Feature Explorations for Hate Speech Classification

Tatjana Scheffler **Erik Haegert** **Santichai Pornavalai** **Mino Lee Sasse**
Linguistics Department
Research Focus Cognitive Sciences
University of Potsdam, Germany
tatjana.scheffler@uni-potsdam.de

Abstract

In this work, we present a hate speech classifier for German tweets for the GermEval2018 Shared Task. Our best models are Linear SVM classifiers using character ngrams as well as additional textual features. We achieve a macro F_1-score of 0.77 (95% confidence interval: ±0.04) in cross validation. We also present an ensemble classifier based on majority voting of the three component models.

1 Introduction

Social media contains large amounts of user-generated text. Unfortunately, a portion of these user comments are hurtful to other people, incite aggression or violence, or contain offensive content. This kind of material is known as "hate speech" on the internet and termed "offensive language" in the GermEval2018 Shared Task[1]. Detecting offensive language automatically is important for moderating online discussions and in order to identify trolls.

In this work, we present a hate speech classifier for German tweets based on the GermEval2018 Shared Task. Our best models are Linear SVM classifiers using character ngrams as well as additional features. We achieve a macro F_1-score of 0.77 (95% confidence interval: ±0.04) in cross validation. In the following, we describe our exploration of the data, the models trained, and some pointers for future research.

2 Related Work

Hate speech detection has received quite a bit of attention recently, in particular for English social media data. Waseem has worked on hate speech classification of tweets, and has shown that the categories are often hard to define and the classification of a tweet as offensive or not depends on features of the recipient as well as of the sender (Waseem, 2016). This indicates that it would be very difficult to detect hate speech only based on the text of a social media comment, since important context is missing, such as who the conversation participants are (Are they themselves part of a marginalized group?), how they usually communicate, and what the surrounding discourse context is. Ross et al. (2017) agree that hate speech annotations are a very subjective task, with low agreement among humans. In other work, Waseem and Hovy identify character ngrams as good predictive features for identifying hate speech from English tweets (Waseem and Hovy, 2016), since they are somewhat robust to misspellings and other variants.

Work by Wulczyn et al. (2017) on attacks in Wikipedia shows that the necessarily subjective judgments about offensive language by annotators can be used to inform a classifier. In their work, they combine many human judgments to build a system that approximates the performance of several naive judges.

For German, Bretschneider et al. (2014) present an early pattern-based hate speech classifier for tweets. They extend this pattern-based approach towards detecting hate speech specifically directed at foreigners in Facebook data (Bretschneider and Peters, 2017).

So while there is some previous work and some discussion on the types of classifiers and even data to use, this is by no means a solved problem and one that is receiving lots of attention. Concurrently to this German Shared Task, the 1st Workshop on Trolling, Agression, and Cyberbullying (TRAC)[2] is taking place, colocated with Coling, which includes a Shared Task on identifying hate speech in English and Hindi.

[1] https://projects.fzai.h-da.de/iggsa/

[2] https://sites.google.com/view/trac1

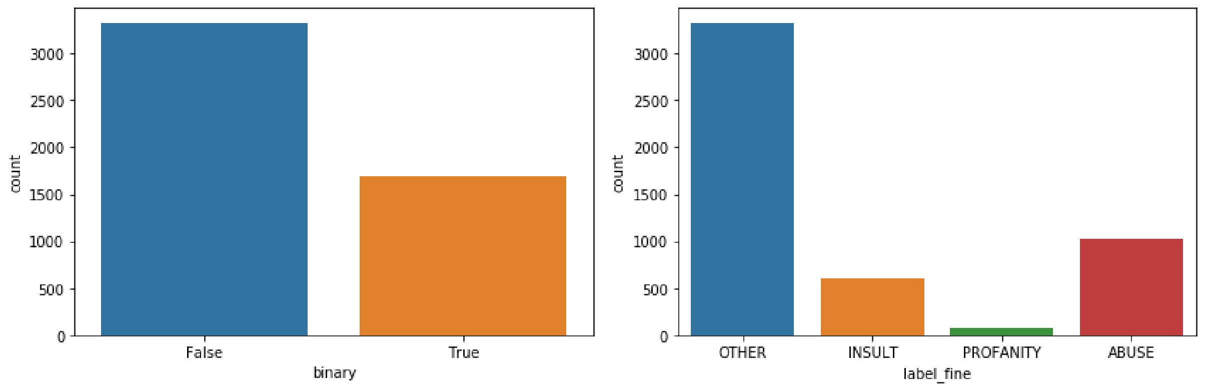

Figure 1: Distribution of "binary" (COARSE) and FINE labels in the training data.

3 Data

The training data for this task consisted of 5009 German tweets provided by the task organizers. The tweets were annotated as specified in the annotation guidelines[3] in two levels: in a COARSE classification into OFFENSE and OTHER, and in a FINE grained classification, further subdividing offensive tweets as PROFANITY, INSULT, or ABUSE. The distribution of labels in the training data is shown in Figure 1. It is obvious that the data is quite unbalanced, in particular for the FINE classification, which contains only 71 cases of PROFANITY. In the following, we concentrate on Task1, the COARSE/binary classification into offensive or non-offensive speech.

3.1 Preprocessing

For preprocessing the data, we use different preexisting packages. For the data exploration reported in this section, we use the SoMaJo[4] social media tokenizer (Proisl and Uhrig, 2016) and the SoMeWeTa[5] part of speech tagger for social media data (Proisl, 2018). These two packages show the best performance for German social media data (for example, with regard to special tokens such as hashtags and emoji). The tokenizer is also able to output token types, which are useful in the computation of further features (e.g., the frequency of emoticons, etc.). The frequency of different token types in the training data is listed in Table 1.

We conjecture that special tokens such as @-mentions and URLs can lead to overfitting in word

token type	**f(OFFENSE)**	**f(OTHER)**
URL	1	4
XML entity	57	135
abbreviation	191	384
action word	12	3
date	3	33
email address	1	3
emoticon	590	997
hashtag	414	1183
measurement	6	8
mention	2321	5693
number	175	509
numb. comp.	55	42
ordinal	15	57
regular	31227	57837
symbol	5060	10095
time	4	15
total	40132	76998

Table 1: Frequency of token types in the training data.

[3] http://www.coli.uni-saarland.de/~miwieg/Germeval/guidelines-iggsa-shared.pdf
[4] https://github.com/tsproisl/SoMaJo
[5] https://github.com/tsproisl/SoMeWeTa

or character ngram models, since the test set may not exactly match the training set. For this reason, we experimented with replacing @-mentions and URLs/email addresses by passepartout-tokens ("*A*" and "*U*", respectively). In addition, we experimented with stemming using the Snowball stemmer.

In some runs described below, we used alternative preprocessors (indicated in the model description). Model 1 employed the TreeTagger[6] and a stop word list from NLTK[7]. Model 2 used the spaCy[8] NLP package for tokenization, lemmatization, and POS tagging. The German spaCy model was computed on the Tiger and WikiNER corpora. This model further removed the 232 stop words from the Python `stop-words` package.

3.2 Data Exploration

In previous work, character ngrams have proven very successful in supervised classification of hate speech, since they are able to capture both profanities and insults, as well as the fact that hate speech often contains misspellings, disguised words ("A***"), or other symbol combinations. In order to see whether these predictions from English surveys of hate speech are mirrored in the German Shared Task data, we analyzed the occurrences of slurs, OOV items, and other special tokens in the offensive and non-offensive tweets.

Slurs. The annotation guidelines focus in part on the person-directed nature of offensive speech. Therefore, we analyze whether offensive tweets contain more slurs than non-offensive tweets. We use three lists to detect slurs: (i) the German insult lexicon[9] linked on the Shared Task site, (2) a manually compiled list of 8 items such as "Lügenpresse" and "Vasall", and the list of words classified as SWEAR words (category 66) or ANGER (category 18) from the German LIWC dictionary (Wolf et al., 2008), including 242 items. We used LIWC because the insult lexicon contains only nouns that can be used to refer to people, excluding many offensive terms such as "verdammt". In Figure 2, we show the number of tweets that contain 0, 1,... swear words in the training corpus, computed on stemmed tokens (see "Preprocessing" above).

[6]http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

[7]https://www.nltk.org/

[8]https://spacy.io/

[9]http://www.hyperhero.com/de/insults.htm

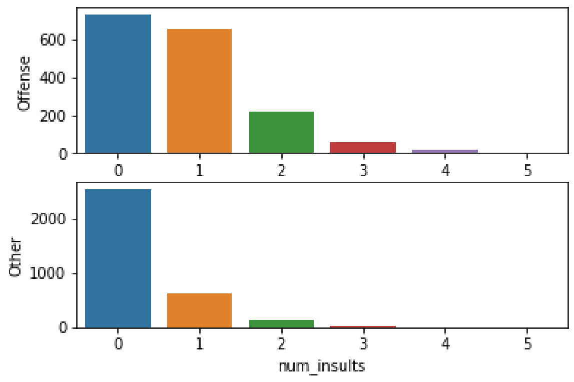

Figure 2: Histogram of the number of insult words per tweet. Top panel = offensive tweets, bottom panel = non-offensive tweets.

It is obvious that offensive tweets (shown in the top panel) contain relatively more slurs than non-offensive tweets (bottom panel). More than half of offensive tweets contain at least one slur, while non-offensive tweets rarely contain any. In fact, this feature alone can be used to classify the tweets for the binary task. Taking the presence of any slurs to indicate an offensive tweet, we reach a macro F1-score of 0.67 on the training set.

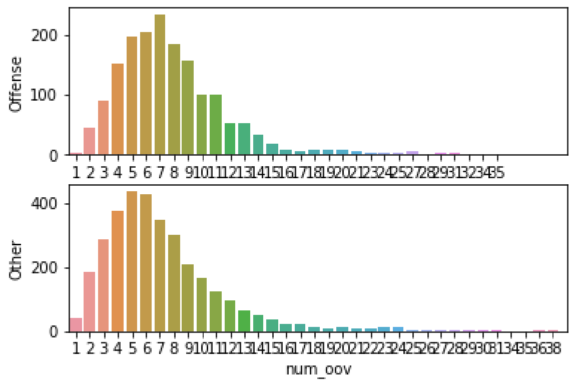

Figure 3: Histogram of OOV tokens per tweet.

Misspellings. Previous research has shown that hate speech is more likely to contain misspellings and alternative spellings (including lengthenings or words disguised by asterisks) than non-hate speech. In Figure 3 we plot frequency counts of out of vocabulary (OOV) items per offensive (top) vs. non-offensive (bottom) tweet in the training data. We use the vocabulary provided by Spacy. The data confirms that offensive German tweets contain slightly more OOV tokens than non-offensive tweets (mode = 7 vs. mode = 5).

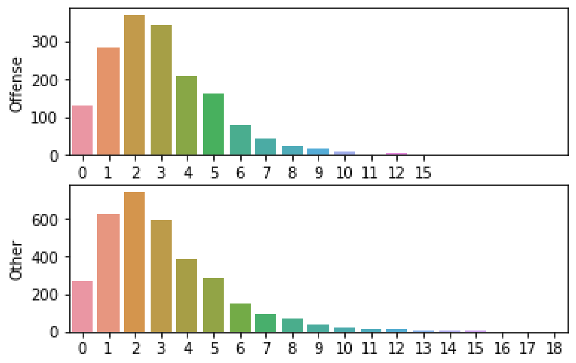

Figure 4: Histogram of the number of symbols per tweet.

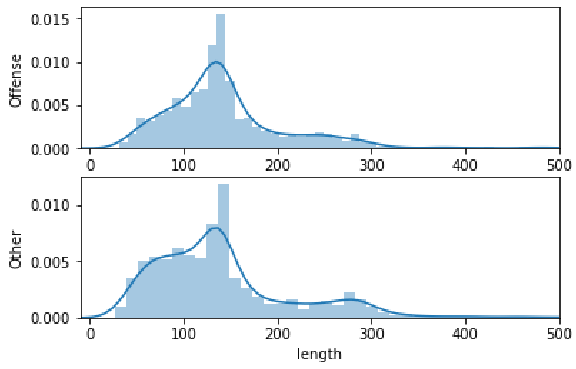

Figure 5: Distribution of tweet lengths in offensive and non-offensive tweets.

Special items. We also analyzed the frequency of special items such as user mentions, hashtags, and symbols in the two subcorpora. However, no significant differences were found, indicating that the mere occurrence of these types of items would not make for very good features for classification. For example, the frequency of punctuation symbols is shown in Figure 4.

Length. Finally, we plot the length of the tweets (in characters) in Figure 5. It can be seen that non-offensive tweets (bottom) are more likely to be shorter (under 140 characters) than offensive tweets. We therefore consider this feature in some of our models.

4 Models

In this work, we report on three supervised classification models for Task1, the binary classification task as offensive/non-offensive tweets. The three models were developed relatively independently and show similar performance, but different classification decisions. In order to combine the information from all models, we created a simple ensemble model of the three classifiers by employing majority voting on the individual systems. We submitted this ensemble prediction as `Potsdam_coarse_3.txt`. Note that this ensemble model could not be evaluated by cross-validation, since the component models were trained on the entire training set. Its performance is therefore unknown at the time of writing. In the remainder of this section, we describe the three component models. The first two were submitted as `Potsdam_coarse_1.txt` and `Potsdam_coarse_2.txt`, but the third one was not individually submitted. Its output on the test set can be provided upon request.

4.1 Model 1: `Potsdam_coarse_1.txt`

We trained a Linear Support Vector Machine using character n-gram features combined with word embeddings.

Feature extraction and preprocessing. We preprocess and extract the word-vectors of both the training and test data offline for ease of development. However we could also implement an online version. Most of the preprocessing time is consumed by loading the Word2Vec model.

We perform the following steps:

- The raw text data is used to extract character n-grams. We have found 4-5-grams as the most optimal.
- We compute pre-trained word embeddings trained on German Twitter data from spinningbytes[10]. We use only the most frequent 1 million words due to space issues.
- The tweet is lemmatized and filtered through a stopword list using TreeTagger and NLTK.
- Each word in the tweet is then fitted in the word2vec model to yield a vector with 200 floats. The vectors are weighted with tfidf scores and averaged to create a feature vector for the tweet. Although both character ngram and word2vec features perform well independently (vanilla character ngrams scoring slightly better), improvements on the combined model are seemingly minute.

[10]`https://www.spinningbytes.com/`

- We add other textual features such as BOW, number of words with all caps, tweets containing insults, and punctuation. However, they don't offer much improvement to word embeddings and character n-grams (but see below).

- Sentiment analysis is added to the feature vector as (polarity, subjectivity). Both are floats between -1/1. The sentiment analyzer used here is the default from TextBlob[11]. This is not state of the art and better SA might yield better results.

- Grid search showed that feature selection of only the 5000 best features leads to the best performance in cross validation (parameters tested: $n \in \{5k, 10k, 50k, 100k\}$). Due to time constraints, we were not able to do a thorough analysis of which features were selected in this step.

Classifier and crossvalidation. We compared 3 different classifiers for this task: Logistic Regression, SVM, and Adaboost. In our experiments, SVM performed consistently better than the other two but not by much. We performed a grid search over 10-fold cross validation over SVM and found the loss penalty C = 0.1 to be optimal. We evaluate the results using 10-fold cross validation and F_1-macro as metric. The model consistently scores $F_1 = 0.77 \pm 0.04$ and is thus our best individual model.

4.2 Model 2: `Potsdam_coarse_2.txt`

Model 2 also trains a Linear Support Vector Machine, but uses the PassiveAggressiveClassifier package from Python's Scikit-Learn to do it. Its cross validation results are $F_1 = 0.74 \pm 0.05$.

Feature extraction and preprocessing. In this model, we use the spaCy NLP toolkit for preprocessing. We perform the following steps:

- Sentences are tokenized, lemmatized and tagged using spaCy.

- Stop words and punctuation are excluded.

- If a word is found in the list of insults (insult dictionary as linked from the Shared Task), a special character "I" is added to the end of it.

[11] `https://textblob.readthedocs.io/en/dev/`

- Finally, part of speech tags are added behind their words in the list of tokens.

- The token-pos list is recombined with spaces and we compute character ngrams in the range of (1,5) on this combined lemma-pos string.

- The features are transformed using tfidf and fed into the classifier.

4.3 Model 3

The third model is based on the analysis of the training data presented in Section 3.2.

Feature extraction and preprocessing. We use three kinds of features:

- POS ngrams: uni-, bi- and trigrams, based on the SoMaJo tokenizer and SoMeWeTa tagger.

- Character ngrams in the range (3,5) based on the tokenized text. This text keeps idiosyncrasies of the original tweet and does not exclude stop words or punctuation, as they may turn out significant for classification. The only normalization done here is tokenization and the replacement of @-mentions and URLs by passepartout-tokens, in order to avoid overfitting.

- Other textual features. These include the number of insults based on the extended slur lexicon we created, the number of OOV tokens, and the length of the tweet. These features were normalized to standard mean and variance.

- Again, we select the 5000 best features before feeding them to the classifier.

Classifier and crossvalidation. We evaluated different classifiers such as Logistic Regression, Decision Trees, and SVM. Logistic Regression and SVM perform consistently better than the others, with SVM a little bit better on some runs. Our further experiments thus concentrated on Linear SVMs. On 10-fold cross validation, the model's score was $F_1 = 0.76 \pm 0.05$. We performed feature ablation between the 3 feature groups, which showed that the performance is mainly carried by the character ngrams (see Table 2). Note that the textual features are only three features in total (in the "text only" condition, there was no feature selection). The ablation also shows that POS ngrams might hurt the performance ever so slightly, which might suggest excluding them in future work.

configuration	F_1-macro
char only	0.755 ± 0.051
pos only	0.608 ± 0.038
text only	0.664 ± 0.045
char + pos	0.752 ± 0.044
char + text	$\mathbf{0.757 \pm 0.049}$
pos + text	0.656 ± 0.037
all	0.756 ± 0.052

Table 2: Feature ablation for model 3. Feature types are character ngrams (char), pos ngrams (pos), or the three textual features (text).

	model 1	model 2	model 3
430	OFFENSE	OFFENSE	OFFENSE
80	OFFENSE	OFFENSE	OTHER
185	OFFENSE	OTHER	OFFENSE
88	OTHER	OFFENSE	OFFENSE
101	OFFENSE	OTHER	OTHER
245	OTHER	OFFENSE	OTHER
166	OTHER	OTHER	OFFENSE
2237	OTHER	OTHER	OTHER

Table 3: Confusion matrix of the three component models on the test set.

4.4 Ensemble: `Potsdam_coarse_3.txt`

Model 3's prediction was not submitted individually. Instead, it was used as the tie-breaker in the majority voting ensemble classifier combining all three individual models. The ensemble's output was submitted as Run 3. The overlap and differences in classification decisions between the component models is shown in Table 3.

In the majority of cases, all models agree (top and bottom sections). In addition, models 1 and 3 agree more often than they each agree with model 2 (which is different wrt. the kind of preprocessing performed). The final ensemble classifier differs in its classification decision from model 1 189 times, from model 2 431 times, and from model 3 247 times. We therefore expect its performance to be similar to the performance of model 1.

5 Results and Discussion

In this work, we present a hate speech classifier for German tweets for the GermEval2018 Shared Task. Our best models are Linear SVM classifiers using character ngrams as well as additional textual features. We achieve a macro F_1-score of 0.77 (95% confidence interval: ±0.04) in cross validation. We also present an ensemble classifier based on majority voting of the three component models. The cross validation performance of our models is summarized in Table 4, but note that the ensemble classifier cannot be included here.

model	F_1-macro	
insult words	0.67	
model 1	0.77	±0.04
model 2	0.74	±0.05
model 3	0.76	±0.05

Table 4: Cross validation performance of the three models.

In our experiments, as in previous work, character ngrams were the most useful features for classification (outperforming word-based lexical features but also manually specified features). The best ngrams at the character level are 4- and 5-grams, which can capture most of a word or even the boundary between two words. It is hard to improve over a character ngram baseline by feature design, but our analysis identified a few phenomena where offensive and non-offensive tweets show significant differences: the presence of slurs (including aggressive words), the frequency of OOV tokens, and the length of the tweets.

In future work, of course a larger amount of data may be helpful for training classification systems. This would be particularly helpful for the second, fine-grained task, where our classifiers showed really poor performance. In addition, we'd like to explore linguistic approaches such as pattern-based approaches, which have been useful for similarly difficult tasks such as sarcasm detection (Davidov et al., 2010). It is also clear that the annotations are difficult even for humans, and thus multiply annotated data would both be more fair to the data, as well as might turn out helpful for classifiers (which could use human (dis)agreements as indicators of high or low label confidence). Finally, we are certain that the discourse context and other metadata could hugely improve performance, and would thus like to explore hate speech classification on data sets that include such metadata, instead of just on isolated tweet texts.

Acknowledgments

We would like to thank Daniela Feinhals for an analysis of the annotation guidelines.

References

Uwe Bretschneider and Ralf Peters. 2017. Detecting offensive statements towards foreigners in social media. In *Proceedings of the 50th Hawaii International Conference on System Sciences*.

Uwe Bretschneider, Thomas Wöhner, and Ralf Peters. 2014. Detecting online harassment in social networks.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 107–116. Association for Computational Linguistics.

Thomas Proisl and Peter Uhrig. 2016. Somajo: State-of-the-art tokenization for german web and social media texts. In *Proceedings of the 10th Web as Corpus Workshop*, pages 57–62.

Thomas Proisl. 2018. Someweta: A part-of-speech tagger for german social media and web texts. In *Proceedings of LREC*.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.

Zeerak 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.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on NLP and computational social science*, pages 138–142.

Markus Wolf, Andrea B Horn, Matthias R Mehl, Severin Haug, James W Pennebaker, and Hans Kordy. 2008. Computergestützte quantitative textanalyse: Äquivalenz und robustheit der deutschen version des linguistic inquiry and word count. *Diagnostica*, 54(2):85–98.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399. International World Wide Web Conferences Steering Committee.