used as the recurrent layer and no dropout was applied. The first run ranked second out of 17 submissions while the second run registered the best score in the Task 3.1: Cross-HaSpeeDe_FB. In both runs, the models showed good performance over the non hate speech class, whereas the precision on the hate speech class does not exceed 0.5 (see Table 3).

First Run				
Category	**P**	**R**	**F1**	**Instances**
Non Hate	0.810	0.675	0.736	676
Hate	0.497	0.670	0.570	324
Macro AVG	0.653	0.672	0.653	1000
Second Run				
Non Hate	0.818	0.660	0.731	676
Hate	0.494	0.694	0.580	324
Macro AVG	0.656	0.677	**0.654**	1000

Table 3: Results on Cross-HaSpeeDe_FB

3.4 Task 3.2: Cross-HaSpeeDe_TW

For Task 3.2: Cross-HaSpeeDe_TW two recurrent models were used. In the first submitted run, two

hidden layers of size 500 were used together with a GRU of size 200 as the recurrent layer. Simple dropout was applied to the output of the feed-forward layers with value 0.2, whereas the recurrent dropout has value 0.2. In the second submitted run, one hidden layer of size 200 was used adopting an LSTM of size 200 as the recurrent layer. Embeddings dropout was applied with value 0.5. The first run ranked fourth out of 17 submissions, while the other run ranked second. Table 4 shows that in both cases the system showed good performance over the hate speech class, while detecting negative instances proved difficult, in particular in terms of precision over the non hate speech class.

First Run				
Category	**P**	**R**	**F1**	**Instances**
Non Hate	0.493	0.703	0.580	323
Hate	0.822	0.656	0.730	677
Macro AVG	0.658	0.679	0.655	1000
Second Run				
Non Hate	0.537	0.653	0.589	323
Hate	0.815	0.731	0.771	677
Macro AVG	0.676	0.692	**0.680**	1000

Table 4: Results on Cross-HaSpeeDe_TW

4 Error Analysis and Discussion

Although all our runs obtained satisfactory results in each task, there is still room for improvement. In particular, we noticed that our models have problems in classifying social media messages containing the following specific phenomena: (i) dialects (e.g. "un se ponno sentì...ma come se fà...") or bad orthography (e.g. "Io no nesdune delle due.....momti pesanti"); (ii) sarcasm, "Dopo i campi rom via pure i centri sociali. L'unico problema sarà distinguere gli uni dagli altri"; (iii) references to world knowledge, typically used for an indirect attack not containing an explicit insult (e.g. "un certo Adolf sarebbe utile ancora oggi con certi soggetti"); (iv) metaphorical expressions, usually referring to ways to physically eliminate the targets of hate speech messages (e.g. "Ruspali").

As for false positives, some errors come from the misclassification of messages containing the lemmas "terrorista", "terrorismo", "immigrato" that are extremely frequent in particular in the Twitter dataset. These lemmas are associated to the hate speech class even when they appear in messages reporting the title of a news, eg. "Il Giappone senza immigrati a corto di forza lavoro".

In Task 2: HaSpeeDe-TW, when the classifier relies on sentiment and emotion features, we registered several misclassified instances containing relevant content words not covered by EmoLex. This is due to the fact that for every English word, EmoLex provides only one translated entry, thus limiting the overall coverage. For instance, "to kill" is translated in Italian with "uccidere" not considering synonyms such as "ammazzare" often used in the dataset.

Finally, we noticed some inconsistencies in the gold standard. For example, the message "Al solo vederle danno il voltastomaco!" is annotated as hate speech while, the almost equivalent, "Appena le ho viste ho vomitato" is considered a non hate speech instance while our models identify it as hate speech. Similarly, an insult like "ridicoli" is annotated as non hate speech in "CERTO CHE GLI ONOREVOLI DEL PD SI RICONOSCONO A KILOMETRI ... RIDICOLI" but as hate speech in "Ci vorrebbe anche qua Putin, invece di quei RIDICOLI...PAROLACCE PAROLACCE".

5 Conclusions

In this paper we presented an overview of the runs submitted for the four subtasks of HaSpeeDe evaluation exercise. We implemented a number of different models, comparing recurrent neural networks, ngram-based neural networks and linear SVC. While RNNs perform better in three of four tasks, classification on Twitter data achieves a better ranking using the ngram based neural network. Our system was ranked first among all the teams in one of the cross-domain task, i.e. Cross-HaSpeeDe_FB. This is probably due to the fact that considering the whole sequence of inputs with a recurrent neural networks and using a pre-learned representation by using word embeddings help the model to learn some common traits of hate speech across different social media.

Acknowledgments

Part of this work was funded by the CREEP project (`http://creep-project.eu/`), a Digital Wellbeing Activity supported by EIT Digital in 2018. This research was also supported by the HATEMETER project (`http://hatemeter.eu/`) within the EU Rights, Equality and Citizenship Programme 2014-2020.

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August. Association for Computational Linguistics.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 HaSpeeDe Hate Speech Detection (HaSpeeDe) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA18)*, Turin, Italy, December. CEUR.org.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

François Chollet et al. 2015. Keras. `https://keras.io`.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM.

Saif M Mohammad and Peter D Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34. Association for Computational Linguistics.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

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. 2011a. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.