Examining the Effects of Preprocessing on the Detection of Offensive Language in German Tweets

Sebastian Reimann
Uppsala University
Department of Linguistics
reimann.sebastian9483@gmail.com

Daniel Dakota
Uppsala University
Department of Linguistics
ddakota@lingfil.uu.se

Abstract

Preprocessing is essential for creating more effective features and reducing noise in classification, especially in user-generated data (e.g. Twitter). How each individual preprocessing decision changes an individual classifier's behavior is not universal. We perform a series of ablation experiments in which we examine how classifiers behave based on individual preprocessing steps when detecting offensive language in German. While preprocessing decisions for traditional classifier approaches are not as varied, we note that pre-trained BERT models are far more sensitive to each decision and do not behave identically to each other. We find that the cause of much variation between classifiers has to do with the interactions specific preprocessing steps have on the overall vocabulary distributions, and, in the case of BERT models, how this interacts with the WordPiece tokenization.

1 Introduction

The task of abusive language detection has become increasingly popular for a variety of languages (Zampieri et al., 2019; Basile et al., 2019; Al-Khalifa et al., 2020). German specifically has had two shared tasks on the topic, one in 2018 (Wiegand et al., 2018) and a second in 2019 (Struß et al., 2019).

Not only is offensive language detection somewhat subjective in nature, particularly in the need for contextual requirements, but is often examined through user generated mediums, creating another layer of complexity to successfully identify possible abusive language. Often, in order to create more useful features out of the text for the classifier, we must first treat the text to reduce the noise. While for standard feature generation via count vectors the impact is far more obvious (e.g. reduction of feature space), even when we feed a dense vector representations to a classifier (e.g. a sentence embedding), that embedding still represents the textual representation, simply in an alternative way. Thus, it too is influenced by individual alterations to the text. With languages that show large variation in dialect preferences and orthographic representations, this has been shown to be particularly important (Husain, 2020).

Twitter has proven to be a typical source for not only research on offensive language, but also necessitating additional preprocessing approaches given its different style of communication and lexicon. In this work we look to perform a set of ablation experiments in which we evaluate how different preprocessing techniques impact classifier behavior over three different approaches to classification when detecting offensive language in German Twitter. We seek to answer the following questions:

1. How do different preprocessing techniques influence performance across different classifiers?

2. Can we identify features within different preproccesing techniques that can help explain specific classifier behaviors?

2 Related Work

2.1 Data

Ross et al. (2016) introduced a Twitter corpus of offensive language detection, examining the 2015 refugee crisis. They predominantly focused on user's perceptions of hate speech and the reliability of annotations. They found that agreement among annotators was relatively low and that also the opinions of users asked in a survey diverge greatly and thus stress the necessity of specific guidelines. However, even with such guidelines, annotators can still show large differences (Nobata et al., 2016).

Task 2 of the GermEval 2018 shared task (Wiegand et al., 2018) focused on detecting offensive and non-offensive Tweets and was further examined in GermEval 2019 (Struß et al., 2019).

A different approach was taken by Zufall et al. (2019) who instead label offensive Tweets based on whether they may be punishable by law or not. This decision is based on two criteria: the type of target and the type of offense. A Tweet may be punishable if it is targeted at either a living individual or a specific group of people, and if it expresses either a wrong factual claim, abusive insults, or abusive criticism.

2.2 Classifiers

Abusive language detection in German has shown a great deal of variation across classifiers and feature thresholds (Steimel et al., 2019). In the 2018 shared tasks, SVMs were a popular choice (Wiegand et al., 2018), achieving effective results. Popular features include pre-trained word embeddings, mostly either fastText (Bojanowski et al., 2017) or word2vec (Mikolov et al., 2013), and lexical features based on polarity lexicons or lexicons on offensive language and effective results were achieved with only a few hundred features (De Smedt and Jaki, 2018). Other classifiers included standard Decision Trees or Boosted Classifiers, but tended to yield slightly worse performance (Scheffler et al., 2018).

The most effective approaches tended to use ensemble classifiers: CNNs with logit averaging (von Grünigen et al., 2018), a combination of RNNs and CNNs (Stammbach et al., 2018), or combination of Random Forest classifiers (Montani and Schüller, 2018).

With the introduction of BERT (Devlin et al., 2019), the 2019 shared task saw a different trend, with many participants submitting fine-tuned models (Struß et al., 2019). Paraschiv and Cercel (2019) pre-trained a BERT model on German Twitter data, obtaining the best reported macro F-score of 76.95. Other approaches included fine-tuning an ensemble of BERT models trained on different German textual sources (Risch et al., 2019).

SVMs continued to be a popular choice however, with some systems achieving results almost equal to BERT-based approaches by using word embeddings pre-trained on German Tweets and lexical features (Schmid et al., 2019).

2.3 Preprocessing

Angiani et al. (2016) experimented with the preprocessing methods of replacement of emoticons with a text representation, replacing negation contractions such as *don't* with *do not*, detection of spelling errors, stemming, and removal of stopwords for general sentiment analysis on Twitter data. Using a Naive Bayes classifier to classify whether the sentiment was positive, neutral or negative, most techniques yielded slight improvements over the baseline with little preprocessing.

While Risch et al. (2019) had a minimalistic approach to preprocessing and only normalized user names, Paraschiv and Cercel (2019), whose contribution performed best in the GermEval 2019 shared task, made use of a wide range of preprocessing methods when fine-tuning BERT. They replaced emojis with spelled-out representations; 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 however do not explicitly state how much this contributed to achieving a higher performance.

Schmid et al. (2019) lowercased and lemmatized words, while also removing the #-character of the hashtag and stop words when creating features for their SVM. Sentiment scores were also obtained for emojis through the sentiment ranking for emojis by Kralj Novak et al. (2015) and added to the sentiment scores obtained trough SentiWS (Remus et al., 2010) for all words in the sentence. Both scores were treated as separate features and ranged range from -1 to 1. Scheffler et al. (2018) also lemmatized and removed stop words, but did not explicitly state their treatment of hashtags and capitalization for their experiments involving SVMs, decision tree, and boosted classifiers. Moreover, they did not include emojis when modeling a sentiment score as one of their features.

3 Methodology

3.1 Data

For all experiments, we use the the dataset from the GermEval 2019 Task 2 (Struß et al., 2019). Tweets were sampled from a range of political spectrums and labeled as either OFFENSE or OTHER for the binary classification task (see Table 1 for data splits).

	OFFENSE	OTHER	Total
Train	1287	2709	3996
Test	970	2061	3031

Table 1: Train and Test Data Splits

3.2 Preprocessing

Base Methods Lemmatization is a relatively common preprocessing step applied in the shared task of Wiegand et al. (2018), examples include Scheffler et al. (2018) and Schmid et al. (2019), on which we base our experimental setup for our SVM and AdaBoost classifiers. Consequently, we lemmatize all words[1] for our AdaBoost and SVM experiments. A second base step, carried out in all experiments, including those when fine-tuning BERT for classification, is replacing user names with the token USER.

Emojis We try the approaches in the contributions to the GermEval 2019 shared task by both Paraschiv and Cercel (2019), who replaced emojis with textual representations, and Risch et al. (2019), who did not adress emojis in preprocessing, respectively. Additionally, we 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). Unlike Paraschiv and Cercel (2019) however, who use English descriptions of emojis, we translate the descriptions into German.[2]

Hashtags We remove the #-character at the beginning of each hashtag. Additionally, we try splitting camel-cased hashtags as done by Paraschiv and Cercel (2019).

Capitalization We perform three strategies: retaining the original capitalization, lowercasing the entire text, and truecasing. Truecasing is "the process of restoring case information to raw text" (Lita et al., 2003). For German, this is beneficial since it keeps orthographic characteristics (e.g. all nouns are capitalized) but removes situational one (e.g. words capitalized only because they begin a sentence). Additionally, Risch et al. (2019) point out that in their experiments, for words written in Caps Lock, each letter is frequently recognized as a separate token. Additionally, truecasing has been shown to be useful for NLP on noisy data (Lita et al., 2003).

Truecasing the test and training data is performed by using the truecasing scripts from the Moses system (Koehn et al., 2007), which are normally used for statistical machine translation. We create a truecasing model by training on a large, cleaned, preprocessed German Wikipedia Text Corpus.[3] We use the SoMaJo tokenizer for German social media data (Proisl and Uhrig, 2016) to tokenize the Twitter data.

3.3 Classifiers

All hyperparameter optimization is performed using a 5-fold cross validation and results for all experiments are reported using macro-averaged F scores since the dataset is imbalanced and we wish to give equal weight to both the minority and majority classes.

SVM The features for the SVM (Boser et al., 1992) are similar to the ones used in the second system of Schmid et al. (2019), where pre-trained fastText vectors (Bojanowski et al., 2017) were used to create Tweet level vector representations. We initially experimented with a set of fastText vectors pre-trained on a smaller set of Twitter data as well as with different dimensions, but results were poor relative to other pre-trained fastText embeddings. We ultimately settled on on the default 300 dimensional fastText German embeddings (Grave et al., 2018), trained on the German CommonCrawl and Wikpedia, as they yielded the most stable performance.

We also add a binary feature which signals if a Tweet contains one or more German slurs from the slur dictionary of Hyperhero,[4] similar to that of Scheffler et al. (2018) and Schmid et al. (2019), although we do not manually create a lexicon of offensive terms as performed by the latter. The vectors plus the binary feature and the sentiment scores are concatenated and fed to the SVM.

We use a linear kernel and in order to reduce attributes with greater numerical ranges from dominating, we perform feature scaling (Hsu et al., 2008), and only hyperparameterize for the regularization parameter C.[5]

[1] We use spaCy.

[2] Translations are done using Google Translate.

[3] https://github.com/t-systems-on-site-services-gmbh/german-wikipedia-text-corpus

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

[5] We only optimize C for 0.1, 1. 10 and 100

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

Table 2: Values for the grid search for hyperparameter tuning for the AdaBoost experiments

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

Table 3: Hyperparameters for fine-tuning BERT

AdaBoost Additionally, we experiment with AdaBoost (Freund and Schapire, 1996) as it was used by Scheffler et al. (2018). AdaBoost is a boosting technique that will combine multiple weak classifiers (in our cases tree stumps) by giving more weight to incorrectly classified training instances, importantly without large weight reduction to the correctly classified instances. We also hyperparameterize using grid search following values taken from Brownlee (2020).

BERT We use both the *bert-base-german-cased*[6] and the *dbmdz/bert-base-german-cased*[7], referring to them as DeepAI and dbmdz respectively hereafter. The DeepAI model was pre-trained on a German Wikipedia dump, the OpenLegal dump, a large data collection involving German court decisions, and 3.6 GB of news articles. This data was cleaned and segmented into sentences by the spaCy library. The dbmdz model was pre-trained on a collection of Wikipedia, the EU Bookshop corpus, Open Subtitles, CommonCrawl, ParaCrawl, and NewsCrawl. Both models make use of a WordPiece vocabulary which was created through the WordPiece tokenizer (Wu et al., 2016). As neither are pre-trained on any particular social media text, we assume that they are not well equipped to handle more common social media orthographic standards, such as hashtags and emojis. Following Risch et al. (2019) we fine-tune for two epochs using a batch size of 32 (see Table 3 for all hyperparameters).

4 Results

First we must note that we ran some BERT models with different initial seeds and noted instabilities in performance. Given this, results should not be viewed as entirely explained by the different preprocessing choices, rather and indication of the volatility of the models in interaction with preprocessing. We are more interested in highlighting the interaction and variation across BERT models and preprocessing than determining an optimal solution. For this reason, we only report scores using the default seed in order to allow a better analysis (see Section 5) in terms of linking observed differences to specific preprocessing choices, and interaction with both the WordPiece tokenization and the vocabulary distribution.

We first begin by establishing baselines for each classifier, in which minimal preprocessing is performed. For BERT, we only replaced user names with a USER token (baseline BERT in Table 4). For the SVM and AdaBoost we perform the former, but since we experiment with two different ways of treating emojis (replacement vs. inclusion in sentiment scores) and want to compare the results against an experiment where emojis are not taken into account at all, we additionally remove emojis in our baseline here and lemmatize all words (baseline SVM/AdaBoost in Table 4).

In Table 4 we present F-scores for our ablation experiments. We can see that AdaBoost tends to exhibit a degradation in performance in respect to performing only base preprocessing operations when any additional preprocessing techniques are applied. The only exception tends to be in experiments that have a combination of splitting hashtags and truecasing. This may simply be due a reduction of overall features, but it is not inherently clear what is causing the degradation.

The SVM tends to outperform AdaBoost overall, which is in line with Scheffler et al. (2018) though only in a couple of instances, shows any noticeable improvements. Lowercasing, as performed in Schmid et al. (2019), leads here to the biggest drop in performance. Surprisingly, the classification overall does not profit from the information offered by emojis, as both the experiment with emoji replacement as well as the experiment with only basic preprocessing and without emoji removal do not perform above the baseline. This also holds true for emoji replacement in combination with splitting hashtags since the performance here is slightly worse than for only splitting hashtags. Interestingly, we see that only truecasing the data yields the best performing model, and that, similar to AdaBoost, a

[6] https://deepset.ai/german-bert
[7] https://github.com/dbmdz/berts

Experiment	AdaBoost	SVM	DeepAI	dbmdz
emojis removed (baseline SVM/AdaBoost)	66.74	66.81	72.74	69.04
only basic methods (baseline BERT)	66.90	66.78	71.31	71.93
replacing emojis	66.39	66.23	71.34	72.60
splitting hashtags	66.20	66.90	68.62	**74.49**
only truecasing	66.09	**67.64**	73.25	69.13
replacing emojis + splitting hashtags + lowercasing	64.62	64.33	70.04	73.32
splitting hashtags + truecasing	**67.65**	66.74	**73.62**	71.85
replacing emojis + splitting hashtags	66.62	66.78	73.10	70.88
replacing emojis + splitting hashtags + truecasing	67.31	67.23	72.68	71.33

Table 4: F1-macro Scores for All Classifiers

combination of truecasing with emoji replacement and splitting hashtags led to improvments over the baseline as well, although here, splitting hashtags and truecasing without emoji replacement led to a slight decrease.

One striking difference is the performance of DeepAI vs dbmdz and their behaviors not only in respect to the preprocessing techniques, but also to each other. Firstly, we see that the baselines are slightly different and all applied preprocessing techniques benefit dbmdz, even if minimally, compared to DeepAI, where we some techniques result in worse performance relative to the baseline. Additionally, we can see that in some cases, the models actually have opposite behaviors. For example, simply splitting hashtags resulted in the best performance for dbmdz, yet was the worst performance for DeepAI. A counter example is only truecasing which yielded minimal performance gains for dbmdz but produced the second best results for DeepAI.

5 Analysis

While results for AdaBoost and the SVM do show some variation, the DeepAI and dbmdz exhibit much more noticeable changes. For this reason, we choose to examine only these two model in terms of how the minority and majority classes are behaving in order to try and glean insight into the underlying causes. Table 5 shows the precision and recall of classes for the these models. We can clearly see a great deal of volatility on the minority (OFFENSE) class, particularly on recall. This could again be because of a general instability but it may also suggest that, querying Tweets deemed offensive is far more sensitive to the preprocessing methods than labeling them correctly when the models are being fine-tuned. We perform a more in-depth analysis into possible reasons behind the variations between all classifiers and present the findings below.

5.1 Emojis

Without emoji replacement, the WordPiece tokenization used by the BERT models splits the unicode representations into single letters or chunks of two or three numbers. It can be assumed that these models cannot effectively make use of such representations. Replacing emojis with text on the other hand presents a way to retain the meaning of the emoji in the text, which seems to have helped DeepAI in particular in finding offensive tweets as all experiments with emoji replacement constantly outperform its baseline with respect to recall for the OFFENSE class.

One example is the case of the middle finger emoji, for which emoji replacement helps detect offensive Tweets. It occurs in 37 Tweets, 35 of which have the gold label OFFENSE. The DeepAI model trained on truecased data with emoji replacement managed to correctly label 13 of these without wrongly classifying posts that were labeled as OTHER. Table 6 shows how many of these 13 instances were detected DeepAI when different preprocessing methods were applied. This suggests that replacing emojis improves detecting offensive Tweets when the middle finger emoji is present. In the experiment with emoji replacement + hashtag splitting + lowercasing however, one of the two non-offensive Tweets was wrongly classified as offensive.

Replacing emojis also present some pitfalls. One such case for the DeepAI BERT classifier is related to the winking face emoji, which is not inherently associated with offensive behavior. However, the models trained on data where emojis were replaced

	DeepAI				dbmdz			
	OFFENSE		OTHER		OFFENSE		OTHER	
Experiment	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
emojis removed (baseline SVM/AdaBoost)	66.27	57.94	81.31	83.65	**84.09**	38.14	76.84	**96.60**
only basic methods (baseline BERT)	**78.62**	44.74	78.38	**94.27**	80.15	45.36	78.65	94.71
replacing emojis	58.51	**66.60**	**83.19**	77.78	73.36	50.82	79.78	91.31
splitting hashtags	75.78	40.31	76.98	93.93	72.27	**56.70**	**81.50**	89.76
only truecasing	69.49	55.88	80.99	88.45	82.31	38.87	76.95	96.07
replacing emojis + splitting hashtags + lowercasing	67.67	48.56	78.63	89.08	73.73	52.37	80.27	91.22
splitting hashtags + truecasing	77.24	50.72	80.03	92.96	77.49	46.49	78.81	93.64
replacing emojis + splitting hashtags	67.55	57.53	81.32	87.00	75.91	45.15	78.32	93.26
replacing emojis + splitting hashtags + truecasing	62.36	63.71	82.75	81.90	77.23	45.46	78.50	93.69

Table 5: Class Results for BERT DeepAI and dbmdz

Preprocessing	# of Tweets
emoji + splitting hashtags + lowercasing	10
truecasing	5
splitting hashtags + truecasing	3
baseline	2

Table 6: Number of Tweets classified as offensive by DeepAI BERT, out of the 13 Tweets with the middle finger emoji that were classified correctly in the experiment involving emoji replacement, hashtag splitting and truecasing

had a surprisingly high tendency to misclassify non-offensive Tweets containing this emoji with 9 (hashtag splitting + truecasing) and 15 (hashtag splitting + lowercasing) instances respectively wrongly considered to be offensive language. This suggests that they may have learned to associate the emoji in an unintended manner, especially as, in most cases, the emoji occurs in contexts without other elements that may potentially cause the classification as offensive.

Emoji replacing also helps the SVM classifier in detecting offensive Tweets that contain the middle finger emoji as in both the experiment with emoji replacement only and emoji replacement in combination with hashtag splitting and truecasing, 29 out of the 35 offensive Tweets with this emoji were classified as offensive. This in contrast to the experiment with only hashtag splitting and the experiment with neither emoji replacement nor hashtag splitting, in which 12 of these instances were not classified correctly. 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). Their ranking of most frequent emojis was determined from Tweets collected between 2013 and 2015, while the middle finger emoji was only introduced in 2014. Given this, it is not surprising that the emoji was not in the top 750 most frequently used. This demonstrates a limitation of even newer external social media lexicons, as the medium of communication is rapidly evolving and even emojis can be time sensitive with the introduction of new ones, the discontinuation of older ones, or simply a decrease in usage.

Another interesting observation from the SVM experiments that included emoji replacement is the classification of Tweets containing the pig face emoji. While 14 Tweets contain the pig face emoji, only three have the gold label of OFFENSE. However, the SVM strongly prefers classifying Tweets with this emoji as offensive, as it does so in 11 cases. 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.

In the experiments with emoji replacement, this emoji was replaced with the word *Schweinegesicht* ("pig face"), which is included in Hyperhero's dictionary of German slurs. This results in Tweets with this specific replacement being marked as containing slurs, even if they are not labeled as OFFENSIVE. On the other hand, the sentiment score of the pig face emoji according to the ranking of Kralj Novak et al. (2015) was 0.375 and thus relatively neutral, which gives credence to the idea that emoji replacement was decisive for the wrong classifications.

Moreover, in the training data the word *Schwein* ("pig") occurs quite often in offensive Tweets. Given the use of character-level embeddings via fastText, there may be similarity between compound words that contain one or more subwords that may deemed offensive on their own, but not necessarily within the compound itself. Thus, it may be that the representations of *Schweinegesicht* and *Schwein* in the training data are inherently similar enough in vector space and are thus influencing their wrongly classified instances here.

	Basic	S-HT
Misclassified by Adaboost	95	80
Of which correct in Baseline	75	61
Misclassified by SVM	45	46
Of which correct in Baseline	4	5

Table 7: Tweets containing emoijs misclassified as offensive in two experiments with the AdaBoost classifier and the SVM classifier with Only Basic Methods (Basic) and Splitting Hashtags (S-HT)

For the AdaBoost classifier, similar patterns concerning the pig face emoji and the middle finger emoji are observed. Moreover, it seems that using sentiment scores for emojis in the AdaBoost experiments led to a general increase in the amount of Tweets with emojis being misclassified as offensive. Table 7 compared the misclassified examples that contain emojis in experiments, where emojis were turned to sentiment scores without hashtag splitting and with emoji scores and hashtag splitting, for both SVM and AdaBoost. It also shows how many of these wrongly classified instances were classified correctly in the respective baseline experiment. The numbers suggest that while the problem occurs also for the SVM classifier, it is not as pronounced and the differences between the two experiments under observation and the baseline are much less drastic.

5.2 Hashtags

For DeepAI, splitting hashtags and truecasing produced the best model. However, upon closer inspection, the impact of splitting hashtags does not seem to be as pronounced, albeit still positive. In the test set, only 147 Tweets require splitting of camel-cased hashtags, of which only 66 instances resulted in a different classification upon splitting (43 correctly classified, 23 incorrectly).

Instead, hashtag splitting results in a more natural tokenization by the WordPiece tokenizer used by each BERT model given its source data. For example, #HambacherForst (name of a German forest that was supposed to be cleared) is split correctly by DeepAI into ##Ham ##bacher Forst whereas when hashtag splitting is not performed, the tokenization is ##Ham ##bacher ##For ##st. The method of hashtag splitting however can produce errors. The abbreviation of the German party *AfD* for example is recognized as camel-cased and split into *Af* and *D* as separate tokens. Additionally, hashtag splitting is not always successful when one word in the hashtag is not written with a capital letter, such as #VerhöhnungderMaueropfer ("mockery of the victims of the Berlin Wall"), which is split into *Verhöhnungder*, the German word for *mockery* plus the article (*der*), and *Maueropfer*.

Splitting hashtags leads in both cases to a lower number of subword tokens in terms of both the overall number of tokens produced as well as the number of individual token types present after a WordPiece tokenization is applied (see section 5.4 for more discussion on vocabulary distributions). This suggests that the WordPiece tokenizer for both models struggles in splitting hashtags into representative subwords, if hashtag splitting is not performed. This decrease in subword tokens is slightly higher for the dbmdz model, suggesting that without hashtag splitting, the the dbmdz WordPiece tokenizer creates more unwanted splits and thus, that the necessity for hashtag splitting may be greater for dbmdz than for the DeepAI.

Similarly, 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 where the hashtag was split hashtags, as no clear pattern emerged when examining Tweets that were classified differently.

5.3 Capitalization

No obvious positive effects could be observed when only changing the capitalization of the data before lemmatizing it in the experiments with AdaBoost, but a slightly positive effect is noted for the SVM. Indeed, a comparsion between the experiment involving base preprocessing and truecasing, shows that there are 27 offensive examples where truecasing changed the capitalization, and which were detected in the former but not in the latter setting. However, none of the truecased words in these examples seemed to be obviously decisive for the correct classification. Truecasing also seemed to have had a positive effect on the performance of DeepAI as seen in example (1):

(1) *seufz und bennent die WLAN SSID mal wieder in "FICKT LEISER!üm*"

sight and rename the WLAN SSID once again to "FUCK QUIETER!"

In experiments, where the original casing of the data remained untouched, the tag OTHER was used, whereas DeepAI trained on truecased text with

emojis replaced and camel-cased hashtags split correctly labeled it as offensive.

The tokenizer of the baseline model tokenized it FI ##C ##K ##T L##E ##IS ##ER, thus treating almost each capital letter as a different subword 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 seemingly made it easier for the model to recognize the offensive language.

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 setting using only truecased text without replacing emojis or splitting camel-cased hashtags did not manage to classify this sentence as offensive, a decision which cannot be explained by the absence of the other two preprocessing steps.

However, the truecasing approach sometimes struggles with sentences that were entirely written in Caps Lock, where it simply did not change anything, as well as with English words since it was trained entirely on German data.

While lowercasing helped in the case of example (1) since it was also lowercased and the Tweet was labeled correctly, this is not always the case and at times, lowercasing is not helpful. In example (2), *Einzelheiten* was turned to *einzelheiten* and *Veranstaltung* to *veranstaltung*. A consequence of lowercasing was that the DeepAI struggled to recognize the nouns. Lowercased *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.

(2) @dr0pr0w @kinzig9 Gibt es irgendwo mehr Einzelheiten yu der Veranstaltung?

@dr0pr0w @kinzig9 are there more details anywhere about the event?

Truecasing seemingly had a higher impact on the performance for DeepAI than dbmdz. A reason for this may lie in the way the respective WordPieces tokenizers for each model splits up words into subword units. In cases where non-capitalized words are written with sentence initialized capitalization, DeepAI splits these words up in an unnatural manner. The interrogative pronoun *Wozu* ("for what") at the beginning of a sentence is split into *Wo* and *zu*, the adverb *Gestern* ("yesterday") is split into *Gest* and *ern* and the verb *Geht* is split into *Geh* and *t*. When they are converted into their original, lowercased form, DeepAI does not split the words, while dbmdz, on the other hand, manages to recognize them correctly as one word without needing extra truecasing.

5.4 Vocabulary Distributions

We perform a high-level analysis on both the fastText and WordPiece coverage of the training data. For fastText, coverage ranges between 88.01-90.26% in terms of overall token coverage in the training data, with token types ranging from 69.72-71.40%, with the exception being the preprocessing setting of replacing emojis+splitting hashtags+lowercasing (which also had the lowest overall token coverage) yielding a type coverage of only 59.23%.

For DeepAI, a similar trend is seen with its WordPiece coverage. This specific setting shows over 3,000 fewer subtoken types after tokenization even though it produces overall more subtokens. These distributions may also explain the emojis+splitting hashtags+lowercasing results seen in Table 4, as this setting yields the worst performance for AdaBoost, the SVM, and DeepAI. It is also clear that while the other preprocessing distributions may yield similar coverage, the individual token distributions are not the same. These effects are evident in Table 5 in the high volatility of reported recall metrics for the OFFENSE class.

Similary however, the WordPiece tokenization by dbmdz yields a far lower number of token types in the emojis+splitting hashtags+lowercasing setting, and produces over 16000 more tokens, but does not show the same degradation in performance. Interestingly, dbmdz contains anywhere between 700-1000 more found token types in the training than its DeepAI counterpart for each preprocessing setting, and an average of 2-3% more overall total token coverage ($\approx$ 97% to 94% respectively). This just further emphasizes that the distributional coverage is not easily disentangled from the individual impact the combined feature sets (or even a single feature) have on classification, since the subtoken representations and distributions are not identical.

6 Conclusion

We have performed an in-depth analysis on the effects that preprocessing has on the performance of different classifiers on the detection of abusive language in German Tweets. While the fact that fine-tuned BERT models outperform more traditional machine learning approaches is not surprising, they however appear to be extremely sensitive to preprocessing decisions and different models behave somewhat unexpectedly, particularly when contrasted to each other. Standard preprocessing techniques, such as hashtag splitting, yield two very different behaviors from the the models, which, on the surface, is not intuitive.

Our analysis shows that the underlying word representations created by the various preprocessing techniques interact with the vocabulary coverage of fastText and the WordPiece tokenizer and plays a crucial role. Each individual preprocessing step is altering these distributions within the data which then derives slightly different sentence representations when generating sentence level embedding representations, the effects of which are not always clearly understood on the surface level. This is highlighted when some preprocessing steps, which would seem intuitively helpful, ultimately yield a degradation in performance.

Future areas of research include examining model stability with respect to preprocessing, and how preprocessing interacts with models that have been pre-trained on Twitter data with an updated WordPiece tokenizer. A deeper look at then identifying specific (sub)tokens that carry more decision making power through techniques, such as saliency (Li et al., 2016), would be of valuable insight.

References

Hend Al-Khalifa, Walid Magdy, Kareem Darwish, Tamer Elsayed, and Hamdy Mubarak, editors. 2020. *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*. European Language Resource Association, Marseille, France.

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*, Cagliari, Italy.

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*, pages 54–63.

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.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA. Association for Computing Machinery.

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 GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*, pages "27–32", Vienna, Austria.

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*, pages 4171–4186, Minneapolis, Minnesota.

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)*, pages 3483–3487.

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 GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*, pages "130–137", Vienna, Austria.

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

Fatemah Husain. 2020. OSACT4 shared task on offensive language detection: Intensive preprocessing-based approach. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language*

Detection, pages 53–60, Marseille, France. European Language Resource Association.

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.

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

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California.

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.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Joaquín Padilla Montani and Peter Schüller. 2018. TUWienKBS at GermEval 2018: German abusive tweet detection. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*, pages 45–50, Vienna, Austria.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153, Montréal, Canada.

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.

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.

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 GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*, pages 51–57, Vienna, Austria.

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.

Dominik Stammbach, Azin Zahraei, Polina Stadnikova, and Dietrich Klakow. 2018. Offensive language detection with neural networks for Germeval task 2018. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*, pages "58–62", Vienna, Austria.

Kenneth Steimel, Daniel Dakota, Yue Chen, and Sandra Kübler. 2019. Investigating multilingual abusive language detection: A cautionary tale. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1151–1160, Varna, Bulgaria. INCOMA Ltd.

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.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

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). In *Proceedings of the 13th International Workshop on Semantic Evaluation*.

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*, pages 1337–1347, Minneapolis, Minnesota.