InriaFBK at Germeval 2018: Identifying Offensive Tweets Using Recurrent Neural Networks

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

In this paper, we describe two systems for predicting message-level offensive language in German tweets: one discriminates between offensive and not offensive messages, and the second performs a fine-grained classification by recognizing also classes of offense. Both systems are based on the same approach, which builds upon Recurrent Neural Networks used with the following features: word embeddings, emoji embeddings and social-network specific features. The model is able to combine word-level information and tweet-level information in order to perform the classification tasks.

1 Introduction

The widespread use of social media platforms such as Twitter and Facebook yields a huge number of interactions on the Web. Unfortunately, social media messages are often written to attack specific groups of users based on their religion, ethnicity or social status, and they can be particularly threatening to vulnerable users such as teenagers.

Due to the massive rise of hateful, abusive, offensive messages, social media platforms such as Twitter and Facebook have been searching for solutions to tackle hate speech (Lomas, 2016). As a consequence, the amount of research targeting the detection of hate speech, abusive language and cyberbullying also shows an increase (Waseem et al., 2017). Various (predominantly supervised) classifiers have been used for hate speech detection (Greevy and Smeaton, 2004; Warner and Hirschberg, 2012). In recent research, deep learning approaches with Recurrent Neural Networks were also used (Mehdad and Tetreault, 2016).

In this paper, we build our model on Recurrent Neural Networks (RNN) for both binary and fine-grained classification tasks. We combine recurrent layers with feed-forward layers so that we can combine word embeddings with other features, i.e., emoji embeddings and social network-specific features. We also apply some specific dropout techniques not only to recurrent layers but also to feed-forward layers, aimed at reducing the variance of our model.

2 Data

Within the Germeval evaluation, two different tasks were proposed: one for the detection of offensive messages, and the other for a fine-grained classification in four classes, namely *Profanity*, *Insult*, *Abuse* and *Other*. For both Task I (binary classification) and Task II (fine-grained classification), we used the data provided by the Germeval organizers. It consists of 5,009 German tweets from Twitter with a manual annotation at the message level.

Task I - Binary classification: The two labels are 'offensive' and 'other'. The latter was reserved for tweets which were not offensive. The binary classification task involved 1,688 messages with 'offensive' label and 3,321 messages with the 'other' label.

Task II - Fine-grained classification: The four classes annotated are 'profanity', 'insult', 'abuse' and 'other'. In the corpus, there are 595 messages for 'insult', 71 for 'profanity', 1,022 for 'abuse', and 3,321 messages for 'other'.

3 System Description

Given that the amount of training data is enough to adopt a supervised approach, we select the best classifier by using a grid-search approach over different machine learning models, such as Neural Networks (NN), Support Vector Machines (SVM) and Logistic Regression (LR). Both ngram-based models and recurrent models using embeddings

Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)
Vienna, Austria – September 21, 2018

were tested, but we will describe in detail only the model performing best on our validation set, using Recurrent Neural Networks.

In order to evaluate our system, the training set was split in three parts: 60% was used for training, while the remaining 40% was split in half to create a validation and a test set. This was achieved by using the `train_test_split` function of scikit-learn. In order to be able to compare the results of the experiments, a seed value of 42 was used as input to that function.

3.1 Pre-processing

One of the challenges that arise from working on social media interactions derives from the specific language used in posts, including misspelled words, neologisms and jargon. As a consequence, most standard models built for news are unsuitable for tweets. In order to extract as much information as possible from such interactions and use them for classification, some pre-processing steps are necessary. The simplest ones involve the normalization of URLs and '@' mentions, which we performed using simple regular expressions that replace URLs with the string 'URL' and mentions with the string 'username'.

Another aspect that is typical of social media interactions is the presence of hashtags, that sometimes convey a semantic content in a concise way. It is therefore important to normalize them by splitting them in a sequence of meaningful terms, as some of them are composed of multiple words that would not be recognized as such if they are not tokenized correctly. To this purpose, we propose an extension of the tokenizer presented by Baziotis et al. (2017), which is tailored to social media messages but is available only for English.

Once a hashtag composed by two or more concatenated words (e.g., #StandwithBoris) is found in a post, the algorithm uses n-grams (both unigrams and bigrams) to obtain word probabilities and identify the most likely way to split the input string (e.g., 'Stand with Boris'). In order to adapt it to German, we use as n-gram model all German Google n-grams starting from year 2000. We avoid older n-grams considering them less representative of the current language.

3.2 Feature description

In order to identify offensive language, a small set of features was used, that are derived from the textual information included in the tweets. The features we used are the following:

- **Word Embeddings**: we use German fastText word embeddings (Bojanowski et al., 2016)[1], pre-trained on Wikipedia.

- **Emoji Embeddings**: the German fastText embeddings were extracted from Wikipedia, where there are basically no emojis. However, emojis are very frequent in social media data, and are often used to conveyed emotions and feelings associated with offenses or ironic messages. Therefore, we needed to add this information for classification, which we perform in two steps: first, we download the embeddings trained on 10 millions English tweets containing also a representation for emojis (Barbieri et al., 2016). We use this corpus because no equivalent dataset of this size is available for German, Then, we follow the approach by Smith et al. (2017) to align the English vector space containing the emojis with the German one, using a bilingual dictionary.

- **Social-network specific features**: a collection of features that capture some aspects of social media interactions is considered. They include 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.

3.3 The Recurrent Neural Network model

In order to tackle the complexity of offensive messages in social media, we believe that recurrent neural networks are a useful tool, as they have an advantage over the classic feed-forward models: they consider the data they process in order and they remember the whole sequence of inputs. In the context of Natural Language Processing, this allows the network to remember the whole sequence of words or characters provided as input in the order in which they appear.

The models were implemented using Keras (Chollet and others, 2015), a Python library for deep-learning that makes it easy to prototype different models without re-writing the core layers that are needed. Our models combine both recurrent layers and feed-forward layers, to combine word

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

embeddings (that have a variable length and encode each tweet as a sequence) and tweet-level features such as the number of emojis. To achieve that, we adopt an asymmetric topology for the model. First, a recurrent layer is used to process the word embedding sequences. The output that the recurrent layer produces at the last timestep is then concatenated with the other features and passed through a variable number of hidden feed-forward layers that use the Rectified Linear Unit (ReLU) as their activation function.

The output layer of the network varies depending on the task. We use a sigmoid-activated single neuron for the coarse classification task, while we use 4 neurons with a softmax activation function for the fine-grained classification. For binary classification, the binary cross-entropy function from Keras is used, while categorical cross-entropy is used for the multiclass version of the model.

In order to reduce the variance of the model, different techniques were tested, in particular we have used various dropout techniques and batch normalization. Specifically, three different dropout methods have been used: a simple dropout layer (Srivastava et al., 2014) is applied to the output of the feed-forward layers. Furthermore, to increase the noise of the input for the recurrent layer, a dropout on the embeddings input is applied (Gal and Ghahramani, 2016). This technique operates by dropping a single embedding at a time, instead of dropping only part of each embedding. This is motivated by the fact that for the embeddings input, the whole vector is important and therefore dropping part of each embedding would cause some loss of information. In addition to these techniques, dropout is also applied to the recurrent layer of the model, using the approach proposed by Gal and Ghahramani (2016).

As for batch normalization (Ioffe and Szegedy, 2015), from experimental results it was clear that applying it directly to the output of a recurrent layer introduces too much noise and results in worse performance. We therefore apply batch normalization only to the output of the hidden feed-forward layers.

While evaluating the model's hyperparameters, both a Long Short Term Memory (LSTM) (Gers et al., 1999) layer and a Gated Recurrent Unit (GRU) (Cho et al., 2014) layer were tested. The latter is very similar in nature to an LSTM, but it has the advantage of using a smaller number of weights, reducing overfitting on the training data. Details on which configuration was chosen for each task and the submitted runs are reported below.

3.4 System description - Task 1

For the coarse classification task, the aforementioned architecture was used. We performed a grid search to select the best performing parameters on the validation set. We selected among two different sets of models, one with two feed-forward layers and one with one feed-forward layer.

The first submitted run (`InriaFBK_coarse_1`) is the best performing one among the models with two hidden feed-forward layers. We used no dropout on the embeddings and no dropout on the feed-forward layers, while the recurrent dropout is set to 0.2. No batch normalization was applied, and a GRU layer was used as the recurrent layer. The two feed-forward layers have 500 neurons each, while the recurrent layer has size 300.

The second submitted run (`InriaFBK_coarse_2`) is the best performing one among the models with one hidden feed-forward layer. We used no dropout on the embeddings, a dropout layer on the output of the hidden layer (dropout value of 0.5), thr recurrent dropout was set to 0.2. Batch normalization was used. The recurrent layer is a GRU of size 300, while the hidden layer has size 200.

The third submitted run (`InriaFBK_coarse_3`) is derived from the parameters of the first run, but we reduced the size of both the hidden and the feed-forward layers. The dropouts, batch normalization, recurrent layer type are therefore the same as in the first run. while the two hidden feed-forward layers have size 200. The recurrent layer has size 100.

3.5 System description - Task 2

For the fine-grained classification task, an approach similar to the first task was used. Grid search was performed over two different sets of models, with one and two feed-forward layers, respectively.

The first submitted run (`InriaFBK_fine_1`) is the best performing one among the models with two hidden feed-forward layers. It uses no batch normalization and no recurrent dropout. Dropout was applied on the output of the feed-forward layer, with value 0.2. The size of the hidden layer is 500, and the recurrent layer has size 300. We use a GRU as the recurrent layer.

The second submitted run (InriaFBK_fine_2) is the best performing one among the models with one hidden feed-forward layer and batch normalization. It uses recurrent dropout with value 0.2. Dropout was applied on the output of the feed-forward layers with value 0.5. The size of the hidden layer is 500, and the recurrent layer has size 300. We use a GRU as the recurrent layer.

The third submitted run (InriaFBK_fine_3) is the best performing one among the models with one hidden feed-forward layer but no batch normalization. It uses recurrent dropout with value 0.2. Dropout was applied on the output of the feed-forward layer, with value 0.5. The size of the hidden layer is 500, the recurrent layer has size 300. We use a GRU as the recurrent layer.

The system developed for the two tasks is available at `https://gitlab.com/ashmikuz/creep-cyberbullying-classifier`.

4 Evaluation

We report in this Section the preliminary results on the test set, using the splits described in Section 3.

4.1 Preliminary Results - Task 1

Results on Task 1 show that there are only slight differences among the three runs submitted for the task. The configuration *coarse_1* achieves the best performance on the 'Offensive' class, while on the 'Other' class *coarse_2* it yields a slightly better improvement. Overall, it seems that *coarse_1* is less sensitive to the imbalance of the two classes, since it can classify better the offensive tweets with less training instances.

Category	P	R	F1	Support
Offensive	0.65	0.72	**0.68**	333
Other	0.85	0.80	0.83	669
Macro AVG	0.75	0.76	**0.75**	1002
Micro AVG	0.78	0.78	**0.78**	1002

Table 1: Results for InriaFBK_coarse_1

Category	P	R	F1	Support
Offensive	0.70	0.62	0.65	333
Other	0.82	0.87	**0.84**	669
Macro AVG	0.76	0.74	**0.75**	1002
Micro AVG	0.78	0.78	**0.78**	1002

Table 2: Results for InriaFBK_coarse_2

Category	P	R	F1	Support
Offensive	0.67	0.64	0.65	333
Other	0.83	0.84	0.83	669
Macro AVG	0.75	0.74	0.74	1002
Micro AVG	0.77	0.77	0.77	1002

Table 3: Results for InriaFBK_coarse_3

4.2 Preliminary Results - Task 2

Results on Task 2 show that the configuration with one hidden feed-forward layer (*fine_2*) is generally best performing on all categories apart from 'Profanity', which is outperformed by the model with two hidden feed-forward layers (*fine_1*) . The reason behind this difference will be further investigated in the future with additional experiments.

Category	P	R	F1	Support
Abuse	0.51	0.51	0.51	210
Insult	0.37	0.44	**0.40**	111
Profanity	0.43	0.25	**0.32**	12
Other	0.84	0.82	0.83	669
Macro AVG	0.54	0.51	**0.52**	1002
Micro AVG	0.71	0.71	0.71	1002

Table 4: Results for InriaFBK_fine_1

Category	P	R	F1	Support
Abuse	0.59	0.51	**0.55**	210
Insult	0.37	0.44	**0.40**	111
Profanity	0.50	0.17	0.25	12
Other	0.83	0.85	**0.84**	669
Macro AVG	0.57	0.49	0.51	1002
Micro AVG	0.72	0.72	0.72	1002

Table 5: Results for InriaFBK_fine_2

Category	P	R	F1	Support
Abuse	0.60	0.50	**0.55**	210
Insult	0.38	0.41	**0.40**	111
Profanity	0.50	0.17	0.25	12
Other	0.82	0.86	**0.84**	669
Macro AVG	0.58	0.49	0.51	1002
Micro AVG	0.73	0.73	**0.73**	1002

Table 6: Results for InriaFBK_fine_3

The differences between *fine_2* and *fine_3* are minimal, with all F1 values being identical between the two sets of classes (apart from the Micro AVG).

Please note that the three runs submitted to the shared evaluation for each Task were obtained by re-training the models with the configurations described above, keeping the same validation set (20%) and merging the training and the test introduced in Section 3 to increase the amount of training data.

5 Conclusions

In this paper, we have described the system submitted to Germeval 2018 by a team composed of researchers from INRIA Sophia Antipolis and Fondazione Bruno Kessler in Trento. We adopt an approach based on Recurrent Neural Networks that does not require any external lexicon or semantic resource, and that is based on features extracted directly from text. It also makes use of the fastText embeddings and emoji embeddings extracted from a large English corpus and automatically aligned to the German ones. We chose this approach because we want to build a framework able to work on multiple languages, given a language-specific training set. Indeed, we plan to participate with the same system to another task for hate speech detection in Italian.

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

Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. 2016. What does this Emoji Mean? A Vector Space Skip-Gram Model for Twitter Emojis. In *LREC*.

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.

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

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`.

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.

Edel Greevy and Alan F Smeaton. 2004. Classifying racist texts using a support vector machine. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 468–469. ACM.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Natasha Lomas. 2016. Facebook, Google, Twitter commit to hate speech action in Germany.

Yashar Mehdad and Joel Tetreault. 2016. Do Characters Abuse More Than Words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Samuel L. Smith, David H.P. Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

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.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the World Wide Web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

Zeerak Waseem, Wendy Hui Kyong Chung, Dirk Hovy, and Joel Tetreault. 2017. Proceedings of the First Workshop on Abusive Language Online. In *Proceedings of the First Workshop on Abusive Language Online*. Association for Computational Linguistics.