Preprocessing and the Detection of Offensive Language in German Tweets: How Different Methods May Improve the Performance of Different Classifiers

Sebastian Reimann
Uppsala University
sebastianmichael.reimann.9692@student.uu.se

Abstract

So far, the preprocessing methods applied in papers dedicated to the detection of offensive language in German Twitter data varied greatly. This paper attempts to bring some more clarity into the issue of which techniques are recommended for which classifier by experimenting with emoji replacement, hashtag splitting and different ways of capitalization. In conclusion, it can be said that replacing emojis with text may have some advantages but methods for avoiding unwanted bias here need to be explored. Moreover, for non-neural classifiers a better way may be a sentiment score as a separate feature. Splitting camel-cased hashtags had only minor but rather positive effects. Changing capitalization, especially truecasing helped the BERT classifier but was not beneficial for the other classifiers.

1 Introduction

In recent years, the detection of offensive language in German tweets was tackled in two shared tasks, namely the GermEval 2018 shared task and task two of GermEval 2019. An overview over the former is given in Wiegand et al. (2018) and the latter is presented in Struß et al. (2019). Different classifier architectures were used in these shared tasks, some of which are presented in the next section.

Moreover, a considerable variation when it comes to preprocessing the data can be observed. Preprocessing is not trivial, especially when working with data from Twitter. Angiani et al. (2016) argue that data from Twitter may be noisy and contain uninformative parts. On the other hand, some elements specific to social media data could contain information of particular relevance for the present task. Kralj Novak et al. (2015) for example demonstrate in their emoji ranking the emotional value of a wide range of emojis, which could for example also be interesting in the detection of offensive tweets.

The question thus arises how the automatic identification of offensive language in German Twitter data may be improved through preprocessing. This study is dedicated to this question. A German Twitter dataset for the detection of offensive language will be prepared in different ways and three types of classifiers will be trained and tested on these variations and their performance will be compared, in order to find out which preprocessing methods are beneficial here, to find a good combination of methods for each classifier and eventually make a small contribution to improving the detection of offensive language overall.

2 Related Work

A first dataset for the detection of offensive language in German Twitter data was presented by (Ross et al., 2016). Their corpus, consisting of 541 tweets, related to the 2015 refugee crisis, is however small and their study was rather on the user's perception on what can be considered to be hate speech and the reliability of annotations than training an actual system that detects offensive language. Larger datasets were presented in GermEval 2018 and in task 2 of GermEval 2019. Both tasks consisted of subtasks distinguishing between offensive and non-offensive language and a more fine grained classification. The latter one had an additional task on the detection of implicit and explicit offensive language. Another dataset related to offensive German language was presented by Zufall et al. (2019). Here however, it was focussed on whether the tweet presents an offence according to German law. The annotations were based on different binary categories related to the German legal system, such as the type of target and the specific type of offence.

2.1 Classifiers

The classifiers used in the binary subtasks of GermEval are of particular interest for this paper. Here, several ensemble classifiers were among the best performing ones, such as a CNN model as a base classifier and logit averaging on top of that (von Grünigen et al., 2018), which achieved an F1-macro score of 75.52 on the GermEval 2018 test set, or ensembles of RNNs and CNNs combined (Stammbach et al., 2018), which yielded the an F1-Macro of 74.64 on the test set of the shared task.

Overall though, SVMs were the most popular classifier type (Wiegand et al., 2018). The contribution of De Smedt and Jaki (2018), which is based on a linear SVM and where each tweet has about 350 features performed best with an F1-Macro of 72.74. Decision trees or boosted decision trees were sometimes considered as well, but often outperformed by other classifiers such as SVMs, like in Scheffler et al. (2018), even though their AdaBoostClassfier reached an almost comparable performance to the SVM.

In task 2 of GermEval 2019, the classifier of Paraschiv and Cercel (2019) performed best with an F1-score of 76.95. It used a BERT-model, a pretrained, bi-directional transformer model (Devlin et al., 2019), which was in this particular case pretrained on a vast amount of German Twitter data. BERT was also used in Risch et al. (2019), which created an ensemble of multiple models that performed slightly worse with an F1-score of 76.4 on the first subtask. This model was not pretrained on Twitter-specific data but on data from Wikipedia and news texts and was subsequently finetuned on the specific training data of the shared task. SVMs remained a popular choice as well and (Schmid et al., 2019), an SVM-based classifier, which used word embeddings pretrained on German tweets in combination with several lexical features, even achieved an F1-score of 76.63.

2.2 Preprocessing in Previous Papers

Angiani et al. (2016) experimented with the preprocessing methods of replacement of emoticons with text, replacing negation constructs such as *don't* with *do not*, detection of spelling errors, stemming and removal of stopwords for general sentiment analysis on Twitter data. Most of their experiments led to a slight improvement over a given baseline with only little preprocessing applied. This shows that preprocessing may play a relevant role here. However, their work did not focus specifically on the detection of offensive langauge and they used a Naive Bayes Classifier, which was rarely used in the previously mentioned shared task. Thus, a look at some of the preprocessing there is necessary.

Paraschiv and Cercel (2019) made use of a wide range of preprocessing methods. They replaced emojis with spelled-out words, removed the #-character at the beginning of hashtags and split hashtags into words, transformed usernames, weblinks, newline markers, numbers, dates and timestamps to standard tokens and manually corrected spelling errors. They retained the orignal capitalization of the data. On the other had, Risch et al. (2019) had a much more minimalistic approach to preprocessing through only normalizing user names and leaving the rest as it is.

For the SVM classifier of Schmid et al. (2019), the data was lowercased and lemmatized, the #-character of the hashtag was removed, as well as stop words. Emojis were dealt with by obtaining a sentiment score for each emoji through the sentiment ranking for emojis by Kralj Novak et al. (2015). They were added together with sentiment scores ranging from obtained trough SentiWS Remus et al. (2010) for words of the sentence. Both scores range from -1 to 1 and this eventually resulted in a separate feature. Scheffler et al. (2018) lemmatized the data as well and removed stop words, however they neither mentioned hashtags nor capitalization when it comes to preprocessing. Moreover, they did not include emojis when modelling a sentiment score as one of their features.

3 Data

For all experiments, the dataset from the GermEval 2019 Task 2 was used. It consists of 7,025 German tweets, sampled from users that were identified to regularly post offensive content. In the selection they sampled data from both extremes of the political spectrum so that there will be a considerable variety of topics. For subtask one, the tweets were either labeled as *OFFENSE* or as *OTHER*, which will also be used in this project. Apart from the addition of tokens that mark linebreaks, no preprocessing steps were conducted for the original data.

4 Preprocessing

4.1 Emojis

When looking at the literature in section 2.2, differences with respect to the treatment of emojis could be identified. We will explore the approaches of both Paraschiv and Cercel (2019) and Risch et al. (2019) in this study. Moreover, we will calculate for the non-neural classifiers an emoji sentiment score, also through the ranking of Kralj Novak et al. (2015), together with sentiment scores for words, through SentiWS (Remus et al., 2010).

Paraschiv and Cercel (2019) point out that there was room for improvement compared to spelling out the emojis as they did. Consequently, we approach this in a slightly different way. Instead of using the English textual descriptions, we translate the descriptions into German. More precisely, we feed the textual descriptions that are used in the emoji python library to Google Translate and then replace the English descriptions with the corresponding German ones.

4.2 Hashtags

Concerning hashtags, it was a relatively common practice to remove the #-character at the beginning of a hashtag. Consequently, we will do this in most of the experiments here as well. Additionally, we also try out splitting camel-cased hashtags such as in Paraschiv and Cercel (2019). Here, we use a regular expression to achieve such splits.

4.3 Capitalization

A final point that was approached in different manners was capitalization. We attempt both, retaining the original capitalization and lowercasing here. Moreover, we explore a third option: Truecasing, "the process of restoring case information to raw text" (Lita et al., 2003), which essentially results in instances of capitalized words that are normally written in lowercase turned to lowercase again, whereas words written with a capital letter at the beginning such as German nouns may retain this capitalization. Lita et al. (2003) even reccommend their truecasing approach for NLP on noisy data. Additionally, Risch et al. (2019) point out that in their experiments, in words in capslock, almost each letter was recognized as a separate token. Theoretically, truecasing appears to be an elegant solution for this and thus shall be attempted as well.

Truecasing the test and training data shall be achieved by the truecasing scripts from the Moses system (Koehn et al., 2007), which are normally used for Statistical Machine Translation. One script here is for obtaining a truecasing model, trained on the German Wikipedia Text Corpus in the present study. This corpus consists of large, cleaned and preprocessed data from the German Wikipedia. A second script then truecases our data via the trained model. Truecasing in that way moreover requires some previous tokenization before, which is done here through the SoMaJo tokenizer for German social media data (Proisl and Uhrig, 2016).

4.4 Basic Methods

One aspect that was approached by the majority of papers here is normalization of user names. This is done for the data in all experiments, by replacing usernames with a simple *User* token. Moreover, a great number of non-neural classifiers used lemmatizaion in preprocessing, which will consequently be done for the experiments with the SVM classifier and the AdaBoost classifier as well. For lemmatization, we use the spaCy lemmatizer.

5 Classifiers

Given the recent popularity and its the result of the first subtask in the GermEval 2019 shared task on offensive language, BERT seems to be a logical choice for one of the classifier for the experiments in the present study. The german BERT model from deepset.ai [1], which is available through the transformers library in python and which was also used in Risch et al. (2019), thus presents a good possibility for comparison. The model was pretrained on data from a German Wikipedia dump, the OpenLegal dump and 3.6 GB of news articles. This data was cleaned and segmented into sentences by the spaCy library. It makes use of a word piece vocabulary which was created through the sentencepiece tokenizer. As it was not pretrained on particular social media text, it can be assumed that it was originally not necessary to deal with hashtags and emojis.

Several papers from the GermEval shared task reported that more than two epochs already lead to first signs of overfitting (Risch et al., 2019; Paraschiv and Cercel, 2019) and Risch et al. (2019) also recommend batch sizes not to be

[1] https://deepset.ai/german-bert

smaller than 32, which will be done here as well. Table 5 in the appendix presents the hyperparameters of the BERT model. The code for finetuning the BERT model was written with the help of the documents of the transformers library, as well as the tutorials of Joshi (2020) and McCormick and Ryan (2020).

SVMs were previously popular in the detection of offensive language and the SVM in Risch et al. (2019) for example managed to compete with or even outperform many neural classifiers in this field. Thus, we use an SVM as well. The features for the SVM in the present experiments will be similar to the ones used in the second system of Schmid et al. (2019), where word vectors pretrained with a Fasttext model were used. Unfortunately, Schmid et al. (2019) did not make their word embeddings publicly available. Thus, several word embeddings were tested and the 300-dimensional Fasttext embeddings of Grave et al. (2018), trained on the German CommonCrawl and Wikpedia corpora and distributed on the Fasttext page performed best were eventually chosen for the experiments.

In addition to the 300-dimensional vectors, we also add a binary feature which signals if a tweet contains one or more german slurs from the slur dictionary of Hyperhero [2], similar to what was done in Scheffler et al. (2018) and Schmid et al. (2019). Manually extracting and including offensive German verbs and adjectives, such as done in Schmid et al. (2019) was however not possible. The vectors plus the binary feature and the sentiment scores mentioned in the previous were concatenated into a matrix and fed to the SVM classifier.

The ranges of the features in the matrices will differ slightly. In such cases, attributes with a greater numerical range could dominate those with a smaller range and such cases normally require scaling before feeding the data to an SVM (Hsu et al., 2008). We scale the matrices by using the StandardScaler from sklearns preprocessing library. Contrary to Schmid et al. (2019), who were making use of a radial kernel, we will use an SVM with a linear kernel here, namely the SVC from sklearns svm library. Hyperparameter tuning is done through running a grid search over possible values for the regularization hyperparameter C, which are presented in table 6 in the appendix and 5-fold cross validation.

Additionally, in order to get a broader overview over different classifiers, similar to Scheffler et al. (2018) we test an AdaBoost classifier (Freund and Schapire, 1996). Here, we use the AdaBoost classifier from the sklearn package, with a decision tree classifier as a base classifier. The hyperparameters of the decision tree classifier are the default ones of the classifier in the sklearn package. For hyperparameter tuning, we perform a grid search over the values presented in table 7 in the appendix. The suggestions for these values are taken from Brownlee (2020).

6 Experiments

In the experiments, we apply several combinations of preprocessing methods. To be able to better measure the impact of the chosen preprocessing steps, we select a baseline for each experiment, where only a minimum of preprocessing is applied. The baseline data for the BERT classifier is a dataset where only the replacement of usernames will be performed. In a second dataset, both emoji replacement and hashtag splitting will be performed, in combination with truecasing. In other variants, only hashtag splitting and truecasing and only truecasing will be applied and finally, one variant, where emojis were replaced, hashtags were split and the data was lowercased instead of truecased.

The baseline for the SVM and the AdaBoost classifier, will be a version of the dataset, where all the basic operations will be carried out and emojis are removed, to have a better idea which method (emoji replacement or separate feature) leads to the higher improvement over the baseline. Moreover, we will carry out experiments either involving emoji replacement or emoji sentiment scores in combination with hashtag splitting and an experiment were no method apart from extracting the sentiment scores. To test the effects of the different casing approaches, we add an experiment with lowercasing and one with truecasing, too.

7 Results and Evaluation

7.1 Quantitative Analysis

The tables 1 and 2 show the respective scores for each experiment. According to the results for the BERT experiments, the model where hashtags were split and truecasing was performed but the emojis were not replaced performed best and the

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

Experiment	BERT
baseline	71.77
splitting hashtags + lowercasing	72.49
replacing emojis + splitting hashtags + truecasing	72.51
only truecasing	73.24
splitting hashtags + truecasing	73.81

Table 1: F1-macro scores for all experiments with the BERT classifier

Experiment	SVM	AdaBoost
baseline	66.48	58.27
splitting hashtags + lowercasing	66.73	54.80
replacing emojis + splitting hashtags + truecasing	66.74	57.18
replacing emojis + splitting hashtags	66.78	55.35
splitting hashtags	66.90	57.82
only basic methods	67.06	57.46

Table 2: F1-macro scores for all experiments with the SVM and Adaboost classifiers

baseline model performed worst. However, the resulting models after finetuning BERT always differ slightly and the all results presented in Table 1 only differ by a slight margin. Thus, in order to draw conclusions on whether the preprocessing methods here helped, conclusions can only be drawn after the the qualitative analysis in the following subsection. However, the fact that all models where preprocessing was carried out performed slightly better than the baseline is a promising tendency that additional preprocessing steps may have had a positive effect.

Looking at the results of the SVM classifier, it can be stated that the classification benefits from the information offered by emojis. The results in experiments where emojis were considered in vectorization outperform the baseline. Moreover, the results suggest that the attempt of Schmid et al. (2019) to assign emojis a numerical sentiment score seems to do slightly better than the replacement approach. Schmid et al. (2019) lowercase the data in additon to lemmatizing it. With the current lemmatizer however, this seemed to have rather hurt performance.

The AdaBoost classifier here performed worse than the SVM, which is in line with findings of Scheffler et al. (2018). Again, it seemed that changing the casing before feeding it to the lemmatizer was harmful. Moreover, surprisingly, none of the preprocessing methods that were experimented with helped in improving over the baseline.

7.2 Qualitative Analysis

7.2.1 Emojis

One example, where replacing emojis with text had an impact on the BERT classifier was the following one:

(1) *@Superliebling @jouwatch Selbst mit*
@Superliebling @jouwatch Even with
der neuen, staatlichen
the new, governmental
#Abschiebeindustrie wird Kapital
#Deportation industry is capital
geschlagen. Widerlich....
being made of. Disgusting....
<U+0001F595><U+0001F3FB>
<U+0001F595><U+0001F3FB>

Example (1) contains negatively connotated words, but, without the middle finger emoji, it may even be argued if the gold tag *OFFENSE* here would be accurate. In fact, in none of the experiments without emoji replacement was it considered to be offensive, whereas the models trained on data where emojis were replaced did so. A look at the tokenization may explain here why it could not learn this from the data without emojis replaced through text. There, the unicode representation was basically split into single characters, which understandably made learning harder.

In total, the middle finger emoji occurs in 37 instances, 35 of these were annotated with the gold label *OFFENSE*. The model trained on truecased data with emoji replacement managed to catch 13

Experiments	Tweets
emoji replacement + splitting hashtags + lowercasing	10
truecasing	5
splitting hashtags + truecasing	3
baseline	2

Table 3: Number of tweets classified as offensive, out of the 13 tweets with the middle finger emoji that were classified correctly in the experiment involving emoji replacement, hashtag splitting and truecasing

of these without wrongly classifying posts that were labeled as *OTHER*. Table 3 shows how many of these 13 instances were detected by the other classifiers. This shows a clear tendency that replacing emojis may have helped in detecting some offensive tweets where the middle finger emoji occurred. In the experiment with emoji replacement, hashtag splitting and lowercasing however, also one of the two non-offensive tweets was considered offensive.

However, replacing emojis may also present some pitfalls. One such case in the results of the BERT classifier is related to the winking face emoji, which is not really associated with offensive behaviour. However, the models trained on data where emojis were replaced had a surprisingly high tendency to misclassify non-offensive emojis containing the emoji with 9 (hashtag splitting + truecasing) respectively 15 (hashtag splitting + lowercasing) instances wrongly considered to be offensive language, which suggests that they may have learned to interpret the emoji in a wrong way.

Replacing emojis with text also helped the SVM classifier in detecting offensive tweets that contain the middle finger emoji. 29 out of the 35 offensive tweets with this emoji were classified as offensive, whereas the in both the experiment with only hashtag splitting and the experiment with neither emoji replacement nor hashtag splitting, 12 of these instances were not detected. This may however be traced back to the fact that the middle finger emoji is not in the ranking of Kralj Novak et al. (2015). It only was introduced in 2014, whereas the tweets on which Kralj Novak et al. (2015) based their ranking were collected between 2013 and 2015, and thus it appears as a logical consequence that the emoji was not among the 750 most frequently used emojis in the data of Kralj Novak et al. (2015). However, it still signals, that using emojis that convey such a strong message like the middle finger emoji should absolutely be taken into consideration.

One more striking detail when observing the results of SVM experiments involving emoji replacement is a strong tendency to classify tweets containing the pig face emoji as offensive. Here, eleven tweets were considered to be offensive. However, out of the 14 tweets where the pig face emoji occurs, only three have received the gold tag *OFFENSE*. The classifier trained and tested on data where only hashtags were split but no emojis were replaced recognized the majority of the non-offensive instances correctly.

The sentiment score of the emojis according to the ranking of Kralj Novak et al. (2015) was 0.375, thus relatively neutral and probably not that decisive for classification. For the other experiments, they were replaced with the word *Schweinegesicht*, the literal German translation of *pig face*. *Schweinegesicht* is included in Hyperhero's dictionary of German slurs and thus, even tweets that do not contain actual slurs will then be marked as doing so through this replacement. Moreover, in the training data the word *Schwein*, meaning *pig* occurs quite often in offensive contexts. Fasttext makes use of character-level similarities between compound words and the words they are made of and does not represent them as completely different words (Bojanowski et al., 2017). It can thus be assumed that the representations of *Schweinegesicht* and *Schwein* in the training data are similar and that this may be another reason for the wrongly classified instances here.

For the AdaBoost classifier, similar patterns concering the pig face emoji and the middle finger emoji could be observed. Here however, only 25 tweets containing the middle finger emojis were considered to be offensive, among them there were also the two instances labeled originally as non-offensive.

Moreover, it seems that using sentiment scores for emojis in the AdaBoost experiments led to an increased amount of emojis being misclassified as

Experiments	Tweets misclassified as offensive	Correct in the baseline experiment
Only Basic Methods	98	61
Splitting Hashtags	102	64

Table 4: Misclassified tweets in two experiments with the AdaBoost classifier

offensive. Table 4 shows the number of emojis that were misclassified tn the experiment, where emojis were turned to sentiment scores without hashtag splitting and with emoji scores and hashtag splitting and how many of these wrongly classifed instances were classified correctly in the baseline experiment, In the corresponding experiments with the SVM classifier here on the other hand, only 71 tweets were misclassified as offensive, suggesting that this problem was there at least not as pronounced as it was here.

7.2.2 Hashtags

For BERT, splitting hashtags and truecasing was the best combination of preprocessing methods according to quantitative results. However, when looking at the actual numbers when such splitting was done it becomes clear that it only had a small, albeit rather positive impact. Only in 147 out of the 3031 tweets in the test set was splitting of camel-cased hashtags performed. In 66 of these tags was the classification in the end different, in 43 cases performed the model trained and tested on data with split hashtags better whereas it was wrong in the 23 other cases. In none of these 43 differently classified tweets it looked like as if the split hashtag was decisive.

However, it often led in the end to a more natural tokenization by the BERT-tokenizer. *#HambacherForst* (name of a German forest that was supposed to be cleared, followed by protests) was thus split for example correctly recognized as ##Ham ##bacher Forst whereas the tokenizer turned it in the other case into ##Ham ##bacher ##For ##st. The method of hashtag splitting however had some minor faults here. The name of the German far right party *AfD* for example was recognized as camel-cased and split and it struggled to split hashtags where one word in the hashtag was not written with a capital letter, such as *#VerhöhnungderMaueropfer*, which was split into *Verhöhnungder* and *Maueropfer*.

Similarly, also in the SVM experiments, hashtag splitting only had marginal effect. In most of the examples, the decision on whether the tweet can be considered offensive or not was the same, regardless of split hashtags. In most of the examples where were split and classified differently, the effects of the split hashtag do not seem obvious.

An exception may be a case involving the hashtag *#NazisRaus*. Here, the hashtag was accordingly split into *Nazis* and *raus* and *Nazis* was transformed into its singular form, on the other hand, if *NazisRaus* was left as it is, then the lemmatizer did not recognize it and left it as it is. It can strongly be assumed that *Nazi* is rather associated with offensive tweets and that this then made the final decision towards classifying it as offensive. However, this was the only instance in which a non-offensive tweet containing the hashtag *#NazisRaus* was considered to be offensive.

7.2.3 Capitalization

No imminent positive effects could be observed when changing the capitalization of the data before lemmatizing it in the experiments with the SVM and the AdaBoost classifier. However, truecasing seemed to have had a positive effect on the performance of the BERT classifier in example (2):

(2) **seufz und benennt die WLAN SSID*
*sigh and rename the Wi-Fi SSID
*mal wieder in ¨FICKT LEISER!üm**
once again to "FUCK QUIETER"*

In experiments, where the original casing of the data remained untouched, the tag *OTHER* was used, whereas the BERT model trained on truecased text with emojis replaced and camel-cased hashtags split tagged correctly it as offensive. Truecasing here changed the entirely capitalized *FICKT LEISER*. The tokenizer of the baseline model tokenized it with FI ##C ##K ##T L ##E ##IS ##ER, thus treating almost each capital letter as a different subwort unit. The crucial part that renders the sentence offensive was tokenized wrongly here, and the logical consequence is that it remained undetected.

The truecased sentence on the other hand was split into f ##ickt lei ##ser. Even though, this tokenization is not completely in line with the intuitively correct one (fick##t leise##r), it apparently

made it at least easier for the model to recognize the offensive language here. The fact that even the tokenization for the truecased text does not seem to be ideal is underlined by the fact that for example the model only using truecased text without getting emojis replaced or camel cased hashtags split did not manage to classify this sentence as offensive, a decision which cannot be explained by the absence of the other two preprocessing steps.

Unfortunately, the truecasing approach here presented also some deficits. The truecaser sometimes struggled with sentences that were entirely written in capslock, where it simply did not change anything. Moreover, it sometimes struggled with English words since it was trained entirely on German data and in some cases it simply did not change words written in capslock to their original capitalization due to unknown reasons. Lowercasing also helped here since also in the experiment with lowercased data, the tweet was labeled correctly.

(3) *@dr0pr0w @kinzig9 Gibt es*
@dr0pr0w @kinzig9 Are there
irgendwo mehr Einzelheiten zu der
anywhere more details about the
Veranstaltung?
event?

Lowercasing however has not only positive effects. In example (3), *Einzelheiten* was turned to *einzelheiten* and *Veranstaltung* to *veranstaltung*. A consequence of this lowercasing was that the BERT tokenizer eventually did not recognize the nouns anymore. *einzelheiten* was then split into einzel ##heiten and *veranstaltung* resulted in veranst ##altung. For the truecased data, where the original capitalization was retained, the tokenizer recognized both nouns correctly without problems.

8 Conclusion

Summing up the results for BERT, it can be said that replacing emojis with text had positive and negative effects, splitting hashtags had minor positive effects and, given the current BERT tokenizer, truecasing was beneficial and may be preferred over lowercasing or retaining the actual capitalization but a different truecaser than the present one may be picked. For SVM, the approach of Schmid et al. (2019) outperformed replacing emojis and neither splitting camel-cased hashtags, nor changing the capitalization helped here. Regarding the AdaBoost classifier with decision trees, none of the methods had an impact. More exploration could be beneficial here but the results in combination with Scheffler et al. (2018) and the general, sparse occurrence of this classifier type in shared tasks may also raise doubts if this classifier is even a good choice for the detection of offensive language.

If a BERT classifier relies on subword units in tokenization and has not been exposed to emojis in pretraining, it can be said that replacing emojis with text seems to be a good option for learning the relevance of some emojis. However, special attention needs to be given to the possibility of introducing some kind of bias for rather neutral emojis. In further experiments it would maybe be interesting to focus only the subset of emojis that have the biggest emotional impact in a tweet, such as the middle finger emoji, to make use of the sentimental content of such emojis on the one hand and on the other hand avoid misinterpretations of emojis that would be of low relevance for the detection of offensive language in the first place.

For the SVM, the replacement method did also lead to some wrongly learned interpretations of neutral emojis, which was a factor that let to better results through sentiment scores for emojis. The number of emojis offered is ever-growing. The case of the missing middle finger emoji underlines that, when using resources such as emoji rankings, then these resources need to be as up-to-date as possible, so that potentially important emojis for sentiment analysis and detecting offensive are not left out.

Splitting hashtag had on this specific dataset a rather minor influence but led in some cases to a more natural tokenization and thus may rather be recommended for BERT when using a tokenizer relying on subwords . Truecasing presents big potential given the anomalies in capitalization for example. However, a truecaser, that is better tailored to the challenges given through social media data may be needed here to use the full potential of that technique.

References

Giulio Angiani, Laura Ferrari, Tomaso Fontanini, Paolo Fornacciari, Eleonora Iotti, Federico Magliani, and Stefano Manicardi. 2016. A

comparison between preprocessing techniques for sentiment analysis in twitter. In *Proceedings of the 2nd International Workshop on Knowledge Discovery on the WEB, KDWeb 2016, Cagliari, Italy, September 8-10, 2016*, volume 1748 of *CEUR Workshop Proceedings*, Cagliari, Italy. CEUR-WS.org.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jason Brownlee. 2020. How to develop an AdaBoost ensemble in python.

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 the GermEval Workshop.*, pages "27–32", Vienna, Austria. Verlag der Österreichischen Akademie der Wissenschaften.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yoav Freund and Robert E Schapire. 1996. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Dirk von Grünigen, Ralf Grubenmann, Fernando Benites, Pius von Däniken, and Mark Cieliebak. 2018. spMMMP at GermEval 2018 shared task: Classification of offensive content in tweets using convolutional neural networks and gated recurrent units. In *Proceedings of the GermEval Workshop.*, pages "130–137", Vienna, Austria. Verlag der Österreichischen Akademie der Wissenschaften.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. 2008. A practical guide to support vector classication.

Pratheek Joshi. 2020. Transfer learning for NLP: Fine-tuning BERT for text classification.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *PLOS ONE*, 10(12):1–22.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. tRuEcasIng. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 152–159, Sapporo, Japan. Association for Computational Linguistics.

Chris McCormick and Nick Ryan. 2020. BERT fine-tuning tutorial with Pytorch.

Andrei Paraschiv and Dumitru-Clementin Cercel. 2019. UPB at GermEval-2019 task 2: BERT-based offensive language classification of german tweets. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)*, pages 398–404, Erlangen, Germany. German Society for Computational Linguistics & Language Technology.

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, Berlin. Association for Computational Linguistics.

Robert Remus, Uwe Quasthoff, and Gerhard Heyer. 2010. SentiWS - a publicly available German-language resource for sentiment analysis. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Julian Risch, Anke Stoll, Marc Ziegele, and Ralf Krestel. 2019. hpiDEDIS at GermEval 2019: Offensive language identification using a german BERT model. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)*, pages 405–410, Erlangen, Germany. German Society for Computational Linguistics & Language Technology.

Björn Ross, Michael Rist, Guillermo Carbonell, Ben Cabrera, Nils Kurowsky, and Michael Wojatzki. 2016. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. In *Proceedings of NLP4CMC III: 3rd Workshop on Natural Language Processing for Computer-Mediated Communication*, pages 6–9.

Tatjana Scheffler, Erik Haegert, Santichai Pornavalai, and Mino Lee Sasse. 2018. Feature explorations for hate speech classification. In *Proceedings of the GermEval Workshop.*, pages 51–57, Vienna, Austria. Verlag der Österreichischen Akademie der Wissenschaften.

Florian Schmid, Justine Thielemann, Anna Mantwill, Jian Xi, Dirk Labudde, and Michael Spranger. 2019. Fosil - offensive language classification of german tweets combining svms and deep learning techniques. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)*, pages 382–386, Erlangen, Germany. German Society for Computational Linguistics & Language Technology.

Dominik Stammbach, Azin Zahraei, Polina Stadnikova, and Dietrich Klakow. 2018. Offensive language detection with neural networks for Germeval task 2018. In *Proceedings of the GermEval Workshop.*, pages "58–62", Vienna, Austria. Verlag der Österreichischen Akademie der Wissenschaften.

Julia Maria Struß, Melanie Siegel, Josep Ruppenhofer, Michael Wiegand, and Manfred Klenner. 2019. Overview of GermEval task 2, 2019 shared task on the identification of offensive language. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)*, pages 354–365, Erlangen, Germany. German Society for Computational Linguistics & Language Technology.

Michael Wiegand, Melanie Siegel, and Joseph Ruppenhofer. 2018. Overview of the GermEval 2018 shared task on the identification of offensive language. In *Proceedings of the GermEval Workshop.*, pages "1–10", Vienna, Austria. Verlag der Österreichischen Akademie der Wissenschaften.

Frederike Zufall, Tobias Horsmann, and Torsten Zesch. 2019. From legal to technical concept: Towards an automated classification of German political Twitter postings as criminal offenses. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1337–1347, Minneapolis, Minnesota. Association for Computational Linguistics.

9 Appendix

Epochs	2
Batch Size	32
Maximum Length	150
Learning Rate	$2e^5$
Optimizer	Adam
Loss Function	Cross-Entropy Loss

Table 5: Hyperparameters for finetuning BERT

C	0.1	1	10	100

Table 6: Values for the grid search for hyperparameter tuning in the SVM experiments

Number of Iterators	10	50	100	500	
Learning Rate	0.0001	0.001	0.01	0.1	1

Table 7: Values for the grid search for hyperparameter tuning for the AdaBoost experiments