Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)

HAU at the GermEval 2019 Shared Task on the Identification of Offensive Language in Microposts: System Description of Word List, Statistical and Hybrid Approaches

Johannes Schäfer[1], **Tom De Smedt**[2], and **Sylvia Jaki**[3]

[1] Institute for Information Science and Natural Language Processing, University of Hildesheim
[2] Computational Linguistics Research Group, University of Antwerp
[3] Department of Translation and Specialized Communication, University of Hildesheim

johannes.schaefer@uni-hildesheim.de, tom.desmedt@uantwerpen.be, jakisy@uni-hildesheim.de

Abstract

This paper presents our contribution (HAU) for the three subtasks of GermEval 2019 Task 2. To detect offensive microposts, we have experimented with different approaches and a combination thereof, namely, a Convolutional Neural Network (CNN), a Random Forest, and a lexicon-based approach. In this paper, we report our methodology, demonstrate how it includes insights from GermEval 2018, and compare the different approaches for the different subtasks in view of future directions in the detection of offensive language and hate speech online.

1 Introduction

This year's *Terrorism Situation and Trend Report* from Europol (2019) again notes the increase of far-right incidents in Germany, including attacks against immigrants and mosques, the rioting in Chemnitz, the fatal stabbing of a pro-migrant politician, and in particular mentions the instigators' affinity with weapons. It is becoming difficult to ignore the role of unmoderated social media platforms like *Gab*, *4chan* and *8chan* in radicalization processes and the proliferation of hatred. This in itself justifies organizing a second GermEval task on offensive language (or other shared tasks such as OffensEval 2019 (Zampieri et al., 2019) and HatEval 2019 (Basile et al., 2019)), encouraging the scientific community to investigate hateful online discourse, for example to help develop better content moderation tools, or early warning systems for illegal content.

HAU1 is our best submission: a CNN with extensive feature engineering. HAU2 is a lexicon of offensive words and manually-annotated word scores; our main interest here is to observe how well it does compared to other techniques. HAU3 is a Random Forest trained on character trigrams and word unigrams, which we used as a baseline for the other two. The choice of the techniques is based on last year's GermEval, where CNNs and Random Forests were among the best-performing systems (see Wiegand et al., 2018), and where we also demonstrated that competitive results can be achieved with lexicons.

In the following sections, we present our three approaches, with the main emphasis on the CNN in Section 2. The Random Forest is discussed in Section 3, and our new "POW" lexicon in Section 4, followed by a discussion of the systems' performance in the three subtasks in Section 5.

2 HAU1: CNN

In this section, we describe our neural network model developed for the identification of offensive language in microposts. The overall network structure is based on the approach described in detail in Schäfer (2018) where text, metadata and linguistic features are handled by parallel sub-networks. We use a variant that yielded the best results in previous experiments, and which considers only text-based and metadata features. We elaborate on a few choices for the structure and hyperparametrization in Section 2.3, based on early experiments on the given GermEval 2019 training dataset.

2.1 Model Description

In the following, we list the differences of our model in comparison to the approach described in Schäfer (2018), firstly, the changes to the model

architecture, and, subsequently, the changes to the training procedure.

Model architecture: The original system used a recurrent neural network (RNN) with long short-term memory (LSTM) units for encoding the text-based features. However, during experiments we observed an improved performance when replacing this encoder by a CNN. Our new CNN model is based on the architecture given in Schäfer and Burtenshaw (2019), where multiple sequences of convolution, dropout and max pooling are combined in parallel in a text encoder. This encoder operates on a word embedding of the input text sequence (both in GermEval 2018 and 2019). The parallel structure was specifically designed for the detection of offensive language. Each branch in the CNN is trained to identify those n-grams (with n from 1 to 6) from the text which are significant for the classification task. Based on that, different types of offensive expressions can be captured and also mentions of targets (proper names) or other multi-word features.

The sub-network with metadata features has remained unchanged in comparison to Schäfer (2018). It considers a variety of numerical features calculated on the input text, by using rule-based formulas and partially counting matches in pre-defined word lists. The list of 27 basic metadata features[1] is given in the referenced paper.

Model training procedure: We briefly mention two commonly used methods for training a neural network, which were not considered in Schäfer (2018) but included in the system at hand. First, the label ratio in the given dataset is imbalanced: about 1:2 in the binary classification task, or 1 offensive comment per 2 non-offensive. When trying to optimize all labels equally (which is considered by the macro-average F1-score evaluation metric of the shared task), it is beneficial to use class weights during the training process. By doing so, we can, for example, boost the importance of the more infrequent label `OFFENSE` in the binary classification task. We give our exact class weights used for the submitted system runs for the different tasks computed on the entire training dataset in Section 2.3.

Second, as we train the model by iterating over the training data multiple times, a stopping point has to be determined, early enough to avoid overfitting to the given training data. To automatically determine a suitable number of training epochs, we use early stopping as follows. About 5% of the training data is retained as a validation set. Then, after each epoch over the training data, the performance of the resulting model is evaluated on this validation set. If the performance did not improve in the last few epochs, the weights of the model that resulted in the best score on the validation set are loaded, and this model is then finally returned. Since we optimize on fewer training data instances during cross-validation - in comparison to the entire training dataset for the final prediction of the shared task test dataset - it is crucial to automate this process by adjusting the number of training epochs depending on a validation set performance. In our experiments, early stopping was usually executed after 7 to 15 epochs on the training dataset.

[1] Text length, number of proper names, hashtags, etc.

2.2 Model Features

In this section we discuss the integration of features from our new POW lexicon (Section 4) into the neural network. We expected the lexicon to provide additional guiding features for offensive language detection, and we experimented with different ways of considering those features as input for the neural network. Computing additional features on the tweet as well as on the word level proved to be beneficial. Furthermore, in our overall network architecture, features based on the lexicon led to performance improvements when including them directly in the text-based sub-network, as well as in the parallel metadata sub-network. We implemented this as follows:

Tweet-level features: For each tweet, we check for each word from the lexicon if it is contained in the tweet (untokenized string). If we find a match, we add 10 feature values to the tweet: one for the word's manually annotated intensity score (0-4) in the lexicon, and nine more for each fine-grained category in the lexicon (0 or 1). If multiple words from the lexicon are matched in a tweet, the values are summed up. For example, a tweet with features [6, 0, 1, 2, 0, 0, 0, 0, 0, 0] might contain 2 words from the lexicon, where each of these has an intensity score of 3, both of them have the `RIDICULE` label (forth position in the vector), and one has the `DEHUMANIZATION` label (third position in the vector). Note that it could also be beneficial to only use the score of the matched word with the highest values (instead of the sum); a configuration which we did not test. Since the task

of offensive language detection is formulated as a classification task where we need to identify text that "contains" offensive language, one highly offensive word should be enough to lead to a high prediction score. In our model, we include these 10 features for each tweet as additional metadata features in the parallel sub-network, which then considers a total of 37 features.

Word-level features: In a similar way to the tweet-level features we calculate additional features on word-level. We also apply the abovementioned procedure on words in a loop on the tokenized tweet, thus resulting in 10 additional features for each word in a tweet. We add these features into the text encoder by stacking them directly on top of the word embeddings. Subsequently, these augmented word embeddings (which are hybrid features containing the distributional semantics of words as well as direct lexicon-based features) are fed into the CNN. The special property of this method is that any generic pre-trained word embeddings can be used, while a task-specific augmentation is included.

Integration into the CNN: Figure 1 displays the architecture of the overall network. The additional tweet-level features are included in the input layer for metadata features on the top right (see `Input_Meta` with dimension 37 for the in total 27 + 10 features). The additional word-level features are given as input in the layer `Input_POW-meta` next to the text input layer (see on top in the middle; last dimension 10 for the 10 features for each word). The 10 dimensions of the additional word-level features are added to the (here: 200) dimensions of the word embedding layer, resulting in augmented word embeddings (see the output of the `Concatenate` layer, last dimension 210).

2.3 Hyperparametrization & Test Results

In this section we give explanations for our choice of parameters and the structure of our final neural network architecture, which was optimized in a 3-fold cross-validation on the GermEval 2019 training dataset. The given numbers are averages over the 3 folds of macro-average F1-scores for offensive language detection (i.e., **Subtask 1**, binary classification). The performance of the basic CNN model without any additional metadata features in the configuration of Schäfer (2018) was 71.98%. Including the basic 27 metadata features in an additional sub-network then resulted in 72.84%. To measure the effect of using our POW lexicon features, we evaluated different configurations:

- CCN (augmented embeddings) + meta (27 basic features): 73.56%,
- CCN (basic embeddings) + meta (27 basic features + 10 POW): 75.17%,
- CCN (augmented embeddings) + meta (27 basic features + 10 POW): 75.46%.

We can see that both additions seem to be beneficial when activating individual feature categories, while the best results are achieved when we include the lexicon features both on word level as well as on tweet level. A clear improvement can be observed when adding the features on the tweet level, perhaps because the separate sub-network has the capability of learning to identify different types of offensive language than the text encoder sub-network, which could possibly be captured by the lexicon.

We mostly followed the text preprocessing steps described in Schäfer (2018), but we had to make changes to the normalization technique as we ran into a considerable amount of out-of-vocabulary words with the GermEval 2019 dataset. It is important when using word embeddings to have a low number of unknown words, as all such words get assigned a dummy embedding and are basically ignored. We defined our vocabulary as words from the training dataset with a frequency of occurrence greater than 1. This method can be problematic when applied to social media data, due to the high amount of spelling errors, variants or neologisms. Thus, an extensive normalization technique is required.

We attempted to use compound splitting, rule-based/statistical lemmatization, and using the further training data to enrich the vocabulary. However, the best performance was achieved by using a fallback based on *Levenshtein*-similarity for unknown words. When a word is not in the vocabulary, we select the word with the most similar string (i.e., lowest *Levenshtein* distance) instead. This approach is questionable, and in a real-world application we would not suggest it, since it basically guesses unknown words. Nevertheless, it seemed to work here, as it leads to a 1% performance improvement in the given dataset. A proper solution would be to use more training data, which did not work in our evaluation, probably because the training and test sets are too similar. For a

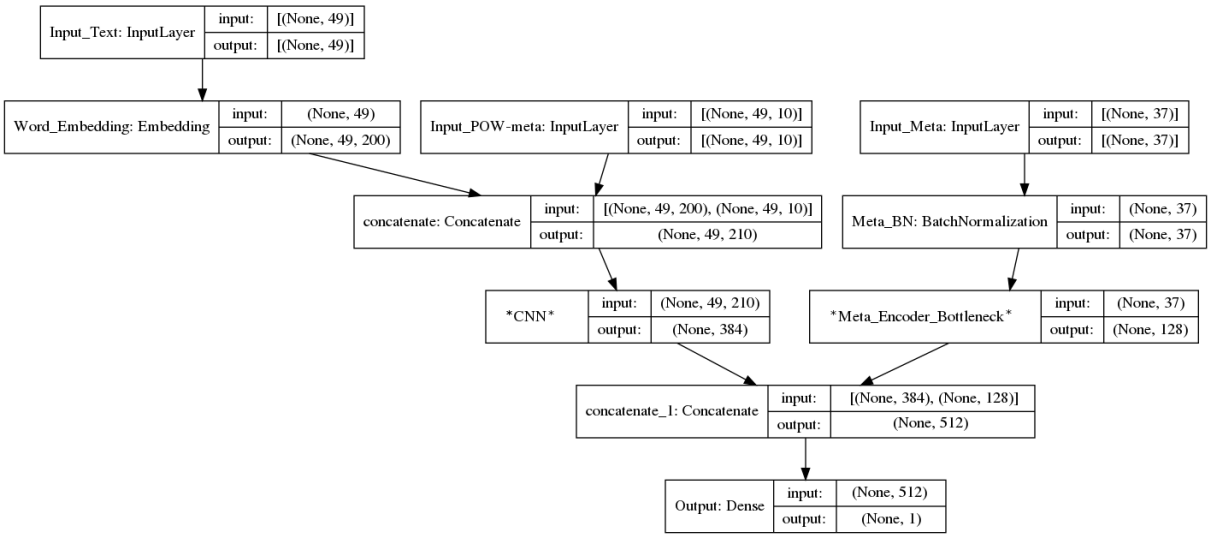

Figure 1: Neural network model structure for binary classification. The blocks correspond to layers, and those marked with * consist of sub-networks (for the exact structures refer to Schäfer and Burtenshaw (2019) for the CNN and to Schäfer (2018) for the `Meta_Encoder`). Each block contains the name and type of the layer with the dimensions of its input and output. The first value of the dimensions corresponds to the batch size and is given as None in the figure (64 in our experiments).

real-world application it might also be advisable to evaluate on multiple datasets from different sources when developing a system.

For hyperparametrization in general, we found that setting the number of filters of each convolutional layer to 8 leads to the best results. In our final model, we mainly optimized the values for the class weights specifically for each classification task. First, we automatically calculated class weights based on the distribution of the different labels in the training dataset and then optimized these using a smoothing factor.

Class weights: To predict binary labels for offensive language in **Subtask 1**, we used full weights with the values for `OTHER`: 1.24, and `OFFENSE`: 2.05. With 3-fold cross-validation on the training dataset, this leads to a macro-average F1-score of 76.37%. For **Subtask 2** (fine grained classification) we found a smoothing factor of 5 applied to the class weights to be optimal. This leads to the weights for `OTHER`: 1.10, `ABUSE`: 2.37, `INSULT`: 2.08, and `PROFANITY`: 6.06. This means that we try to boost the scores for the infrequent classes, but not as much as their inverse frequency would suggest, since having too many different classes might be detrimental to the overall performance. We achieve a macro-average F1-score of 58.06% using 3-fold cross-validation. For **Subtask 3**, we found it most optimal to use a smoothing factor 0.5, increasing the values suggested by the inverse frequency of the labels in the dataset, which might be justified by the highly skewed distribution of the two labels. The resulting class weights are `EXPLICIT`: 1.30, and `IMPLICIT`: 14.12. With 3-fold cross-validation, this leads to a macro-average F1-score of 64.79%.

Final neural network: Our final neural network (variant for **Subtask 1**) is shown in Figure 1. The network takes three inputs: `Input_Text` is the tokenized/normalized tweet, `Input_POW-meta` are the additional word-level features from our POW lexicon, and `Input_Meta` are the tweet-level metadata features (37 in total: 27 basic + 10 from the lexicon).

To explain the sequence input of the tokenized data into the network, it should be noted that these sequences are all set to a fixed length (49 here, see the 2nd dimension size of the first 2 input layers in the figure). We calculated a suited maximum sequence length of the normalized input tweets to be 49 based on the given training dataset. This value is set so that a maximum of 5% of the tweets has to be cut off, i.e., 95% of the tweets have less than than or equal to 49 words after our normalization (such shorter sequences are (pre-)padded). The input sequence (tokenized and normalized tweet) is then transformed into numerical values using the augmented word embeddings (as described above)

and further encoded using a CNN with 6 parallel branches. The (flat) output of the text encoder (see output of the layer CNN; 384 dimensions) is then concatenated with the output of the metadata sub-network (see output of the `Meta_Encoder`; 128 dimensions) resulting in 512 encoded feature values, on which the final layer produces a value that we can interpret as a binary prediction. We understand the decision-making process of this neural network to be based on two main criteria:

- n-grams of words that hint at the use of offensive language, which the CNN is constructed to identify,
- predefined metadata features based on rules and lexicons (metadata sub-network).

3 HAU3: Random Forest

This model was only trained on this year's training data, where each message was mapped to a vector of (lowercase) character trigrams features, e.g., scheiß = {sch, che, hei, eiß}, and word unigram features. All features are binary (i.e., weight 1 if present in message, weight 0 if not). We used the Random Forest algorithm with the following hyperparameters: a 100 trees, each with a random subset of no more than 750 features, and a minimum leaf node size of 3. The algorithm was written from scratch; an additional incentive was to test its performance in a real-world task and check whether it is eligible for inclusion in our new open source NLP & ML toolkit for the Python programming language, Grasp.py.[2] This is a smaller, faster and easier-to-use version of our Pattern toolkit (De Smedt and Daelemans, 2012).

In general, the performance of the model is unremarkable: an average F1-score of 69.75% for **Subtask 1**, with a poor recall for `OFFENSE` tweets (43.71%) and a good recall for OTHER tweets (90.34%). To freshen up, recall and precision can be understood intuitively as follows: suppose we create a dashboard of today's offensive messages on Twitter. The dashboard shows a 100 messages, which are all offensive, so precision (cf. quality) is 100%. If the list also shows irrelevant messages (false positives) then precision will be lower. Now suppose that in reality there were a 1,000 offensive messages today. This means that the dashboard missed out on 900 (false negatives) and has a low recall (cf. quantity) of only 10%.

[2] `https://github.com/textgain/grasp`

In this light, in our approach a lot of offensive messages slip through undetected, but few non-offensive messages are misclassified. Arguably, this could still make the classifier useful in real-life, since, from a user experience standpoint, under-blocking (more offensive content slips through) is better than over-blocking (more users are falsely accused of being offensive). A human moderator equipped with a dashboard built on top of our classifier would see about half of the daily offensive content, in a list where he or she would have to ignore about 1/3 of irrelevant results (68.06% precision).

4 HAU2: POW lexicon

This system is based on a lexicon of 2,850 German words that express profanity and offense, with manually-annotated intensity scores. The Profanity & Offensive Word list (POW) originates from our work in last year's GermEval (see De Smedt and Jaki, 2018), where we used 50 handpicked, high-precision "seed" words to automatically extract 1,250 similar words from the German Twitter Embeddings (Ruppenhofer, 2018). During the past year, the list was further expanded and annotated for intensity (0-4, e.g., *radikal* 'radical' = 1, *schwein* 'pig' = 3, *abschaum* 'scum' = 4) by four annotators at the University of Hildesheim. Each entry in the list also has an English translation, extracted from Google Translate and manually reviewed, and up to 9 fine-grained tags such as `PROFANITY`, `DEHUMANIZATION`, `RIDICULE`, `SEXISM`, `RACISM` and/or `EXTREMISM`, added by the annotators. The set of tags is an open and growing collection. The current tags were chosen intuitively based on prior studies in online German right-wing extremism (Jaki and De Smedt, submitted), misogyny (Jaki et al., 2019), and jihadism (De Smedt et al., 2018). The advantage of such a richly-annotated resource is that offensive words can easily be highlighted (e.g., in a dashboard), making it a useful and explainable support tool for real-world applications where human moderators have the final say.

The classification algorithm is a rule-based script that takes a given message as input, scans which of its words are also in the lexicon, and then returns whether or not it is `OFFENSE` as output, based on a weighting scheme tweaked by trial-and-error for each variable (i.e., intensity scores + tags). As it turns out, this heuristic approach achieves is not

outperformed a lot by the Random Forest classifier: an average F1-score of 68.13% for coarse-grained **Subtask 1**. Its main weakness is a low recall for OFFENSE (37.11%), which in theory could be overcome by expanding the lexicon with more entries, which is a cost-effective solution when student assistants are available. This is a task that we have planned for future work. For example, our Dutch counterpart lexicon currently has 10,000 entries. Again, moderators of such tools will miss out on half of the offensive content, but it is often more desirable to review 1,000 messages of which 2/3 are really offensive than to wade through 10,000 without any prior warning system.

Interestingly, this approach has a better precision (cf. quality) in the fine-grained task for predicting INSULT (48.28%) than our best CNN (35.56%), presumably because it can depend on handcrafted insights from the PROFANITY + RIDICULE tags. This begs the question what other tags can be useful for future work, with THREAT and ILLEGAL on top of our list.

In related research on political debate (Jaki et al. 2019), we have also used the lexicon to detect offensive content on the Facebook pages of the major political parties in the German federal elections 2017 and their leading candidates. Employing the POW list, we showed that the female candidate's pages displayed a higher proportion of offensive content, for example.

5 Discussion

In this section, we sum up the HAU results for the different subtasks and discuss their implications.

In GermEval 2018, we have focused primarily on the coarse-grained classification task, with first attempts in the fine-grained classification task (Schäfer, 2018). This year, **Subtask 1** (binary classification) was tackled by employing the three models described in Sections 2 to 4, and **Subtask 2** (fine-grained classification) with the first and last models (CNN and lexicon). We assumed that the most difficult task to solve computationally might be **Subtask 3** (implicit vs. explicit offense) as understanding implicit offense often involves inferral processes that involve a high amount of contextual knowledge. As a consequence, implicit offense is often hard to pin down on the text level.

As the best results were yielded for **Subtask 1** in GermEval 2018 (Wiegand et al., 2018), it is little surprising that the models achieved our best results for the binary classification task in 2019: a macro-average F1 score of 70.46% for the CNN, 69.75% for the Random Forest, and 68.13% for the POW lexicon. Surprisingly however, the overall performance in **Subtask 3** was higher than in **Subtask 2**, which means that the identification of different types of offense turned out more difficult to solve (macro-average F1 score of 45.34% for the CNN and 40.8% for the POW lexicon) than the identification of implicit offense (macro-average F1 score of 69.3% for the CNN).

The general conclusions we can draw from our submission are the following: Firstly, it is very difficult to outperform CNNs in the detection of offensive language, especially if they have been extensively enriched by a multitude of additional features. Secondly, although even a very substantial lexicon will not outperform more up-to-date approaches, it can still achieve surprisingly good results, in so far as word lists do not necessarily entirely fall behind in comparison to CNNs and Random Forests. This is particularly important to know because there are many real-life scenarios where lexicon-based approaches could be employed as a simple, but useful aid in the detection of offensive speech, with the advantage of bringing a maximum of transparency to the identification process. Thirdly, the integration of lexicons like POW into CNNs can help to trigger better overall results. This shows that we should not exclusively rely on statistical approaches for solving the problem in future work, but recognize the value of thorough qualitative preparation work (such as the result of the annotation process).

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, 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, pp. 54-63.

Tom De Smedt and Walter Daelemans. 2012. *Pattern for Python*. JMLR, 13:20632067.

Tom De Smedt, Guy De Pauw, and Pieter Van Ostaeyen. 2018. *Automatic detection of online jihadist hate speech*. CLiPS CTRS, 7:130.

Tom De Smedt and Sylvia Jaki. 2018. *Challenges of Automatically Detecting Offensive Language Online: Participation Paper for the Germeval Shared Task*

2018 (HaUA). In Proceedings of GermEval 2018, 2732.

Europol. 2019. *European Union Terrorism Situation and Trend Report. European Union Agency for Law Enforcement Cooperation*. https://w019_final.pdfww.europol.europa.eu/sites/default/files/documents/tesat_2.

Sylvia Jaki and Tom De Smedt. 2018, submitted. *Right-wing German hate speech on Twitter: analysis and automatic detection*.

Sylvia Jaki, Tom De Smedt, Maja Gwóźdź, Rudresh Panchal, Alexander Rossa, and Guy De Pauw. 2019. *Online hatred of women in the Incels.me forum: Linguistic analysis and automatic detection. Journal of Language Aggression and Conflict*. http://doi.org/10.1075/jlac.00026.jak.

Sylvia Jaki, Wolf Schünemann, Tom De Smedt, and Stefan Steiger. 2019. *Who is polluting the debate?* Unpublished conference paper.

Josef Ruppenhofer. 2018. *German Twitter Embeddings*. http://www.cl.uni-heidelberg.de/english/research/downloads/resource_pages/GermanTwitterEmbeddings/GermanTwitterEmbeddings_data.shtml.

Johannes Schäfer. 2018. *HIIwiStJS at GermEval-2018. Integrating Linguistic Features in an Neural Network for the Identification of Offensive Language in Microposts*. In Proceedings of GermEval 2018, 104112.

Johannes Schäfer and Ben Burtenshaw. 2019. *Offence in Dialogues: A Corpus-Based Study*. In Proceedings of Recent Advances in Natural Language Processing (RANLP 2019), pages 1085-1093, Varna, Bulgaria, Sep 2-4 2019.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. *Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language*. In Proceedings of GermEval 2018, 110.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. *Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval)*. arXiv preprint arXiv:1903.08983.