Proceedings of the GermEval 2018 Workshop

14th Conference on Natural Language Processing

KONVENS 2018

Austrian Academy of Sciences, Vienna
September 21, 2018

Edited by

Josef Ruppenhofer
Melanie Siegel
Michael Wiegand

Contents

Preface

Offensive language in social media is a problem currently widely discussed. Researchers in language technology have started to work on solutions to support the classification of offensive posts. We present the pilot edition of the GermEval Shared Task on the Identification of Offensive Language. This shared task deals with the classification of German tweets from Twitter. GermEval 2018 is the fourth workshop in a series of shared tasks on German processing. These shared tasks have been run informally by self-organized groups of interested researchers and were endorsed by special interest groups within the German Society for Computational Linguistics (GSCL). The workshop was co-located with the Conference on Natural Language Processing (Konvens) 2018 in Vienna. The results indicate that the domain of offensive language in social media offers challenging tasks. There were two tasks, a coarse classification and a more fine-grained classification of tweets. We received 76 submissions from 20 groups. The results and the full dataset can be found at the task website at `https://projects.fzai.h-da.de/iggsa/`.

We are grateful to the large number of participants whose enthusiastic participation made GermEval 2018 a success. We would like to thank Markus Meyer for maintaining home page and mailing lists and supporting the evaluation process. We also thank the Konvens 2018 conference organizers for their support.

Vienna, September 2018

The organizing committee

Organizers:

Josef Ruppenhofer (Institute for German Language, Mannheim)
Melanie Siegel (Darmstadt University of Applied Sciences)
Michael Wiegand (Saarland University)

group id	authors	affiliation	paper title
CLuzh	Klenner	University of Zurich	Offensive Language without Offensive Words (OLWOW)
hda	Siegel & Meyer	Darmstadt University of Applied Sciences	h_da Submission for the Germeval Shared Task on the Identification of Offensive Language
UdSW	Wiegand et al.	Saarland University	Saarland Universitys Participation in the GermEval Task 2018 (UdSW) Examining Different Types of Classifiers and Features
HaUA	De Smedt & Jaki	Hildesheim University & Antwerpen University	Challenges of Automatically Detecting Offensive Language Online: Participation Paper for the Germeval Shared Task 2018 (HaUA)
KAUSTmine	Bachfischer et al.	King Abdullah University for Science and Technology	KAUSTmine - Offensive Comment Classification on German Language Microposts
hpiTM	Risch et al.	University of Potsdam	Fine-Grained Classification of Offensive Language
TUWienKBS	Montani & Schüller	TU Wien	TUWienKBS at GermEval 2018: German Abusive Tweet Detection
Potsdam	Scheffler et al.	University of Potsdam	Feature Explorations for Hate Speech Classification
SaarOffDe	Stammbach et al.	Saarland University	Offensive Language Detection with Neural Networks for Germeval Task 2018
RuG	Bai et al.	Rijksuniversiteit Groningen & Universita degli Studi di Salerno	RuG at GermEval: Detecting Offensive Speech in German Social Media
upInf	Birkeneder et al.	University of Passau & University of St. Gallen	upInf - Offensive Language Detection in German Tweets
InriaFBK	Corazza et al.	Universite Cote dAzur & Fondazione Bruno Kessler	InriaFBK at Germeval 2018: Identifying Offensive Tweets Using Recurrent Neural Networks
uhhLT	Wiedemann et al.	University of Hamburg	Transfer Learning from LDA to BiLSTM-CNN for Offensive Language Detection in Twitter
DFKILT	Moreno Schneider et al.	DFKI GmbH	Towards the Automatic Classification of Offensive Language and Related Phenomena in German Tweets

HIIwiStJS	Schäfer	University of Hildesheim	HIIwiStJS at GermEval-2018: Integrating Linguistic Features in a Neural Network for the Identification of Offensive Language in Microposts
ULMFiT	Rother & Rettberg	Hochschule Hamm-Lippstadt	ULMFiT at GermEval-2018: A Deep Neural Language Model for the Classification of Hate Speech in German Tweets
fkieITF	Kent	Fraunhofer FKIE	German Hate Speech Detection on Twitter
FoSIL	Xi et al.	University of Applied Sciences Mittweida & Fraunhofer SIT	CNN-Based Offensive Language Detection
spMMMP	von Grünigen et al.	Zurich University of Applied Sciences & SpinningBytes AG	spMMMP at GermEval 2018 Shared Task: Classification of Offensive Content in Tweets using Convolutional Neural Networks and Gated Recurrent Units
iam	Mishra et al.	IIIT-Hyderabad & i.am+ LLC	GermEval 2018 : Machine Learning and Neural Network Approaches for Offensive Language Identification

Table 1: Group IDs, Authors and Paper Titles

Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language

Michael Wiegand
Spoken Language Systems
Saarland University
michael.wiegand@lsv.uni-saarland.de

Melanie Siegel
Information Science
Darmstadt University of Applied Sciences
melanie.siegel@h-da.de

Josef Ruppenhofer
Empirical Linguistics and Language Modelling
Institut für deutsche Sprache
ruppenhofer@ids-mannheim.de

Abstract

We present the pilot edition of the GermEval Shared Task on the Identification of Offensive Language. This shared task deals with the classification of German tweets from Twitter. It comprises two tasks, a coarse-grained binary classification task and a fine-grained multi-class classification task.

The shared task had 20 participants submitting 51 runs for the coarse-grained task and 25 runs for the fine-grained task. Since this is a pilot task, we describe the process of extracting the raw-data for the data collection and the annotation schema. We evaluate the results of the systems submitted to the shared task. The shared task homepage can be found at https://projects.cai.fbi.h-da.de/iggsa/

1 Introduction

Offensive Language is commonly defined as hurtful, derogatory or obscene comments made by one person to another person. This type of language can be increasingly found on the web. As a consequence, many operators of social media websites no longer manage to manually monitor user posts. Therefore, there is a pressing demand for methods to automatically identify suspicious posts.

The GermEval Shared Task on the Identification of Offensive Language is intended to initiate and foster research on the identification of offensive content in German language microposts. Offensive comments are to be detected from a set of German tweets. We focus on Twitter since tweets can be regarded as a prototypical type of micropost.

The shared task was endorsed by two of the special interest groups of the German Society for Computational Linguistics and Language Technology (GSCL): the Interest Group on German Sentiment Analysis (IGGSA) as well as the Interest Group on Social Media Analysis.

This paper will give a short overview on related work in §2. We will then describe the task in §3 and the data in §4. 20 teams participated in the shared task. We describe the participants and their approaches in §5 and give an overview of the results in §6.

2 Related Work

For a detailed summary of related work on the detection of abusive language, we refer the reader to Schmidt and Wiegand (2017). In the following, we will briefly comment on related shared tasks and datasets in German language. We will also provide some information on the GermEval evaluation campaign.

- Kaggle's Toxic Comment Classification Challenge[1] is a shared task in which comments from the English Wikipedia are to be classified. There are 6 different categories of toxity to be identified (i.e. *toxic*, *severe toxic*, *obscene*, *insult*, *identity hate* and *threat*). These categories are not mutually exclusive.

- The shared task on aggression identification[2] includes both English and Hindi Facebook comments. Participants have to detect abusive comments and to distinguish between *overtly aggressive comments* and *covertly aggressive comments*.

- The shared task on Automatic Misogyny Identification (AMI)[3] is jointly run by IberEval[4]

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

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

[3] https://amievalita2018.wordpress.com https://amiibereval2018.wordpress.com

[4] https://sites.google.com/view/ibereval-2018

and EVALITA[5]. It exclusively focuses on the detection of misogynist tweets on Twitter. There are two subtasks. Task A addresses the identification of misogynist tweets, while Task B focuses on the categorization of misogynist tweets (i.e. *Discredit*, *Derailing*, *Dominance*, *Sexual Harassment & Threats of Violence*, *Stereotype & Objectification*, *Active* and *Passive*). Both IberEval and EVALITA include a task on English tweets. IberEval also includes a task on Spanish tweets while EVALITA also includes a subtask on Italian tweets.

We are not aware of any shared task on the detection abusive language that includes German language data. With regard to publicly-available German datasets for this task, we only know of Ross et al. (2016) who present a dataset of about 500 tweets which has been annotated regarding hate speech. The authors employed a binary categorization scheme. While the dataset from Ross et al. (2016) may be too small for some data-hungry learning-based approaches, we hope that the German dataset we introduce in this shared task is sufficiently large (i.e. more than 8,000 tweets) even for those approaches.

GermEval is a series of shared task evaluation campaigns that focus on Natural Language Processing for the German language. So far, there have been three iterations of GermEval, each with a different type of task: named entity recognition (Benikova et al., 2014), lexical substitution (Tristan Miller et al., 2015) and aspect-based sentiment analysis in social media customer feedback (Wojatzki et al., 2017). GermEval shared tasks have been run informally by self-organized groups of interested researchers.

3 Task Description

Participants were allowed to participate in one or both tasks and submit at most three runs per task.

3.1 Task 1: Coarse-grained Binary Classification

Task 1 was to decide whether a tweet includes some form of offensive language or not. The tweets had to be classified into the two classes OFFENSE and OTHER. The OFFENSE category covered abusive language, insults, as well as merely profane statements.

[5] http://www.evalita.it/2018

3.2 Task 2: Fine-grained 4-way Classification

The second task involved four categories, a non-offensive OTHER class and three sub-categories of what is OFFENSE in Task 1. In the case of PROFANITY, profane words are used, however, the tweet does not want to insult anyone. This typically concerns the usage of swearwords (*Scheiße*, *Fuck* etc.) and cursing (*Zur Hölle! Verdammt!* etc.). This can be often found in youth language. Swearwords and cursing may, but need not, co-occur with insults or abusive speech. Profane language may in fact be used in tweets with positive sentiment to express emphasis. Whenever profane words are not directed towards a specific person or group of persons and there are no separate cues of INSULT or ABUSE, then tweets are labeled as simple cases of PROFANITY.

In the case of INSULT, unlike PROFANITY, the tweet clearly wants to offend someone. INSULT is the ascription of negatively evaluated qualities or deficiencies or the labeling of persons as unworthy (in some sense) or unvalued. Insults convey disrespect and contempt. Whether an utterance is an insult usually depends on the community in which it is made, on the social context (ongoing activity etc.) in which it is made, and on the linguistic means that are used (which have to be found to be conventional means whose assessment as insulting are intersubjectively reasonably stable).

And finally, in the case of ABUSE, the tweet does not just insult a person but represents the stronger form of abusive language. By abuse we define a special type of degradation. This type of degrading consists in ascribing a social identity to a person that is judged negatively by a (perceived) majority of society. The identity in question is seen as a shameful, unworthy, morally objectionable or marginal identity. In contrast to insults, instances of abusive language require that the target of judgment is seen as a representative of a group and it is ascribed negative qualities that are taken to be universal, omnipresent and unchangeable characteristics of the group. (This part of the definition largely co-incides with what is referred to as abusive speech in other research.) Aside from the cases where people are degraded based on their membership in some group, we also classify it as abusive language when dehumanization is employed even just towards an individual (i.e. describing a person as scum or vermin etc.).

3.3 Evaluation Metrics

We evaluate the classification performance by the common evaluation measures *precision*, *recall*, and *F1-score*. These measures are computed for each of the individual classes in the two tasks. For each task, we also compute the *macro-average* precision, recall and F1-score. We also compute accuracy. We rank systems by their macro-average scores. We do not use accuracy since in both tasks the class distribution is fairly imbalanced. Accuracy typically rewards correct classification of the majority class.

An evaluation tool computing all of the above evaluation measures on the two tasks of the shared task was provided by the organizers prior to the release of the training data. It is publicly available and can be downloaded via the webpage of the shared task.

4 Data Set

As a source for our data collection, we chose Twitter. Thus we are able to make our collection publicly available. Unlike existing corpora, Twitter also contains a much higher proportion of offensive language (Wiegand et al., 2018).

4.1 Data Collection

Much care was taken in sampling the tweets for our gold standard. Although a natural sample of tweets would represent the most unbiased form of data, we decided against it. A sample of a few thousand tweets would have resulted in just too few occurrences of offensive language as the proportion of offensive tweets is known to be generally low (Schmidt and Wiegand, 2017). We also decided against sampling by specific query terms (as Waseem and Hovy (2016) suggest) since our initial experiments showed that using offensive query terms, such as *Idiot* or *Schmarotzer*, greatly reduced the variety of offensive terms occurring in the retrieved tweets.[6]

Instead, we sampled tweets from the timeline of various users. In total, we considered about 100 different users. We started by heuristically identifying users that regularly post offensive tweets. By sampling from their timeline, we obtained offensive tweets that exhibited a more varied vocabulary than we would have obtained by sampling by predefined query terms. It also enabled us to extract a substantial amount of non-offensive tweets since only very few users *exclusively* post offensive content.

Although this extraction process prevents the dataset from becoming biased towards specific topics trending at the point in time when the extraction is run (a problem one typically faces when extracting data from the Twitter-stream), we still found certain topics dominating our extracted data. Most of the extracted offensive tweets concerned the situation of migrants or the German government. The tweets not considered offensive, however, often addressed different topics. For example, the politician names *Maas* and *Merkel* and the common noun *Flüchtlinge* 'refugees' were almost exclusively observed in offensive tweets. Since these high-frequency words undoubtedly do not represent offensive terms, we decided to *debias* our data collection by sampling further arbitrary tweets containing these terms. We specifically sought tweets from across the entire political spectrum. We also deliberately included tweets from users that regularly post highly-critical tweets with respect to the above topics. Otherwise, our data collection would allow classifiers to score well that simply infer offensive content by observing a negative polarity cooccurring with particular topics (e.g. *Maas*, *Merkel* or *Flüchtlinge*).

When sampling tweets from Twitter, we also imposed certain formal restrictions on the tweets to be extracted. They are as follows:

(1) Each tweet had to be written in German.

(2) Each tweet had to contain at least five ordinary *alphabetic* tokens.

(3) No tweet was allowed to contain any URLs.

(4) No tweet was allowed to be a retweet.

All of these restrictions are mainly designed to speed up the annotation process (cf. §4.2) by removing tweets that are not relevant to the gold standard. (2) was included to remove tweets that just function as an advertisement or spam. We wanted to exclude URLs (3) since our data collection should be self-contained to the degree possible.[7] We avoid retweets since they represent a form of reported content where it is often difficult to decide whether the views expressed in the reported content are shared by the user retweeting or not.

[6] Our observation was that the overwhelming proportion of retrieved tweets would contain just the query words as offensive terms.

[7] The offensive nature of tweets with an URL often only becomes visible by taking into account their linked content.

In splitting our data collection into training and test set, we made sure that any given user's complete set of tweets was assigned to either the training set or the test set. In this way, we wanted to avoid that classifiers could benefit from learning user-specific information. For example, if a user, who very often posts offensive tweets has a very idiosyncratic writing style and his/her tweets were distributed across training and test set, then a classifier could exploit the knowledge about the writing style in order to infer offensive language. Such a classifier would not really have learned to detect offensive language but a very specific writing style which, beyond that given dataset, would not be of any use for detecting offensive language.

The data collection was also divided up in such a manner that the training and test sets have a similar class distribution. This is one of the major prerequisites for supervised learning approaches to work effectively.

4.2 Annotation

Each tweet of the resulting data collection with an overall size of 8541 tweets was manually annotated by one of the three organizers of the shared task. All annotators are native speakers of German.

In order to measure inter-annotation agreement, a sample of 300 tweets were annotated by the three annotators in parallel. We removed all tweets that were marked as HUNH or EXEMPT at least by one annotator. HUNH was used for incomprehensible utterances. We do not require that a sentence is perfectly grammatically well-formed and correctly spelled to be included in our data. However, if a sentence is so erroneous that the annotator does not understand its content, then this sentence was labeled as HUNH and removed. This label also applies if the sentence is formally correct but the annotator still does not understand what is meant by this utterance. Tweets that are EXEMPT from the subtyping annotation involve tweets which only contain abuse or insults that represent the view of somebody other than the tweeter, utterances which depend on non-textual information, utterances that are just a series of hashtags and/or usernames, even if they indicate abusive speech (e.g. #crimigrants or #rapefugees), or utterances that are incomplete.

On the remaining 240 tweets, an agreement of $\kappa = 0.66$ was measured. It can be considered substantial (Landis and Koch, 1977). All remaining tweets of the gold standard were only annotated by one of the three annotators.

Table 1 displays the class distribution among the training and the test set. It comes as no surprise that non-offensive tweets represent the majority class. The most frequent subtype of offensive language are cases of abuse followed by (common) insults. By far the smallest category are profane tweets.

4.3 Data Format

Our data is distributed in the form of tab-separated value files. An example row representing one tweet is shown in Table 2. As the task is focused only on the linguistic aspect of offensive language, each tweet is represented only by its text in column 1. Meta-data contained in Twitter's json files was not used. The text column is followed by the coarse-grained label in column 2 and the fine-grained label in column 3. Note that we applied no preprocessing to the tweet text with one exception: as shown in Table 2, line breaks were replaced with the special 5-character string `|LBR|` so that each tweet could be stored on one line.

5 Participants and Approaches

Overall, we had 20 teams participating in the shared task. All teams participated in Task 1 and 11 of them took part in Task 2.

Across both tasks, the teams made use of a variety of approaches. Below, we identify some major trends and commonalities between the teams. For a detailed description of the systems, we refer readers to the dedicated system description papers.

5.1 Preprocessing

Tokenization 9 teams mention tokenization as a preprocessing step in their papers. Most used tokenizers adapted to social media: 3 teams used the TweetTokenizer in nltk (Bird et al., 2009), one team used the SoMaJo social media tokenizer (Proisl and Uhrig, 2016), one team used twokenize (Owoputi et al., 2013) and one team developed an extension of the tokenizer of Baziotis et al. (2017). Of the others, one team used the tokenizer in spaCy[8], one team split based mostly on punctuation and the last team did not give any details about its tokenizer.

POS-Tagging 6 teams used POS-Tagging. In most cases, the POS-tags were not produced by a stand-alone tagger but derived from a more complex software tool such as spaCy, the TextBlob[9]

[8] https://spacy.io/

[9] https://github.com/sloria/TextBlob

		training set		test set	
categories		**freq**	**%**	**freq**	**%**
coarse-grained	OFFENSE	1688	33.7	1202	34.0
	OTHER	3321	66.3	2330	66.0
fine-grained	ABUSE	1022	20.4	773	21.9
	INSULT	595	11.9	381	10.8
	PROFANITY	71	1.4	48	1.4
	OTHER	3321	66.3	2330	66.0
total		5009	100.0	3532	100.0

Table 1: Class distribution on training and test set.

@Ralf_Stegner Oman Ralle..dich mag ja immer noch keiner. Du willst das die Hetze gegen dich aufhört? |LBR| Geh in Rente und verzichte auf die 1/2deiner Pension	OFFENSE	INSULT

Table 2: Data format

package or the ParZu dependency parser (Sennrich et al., 2013).

Lemmatization and stemming 5 systems used lemmatization. Three teams used spaCy, and one team each used the TreeTagger (Schmid, 1995) and ParZu. 2 teams used stemming.

Parsing Only two teams used parsing, one the ParZu parser (Sennrich et al., 2013) and the other the mate-tools parser (Björkelund et al., 2010).

5.2 Lexical Resources

While 8 teams used no task-specific lexicon, 8 other teams used one or more publicly available lexicons, and 7 teams created a new lexicon.[10] 9 teams used polarity lexicons, chief among them PolArt (Klenner et al., 2009), PolarityClues (Waltinger, 2010) and SentiWS (Remus et al., 2010), and 8 teams used dictionaries containing swearwords, slurs or offensive words. Several teams expanded available polarity of swearword lexicons. One team translated and post-edited the English dictionary of abusive terms provided by Wiegand et al. (2018).

5.3 Word Vectors

15 teams used pre-trained word embeddings in their systems. The most commonly used vectors were those provided by spinningbytes (word2vec, fasttext) on the one hand and those provided by the organizers (word2vec) on the other hand. Some teams trained on tweet collections of their own. Two teams pursuing a cross-lingual or translation approach used multi-lingual word embeddings, the aligned languages being German and English in both cases. One team used embeddings only for the purpose of lexicon expansion but not as a feature fed to their classifier.

5.4 Classifiers

The classifiers used involve a fairly broad variety of familiar non-neural types as well as (variations on) recent neural network-type classifiers. Among the non-neural types, SVMs were the most common type. 12 teams used a flavor of SVM, either as a baseline or their main system. Logistic regression was used by 7 teams, in two cases as a meta-classifier. Decision Trees were used by 2 teams and 1 team used a Naive Bayes classifier. Among the neural network classifiers common recent architectures are found: CNN (10 teams), LSTM and variants (11 teams), GRU (6 teams), as well as combinations of these.

6 Submissions and Results

The full set of results for both tasks is available at the shared task website.

A high-level summary of the results is given in Table 3, which provides summary statistics on the macro-average F1 score that was used as the official ranking criterion in the shared task. As the table shows, the scores achieved span a substantial range: more than 25% points in the case of the coarse task and more than 20% points in the case

[10] The publicly available lexicons used were often ones that the shared task organizers had pointed out on the shared task's web pages.

of the fine-grained task.

6.1 Coarse-grained Classification

We received 51 different runs from 20 teams for the binary classification into OFFENSE vs OTHER. For lack of space, we only show the best 15 runs in Table 4. As a baseline, we also included the performance of majority-class classifier always predicting the majority class OTHER.

6.2 Fine-grained Classification

We received 25 different runs from 10 teams for the fine-grained task that distinguishes three sub-types of offensive language from OTHER. We report the best 10 submissions in Table 5. As a baseline, again, we included the performance of majority-class classifier always predicting the majority class OTHER.

6.3 General Conclusions Drawn from the Evaluation

6.3.1 System Design

Given the diversity of approaches and the large number of participating groups in this shared task, it is difficult to draw general conclusions about the effectiveness about specific types of features.

With regard to the choice of classifiers, there is a competition between traditional feature-based supervised learning (typically represented by SVMs) and the more recent deep learning methods. Undoubtedly, most top performing systems in both shared tasks employed deep learning (e.g. *spMMMP, uhhLT, SaarOffDe, InriaFBK*), yet the top performing system in Task 1 and the second-best performing system in Task 2 (i.e. *TUWienKBS*) exclusively employed traditional supervised learning. This team even explicitly states in its participation paper that the usage of deep learning did not improve their results. This makes us wonder whether the frequent occurrence of such methods in top performing systems is just a result of the current popularity of deep learning algorithms and whether traditional engineering is not similarly effective (at least for the classification task in GermEval 2018). We also note that there was quite a bit of variation among the specific deep learning approaches used. It was not necessarily the most complex approach that produced the best results. For example, *SaarOffDe* with its straightforward approach of using RNNs and CNNs produced top scores. The scores of systems employing complex transfer-learning (e.g. *spMMMP*, *InriaFBK* or *uhhLT*) are not necessarily better.

Although overall it may not always be a crucial aspect of system design, the usage of ensemble classification seems to very often improve classification approaches (e.g. *Potsdam*, *RuG*, *SaarOffDe*, *TUWienKBS*, *UdSW*).

With regard to traditional feature engineering, the features found effective very much reflect the insights of recent research on English data, particulary the extensive study presented in Nobata et al. (2016). Several submissions include a combination of word embeddings, character n-grams and some form of (task-specific) lexicon. Both HaUA and *UdSW* report that high performance scores can already be achieved with a classifier solely relying on a lexicon. Yet both groups show that such classifiers can be outperformed by classifiers using additional (typically more generic) features, e.g. character n-grams.

The usage of datasets from other languages (typically English) to augment the training data provided by GermEval may be a very popular idea (e.g. *InriaFBK*, *hpiTM*, *UdSW*, *spMMMP*), however, the results of this shared task do not support systematic effectiveness.[11] There are two issues that may stand in the way. Firstly, the definition of abusive language varies throughout the different datasets. Secondly, the predominant type of abuse may be different: Not every English dataset on abusive language detection similarly has so many abusive comments towards migrants as the GermEval dataset.

6.3.2 Task and Data

Overall, we can conclude that the task of identifying offensive language on German tweets is doable. However, with the highest F-scores up to 76% F1-score on Task 1 and 52% on Task 2, the task is clearly far from solved. If we consider the large span of different F1-scores within the same task (i.e. 27% points on Task 1 and 20% points on Task 2), we also have to acknowledge that building classifiers that achieve top scores is not a trivial undertaking.

The overall performance scores achieved on Task 2 are considerably lower than on Task 1. This does not come as a surprise as Task 2 is considerably more difficult, having 4 instead of 2 classes. More-

[11] UdSW reports that no matter how crosslingual information is added to a classifier, the performance compared to a monolingual classifier drops.

task	# teams	# runs	min	max	median	mean	sd
coarse	20	51	49.03	76.77	69.15	66.35	8.45
fine	11	25	32.07	52.71	38.76	39.71	5.00

Table 3: Summary statistics for overall macro-F1 scores in the two tasks

over, for some classes, particularly PROFANITY there are simply too few instances in the dataset (Table 1).

On several comparable English datasets, much higher classification scores have been reported (Agrawal and Awekar, 2018; Badjatiya et al., 2017). Again, there may be several reasons for that. German is undoubtedly more difficult than English. Due to its higher degree of inflection and compounding, the issue of data sparseness is more prominent. Additionally, we took great efforts in removing biases from the dataset allowing classifiers to overfit (§4.1). For example, we found that if we were to eliminate the constraint that tweets in training and test data have to originate from different users, performance of supervised classifiers would increase by approximately 7% points in F1-score.

Although a proper error analysis is beyond the scope of this overview paper, we inspected the output of the best performing systems and found that while offensive utterances that contain predictive keywords, also referred to as *explicit offense* (Waseem et al., 2017), are mostly detected, offensive utterances that lack such keywords, also referred to as *implicit offense* (Waseem et al., 2017), are mostly missed. Examples (5)-(9) display some of the latter tweets. Clearly, many of these cases require world knowledge and thus remain out of reach for systems that solely employ text classification.

(5) Ich verstehe immer weniger, warum die Polen, Tschechen und Ungarn unsere vorbildliche Migrationspolitik nicht mitmachen wollen. Ist es denen nicht langweilig mit Weihnachtsmärkten so ganz ohne Barrieren, Polizisten und Nagelbomben?

(6) Sei mal ehrlich, wie sollen man Frauen noch ernst nehmen?

(7) Zion wird sein Nürnberg jetzt erleben!

(8) Wenn wir Glück haben, wird China die Welt beherrschen. Wenn wir Pech haben, der Islam.

(9) Da zeigt sich leider mal wieder dass uns der Fall der Mauer nicht nur viel Gutes gebracht hat sonder eben auch @RenateKuenast. #falldermauer

A final aspect of the task design and evaluation that leads to significantly lower scores on the fine-grained task is the combination of macro-F1-based scoring and the inclusion of a very low-frequency class among the labels, namely PROFANITY. Performance on that class was low even for the overall best teams (cf. Table 5), dragging down the macro-F1 score. By comparison, the accuracy for the fine-grained task is only about 6% lower than for the coarse-grained task.

7 Conclusion

In this paper, we described the pilot edition of the GermEval Shared on the Identification of Offensive Language. The shared task comprises two tasks, a coarse-grained binary classification task and a fine-grained multi-class classification task. 20 groups submitted to the former task while 10 groups submitted to the latter task.

Our results show that both tasks are doable but difficult and far from solved. In terms of features and classifiers, there is no clear winner. While many deep-learning approaches produce good scores, traditional supervised classifiers may produce similar scores. Word embeddings, character n-grams and lexicons of offensive words are popular features, but a robust system does not necessarily have to include all three components. Ensemble methods mostly help. The effectiveness of crosslingual methods is debatable. Implicitly offensive language seems particularly difficult.

Though much care was taken in creating the annotated data of the shared task, it is not clear in how far the top performing systems in our shared task overfit to the dataset we created. Therefore, an obvious extension to this task that could shed more light onto the question of generalization would consist of including data from additional domains.

We introduced a new dataset of 8,000 annotated tweets as part of this shared task. All this data has been made publicly available to the research community via the shared task website.

	Submission		Accuracy			Offense			Other			Average		
	Team	RunID	Percent	Correct	Total	P	R	F	P	R	F	P	R	F
1	TUWienKBS	coarse_1	79.53	2809	3532	71.87	65.47	68.52	82.97	86.78	84.83	77.42	76.13	76.77
2	spMMMP	coarse_2	78.85	2785	3532	74.65	57.32	64.85	80.34	89.96	84.88	77.49	73.64	75.52
3	spMMMP	coarse_1	78.60	2776	3532	73.98	57.24	64.54	80.25	89.61	84.67	77.11	73.43	75.22
4	uhhLT	coarse_3	77.49	2737	3532	66.29	68.89	67.56	83.62	81.93	82.77	74.96	75.41	75.18
5	SaarOffDe	coarse_3	77.27	2729	3532	66.72	66.22	66.47	82.64	82.96	82.80	74.68	74.59	74.64
6	SaarOffDe	coarse_1	77.32	2731	3532	67.12	65.39	66.25	82.38	83.48	82.92	74.75	74.43	74.59
7	InriaFBK	coarse_1	76.90	2716	3532	66.11	65.89	66.00	82.43	82.58	82.50	74.27	74.23	74.25
8	InriaFBK	coarse_2	78.20	2762	3532	73.18	56.74	63.92	80.00	89.27	84.38	76.59	73.00	74.25
9	SaarOffDe	coarse_2	76.50	2702	3532	65.68	64.81	65.24	81.97	82.53	82.25	73.83	73.67	73.75
10	InriaFBK	coarse_3	77.24	2728	3532	70.43	57.07	63.05	79.83	87.64	83.55	75.13	72.36	73.72
11	HaUA	coarse_1	76.70	2709	3532	72.91	50.17	59.44	77.86	90.39	83.65	75.38	70.28	72.74
12	UdSW	coarse_3	75.62	2671	3532	66.47	57.24	61.51	79.42	85.11	82.16	72.94	71.17	72.05
13	DFKILT	coarse_2	76.02	2685	3532	77.95	41.18	53.89	75.60	93.99	83.80	76.77	67.59	71.89
14	Potsdam	coarse_3	75.91	2681	3532	72.41	47.17	57.13	76.90	90.73	83.24	74.66	68.95	71.69
15	uhhLT	coarse_1	75.42	2664	3532	71.52	46.17	56.12	76.52	90.52	82.93	74.02	68.34	71.07
	majority-class classifier		65.97	2330	3532	-N/A-	-N/A-	-N/A-	65.97	100.00	79.50	32.98	50.00	39.75

Table 4: Top 15 runs for Task 1: coarse-grained classification

	Submission		Accuracy			Abuse			Insult			Other			Profanity			Average		
	Team	RunID	Percent	Correct	Total	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
1	uhhLT	fine_3	73.67	2602	3532	54.71	51.88	53.25	55.19	30.71	39.46	81.13	88.93	84.85	36.36	25.00	29.63	56.85	49.13	52.71
2	TUWienKBS	fine_1	74.52	2632	3532	63.70	44.50	52.40	50.87	38.32	43.71	80.83	91.42	85.80	17.14	25.00	20.34	53.14	49.81	51.42
3	uhhLT	fine_2	72.79	2571	3532	56.64	47.99	51.96	46.39	35.43	40.18	80.52	88.37	84.26	20.69	12.50	15.58	51.06	46.07	48.44
4	uhhLT	fine_1	70.44	2488	3532	49.92	42.82	46.10	43.80	13.91	21.12	76.61	90.26	82.88	33.33	2.08	3.92	50.92	37.27	43.04
5	InriaFBK	fine_2	70.50	2490	3532	58.99	31.82	41.34	37.76	29.13	32.89	76.46	91.46	83.29	5.88	4.17	4.88	44.77	39.15	41.77
6	InriaFBK	fine_3	68.66	2425	3532	54.24	37.26	44.17	29.75	28.35	29.03	77.57	86.95	81.99	11.54	6.25	8.11	43.27	39.70	41.41
7	spMMMP	fine_3	67.89	2398	3532	48.86	38.68	43.18	32.43	18.90	23.88	75.57	86.82	80.81	19.05	8.33	11.59	43.98	38.18	40.88
8	InriaFBK	fine_1	67.89	2398	3532	51.64	30.53	38.37	30.24	33.33	31.71	77.04	87.25	81.83	12.50	4.17	6.25	42.85	38.82	40.74
9	fkielTF	fine_1	68.74	2428	3532	66.36	18.89	29.41	34.31	18.37	23.93	71.21	94.89	81.36	33.33	2.08	3.92	51.30	33.56	40.58
10	RuG	fine_1	69.42	2452	3532	53.29	31.44	39.54	43.17	15.75	23.08	73.34	92.19	81.69	12.50	2.08	3.57	45.57	35.35	39.82
	majority-class classifier		65.97	2330	3532	-N/A-	-N/A-	-N/A-	-N/A-	-N/A-	-N/A-	65.97	100.00	79.50	-N/A-	-N/A-	-N/A-	16.49	25.00	19.87

Table 5: Top 10 results for Task 2: fine-grained classification

Acknowledgments

We would like to thank Torsten Zesch for providing very constructive feedback in the early stages of producing the gold standard data of the shared task.

We would like to thank Markus Meyer for maintaining home page and mailing lists and supporting the evaluation process.

Michael Wiegand was partially supported by the German Research Foundation (DFG) under grant WI 4204/2-1.

References

Sweeta Agrawal and Amit Awekar. 2018. Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms. In *Proceedings of the European Conference in Information Retrieval (ECIR)*, pages 141–153, Grenoble, France.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 759–760, Perth, Australia.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August. Association for Computational Linguistics.

Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Padó. 2014. GermEval 2014 Named Entity Recognition Shared Task: Companion Paper. In *Workshop Proceedings of the KONVENS Conference*, pages 104–112, Hildesheim, Germany.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, COLING '10, pages 33–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

Manfred Klenner, Angela Fahrni, and Stefanos Petrakis. 2009. Polart: A robust tool for sentiment analysis.

J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 145–153, Republic and Canton of Geneva, Switzerland.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 380–390.

Thomas Proisl and Peter Uhrig. 2016. Somajo: State-of-the-art tokenization for german web and social media texts. In *WAC@ACL*.

Robert Remus, Uwe Quasthoff, and Gerhard Heyer. 2010. Sentiws-a publicly available german-language resource for sentiment analysis. In *LREC*.

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

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *Proceedings of the ACL SIGDAT-Workshop. Dublin*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the EACL-Workshop on Natural Language Processing for Social Media (SocialNLP)*, pages 1–10, Valencia, Spain.

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting synergies between open resources for german dependency parsing, pos-tagging, and morphological analysis. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 601–609.

David J. Tristan Miller, Darina Benikova, and Sallam Abualhaija. 2015. GermEval 2015: LexSub A Shared Task for German-language Lexical Substitution. In *Proceedings of GermEval 2015: LexSub*, pages 1–10, Essen, Germany.

Ulli Waltinger. 2010. Germanpolarityclues: A lexical resource for german sentiment analysis. In *LREC*.

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, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the ACL-Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada.

Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018. Inducing a Lexicon of Abusive Words – A Feature-Based Approach. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, New Orleans, USA.

Michael Wojatzki, Eugen Ruppert, Sarah Holschneider, Torsten Zesch, and Chris Biemann. 2017. GermEval 2017: Shared Task on Aspect-based Sentiment in Social Media Customer Feedback. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 1–12, Berlin, Germany.

Offensive Language without Offensive Words (OLWOW)

Manfred Klenner
Institute of Computational Linguistics
University of Zurich
Switzerland
klenner@cl.uzh.ch

Abstract

In our contribution, we have applied stance analysis in order to identify offensive discourse. This gives us access to the pros and cons of the writer of some tweets and reveals his/her role framing of the discourse referents. We also semi-automatically augmented our polarity lexicon with a new type of polarity labels, namely P for profanity. Starting from seed words, we derived new entries on the basis of word embeddings. Our approach also focuses on offensive language without offensive words (OLWOW) and discusses strategies to cope with it.

1 Introduction

The GermEval Task 2018 deals with offensive language. The training material are about 5,000 German tweets classified (task I) as offensive (label OFFENSE) compared to not offensive (label OTHER). Task II further specifies offensive language as profanity, abuse or insult. According to the annotation guidelines, profanity indicates the use of indecent, nasty or vulgar vocabulary, while insult and abuse moreover are given, if such words are used to characterise the attributes of a person (INSULT) or to assign a negatively connotated social class to a person (ABUSE). See the following examples insult (ex. 1), abuse (ex. 2) and profanity (ex .3).

ex. 1. Merkel ist die grösste Versagerin der Weltgeschichte !!! (Merkel is the biggest loser in world history)

ex. 2. Clinton - Der Antichrist (Clinton -the antichrist)

ex. 3. Ist zum kotzen (it sucks)

After a couple of attempts to predict the annotations of the gold standard, the author of this paper is convinced that this annotation task was not trivial. I still believe that the annotations of a couple of sentences are debatable.

About one third of the data is classified as offensive language, where abuse is the majority class (about 1,000 tweets), followed by insult (600 tweets) and complemented by a small profanity sample (70 examples). The majority baseline for task I - assigning OTHER - yields an accuracy of 66.3%.

A glimpse at the data reveals that - as expected - the vocabulary being used is the central indicator of offensive language. This seems to prompt for a lexicon-based solution, although the resulting task then is to deal with unknown words. Especially compounds are a very flexible means to create new words in German. But the number of vulgar words is large, anyway, so a mechanism to induce such words is needed. Word embeddings might help. Thus deep learning comes into play. However, we were not so much interested in the best performing black box, but wanted to find out whether our stance analysis system based on a purely symbolic computation could be of any use.

2 Resources

The organisers provided a couple of resources, among others German word embeddings, but also lexicons with e.g. German swearwords. We only integrated one resource, the swearword lexicon. We did it semi-automatically. First, we determined the frequency of each word in a corpus of Facebook posts from a German right-wing party. Then we had a look at the most frequent words and kept 300 of them. We added these words to our polarity lexicon for German, comprising 6,800 nouns and adjectives classified as positive or negative in one of three dimension, namely, the dimension of emotion, moral or appreciation (following the distinction of the appraisal theory, cf. (Martin and White., 2005)). We also have specified a verb lex-

icon comprising 1.100 verbs, where a verb might have various frames indicating the syntactic frame of the verb and whether the verb has a polar effect on its arguments (positive or negative). For example, the verb *anpöbeln* (to accost sb, to molest sb) casts a negative effect on its agent role (which is the source) and on its patient role (the target). Also a negative relation (con) between source and target is assigned (given that the verb is being affirmatively used). This forms the basis of our system for stance analysis. We also assigned verb specific polar roles to source and target. For instance, the patient of the target role of *verleumden* (to slander, slur, vilify) is said to be a victim while the source takes the role of a villain. We call the assignment of polar roles to discourse referents *role framing*, since it conceptualizes a referent in a specific way. It represents the writer perspective. It indirectly indicates the writer's stance towards the referents: he/she is against the villain but in favour of the victims.

Although we are dealing with tweets, we applied an ordinary dependency parser (Sennrich et al., 2013). We just stripped hash tags, emoticons and other social media language noise.

3 Qualitative Analysis

Although it is rather evident that - for a good performance - a subsymbolic approach would be well suited (either character level n-grams or deep learning), we pursued another approach. Our goal was to find out, whether our system for stance analysis could help to understand the problem and help to solve the task. The idea was to first identify the proponents and opponents of the writer of the (offensive) tweets and then to look for polar relations where e.g. a proponent of the writer received a negative effect, or the opponent of the writer received a positive effect. We thought that such constellations might bear conflict potential which - in the best case - would be the yeast of offensive language usage. Very soon we realised that we still had to deal with vocabulary gaps, since most of the time offensive language is based on the usage of offensive words. Actually, our hope was that we were able to identify exactly those cases of (implicit) offensive language that are not indicated by offensive words. We give a couple of examples (cf. examples ex.4 to ex.6).

ex. 4. Das deutsche Volk wird unaufhörlich belogen! (The German people are constantly being lied to!)

ex. 5. Merkel muss weg. (Merkel has to go.)

ex. 6. Sie warnen vor Nazis und führen deren Methoden der Bücherverbrennung und Meinungsunterdrückung ein. (They warn against Nazis and introduce their methods of burning books and suppressing opinion.)

Example ex.4: our system derives that *Volk* is a victim (after passive normalization), since the target of *belügen* (lie to) in an affirmatively used sentence is a victim (the source is a villain, but no source is given here). Example ex.5: a negative effect applies to *Merkel* stemming from *wegmüssen*. We are not able to deal with example ex.6 at the moment. Although a *con* relation from the source (they) to the target (Nazis) is derived, and although we were able to deduce a positive effect on *they*[1] the implicit contrast with the second conjunct (following the "and") is beyond the current capabilities of our system.

These sentences contain no offensive words, but are annotated as offensive language. How to deal with these sophisticated examples?

4 Model Based on Lexicon

We trained a word2vec model on the basis of three Swiss newspapers (NZZ, Tagesanzeiger, Blick). In order to find new examples of offensive words, we manually specified a seed lexicon comprising 20 words. On the basis of the gensim module, we then generated for each seed word the 25 closed neighbors and manually removed false positives. After three rounds, we ended up with 275 entries.

We randomly extracted 500 tweets from the training set as a preliminary test set and carried out several experiments with the full polarity lexicon and subversions of it. This revealed that the precision was ok, but recall was a bit low. Next we calculated the correlation between words of the training set and the offensive class. This gave better results. The precision of OFFENSE was 61.41%, recall was 69.32%, f1 was 65.12% and accuracy was 75.80 %. We took this as our starting point. We now turn to a more detailed description of our approach.

Rather quickly it became clear that some words are very good indicators of offensive language. For instance, the word *Scheiss* (shit) perfectly indicates the class OFFENSE. We thus decided to simply predict the class of a tweet on the basis of these words.

[1] A negative effect on a negative target gives a positive effect on the source of such a situation.

We estimated the probability of an offensive class given a word W with the following approximation:

$$P(OFFENSE|W) \approx \frac{\#(W, OFFENSE)}{\#W}$$

This is: the number of times OFFENSE is the label of a tweet that includes word W divided by the number of times word W occurs in the training corpus. We kept those words that have a probability above 0.75 and of a frequency in the corpus above a THRESHOLD which is 2 for words not in the polarity lexicon and 0 for words from the polarity lexicon. We call this filter the *word indicator filter*(it comprises 508 words) and used it as a classifier in the following way.:

$$P(OFFENSE|TWEET) = 1 \; if$$
$$\exists\, W \in TWEET : P(OFFENSE|W) > 0.75$$
$$\wedge\; freq(W, CORPUS) > THRESHOLD$$

If a tweet contains one word of the filter it is classified as OFFENSE. There are other filters: verb related filters (see next section) and an exclamation mark filter. Those tweets that pass all filters are classified as OTHER.

There are a couple of possible correlations one could take into consideration and a machine learning tool could do this much better than a manual engineer. However, since we were not so much interested in exploiting indicators that are language independent (like the number of hash tags being used, capital letter usage etc.), but rather in the language specific means, we have not undertaken a detailed analysis on that level. The only exception are exclamation marks. If a tweet contains more than two successive exclamation marks, it is classified as offensive. This is the *exclamation mark filter*. Let us now turn to our stance-based filters.

5 Model Based on Stance Detection

Our stance analysis is verb-based (Klenner et al., 2016). It only triggers if a model verb with the right syntactic frame (and sometimes further lexical restrictions) is present. Then, dependent on the verb and its affirmative status (negated or not), role framing, i.e. the assignment of polar roles occurs and a polar relation (pro or con) is established from the source towards the target. The main polar roles are *victim, villain, benefactor, beneficiary, pos_actor, neg_actor, neg_affected, pos_affected*. They are associated with the *source* and *target* (cf. (Wiegand et al., 2016)) of a verb. The source marks the semantic roles of the initiator of the positive or negative relation that a verb expresses towards the target. For instance the verb *to cheat*: the direct object (patient) is the target as well as a victim and the source is the (logical) subject (agent) and it is modelled as a villain (since *to cheat* is morally negative). Our stance model claims that role framing, the assignment of polar roles, reveals the writer perspective, since if the writer conceptualises someone as e.g. a villain, he/she is against this referent. Finding the targets of the stance of the author, thus, boils down to analyse his/her role framing. If the proponents and opponents of the writer are known, we can start to infer additional proponents and opponents of his/her. For instance, if someone is in favour of a proponent of the writer, then this person becomes a candidate proponent of the writer. So if the EU is a proponent of mine and you praises the EU, you might be a proponent of mine. We do not need the full-fledged capabilities of our stance system. We wanted to explore the idea that we were able to identify offensive language, namely the cases where no offensive vocabulary is present.

But the first question was: is our approach comprising 1,100 verbs and about 1,700 different frames plagued by sparseness? In 827 of the 3532 sentences from the test set it triggered. This is 23.4 % of all sentences (for the training set it is 25.38%). This is not too sparse. This gave us 818 polar roles and 176 pro (73) and con (103) relations, altogether 994 assignments. The first step in stance analysis is to find the targets of the writer: who is he/she against or in favour of? We just took those referents conceptualized as villains and neg_actors: $\lambda x : villain(x) \vee neg\_actor(x)$. The result comprises *SPD (a political party), Mob (mob), Salafisten (Salafists), Einwanderer (immigrants), Lügenpresse (lie press), Merkel (German chancelor), Allah (Allah)*. Obviously, the (some) writers are against these referents. And who are the victims? We get (among others): Volk (people), Jude (jew), Planet (planet), Polizist (cop), Deutschland (Germany), Sicherheit (safety), Kind (child), Frauen (women).

Are there correlations we could exploit: e.g. between role framing and the class OFFENSE? We run quite a number of tests. E.g. we determined the probability $P(OFFENSE|villain) = 0.66$, but the are only 35 cases. Other examples are: $P(OFFENSE|neg\_actor) = 0.51$, $P(OFFENSE|victim) = 0.58$, $P(OFFENSE|pos\_actor) = 0.29$. That is,

pos_actor indicates OTHER with a probability of 71%. When it comes to pro and con relations, we got $P(OTHER|pro) = 0.73$ and $P(OTHER|con) = 0.60$. As we can see, a correlation between polar facts and binary classes (task 1) is given, but is not very striking. We use it as filters in our pipeline architecture.

The strongest filter is the word indicator filter. It is applied first. Tweets that do not pass it, are classified as OFFENSE. The rest runs through the filters: pro, pos_actor, villain and victim. Those who pass all filters are classified as OTHER. For our 500 sample test set derived from the training set, this gave us 61.41% precision and 69.32% recall.

6 Offensive Language without Offensive Words

In the training set there are a couple of examples of offensive language without offensive words (OLWOW). We created filters to identify such tweets. If a tweet triggers stance analysis and if a negative polar fact is derived, but none of its words are in our polarity lexicon, then this tweet is a candidate for an OLWOW. If, additionally, a negative polar fact hits an opponent of the writer, it is a candidate of OFFENSE. Here are three examples.

ex. 7. Es gibt bei uns keine Pressefreiheit mehr. (There is no longer a free press.)

ex. 8. Mal schauen wieviel Frauen dieses Jahr von illegalen Einwanderern vergewaltigt oder belästigt werden. (Let us see how many women get raped or harrased by illegal immigrants this year.)

ex. 9. Hier wird Vergewaltigung legalisiert! (Here, rape gets legalized!)

Example 7 and example 8 are annotated as ABUSE, while example 9 is a negative one, since it is annotated with OTHER. Our system is not able to deal with example 7 but correctly identifies example 8: women is classified as victim, immigrants as villain. Since immigrants are an element of the opponents and, in this sentence, are conceptualized as a villain (which is a negative effect), we are entitled to conclude that this tweet is offensive - although neither rape nor harass are offensive words. They denote aggressive events.

The concept of an OLWOW is demanding. According to the gold standard and our filters, 175 tweets are OLWOW tweets. However, if we require that the polar effect hits an opponent (our criteria for offensiveness), this is reduced to 9 cases. There are various reasons for the resulting sparseness: sometimes the parser has introduced wrong sentence boundaries, sometimes a pronoun occupies the polar role and we do not do coreference resolution, sometimes the cause for offensiveness is distributed over more than one sentence, etc. An example of a distributed representation is:

ex. 10. Wir haben Jerusalem vom Islam befreit und das heutige Banken System erfunden. Wer oder was sollte uns aufhalten. Merkel oder Maas etwa. Lachhaft. (We liberated Jerusalem from Islam and invented today's banking system. Who or what should stop us. Merkel or Maas? Ridiculous.)

As we can see, no offensive words are used and the abusive argumentation is distributed among 4 pieces. OLWOW annotations are also debatable since sometimes it is unclear whether we are talking about offensive language or just the freedom of speech. For instance example 7: is this not just an ordinary opinion?

We believe that OLWOW is an interesting and demanding research topic. Although we have explicated some conditions and discussed some ideas how to operationalize OLWOW detection, we could not make it fruitful for the task at hand because of sparseness.

7 Filter-based Model: GermEval Runs

We submitted three runs in the coarse-grained task setting.

We have filters that classify tweets as OFFENSE (word indicator, exclamation mark, neg_actor, villain, victim) and filters that classify tweets as OTHER (pro, pos_affected).

Run 1 (cluzh_coarse1.txt') includes the filters (in that sequence): pro, pos_affected, pos_actor, word indicator, exclamation mark. Run 2 (cluzh_coarse2.txt') includes the filters (in that sequence): word indicator, exlamation mark, neg_actor, villain and victim. Run 3 (cluzh_coarse1.txt') only includes the word indicator filter.

Tweets that pass all filters are classified as OTHER. We did not use the filters con, neg_affected, benefactor, beneficiary. Also the filters from the last section were not part of any submission because of the sparseness problem.

8 Conclusion

We presented a plain vocabulary-based approach to the detection of offensive language. We realised a

cascade of filters including verb-based ones coming from stance analysis. We also focussed on a particular interesting research topic that we named OLWOW, offensive language without offensive words (known as implicit offensive language). We discussed ideas how to cope with it, pointed out problems with the annotation process of OLWOW and presented of a couple of examples our stance analysis system is able to cope with. We could, however, not exploit this notion for our shared task runs due to the sparseness of trigger conditions. We have, however, gained some insights that we will explore in our future work.

Acknowledgments

I would like to thank Michi Amsler for interesting discussions, useful word embeddings and a list of nice swearwords.

References

Manfred Klenner, Don Tuggener and Simon Clematide (2016). *Stance Detection in Facebook Posts of a German Right-wing Party.* In:LSDSem 2017/LSDSem Linking Models of Lexical, Sentential and Discourse-level Semantics, Valencia, 2017

J. R. Martin and P. R. R. White (2005). *Appraisal in English.* Palgrave, London, 2005

Rico Sennrich, Martin Volk and Gerold Schneider (2013). *Exploiting Synergies Between Open Resources for German Dependency Parsing, POS-tagging, and Morphological Analysis.* In: Proceedings of the International Conference Recent Advances in Natural Language Processing Hissar, Bulgaria, 2013

Michael Wiegand and Josef Ruppenhofer (2015). *Opinion Holder and Target Extraction based on the Induction of Verbal Categories.* Proceedings of the 19th Conference on Computational Natural Language Learning (CONLL) , Beijing, China, July 30-31, 2015

h_da Submission for the Germeval Shared Task on the Identification of Offensive Language

Melanie Siegel
Darmstadt University
of Applied Sciences
melanie.siegel@h-da.de

Markus Meyer
Darmstadt University
of Applied Sciences
markus.meyer@stud.h-da.de

Abstract

This paper describes the Darmstadt University of Applied Sciences (h_da) submission to the binary classification task of the Germeval Task 2018 - Shared Task on the Identification of Offensive Language. We submitted three runs, both a combination of lexical resources and an adapted sentiment analysis system. In run 1 (**hda_coarse_1.txt**) and 2 (**hda_coarse_2.txt**), we had a threshold-based approach (with different thresholds) and in run 3 (**hda_coarse_3.txt**), an approach based on machine learning.

1 Introduction

The social media such as Twitter, Facebook and also the commentary columns of the online presences of newspapers and radio stations are increasingly dominated by people who defame, insult and threaten. Automatically generated messages are also used to give the impression that these extreme opinions are widespread among the population. The "Germeval Shared Task on the Identification of Offensive Language" tries to develop and compare methods that automatically recognize such statements. The special features of this shared task: It is the first competition of its kind that deals with German language, and it analyzes data from Twitter.

In summer semester 2018, we participated in the shared task with a group of students in the Information Science Bachelor's programme of Darmstadt University of Applied Sciences.

We have formed working groups covering the following areas:

- project management
- programming
- documentary
- resources
- methods in literature

An NLP task for German language is complex because most publicly available resources are made for the English language. Therefore, we had to develop or significantly expand some resources ourselves. The resources that are available for the German language are also mostly targeted at newspaper text. However, Twitter data differ considerably from newspaper texts in terms of language, so we also had to make adjustments here.

The Germeval Shared Task consists of two subtasks: Task 1 is a binary classification of tweets into the categories OFFENSIVE and OTHER. In task 2, the tweets of the class OFFENSIVE are further classified in PROFANITY, ABUSE and INSULT. We worked on task 1, the binary classification, taking the PROFANITY class of task 2 into account. We also did some preliminary work on task 2 that we describe in section 5.

We have chosen a combination of a lexical approach and a sentiment analysis approach. For the lexical approach we have created resources mainly from the training data of the shared task. The sentiment analysis program was created in previous projects for Amazon product reviews and had to be adapted to Twitter data.

2 Resources

We were able to build on and adapt some existing dictionaries and have created some additional dictionaries based on the training data.

2.1 Sentiment Lexicon

In previous projects for sentiment analysis we have created a comprehensive sentiment dictionary with approx. 7800 entries and polarity measurements from Amazon product reviews and cinema reviews. We have applied this to the training data of the shared task. We looked at the cases where tweets

were recognized as positive or neutral, although they were marked as OFFENSIVE. OFFENSIVE statements are not always negative opinions, as in this example from the training data:

(1) Irgendwie verständlich daß Berlusconi diesen #Schulz mit einem KZ-Aufseher verglich.
(Somehow understandable that Berlusconi compared this #Schulz with a concentration camp guard.)

In many other cases, however, words were missing from the sentiment lexicon, which we could add. This lexicon now contains 9385 words.

2.2 Offensive Words

To create a lexicon of offensive words, we used the training data. We divided the data into tweets marked with 'OFFENSIVE' and those marked with 'OTHER'. We then made a list of the tokens in these tweets. We omitted the names (tokens beginning with '@') and hashtags (tokens beginning with '#'). We have also filtered stop words such as articles or pronouns. Then, we included in the list of offensive words all words that occurred only in the OFFENSIVE data, but not in the OTHER data. The result is a list of 3536 words.

2.3 Profane Words

To get a list of profane words, we proceeded in a similar way as with the offensive words. We divided the training data into those marked with 'PROFANITY' and extracted the words that only appeared in the tweets marked with 'PROFANITY', in the way described above. With 57 entries, this list is significantly smaller than the other lists. This is because there are only 71 tweets in the training data marked 'PROFANITY'.

3 Classifying Tweets

We have opted for a combined approach of sentiment analysis and lexicon-based analysis. We have worked with both machine learning and a threshold-based approach, both of which use the same resources. This allows us to compare the two approaches. Figure 1 shows the pipeline of our approach.

3.1 Preprocessing

In preprocessing, we first delete special characters such as Emojis. In the second step we analyze

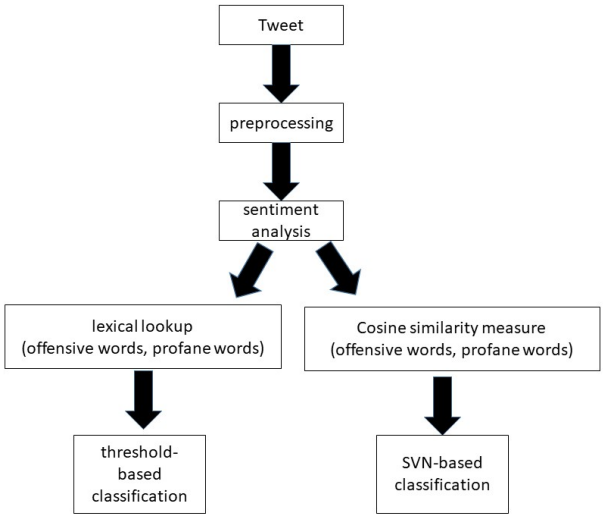


Figure 1: Pipeline of h_da classification

Lemma	POS	DEP	Head.Text	Text
Abschieben	VERB	sb	sind	Abschieben
ich	PRON	sb	Abschieben	es
sein	AUX	ROOT	sind	sind
doch	ADV	mo	sind	doch
nur	ADV	mo	Moslems	nur
Moslem	NOUN	pd	sind	Moslems
!	PUNCT	punct	sind	!
!	PUNCT	punct	sind	!

Table 1: Spacy analysis of
'Abschieben es sind doch nur Moslems!!'
(Deport! They're only Muslims!!)

the tweet with the Python module Spacy, such that we get tokens, POS information, lemmas and also dependencies.

Table 1 shows the spacy analysis of a tweet from the training set.

3.2 Sentiment Analysis

The purpose of sentiment analysis is to find out whether a tweet contains a strong negative expression of opinion or a rather positive expression of opinion. In the case of a positive expression of opinion it is rarely a case of ABUSE or INSULT, while PROFANITY is quite common. Therefore, although we worked mainly on task 1 we took PROFANITY into account.

As mentioned earlier, we used and customized a sentiment analysis program designed for Amazon product reviews. It is a Python program that compares words in the text with a sentiment lexicon and includes negations (e.g. 'nicht' - not) and intensifiers (e.g. 'sehr' - very) by using Spacy

dependencies to determine the scope.

In addition to the extension of the sentiment lexicon, however, further adjustment was necessary: While it makes sense to exclude conditional clauses and questions in sentiment analysis of product reviews, this does not make sense for tweets:

(2) Kennt jemand ein gutes Autoradio?
(Does anyone know a good car radio?)
Example for a question in product reviews

(3) Kann man diesen ganzen Scheiß noch glauben..?
(Can you believe all this shit...?)
Example for a question in tweets

The calculated numerical sentiment values are positive if the expression is positive, negative if the expression is negative and 0 if the expression is neutral.

3.3 Lexical Lookup

The lexical lookup in the lexicons for offensive words and for profane words was realized in two ways. In both cases we compare the words in lower case, because the upper and lower case is not standardized in tweets. In the first case, we check for a tweet how many words can be found in the respective lexicon and output them as numerical values. The disadvantage of this method is that these numerical values can lie between 0 and potentially the number of tokens in the tweet, which means that no value range can be determined.

In the second case we use the Cosine coefficient to calculate the similarity of a tweet with the words in the lexicons. This procedure was described among others by Liu (2007). Here the tweet is broken down into a set of tokens, which is then compared with the lexicon of offensive words. The return value is a floating point number between 0 and 1, where 0 means that there is no match to the words in the dictionary and 1 means that all words match. The disadvantage of the Cosine similarity is that it exhibits high fluctuations, even if the initial data only marginally increase or decrease in size. In addition, the size of the comparative data influences the similarity evaluation.

3.4 Threshold-Based Classification

In experiments with the training data, we set limits for sentiment and words in the lexicons. There are three values that are combined with each other: The sentiment value, the offense value and the profanity value. If the sentiment value is higher than 1 (i.e. if the utterance is strongly positive), then the offense value gets the value '0', since strongly positive utterances are rarely offensive. However, the profanity value is still calculated, i.e. it is checked how many profane words are contained in the tweet. If the sentiment value is less than or equal to a threshold, then the offense and profanity values are calculated. In the classification of tweets in OFFENSE and OTHER, all tweets with an offense value higher than a threshold and a profanity value higher than a threshold were marked as OFFENSE, all others as OTHER.

	000	**100**	**010**	**111**	**110**
Accuracy	0.82	0.91	0.81	0.92	0.93

Table 2: Accuracy on Training Data with Different Thresholds

Table 2 shows the accuracy values for different threshold combinations:

- 000: offense_value > 0, sentiment_value > 0, profanity_value > 0
- 100: offense_value > 1, sentiment_value > 0, profanity_value > 0
- 010: offense_value > 0, sentiment_value > 1, profanity_value > 0
- 111: offense_value > 1, sentiment_value > 1, profanity_value > 1
- 110: offense_value > 1, sentiment_value > 1, profanity_value > 0

We decided to submit two runs for the threshold-based approach: hda_coarse_1.txt (010) and hda_coarse_2.txt (110).

3.5 Classification Based on Supervised Learning

First, we identified the features that are useful for machine learning. Then, we preprocessed the outcomes of these features, using sklearn's Min-Max Scaler[1], because of their different types of return values. These features are the values for sentiment and the Cosine similarity measure of the tweet on the offensive and the profanity lexicons.[2] We trained an RBF SVM as part of sklearn's library on

[1] `http://scikit-learn.org`

[2] Here as well, we took PROFANITY into account, though we worked on task 1.

the first 80% of the training data (4007 tweets). The remaining 20% (1002 tweets) were classified by the trained model. The result (for task 1, training data) is an accuracy of 73.25 on these tweets.

Furthermore we experimented with two more classifiers, namely decision trees and linear-based SVM, which are also part of the python sklearn-library.

Decision trees require a balanced frequency destribution of classes to avoid overfitting, which was a problem in this shared task, as the distribution of PROFANITY Tweets is exceedingly lower than the distribution of INSULT and ABUSE tweets. A reduction of INSULT and ABUSE tweets for reasons of balancing resulted in a precion loss of the decision tree, therefore we dismissed the usage of decision trees.

The linear-based SVM on the other hand resulted in slightly higher f-scores, almost as high as the RBF SVM. However, the implementation of a linear-based SVM uses a random generator to weight the features, making predictions non-deterministic. For that reason we dismissed the usage of a linear-based SVM.

4 Error Analysis

We try to analyse the error sources that lead to offensive tweets not being detected or to non-offensive tweets being classified as offensive. The training data consists of 5009 tweets. Of these, 4675 were correctly classified. 138 tweets have been classified as OTHER, although they fall under the OFFENSE category. 196 Tweets were classified as OFFENSE although they fall under the category OTHER.

4.1 Classified as OTHER Although OFFENSE

Of the 138 tweets falsely detected as OTHER, one was PROFANITY, 42 INSULT and 95 ABUSE. In the PROFANITY case, a high degree of contextual knowledge is required to recognize it:

(4) Wie viel Oblaten muss ich denn jetzt essen bis ich ein Steak von Jesus zusammen hab?
(How many wafers do I have to eat now until I have a steak of Jesus together?)

Most cases of PROFANITY are clearly recognizable by the use of profane words. We have recorded these quite well for the training data. We are curious whether this will also suffice for the analysis of the test data.

Of the 42 INSULT cases that were not recognized, 11 need very complex background information that we could not model. In one case, the insult was part of a hashtag ('#erdoganistderhass'). However, we excluded hashtags from our investigation so far. One case was a positive expression of opinion in which the insult happened on the side:

(5) @Riedegost Dem stimme ich vorbehaltlos zu Ralf. Wenn Merkel nur halb so viel Verstand hätte, wie Du, oder wie Mecklenburger
(@Riedegost I agree wholeheartedly with Ralf. If Merkel had only half as much brain as you, or as Mecklenburger)

In another case, spaces were missing, so the tokenization failed. The remaining 28 cases could potentially be solved with further lexicon entries.

Of the 95 ABUSE cases, two were not recognized because important words were abbreviated:

(6) @AkifPirincci Es gibt in Deutschland nur eine Art von Flüchtling und das ist der Wirtschaftsflü. ! Alle Kriegsfl. sind durch sichere 3tstaat. gereist!
(@AkifPirincci There is only one kind of refugee in Germany and that is the economic ref. ! All war ref. have traveled through safe 3rdcount.)

In 46 cases, a classification requires substantial background information, as in this example:

(7) Warum soll die natürliche Selektion in Afrika bekämpft werden? Zu |LBR| viele Menschen haben eben nicht genug zu essen. Geburtenkontrolle!
(Why fight against natural selection in Africa? A lot of people don't have enough to eat. Birth control!)

In 6 cases the offensive word is part of a hashtag and in 8 cases the expression is positive:

(8) @IAMMASCHO Hitler war auch nicht absolut böse sondern hat viel gutes auch gemacht
(@IAMMASCHO Hitler was also not absolutely evil but has done a lot of positive things too)

In the remaining 33 cases, we hope to improve

detection with lexicon work.

5 First Steps on the Fine-Grained Classification: Targets of Hate

We have carried out initial work necessary for the fine-grained classification (task 2). The PROFANITY classification was already necessary for the binary classification task.

To distinguish the tweets in ABUSIVE and INSULT, it is necessary to recognize the targets of hate. While in the case of INSULT the targets are individuals, in the case of ABUSIVE they are groups of people, or the membership of a person in a group is targeted.

The annotation guidelines of the Shared Task states [3]:

> In contrast to insults, instances of abusive language require that the target of judgment is seen as a representative of a group and it is ascribed negative qualities that are taken to be universal, omnipresent and unchangeable characteristics of the group.

Therefore, we first extracted all named entities that appear in the tweets marked with OFFENSIVE from the training data. For this we used the Python package Spacy[4]. We also looked for other nouns, verbs and adjectives that appeared in tweets together with these named entities. The result is a list of 187 named entities with their cooccurring offensive words. The next step was to distinguish whether the addressee of the offensive expression represents an individual or a group. So we had to determine the semantic number. This may well differ from the syntactic number, as in e.g., 'Lügenpresse' (fake news), where syntactically it is a singular and semantically a group name, because the expression stands for a group of journalists and media representatives. Also 'Islam' (Islam) is syntactically singular, but semantically (especially in this context) often describes a group of people with Islamic faith. We made this distinction manually. In addition, we extracted 16,600 nouns from the German TIGER corpus (Brants et al., 2004) and stored them with their syntactic numbers. We did not proceed to work on the fine-grained task, though, because of time limits.

[3] http://www.coli.uni-saarland.de/~miwieg/Germeval/guidelines-iggsa-shared.pdf

[4] https://spacy.io/

6 Conclusion and Future Work

In this paper we presented the contribution of Darmstadt University of Applied Sciences to the first task of the Germeval Shared Task on the Identification of Offensive Language. This first task is a binary classification of tweets into the classes OFFENSIVE and OTHER.

Our approach combines lexical resources lookup with rule-based sentiment analysis. Together with a group of students we built up lexical resources, partly manually and partly automatically extracted from the training data, and adapted the existing sentiment analysis tool to the training data material. With these resources and results, we applied a threshold-based approach (**hda_coarse_1.txt** and **hda_coarse_2.txt**) and a machine learning-based approach (**hda_coarse_3.txt**).

As a next step, we want to expand the lexical resources - also with the test data - and thus refine the detection. We also want to work on task 2 - the fine-grained classification. Due to time constraints, we were unable to complete the work on this part of the project. First steps in this direction have been taken: We already recognize PROFANITY quite reliably in tweets and have developed first approaches for distinguishing between ABUSE and INSULT. Further development could be part of a final thesis in the Information Science programme at Darmstadt University of Applied Sciences.

Acknowledgments

We would like to thank Ante Bilic, Rio Fiore, Chris Gernand, Sascha Haas, Tahseena Khan, Vera Khramova, Kjell Kunz, Felix Marechal, Johanna Pfau, and Nadia Shah, who attended Melanie Siegel's course on "Advanced Methods in Natural Language Processing" at Darmstadt University of Applied Sciences and created the very first version on which these results are based.

References

S. Brants, S. Dipper, P. Eisenberg, S. Hansen-Schirra, E. König, W. Lezius, C. Rohrer, G. Smith, and H. Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on language and computation*, 2(4):597–620.

Bing Liu. 2007. *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media.

Saarland University's Participation in the GermEval Task 2018 (UdSW) – Examining Different Types of Classifiers and Features

Michael Wiegand, Anastasija Amann, Tatiana Anikina, Aikaterini Azoidou, Anastasia Borisenkov, Kirstin Kolmorgen, Insa Kröger, Christine Schäfer
Spoken Language Systems
Saarland University
D-66123, Saarbrücken, Germany
michael.wiegand@lsv.uni-saarland.de

Abstract

We report on our participation in *GermEval Task 2018 – Shared Task on the Identification of Offensive Language*. In our submission we considered both lexicon-based approaches and supervised learning. We experimented with both monolingual and crosslingual information. We compared traditional SVMs with the more recent neural networks.

1 Introduction

We report on our submission for *GermEval Task 2018 – Shared Task on the Identification of Offensive Language*. We participated in Task I, the binary classification task distinguishing offensive from non-offensive tweets.

The choice of our approach is mostly guided by the methods that have previously been reported effective on English data (Schmidt and Wiegand, 2017). In our submission we considered both lexicon-based approaches and supervised learning. We compared traditional SVMs with the more recent neural networks.

Since this is the first shared task on German data, there are only very few task-specific resources for German. This is why we also experimented with crosslingual information that takes into account English data.

2 The Different Classification Approaches

2.1 Task-specific Lexicon

A popular resource for text-classification tasks is a task-specific lexicon, i.e. a list of words predictive for the classes to be detected. With regard to the detection of abusive language, one typically uses a list of explicitly abusive words (e.g. *cunt*, *idiot*, *nigger*). Such lexicons can be easily converted into a text classifier. One predicts a comment to be abusive in case at least one of the words included in the task-specific lexicon is found in the comment.

Though lexicon-based approaches are, by design, restricted and unable to detect certain subtypes of abusive language, such as *implicit* abuse (Waseem et al., 2017), they are fairly robust when it comes to cross-domain evaluations (Wiegand et al., 2018). The reason for this is that, unlike many other classifiers, they are less susceptible to overfit to some specific training data. Since we report on building a classifier for the first edition of a shared task and only a limited amount of training data have been released, we may always run the risk of overfitting when applying supervised learning. A lexicon-based approach may be a safer alternative.

Since we are not aware of any comprehensive publicly available lexicon with abusive words for German, we created a lexicon ourselves. The lexicon was created semi-automatically. We first started with the large bootstrapped English lexicon from Wiegand et al. (2018) which had been extensively evaluated on several English datasets for detecting abusive language. This lexicon was automatically translated into German with *Google Translate*.[1] The result was manually edited. For more than half of the entries no appropriate German translation was found. These entries were removed from the German lexicon. We added abusive words we could extract from the German version of Wiktionary[2] using the Wiktionary-API JWKTL (Zesch et al., 2008). We mainly focused on those entries that contained some predictive *word-usage* tag, e.g. *abwertend (pejorative)* or *beleidigend (offensive)*. Figure 1 illustrates such a tag in the entry of the abusive word *Vollidiot (wally)*. In order to further increase the coverage, we also added the entries linked as synonyms to these expressions. Again, the resulting list was manually filtered. We also

[1] https://translate.google.com/

[2] https://de.wiktionary.org/wiki/Wiktionary:Hauptseite

Figure 1: Illustration of Wiktionary-entry of the abusive word *Vollidiot (wally)* with its word-usage tag *beleidigend (offensive)*.

extracted words that possess a high distributional similarity with the words from our lexicon. Distributional similarity was computed on the basis cosine-similarity of the word embeddings induced on Twitter released by Heidelberg University.[3] By using embeddings from Twitter, we hope to include some more domain-specific information. However, this step only resulted in a meagre yield of less than 100 additional words. Our final lexicon contains 1566 (unigram) entries.

In order to increase the coverage of our lexicon, we also implemented a soft-matching function that is more flexible than strict token matching. We compared prefix-matching, suffix-matching, infix-matching and some combinations. We found that prefix-matching works best. We therefore decided to use this in our final system. We also investigated whether even more flexible matching might increase classification performance. However, after running some experiments with Levenshtein-distance that turned out not to be effective, we abandoned these experiments.

[3] http://www.cl.uni-heidelberg.de/english/research/downloads/resource_pages/GermanTwitterEmbeddings/GermanTwitterEmbeddings_data.shtml

2.2 Traditional Supervised Learning with Feature Engineering

Schmidt and Wiegand (2017) report that traditional supervised-learning methods, particularly, SVMs are still the most frequently used classification approach for the detection of abusive language. This is why we also took this approach into account in our participation. As a tool, we used LIBLINEAR.[4] Our choice of features is mostly inspired by the feature set proposed by Nobata et al. (2016), particularly since Wiegand et al. (2018) report it to be the most effective classification approach on four established datasets (for in-domain evaluation).

The specific features we explored are displayed in Table 1. Regarding word embeddings for German, we experimented with the pre-trained embeddings induced from Twitter released by Heidelberg University.[3] In addition, we also induced embeddings from German ourselves using *Web As Corpus* (Baroni et al., 2009) and *COW16* (Schäfer and Bildhauer, 2012; Schäfer, 2015). For induction we employed *Word2Vec* (Mikolov et al., 2013) in its standard configuration. (With regard to vector dimensions, we tested 100, 200 and 500 dimensions.) In order to obtain a vector representation based on embeddings of an entire tweet, we simply averaged the word embeddings of the words found in the tweet.

We tested various combinations of different feature sets from Table 1. For those experiments, we divided the training data from GermEval into further subsets (see also §3.1). With regard to word embeddings, we always got best performance with the highest dimensional embeddings that were available to us (i.e. 500 dimensions). We found that only the subset of the features comprising character n-grams, word embeddings and our task-specific lexicon (§2.1) is actually necessary. In the official evaluation of the shared task, we, therefore, took only these features into account. In combination with other features, the most effective embeddings turned out to be the ones induced on COW16. We ascribe it to the fact that this is by far the largest corpus which we used for embedding induction.

2.3 Neural Networks

We considered two types of standard network architectures: Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). As an implementa-

[4] https://www.csie.ntu.edu.tw/~cjlin/liblinear/

Feature	Further Comments	Used in the Official Run?
bag of words	lemmatized unigrams	no
part-of-speech information	no. of nouns, adjs, verbs	no
surface features	looking for suspicious words (e.g. *b*tch* or *fxck*)	no
character ngrams	$n = 6$	yes
word-embeddings	COW16, 500 dimensions	yes
prediction of task-specific lexicon	we use the lexicon from §2.1	yes

Table 1: Feature set used for traditional supervised learning.

tion we mainly relied on the pre-implemented networks from *Keras*.[5] The following hyperparameters were optimized using an informal development set that we split off from the GermEval-training data (§3.1):

- types of embeddings[6]
- activation function
- batch size
- drop-out rate
- number of epochs
- optimizer

2.4 Crosslingual Approaches

One challenge of the setting of the shared task is that only a limited amount of labelled training data has been made available. For English, however, there meanwhile exists a plethora of different datasets. Some of them are also fairly large comprising more than 100K labelled instances. Therefore, we also wanted to examine whether we can leverage large collections of labelled training data from English for the present task. As English datasets, we considered the datasets from Waseem and Hovy (2016) and from Wulczyn et al. (2017). The former focuses on sexism and racism, particularly Islamophobia. That dataset may be suitable since we observed that the abusive tweets from the training collection of the GermEval-shared task also predominantly address Islamic migrants. Moreover, like the GermEval-data, this corpus exclusively comprises tweets from Twitter. The dataset from Wulczyn et al. (2017), which consists of Wikipedia comments, on the other hand, was chosen because of its size (the entire collection contains about 115K comments – the dataset by Waseem only 16K tweets).

We considered two different approaches:

Translation-based approach. On the one hand, we simply automatically translated the existing English datasets to German (again with *Google Translate*) and trained classifiers on the translated datasets. The supervised classifier was trained as a typical monolingual classifier with the most effective embeddings (COW16) as features. The resulting architecture is illustrated in Figure 2.

Crosslingual embeddings. On the other hand, we considered crosslingual embeddings. These are embeddings that represent words in two languages, in our case, English and German, in the same embedding space. The embedding space is induced in such a way that two words from the different languages that have either similar or identical meaning (e.g. *Dummkopf* and *blockhead*) should have similar word embeddings. Such a representation allows us to train on the original English data directly (i.e. without translating them into German) and test them on the German tweets from the shared task. This pipeline is illustrated in Figure 3. In order to obtain crosslingual embeddings, we used the publicly available software *VecMap* (Artetxe et al., 2017). This method does not even require parallel corpora but only two large monolingual corpora.[7] We chose the *Amazon Review Corpus* (Jindal and Liu, 2008) for English and again *Web As Corpus* for German.

For both crosslingual approaches we always trained on our German GermEval-training data and added some English dataset (either in its original version or translated into German). Our interest therefore exclusively lies in improving classification performance of a monolingual (German) classifier with additional crosslingual information.

3 Experiments

3.1 Experiment Set-Up

For the preparation of the shared task, we randomly split the official training data of the GermEval-2018 shared task into three sets:

[5] https://keras.io/

[6] We considered the same embeddings as in §2.2.

[7] The seed word-alignment happens via numerals which are identical in both languages.

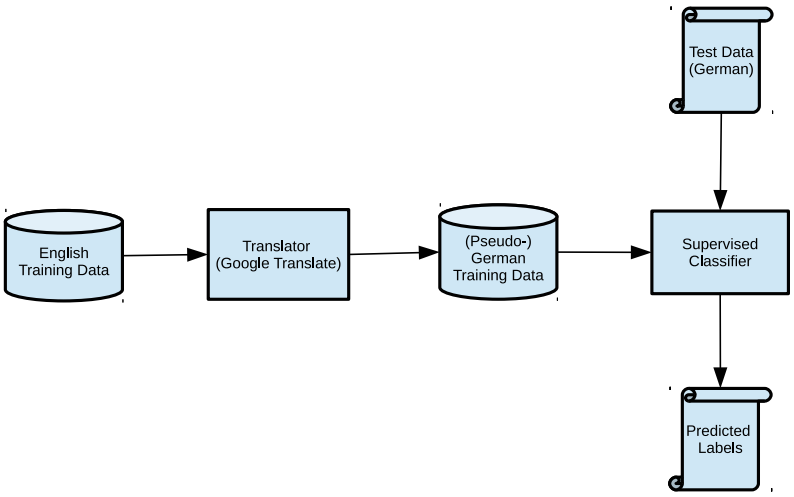


Figure 2: Illustration of translation-based classifier.

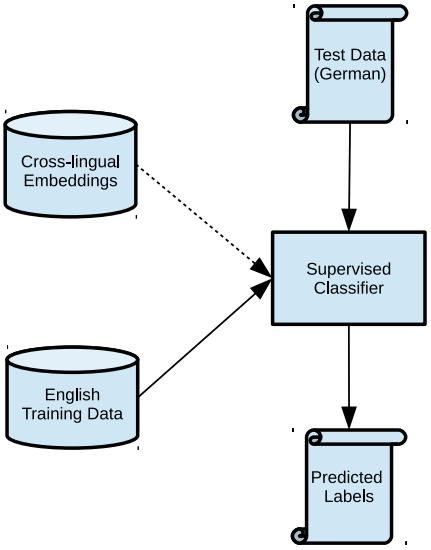


Figure 3: Illustration of translation-based classifier using crosslingual embeddings.

- 3009 tweets were used as a training set.
- 1000 tweets were used as a development set.
- 1000 tweets were used as a test set.

In the following we report some preliminary evaluation on our informal test set. For the crosslingual experiments, we downsampled the English datasets and the additional German datasets translated from English so that their class distribution resembles that of the GermEval-training data. We evaluated with the evaluation tool provided by the shared task. We report macro-average precision, recall and f-measure.

3.2 Results

Table 2 shows the performance of best classifiers of our different approaches on our informal test set. The best crosslingual approach (i.e. the translation-based approach) scores lowest. The F-score of the lexicon at 71.2% is quite respectable given that it was not specifically tuned on the available GermEval data. The best neural network (GRU) scores reasonably but lower than the SVM. This result is reminiscent of the in-domain evaluation from Wiegand et al. (2018). Obviously, the SVM also benefits from features other than embeddings (i.e. character n-grams and the task-specific lexicon) to which the neural networks do not have access.

Table 3 sheds more light on the behaviour of the two crosslingual approaches. We also include monolingual baselines. It is interesting to note that for the translation-based approach better results are obtained by training from the dataset by Wulczyn

Feature	Prec	Rec	F1
crosslingual (translation-based)	69.5	71.0	70.2
lexicon	73.8	69.1	71.4
deep learning (GRU)	75.1	72.1	73.5
SVM	79.7	75.2	77.4

Table 2: Comparison of different classifiers.

Feature	Prec	Rec	F1
LSTM	67.5	66.8	67.2
GRU	75.1	72.1	73.5

Table 4: Comparison of different neural networks.

et al. (2017) while for crosslingual embeddings, we obtain better results by training on the dataset by Waseem and Hovy (2016). However, for both approaches, we actually observe that each time when we add some English training data, the performance score slightly decreases. This means that none of the English datasets is really helpful for this type of classification.

Table 4 compares the two different neural networks we experimented with. GRUs outperform LSTMs. Given that we only have a limited amount of training data, this result does not come as a surprise. LSTMs are more complex in design and require more parameters to be optimized. Obviously, a simpler model is more suitable for this task.

Table 5 examines the feature set of the SVM more closely. All of these three features when evaluated individually produce very similar scores. However, since their combination results in an increase by approximately 7% points, we assume that the information contained in those different features is complementary to some extent.

4 Description of the Submitted Runs

We submitted three runs. The configurations are as follows.

Feature	Prec	Rec	F1
character ngrams	67.4	69.3	68.3
embeddings	70.1	68.6	69.4
lexicon	73.8	69.1	71.4
char. ngrams + embed. + lexicon	79.7	75.2	77.4

Table 5: SVM with different feature sets.

4.1 Run I (`UdSW_coarse_1`) – Lexicon-based Approach

In our first run, we employed the full lexicon we described in §2.1.[8] We chose this configuration since Wiegand et al. (2018) have shown that lexicon-based classification is usually the safest bet for cross-domain detection of abusive language because it is less susceptible to overfitting. In shared tasks like GermEval, where only limited training data are available, there is always the risk for supervised classifiers to overfit to the given training data.

4.2 Run II (`UdSW_coarse_2`) – SVM with Large Feature Set

In our second run, we employed an SVM with the feature set described in Table 1. From all classifiers we tested on our informal test set, we achieved, by far, the highest performance scores with this approach.

4.3 Run III (`UdSW_coarse_3`) – Ensemble

In our third run, we combined the output of all individual classifiers from Table 2. Since we do not have any further training data from which to learn a combination of those classifiers, we simply created a classifier that takes the majority vote of the predictions made by the individual classifiers. This run should be considered a *wild guess*.

5 Conclusion

We presented our submission for GermEval Task 2018 – Shared Task on the Identification of Offensive Language. We participated in Task I, the binary classification task distinguishing offensive from non-offensive tweets. We experimented with lexicon-based classification, supervised learning with traditional feature engineering, crosslingual classification and deep learning. On our informal test set, we obtained the best performance scores with supervised learning using traditional feature engineering using a task-specific lexicon, character n-grams and word embeddings.

Acknowledgements

The authors would like to thank Marc Schulder for providing the crosslingual embeddings used in the experiments presented in this paper. Michael Wiegand was partially supported by the

[8] To be precise, we ran our SVM with just the prediction of our task-specific lexicon as a feature.

Feature	Prec	Rec	F1
crosslingual embeddings: GermEval (monolingual baseline)	70.0	71.6	70.8
crosslingual embeddings: GermEval+Wulczyn	57.4	56.0	56.7
crosslingual embeddings: GermEval+Waseem	69.8	68.3	69.0
translation-based: GermEval (monolingual baseline)	70.7	72.2	71.4
translation-based: GermEval+Wulczyn	69.5	71.0	70.2
translation-based: GermEval+Waseem	59.6	58.5	59.0

Table 3: Comparison of different crosslingual classifiers.

German Research Foundation (DFG) under grant WI 4204/2-1.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 451–462, Vancouver, Canada.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetti. 2009. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora. *Language Resources and Evaluation*, 43(3):209–226.

Nitin Jindal and Bing Liu. 2008. Opinion Spam and Analysis. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 219–230, Palo Alto, CA, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at the International Conference on Learning Representations (ICLR)*, Scottsdale, AZ, USA.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 145–153, Republic and Canton of Geneva, Switzerland.

Roland Schäfer and Felix Bildhauer. 2012. Building Large Corpora from the Web Using a New Efficient Tool Chain. pages 486–493, Instanbul, Turkey.

Roland Schäfer. 2015. Processing and querying large web corpora with the COW14 architecture. In *Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC)*, pages 28–34, Lancaster, United Kingdom.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the EACL-Workshop on Natural Language Processing for Social Media (SocialNLP)*, pages 1–10, Valencia, Spain.

Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL – Student Research Workshop)*, pages 88–93, San Diego, CA, USA.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the ACL-Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada.

Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018. Inducing a Lexicon of Abusive Words – A Feature-Based Approach. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, pages 1046–1056, New Orleans, LA, USA.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 1391–1399, Perth, Australia.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 1646–1652, Marrakech, Morocco.

Challenges of Automatically Detecting Offensive Language Online: Participation Paper for the Germeval Shared Task 2018 (HaUA)

Tom De Smedt
University of Antwerp
Computational Linguistics Research Group
Experimental Media Research Group
tom.desmedt@uantwerpen.be

Sylvia Jaki
University of Hildesheim
Department of Translation and
Specialized Communication
jakisy@uni-hildesheim.de

Abstract

This paper presents our submission (HaUA) for Germeval Shared Task 1 (Binary Classification) on the identification of offensive language. With feature selection and features such as character n-grams, offensive word lexicons, and sentiment polarity, our SVM classifier is able to distinguish between offensive and non-offensive German-language tweets with an in-domain F_1 score of 88.9%. In this paper, we report our methodology and discuss machine learning problems such as imbalance, overfitting, and the interpretability of machine learning algorithms. In the discussion section, we also briefly go beyond the technical perspectives and argue for a thorough discussion of the dilemma between internet security and freedom of speech, and what kind of language we are actually predicting with such algorithms.

1 Introduction

The new German Netzwerkdurchsetzungsgesetz law (NetzDG) allows for the removal of illegal content posted on social media platforms, where *illegal* pertains to one of 21 elements of offense according to the German Strafgesetzbuch. Recent reports expose several points of interest (Brühl and von Au, 2018). Firstly, the most common reasons for suspension on Twitter are incitement to hatred (§130), insults (§185), unconstitutional symbols (§86a), incitement to crime (§111), and pornography (§184). Secondly, only a fraction of the reported content has been blocked (11%, or 28,645 out of 264,828 tweets). Thirdly, primarily relating to Facebook, the decision-making is not transparent, with various reported cases of under- and overblocking. As a result, it is not surprising that many people feel that the current situation, in which for-profit IT companies independently decide what should be removed, is undesirable.

The recent surge of workshops on offensive language such as this year's Shared Task, and the large number of participants, reveals a commitment of the linguistics community to collaborate towards a safer internet, by providing algorithms that can help to detect abusive content online. In this workshop, comparing approaches, methods, and opinions will foster advances in the long run, which may be useful to German policy makers and human-rights organizations to counter online polarization and the proliferation of hate.

In our contribution, we have paid attention to the ethical consequences of releasing AI in the wild. We can offer a model that is not perfect, but interpretable. In section 2, we will discuss a brief analysis of the training data. In section 3, we will discuss the (unknown) test data and how we have approximated it by in-domain and cross-domain evaluation. We will then describe our algorithm in section 4, and zoom in on the model's features in section 5 and methods for feature selection in section 6. After the technical report, we briefly discuss some implications of our approach and challenges that, as of yet, cannot be solved with automatic NLP techniques alone in section 7.

Figure 1a: Example OFFENSE tweet.

Figure 1b: Example OTHER tweet.

2 Training Data

The training data for the Shared Task consists of 5,009 manually annotated German tweets, each about 70-210 characters long, of which 1,688 are labeled OFFENSE (33.7% or about 1/3) and 3,321 are labeled OTHER (66.3% or about 2/3). Tweets labeled OFFENSE use offensive language (Fig 1).

2.1 Data Distribution

The training data is imbalanced (1:2 ratio), which reflects reality – assuming most Twitter users will not post offensive tweets – but which can also be problematic, since classifiers tend to be "overwhelmed" by the majority class (Chawla et al., 2004). Solutions for imbalanced data that are reported to be effective include undersampling, i.e., discarding training examples of the majority class until the data is balanced, oversampling, e.g., training on examples of the minority class multiple times, and feature selection, removing ambiguous features to increase the separability of the classes. We tested with both undersampling and oversampling as well as feature selection, where oversampling + feature selection seems to work best in our case (~ +5% F_1 score).

2.2 Data Entropy

Ideally, a given machine learning algorithm will automatically discover features in the training data that can be used to predict whether unknown tweets are OFFENSE or OTHER. Such features might be words like *Scheiße* that are statistically biased, i.e., occurring more often in offensive tweets. To get a sense of the biased words in the data, we used the chi-squared test ($p \leq 0.01$; see also Liu and Motoda, 2007) with word counts per class to expose them. The results[1] are in line with what we observed in previous work on German far-right propaganda (Jaki and De Smedt, 2018) and jihadist extremism (De Smedt et al., 2018).

Broadly, offensive tweets seem to be marked by:

- **defamation** (often political opponents), e.g., *Gutmensch*, *Nazi*, *Volksverräterin* (do-gooder, fascist, traitor of the people),
- **dehumanization** (refugees), e.g., *Abschaum*, *Pack*, *Schmarotzer* (scum, rabble, parasites),
- **stereotyping**, e.g., *Kanakenstadt*, *Museldiebe* (Turk town, Muslim thieves),
- **racism**, e.g., *Nafris*, *Neger* (North African repeat offenders, niggers),
- **profanity**, e.g., *Arsch*, *Dreck*, *Scheiße* (ass, crap, shit),
- **negativity**, e.g., *dumm*, *Gelaber*, *kotzen*, 🤢 (dumb, drivel, to vomit),
- **capitalization**, e.g., *DEUTSCH*, *ISLAM*, *LINKS* (German, Islam, left),
- **propaganda** and **fake news** posted by known user profiles (see Netzpolitik, 2017).

About 50% of the most biased nouns exposed by the chi-squared test occur in our automatically generated list of offensive words, which is an important feature in our model (see section 5).

3 Test Data

The Shared Task 1 entails a test dataset of 3,532 German tweets for which we have to accurately predict either OFFENSE or OTHER.

3.1 In-domain Evaluation

Various statistical techniques exist to predict how well our trained classifier is going to perform. Most notably, *k*-fold cross-validation partitions the training data into *k* training / test subsets and reports the average recall and precision, where recall is an estimation of *how many* offensive tweets are found, and precision is an estimation of how many reported offensive tweets *are really* offensive (henceforth called the IN evaluation). For example, a classifier with 75% recall finds 3/4 of offensive tweets (1/4 goes by undetected). A classifier with 75% precision mislabels 1/4 of "normal" tweets as offensive.

The main drawback of this approach is that it only reports in-domain performance, it assumes that unknown tweets on which the classifier will eventually be applied will have features identical to those in the training data, which may be false.

[1] https://docs.google.com/spreadsheets/d/1Q3fLs4mfjWEWYJtv26ddUd8jk_1Tz2svk-94LxtONwA

3.2 Cross-domain Evaluation

Domain adaptation refers to a machine learning problem where a classifier seems to perform well on its training data (in-domain performance) but not on related data (out-of-domain performance). To test the scalability of our classifier, we cut 500 tweets (~10%) from the training data as a holdout testing set, for which we know the class labels (henceforth called the OUT evaluation). Since we do not know the distribution of the class labels in the *actual* test data, we did three runs with the holdout set having respectively a 1:1 (250/250), 1:2 (150/300), and 1:4 (100/400) ratio of OFFENSE/OTHER tweets.

We also used a manually annotated subset of Jaki and De Smedt (2018) for testing (henceforth called the CROSS evaluation). This set consists of 800 German right-wing extremist tweets with offensive language + 1,600 other German tweets. The 1:2 ratio means that a classifier that always predicts OTHER (the majority class) would score F_1 44% on this data. We can use this as a baseline for our classifier (see also Table 2 & 3). In other words, it must score at least F_1 45% to have any predictive value.

4 Algorithm

We used the LIBSVM machine learning algorithm (Chang and Lin, 2011) in the Pattern toolkit for Python (De Smedt and Daelemans, 2012) to train our classifier, and Pattern helper functions.

4.1 Interpretability

No doubt, the most recent multi-layered neural networks ("Deep Learning") will achieve better results, especially in combination with word embeddings. The downside of deep neural nets is that their decision-making might be difficult to interpret (Lipton, 2016). This is problematic once such systems are applied in the wild: as of yet, there is still ongoing debate as to what exactly constitutes offensive language / hate speech, and laws such as NetzDG tend to be vague (Human Rights Watch, 2018). Introducing "black box" AI systems to the decision-making may be morally questionable and may jeopardize the freedom of expression (see section 7), particularly in light of the new privacy protection regulations in the EU (GDPR; European Commission, 2018).

By comparison, classic machine learning algorithms such as *k*-NN, decision trees, and linear SVMs are often more interpretable. In fact, in our tests a lexicon of offensive words with confidence scores (e.g., *autoritär* = 0.5) is only about 3% less accurate and might also be useful, e.g., offensive words can be visually highlighted for human moderators.

5 Features

The LIBSVM algorithm expects its input to be vectorized, where each tweet is represented as a vector of feature/weight pairs. The features could be the words that appear in the tweet and the weights could be word count. In our case, we use lexical features such as character trigrams, e.g., *Scheiß* → { *Sch*, *che*, *hei*, *eiß* }, and binary weights, i.e., a feature is present or not. Character n-grams efficiently capture spelling errors, word endings, function words, emoticons, and so on. For example, *Scheiß* and *Scheiss* have multiple matching trigrams (*Sch*, *che*, *hei*).

An overview of the features we used:

- each tweet is lowercased: *Dreck* → *dreck*,
- `C1`, character 1-grams, e.g., *d*, *r*, *e*, *c*, *k*,
- `C3`, character 3-grams, e.g., *dre*, *rec*, *eck*,
- `C5`, character 5-grams, e.g., *dreck*,
- `W1`, word 1-grams, e.g., *dreckiger*,
- `W2`, word 2-grams, e.g., *dreckiger neger*,
- `W3`, word 3-grams, e.g., *neger dürfen bleiben*,
- `UP`, if tweet has +40% uppercase characters,
- `!!`, if tweet has 2+ exclamation marks,
- `O?`, if tweet has an offensive word,
- `O+`, if tweet has 2+ offensive words,
- `O%`, if tweet has *autoritär* (for example) then a feature `O%50` will be present,
- `:(`, if tweet has a negative polarity.

Offensive words are those words that occur in our automatically generated lexicon of 1,750 words and their confidence scores. To populate the lexicon, we started with 50 high-precision seed words to which we assigned a score (e.g., *Abfall* = 0.50, *Arsch* = 0.75, *Gesindel* = 1.00) and then queried the German Twitter Embeddings (Ruppenhofer, 2018) to find semantically similar words (Mikolov et al., 2013).

For each seed word, we then took the 100 most similar words (*Gesindel* → 81% *Dreckspack*), propagated the seed score (1.00 x 0.81 = 0.81), and then assigned new words to one of five bins (`0.00 | 0.25 | 0.50 | 0.75 | 1.00`; e.g., *Dreckspack* = 0.75, *Schnurrbart* = 0.25).

Sentiment analysis (Pang and Lee, 2008) refers to the task of automatically detecting the polarity (positive or negative tone) of a text. Polarity was predicted using a Perceptron classifier trained on German tweets containing emoji from the POLLY corpus (De Smedt and Jaki, 2018). The model is about 85% accurate. For example, *sehr schöner Urlaub!* (very nice holiday!) is labeled positive while *islamgeile Propaganda* (Islam-loving propaganda) is labeled negative.

Using this set of features, the LIBSVM algorithm trained on the given data (1:2 OFFENSE/OTHER) yields recall 75.8% and precision 78.7% with in-domain 10-fold cross-validation.

Table 1 provides an overview of performance (i.e., F_1 score = mean of recall and precision) for different combinations of features. Interestingly, offensive words and shape features are nearly as predictive (**`O`** + **`UP`** + **`!!`** = 74.5%) as all features combined (77.2%). However, the best results are achieved by applying feature selection (FSEL, i.e., removing noisy features), which raises F_1 score from 77.2% to 88.9% (1 mistake per 10 tweets).

6 Feature Selection

Using this set of features, the trained model (after holdout) has about 250K features in total. Each tweet has about 350 features. To improve the performance for imbalanced data, we computed the posterior probability of each feature (e.g., *der* = 50% OFFENSE vs 50% OTHER, and *Dreckspack* = 100% OFFENSE vs 0% OTHER) and removed the most ambiguous ones with probabilities between 25% and 75% until each vector has at most 100 features. This removes about 50K features in total: 90% of **`C1`** (e.g., @ is too noisy), 50% of **`C3`**, 25% of **`W1`** (e.g., *skeptisch* is too noisy), 10% of **`W2`** (e.g., *und mit*), and so on.

6.1 Model Overfitting

This raises the F_1 score by about 10% for the IN evaluation, from 77.2% to 88.9% (recall 87.3% and precision 90.6%). We can remove even more features, eventually training a model that has 99% in-domain performance, but which also has no features left to fit out-of-domain data. This is known as overfitting (Hawkins, 2004). To assess whether we might be overfitting our classifier, we tested on the OUT and CROSS sets. In general, our feature selection method raises F_1 score by about 2% on the OUT set (with varying OFFENSE/OTHER distributions) and by 6% on the CROSS set (see Table 2 for an overview). Removing more features lowers the F_1 score on both sets.

6.2 Model Oversampling

We also experimented with undersampling and oversampling to boost performance. For given training data of ~1,500 OFFENSE + 3,000 OTHER tweets, we either removed 1,500 OTHER tweets (= undersampled 1500/1500) or trained OFFENSE tweets twice (= oversampled 3000/3000).

Table 2 provides an overview of performance (F_1) for the imbalanced and balanced classifiers, with or without feature selection (100 vs 350), on the in-domain (IN) and cross-domain tests (OUT set of 500 tweets, CROSS set of 800/1600 political tweets). Oversampling combined with feature selection works well if there are less OFFENSE than OTHER tweets. With a 1:4 ratio the F_1 score is about 76% on the OUT set, and about 70% on the CROSS set with a 1:2 ratio, which is above the 44% majority class baseline.

Table 3 provides an error analysis with recall and precision by class, as measured on the OUT 1:4 (100/400) test set, which we think is the most representative of real-life. Not surprisingly, most classification errors occur in the OFFENSE class. The best AUC score (Area Under Curve) is 0.83 for the oversampled model with feature selection.

This is the classifier that we submitted for the Shared Task 1 (`HaUA-coarse`).

7 Discussion

Our tests with domain adaptation highlight the importance of clearly defining what exactly we are detecting. To illustrate this: There is overlap between the task's training data and the CROSS data we used. Looking at retweeted usernames, both sets appear to draw from the same sources, but where the CROSS data focuses on politically motivated hate speech grounded in racism, the task's data focuses on disrespect and contempt of individuals and groups (who are not necessarily refugees or political factions). The difference is

subtle, and there is some overlap in performance, however it is not a perfect fit. The divergence is in part due to different views of what constitutes offensive language online. Profanity like *Scheiße* is unacceptable by Ruppenhofer et al. (2018: 2) while the CROSS data focuses more exclusively on ideologically disparaging language.

This stresses the need to discuss how we will operationalize regulations on a linguistic level. Which "bad content" should AI be detecting? Do we train systems according to society's norms of what is inappropriate, or legal definitions? This means that the challenge is not purely linguistic but also societal and political (cf. Ruppenhofer et al., 2018: 4). What language can we ethically and legally justify to remove from the internet?

There is little doubt that content classified as illegal by the German Strafgesetzbuch should be removed, justifying the need for AI tools. People who criticize NetzDG claim that it infringes on freedom of speech, which is anchored in German Grundgesetz, but they forget that these freedoms are also limited by StGB. Apart from such cases, there is admittedly a grey area between offensive language and freedom of speech. For example, what is the line between an expressed opinion of a foreign culture and incitement to hatred? To avoid the shadow of censorship, policy makers should not be satisfied with the current legal situation, but strive to continue the discussion about the boundaries of freedom of speech and the measures to take against offensive behavior on social media.

Acknowledgements

The authors wish to thank "Schmutzi" for giving insight into profanity, slurs and slang language in online social media.

References

Jannis Brühl and Caspar van Au. 2018. Was das NetzDG mit Deutschland macht. *Süddeutsche Zeitung*. https://www.sueddeutsche.de/digital/bilanz-was-das-netzdg-mit-deutschland-macht-1.4072480

Nitesh Chawla, Nathalie Japkowicz, and Aleksander Kotcz. 2004. Special issue on learning from imbalanced data sets. *ACM SIGKDD*, 6(1):1–6.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM TIST*, 2(3), 27.

Tom De Smedt and Walter Daelemans. 2012. Pattern for Python. *JMLR*, 13:2063–2067.

Tom De Smedt, Guy De Pauw, and Pieter Van Ostaeyen. 2018. Automatic detection of online jihadist hate speech. *CLiPS CTRS*, 7:1–30.

Tom De Smedt and Sylvia Jaki. 2018. The Polly corpus: online political debate in Germany. In *Proceedings of CMC and Social Media Corpora*.

Douglas M. Hawkins. 2004. The problem of overfitting. *ACS JCIM*, 44(1):1–12.

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

European Commission. 2018. Data protection. https://ec.europa.eu/info/law/law-topic/data-protection_en

Zachary C. Lipton. 2018. The mythos of model interpretability. *Queue*, 16(3).

Human Rights Watch. 2018. Germany: flawed social media law. https://www.hrw.org/news/2018/02/14/germany-flawed-social-media-law

Huan Liu and Hiroshi Motoda (eds.). 2007. *Computational methods of feature selection*. Chapman & Hall/CRC, Boca Raton.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1/2):1–135.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv:1301.3781

Netzpolitik. 2017. Datenrecherche: offizielle AfD-Accounts retweeten Neonazi-Kanal auf Twitter. https://netzpolitik.org/2017/datenrecherche-offizielle-afd-accounts-retweeten-neonazi-kanal-auf-twitter/

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

Josef Ruppenhofer, Melanie Siegel, and Michael Wiegand. 2018. Guidelines for IGGSA Shared Task on the Identification of Offensive Language. http://www.coli.uni-saarland.de/~miwieg/Germeval/guidelines-iggsa-shared.pdf

C1	C3	C5	W1	W2	W3	UP	!!	O?	O+	O%	:(	FSEL	F1
✓	-	-	-	-	-	-	-	-	-	-	-	-	59.1%
✓	✓	-	-	-	-	-	-	-	-	-	-	-	68.4%
✓	✓	✓	-	-	-	-	-	-	-	-	-	-	72.5%
✓	✓	✓	✓	-	-	-	-	-	-	-	-	-	73.3%
✓	✓	✓	✓	✓	-	-	-	-	-	-	-	-	73.1%
✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	-	73.4%
✓	✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	73.8%
✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	-	-	74.0%
✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	-	74.9%
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	76.2%
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	77.2%
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	**88.9%**
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	76.6%
-	-	-	-	-	-	✓	✓	✓	✓	✓	-	-	74.5%

Table 1: Performance (F1) for 10-fold cv on 1,500 OFFENSE + 3,000 OTHER tweets represented as character n-grams (C), word n-grams (W), offensive words (O), and after feature selection (FSEL).

Model		# features	IN 10-fold cv	OUT			CROSS
				1:1	1:2	1:4	1:2
imbalanced	1500/3000	350	77%	72%	72%	70%	63%
balanced	1500/1500	350	76%	77%	73%	69%	64%
balanced	3000/3000	350	91%	72%	71%	70%	64%
imbalanced	1500/3000	100	89%	72%	73%	73%	70%
balanced	1500/1500	100	88%	77%	75%	71%	70%
balanced	3000/3000	100	96%	73%	75%	**76%**	70%
baseline	-	-	-	33%	40%	44%	44%

Table 2: Performance (F1) for balanced/imbalanced classifiers using 10-fold cv (IN), on holdout sets with different OFFENSE/OTHER distributions (OUT), and on a set labeled by other authors (CROSS).

Model		# features	OUT 1:4				AUC
			OFFENSE		OTHER		
			P	R	P	R	
imbalanced	1500/3000	350	49%	57%	89%	85%	0.77
balanced	1500/1500	350	42%	69%	90%	76%	0.74
balanced	3000/3000	350	49%	57%	89%	85%	0.77
imbalanced	1500/3000	100	57%	56%	89%	89%	0.80
balanced	1500/1500	100	45%	71%	92%	78%	0.76
balanced	3000/3000	100	62%	60%	90%	91%	**0.83**
baseline	-	-	0%	0%	80%	100%	0.50

Table 3. Precision and Recall by class label and AUC score for balanced/imbalanced classifiers, as measured on the holdout set with 1:4 ratio of 100 OFFENSE + 400 OTHER tweets.

KAUSTmine - Offensive Comment Classification on German Language Microposts

Matthias Bachfischer* **Uchenna Akujuobi †** **Xiangliang Zhang‡**
Computer, Electrical and Mathematical Sciences and Engineering Division
King Abdullah University for Science and Technology (KAUST)

Abstract

In this paper, we present two deep-learning based classifier systems for the identification of offensive comments in German Language microposts: A bidirectional LSTM model and a CNN model. Our objective is to compare the performance of these two systems with a traditional, machine-learning based SVM classifier and to evaluate our approach on Task 1 (binary classification) of the GermEval 2018 shared task.

1 Introduction

Modern communication devices and social media play an increasingly important role in our daily lives and the Internet has created tremendous opportunities for exchanging information with people from all over the globe in real-time. Unfortunately however, this freedom gets frequently abused, and hate speech and toxic comments are present in virtually all online communities. A 2017 report by Pew Research even came to the conclusion that up to 41% of all adults have personally experienced online harassment (Duggan, 2017).
Automated detection routines to identify and block toxic messages have proven to be viable methods in shielding online communities from harassment (Wulczyn et al., 2017). Training a computer to understand the emotions and opinions expressed in a document is a common task in the area of Natural Language Processing (NLP), and the results from previous publications (Georgakopoulos et al., 2018) as well as a Kaggle competition [1] sponsored by Google Jigsaw have already shown promising results for the identification of toxic content in online messages.
The intention of the paper at hand is to create a series of deep-learning based neural network models to compete in Task 1 (binary classification) of the GermEval 2018[2] competition. The GermEval 2018 competition is a shared task for the identification of offensive comments in German language microposts. For our research, we choose a simple Support Vector Machine (SVM) model as a baseline and compare its performance against our implementations of a bidirectional Long Short-Term Memory (LSTM) and a Convolutional Neural Network (CNN) model.

2 Related Works

So far, most of the research in the area of toxic comment classification has been focused on English language, and a variety of machine-learning and deep-learning models have been produced to tackle this problem (Schmidt and Wiegand, 2017). Amongst others, Georgakopoulos et al. (2018) used a deep-learning based CNN model to detect toxic language in online content, while Nobata et al. (2016) developed a machine-learning model using a variety of feature classes (N-grams, Syntactic and Semantic Features etc.) and were able to outperform existing deep-learning based approaches. In another research paper published by Razavi et al. (2010), multi-level classification was used to detect offensive comments, mainly in Usenet messages.
The identification of toxicity in German language messages has received less attention by the research community so far, and comparable research is sparse. In the related domain of sentiment analysis for tweets, Cieliebak et al. (2017) created a corpus consisting of 10.000 tweets in German language and provided benchmarks for the classi-

*bachfischer.matthias@googlemail.com
†uchenna.akujuobi@kaust.edu.sa
‡xiangliang.zhang@kaust.edu.sa

[1] Toxic Comment Classification Challenge https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[2] Germeval 2018 - Shared Task on the Identification of Offensive Language - https://projects.cai.fbi.h-da.de/iggsa

fication of these tweets into sentiment classes of either *positive, negative* or *neutral* using a CNN.

3 Data

Before training our systems, we first obtain the training set from the GermEval 2018 competition mailing list. The training set contains a total of 5009 messages which have been labeled either as *OFFENSE* or *OTHER*. A detailed breakdown of the class distribution in the dataset is presented in Table 1.

Dataset	Offense	Other	Total
Training set	1688	3321	5009

Table 1: Class distribution - GermEval 2018 dataset

The dataset is imbalanced, and the majority of the tweets (66%) belong to the neutral class, whereas the remaining data (34%) belongs to the offensive class. The microposts within the dataset were extracted exclusively from Twitter[3] because the conference organizers regarded tweets "as a prototypical type of micropost"[2].

4 Experimental Setup

We present two classification systems in our research: a bidirectional LSTM model and a CNN model. Both models were implemented in Python and make use of the Keras library (Chollet, 2015) for training the classifier. In addition to this, we create a SVM classifier using Scikit-learn (Pedregosa et al., 2011) and consider this as a baseline for testing and improving our deep-learning models. The experiments were performed on a workstation running Ubuntu 16.04 with 64 cores and 128 GB of RAM.

We use the word vectors published by Deriu et al. (2017) for this research. These vectors were trained on a total of 200 million tweets and have a dimensionality of $d = 200$.

Preprocessing: Before extracting features, we first preprocess the data according to the following procedure:

1. Replace URLs, usernames and retweets with replacement tokens *URLTOK*, *USRTOK* and *rt*
2. Convert tweet text to lowercase
3. Convert categorical classification variables into an One-Hot encoded vector
4. Tokenize tweets (using Keras's builtin Tokenizer) and create a list of word indexes with length $l = 100$ (comments shorter than 100 are padded with 0)

For further reference, a preprocessed tweet is presented in Example 4.1.

Example 4.1:
Original: *@salzmanufaktur @Renft1964* Jetzt bekommt Merkel noch Grüne Untergangs-Beschleuniger dabei!
Preprocessed: *USRTOK USRTOK* jetzt bekommt merkel noch grüne untergangs-beschleuniger dabei!

5 System Description

After preprocessing, we feed the data into our classification models: a bidirectional LSTM and a CNN model. By using the word vectors from Deriu et al., we create an embedding matrix where we randomly initialize the words that are not in the word embeddings with the arithmetic mean and standard deviation obtained from the embeddings. The resulting embedding matrix has the size of $|\vec{w}_1; ...; \vec{w}_L| \in \mathrm{IR}^{\mathrm{L}\times 200}$ with L being the number of unique words in our training set.

While training our models, we try to minimize the binary cross entropy loss on the training set given per the formula below:

$$-\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \qquad (1)$$

The final outputs of the models are connected to a softmax regression layer which returns the class $\hat{y} \in [1, K]$ with the largest propability

$$\hat{y} = \arg\max_j \; P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathrm{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathrm{T}\mathbf{w}_k}} \qquad (2)$$

where w_j denotes the weight vector for class j. For the optimization step, we choose the *Adam* optimizer (Kingma and Ba, 2015) with a learning rate $lr = 0.001$, $\beta_1 = 0.99$, $\beta_2 = 0.999$ and $\varepsilon = 1^{-8}$.

5.1 SVM Model

As a baseline for the evaluation of our results, we use a simple SVM classifier trained on Term-Frequency times Inverse Document-Frequency (TF-IDF) vectors (Ramos, 2003) of the tweets in

[3]Twitter social network - `https://twitter.com`

the dataset. The TF-IDF scores were calculated by using the count matrices of 5-grams where the tweet texts serve as input tokens. The classifier uses Stochastic Gradient Descent (SGD) with the logistic regression loss function where we multiply the regularization term with a constant $\alpha = 1^{-5}$.
The output of the SVM classifier was submitted to the GermEval competition under the submission name *KAUSTmine_coarse_1.txt*.

5.2 LSTM Model

The LSTM model in this research was derived from the works of Hochreiter and Schmidhuber (1997). So far, LSTM networks have been successfully applied in a variety of tasks such as Machine Translation (Sutskever et al., 2014) and Image Captioning (Vinyals et al., 2015). Recent research however shows that LSTM models also perform well when applied to NLP tasks such as text classification (Zhang et al., 2015; Zhou et al., 2016).
In this paper, we employ a bidirectional LSTM model with 64 units. The output is passed to two fully connected layers with 64 and 2 units respectively. To prevent our model from overfitting, we apply the early-stopping technique (Prechelt, 1998) in combination with a dropout of $d = 0.5$ after the first dense layer as well as on the recurrent input signal of the LSTM units (Gal and Ghahramani, 2016). We furthermore use Rectified Linear Unit (RELU) as the activation function of the first hidden layer (Srivastava et al., 2014; Glorot et al., 2011).
The output of the LSTM classifier was submitted to the GermEval competition under the submission name *KAUSTmine_coarse_2.txt*.

5.3 CNN Model

The CNN model used in this research builds on the work of Kim (2014) who proposed to use a 2 layered CNN to perform sentence classification in NLP tasks. We create our model by using one convolutional layer with 64 filters (one layer consists of a convolution and a pooling layer). The output of the convolutional layers is then fed into one dense layer of 64 units. As in the previous model, we again make use of the RELU function for the activation of the layer and apply early-stopping in combination with a dropout of $d = 0.5$.
The output of the CNN classifier was submitted to the GermEval competition under the submission name *KAUSTmine_coarse_3.txt*.

6 Results

The models presented in this paper were tested on the test data provided by the GermEval organizers. Since the GermEval organizers did not publish labels for the test data before the submission deadline, there was no possibility for the authors of this paper to evaluate the performance of the proposed systems. To view the results from the systems proposed in this paper, please refer to the following submission runs in the evaluation material published by the conference organizers:

System	Submission Run
SVM	KAUSTmine_coarse_1.txt
LSTM	KAUSTmine_coarse_2.txt
CNN	KAUSTmine_coarse_3.txt

Table 2: Submissions from team KAUSTmine

7 Conclusion

The objective of our participation in the GermEval 2018 shared task was to evaluate the performance of deep-learning based models on the classification of offensive language in German microposts. In the paper at hand, three classification systems were used to participate in Task 1 of the GermEval competition. Using a SVM classifier as a baseline, we further developed two deep-learning based systems in order to compare our results: a bidirectional LSTM model and a CNN model.
We hope that online social network platforms can use our results to build systems that can successfully detect and combat toxicity in online conversations. Services such as Perspective API [4] are taking a step into the right direction, and we expect to see more fascinating research for making the Internet a friendlier and more welcoming place.

References

François Chollet. 2015. Keras library. `https://keras.io` (accessed June 19, 2018).

Mark Cieliebak, Jan Milan Deriu, Dominic Egger, and Fatih Uzdilli. 2017. A twitter corpus and benchmark resources for german sentiment analysis. In *Proceedings of the Fifth International Workshop on*

[4] Perspective API - `https://www.perspectiveapi.com/`

Natural Language Processing for Social Media, Valencia, Spain, April 3-7, 2017, pages 45–51. Association for Computational Linguistics.

Jan Deriu, Aurelien Lucchi, Valeria De Luca, Aliaksei Severyn, Simon Müller, Mark Cieliebak, Thomas Hofmann, and Martin Jaggi. 2017. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, April 3–7, 2017*, pages 1045–1052. International World Wide Web Conferences Steering Committee.

Maeve Duggan. 2017. Online harassement 2017. Report, Pew Research Center.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, New York, USA, June 19 - 24, 2016*, pages 1050–1059. JMLR.org.

Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, and Vassilis P. Plagianakos. 2018. Convolutional neural networks for toxic comment classification. *arXiv preprint arXiv:1802.09957*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, USA, April 11-13 2011*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Lstm can solve hard long time lag problems. *Neural Computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, October 25-29, 2014*, pages 1746–1751. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR), San Diego, USA, May 7-9, 2015*.

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, Montreal, Quebec, Canada, April 11-15 2016*, pages 145–153. International World Wide Web Conferences Steering Committee.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Lutz Prechelt. 1998. *Early stopping-but when?* Neural Networks: Tricks of the trade. Springer.

Juan Ramos. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the First instructional Conference on Machine Learning, Piscataway, USA, December 3-8, 2003*, volume 242, pages 133–142.

Amir H. Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *Proceedings of the 23rd Canadian Conference on Advances in Artificial Intelligence, Ottawa, Canada, May 31 - June 02, 2010*, pages 16–27. Springer.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, Valencia, Spain, April 3-7, 2017*, pages 1–10.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, Montreal, Canada, December 8-13, 2014*, pages 3104–3112.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, USA, June 7-12, 2018*, pages 3156–3164. IEEE.

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, Perth, Australia, April 3-7, 2017*, pages 1391–1399. International World Wide Web Conferences Steering Committee.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, Montreal Canada, December 7-12, 2015*, pages 649–657.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings*

of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, December 11-17, 2016, page 3485–3495.

Fine-Grained Classification of Offensive Language

Julian Risch[1], Eva Krebs[2], Alexander Löser[2], Alexander Riese[2] and Ralf Krestel[1]

Hasso Plattner Institute, University of Potsdam
[1]firstname.lastname@hpi.de
[2]firstname.lastname@student.hpi.de

Abstract

Social media platforms receive massive amounts of user-generated content that may include offensive text messages. In the context of the GermEval task 2018, we propose an approach for fine-grained classification of offensive language. Our approach comprises a Naive Bayes classifier, a neural network, and a rule-based approach that categorize tweets. In addition, we combine the approaches in an ensemble to overcome weaknesses of the single models. We cross-validate our approaches with regard to macro-average F_1-score on the provided training dataset.

1 Toxic Comment Classification

With the ever growing popularity of the Internet, social networks nowadays have large user bases. The users of those social networks produce huge amounts of text data in form of posts. As of 2017, even if we only consider the website Twitter, there are 500 million Twitter posts (tweets) per day[1]. While the majority of those tweets uses appropriate language, there are also tweets that contain offensive language.

There are different kinds and severity levels of offensiveness. If a user describes the weather with profane words, the resulting tweet would be considered offensive. However, compared to tweets containing a direct insult or identity hate, which may even be criminal offenses, the previous example is a rather harmless offense.

Regardless of their severity, those offensive posts need to be found and moderated. Due to the high number of posts, it is not feasible to manually check each post for offensiveness. Therefore, we propose to automatically classify offensive language in tweets. In this paper, we describe a machine-learning-based approach, using ensembles of different classifiers to detect and classify different severity levels of offensive language.

2 Related Work

An important issue in the field of online comment classification is the availability of labeled data. Thanks to Kaggle's recent Toxic Comment Classification Challenge[2] there is a publicly available dataset of more than 150,000 comments. In this challenge participants classified Wikipedia talk-page comments at different levels of toxicity but also distinguished between obscene language, insults, threats, and identity hate. Similarly, the First Shared Task on Aggression Identification (Kumar et al., 2018) dealt with the classification of the aggression level of user posts at Twitter and Facebook. It was part of the First Workshop on Trolling, Aggression and Cyberbullying at the 27th International Conference of Computational Linguistics (COLING 2018). The task considered the three classes "overtly aggressive", "covertly aggressive", and "non-aggressive". In general, we perceive a trend towards finer-grained classification of toxic comments. Thereby the challenge shifts from detecting toxic comments to giving more specific reasons why a particular comment is considered toxic (on the basis of its subclass).

Previous research agrees that word n-grams are well-performing features for the detection of hate speech detection and abusive language (Nobata et al., 2016; Badjatiya et al., 2017; Warner and Hirschberg, 2012; Davidson et al., 2017; Schmidt and Wiegand, 2017). However, ensembles, which combine different, complementing approaches outperform single approaches and achieve especially robust results (Risch and Krestel, 2018a). Word n-grams, character n-grams, and — given a

[1]https://www.omnicoreagency.com/twitter-statistics/

[2]https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

large amount of training data — deep learning approaches perform well in combination.

The task of toxic comment classification is not only of theoretical significance but also has practical applications, for example at the moderation of user-generated content. It has become an industry-wide, costly challenge for online news providers to moderate their discussion platforms. To this end, different approaches have been proposed, which deal with predicting the moderation effort (Ambroselli et al., 2018) or semi-automated classification (Risch and Krestel, 2018b).

3 GermEval Task 2018

We consider the GermEval task 2018[3], which is to classify the offensiveness of German-language tweets. The provided training dataset consists of 5009 categorized tweets and the provided test dataset consists of 3532 uncategorized tweets. There are two tasks: (1) a coarse-grained binary classification with the categories `OFFENSIVE` and `OTHER` and (2) a fine-grained classification with the four categories `PROFANITY`, `INSULT`, `ABUSE` and `OTHER`. Both tasks are multi-class classification tasks (as opposed to multi-label classification), because the classes are mutually exclusive. In this paper, we focus on the more challenging, fine-grained classification task.

While the training data contains examples from all categories, the categories are not uniformly distributed: The majority of tweets (66.3%) is labeled `OTHER`, while `ABUSE` (20.4%) and `INSULT` (11.9%) also occur relatively often. The category `PROFANITY` is underrepresented and constitutes only 71 of the 5009 tweets (1.4%).

The category `PROFANITY`, consists of all tweets that include profane words that are not directed towards a person or group, see Figure 1a. The category `INSULT` includes tweets with negative content directed towards individuals, see Figure 1b. In contrast to the `INSULT` category, `ABUSE` encompasses negative sentiments towards social groups or their members, because of traits associated with that group, see Figure 1c. The last category, `OTHER`, contains every tweet that is not covered by the previous categories. The GermEval task is evaluated with regard to macro-average F_1-score, which is the unweighted mean of the F_1-scores of each individual category.

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

> @anna_IIna Kann man diesen ganzen Scheiß noch glauben..?

(a) Training sample categorized as `PROFANITY`

> @AchimSpiegel ”Sigmar Dumpfbacke Gabriel” gefällt mir richtig gut

(b) Training sample categorized as `INSULT`

> @diMGiulia1 Araber haben schon ekelhafte Fressen....!!

(c) Training sample categorized as `ABUSE`

Figure 1: Example tweets from the training dataset and their fine-grained labels.

4 Fine-Grained Classification of Offensive Language

We propose different approaches for the task of fine-grained classification of offensive language. These approaches are tailored to have different strengths and weaknesses. In an ensemble, we leverage that the approaches complement each other. To this end, we propose diverse approaches, such as a Naive Bayes classifier, Sentiment Polarity Lexicons, and Deep Neural Networks.

4.1 Naive Bayes Classifier

Our first approach uses a Naive Bayes classifier with logistic regression to categorize the tweets. Thereby the logistic regression is trained with the log-count ratios of the Naive Bayes model. Wang and Manning proved that this kind of model works very well as a baseline (Wang and Manning, 2012). Because of the underlying bag of words model, it works well with texts that contain words, more specifically bigrams, that are strong indicators for one of the categories. On the downside, it does not work well with test data that contains many unseen words.

4.2 Neural Network Classifier

Neural networks achieved state of the art results in different classification tasks, including Natural Language Processing centered tasks such as sentiment analysis (Zhang et al., 2018). Our network is based on an Long Short-Term Memory (LSTM) layer and a Global Maximum Pooling layer. For

the final classification, we use a Dense layer with softmax activation. The given dataset in our task is relatively small with about 5000 samples and therefore does not work well with typical deep neural networks. To solve this problem, we make use of transfer learning.

Transfer Learning Instead of training the network with the limited training data of the task, we pre-train the network on a related task with a larger amount of data. We use a dataset of more than 150,000 German, machine-translated from English, Wikipedia talk page comments. This dataset originates from the Kaggle Toxic Comment Classification challenge and is human-labeled with several toxicity categories. After this training phase, the weights in the neural network are adjusted to the GermEval task. Because the Kaggle challenge is similar to the GermEval task, we kept the first layers with the corresponding weights and added a shallow network of Dense layers on top of them. Afterwards, the modified network is trained on the GermEval data, whereby only the newly added Dense layers get adjusted by the backpropagation. The other weights remain unaffected with the intent to include general representations (learned on a larger dataset) in the first layers.

Imbalanced Classes Besides the small size of the training dataset, the distribution of the different categories is challenging in combination with the evaluation metric. In many cases, `OTHER` is wrongly predicted instead of the correct category (false positives), because this is by far the largest fraction of the training dataset and therefore often the correct result. However, the macro-average F_1-score takes the F_1-score of each category uniformly into account. This evaluation measure results in an overall bad performance if there are many false positives for the majority class.

To address this concern, we consider two approaches: class weights and generating synthetic training data with the synthetic minority oversampling technique (SMOTE) (Chawla et al., 2002). The class weights add a factor to the loss function dependent on the predicted class. In our case, this parameter was set to 'balanced' to use class weights that are inversely proportional to the class sizes and therefore increase the penalty for misclassifying minority category examples.

The SMOTE algorithm operates on the input data and generates additional samples of the minority classes in order to balance the data. This is achieved by repetitively taking samples and a number of nearest neighbors in the feature space and randomly interpolating between them. The resulting interpolation point corresponds to the newly created, additional data point for the appropriate minority class. This procedure is executed for each minority class.

4.3 Rule-based Classifier

The small amount of provided training data motivates to develop classifiers based on specific rules tailored to the GermEval task. For example, a tweet in the category `PROFANITY` will definitely contain a profane word, but likely not a person or group.

We collected several word lists for the rules. Some are from external sources, such as an exhaustive list of profane or insulting words, a list of German politicians and political parties, and words that are usually used in a negative context. In addition, we manually created lists with words that appeared very often in a specific context. For example, words related to the refugee crisis appeared more frequently in tweets classified as `ABUSE`.

The classifier has scores for all categories, `OTHER` being the default. The rules check for word occurrences. Each time a word is found, scores of categories related to the rule are increased. The highest score determines the predicted category.

4.4 Ensemble Classifier

Table 1 lists the Pearson correlation of the different classifiers' out-of-fold predictions on the training dataset. The correlation is very small, which shows that the classifiers have different strengths and weaknesses. As a consequence, they provide the opportunity to combine the individual results with an ensemble classifier, which potentially further improves predictions. We discuss two ensembling methods: logistic regression and gradient boosting trees.

Logistic Regression and Gradient Boosting Ensembles Due to the imbalanced class labels in the training dataset, the learning uses balancing class weights. The logistic regression approach takes only the final results of the classifiers into account. In contrast, our gradient boosting approach also considers features of the text. These features are the text length, the ratio of exclamation marks and the ratio of uppercase characters. We use a gradient boosting ensemble, in form of a light gradient

	NB - NN	NN - RB	NB - RB
Profanity	0.0037	0.0604	-0.0052
Insult	0.0723	0.0235	0.1154
Abuse	-0.0015	0.0809	0.2278
Other	0.1185	0.0778	0.2434

Table 1: The Pearson correlation values for each label with pairwise comparisons for Naive Bayes (NB), the neural network (NN), and the rule-based approach (RB)

boosting machine classifier (Ke et al., 2017).

4.5 Sentiment Polarity Lexicons

In addition to the previously described approaches, we investigate sentiment polarity lexicons, which provide a large knowledge base of word-polarity pairs. This external knowledge can potentially compensate for the relatively small amount of provided training data. Given a tweet, we infer the sentiment of each contained verb. For the classification, we consider the presence or absence of verbs with negative polarity. Further, we consider whether the negative verb refers to an entity, such as a particular person or group. Thereby, we aim to distinguish insult and abuse from profanity. We incorporate sentiment scores obtained from a variety of external sources, such as "German Polarity Cues" (Waltinger, 2010), "German Sentiment Lexicon" (Clematide and Klenner, 2010), and "SentiWS"(Remus et al., 2010). Further, we extract character n-grams and word unigrams as features for profane language based on a list of swear words.

5 Evaluation

As of writing this paper, the test dataset of the GermEval task is published, but not its ground truth labels. To this end, we analyze only the predicted class distribution on the test dataset. We evaluate our approaches on the provided training dataset with cross-validation.

5.1 Evaluation Measures

The GermEval task defines the macro-average F_1-score as its evaluation measure. With the measure given, we still need a set of labeled test data to evaluate our classifiers. As of writing this paper, the test dataset of the GermEval task is published, but not its labels. As a result, we can use only the training dataset as evaluation data. Since the training dataset is rather small with only 5009 labels, we decided against splitting it up in a disjoint training and test set for the evaluation. Instead, we use 5-fold cross-validation and analyze out-of-fold predictions. To this end, we split our training set into five equally sized folds. Then we choose one fold as the test set that we want to predict, and train on all other folds. We repeat this step until each fold was the test set, and thus predicted, once. This way we can predict labels for the whole test set, without ever seeing the tweets we make predictions for in the training set.

5.2 Discussion of the Results

Table 2 lists the evaluation results for our individual classifiers. The Naive Bayes classifier identifies most of the tweets that should be labeled `OTHER`, nearly none that are `PROFANITY` and a small amount with a relatively high precision that should be in category `INSULT` or `OTHER`. The recall of category `PROFANITY` might be especially low because this category is represented the least in the training dataset and the classifier only learns on words found in the training dataset. The opposite may be true for `OTHER`, which is the most often occurring category. In total the Naive Bayes classifier achieved an F_1-score of 0.366.

In comparison to the Naive Bayes classifier, the neural network detects considerably less `OTHER`, but it detects a certain amount of `PROFANITY`. The recall values for `INSULT` and `ABUSE` are also higher, but similar to `PROFANITY` they have a relatively low precision. The neural network achieved a total F_1-score of 0.261. This evaluation already considers our approaches against class imbalance. Both approaches, SMOTE and class weights, increased the F_1-score from about 0.22 to about 0.26, while the SMOTE approach performs slightly better than the class weights.

The rule-based classifier finds slightly less `OTHER` than the Naive Bayes classifier, but has a higher recall and lower precision on the other three categories. Since the rules work with very specific word lists, the classifier may be able to detect more tweets that fit the rules, but cannot differentiate them from non-offensive texts that also contain those words. The rule-based approach is the best individual classifier with an F_1-score of 0.390.

Our ensemble classifiers performed better than the individual classifiers: the gradient boosting ap-

	Naive Bayes			Neural Network			Rule-based		
	precision	recall	F_1	precision	recall	F_1	precision	recall	F_1
Profanity	0.20	0.01	0.03	0.02	0.25	0.04	0.15	0.28	0.20
Insult	0.49	0.13	0.20	0.17	0.32	0.22	0.23	0.21	0.22
Abuse	0.70	0.29	0.41	0.22	0.32	0.26	0.46	0.32	0.37
Other	0.73	0.97	0.83	0.78	0.39	0.52	0.73	0.81	0.77

Table 2: The F_1-scores for each category predicted by the Naive Bayes classifier, the neural network, and the rule-based classifier

	Gradient Boosting Ensemble			Logistic Regression Ensemble			Sentiment Lexicons		
	precision	recall	F_1	precision	recall	F_1	precision	recall	F_1
Profanity	0.12	0.51	0.19	0.17	0.44	0.25	1.00	0.03	0.05
Insult	0.30	0.43	0.36	0.43	0.30	0.35	0.44	0.29	0.35
Abuse	0.47	0.51	0.49	0.57	0.43	0.49	0.56	0.39	0.46
Other	0.85	0.70	0.77	0.80	0.87	0.83	0.77	0.90	0.83

Table 3: The F_1-scores for each category predicted by the gradient boosting ensemble, the logistic regression ensemble classifier, and the sentiment lexicon approach for comparison

proach reached a score of 0.450 and the logistic regression ensemble achieved a score of 0.480. Notice that no individual classifier exceeds a macro-average F_1-score of 0.4. The detailed results can be seen in Table3. The gradient boosting classifier has higher recall values for the three offensive categories, but a lower precision. In contrast, the logistic regression ensemble classifier has lower recall values, except for `OTHER`, but a higher precision and total score.

In context of the GermEval task 2018, the logistic regression ensemble classifier provides the best result, as it has the highest total F_1-score. However, if the classifiers were to be used for a real-world application (e.g. helping Twitter moderators to find tweets that they should assess), the gradient boosting approach may be better suited. The gradient boosting approach has the highest combined recall values for the three offensive labels of all our classifiers, which means that more offensive tweets would be found. In a second step, the false positives could be removed by another algorithm or a human worker.

While we cannot provide an F_1-score for the test set, we still use the ensemble classifiers to predict its labels. We also use out-of-fold predictions, but instead of predicting for the remaining fold, we predict the entire test set. The result of this procedure are five complete prediction files, which are later combined into a final prediction by calculating the average.

The gradient boosting ensemble predicts more tweets to be in the three offensive categories. In contrast, the logistic regression approach classifies more tweets as `OTHER`. We assume that the samples' ground truth categories follow the same frequency distribution in the training set and the test set. The general category distribution of both classifiers' predictions is similar to the distribution of the categories in the training data. The `OTHER` category occurs the most often and `PROFANITY` the least often, which is shown in Figure 2. However, the distribution of the training set and the predictions for test set do not match exactly. This discrepancy is an opportunity for more optimization, which goes beyond this paper.

5.3 Test Dataset Submission

We submitted prediction files for the two tasks of fine-grained and coarse-grained classification. The logistic regression ensemble, the sentiment polarity lexicons, and a combination of both approaches comprise our final submission. The combination is the mean of the predicted probabilities of both approaches. The files correspond to our previously described approaches as follows:

- `hpiTM_fine_1.txt`: logistic regression ensemble

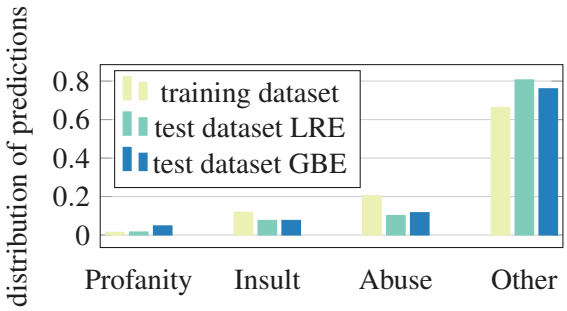


Figure 2: Category distribution predicted by the logistic regression ensemble (LRE) and gradient boosting ensemble (GBE) for the 3532 tweets in the test dataset compared with the training distribution of 5009 tweets

- `hpiTM_fine_2.txt`: sentiment polarity lexicons
- `hpiTM_fine_3.txt`: sentiment polarity lexicons and logistic regression ensemble combined
- `hpiTM_coarse_1.txt`: logistic regression ensemble

6 Conclusion

In this paper we considered the problem of classifying German tweets into four different categories of offensive language in context of the GermEval task 2018. This task uses the macro-average F_1-score as evaluation measure. In order to maximize this score, we proposed different classifiers, such as a Naive Bayes classifier, a neural network, and a rule-based approach. The results of these classifiers were combined in two different ensemble methods to achieve a higher score. This ensemble achieves a macro-average F_1-score of 0.48 at cross-validation on the provided training dataset. We provide our source code online[4].

An interesting path for future work is to provide fine-grained classification labels to content moderation teams at online platforms. The fine-grained labels can provide an explanation for why a particular user comment is considered toxic and may be deleted by the moderation team. To this end, even finer-grained labels that describe the target group of an insult, such as a particular religion, ethnic minority or nationality are needed. Based on such labels, also an analysis of offensive language could go into more detail and shine a light on reasons for and intentions of toxic comments.

[4] `https://hpi.de/naumann/projects/repeatability/text-mining.html`

Acknowledgments

We thank Samuele Garda for his help with this project and for his valuable feedback.

References

Carl Ambroselli, Julian Risch, Ralf Krestel, and Andreas Loos. 2018. Prediction for the newsroom: Which articles will get the most comments? In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 193–199. ACL, June 1–6.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 759–760. International World Wide Web Conferences Steering Committee.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Simon Clematide and Manfred Klenner. 2010. Evaluation and extension of a polarity lexicon for german. In *Proceedings of the First Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 7–13.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*, pages 512–515.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the Workshop on Trolling, Aggression and Cyberbulling (TRAC)*.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 145–153. International World Wide Web Conferences Steering Committee.

Robert Remus, Uwe Quasthoff, and Gerhard Heyer. 2010. Sentiws - a publicly available german-language resource for sentiment analysis. In *Proceedings of the Conference on International Language Resources and Evaluation (LREC)*. European Languages Resources Association.

Julian Risch and Ralf Krestel. 2018a. Aggression identification using deep learning and data augmentation. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (co-located with COLING)*, pages 150–158, August.

Julian Risch and Ralf Krestel. 2018b. Delete or not delete? semi-automatic comment moderation for the newsroom. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (co-located with COLING)*, pages 166–176, August.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the International Workshop on Natural Language Processing for Social Media (SocialNLP)*, pages 1–10. ACL.

Ulli Waltinger. 2010. Germanpolarityclues: A lexical resource for german sentiment analysis. In *Proceedings of the Conference on International Language Resources and Evaluation (LREC)*. European Language Resources Association.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Workshop on Language in Social Media (LSM)*, pages 19–26. ACL.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1253.

TUWienKBS at GermEval 2018: German Abusive Tweet Detection

Joaquín Padilla Montani
TU Wien
Institut für Logic and Computation
Favoritenstraße 9-11, 1040 Austria
jpadillamontani@gmail.com

Peter Schüller
TU Wien
Institut für Logic and Computation
Favoritenstraße 9-11, 1040 Austria
ps@kr.tuwien.ac.at

Abstract

The TUWienKBS system for abusive tweet detection in the GermEval 2018 competition is a stacked classifier. Five disjoint sets of features are used: token and character n-grams, relatedness to the, according to TFIDF, most important tokens and character n-grams within each class, and the average of the embedding vectors of all tokens in a tweet. Three base classifiers (maximum entropy and two random forest ensembles) are trained independently on each of these features, which yields 15 predictions for the type and/or level of abusiveness of the given tweets. One maximum entropy meta-level classifier performs the final classification. As word embedding fallback for out-of-vocabulary tokens we use the embeddings of the largest prefix and suffix of the token, if such embeddings can be found.

1 Introduction

We describe the TUWienKBS system that participated in the GermEval 2018 competition for abusive tweet detection.

This task is relevant for supporting humans when they moderate online content. In the pseudo-anonymous environment of microposts, abusive language is easily produced by users and it is an important objective to prevent that such content is broadcast to a large number of readers.

Our system is based on a stacked architecture where a set of three types of classifiers is trained on a set of five feature groups, and the resulting fifteen trained models are forwarded to a meta-level classifier that decides the final outcome of the prediction. This architecture and its training method is inspired by the EELECTION system (Eger et al., 2017) however our features and classifiers are different.

In particular we produce features based on a per-class selection of, according to TFIDF, most important characteristics of tokens and character n-grams. For each class we create the same number of features in this feature category and we found that this helps with the imbalanced training set of 5009 tweets where one of four classes to be predicted contains only 71 samples.

This paper is organized as follows. In Section 2 we give details about the competition tasks and evaluation metrics. In Section 3 we describe tweet preprocessing and the features we use. In Section 4 we describe the machine learning model we use and the stacked predictor model and we describe how we train this architecture. Section 5 describes our submission files and provides an evaluation of our model and features on training data. In Section 6 we describe additional experiments we performed that did not yield improvements of scores. We conclude the paper in Section 7.

2 Competition Tasks

The GermEval 2018 Shared Task on the Identification of Offensive Language[1] solicited the submission of systems that classify German microposts (in a Twitter dataset) with respect to their offensiveness. Such predictions are a valuable tool for assisting human moderators with the job of reducing the amount of hurtful, derogatory or obscene online content.

The competition contained two tasks:

- Task 1: coarse-grained classification into the two classes "OFFENSIVE" and "OTHER" (where "OTHER" means non-offensive), and
- Task 2: fine-grained classification into the four classes "PROFANITY", "INSULT", "ABUSE", and "OTHER".

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

Each micropost is tagged with exactly one class of Task 1 and with exactly one class of Task 2. The classes are mutually exclusive, in particular, the "PROFANITY" class does not contain any insults, "ABUSE" does not insult a single concrete person but a whole group of people and is also abusive in a way that is not simply "PROFANITY", see also the annotation guidelines (Ruppenhofer et al., 2018).

The competition evaluation uses macro-averaging of the F1-score of the predictions as final score, i.e., each class contributes equally to the final score independent from the number of samples in the class. The training set contains 5009 tweets where 3321 are marked as "OTHER" and the remaining ones as "OFFENSE" in Task 1. Of the offensive tweets, 1022 are marked as "ABUSE", 595 as "INSULT", and 71 as "PROFANITY".

This imbalance in the training set gave rise to several decisions we made while creating our system, we discuss these in particular in Sections 3.4, 6.3, and 7.

3 Features

We implemented feature computation using the libraries Scikit-learn (Pedregosa et al., 2011) for TFIDF computations, NLTK (Bird et al., 2009) for tokenization and stemming, and GenSim (Řehůřek and Sojka, 2010) for managing precomputed word embeddings.

3.1 Preprocessing

Tweet preprocessing removes all handles (@username) and replaces the characters "#-_,;:/+)<>&" and line break characters by spaces and we replace the substring "'s" (as in "geht's") by a space.

We use NLTK's `TweekTokenizer` with `reduceLen=True`. That means repetitions of the same character are shortened to at most three letters (e.g., "cooool" is normalized to "coool").

For features with stemming we use the German stemmer of NLTK.

Table 1 gives an overview of the groups of features we use. We describe these in the following.

Special Preprocessing indicates which additional preprocessing is done beyond handle removal and tokenization. For creating character-level features we concatenate (Join in Table 1) the resulting tokens with spaces into one string for extracting character-level n-grams. We always use the tokenizer (even for character-level features) to make use of its `reduceLen` feature.

3.2 Character and Token N-Gram Features

The feature groups CNGR and TNGR are similar so we describe them together. Both operate on a lowercased version of the input, and TNGR additionally performs stemming on each token.

CNGR extracts all character-level n-grams of length 3 to 7, while TNGR extracts all stemmed-token-level n-grams of length 1 to 3. In both cases, we perform TFIDF over all extracted n-grams, keep only those with a document frequency between 0.01 and 0.0002 (i.e., those that are rare enough to carry some signal, but frequent enough to have a potential to generalize over unseen data). The document frequency thresholds were tuned by means of a grid search on a 90%/10% split of the training data, with the aim to maximize prediction scores of the base classifiers (see Section 4).

We use the TFIDF score of the relevant n-grams as input features (realized with `TfidfVectorizer`).

3.3 Word Embedding Features

We use the pretrained word2vec model `twitter-de_d100_w5_min10.bin` with 100 dimensions and window size 5, created from Twitter data of 2013–2017 by Josef Ruppenhofer.[2]

For each tweet, we create 100 real-valued features by taking the average embedding of all tokens in the tweet, normalized to unit length with L2 norm.

Whenever a word embedding is required, i.e., for feature groups TIMP and EMB, and whenever the token is not in the vocabulary of the pretrained list of word embeddings, we perform a fallback operation. We search for the largest prefix and the largest suffix of the token of length 3 or greater where we know a word embedding. If we find such affixes with embeddings, we use the embeddings of these affixes as if they were separate tokens in the tweet. As an example, the word "Nichtdeutsche" (non-Germans) in the dataset does not exist in some pretrained word embedding models so we encounter an OOV exception. Our method would use as a fallback two word embeddings for affixes "Nicht" (not) and "deutsche" (German+Adj) because both affixes are present in the word embedding model. This fallback reduces the number of OOV exceptions in the training set from 1903 to 90 and in the

[2] `http://www.cl.uni-heidelberg.de/english/research/downloads/resource_pages/GermanTwitterEmbeddings/GermanTwitterEmbeddings_data.shtml`

Symbol	Name	Level	Special Preprocessing	Word Embeddings
CNGR	Character N-Grams	C	Lowercase + Join	-
CIMP	Important N-Grams	C	Join	-
TNGR	Token N-Grams	T	Lowercase + Stemming	-
TIMP	Important Tokens	T	-	min/max cos distance
EMB	Word Embeddings	T	-	average

Table 1: Groups of features used for classification. Handle removal and tokenization is used for all features. C and T stand for character and token level, respectively.

Task	m	Feature	k
Task 1	2	CIMP	3200
Task 1	2	TIMP	1250
Task 2	4	CIMP	370
Task 2	4	TIMP	170

Table 2: Number of important types selected for each task and feature group.

testing set from 1069 to 51.

We also experimented with other pretrained word embedding models but none of them achieved a comparable performance. Combining above mentioned word embeddings with other embeddings increased performance in single classifiers, however it decreased performance when these models were used as part of the ensemble described in Section 4.

3.4 Important N-Gram and Token Features

These two groups of features are based on the same idea: to perform TFIDF over the whole dataset, select the k most important types relative to each of the m classes ($m = 2$ in Task 1, $m = 4$ in Task 2). We determine importance by ranking features according to their average TFIDF value in all documents in the respective class. Based on the resulting list of $k \cdot m$ most important type/class combinations we create a feature for each $k \cdot m$ combination.

For CIMP each type is a character n-gram, while for TIMP each type is a token. Intuitively this selects the most distinguishing types per category, a related analysis is described in the blog of Thomas Buhrman.[3]

Table 2 shows the number of important types selected for each task and each feature group. These values were adjusted with a grid search on a 90%/10% split of the training data in order to maximize prediction scores of the base classifiers (see next section).

So far we have only discussed how important types are selected. We next describe which features are generated from these important types.

For TIMP, for each important type t in a tweet we obtain its word embedding $\vec{t}$ and compute the maximum and the minimum cosine distance from $\vec{t}$ to all other embeddings of other types in the same tweet. We use the same OOV-fallback described in Section 3.3. This yields a minimum and a maximum feature for each important type and each class: $2 \cdot k \cdot m$ real features for each tweet.

For CIMP we have no embedding information, therefore we create for each important type t a Boolean feature that indicates whether t is contained in the tweet or not. This yields a feature for each important type and each class: $k \cdot m$ Boolean features for each tweet.

By creating a set of features for each class, we increase the signal that can be learned for the "PROFANITY" class in Task 2 which contains a small set of samples.

4 Classification

Our system is a stacked ensemble system inspired by the EELECTION system of Eger et al. (2017).

We implemented most the classification using the library Scikit-learn (Pedregosa et al., 2011) and refer to class and function names of Scikit-learn in the following (unless explicitly stated otherwise).

4.1 Base Classifiers

For each of the 5 feature groups discussed in Section 3, we train three independent classifiers:

- a MaxEnt model with balanced class weight (class `LogisticRegression`),

[3] https://buhrmann.github.io/tfidf-analysis.html

- an ensemble of random forests trained on samples of the training set (Geurts et al., 2006) using information gain as criterion for scoring the sample splits (class `ExtraTreesClassifier` with `criterion=entropy`), and
- another ensemble of random forests trained using Gini impurity for scoring sample splits (`criterion=gini`).

For `ExtraTreesClassifier` we use 100 and 150 estimators for Task 1 and Task 2, respectively. This yields $5 \cdot 3$ distinct *base classifiers*, i.e., feature/classifier combinations.

We train each base classifier on 90% of the training data and perform predictions on the remaining 10%. We perform this process 10 times in a cross-validation manner to obtain predictions for the whole training data. To obtain more reliable results we repeat the whole process 10 times with different random seeds for determining the cross-validation folds. At that point we have 15 base classifiers and their predictions for each tweet and each class in the training data.

4.2 Meta Classifier

Using predictions of 15 base classifiers for each class, we create 30 meta level features per tweet for Task 1 (two classes) and 60 meta level features per tweet for Task 2 (four classes).

On these features and the known true classes we train a maximum entropy model (`LogisticRegression`). We use one-vs-rest classification, balanced class weights, and tuned parameter $C = 0.17$ for Task 1 and $C = 0.2$ for Task 2.

5 Submission and Pre-Competition Evaluation

We submitted a single run for Task 1 and a single run for Task 2 to the competition, named `TUWienKBS_coarse_1.txt` and `TUWienKBS_fine_1.txt`, respectively.

The source code of our system, i.e., feature computation, training, and classification, is available online.[4]

Based on 10-fold cross-validation with stratified folds (i.e., ensuring stable class ratios in each fold) we performed a pre-competition evaluation of our system and its features on the training data. Table 3 shows that the ensemble achieves a macro-averaged F1 score of 81.72 on the training data for coarse-grained prediction (Task 1), where the most important feature in the ensemble is TIMP, followed by CNGR. Table 4 shows results for fine-grained prediction (Task 2) where the ensemble obtains a score of 61.89, again with TIMP and CNGR as most important feature groups in the ensemble.

Features	F1 score	F1 reduction
ALL	81.72	
without TIMP	79.92	1.80
without CNGR	81.14	0.58
without CIMP	81.15	0.57
without TNGR	81.43	0.29
without EMB	81.69	0.03

Table 3: Evaluation on training set (Task 1).

Features	F1 score	F1 reduction
ALL	61.89	
without TIMP	59.77	2.12
without CNGR	60.24	1.65
without CIMP	60.43	1.46
without EMB	61.61	0.28
without TNGR	61.67	0.22

Table 4: Evaluation on training set (Task 2).

Across both tasks we can say that CNGR features are more useful than TNGR features, however with TIMP and CIMP the situation is reversed: TIMP is the most important feature group in both tasks. The reason is that TIMP uses word embeddings and min/max cosine distances from important types to tokens in the tweet at hand, while CIMP only uses membership in the tweet. This makes TIMP a more powerful feature than CIMP.

EMB and TNGR features contribute only little to the overall ensemble score. If we would use only EMB features to predict the class this would yield reasonable results even without an ensemble, while we would obtain worse results when using only TNGR features (these results are not shown in tables).

Altogether, word embeddings are a crucial component of our system and we use them in different ways in TIMP and EMB feature groups.

[4] `https://github.com/jpadillamontani/germeval2018`

6 What did not work?

While creating our submission for the competition we experimented with several methods that did not improve the system score.

6.1 Feature Selection for Character N-grams

Generating n-grams of length 3–7 at the character level yields more than 200.000 features. We tried to reduce this with several feature selection functionalities implemented in `feature_selection.SelectKBest` in Scikit-learn (Pedregosa et al., 2011) with χ^2 (`chi2`) or ANOVA (`f_classif`) as feature scoring functions. Any reduction in the dimensionality impacted the score negatively.

6.2 Stop Word Removal

We removed stop words from the German stop word list of NLTK ("stopwords corpus") and from another list of publicly available German stopwords.[5]

Stop word removal impacted the score negatively, when applying stop word removal before computing token n-grams as well as when applying stop word removal before the computation of tweet embeddings.

6.3 Under- and Oversampling

To overcome the imbalance in the training set (see also Section 2) we tried several sampling methods for re-balancing the dataset.

We used the `imblearn` package[6] (Lemaître et al., 2017) in particular the classes `RandomUnderSampler` and `RandomOverSampler` to undersample the largest class (OTHER) and to oversample the smallest class (PROFANITY), respectively.

This decreased evaluation scores.

6.4 Deep Learning

To our ensemble we added three architectures based on Keras and TensorFlow.[7] These experiments replicated successful approaches for tweet classification and applied them to the GermEval dataset.

In particular we tried the LSTM and CNN methods[8] of Badjatiya et al. (2017) and the Convolution+GRU method[9] of Zhang et al. (2018).

We trained these models, evaluated them individually and also integrated them into the ensemble (we trained the probabilities as we did for the other classifiers).

All three architectures performed similarly to the other, classical, classifiers, reaching F1 scores around 76 for Task 1 and around 55 for Task 2. Adding these deep learning classifiers to the ensemble decreased its overall score, so in the end we excluded them from the ensemble.

6.5 Using sent2vec instead of word2vec

Instead of using word2vec pretrained word embeddings, we experimented with sent2vec[10] models of Lee et al. (2017).

The features generated this way scored significantly worse than the normal averaging of word2vec embeddings. We also tried combining both word2vec and sent2vec features by simply concatenating their vectors, but this still performed worse than the averaging approach we used in the final submission.

7 Conclusion

Our system combines existing approaches that have been reported to work and includes a group of features that is, to the best of our knowledge, novel: the group of "Important N-Gram and Token Features" (Section 3.4). These features (TIMP and CIMP) are generated from the most important (according to the average of their TFIDF scores) tokens (respectively, character n-grams) and this importance is computed within each class that we aim to predict. Essentially, we identify features that are most suitable for distinguishing documents *within each class* and not across classes. Our experiments showed that these features on their own already obtain high prediction scores on both tasks. In the ensemble, feature group TIMP causes the largest drop in prediction score when removed, making it an important component of the prediction.

A major challenge in this competition was the evaluation mode in combination with the class imbalance in the training data. The competition evalu-

[5] `https://github.com/stopwords-iso/stopwords-de/blob/master/stopwords-de.txt`

[6] `http://contrib.scikit-learn.org/imbalanced-learn/stable/index.html`

[7] `https://www.tensorflow.org/guide/keras`

[8] `https://github.com/pinkeshbadjatiya/twitter-hatespeech/`

[9] `https://github.com/ziqizhang/chase`

[10] `https://github.com/UKPLab/germeval2017-sentiment-detection`

ation uses macro-averaging, i.e., each class counts the same. At the same time, in Task 2, there is one class ("PROFANITY") with only 72 tweets as samples within a training set which contains 5009 tweets. Due to this imbalance, making mistakes in this one class has a higher weight on the result than making mistakes in other classes, and we focused our tuning efforts on managing this class imbalance (partially, but not exclusively, by creating the above mentioned class-wise important features).

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly.

Steffen Eger, Erik-Ln Do Dinh, Ilia Kutsnezov, Masoud Kiaeeha, and Iryna Gurevych. 2017. EELECTION at SemEval-2017 Task 10: Ensemble of nEural Learners for kEyphrase ClassificaTION. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval 2017)*, pages 942–946.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning*, 63(1):3–42.

Ji-Ung Lee, Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. UKP TU-DA at GermEval 2017: Deep learning for aspect based sentiment detection. In *Proceedings of the GSCL GermEval Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 22–29.

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Josef Ruppenhofer, Melanie Siegel, and Michael Wiegand. 2018. Guidelines for IGGSA Shared Task on the identification of offensive language. `http://www.coli.uni-saarland.de/~miwieg/Germeval/guidelines-iggsa-shared.pdf`.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-GRU based deep neural network. In *The Semantic Web*, pages 745–760.

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: $\pm$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: $\pm$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

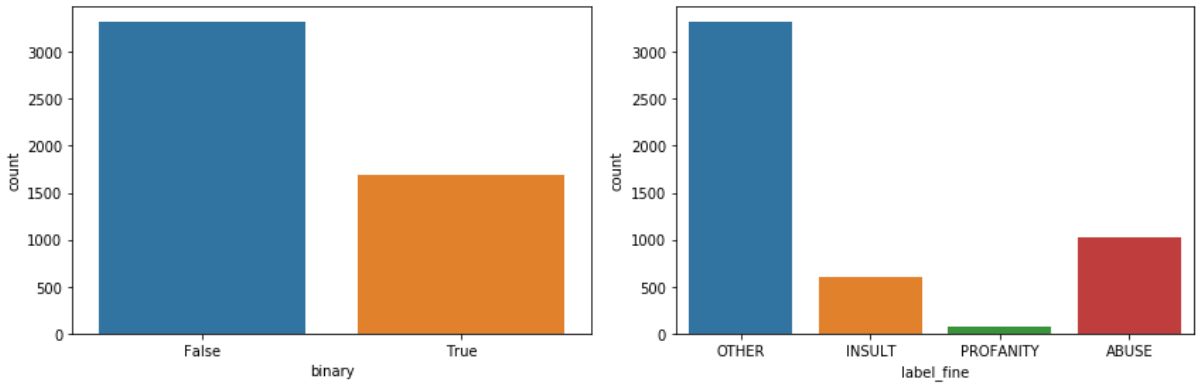


Figure 1: Distribution of "binary" (COARSE) and FINE labels in the training data.

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`

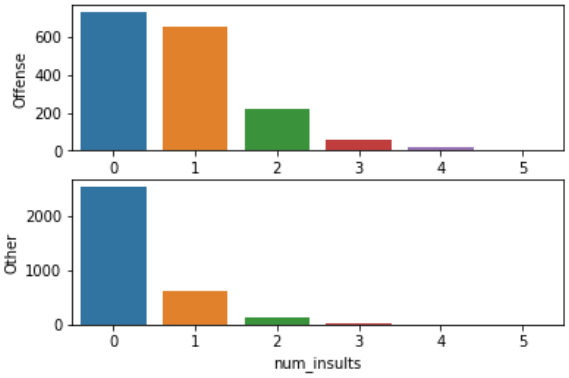


Figure 2: Histogram of the number of insult words per tweet. Top panel = offensive tweets, bottom panel = non-offensive tweets.

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.

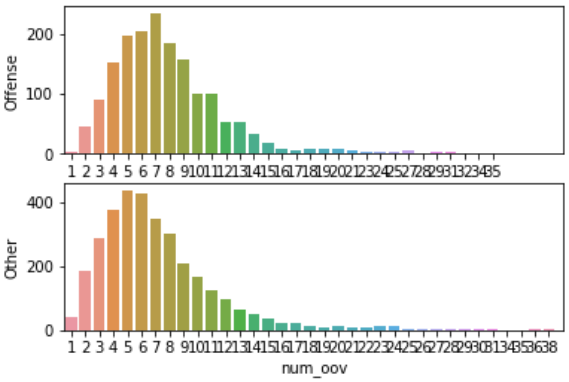


Figure 3: Histogram of OOV tokens per tweet.

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).

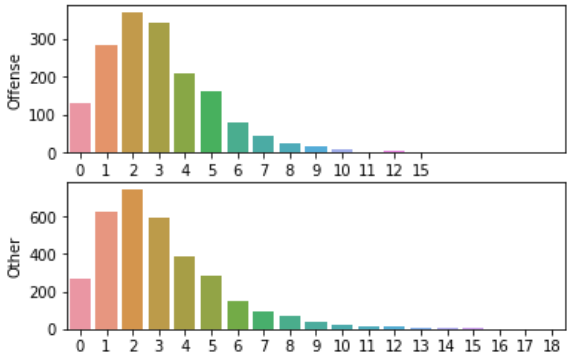


Figure 4: Histogram of the number of symbols per tweet.

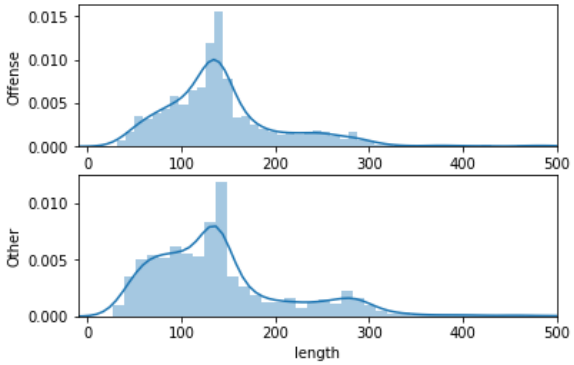


Figure 5: Distribution of tweet lengths in offensive and non-offensive tweets.

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.

- 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.

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

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.

Offensive Language Detection with Neural Networks for Germeval Task 2018

Dominik Stammbach
Saarland University
dominiks@coli.uni-saarland.de

Azin Zahraei
Saarland University
azinz@coli.uni-saarland.de

Polina Stadnikova
Saarland University
polinas@coli.uni-saarland.de

Dietrich Klakow
Saarland University
dietrich.klakow@lsv.uni-saarland.de

Abstract

In this paper we describe our submissions to task I of the GermEval 2018 Shared Task with the goal of identifying offensive language in a set of German tweets. We experiment with two neural architectures and different features. Our submission consists of 3 runs using ensembles of different neural network architectures, each achieving approximately 78 % macro-F1 measure on the last 500 tweets from the training set. The source code for our experiments is publicly available on Github. [1]

1 Introduction

In recent years, it has become increasingly important to come up with countermeasures to deal with offensive language in social media. The NetzDG law which has been in effect since January 1 2018 in Germany requires tech companies like Twitter to delete obviously illegal content. (Wikipedia contributors, 2018) The huge amount of data posted on Twitter and the fact that German is in the top 10 languages of this social media platform (Hong et al., 2011) makes manually monitoring the data unfeasible and calls for automatic methods of identifying offensive language.

The GermEval 2018 Shared Task is focused on detecting offensive comments in a set of German tweets in two subtasks. Task I is a binary classification of tweets. Task II requires a more fine-grained classification of the offensive tweets into 3 subcategories, namely profanity, insult and abuse. But because of the small number of examples for the profanity class, training a neural network to detect profanity was infeasible. Because of the nature of the evaluation metric it was unlikely to get competitive results in task II so we only submit our model for task I.

[1] https://github.com/polinastadnikova/-neurohate

For our submission we have used neural networks which have become the top-performing technique for many tasks in the field of natural language processing. Convolutional Neural Networks (CNN), which were initially invented for the computer vision domain, have proven to be effective for many Natural Language Processing tasks. This architecture allows for extraction of local features in text, e.g. word order. This way, we are able to make use of combinations of words and use fixed size regions of text, e.g. bigrams, trigrams and so on as features. Yoon Kim (2014) shows the effectiveness of using a CNN for text classification by comparing results on different benchmarks. Recurrent Neural Networks (RNN), on the other hand, are able to extract long term dependencies. This is a feature that is definitely useful in offensive language detection. RNN-based methods have produced state-of-the-art scores for offensive language detection in other languages (Del Vigna et al., 2017). Thus, we have implemented both a CNN and a RNN model for this task.

Following many experiments with different ways of handling the data and different architectures for our prediction model, we selected our best models based on their macro-averaged F1 scores. More specifically, we compared the models based on their mean F1 score when 10 fold cross-validating on all the training data. We submit three runs, where the first, second and third runs are an ensemble of RNNs, an ensemble of CNNs and an ensemble of CNNs and RNNs respectively. After describing the data and how we preprocessed them in Section 2, we introduce the architectures and hyperparameters used in our best models in Section 3. In Section 4, we talk about our experimental setups and their results.

2 Data

The training data consists of 5009 tweets in German, where some tweets contain different types

of hate speech. The data is annotated according to the tasks: binary and fine-grained classification. Therefore each tweet has two labels, OFFENSE or OTHER as the first label and as the second label one of the following: INSULT, ABUSE, PROFANITY, OTHER. In our work, we focus on the binary classification, that means we have 1688 training examples containing offensive language and 3321 without hate speech. The reason for our decision not to participate in the fine-grained classification task is that there are only 71 examples for the PROFANITY label, 1022 examples for ABUSE out of 1688 tweets. We believe it is not enough for neural network training and furthermore our system would be biased towards the ABUSE label.

2.1 Preprocessing

For classification, as well as for many other NLP tasks, preprocessing of the training data has an impact on the system's performance (Kannan and Gurusamy, 2014; Qu et al., 2015). Since we use neural networks for our classifier and such approaches are data-driven, preprocessing becomes a crucial part of the system.

First of all, we tokenize the data using the *twokenize* package [2] for Python, which was specially designed for tokenization of tweets. This forms the *basic* preprocessing.

For the *advanced* preprocessing, we continue working with the tokenized tweets. We remove punctuation and words containing non-alphanumerical characters (including emojis) and we lowercase all the words. We consider hashtags, words with the # sign, as a special case since they are widely used on Twitter. We do not want to remove them because hashtags can be repetitive and capture some relevant information. For this reason, we just remove the hash sign. Mentions, denoted by the @ sign, are also popular on Twitter but they are often random and we decided that they are not relevant for our classifier. By removing them, we back down from using implicit information captured in the word embeddings about specific users.

Since neural networks cannot handle categorical features as input, we need to convert the input tweets into a numerical representation. Following convention, we make use of pre-trained word embeddings. We use the German Twitter embeddings collected by the researchers at Heidelberg University[3]. The embeddings are trained using word2vec, with 100 dimensions for each word, a context window size of 7 and a minimum occurence of at least 50 times per word in the data. They are also tokenized using the *twokenize* package, hence our decision to use the same library to tokenize the tweets.

We vectorize tweets in the following way: each tweet is a vector with word IDs as its elements. Word IDs correspond to the row of a word in the embeddings matrix. For words which occur only in the training data but not in the embeddings (OOV) we introduce the label UNKNOWN.

2.2 Features

Features have a large impact on performance, especially in domain specific tasks (Schmidt and Wiegand, 2017). The information, relevant for the features, is extracted during preprocessing.

- **Word embeddings** represent one of the most common features in neural NLP (Ruder et al., 2017) . As already introduced above, they are vector-based word representations which are usually pre-trained on large datasets. The embeddings which we use perfectly fit our purpose since they are trained on the Twitter data. It is known that word embeddings trained on out-of-domain data lower performance of systems(Qu et al., 2015). Interestingly, words in the embeddings are true-cased, most nouns appear twice in the embeddings, once true-cased and once lowercased. Therefore the question arises whether we benefit from lowercasing the data. We design our experiments with regard to this fact.

We also tried out other features like emphasizing some categories or considering punctuation, all of which lowered the performance and thus are not included in our final models. We will briefly describe them in Section 4.

3 Model

We experimented with two different neural network architectures, namely convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

[2] https://github.com/nryant/twokenize_py

[3] http://www.cl.uni-heidelberg.de/english/research/downloads/resource_pages/GermanTwitterEmbeddings/GermanTwitterEmbeddings_data.shtml

3.1 CNN

When using CNNs in NL, a window size is defined and a shared weight matrix is trained which is slided along sentences to produce a feature map for every *n*-gram in the sentence where *n* is the window size. Afterwards, we do max pooling over the different features generated and use this as a hidden representation for the sequence. The main benefit is that it is very fast and has few trainable parameters, but can only consider local information. For our final CNN model, we use word embeddings which are initialized with the values from the Heidelberg embeddings and can be trained. We max pool over 1 layer of bi- and trigram features with 64 filters per filter feature. We use a stride of 1 to extract such features and to do max pooling over all the resulting feature maps. Then this hidden representation is fed into a two-layer deep feed-forward network with the first layer having 128 hidden units and the second layer with only two units to perform classification. These parameters were chosen by grid searching over a number of different settings.

3.2 RNN

While using RNNs, one can encode the sequence in a very intuitive way, namely as word representations for every word. In this case, a recurrent neural network starts at the beginning of the sequence and computes a hidden state given the input. This hidden state is propagated through the sequence and updated at each timestep given the current input. The hidden state can also be thought of as the memory of the network and thus is able to capture global information from the sentence. The downsides consist of having more trainable parameters to be optimized using a limited amount of training data. For our final RNN model, we use bidirectional gated recurrent units (GRUs) (Cho et al., 2014) with 50 hidden units for each direction. We also experimented using LSTMs but they perform worse. We think this is explainable by the lower numbers of trainable parameters in the GRU-case which performs better on the small number of training examples we actually have. We performed a max-pool operation over the hidden timesteps because important features at a given timestep may be forgotten towards the end of the sequence and this is a straight-forward way to keep such features. The resulting hidden representation of the sequence (output of the GRUs) is fed into a 4-layer deep feed-forward neural network with 100 hidden units for the first three layers and two neurons in the final layer to perform classification.

Both architectures share some common settings which we describe here: All the layers in the feed-forward neural networks use a dropout-rate of 0.2, a ReLU-activation and L2-regularization with λ 0.0001. We also apply the same dropout to the input sequence and the output representation of the CNN/RNN respectively. We used cross-entropy as a loss function and optimized it using the Adam optimizer with default parameters. Additionally, we weighted offensive tweets twice as much as the non-offensive ones to overcome the imbalance with respect to the number of training examples in the data.

The training was completed using a batch size of 64 examples per batch, with the data shuffled after every epoch and early stopping on a development set with a patience of 4. We selected all the parameters described above by peforming grid search over the training set in a 10-fold cross-validating fashion. The two configurations described above turned out to be the ones yielding the highest average macro F1 measure on different parts of the data. The ensemble method is a loose version of bagging which furthermore increases the robustness and accuracy of the classification. We decided to use it since the high fluctuations in the results were observed when running the same configuration multiple times. A possible reason for this might be the random parameter initialization. Moreover, the problem of finding the right seed in training neural networks also plays an important role here (Bajgar et al., 2018). To counter such behaviour while grid searching, we use 10-fold cross-validation. Finally, using an ensemble of 9 identical models trained on different parts of the data [4], we do predictions based on the majority vote from these models and observe an increase of approximately 2% F1 measure compared to when only one model was used. Our final macro-F1 scores are discussed in the next section.

4 Experiments and Discussion

In Table 1 we show our results with different experiments. All experiments (except the Character CNN) are conducted using the GRU-architecture described above. For each experiment, we use 10-fold cross-validation and in each fold, we split the

[4] one part of training data is reserved for performing early stopping

data in three parts: a training set, a validation set for early stopping and a testset to evaluate. We report the average macro-F1 score over all the ten folds. Our system is optimized for the F1-measure and not for precision and recall, for this reason we report only the first one. Throughout the experiments, we fixed the different splits so that we do not evaluate every experiment on different parts of the data.

In the first row, we just looked up the true-cased version of a word in our embeddings vocabulary. In case we cannot find it there, we try to back off to the lowercased version of the word and otherwise, we just use the UNKNOWN token.

In the second row, we report the results for replacing tokens which appear in a swear word dictionary[5] by a special SWEAR token. The motivation for this feature was the fact that offensive tweets tend to contain some swear words. Interestingly, compared to true-cased data, this significantly improves performance, but by just lowercasing all the words, we get even better results(row 3). This can be justified by the fact that for most nouns, two versions, one true-cased and one lowercased copy, exist in the embeddings and words are not always accurately true-cased in tweets. Thus, by lowercasing all words, we avoid confusing the network with inconsistently true-cased words.

In row 5, we run the model without excluding non-alphanumerical tokens, punctuation and emojis. This again decreases the system's performance. Another issue we tried to overcome here is the out-of-vocabulary (OOV) words treatment, which is common in NLP, especially with small datasets like ours. For this, we use *hunspell* spellchecker [6]. Many tweets contain spelling errors, therefore the spellchecker helps to reduce the number of OOVs: from 2511 OOV tokens to 91. The only problem here is that the spellchecker generates words which are correct but do not occur in the embeddings and therefore are not very useful[7]. This might be an explanation for the slightly worse model performance.

Row 7 shows the results from running our RNN model using LSTMs instead of GRUs. We speculate that since LSTMs have a larger number of trainable parameters, training them on our small training data is producing worse results than GRUs.

[5]https://www.schimpfwoerter.de/

[6]https://pypi.org/project/hunspell/

[7]For instance, SPDler is corrected to Spieler, and Antifantenbrut to quantifizieren.

In row 8, we see the results when using our CNN model with character embeddings. We grid-searched over a number of settings and our best result was a setting with 50 hidden units, a dropout of 0.1 and a batch size of 256. Despite the fact that using character embeddings solves the OOV issue, the model still fails to capture lots of the more broad-scale features in a sentence and therefore yielded very low results compared to our other runs.

Table 2 summarizes the runs which we submit for task I. For each run, we evaluated our system on the last 500 tweets from the training set. The last run consists of an ensemble of 18 models, 9 RNN GRUs and 9 CNNs. We expect that this might slightly boost the performance. We combined the predictions from the two sets of models on the test set and predicted offense tags if at least half the models predicted a tweet as offensive.

Note that the results from Table 1 and 2 are not directly comparable since we evaluate the features using 10-fold cross-validation and the submission runs using the last 500 examples which we excluded during the training time. For the final submission, we retrained the ensembles including the last 500 tweets as additional training material.

Method	**F1(%)**
True-cased	61.6
True-cased + swear word dictionary	74.2
Lowercased	75.9
Lowercased + swear-word dictionary	74.9
Lowercased + non-alpha numerical tokens	72.6
Spellchecker for OOVs	69.7
Using LSTM instead of GRU	68.3
Character embeddings	49

Table 1: Results for different experiments

Submission File	**Ensemble**	**F1(%)**
SaarOffDe_coarse_1.txt	RNN	77.7
SaarOffDe_coarse_2.txt	CNN	78.6
SaarOffDe_coarse_3.txt	CNN + RNN	77.6

Table 2: Submitted runs

5 Conclusion

In this paper, as part of the Germeval 2018 shared task, task I, we implemented neural networks for

the Identification of Offensive Language in German.

We evaluated the two most common neural network approaches for sequence classification on a new German dataset and reported different preprocessing techniques and their impact on the final classification. The most surprising fact seems to be that the best models rely on lowercased words even though the word embeddings we use are true-cased. The overall best performance was achieved with a CNN model with a bi- and trigram filter.

We submit three runs for task I consisting of an ensemble of RNNs[8], CNNs[9] and a combination of both RNNs and CNNs together[10].

References

Lichan Hong, Gregorio Convertino, and Ed Chi. 2011. *Language matters in Twitter: A large scale study* In International AAAI Conference on Weblogs and Social Media.

Wikipedia contributors. The Free Encyclopedia, 30 Jul. 2018. Web. 3 Aug. 2018. *Netzwerkdurchsetzungsgesetz. Wikipedia, The Free Encyclopedia. Wikipedia*. American Psychological Association, Washington, DC.

Anna Schmidt and Michael Wiegand. 2017. *A Survey on Hate Speech Detection using Natural Language Processing* In: Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio 2014. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*

Ondrej Bajgar, Rudolf Kadlec, Jan Kleindienst 2018. *A Boo(n) for Evaluating Architecture Performance* In: Proceedings of the 35th International Conference on Machine Learning, PMLR 80:344-352, 2018.

Yoon Kim 2014. *Convolutional Neural Networks for Sentence Classification* CoRR abs/1408.5882

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi . 2017. *Hate me, hate me not: Hate speech detection on Facebook.* In: Proceedings of ITASEC.

Subbu Kannan and Vairaprakash Gurusamy. 2014. *Preprocessing Techniques for Text Mining.* In: Proceedings of RTRICS.

Sebastian Ruder, Ivan Vulić and Anders Sogaard. 2017. *A Survey of Cross-lingual Embedding Models.*

Lizhen Qu, Gabriela Ferraro, Liyuan Zhou, Weiwei Hou, Nathan Schneider, and Timothy Baldwin. 2015. *Big Data Small Data, In Domain Out-of Domain, Known Word Unknown Word: The Impact of Word Representation on Sequence Labelling Tasks* . In: Proceedings of the 19th Conference on Computational Language Learning.

[8] corresponds to the run SaarOffDe_coarse_1.txt from our submission.

[9] corresponds to SaarOffDe_coarse_2.txt.

[10] corresponds to SaarOffDe_coarse_3.txt.

RuG at GermEval: Detecting Offensive Speech in German Social Media

Xiaoyu Bai[*], **Flavio Merenda**[*∓], **Claudia Zaghi**[*], **Tommaso Caselli**[*], **Malvina Nissim**[*]
[*] Rikjuniversiteit Groningen, Groningen, The Netherlands
[∓] Università degli Studi di Salerno, Salerno, Italy
f.merenda|t.caselli|m.nissim@rug.nl x.bai.5|c.zaghi@student.rug.nl

Abstract

This paper reports on the systems the RuG Team submitted to the GermEval 2018 - Shared Task on the Identification of Offensive Language in German tweets. We submitted three systems to Task 1, targeting the problem as a binary classification task, and only one system for Task 2, addressing a fine-grained classification of offensive tweets in different categories. Preliminary evaluation of the systems has been conducted on a fixed validation set from the training data. The best macro-F1 score for Task 1, binary classification, is 75.45, obtained by an ensemble model composed by a Linear SVM, a CNN, and a Logistic Regressor as a meta-classifier. As for Task 2, multi-class classification, we obtained a macro-F1 of 40.75 using a multi-class Linear SVM.

1 Introduction

The spread of Social Media, and especially of micro-blog platforms such as Facebook and Twitter, has been accompanied by a growth in on-line **hate speech**. Several countries, including the EU, use this expression as a legal term. For instance, the EU Council Framework Decision 2008/913/JHA[1] specifically defines hate speech as "the public incitement to violence or hatred directed to groups or individuals on the basis of certain characteristics, including race, colour, religion, descent and national or ethnic origin". In this work, following (Schmidt and Wiegand, 2017), hate speech is used as an umbrella term to cover a variety of user-generated content phenomena, such as abusive or hostile messages (Nobata et al., 2016), offensive language, cyberbullying (Reynolds et al., 2011; Xu et al., 2012; Zhong et al., 2016), profanity, insults, toxic conversations (Wulczyn et al., 2017), among others.

Although the EU code of conduct on illegal on-line hate speech forces companies to actively remove hate speech messages in their platforms, the phenomenon is so widespread that ways for the automatic classification of on-line content are advocated and necessary (Bleich, 2014; Nobata et al., 2016; Kennedy et al., 2017). The growing interest in this topic is also shown by recent dedicated workshops (e.g. the Abusive Language Workshop (AWL)[2], now at its second edition), datasets in English and other languages[3], and evaluation exercises, such as the Hate Speech Detection task[4] at the EVALITA 2018 Evaluation Campaign for Italian.

The GermEval 2018 - Shared Task focuses on the automatic identification of *offensive language* in German tweets. In the task setting, *offensive language* is defined as "hurtful, derogatory or obscene comments made by one person to another person". The task is organized into two sub-tasks: i.) Task 1, formulated as a binary classification problem, where each tweet has to be classified either as `OFFENSIVE` or as `OTHER`; and ii.) Task 2, formulated as multi-class classification problem, addressing a fine-grained distinction of the offensive tweets, labeled as `INSULT`, `ABUSE`, and `PROFANITY`, as well as the `OTHER` category. According to the Annotation Guidelines (Ruppenhofer et al., 2018), the `OTHER` category is defined as any utterance either having a positive or neutral polarity, or having a negative polarity but not expressing any of the target categories of `INSULT`, `ABUSE`, and `PROFANITY`. Notice also that the category

[1] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=LEGISSUM:l33178

[2] https://sites.google.com/view/alw2018

[3] https://sites.google.com/view/alw2018/resources?authuser=0

[4] http://di.unito.it/haspeedeevalita18

PROFANITY is used to mark utterances that express non-acceptable language (e.g. swearwords) without targeting (an) individual(s), thus basically not expressing hate speech.

This paper illustrates the settings of our participating systems. Although we mainly focused on Task 1, to which we submitted three different runs, we also participated to Task 2 with only one run. Code and outputs are publicly available [5]. In the remainder of the paper, we first discuss some of the resources we used, including additional publicly available data we obtained (Section 2), then describe each of our submitted system runs, including their results on a validation set (Section 3 and Section 4). We also present a discussion on what we tried but did not work during system development (Section 5). We then conclude with a quick overview of previous works in this topic (Section 6) and reflections on future directions (Section 7).

2 Data and Resources

All of our runs, both for Task 1 and for Task 2, are based on supervised approaches, where data (and features) play a major role for the final results of a system. This section illustrates the datasets and language resources used in the final submissions.

2.1 Resources Provided by Organizers

We have been provided with 5009 labeled German tweets as training data. Table 1 illustrates the distribution of the classes for each of the subtask.

Class	Samples
Task 1: Binary task	
OFFENSE	1,688
OTHER	3,321
Task 2: Multi-class task	
ABUSE	1,022
INSULT	595
PROFANITY	71
OTHER	3,321

Table 1: Class distribution in the share task training data for Task 1 and Task 2.

We also experimented with the following resources made available by the organizers:

- German word embeddings pre-trained on either Twitter or Wikipedia data (Cieliebak et al., 2017; Deriu et al., 2017) available from SpinningBytes[6]. Embeddings of sizes 200, 100 and 52 dimensions are available. We used the 52 dimension embeddings.
- A comprehensive list of offensive words in German, obtained from the website http://www.hyperhero.com/de/insults.htm.

2.2 Additional Resources

Source-driven Embeddings A major focus of our contribution is the development of offense-rich, or highly polarized, word embedding representations. To build them, we scraped data from social media communities on Facebook pages. The working hypothesis, grounded on previous studies on on-line communities (Pariser, 2011; Bozdag and van den Hoven, 2015; Seargeant and Tagg, 2018), is that each on-line community represents a different source of data, and consequently, their user-generated contents can be used as proxies for specialized information. We thus acquired *source-driven embeddings* by extracting publicly available comments from a set of German-language Facebook communities that are likely to contain offensive language, and induce word embeddings on the data extracted. The idea is that the embeddings obtained in this manner will be more sensitive to offensive language, with similarly offensive terms being placed closer to each other in the vector space. Table 2 shows the Facebook pages we used (which largely relate to right-wing populist political groups) and the respective number of comments we extracted from each page.

Page Name	Comments
AfD-Fraktion AGH	6,933
Alice Weidel	279,435
Asylflut stoppen	3,461
NPD - Die soziale heimatpartei	138,611
Total	**428,440**

Table 2: List of public Facebook pages from which we obtained comments and number of extracted comments per page.

The embeddings were randomly initialized and generated with the word2vec skip-gram model (Mikolov et al., 2013), using a context window of 5, and minimum frequency 1. The final vocabulary amounts to 313,443 words. These embeddings, referred to as "hate embeddings" here-

[5] https://github.com/malvinanissim/germeval-rug

[6] https://www.spinningbytes.com/resources/wordembeddings/

after, were induced as vectors of 300 dimensions in one setting and of 52 dimensions in another.

We also trained 52 dimensional word embeddings on the shared task training data, using our 52 dimension hate embeddings to initialize the process instead of random initialization. We refer to this further set of embeddings as "hate-oriented embeddings".

To summarize, we generated three sets of word embeddings:

- 300 dimension *hate embeddings* based on Facebook comments;
- 52 dimension *hate embeddings* based on Facebook comments;
- 52 dimension *hate-oriented embeddings*, that incorporate information from the hate embeddings plus the shared task training data.

Extra Training Data Given the dimension of the training data, and especially the lower number of "offensive" tweets, we found an additional dataset of social media messages annotated for offensive language and hate speech, the Political Speech Project (Bröckling et al., 2018). The dataset is part of a journalistic initiative to chart the quality of on-line political discourse in the EU. Almost 40 thousands Facebook comments and tweets between February 21 and March 21, 2018, were collected and manually annotated by an international team of journalists from four countries (France, Italy, Germany, and Switzerland) for level and category of offense. Out of a total of 9,861 utterances from Germany, we extracted and used as extra-training data 549 utterances that were labeled as offensive. We will refer to this extra dataset henceforth as PSP data.

3 Our Submissions

We detail in this section our final submissions to the task, three of which address Task 1, binary classification, and one Task 2, multi-class classification.

3.1 Submission 1: Binary Model with SVM

Our first submission, named `rug_coarse_1.txt`, contains the predictions for the binary task made by an SVM model using various linguistic features.[7] The system was implemented using the Scikit-Learn Python toolkit (Pedregosa et al., 2011).

We performed minimal pre-processing by:

- replacing all mentions/usernames with the generic form *User*;
- removing the line break characters $|LBR|$;
- removing the hash character from all hashtags;
- removing stop words using the Python module `stop-words`[8]

We used two groups of surface features, namely: i.) unigrams and bigrams; and ii.) character n-grams in the range between 3 and 7.

The resulting sparse vector representation of each (training) sample is concatenated with its dense vector representation. The dense vector representation for each tweet is obtained as follows: for every word w in a tweet t, we derived a 52 dimension representation, $\vec{w}$, by means of a look-up in the 52 dimension hate-oriented embeddings. We then performed max pooling over all these word embeddings, $\vec{w}$, to obtain a 52 dimension embedding representation of the full tweet, $\vec{t}$. Words not covered in the hate-oriented embeddings were ignored.

The classifier is a linear SVM with unbalanced class weights. Since the training data is unbalanced and the class `OFFENSE` under-represented, we chose to specify the SVM class weights for `OTHER` and `OFFENSE` as 1 and 3, respectively. We used default values for the other hyper-parameters.

3.2 Submission 2: Binary Model with CNN

Our second submission, `rug_coarse_2.txt`, is based on a Convolutional Neural Network (CNN) architecture for sentence classification (Kim, 2014; Zhang and Wallace, 2015) using Keras (Chollet and others, 2015). The architecture of the model is composed of the following layers:

- A word embeddings input layer using the 300 dimension hate word embeddings (see 2.2);
- A convolution layer;
- A max-pooling layer;
- A fully-connected layer;
- A sigmoid output layer.

[7] In all of our submissions we use the string `XXX` as the dummy label for the task not worked on.

[8] `https://pypi.org/project/stop-words/`

This is a simple architecture with one convolutional layer built on top of a word embedding layer. The embedding layer output corresponds to a tensor of shape three: instances, sequence length and embedding dimension. Later, this output is connected to the convolution layer.

The max-pooling layer output is flattened, concatenated, and fed to the fully-connected layer composed of of 50 hidden-units with the ReLU activation function. The final output layer with the sigmoid activation function computes the probabilistic distribution over the two labels (other network hyperparameters: `Number of filters`: 6; `Filter sizes`: 3, 5, 8; `Strides`: 1; `Activation function`: Rectifier; `Padding`: valid). For our model we chose the binary cross-entropy loss function. As optimization function we employed the Adaptive Moment Estimation (Adam). To train our system, we set a batch size of 64 and we ran it for 10 epochs. To reduce risks of overfitting, we applied two dropout values, 0.6 and 0.8 We added the first dropout layer between the embeddings and the convolution layer, and the second one between the max-pooling and the fully-concatenated layer.

Finally, for this system, the original training data was extended with the 549 PSP data labeled as offensive, thus yielding a new class distribution as shown in Table 3.

Class	Samples
OFFENSE	2,237
OTHER	3,321
Total	**5,558**

Table 3: Class distribution in the training data extended with PSP

3.3 Submission 3: Binary Ensemble Model

Our third submission, named `rug_coarse_3.txt`, is an ensemble model that combines the SVM and CNN models described in Submissions 1 and 2 (Sections 3.1 and 3.2) and a meta-classifier based on a Logistic Regressor classifier.

Each message is composed by 2 groups of surface features, namely, the length of the tweet in terms of number of characters (tweet length), and the number of times an offensive term from the above-mentioned list of offensive German terms (Section 2.1) occurs in the tweet, normalized by the tweet's length (offensive terms), plus the predictions from the Linear SVM and the CNN models. Figure 1 graphically illustrates the representation of each message. The top part illustrates a training sample, while the bottom part a test sample. Such representations are fed as features to the Logistic Regressor, implemented using Scikit-Learn using the default parameters.[9]

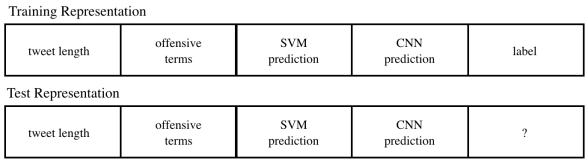


Figure 1: Feature representation of each sample fed to the ensemble model. On top, the representation of a training sample, on bottom, the representation of a test sample.

The predictions outputted by the SVM are in the form of the complementary probabilities for either of the two classes, those by the CNN are in form of the probability of the class `OFFENSE`. The predictions of the SVM and the CNN for the 5009 training samples which we need to feed to the meta-classifier at training time were obtained via 5-fold cross validation. At test time, each system was trained on the full training dataset and produced predictions for each of the test samples, which are then fed as features to the meta-classifier.

Notice that, as described in the previous sections, the CNN was trained on a dataset which featured the addition of the PSP data, while the SVM did not, as this did not prove useful at development time (see Section 5). Thus, in the case of the CNN system, 5-fold cross validation in fact yielded predictions for each of the 5009 training samples, plus the 549 added samples from the PSP data, which were then discarded when training the meta-classifier.

3.4 Submission 4: Multi-Class with SVM

The file named `rug_fine_1.txt` is our only submission to the fine-grained/multi-class task (Task 2), containing predictions by an SVM model. The system and features used are identical to those used in Submission 1 (Section 3.1), except that the SVM class weights for the four classes `OTHER`, `ABUSE`, `INSULT` and `PROFANITY` were set as 0.5, 3, 3 and 4, respectively. `PROFANITY` was

[9] `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

given the highest weight since it is a severely under-represented class.

4 Preliminary Results

Table 4 gives an overview of the preliminary results of our systems in terms of accuracy and macro-F1 score. The systems' results are also compared against two naive baseline models based on the majority class (i.e. OTHER). All scores were obtained by training on 80% of the 5009-sample training data and testing on a fixed development set of 20%.

	Accuracy	**F1 (macro)**
Task 1: Binary task		
Baseline	*65.27*	*39.49*
SVM binary	76.25	71.90
CNN binary	76.85	73.05
Ensemble binary	78.34	74.45
Task 2: Multi-class task		
Baseline	*65.27*	*19.75*
SVM multi-class	71.66	40.75

Table 4: Results of our submitted systems and majority-class baselines in terms of accuracy and macro-average F1 training on 80% of the training set provided, and testing on the remaining 20%.

5 Methods Not Adopted

When developing our system we experimented with a series of additions and variations aimed at improving performance. Not everything worked or made a difference either using cross-validation or randomly picked development sets, but we deem it interesting to report on such attempts in this paper.

Data Given the significant under-representation of the classes INSULT and PROFANITY in the multi-class setting, we experimented with upsampling them by duplicating the samples from these two classes. However, this did not yield any gains in performance. With respect to the additional PSP dataset, we found that unlike the CNN, the SVM did not benefit from the addition of the 549 additional offensive samples and therefore did not adopt this for the final submissions. Moreover, we also experimented with the extension of the training data with *all* samples from the PSP dataset (9,312 neutral/other, 549 offensive), instead of only adding the 549 samples annotated as offensive. However, both the CNN and the SVM suffered from this, likely due to the resulting inflation of the class OTHER.

Representations For the SVM we experimented with different sets of word embeddings which were used to obtain dense-vector representations of full samples in the manner described in Section 3.1. The 52 dimension Twitter and Wikipedia embeddings from SpinningBytes performed similarly. Furthermore, we also tried to join them by concatenating their representations for each word and tested different methods of dealing with the words that are covered by one set of embeddings only. In one setting, we left the embeddings of these words unchanged and used Principle Component Analysis to reduce the dimensions of all other word vectors back to 52. Thus, all embeddings were of 52 dimensions, but those words covered by both sets of embeddings incorporated distributional information from both Twitter and Wikipedia in their representations. In another setting, we obtained unreduced, concatenated embeddings of 104 dimensions, using padding for words which only occur in either the Twitter or the Wikipedia embeddings. Our experiments showed, however, that these alternative word embeddings performed worse than those we used in our final submissions.

Algorithms In the ensemble system we also experimented with using another Linear SVM as the meta-classifier. However, its performance in this capacity was inferior to that of our final choice, i.e. a Logistic Regressor.

6 Related Work

Several models have been presented in the literature to detect hate speech and its related concepts (offensive language, cyberbullying and profanity among others).

The task has been mainly addressed by means of rule-based methods or supervised classifiers. Rule-based methods (De Marneffe and Manning, 2008; Mondal et al., 2017; Pelosi et al., 2017; Xu and Zhu, 2010; Su et al., 2017; Palmer et al., 2017) heavily rely on lexical resources such as dictionaries, thesauri, sentiment lexicons, as well as syntactic patterns and POS relations.

Supervised approaches have shown to obtain good results, although they suffer from limitations as far as the size and domain of the training data is concerned. Support Vector Machine and Convolutional Neural Network classifiers turned out to be efficient algorithms for this task. Simple SVM models with word embeddings (Del Vigna et al., 2017) and TF-IDF n-grams (Davidson et al., 2017)

showed competitive performances. On the other hand, CNN architectures are initialized with word embeddings that can be obtained "on the fly" using the training data or from some pre-trained representations (Badjatiya et al., 2017; Gambäck and Sikdar, 2017; Park and Fung, 2017; Badjatiya et al., 2017). Other classifiers widely employed in literature are LSTMs (Del Vigna et al., 2017; Badjatiya et al., 2017; Gao and Huang, 2017; Chu et al., 2016), and Logistic Regressors (Djuric et al., 2015; Davidson et al., 2017; Gao and Huang, 2017).

A remarkable experiment developed an ensemble classifier combining the predictions of a logistic regression model with the ones obtained with an LSTM neural network (Gao and Huang, 2017).

7 Conclusions and Future Work

This paper reports on the RuG Team submissions to Task 1 and 2 of the GermEval 2018 - Shared Task on the Identification of Offensive Language. Our team focused mainly on Task 1, a binary classification task aiming at classifying German tweets either as `OFFENSIVE` or `OTHER`. In the development of our systems, we put our efforts on the development of embedding representations that could reduce the dependence of the models on the training data, exploiting Facebook on-line communities to generate such data (source-based embeddings). The results on a fixed validation set composed by 20% of the training data have shown that the use of these "hate embeddings" is beneficial. Of the three systems we submitted for Task 1 (a linear SVM, a CNN, and an ensemble model based on the SVM and CNN predictions and extended with basic surface features), the ensemble model obtains the best results (macro-F1 74.45), followed by the CNN (macro-F1 73.05), and, finally, the SVM (macro-F1 71.90).

Task 2, fine-grained classification, was addressed with a simple Linear SVM, using as features word and characters n-grams. The fine-grained classification proved harder than the binary one, also for the limited amount of the training data. The system has a macro-F1 of 40.75 on the same validation set as the binary task.

We are planning to conduct a deep error analysis once the official scores and gold test data will be made available, so as to have a better understanding of the limitations of our models. Furthermore, we also plan to extend the source-based approach to collect polarized embeddings and to test it on other languages as well.

Acknowledgments

The authors want to thank Angelo Basile for his feedback in the early stages of this work. A special thank goes to Rania Wazir for her help in obtaining the PSP data.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Erik Bleich. 2014. Freedom of expression versus racist hate speech: Explaining differences between high court regulations in the usa and europe. *Journal of Ethnic and Migration Studies*, 40(2):283–300.

Engin Bozdag and Jeroen van den Hoven. 2015. Breaking the filter bubble: democracy and design. *Ethics and Information Technology*, 17(4):249–265.

Marie Bröckling, Vincent Coquaz, Alexander Fanta, Alison Langley, Mauro Munafò, Julian Pütz, Francesca Sironi, Leo Thüer, and Rania Wazir. 2018. Political Speech Project. https://rania.shinyapps.io/PoliticalSpeechProject/, May.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Theodora Chu, Kylie Jue, and Max Wang. 2016. Comment abuse classification with deep learning. Von https://web. stanford. edu/class/cs224n/reports/2762092. pdf abgerufen.

Mark Cieliebak, Jan Milan Deriu, Dominic Egger, and Fatih Uzdilli. 2017. A twitter corpus and benchmark resources for german sentiment analysis. In *5th International Workshop on Natural Language Processing for Social Media, Boston, MA, USA, December 11, 2017*, pages 45–51. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

Fabio Del Vigna, Andrea Cimino, Felice DellOrletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. *In Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy*.

Jan Deriu, Aurelien Lucchi, Valeria De Luca, Aliaksei Severyn, Simon Müller, Mark Cieliebak, Thomas Hofmann, and Martin Jaggi. 2017. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th international conference on world wide web*, pages 1045–1052. International World Wide Web Conferences Steering Committee.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30. ACM.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Lei Gao and Ruihong Huang. 2017. Detecting online hate speech using context aware models. *arXiv preprint arXiv:1710.07395*.

George Kennedy, Andrew McCollough, Edward Dixon, Alexei Bastidas, John Ryan, Chris Loo, and Saurav Sahay. 2017. Technology solutions to combat online harassment. In *Proceedings of the First Workshop on Abusive Language Online*, pages 73–77.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mainack Mondal, Leandro Araujo Silva, and Fabricio Benevenuto. 2017. A measurement study of hate speech in social media. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, pages 85–94. ACM.

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. International World Wide Web Conferences Steering Committee.

Alexis Palmer, Melissa Robinson, and Kristy K Phillips. 2017. Illegal is not a noun: Linguistic form for detection of pejorative nominalizations. In *Proceedings of the First Workshop on Abusive Language Online*, pages 91–100.

Eli Pariser. 2011. *The filter bubble: What the Internet is hiding from you*. Penguin UK.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Serena Pelosi, Alessandro Maisto, Pierluigi Vitale, and Simonetta Vietri. 2017. Mining offensive language on social media. In *CLiC-it*.

Kelly Reynolds, April Kontostathis, and Lynne Edwards. 2011. Using machine learning to detect cyberbullying. In *Machine learning and applications and workshops (ICMLA), 2011 10th International Conference on*, volume 2, pages 241–244. IEEE.

Josef Ruppenhofer, Melanie Siegel, and Michael Wiegand. 2018. Guidelines for IGGSA Shared Task on the Identification of Offensive Language. http://www.coli.uni-saarland.de/ miwieg/Germeval/guidelines-iggsa-shared.pdf, March.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics, Valencia, Spain*, pages 1–10.

Philip Seargeant and Caroline Tagg. 2018. Social media and the future of open debate: A user-oriented approach to Facebook's filter bubble conundrum. *Discourse, Context & Media*.

Hui-Po Su, Zhen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing profanity in chinese text. In *Proceedings of the First Workshop on Abusive Language Online*, pages 18–24.

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.

Zhi Xu and Sencun Zhu. 2010. Filtering offensive language in online communities using grammatical relations. In *Proceedings of the Seventh Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, pages 1–10.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666. Association for Computational Linguistics.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J Miller, and Cornelia Caragea. 2016. Content-driven detection of cyberbullying on the instagram social network. In *IJCAI*, pages 3952–3958.

upInf - Offensive Language Detection in German Tweets

Bastian Birkeneder[1], Jelena Mitrović[2], Julia Niemeier[3], Leon Teubert[4], and Siegfried Handschuh[5]

[1,2,3,4,5]Department of Computer Science and Mathematics, University of Passau
[5]Chair of Data Science, University of St. Gallen

birkeneder | niemeier | teubert @fim.uni-passau.de
jelena.mitrovic | siegfried.handschuh @uni-passau.de

Abstract

As part of the shared task of GermEval 2018 we developed a system that is able to detect offensive speech in German tweets. To increase the size of the existing training set we made an application for gathering trending tweets in Germany. This application also assists in manual annotation of those tweets. The main part of the training data consists of the set provided by the organizers of the shared task. We implement three different models. The first one follows the n-gram approach. The second model utilizes word vectors to create word clusters which contributes to a new array of features. Our last model is a composition of a recurrent and a convolutional neural network. We evaluate our approaches by splitting the given data into train, validation and test sets. The final evaluation is done by the organizers of the task who compare our predicted results with the unpublished ground truth.

1 Introduction

According to Domo (2018), in June 2018, Twitter users generated 473,400 tweets per minute. Due to this enormous amount of data it is reasonable to assume that many offensive micro-posts are published on a daily basis. The goal of the shared task of IGGSA (2018), which we participate in, is to find and evaluate approaches for classifying those tweets. We contribute to the coarse task which consists of the binary classification problem whether a tweet is considered offensive or not. The second task includes a fine-grained differentiation in the four classes: profanity, insult, abuse and other.
An important task in social media and natural language processing is to detect offensive speech and profanity. The concrete challenge of this assignment is that most papers discuss this topic for English language and regard semantic and syntactic differences of other languages. In addition, only a limited amount of data is publicly available for examples in German. In this paper we try to overcome this impediment by extracting trending German tweets over a time period of three months. We annotated part of this data and combined these with the provided training data of the shared task to train our three models. Our collected data is publicly available in our GitHub repository[1].
Our paper is divided as follows. First we give a short overview of work done in the field of offensive language detection as well as the analysis of German tweets. The next section describes the data we have used and acquired. In section 4, we describe our three approaches and evaluate their performance in section 5. Lastly, we conclude our results and describe possible future work in this field of research.

2 Related Work

Nobata et al. (2016) describe an approach to detect abusive language in English comments of 'Yahoo! Finance and News'. They combine lexical features like n-gram, as well as linguistic and syntactic features with distributional semantics and evaluate their data using four datasets. The resulting f1-score on the Yahoo comments totals 83.6%. To compare the approach to other models they also predicted on the 'WWW2015' dataset were they reached an f1-score of 78.3%.
In Razavi et al. (2010), two data sets are used: log files of the 'Natural Semantic Module' that contain questions of users and 'Usenet newsgroup' messages that have already been annotated. The two

[1]https://github.com/upInf/germeval2018

data sets are combined to get short sentences with abusive language as well as long sentences with sarcasm and irony. They used a three-level classification system and created a dictionary of flame patterns containing weights from one to five. In the first level, they selected the most discriminative features using a Complement Naive Bayes classifier. The result of this phase was subsequently analyzed using a Multinomial Updateable Naive Bayes classifier. The last step utilizes the DecisionTable/Naive Bayes hybrid classifier. Their composite system reached an accuracy of 96.72% on the test set.

Chen et al. (2012) introduced a framework called 'Lexical Syntactic Feature' that combines the offensiveness rating of a word and its context. The offensiveness rating is determined by two lexicons. The context is derived by parsing sentences into dependency sets. To get a rating for the whole sentence, these features are combined linearly. This approach is compared to standard text mining approaches like n-grams, bag-of-words and an appraisal approach using YouTube comments. They conclude that their self-defined framework performs better than the compared baseline approaches.

Xiang et al. (2012) describe a method to detect offensive English tweets using topical features. Due to the colloquial fashion of tweets, they apply a self designed preprocessing algorithm. To annotate a topic for each tweet, they create a bootstrapping algorithm. The classification is done with the Latent Dirichlet Allocation described in Blei, Ng, and Jordan (2003). In addition, they use a keyword matching technique assigning a binary indicator whether at least one word is offensive.

Ross et al. (2017) propose a method for annotating German tweets concerning the European refugee crisis. They aim to measure the reliability of given ratings and observe a very low agreement. Tweets were processed by three pairs of annotators. The data set is divided into six equal parts, so the pairs could be rotated after each step. The first annotator is asked to decide whether the tweet is offensive or not. The second one additionally provides a rating on a 6-point Likert scale from one (not offensive at all) to six (very offensive). They conclude that offensive language detection should be considered a regression problem rather than a binary classification.

In the work of Davidson et al. (2017) an approach to classifying English text into three different categories is presented. They distinguish between hate speech, offensiveness and other texts. Based on a hate speech lexicon generated from user ratings, a Twitter corpus of 25,000 tweets has been compiled and manually labeled. Zhou, Sun, Liu, and F. C. M. Lau (2015) proposed a combination of a recurrent and a convolutional neural network for sentence representation and text classification. The convolutional layer extracts n-gram features that are fed forward towards a Long Short-Term Memory to capture long term dependencies. For evaluation they used the Stanford Sentiment Treebank to classify movie reviews. In the binary classification task they accomplished an accuracy of 87.8% and for the fine-grained five-class classification 49.2%.

3 Corpus

Our training corpus is composed of different sources.

3.1 Data Acquisition

The initial training data is provided by the organizers of the shared task. We initially started with a set of approximately 5,000 German tweets labeled either *offensive* or *other*. In order to increase the size of our training data, we acquired additional tweets and labeled them manually.

Compiling our own data set has several advantages. Having a broader spectrum of learning data could lead to improved results and finer tuned models. As stated by Ross et al. (2017), the agreement on whether a tweet is perceived as offensive or not can depend on personal opinions. Additionally, the empirical analysis of the agreement between two or more annotators can be used to evaluate the validity of the trained model. A data set labeled by only one person may tend to reflect their personal mindset, since for example tweets can be ambiguous or opinions can diverge.

Gathering tweets More than 750,000 tweets were gathered during a time interval of three months, to collect a large enough spectrum of current trends and topics. Therefore, tweets of the top 50 German Twitter trends were fetched every 15 minutes, amounting to an average of 11,000 tweets per day. The data was stored in a *mysql* database. Duplicates are avoided by a unique index constraint on the text column. In contrast to the training data we anonymized usernames. Therefore, any occurence of a tagged username is replaced by @*name*. All hyperlinks in posts were

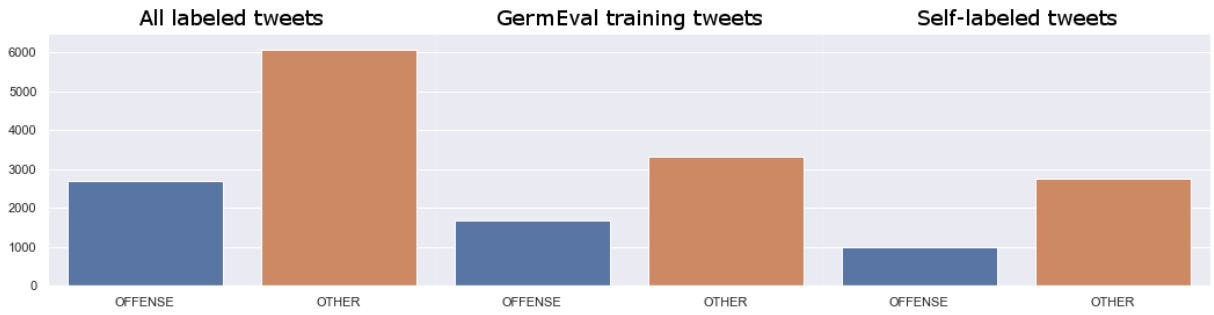


Figure 1: Distribution of offensive tweets per data set

shortened to *http://*. Hence it is recognizable that a link is posted, but the content of the link is not evaluated.

Annotating tweets A supplementary goal is to calculate the agreement between multiple annotators as illustrated by Ross et al. (2017). Therefore, a database relation for multiple ratings was installed. To assign values of offensiveness to the tweets stored in the database, an annotation client was developed. This software can be used in two different modes: the first one is used to annotate new tweets and hereby extend the Twitter corpus. As at least two annotations for one tweet are needed to calculate an agreement score, the second mode of the program displays tweets that have already been annotated by exactly one person. In total, about 4,000 tweets were annotated containing about 1,000 offensive tweets.

3.2 Data Composition

In the following sections three different data sets are used:

GermEval Training Tweets
This data set was provided by the organizers of the shared task. It contains about 5,000 tweets that are divided into offensive and non offensive. Subsequently, this data set is abbreviated by *GETT*.

Self-labeled Tweets
The data collected using the procedure as described in section 3.1 Data Acquisition was combined with *GETT*. A tweet is marked as offensive if at least one annotator labeled it that way. We refer to this data set in the following by *SLT*

Tweets by Davidson
For comparison we used the tweets provided by Davidson et al. (2017) [2]. These are about 25,000 English tweets divided in 19,200 offensive, 1,500 hatespeech and 4,200 other tweets. For our binary classification task, we merged the classes offensive and hatespeech into one class. This set is from now on abbreviated as *TD*.

Our data sets were split into training (80%), validation (10%), and test (10%) set respectively.
Figure 1 shows the arrangement of offensive vs. non-offensive tweets. In both training sets, the amount of non-offensive tweets exceeds the offensive ones. Caused by this imbalanced distribution, the accuracy measure would be ambiguous, so we choose the harmonic mean of precision and recall, known as f1-score.

4 System

We implemented three different models. Therefore, we use the modules *NLTK* from Loper and Bird (2002), *scikit-learn* from Pedregosa et al. (2011), *Keras* from Chollet et al. (2015) and *Gensim* from Řehůřek and Sojka (2010).

4.1 N-gram Model

We choose the n-gram model as our baseline approach, because this basic approach is able to reach good results in text classification tasks. This enables us to evaluate the performance of our other models.
We start by tokenizing and stemming all words in a tweet. Furthermore, we remove the # sign from all hashtags, because these hashtags used in the context of a sentence can often be replaced by the topic-keyword alone, for example *"Schon merkwürdig, dass #Oezil von der Politik des #Erdogan-Fotos*

[2] https://github.com/t-davidson/hate-speech-and-offensive-language

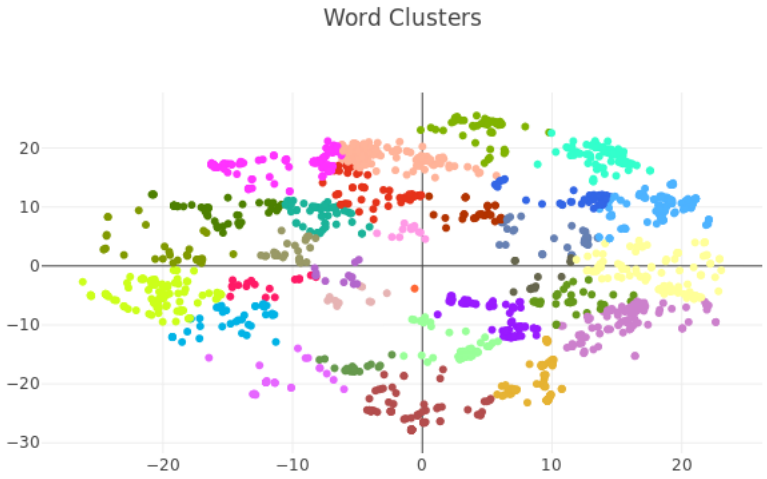


Figure 2: 30 word clusters with k-means

nichts wissen will [...]". In the next step, we remove all usernames and hyperlinks.

We use the *TF-IDF-Vectorizer* from sklearn to retrieve our word counts weighted by the *term frequency-inverse document frequency* of all uni-, bi- and trigrams.

For this model we compare several classifiers, a Support Vector Machine (SVM), Naive Bayes classifier, and a Decision Tree. We implement these models with sklearn, namely the classes *SGDClassifier*, *BernoulliNB*, *DecisionTreeClassifier*. Our SVM reaches the highest f1-score. We conduct a grid search on the validation set to fine tune our hyper-parameters and obtain the best estimator.

The submission file is named *upInf_coarse_1.txt*.

4.2 Word Clustering

Mikolov et al. (2013) proposed a vector space model for word embeddings, such that words that share a similar context in a corpus have related vectors. Our second approach tries to use the advantage of these word vectors for binary classification of tweets. To create those vectors, a *word2vec* model based on the *SLT* vocabulary has been trained. Since the *TD* data set is in English, we acquired an additional corpus of 1.6 million tweets provided by Go, Bhayani, and Huang (2009) to train an English word2vec embedding. Best results were observed without stemming and stop word removal. We choose a 100-dimensional vector and a window size of five tokens. Training the model with 100 epochs turned out to be sufficient.

The goal of this approach is to add some semantic context to the model. The word vectors were clustered with a *k-means* algorithm. Baker and McCallum (1998) state that the clustering of words can provide several advantages. First of all, it can generate semantic word groups. Furthermore, clustering can lead to higher classification accuracy. One drawback of n-gram models is the curse of dimensionality. The semantic word clustering offers a highly reduced dimensional representation.

A sample implementation has been done by Duarte (2018). After a parameter search, we set the number of clusters to 1,000. After the computation of our clusters, every word is related to a nearest centroid. Thus a 1,000 dimensional vector for every sentence can be determined. Every dimension represents the accumulated count of words in the cluster for one tweet. To increase the feature spectrum, a standard TF-IDF vector is attached. Afterwards, we reduce the dimensionality by applying a *SelectFromModel* feature selection. Subsequently, several classifiers are tested with cross-validation and are evaluated against our test sets. The best results are reached by the Naive Bayes classifier.

In figure 2 a visualization of this approach is presented. It shows a simple 2D representation of the 50,000 most frequent words of our own Twitter corpus.

The prediction results can be found in *upInf_coarse_2.txt*.

4.3 C-LSTM

One of the main disadvantages of bag-of-words models is the information loss regarding the word order. Neural network models have shown to perform remarkable results in language modeling tasks. Recurrent neural networks (RNN) are particularly well-suited to model word sequences, since they are able to capture long-term dependencies as described by Sundermeyer, Schlüter, and Ney (2012). Hochreiter and Schmidhuber (1997) developed long short-term memory (LSTM) networks to overcome the vanishing and exploding gradient problem of RNN.
Convolutional neural networks (CNN), first described by Krizhevsky, Sutskever, and Hinton (2012), are another class of neural networks and generally used for object recognition and image classification. CNN can be utilized for sentence modeling by extracting n-gram features through convolutional filters. Similar to RNN, CNN can learn short and long-range relations through pooling operations.
Zhou, Sun, Liu, and F. Lau (2015) suggest a unified model of CNN and LSTM, called C-LSTM for sentence representation and text classification, where the CNN is used to extract n-gram features, which are fed towards an LSTM to capture the sentence semantics.
This model is the foundation of our third approach. The C-LSTM is implemented with *keras* using the *tensorflow* backend. Preprocessing is performed similar to the other implemented models, except we skip stemming and split hashtags into two tokens, the actual hashtag sign (#) and the following keyword. We used our own generated 100-dimensional *Word2Vec* model to initialize the embedding layer, but limit our vocabulary size to the 20,000 most frequent tokens. Unknown words are initialized using a random word embedding with values from the uniform distribution [-0.25, 0.25]. The word vectors are then fine-tuned during the training of our model. To fix the input length, each sentence with a length less than 30 tokens is padded with the representation of an empty string. Sentences which exceed this limit are cut off at the end.
The convolution layer of our model consists of five concatenated one-dimensional convolution layers. Each layer encloses a filter vector of different length, sliding over the embedding vectors of a token sequence. The length n of these vectors range between one to five tokens and allows the detection of n-gram features. ReLu is chosen as the nonlinear activation function. The generated feature maps are then concatenated and fed forward towards the LSTM layer.
The LSTM, which is used in this layer, uses the standard architecture, first described by Hochreiter and Schmidhuber (1997). The memory dimension of the LSTM layer is set to 100.
As a consequence of the binary classification task, our output layer consists of a single neuron and we choose the sigmoid function as activation function. A value greater or equal than 0.5 indicates the label 'OFFENSE', whereas a lower value indicates the label 'OTHER'. Furthermore, we implement two dropout layers with a dropout rate of 0.3 for regularization and to prevent over-fitting. These layers are applied respectively before the convolution layer and after the LSTM layer.
Stochastic gradient descent (SGD) with the optimizer Adam, as described by Kingma and Ba (2014), is used to update the model parameters. Cross-entropy loss is chosen to measure the performance of our model.
A model description can be found in figure 4 in the appendix.
The results of this approach are submitted as *upInf_coarse_3.txt*.

5 Results

Our systems are named according to section 4.
The final results on our test sets are displayed in figure 3.

5.1 Agreement

As mentioned in section 3.1, about 700 of our tweets were annotated by at least two annotators so we are able to calculate an agreement score. Since we want to compare our results with Ross et al. (2017), we calculate the Krippendorff α (Krippendorff, 2004). "This] is a reliability coefficient developed to measure the agreement among observers, coders, judges, raters [...]" (Krippendorff, 2008). Our annotations show a total agreement accuracy of 84% and a Krippendorff α of 78%. In contrast, Ross et al. (2017) reach an α of 38% at the annotations of the experts.

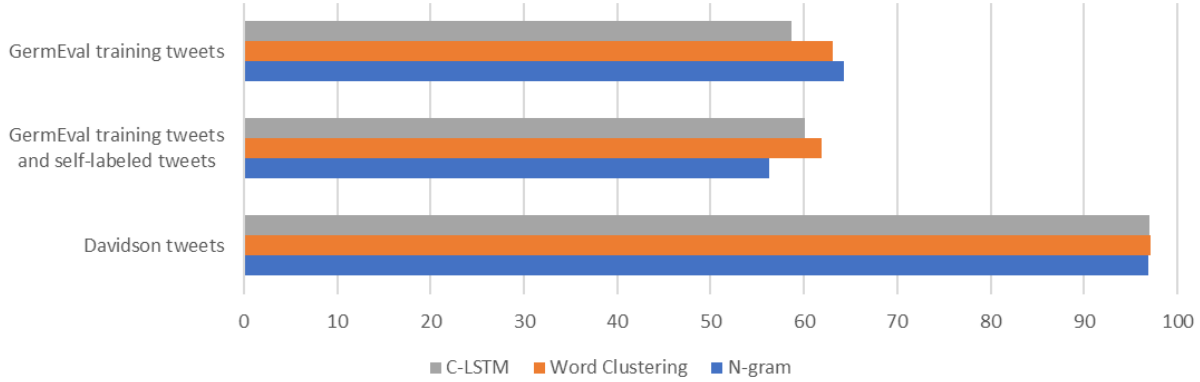


Figure 3: Results of different systems per data set

5.2 N-gram Model

By tuning our n-gram model we are able to achieve an accuracy of 77.84% at an f1-score of 63.49% with the *SGDClassifier* on the *GETT* data set. For the *SLT* data set, this model performs worse with the *SGDClassifier* and just reaches 59.69% f1-score and an accuracy of 67.73%. For the *TD* data set, the best prediction was achieved using the *AdaBoostClassifier* with a Decision Tree as base estimator. The f1-score reaches 96.89% and the accuracy 94.91%.

5.3 Word Clustering

A final f1-score of 65.55% with an accuracy of 75.44% can be reached with a *BernoulliNB* on the *GETT* data set. As in the first approach the system performs worse on the *SLT* data set, where an f1-score using the Naive Bayes classifier of 61.94% is accomplished. A prediction f1-score of 97.11% with the *AdaBoostClassifier* is the optimal result that can be achieved on the *TD* data set.

5.4 C-LSTM

The C-LSTM achieves an accuracy of 74.85% and an f1-score of 56.25% on the *GETT* data set. On the *SLT* data set, this model reaches an accuracy of 74.83% and an f1-score of 60.14%. Similar to our other models, the C-LSTM performs well on the *TD* data set with an accuracy of 95.00% and an f1-score of 96.99%.

6 Discussion

Agreement Our high Krippendorff α can be explained with our search queries. We tried to avoid specific keywords, which could by itself indicate profanity or offensive language. Despite our effort to search for controversial topics, the majority of tweets can be considered as objectively not offensive. Nevertheless, we can agree with the observation of Ross et al. (2017) that a binary classification for offensiveness is a difficult and subjective task.

Classification Task All of our models perform similarly and produce comparable results. For the *GETT* data set, the n-gram model achieved the best scores. It has become evident that our initial goal to improve the classification accuracy by increasing the size of our training set could not be reached.
The first reason for this could be the differing annotations caused by the missing ground truth in the nature of this task. The offensiveness of a tweet is a subjective measure that is difficult to quantify. We tried to annotate according to the provided guidelines, but still observed inconsistencies. Another explanation could be certain characteristics of the German language especially composite words in which words are combined to generate new ones. In our models, a unique word in a vocabulary is embedded by one specific token. Hence certain composite words which could be considered as offensive, like for example *"Hurensohnbande"*, occur less frequently in our training data and therefore affect our results.
Furthermore, it can be difficult to grasp the full context of a random tweet. Tweets are often responses or comments on other tweets. With only fragments of a conversation, the true intention of the author is difficult to determine.

7 Conclusion

Using more than 700,000 tweets crawled from the top 50 Twitter trends for over three months and combining them with the training set of GermEval

2018, three different models were trained to detect offensive speech. Regarding the labeling of our own Twitter corpus, we observe an agreement score of 77.5% measured using Krippendorff α.
The baseline classification approach consists off an n-gram model using Tfidf-Vectorization and an SVM. Subsequently, we combined this approach with a K-Means Word Clustering of a self-trained *word2vec* model. The third system was designed using a C-LSTM.
On the *GETT* data set, these models reach an f1-score between 55% and 65%. Most models could not be improved by extending the data set. The effectiveness of the classifier is likely to depend on the quality of annotations and due to the subjective nature of this task, it is difficult to maintain a consistent set of training data.

8 Future Work

An issue concerning tweet data is the lack of context. Most tweets refer to external resources like articles, images or videos. This information is not available to the classifiers. Tweet meta data like whether the tweet is a response to another tweet or if the user was offensive before could represent useful context and affect the decision-making process. Therefore, including this type of information in the training data could be useful.
Another improvement of our models, which is suggested by Davidson et al. (2017), might be to include part-of-speech (POS) tagging. Since no sufficient POS-tagger is applicable for German language, it is recommended to train a separate classifier. A possible implementation was published by Konrad (2016).

References

Baker, L. Douglas and Andrew Kachites McCallum (1998). "Distributional Clustering of Words for Text Classification". In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '98. Melbourne, Australia: ACM, pp. 96–103. ISBN: 1-58113-015-5.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent Dirichlet Allocation". In: *J. Mach. Learn. Res.* 3, pp. 993–1022. ISSN: 1532-4435.

Chen, Ying et al. (2012). "Detecting Offensive Language in Social Media to Protect Adolescent Online Safety". In: *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust*. SOCIALCOM-PASSAT '12. Washington, DC, USA: IEEE Computer Society, pp. 71–80. ISBN: 978-0-7695-4848-7.

Chollet, François et al. (2015). *Keras*. `https://keras.io`.

Davidson, Thomas et al. (2017). "Automated Hate Speech Detection and the Problem of Offensive Language". In: *Proceedings of the 11th International AAAI Conference on Web and Social Media*. ICWSM '17. Montreal, Canada, pp. 512–515.

Domo, Inc. (2018). *Data Never Sleeps 6.0*. `https://www.domo.com/learn/data-never-sleeps-6`. Accessed 29 Jul 2018.

Duarte, Pedro Arthur (2018). *Sentiment Analysis of IMDB Reviews*. `https://www.kaggle.com/pedroarthur/sentiment-analysis-of-imdb-reviews/notebook`. Accessed 30 Jul 2018.

Go, Alec, Richa Bhayani, and Lei Huang (2009). "Twitter Sentiment Classification using Distant Supervision". In: *Processing*, pp. 1–6.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

IGGSA, Interest Group on German Sentiment Analysis (2018). *Germeval Task 2018*. `https://projects.fzai.h-da.de/iggsa/`. Accessed 29 Jul 2018.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Konrad, Markus (2016). *Accurate Part-of-Speech Tagging of German Texts with NLTK*. `https://datascience.blog.wzb.eu/2016/07/13/accurate-part-of-speech-tagging-of-german-texts-with-nltk/`. Accessed 03 Aug 2018.

Krippendorff, Klaus (2004). "Reliability in Content Analysis: Some Common Misconceptions and Recommendations". In: *Human Communication Research* 30.3, pp. 411–433.

– (2008). "Computing Krippendorff's Alpha-Reliability".

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Ad-*

vances in neural information processing systems, pp. 1097–1105.

Loper, Edward and Steven Bird (2002). "NLTK: The Natural Language Toolkit". In: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*. ETMTNLP '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, pp. 63–70.

Mikolov, Tomas et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Nobata, Chikashi et al. (2016). "Abusive Language Detection in Online User Content". In: *Proceedings of the 25th International Conference on World Wide Web*. WWW '16. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, pp. 145–153. ISBN: 978-1-4503-4143-1.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Razavi, Amir H. et al. (2010). "Offensive Language Detection Using Multi-level Classification". In: *Advances in Artificial Intelligence*. Ed. by Atefeh Farzindar and Vlado Kešelj. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 16–27. ISBN: 978-3-642-13059-5.

Řehůřek, Radim and Petr Sojka (2010). "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. `http://is.muni.cz/publication/884893/en`. Valletta, Malta: ELRA, pp. 45–50.

Ross, Björn et al. (2017). "Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis". In: *CoRR* abs/1701.08118.

Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney (2012). "LSTM neural networks for language modeling". In: *Thirteenth annual conference of the international speech communication association*.

Xiang, Guang et al. (2012). "Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus". In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. CIKM '12. Maui, Hawaii, USA: ACM, pp. 1980–1984. ISBN: 978-1-4503-1156-4.

Zhou, Chunting, Chonglin Sun, Zhiyuan Liu, and Francis Lau (2015). "A C-LSTM neural network for text classification". In: *arXiv preprint arXiv:1511.08630*.

Zhou, Chunting, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau (2015). "A C-LSTM Neural Network for Text Classification". In: *CoRR* abs/1511.08630.

A C-LSTM Architecture

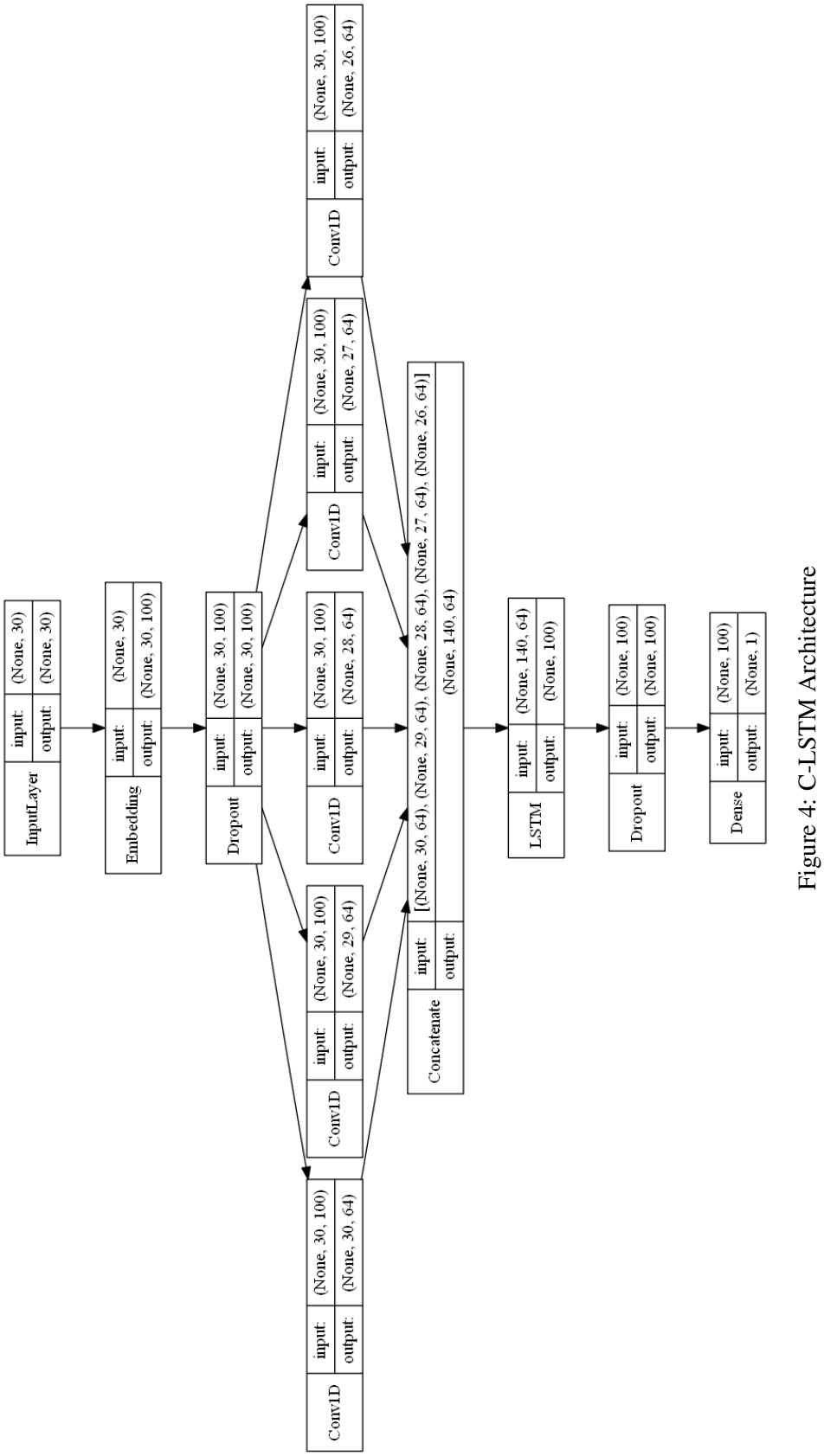


Figure 4: C-LSTM Architecture

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

Michele Corazza[†], Stefano Menini[‡], Pinar Arslan[†], Rachele Sprugnoli[‡]
Elena Cabrio[†], Sara Tonelli[‡], Serena Villata[†]
[†]Université Côte d'Azur, CNRS, Inria, I3S, France
[‡]Fondazione Bruno Kessler, Trento, Italy
{michele.corazza,pinar.arslan}@inria.fr
{menini,sprugnoli,satonelli}@fbk.eu
{elena.cabrio,serena.villata}@unice.fr

Abstract

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

1 Introduction

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

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

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

2 Data

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

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

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

3 System Description

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

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

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

3.1 Pre-processing

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

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

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

3.2 Feature description

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

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

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

- **Social-network specific features**: a collection of features that capture some aspects of social media interactions is considered. They include the number of hashtags and mentions, the number of exclamation and question marks, the number of emojis, the number of words that are written in uppercase.

3.3 The Recurrent Neural Network model

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

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

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

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

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

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

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

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

3.4 System description - Task 1

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

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

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

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

3.5 System description - Task 2

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

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

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

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

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

4 Evaluation

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

4.1 Preliminary Results - Task 1

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

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

Table 1: Results for InriaFBK_coarse_1

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

Table 2: Results for InriaFBK_coarse_2

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

Table 3: Results for InriaFBK_coarse_3

4.2 Preliminary Results - Task 2

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

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

Table 4: Results for InriaFBK_fine_1

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

Table 5: Results for InriaFBK_fine_2

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

Table 6: Results for InriaFBK_fine_3

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

Please note that the three runs submitted to the shared evaluation for each Task were obtained by

re-training the models with the configurations described above, keeping the same validation set (20%) and merging the training and the test introduced in Section 3 to increase the amount of training data.

5 Conclusions

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

Acknowledgments

Part of this work was funded by the CREEP project (`http://creep-project.eu/`), a Digital Wellbeing Activity supported by EIT Digital in 2018. This research was also supported by the HATEMETER project (`http://hatemeter.eu/`) within the EU Rights, Equality and Citizenship Programme 2014-2020.

References

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

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August. Association for Computational Linguistics.

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

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

François Chollet et al. 2015. Keras. `https://keras.io`.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM.

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

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

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

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

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

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

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

Transfer Learning from LDA to BiLSTM-CNN for Offensive Language Detection in Twitter

Gregor Wiedemann **Eugen Ruppert** **Raghav Jindal** **Chris Biemann**
Language Technology Group
Department of Informatics
University of Hamburg, Germany
{gwiedemann, ruppert, biemann}@informatik.uni-hamburg.de
raghavjindal2003@gmail.com

Abstract

We investigate different strategies for automatic offensive language classification on German Twitter data. For this, we employ a sequentially combined BiLSTM-CNN neural network. Based on this model, three transfer learning tasks to improve the classification performance with background knowledge are tested. We compare 1. Supervised category transfer: social media data annotated with near-offensive language categories, 2. Weakly-supervised category transfer: tweets annotated with emojis they contain, 3. Unsupervised category transfer: tweets annotated with topic clusters obtained by Latent Dirichlet Allocation (LDA). Further, we investigate the effect of three different strategies to mitigate negative effects of 'catastrophic forgetting' during transfer learning. Our results indicate that transfer learning in general improves offensive language detection. Best results are achieved from pre-training our model on the unsupervised topic clustering of tweets in combination with thematic user cluster information.

1 Introduction

User-generated content in forums, blogs, and social media not only contributes to a deliberative exchange of opinions and ideas but is also contaminated with offensive language such as threats and discrimination against people, swear words or blunt insults. The automatic detection of such content can be a useful support for moderators of public platforms as well as for users who could receive warnings or would be enabled to filter unwanted content.

Although this topic now has been studied for more than two decades, so far there has been little work on offensive language detection for German social media content. Regarding this, we present a new approach to detect offensive language as defined in the shared task of the GermEval 2018 workshop.[1] For our contribution to the shared task, we focus on the question how to apply transfer learning for neural network-based text classification systems.

In Germany, the growing interest in hate speech analysis and detection is closely related to recent political developments such as the increase of right-wing populism, and societal reactions to the ongoing influx of refugees seeking asylum (Ross et al., 2016). Content analysis studies such as Kreißel et al. (2018) have shown that a majority of hate speech comments in German Facebook is authored by a rather small group of very active users (5% of all accounts engaging in hate speech). The findings suggest that already such small groups are able to severely disturb social media debates for large audiences.

From the perspective of natural language processing, the task of automatic detection of offensive language in social media is complex due to three major reasons. *First*, we can expect 'atypical' language data due to incorrect spellings, false grammar and non-standard language variations such as slang terms, intensifiers, or emojis/emoticons. For the automatic detection of offensive language, it is not quite clear whether these irregularities should be treated as 'noise' or as a signal. *Second*, the task cannot be reduced to an analysis of word-level semantics only, e.g. spotting offensive keyterms in the data. Instead, the assessment of whether or not a post contains offensive language can be highly dependent on sentence and discourse level semantics, as well as subjective criteria. In a crowd-sourcing experiment on 'hate speech' annotation, Ross et al. (2016) achieved only very low inter-rater agreement between annotators. Offensive language is

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

probably somewhat easier to achieve agreement on, but still sentence-level semantics and context or 'world knowledge' remains important. *Third*, there is a lack of a common definition of the actual phenomenon to tackle. Published studies focus on 'hostile messages', 'flames', 'hate speech', 'discrimination', 'abusive language', or 'offensive language'. Although certainly overlapping, each of these categories has been operationalized in a slightly different manner. Since category definitions do not match properly, publicly available annotated datasets and language resources for one task cannot be used directly to train classifiers for any respective other task.

Contribution: For the offensive language detection presented in this paper, our approach is to use semi-supervised text classification to address all of the three challenges. In order to account for atypical language, we use sub-word embeddings to represent word tokens, words unseen during training, misspelled words and words specifically used in the context of social media such as emojis. To represent complex sequence information from tweets, we use a neural network model combining recurrent (e.g. Long-Short term memory, LSTM) (Hochreiter and Schmidhuber, 1997) and convolutional (CNN) layers. Both learning architectures, LSTM and CNN, have already been employed successfully in similar text classification tasks such as sentiment analysis (Kim, 2014). We expect the combination of LSTM and CNN to be especially useful in the context of transfer learning.

The main contribution of this paper is to investigate potential performance contributions of transfer learning to offensive language detection. For this, we investigate three different approaches to make use of knowledge learned by one task to improve classification for our actual offensive language task. To pre-train our BiLSTM-CNN network, we employ 1. Supervised category transfer: social media data annotated with near-offensive language categories, 2. Weakly-supervised category transfer: tweets annotated with emojis they contain, and 3. Unsupervised category transfer: tweets annotated with topic clusters obtained by Latent Dirichlet Allocation (LDA) (Blei et al., 2003). Further, we investigate the effect of three different transfer learning strategies on the classification performance to mitigate the effect of 'catastrophic forgetting'.[2] The results indicate that transfer learning on generic topic clusters of tweets derived from an LDA process of a large Twitter background corpus significantly improves offensive language detection.

We present our findings in the following structure: Section 2 addresses related work to our approach. In Section 3, we introduce the details of the GermEval 2018 Shared Task together with our background corpora for knowledge transfer. In Section 4, we describe our BiLSTM-CNN model for text classification. Section 5 introduces the different transfer learning setups we investigate. To evaluate these setups, we conduct a number of experiments for which results are presented in Section 6. This section also contains a brief discussion of errors made by our model. Finally, we give some concluding remarks.

2 Related Work

Automatic detection of offensive language is a well-studied phenomenon for the English language. Initial works on the detection of 'hostile messages' have been published already during the 1990s (Spertus, 1997). An overview of recent approaches comparing the different task definitions, feature sets and classification methods is given by Schmidt and Wiegand (2017). A major step forward to support the task was the publication of a large publicly available, manually annotated dataset by Yahoo research (Nobata et al., 2016). They provide a classification approach for detection of abusive language in Yahoo user comments using a variety of linguistic features in a linear classification model. One major result of their work was that learning text features from comments which are temporally close to the to-be-predicted data is more important than learning features from as much data as possible. This is especially important for real-life scenarios of classifying streams of comment data. In addition to token-based features, Xiang et al. (2012) successfully employed topical features to detect offensive tweets. We will build upon this idea by employing topical data in our transfer learning setup. Transfer learning recently has gained a lot of attention since it can be easily applied to neural network learning architectures. For instance, Howard and Ruder (2018) propose a generic transfer learning setup for

[2]Catastrophic forgetting refers to the phenomenon that during supervised learning of the actual task in a transfer learning setup the update of model parameters can overwrite knowledge obtained by the previously conducted training task. This will eventually eliminate any positive effect of pre-training and knowledge transfer from background corpora.

text classification based on language modeling for pre-training neural models with large background corpora. To improve offensive language detection for English social media texts, a transfer learning approach was recently introduced by Felbo et al. (2017). Their 'deepmoji' approach relies on the idea to pre-train a neural network model for an actual offensive language classification task by using emojis as weakly supervised training labels. On a large collection of millions of randomly collected English tweets containing emojis, they try to predict the specific emojis from features obtained from the remaining tweet text. We will follow this idea of transfer learning to evaluate it for offensive language detection in German Twitter data together with other transfer learning strategies.

3 Data and Tasks

3.1 GermEval 2018 Shared Task

Organizers of GermEval 2018 provide training and test datasets for two tasks. *Task 1* is a binary classification for deciding whether or not a German tweet contains offensive language (the respective category labels are 'offense' and 'other'). *Task 2* is a multi-class classification with more fine-grained labels sub-categorizing the same tweets into either 'insult', 'profanity', 'abuse', or 'other'.

The training data contains 5,008 manually labeled tweets sampled from Twitter from selected accounts that are suspected to contain a high share of offensive language. Manual inspection reveals a high share of political tweets among those labeled as offensive. These tweets range from offending single Twitter users, politicians and parties to degradation of whole social groups such as Muslims, migrants or refugees. The test data contains 3,532 tweets. To create a realistic scenario of truly unseen test data, training and test set are sampled from disjoint user accounts. No standard validation set is provided for the task. To optimize hyperparameters of our classification models and allow for early stopping to prevent the neural models from overfitting, we created our own validation set. For this, we used the last 808 examples from the provided training set. The remaining first 4,200 examples were used to train our models.

3.2 Background Knowledge

Since the provided dataset for offensive language detection is rather small, we investigate the potential of transfer learning to increase classification performance. For this, we use the following labeled as well as unlabeled datasets.

One Million Posts: A recently published resource of German language social media data has been published by Schabus et al. (2017). Among other things, the dataset contains 11,773 labeled user comments posted to the Austrian newspaper website 'Der Standard'.[3] Comments have not been annotated for offensive language, but for categories such as *positive/negative sentiment*, *off-topic*, *inappropriate* or *discriminating*.

Twitter: As a second resource, we use a background corpus of German tweets that were collected using the Twitter streaming API from 2011 to 2017. Since the API provides a random fraction of all tweets (1%), language identification is performed using 'langid.py' (Lui and Baldwin, 2012) to filter for German tweets. For all years combined, we obtain about 18 million unlabeled German tweets from the stream, which can be used as a large, in-domain background corpus.

4 Text Classification

In the following section, we describe one linear classification model in combination with specifically engineered features, which we use as a baseline for the classification task. We further introduce a neural network model as a basis for our approach to transfer learning. This model achieves the highest performance for offensive language detection, as compared to our baseline.

4.1 SVM baseline:

Model: The baseline classifier uses a linear Support Vector Machine (Fan et al., 2008), which is suited for a high number of features. We use a text classification framework for German (Ruppert et al., 2017) that has been used successfully for sentiment analysis before.

Features: We induce token features based on the Twitter background corpus. Because tweets are usually very short, they are not an optimal source to obtain good estimates on inverse document frequencies (IDF). To obtain a better feature weighting, we calculate IDF scores based on the Twitter corpus combined with an in-house product review dataset (cf. ibid.). From this combined corpus, we compute the IDF scores and 300-dimensional word

[3] http://derstandard.at

embeddings (Mikolov et al., 2013) for all contained features. Following Ruppert et al. (2017), we use the IDF scores to obtain the highest-weighted terms per category in the training data. Here, we obtain words like *Staatsfunk, Vasall* (state media, vassal) or *deutschlandfeindlichen* (Germany-opposing) for the category 'abuse' and curse words for 'insult'. Further, IDF scores are used to weight the word vectors of all terms in a tweet. Additionally, we employ a polarity lexicon and perform lexical expansion on it to obtain new entries from our in-domain background corpus that are weighted on a 'positive–negative' continuum. Lexical expansion is based on distributional word similarity as described in Kumar et al. (2016).

4.2 BiLSTM-CNN for Text Classification

Model: For transfer learning, we rely on a neural network architecture implemented in the Keras framework for Python.[4] Our model (see Fig. 1) combines a bi-directional LSTM layer (Hochreiter and Schmidhuber, 1997) with 100 units followed by three parallel convolutional layers (CNN), each with a different kernel size $k \in 3,4,5$, and a filter size 200. The outputs of the three CNN blocks are max-pooled globally and concatenated. Finally, features encoded by the CNN blocks are fed into a dense layer with 100 units, followed by the prediction layer. Except for this final layer which uses Softmax activation, we rely on LeakyReLU activation (Maas et al., 2013) for the other model layers. For regularization, dropout is applied to the LSTM layer and to each CNN block after global max-pooling (dropout rate 0.5). For training, we use the Nesterov Adam optimization and categorical cross-entropy loss with a learning rate of 0.002. The intuition behind this architecture is that the recurrent LSTM layer can serve as a feature encoder for general language characteristics from sequences of semantic word embeddings. The convolutional layers on top of this can then encode category related features delivered by the LSTM while the last dense layers finally fine-tune highly category-specific features for the actual classification task.

Features: As input, we feed 300-dimensional word embeddings obtained from fastText (Bojanowski et al., 2017) into our model. Since fastText also makes use of sub-word information (character n-grams), it has the great advantage that it can provide semantic embeddings also for words that have not been seen during training the embedding model. We use a model pre-trained with German language data from Wikipedia and Common Crawl provided by Mikolov et al. (2018). First, we unify all Twitter-typical user mentions ('@username') and URLs into a single string representation and reduce all characters to lower case. Then, we split tweets into tokens at boundaries of changing character classes. As an exception, sequences of emoji characters are split into single character tokens. Finally, for each token, an embedding vector is obtained from the fastText model.

For offensive language detection in Twitter, users addressed in tweets might be an additional relevant signal. We assume it is more likely that politicians or news agencies are addressees of offensive language than, for instance, musicians or athletes. To make use of such information, we obtain a clustering of user ids from our Twitter background corpus. From all tweets in our stream from 2016 or 2017, we extract those tweets that have at least two @-mentions and all of the @-mentions have been seen at least five times in the background corpus. Based

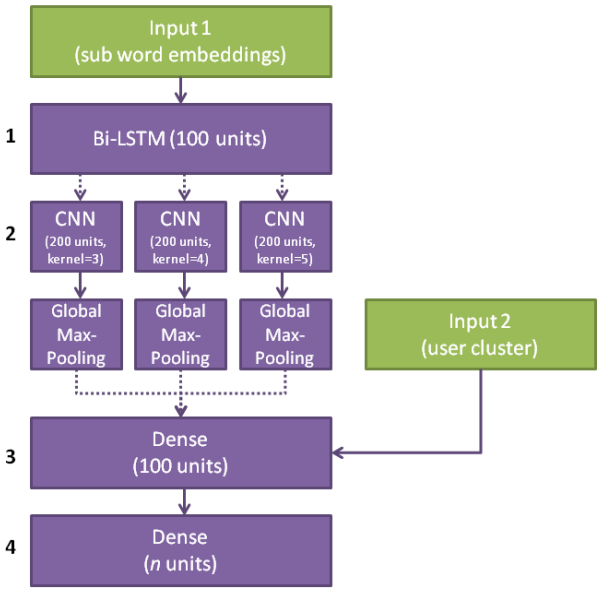


Figure 1: BiLSTM-CNN model architecture. We use a combination of recurrent and convolutional cells for learning. As input, we rely on (sub-)word embeddings. The final architecture also includes clustering information obtained from Twitter user ids. Dotted lines indicate dropout with rate 0.5 between layers. The last dense layer contains n units for prediction of the probability of each of the n classification labels per task.

[4] https://keras.io

Table 1: Examples of Twitter user clusters

Cluster	Accounts
26	breitbartnews, realdonaldtrump, jrcheneyjohn, lindasuhler, barbmuenchen
28	dagibee, lilyachty, youngthug, chrisbrown, richthekid
40	bvb, fcbayern, dfb, young, team
44	spdde, cdu, gruenen, martinschulz, fdp, dielinke
50	tagesschau, spiegelonline, zdf, zeitonline, janboehm

on the resulting 1.8 million lists of about 169,000 distinct user ids, we compute a topic model with $K = 50$ topics using Latent Dirichlet Allocation (Blei et al., 2003). For each of the user ids, we extract the most probable topic from the inferred user id-topic distribution as cluster id. This results in a thematic cluster id for most of the user ids in our background corpus grouping together accounts such as American or German political actors, musicians, media websites or sports clubs (see Table 1). For our final classification approach, cluster ids for users mentioned in tweets are fed as a second input in addition to (sub-)word embeddings to the penultimate dense layer of the neural network model.

5 Transfer Learning

As mentioned earlier, we investigate potential strategies for transfer learning to achieve optimal performance. For this, we compare three different methods to pre-train our model with background data sets. We also compare three different strategies to combat 'catastrophic forgetting' during training on the actual target data.

5.1 Background Knowledge

For a transfer learning setup, we need to specify a task to train the model and prepare the corresponding dataset. We compare the following three methods.

Supervised near-category transfer: As introduced above, the 'One Million Post' corpus provides annotation labels for more than 11,000 user comments. Although there is no directly comparable category capturing 'offensive language' as defined in the shared task, there are two closely related categories. From the resource, we extract all those comments in which a majority of the annotators agree that they contain either 'inappropriate' or 'discriminating' content, or none of the aforementioned. We treat the first two cases as examples of 'offense' and the latter case as examples of 'other'. This results in 3,599 training examples (519 offense, 3080 other) from on the 'One Million Post' corpus. We conduct pre-training of the neural model as a binary classification task (similar to the Task 1 of GermEval 2018)

Weakly-supervised emoji transfer: Following the approach of Felbo et al. (2017), we constructed a weakly-supervised training dataset from our Twitter background corpus. From all tweets posted between 2013 and 2017, we extract those containing at least one emoji character. In the case of several emojis in one tweet, we duplicate the tweet for each unique emoji type. Emojis are then removed from the actual tweets and treated as a label to predict by the neural model. This results in a multi-class classification task to predict the right emoji out of 1,297 different ones. Our training dataset contains 1,904,330 training examples.

Unsupervised topic transfer: As a final method, we create a training data set for transfer learning in a completely unsupervised manner. For this, we compute an LDA clustering with $K = 1,000$ topics[5] on 10 million tweets sampled from 2016 and 2017 from our Twitter background corpus containing at least two meaningful words (i.e. alphanumeric sequences that are not stopwords, URLs or user mentions). Tweets also have been deduplicated before sampling. From the topic-document distribution of the resulting LDA model, we determined the majority topic id for each tweet as a target label for prediction during pre-training our neural model. Pre-training of the neural model was conducted on the 10 million tweets with batch size 128 for 10 epochs.

5.2 Transfer Learning Strategies

Once the neural model has been pre-trained on the above-specified targets and corresponding datasets, we can apply it for learning our actual target task. For this, we need to remove the final prediction layer of the pre-trained model (i.e. Layer 4 in Fig. 1), and add a new dense layer for prediction of one of the actual label sets (two for Task 1, four for Task 2). The training for the actual GermEval

[5]For LDA, we used Mallet (`http://mallet.cs.umass.edu`) with Gibbs Sampling for 1,000 iterations and priors $\alpha = 10/K$ and $\beta = 0.01$.

tasks is conducted with batch size 32 for up to 50 epochs. To prevent the aforementioned effect of forgetting pre-trained knowledge during this task-specific model training, we evaluate three different strategies.

Gradual unfreezing (GU): In Howard and Ruder (2018), gradual unfreezing of pre-trained model weights is proposed as one strategy to mitigate forgetting. The basic idea is to initially freeze all pre-trained weights of the neural model and keep only the newly added last layer trainable (i.e. Layer 4 in Fig. 1). After training that last layer for one epoch on the GermEval training data, the next lower frozen layer is unfrozen and training will be repeated for another epoch. This will be iterated until all layers (4 to 1) are unfrozen.

Single bottom-up unfreezing (BU): Following the approach of Felbo et al. (2017), we do not iteratively unfreeze all layers of the model, but only one at a time. First, the newly added final prediction layer is trained while all other model weights remain frozen. Training is conducted for up to 50 epochs. The best performing model during these epochs with respect to our validation set is then used in the next step of fine-tuning the pre-trained model layers. For the bottom-up strategy, we unfreeze the lowest layer (1) containing the most general knowledge first, then we continue optimization with the more specific layers (2 and 3) one after the other. During fine-tuning of each single layer, all other layers remain frozen and training is performed for 50 epochs selecting the best performing model at the end of each layer optimization. In a final round of fine-tuning, all layers are unfrozen.

Single top-down unfreezing (TU): This proceeding is similar the one described above, but inverts the order of unfreezing single layers from top to bottom sequentially fine-tuning layers 4, 3, 2, 1 individually, and all together in a final round.

Baseline (Pre-train only): All strategies are compared to the baseline of no freezing of model weights, but training all layers at once directly after pre-training with one of the three transfer datasets.

6 Evaluation

Since there is no prior state-of-the-art for the GermEval Shared Task 2018 dataset, we evaluate the performance of our neural model compared to the baseline SVM architecture. We further compare the different tasks and strategies for transfer learning introduced above and provide some first insights on error analysis.

Table 2: Transfer learning performance (Task 1)

Transfer	Strategy	F1	Accuracy
None	-	0.709	0.795
Category	Pre-train only	**0.712**	**0.809**
	GU	0.702	0.796
	BU	0.709	0.802
	TU	0.711	0.799
Emoji	Pre-train only	0.720	0.811
	GU	0.708	0.807
	BU	**0.739**	**0.817**
	TU	0.725	0.814
Topic	Pre-train only	0.733	0.817
	GU	0.712	0.801
	BU	**0.753**	**0.828**
	TU	0.732	0.817

Transfer learning: First, we evaluate the performance of different transfer learning datasets and strategies. Tables 2 and 3 show that we achieve best performances for both tasks on our validation set by pre-training our neural model on the large Twitter datasets.[6] The two approaches, emoji and topic transfer, substantially improve the classification performance compared to not using transfer learning at all ('None'). In contrast, pre-training on the annotated dataset from the 'One Million Posts' corpus does only lead to minor improvements. Comparing the three different strategies to reduce negative effects of forgetting in transfer learning, the strategy of unfreezing single layers during training from the lowest layers to the top of the model architecture (BU) performs best, especially in conjunction with the pre-training on the large Twitter datasets. For these setups, the model can take full advantage of learning language regularities from generic to more task-specific features in its different layers. The other strategies (GU, TU) do not perform better than pre-training the

[6]For the binary classification Task 1, we report precision (P), recall (R), and F1 for the targeted positive class 'offense'. During training, we also optimized for binary F1. For the multi-class classification Task 2, we report macro-F1 (average of precision, recall, and F1 of all individual four categories). During training, we also optimized for macro-F1. All reported results are average values obtained from 10 repeated runs of model training.

Table 3: Transfer learning performance (Task 2)

Transfer	Strategy	F1	Accuracy
None	-	0.578	0.747
Category	Pre-train only	0.578	0.755
	GU	0.560	0.751
	BU	0.580	0.750
	TU	**0.581**	**0.759**
Emoji	Pre-train only	0.572	0.756
	GU	0.564	0.756
	BU	0.577	**0.764**
	TU	**0.592**	0.757
Topic	Pre-train only	0.597	0.762
	GU	0.590	0.755
	BU	**0.607**	**0.764**
	TU	0.582	0.764

neural model and then immediately training the entire network on the actual task ('Pre-train only').

Final results: Tables 4 and 5 show the final results for the two offensive language detection tasks on the official test set. We compare the baseline SVM model with the BiLSTM-CNN neural model with the best performing transfer learning setup (BU). Additionally, we show the results when adding cluster information from users addressed in tweets (cf. Section 4). Due to the fact that training and validation data were sampled from a different user account population than the test dataset (cf. Section 3), evaluation scores on the official test data are drastically lower than scores achieved on our validation set during model selection.

Compared to the already highly tweaked SVM baseline, our BiLSTM-CNN model architecture with topic transfer delivers comparable results for identifying offensive language in Task 1 and significantly improved results for Task 2. The SVM achieves a high precision but fails to identify many offensive tweets, which especially in Task 2 negatively affects the recall.

In contrast, topic transfer leads to a significant improvement, especially for Task 2. Performance gains mainly stem from increased recall due to the background knowledge incorporated into the model. We assume that not only language regularities are learned through pre-training but that also some aspects relevant for offensive language already are grouped together by the LDA clusters used for pre-training.

As a second task-specific extension of our text classification, we feed cluster information for users addressed in tweets into the process. Here the results are mixed. While this information did not lead to major performance increases on our validation set (not shown), the improvements for the official test set are quite significant. For Task 1, the performance score increases several percentage points up to 75.2% F1 (Accuracy 77.5%). For Task 2, increases are still quite remarkable, although the absolute performance of this multi-class problem with 52.7% F1 (Accuracy 73.7%) is rather moderate. From these results, we infer that thematic user clusters apparently contribute a lot of information to generalize an offensive language detection model to unseen test data.

Error analysis: Accuracy values for German offensive language detection around 75% signal some room for improvement in future work. What are the hard cases for classifying offensive language? We look at false positive (FP) and false negatives (FN) for Task 1. In our validation set, the ratio of FP and FN is about 60:40, which means our classifier slightly more often assumes offensive language than there is actually in the data compared to cases in which it misses to recognize offensive tweets. Looking more qualitatively into FP examples, we can see a lot of cases which actually express a very critical opinion and/or use harsh language, but are not unequivocal insults. Another group of FP tweets does not express insults directly but formulates offensive content as a question. In other cases, it is really dependent on context whether a tweet addressing a specific group uses that group signifier actually with a derogatory intention (e.g. calling people *'Jew'*, *'Muslim'*, or *'Communist'*). For FN tweets, we can identify insults that are rather subtle. They do not use derogatory vocabulary but express loathing by dehumanizing syntax (e.g. *'das was uns regiert'* where the definite gender-neutral article 'das' refers to the German chancellor), metaphor (*'Der ist nicht die hellste Kerze'*, i.e. 'he is not the brightest light') or insinuating an incestuous relationship of some persons parents (*'Hier drängt sich der Verdacht auf, das die Eltern der beiden Geschwister waren'*). Another repeatedly occurring FN case are tweets expressing suspicion against the government, democratic institutions, the media or elections. While those tweets certainly in most cases origin from a radi-

Table 4: Offensive language detection performance % (Task 1)

Model	RunID	Offense			Other			Average (official rank score)			
		P	R	F1	P	R	F1	P	R	F1	Acc.
Baseline SVM	coarse_1	**71.52**	46.17	56.12	76.52	**90.52**	**82.93**	74.02	68.34	71.07	75.42
BiLSTM-CNN											
+ Topic transfer	coarse_2	66.30	49.75	56.84	77.03	86.95	81.69	71.67	68.35	69.97	74.29
+ User-cluster	coarse_3	66.29	**68.89**	**67.56**	**83.62**	81.93	82.77	**74.96**	**75.41**	**75.18**	**77.49**

Table 5: Offensive language detection performance % (Task 2)

Model	RunID	Abuse	Insult	Other	Profanity	Average (official rank score)			
		F	F	F	F	P	R	F	Acc.
Baseline SVM	fine_1	46.10	21.12	82.88	3.92	50.92	37.27	43.04	70.44
BiLSTM-CNN									
+ Topic transfer	fine_2	51.96	**40.18**	84.26	15.58	51.06	46.07	48.44	72.79
+ User cluster	fine_3	**53.25**	39.46	**84.85**	**29.63**	**56.85**	**49.13**	**52.71**	**73.67**

cal right-wing worldview and can be considered as abusive against democratic values, their language is not necessarily offensive per se. This more qualitative look into the data opens up some directions to improve offensive language detection incorporating technologies that are able to capture such more subtle insults as well as handling cases of questions and harsh but still not insulting critique.

7 Conclusion

In this paper, we presented our neural network text classification approach for offensive language detection on the GermEval 2018 Shared Task dataset. We used a combination of BiLSTM and CNN architectures for learning. As task-specific adaptations of standard text classification, we evaluated different datasets and strategies for transfer learning, as well as additional features obtained from users addressed in tweets. The coarse-grained offensive language detection could be realized to a much better extent than the fine-grained task of separating four different categories of insults (accuracy 77.5% vs. 73.7%). From our experiments, four main messages can be drawn:

1. Transfer learning of neural networks architectures can improve offensive language detection drastically.

2. Transfer learning should be conducted on as much data as possible regarding availability and computational resources. We obtained best results in a completely unsupervised and task-agnostic pre-training setup on in-domain data. During pre-training, we predicted the primary topics of tweets obtained by an LDA process, which previously clustered our background dataset of 10 million tweets into 1,000 topics.

3. To mitigate the effect of 'catastrophic forgetting' in transfer learning, it is advised to train and optimize the different layers of the neural network model separately. In our experiments on models pre-trained on large Twitter datasets, the bottom-up approach of training from the lowest to the top layer performed significantly better than all other tested strategies to freeze model weights during learning.

4. User mentions in tweets can contribute a lot of information to the classifier since some accounts are much more likely to be targeted by offensive language than others. Clustering users thematically allows including information from users not seen during training.

The fact that our unsupervised, task-agnostic pre-training by LDA topic transfer performed best suggests that this approach will also contribute beneficially to other text classification tasks such as sentiment analysis. Thus, in future work, we plan to evaluate our approach with regard to such other tasks. We also plan to evaluate more task-agnostic approaches for transfer learning, for instance employing language modeling as a pre-training task.

Acknowledgements: The paper was supported by BWFG Hamburg within the "Forum 4.0" project as part of the *ahoi.digital* funding line, and by DAAD via a WISE stipend.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

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.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark. ACL.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. ACL.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, Doha, Qatar. ACL.

Philip Kreißel, Julia Ebner, Alexander Urban, and Jakob Guhl. 2018. *Hass auf Knopfdruck: Rechtsextreme Trollfabriken und das Ökosystem koordinierter Hasskampagnen im Netz*. Institute for Strategic Dialogue, London, UK.

Ayush Kumar, Sarah Kohail, Amit Kumar, Asif Ekbal, and Chris Biemann. 2016. IIT-TUDA at SemEval-2016 Task 5: Beyond sentiment lexicon: Combining domain dependency and distributional semantics features for aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 1129–1135, San Diego, CA, USA. ACL.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Demo Session*, pages 25–30, Jeju, Korea. ACL.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*. Atlanta, GA, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop at International Conference on Learning Representations (ICLR)*, pages 1310–1318, Scottsdale, AZ, USA.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the 11th International Conference on Language Resources and Evaluation*, Miyazaki, Japan. ELRA.

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, Montreal, Canada. International World Wide Web Conferences Steering Committee.

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 3rd Workshop on Natural Language Processing for Computer-Mediated Communication*, pages 6–9, Bochum, Germany.

Eugen Ruppert, Abhishek Kumar, and Chris Biemann. 2017. LT-ABSA: An extensible open-source system for document-level and aspect-based sentiment analysis. In *Proceedings of the GSCL GermEval Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 55–60, Berlin, Germany.

Dietmar Schabus, Marcin Skowron, and Martin Trapp. 2017. One million posts: A data set of German online discussions. In *Proceedings of the 40th International Conference on Research and Development in Information Retrieval*, pages 1241–1244, Tokyo, Japan.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the 5th International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. ACL.

Ellen Spertus. 1997. Smokey: Automatic recognition of hostile messages. In *Proceedings of the 14th National Conference on Artificial Intelligence and*

Ninth Conference on Innovative Applications of Artificial Intelligence, pages 1058–1065, Providence, RI, USA. AAAI Press.

Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1980–1984, New York, NY, USA. ACM.

Towards the Automatic Classification of Offensive Language and Related Phenomena in German Tweets

Julian Moreno Schneider, Roland Roller, Peter Bourgonje, Stefanie Hegele, Georg Rehm
DFKI GmbH, Language Technology Lab, Alt-Moabit 91c, 10559 Berlin, Germany
Corresponding author: julian.moreno_schneider@dfki.de

Abstract

In recent years the automatic detection of abusive language, offensive language and hate speech in several different forms of online communication has received a lot of attention by the Computational Linguistics and Language Technology community. While most approaches work on English data, publications on languages other than English are rare. This paper, submitted to the GermEval 2018 Shared Task on the Identification of Offensive Language, provides the results of several experiments regarding the classification of offensive language in German language tweets.

1 Introduction

In recent years the automatic detection of abusive language, offensive language and general hate speech comments in several different forms of online communication (e. g., Twitter, Facebook, and other forms of social media or, more generally, user-generated content) has received a lot of attention by the Computational Linguistics and Language Technology community. One of the underlying assumptions of nearly all approaches published so far is the idea of setting up a watchdog service that is able to detect instances of offensive language, abusive language, hate speech, or cyberbullying, among others, fully automatically – and with high classification precision – in order to prevent the specific message or content from being posted or to flag the respective piece of content to human experts monitoring the respective system so that they can initiate corrective actions.

While most approaches towards the automatic detection of offensive online language work on and with English data sets, publications on languages other than English are rare. This article, submitted to the GermEval 2018 Shared Task on the Identification of Offensive Language, provides the results of several experiments regarding the classification of offensive language in German language tweets.

The remainder of this article is structured as follows. First, Section 2 provides an overview of related work, while Section 3 briefly describes the data set used in the GermEval 2018 Shared Task on the Identification of Offensive Language as well as the two classification tasks and their respective categories. Section 4 characterises the experiments we carried out including features and classifiers used. Section 5 briefly sketches the results of the experiments, while Section 6 lists the six runs submitted to the Shared Task. Section 7 discusses our results and Section 8 concludes the article.

2 Related Work

Recent years have seen an increasing amount of attention from the NLP community to hateful conduct and aggression online. While at first glance separating constructive, useful content from, for example, hate speech might seem like a typical text classification problem, comparable to spam classification and sentiment analysis where typical text classification approaches may be well applicable, the question whether or not certain utterances are still acceptable within the boundaries of free speech puts this task in the intersection of several research areas and disciplines, including linguistics, sociology (Jones et al., 2013; Phillips, 2015), psychology (Kowalski and Limber, 2013; Dreißing et al., 2014), law (Marwick and Miller, 2014; Banks, 2010; Massaro, 1991) and also common sense. An overview of current NLP-based approaches is collected and presented in Schmidt et al. (2017).

The complexity of the task results in a variety of difficulties that have yet to be solved. What should be considered as offensive, racist, sexist or profane, and the extra-linguistic nature of the issue are complicating factors. The nature of an utterance often depends on factors like context, (ethnicity of the) author, (ethnicity of the) targeted person or group,

whether or not irony is the case, etc. (Nand et al., 2016; Waseem et al., 2016; Warner et al., 2012). All of this makes the creation and annotation of corpora a challenging task. Currently there is no large, universally used data set available. Numerous data sets have been created for specific tasks differing in size (from a couple of hundred labelled tweets to hundred thousands of labelled discussions) as well as text genres, e.g., Twitter (Burnap et al., 2015; Waseem, 2016; Waseem et al., 2016; Davidson, 2017), Yahoo! (Djuric et al., 2015; Nobata et al., 2016) and Wikipedia (Wulczyn et al., 2017).

Most related work on detecting abusive language has been done for English, focusing on the data set by Waseem (2016) annotated for the three categories "Sexism", "Racism" and "Other". Many approaches rely on supervised learning with Support Vector Machines as the most frequently used classifier (Davidson, 2017; Bourgonje et al., 2017). Recent approaches employing deep learning architectures have shown to compete with or even outperform these approaches. For the task on distinguishing the three categories named above the best result (F-score of 0.93) was reached by Badjatiya et al. (2017) using an LSTM model with features extracted by character n-grams, and assisted by Gradient Boosted Decision Trees. Park et al. (2017) implemented three CNN-based models for classification. Pitsilis et al. (2018) suggested a detection scheme consisting of Recurrent Neural Network (RNN) classifiers.

3 Data Set and Tasks

The GermEval 2018 task focuses on the linguistic analysis of offensive content in German tweets, 5009 of which were provided as training data.[1] A detailed description of the annotation process along with the annotation guidelines was also made available. There are two different tasks with the provided training data annotated as follows. Task 1 is a binary classification task deciding whether a tweet is offensive or not (labels OFFENSIVE: 1688, OTHER: 3321). Task 2 is a fine-grained classification task distinguishing four subcategories (labels PROFANITY: 71, INSULT: 595, ABUSE: 1022 and OTHER: 3321). The data set consists of tweets only without any kind of meta information such as the tweet ID etc. The average token size per tweet is 21,9 and consists of 1,6 sentences.

Related tasks for English such as the Workshop on Abusive Language Online (ALW)[2] have chosen different sets of data labels ranging from binary classification (e. g., PERSONAL ATTACK vs. NONE in a Wikipedia corpus (Wulczyn et al., 2017) to more granular tag sets (e. g., RACISM, SEXISM and NONE, applied to Twitter data (Waseem, 2016)). Transparent annotation guidelines are not always made publicly available, making attempts of leveraging knowledge from related data sets a formidable challenge (see the experiments on crosslingual embeddings in Section 7).

4 Experiments

We follow the majority of earlier work in this field, as described in Section 2, that employs neural networks to implement classifiers to tackle the challenge. The data and individual messages in the GermEval 2018 Shared Task is challenging due to their short length (i. e., tweets) and due to the annotated categories that are, conceptually, relatively close to one another. As reflected by rather low inter-annotator agreement scores reported for similar annotations on comparable data sets, when intellectually exploring training data, even for humans it is challenging to reliably and consistently assign labels to tweets or, on a more abstract level, to agree what constitutes "abusive" or "offensive language". In an attempt to find the best way of solving this task using a neural network approach, we not only experimented with different network architectures, but also made an effort to obtain and include additional training data as well as to enrich the given tweets with additional meta information.

4.1 Data Enrichment

Below we present the various techniques to enrich tweets by additional information as well as an automatic generation of further training data.

Gender Information Extra-linguistic information about tweets can be decisive when making a final call on whether or not some piece of content should be considered insulting, profane, abusive or non-offensive. Retrieving identity information of the author would be valuable information to classify content more reliably. Since getting this type of metadata in the form of the user ID is typically not feasible for such data sets, we attempted to classify for one aspect of user identity, i. e., the gender

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

[2] https://sites.google.com/site/abusivelanguageworkshop2017

of the author. We experimented with augmenting the GermEval tweets with gender information to establish whether or not this feature would be helpful in classification. To obtain gender labels for the tweets, we scraped the tweets annotated for the TwiSty corpus (Verhoeven et al., 2016) and classified this using FastText[3] (Joulin et al., 2016), achieving an accuracy of 79.77 for this binary classification task. The GermEval tweets were then labeled using this classifier. The results using this as an additional feature in the classification of the test set are included in Tables 1 and 2.

User Profile Information As another piece of extra-linguistic information, user profile information of Twitter users mentioned in tweets were retrieved. For example, for the tweet [*@StephanJBauer @soskinderdorf Auch in Deutschland hungern Kinder.*], we retrieved the profile descriptions for *@StephanJBauer* and *@soskinderdorf* and added this to the representation of the tweet. The rationale behind this is that certain users with a particular (potentially controversial) political profile and high visibility could be more likely to trigger offensive tweets (i. e., we attempt to model the identity of the target audience, and not that of the author). The results for this setup are included in Tables 1 and 2.

Sentiment Another linguistic feature that we have included is sentiment analysis. This processing step was carried out using a simple dictionary lookup using the data set published by Waltinger (2010). According to the largest number of positive/negative sentiment words found in the tweet, we assigned the labels POSITIVE, NEGATIVE, NEUTRAL and POS_NEG in case the tweet has as many positive as negative sentiment words.

Additional User Friend Data Lastly a set of automatically labelled tweets for Task 1 is generated in order to increase the size of the data set to train the classifier. For this purpose, a small subset of the training data (70 tweets) has been selected to identify the original source (user) of the tweet. From this subset 25 different users have been identified. Most users occurred various times and in various cases it turned out that a user who posted an OFFENSE tweet might have also posted OTHER tweets. However, users who posted an OFFENSE tweet at least once were assigned to the OFFENSE user group and all others to the OTHER group. Using this list of users a set of approximately $4,000$ tweets could be automatically labelled. Thus, a tweet from a person of the OFFENSE group was automatically labelled as OFFENSE and a tweet from a person of the OTHER group as OTHER. In order to further increase the data size, the user list has been extended by taking all twitter friends into account, assigning each person of the friend list to the same user group. In this way a list of $25,000$ users has been created, resulting in 2 million automatically labelled tweets. In order to stick to a practical and feasible setup, i. e., to be able to run the experiments on standard hardware, automatically labelled data was reduced to $50,000$ tweets using the same ratio of OTHER/OFFENSE as in the manually labelled training data. This set of tweets is not added to each tweet as a feature, but used as a new training corpus, i. e., the neural network is first trained with the new corpus of automatically obtained tweets and then the training is refined with the training set of GermEval 2018.

4.2 Architecture

To set a baseline performance we use FastText, which allows for both supervised and unsupervised text classification combining word embeddings with character n-grams instead of CBOW (which is the case for Word2vec). We apply out-of-the-box supervised classification using Wikipedia embeddings to obtain our baseline score. In addition to that we generate embeddings from a German Twitter snapshot described by Scheffler (2014). Due to its higher flexibility we use Keras[4] for all other experiments reported on in this paper.

The neural network that we implemented and tested is based on the architecture by Wang et al. (2017). Their architecture is composed of three layers: (i) a convolutional layer; (ii) a MaxPooling layer; and (iii) a dense layer, that performs the classification itself. We made minor modifications to this setup and instead of using one convolutional layer and one dense layer, we use two convolutional and two dense layers. Due to the relatively large number of dimensions (300), any relevant information in the input data would be better preserved with two convolutional layers. The second dense layer is there to accommodate the more detailed classification for Task 2, which not only comprises more classes but also classes that are conceptually

[3] https://fasttext.cc

[4] https://keras.io

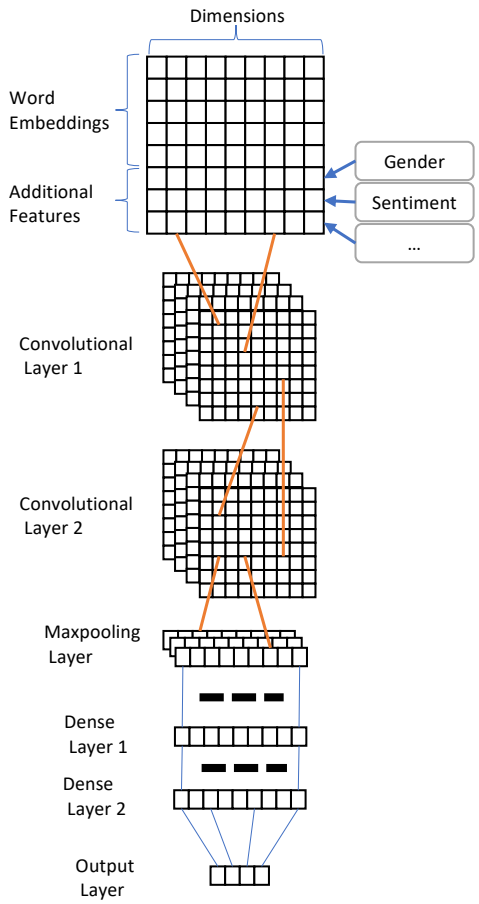


Figure 1: Architecture of the CNN implemented for the GermEval 2018 Shared Task.

closer to one another (see Figure 1).

As illustrated in the architectural overview, the additional features we experimented with are added to the training data in the pre-processing steps, the exact shape or form depending on the individual feature (i. e., binary values for gender or sentiment, embeddings for user descriptions, etc., see Section 4.1 for more details).

5 Results

The results presented below were obtained using cross-validation[5] on the training portion of the data set provided by the organisers of GermEval 2018. We compute the average accuracy for the binary classification (OFFENSE vs. OTHER) for Task 1 (Table 1) and provide accuracy, precision, recall and f1-score for the individual classes (INSULT, ABUSE, PROFANITY and OTHER) in Task 2 (Table 2). Based on the cross-validation over the training data, we consider the Twitter embeddings in combination with user descriptions to be the best setup, with an accuracy of 81 for Task 1 and 72.2 for Task 2. However, because this approach is dependent on the existence of user mentions in the tweet text, which may be proportionally less present in the test set, the figures on the test data may well deviate and show another setup to be the best performing one.[6]

6 Runs

We have submitted six runs (three for each task):

1. **dfkilt_coarse_1.txt:** TE+Desc approach including twitter embeddings and user mentions description (Task 1).

2. **dfkilt_coarse_2.txt:** TE+Sent approach including twitter embeddings and sentiment analysis information (Task 1).

3. **dfkilt_coarse_3.txt:** TE+G+D approach including twitter embeddings, gender classification and mentions descriptions (Task 1).

4. **dfkilt_fine_1.txt:** TE+Desc approach including twitter embeddings and mentions description (Task 2).

5. **dfkilt_fine_2.txt:** TE+S+G+D approach including twitter embeddings, sentiment analysis, gender classification and mentions description (Task 2).

6. **dfkilt_fine_3.txt:** TE+S+D approach including twitter embeddings, sentiment analysis and mentions description (Task 2).

7 Discussion

When dealing with the task of detecting hateful, aggressive, racist and/or sexist behaviour online, a lack of high inter-annotator agreement can be an issue and shows the high complexity of the challenge – even for humans. Ross et al. (2016) for instance introduce a German corpus of hate speech on the European refugee situation and report very low inter-annotator agreement scores (Krippendorff's α between 0.18 and 0.29). Waseem (2016) investigates inter-annotator agreement when comparing amateur annotations (generated using CrowdFlower)

[5]Due to time constraints we performed cross-validation with one single fold only.

[6]Note that we generated additional training data through user friends for the classes OFFENSE and OTHER only and, hence, did not use the data in Task 2.

Table 1: Results for Task 1 using different features

	Acc	OFFENSE			OTHER		
		P	R	F1	P	R	F1
Fasttext (FT)	73.9	–					
Wikipedia Embeddings (WE)	71.8	69.4	36.9	48.2	72.4	91.1	80.7
Twitter Embeddings (TE)	72.7	62.4	58.1	60.2	77.8	80.8	79.2
TE + Sentiment	78.5	**80**	52.5	63.4	78	**92.8**	84.8
TE + Descriptions	**81**	79	62.3	**69.6**	**81.8**	91.1	**86.2**
TE + Gender Classification	76.3	66.7	**66.3**	66.5	81.5	81.8	81.6
TE + Sentiment + Gender	75.4	66.2	62.5	64.3	80	82.5	81.2
TE + Sentiment + Descriptions	76.1	67.3	63.1	65.2	80.4	83.2	81.8
TE + Gender + Descriptions	76.9	72.2	56.9	63.6	78.8	88	83.1
TE + Sentiment + Gender + Descriptions	75.6	67.4	60.6	63.8	79.5	83.8	81.6
TE + User Friends Information	77.2	74.4	53.1	62	78.2	90.2	83.8

Table 2: Results for Task 2 using different features. (FT: Fasttext, WE: Wikipedia Embeddings, TE: Twitter embeddings, S: Sentiment, G: Gender Classification, D: Descriptions, UFI: User friends information)

	Acc	INSULT			ABUSE			PROFANITY			OTHER		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
FT	68.3	–											
WE	67.6	**39.5**	26.8	31.9	49.1	52	50.5	0	0	0	77.7	81.4	79.5
TE	65.2	27.3	5.4	9	41.3	57.8	48.2	0	0	0	78.1	79.7	79
TE+S	69	33.8	46.4	39.1	54.5	47.1	50.5	0	0	0	82.9	81.4	82.1
TE+D	**72.2**	38.9	**55.2**	**45.7**	62.7	42.4	50.6	0	0	0	**83.5**	86.8	**85.1**
TE+G	69.6	37.8	25	30.1	51.5	51	51.2	0	0	0	79.2	85.2	82.1
TE+S+G	70.1	28.6	28.6	28.6	65.7	45.1	53.5	0	0	0	78.2	87.3	82.5
TE+S+D	71.4	33.3	1.8	3.4	51.1	**64.7**	**57.1**	0	0	0	79.9	87.6	83.6
TE+G+D	69.2	30.9	44.6	36.5	63.5	32.4	42.9	0	0	0	79.9	87.3	83.4
TE+S+G+D	71.8	38.5	17.9	24.4	**70.6**	35.3	47.1	0	0	0	74.3	**95.5**	83.6
TE+UFI	—												

and expert annotations and reports a similarly low Cohen's Kappa of 0.14. Van Hee et al. (2015) work on classification of cyberbullying using a Dutch corpus and report Kappa scores between 0.19 and 0.69. Kwok and Wang (2013) report an overall inter-annotator agreement of only 33% when investigating racist tweets. Nobata et al. (2016) report a relatively high agreement for binary classification of *clean* vs. *abusive* for social media comments on Yahoo! (Kappa = 0.843), but this number drops significantly when different subcategories for the abusive comments are introduced (such as *hate*, *derogatory language* and *profanity*, with Kappa decreasing to 0.456).

Using the basic setup of our network with Twitter embeddings does not improve over the FastText baseline (with accuracies of 72.2 vs. 73.9 for Task 1 and 65.2 vs. 68.3 for Task 2, respectively). However, adding additional types of information (or combinations), we do improve over this baseline, by 7.1 points in accuracy for Task 1 and 3.9 points in Task 2 in the best scoring setup.

In addition to different opinions on what constitutes and does not constitute "offensive language" (in terms of inter-annotator agreement), also the usage of automatically labelled data has its limitations. While 'distantly labelled' data might have a beneficial effect if manually labelled data is small, it might lose its effect with increasing gold standard data. The quality of automatically labelled data also plays an important role. As mentioned before, even Twitter users who post large numbers of offensive tweets do not do so exclusively. In various cases people might show a radical opinion without being explicitly offensive, and sometimes people also just talk about daily life using standard, acceptable language. Yet other times, they may use highly offensive language when complaining about the weather. The same rather high variance can be observed for people belonging to the OTHER user group. This means the data contains a large number of false positives and false negatives. A method which is able to deal with noisy data more robustly might have been more suitable.

Adding explicit sentiment information did improve over the setup using only the Twitter embeddings. Intuitively, a negative sentiment can be expected to align with the OFFENSE class for Task 1, and perhaps be less informative for Task 2. This is in any case reflected by the scores, as there is an almost 6 point increase in accuracy for Task 1, but a smaller increase for Task 2 (almost 4 points). However, a closer analysis shows, that many tweets might contain negative sentiment words without being offensive, such as 'arbeitslos' ('unemployed') or 'Flüchtling' ('refugee').

As for the added gender information, doing a factorized analysis of the different classes (in Task 1 and Task 2) and gender distribution, we did not see a clear hint that either male or female authors behave more offensive, profane, abusive or insulting. Yet, this feature improved performance for both tasks. While perhaps a clear correlation could not be established, it is possible that by including gender information, we are implicitly encoding certain features of tweets that help the network in differentiating between the classes.

Adding the descriptions that users publish about themselves (on their Twitter profile pages) increased the most when cross-validating the training set, compared to the setup using only embeddings. As explained in Section 4.1, the idea behind this feature is that certain users could be more likely to trigger hateful language. This would be captured by the classifier without the description as well (i. e., the user name showing up in the user mention would be an important feature). However, since user names are not likely to be present in the embeddings (hence, they will not have an informative representation using only the embeddings setup), adding the description the users added themselves, consisting of individual words which are more likely to be represented in the embeddings, will add information. For Task 1, this additional information improves just over 8 points to the embeddings-only setup, and for Task 2 the improvement is 7 points in accuracy.

Apart from the presented approaches, we made the first steps towards exploiting available resources in other languages to have at our disposal more training data for the neuronal networks. Given that the task is concerned with German tweets with limited amounts of German data available, we experimented with a crosslingual approach, i. e., expanding on the German language data by adding English language data. For the first attempt, we used the NLP+CSS_2017 data set (Jha et al., 2017).[7] The original data set (containing 10,095 unique tweets) was annotated for detecting benevolent sexism (labels used: BENEVOLENT,

[7] https://github.com/AkshitaJha/NLP_CSS_2017

HOSTILE, OTHER). Matching the definition of abusive language according to GermEval's annotation guidelines all instances of sexism found in the cleaned corpus (only tweets with a clear inner-annotator agreement were kept) were tagged as ABUSE and all remaining tweets were simply classified as OTHER.

In order to use data sets in different languages, we mapped the word embeddings of both data sets (one in English, another in German) onto each other, both generated from Wikipedia data, using MUSE.[8] Under the assumption that the specific characteristics (word embeddings) use the same vector space, the neural network should not explicitly register the difference between English and German training data, and should, hence, produce better results. This crosslingual approach produces 71.5% average accuracy in Task 1 and 67.9% average accuracy in Task 2. These preliminary results demonstrate that the accuracy numbers have not increased compared to the other approaches. We will investigate the crosslingual approach in more detail in follow-up work.

8 Conclusions and Future Work

We have developed a CNN-based approach on German Twitter data to predict offensiveness. The data is annotated at two levels; one coarse level indicating whether or not the tweet is offensive (Task 1), and one detailed level indicating whether offensive tweets are insulting, profane or abusive (Task 2). We augment the available training data with several different types of information and in the best scoring setup achieve an accuracy increase of 7.1 points for Task 1 and 3.9 points for Task 2, comparing to a baseline implementation using FastText. This sets the marks of our best attempt at an accuracy of 81 for Task 1 and 72.2 for Task 2.

Various previous studies and also our own experiments demonstrate that the automatic classification of offensive language, including closely related linguistic categories, with a very high degree of accuracy is a very challenging task. The low inter annotator agreement often mentioned above is, obviously, due to the highly subjective nature of language perception and interpretation. For some people certain expressions constitute "offensive language", for others they do not. It is challenging, maybe even impossible, to break this down into a binary classification task or into a task with a small number of categories. This socio-technical challenge notwithstanding, it is surely worthwhile to continue this line of research to arrive at larger data sets, better and more adequate categories and more suitable evaluation procedures. It would also be interesting to investigate the different ways an automatic text classification procedure could help and assist social media users flagging and responding to, but also composing messages. After all, maybe many instances of offensive language could be taken care of by making sure that they never come into existence. For example, Twitter users who are writing a tweet or a reply to a certain user and who use, based on an automatic classifier, offensive language, could be shown an alert window before posting, reminding them that they are probably using offensive language and that there is an actual human being on the other end of the line who may take offense by language of this nature.

[8] https://github.com/facebookresearch/MUSE

Acknowledgments

This work has been partially funded by the project LYNX. The project LYNX has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 780602. More information is available online at http://www.lynx-project.eu.

References

Banks, James. 2010. Regulating hate speech online. *International Review of Law, Computers & Technology, 24(3)*

Badjatiya, Pinkesh and Gupta, Shashank and Gupta, Manish and Varma, Vasudeva. 2017. Deep learning for hate speech detection in tweets *Proceedings of the 26th International Conference on World Wide Web Companion, 759–760*

Bourgonje, Peter and Moreno-Schneider, Julian and Srivastava, Ankit and Rehm, Georg 2017. Automatic classification of abusive language and personal attacks in various forms of online communication *International Conference of the German Society for Computational Linguistics and Language Technology, 180–191* Springer.

Burnap, Pete and Williams, Matthew L. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet, 223–242* Wiley Online Library.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Davidson, Thomas and Warmsley, Dana and Macy, Michael and Weber, Ingmar. 2017. Automated hate speech detection and the problem of offensive language *arXiv preprint arXiv:1703.04009*

Harald Dreißing and Josef Bailer and Anne Anders and Henriette Wagner and Christine Gallas. 2014. Cyberstalking in a large sample of social network users: prevalence, characteristics, and impact upon victims. *Cyberpsychology, Behaviour, and Social Networking, 17(2)*

Djuric, Nemanja and Zhou, Jing and Morris, Robin and Grbovic, Mihajlo and Radosavljevic, Vladan and Bhamidipati, Narayan. 2015. Hate speech detection with comment embeddings *Proceedings of the 24th international conference on world wide web, 29–30*

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Van Hee, Cynthia and Lefever, Els and Verhoeven, Ben and Mennes, Julie and Desmet, Bart and De Pauw, Guy and Daelemans, Walter and Hoste, Veronique. 2015. Detection and Fine-Grained Classification of Cyberbullying Events. *Proceedings of the International Conference Recent Advances in Natural Language Processing, 672-680*

Jha, Akshita and Mamidi, Radhika 2017. When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data. *Proceedings of the Second Workshop on NLP and Computational Social Science, 7-16*

Jones, Lisa M and Mitchell, Kimberly J and Finkelhor, David. 2013. Online harassment in context: Trends from three youth internet safety surveys (200, 2005, 2010). *Psychology of violence, 3(1):53* Educational Publishing Foundation.

Joulin, Armand and Grave, Edouard and Bojanowski, Piotr and Douze, Matthijs and Jgou, Hrve and Mikolov, Tomas. 2016. FastText.zip: Compressing text classification models.

Robin M. Kowalski and Susan P. Limber. 2013. Psychological, physical, and academic correlates of cyberbullying and traditional bullying. *Journal of Adolescent Health, 53(1)*

Kwok, Irene and Wang, Yuzhou. 2013. Locate the Hate: Detecting Tweets Against Blacks. *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, 1621-1622*

Alice E. Marwick and Ross W. Miller. 2014. Online Harassment, Defamation, and Hateful Speech: A Primer of the Legal Landscape. *Fordham Center on Law and Information Policy Report*

Massaro, Toni M. 1991. Equality and Freedom of Expression: The Hate Speech Dilemma. *William & Mary Law Review, 32(211)*

Nand, Parma and Perera, Rivindu and Kasture, Abhijeet. 2016. "How Bullying is this Message?": A Psychometric Thermometer for Bullying *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 695–706*

Nobata, Chikashi and Tetreault, Joel and Thomas, Achint and Mehdad, Yashar and Chang, Yi. 2016. Abusive language detection in online user content *Proceedings of the 25th international conference on world wide web, 145–153*

Park, Ji Ho and Fung, Pascale. 2017. One-step and two-step classification for abusive language detection on twitter *arXiv preprint arXiv:1706.01206*

Phillips, Whitney. 2015. This Is Why We Can't Have Nice Things: Mapping the Relationship between Online trolling and Mainstream Culture. The MIT Press, Cambridge.

Pitsilis, Georgios K and Ramampiaro, Heri and Langseth, Helge. 2018. Detecting Offensive Language in Tweets Using Deep Learning *arXiv preprint arXiv:1801.04433*

Ross, Björn and Rist, Michael and Carbonell, Guillermo and Cabrera, Benjamin and Kurowsky, Nils and Wojatzki, Michael. 2016. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. *Proceedings of NLP4CMC III: 3rd Workshop on Natural Language Processing for Computer-Mediated Communication, 17:6-9*

Tatjana Scheffler 2014. A German Twitter Snapshot *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)* European Language Resources Association (ELRA). Reykjavik, Iceland

Schmidt, Anna and Wiegand, Michael. 2017. A survey on hate speech detection using natural language processing *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, 1–10*

Verhoeven, Ben and Daelemans, Walter and Plank, Barbara. 2016. TwiSty: a multilingual Twitter Stylometry corpus for gender and personality profiling. *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016). Portoro, Slovenia*

Jin Wang and Zhongyuan Wang and Dawei Zhang and Jun Yan 2017. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, 2915–2921*

Warner, William and Hirschberg, Julia. 2012. Detecting hate speech on the world wide web *Proceedings of the Second Workshop on Language in Social Media, 19–26*

Waseem, Zeerak and Hovy, Dirk. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter *Proceedings of the NAACL student research workshop, 88-93*

Waseem, Zeerak. 2016. Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. *Proceedings of the First Workshop on NLP and Computational Social Science, 138-142* Educational Publishing Foundation.

Waseem, Zeerak and Davidson, Thomas and Warmsley, Dana and Weber, Ingmar. 2017. Understanding abuse: a typology of abusive language detection subtasks *arXiv preprint arXiv:1705.09899*

Ulli Waltinger 2010. GERMANPOLARITYCLUES: A Lexical Resource for German Sentiment Analysis *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)* electronic proceedings. Valletta, Malta

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

HIIwiStJS at GermEval-2018: Integrating Linguistic Features in a Neural Network for the Identification of Offensive Language in Microposts

Johannes Schäfer
Institute for Information Science and Natural Language Processing
University of Hildesheim, Germany
johannes.schaefer@uni-hildesheim.de

Abstract

This paper describes our submission for the GermEval-2018 shared task on the identification of offensive language. We use neural networks for both subtasks: *Task I — Binary classification* and *Task II — Fine-grained classification*. We comparatively evaluate the use of typical textual features with extensions also considering metadata and linguistic features on the given set of German tweets. Our final system reaches 73.69% macro-average F_1-score in a cross-validation evaluation for the binary classification task. Our best performing model for the fine-grained classification reaches an macro-average F_1-score of 43.24%. Furthermore, we propose methods to include linguistic features into the neural network. Our submitted runs in the shared task are: **HIIwiStJS_coarse_[1-3].txt** for *Task I* and **HIIwiStJS_fine_[1-3].txt** for *Task II*.[1]

1 Introduction

The automatic analysis of social media microposts such as *Twitter*[2] messages (*tweets*) gained more and more interest in recent years due to the necessity to process their increasing amount and variety. The anonymity of the web allows users to overcome their inhibitions more quickly which fosters the use of offensive language. Operators of social media websites are required to filter overly hurtful, derogatory or obscene comments and strive to acquire methods to automatically identify potentially offensive posts.

Schmidt and Wiegand (2017) present a survey on hate speech detection which is closely related to the detection of abusive language. They give an overview of typically used methods and features for the task, ranging from surface, sentiment, linguistic and knowledge-based features to higher-level features making use of lexical resources or metadata information. They especially point out the variety of the task and the lack of comparability of different research systems typically based on supervised learning, as no benchmark dataset is available.

Abusive language in English online user content is detected by a system developed by Nobata et al. (2016). They tackle the problem of noisiness of the data in conjunction with a need for world knowledge by using a feature-rich regression model. They exploit both character n-gram features as well as a variety of linguistic features including automatic syntactic and semantic annotations.

A multi-level classification of abusive language is given by Razavi et al. (2010). They particularly focus on flame detection and use a combination of classifiers supported by a dictionary of insulting and abusing language.

Most related work operates on English texts for which also the largest amount of data is available. One of the few presented research works on German data is given by Köffer et al. (2018). They collected a dataset of user comments on news articles from the web with a focus on the refugee crisis in 2015/16. Additionally, they provide a labeled dataset with comments marked as hateful or non-hateful and demonstrate the transferability of approaches developed for English data to German.

Neural networks (NNs) have been on the rise only in recent years as they require vast amounts of data and processing power which both only became available recently in the field of natural language processing. In applications on micropost classification neural networks also have seen research. For example, Del Vigna et al. (2017) perform hate speech detection on Italian user posts in the so-

[1] The IDs 1-3 correspond to our developed neural network systems used for the prediction as follows: ID 1 - Baseline model; ID 2 - Text & Metadata model; ID 3 - Text & Metadata & POS model.

[2] https://twitter.com/

cial network *Facebook*[3] using recurrent neural networks (specifically a LSTM network) which they compare to an approach using support vector machines.

Twitter data is analyzed by Founta et al. (2018) in an approach to detect different types of abusive behavior. They analyze both textual and user properties from different angles of abusive posting behavior in a deep learning architecture. In their model they consider a variety of metadata and include it into a NN model which learns text sequence features using a recurrent neural network. They show that training the sub-networks of different input types requires specific attention since simply training all of them at once in a combined model leads to unwanted interactions.

In the present paper, we also utilize neural networks to train models which identify offensive language in microposts from *Twitter*. Neural networks have the advantage to work with a high input dimensionality where it is not clear which features might be helpful concerning the prediction task. Given enough training data, the network is able to learn a complex, non-linear encoding of the input specifically for the desired classification. A certain intuition when selecting the features is however advised, since too many unrelated features can introduce a high amount of noise to the model.

Our approach shows similarities to the methods presented by Founta et al. (2018), however, we adapt the configuration of the network to our classification task, dataset and to tweets in German. Thus, in this work we first present a task, domain and language adaptation of their methods. Typically, neural networks are designed to only operate on raw textual input and are then fine-tuned to be able to learn patterns themselves. In our work the model is additionally given automatically pre-computed linguistic annotations. We present possibilities and early research of including such annotations into a combined neural network.

In the following sections we introduce our models in detail (Section 2), describe our dataset and linguistic processing (Section 3) and finally report on experiments (Section 4).

2 Methods

We implemented our models in *Python* using the module *keras* with the *TensorFlow* backend. To train our NN models we use the *Adam* optimizer and as loss function binary cross-entropy for the binary classification and categorical cross-entropy for the fine-grained classification respectively. We train the models on our dataset using a batch size of 64 which we determined experimentally to be most suitable.

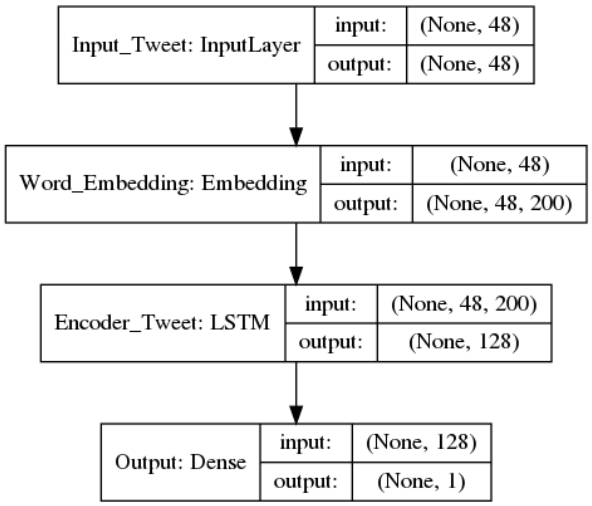


Figure 1: Baseline neural network, binary classification variant.

2.1 Baseline Model: Network Architecture

Our baseline model operates on the raw tweet input with a minimal amount of preprocessing. We compute word embeddings and feed them into a recurrent layer which is typically applied to sequence data. Finally, the output prediction is computed by a fully connected dense network. Figure 1 shows the overall structure of the baseline neural network with the specific dimensions[4] of the data going through the respective layers. In the following three sections we go over the detailed configuration.

2.1.1 Input and Embedding Layer

In the baseline model we use the raw tweet as input for the neural network in a simple input layer (*Input_Tweet* in Figure 1) which instantiates the *Keras* tensor.

Next, we decided to use word embeddings[5] to identify offensive language in tweets since we understand words as linguistically meaningful units in twitter messages; as these are also sufficiently

[3] https://www.facebook.com/

[4] Note that the unspecified ("None") dimension value in the figure corresponds to the number of samples which depends on the batch size (64 in our experiments).

[5] Brief experiments with character embeddings (also including convolutional layers capturing character n-grams) did not seem to lead to promising results on our dataset.

small, they lead to a feature vector of acceptable length and richness. Additionally, choosing word embeddings enables us to extract them from any tweet without relying on linguistic knowledge, only using a simple tokenizer. As further processing of our data in the neural network is more efficient when all input samples are of the same length, we (pre-) pad the tokenized tweets to sequences of 48 tokens. We determined this value specifically for our dataset by including the 95th percentile of all contained tweet lengths. Thus, with this method only 5% of the tweets are truncated.

As initial weights of the embedding layer we utilize pre-trained word embeddings which is a conventional method usually having the effect that the model converges faster. In our experiments we use the word embeddings provided by Cieliebak et al. (2017) which are trained with *Word2Vec* on 200 million German tweets using 200 dimensions and are available on the web[6].

When the weights of the embedding layer are set to not trainable, the total number of trainable parameters in our neural network is substantially reduced. This allowed us to design the remainder of the network in a very precise fashion, however, in this case the model cannot learn from out-of-vocabulary (OOV) words. In early experiments we found that only using the pre-trained embeddings as initial weights and allowing the model to fit these weights to the data during training leads to a better performance. Thus, we pre-compute only an embedding matrix using these weights which also contains randomized vectors for words in our dataset which are not contained in the embeddings. This approach also seems well-grounded since the used word embeddings are not specifically trained on abusive language and texts from social media in general tend to have a high number of OOV words.

Therefore, the output of our embedding layer for tweets (*Word_Embedding* in Figure 1) has the shape of: number of sequences, length of sequences (48), size of word vectors (200).

2.1.2 Recurrent Layer

As main neural network structure to encode the word sequence of a tweet we use a recurrent layer which sequentially processes data samples while constantly updating an internal state. Recurrent neural networks (RNNs) have proven to be highly efficient in modeling text since they intrinsically consider context information when learning predictions on word sequences. In early experiments we achieved the best performance with a RNN containing long short-term memory units (Hochreiter and Schmidhuber, 1997, LSTM), also known as a LSTM network which outperformed both a simple RNN and gated recurrent units (Cho et al., 2014, GRU). Even though our sequences are of relatively short length, we assume that the LSTM can track long-term dependencies nevertheless, for example, mentions of typical targets of insults at the start of a tweet with the actually insulting word being at the end of the tweet.

Experimentally, we determined a number of 128 units to be best performing for our LSTM network (*Encoder_Tweet* in Figure 1). To avoid overfitting, we set the recurrent dropout value of 0.5 in this layer. For further encoding we tested additional recurrent or fully-connected dense layers, however, did not achieve performance improvements.

2.1.3 Output Layer

The output layer (*Output* in Figure 1) of our neural network consists of a fully-connected dense layer which maps the output of the recurrent layer to a probabilistic prediction. To avoid overfitting to the training data, we apply L2 regularization in the kernel of this layer (with $\lambda = 0.01$).

For the binary classification task we use the sigmoid activation function and chose a single output unit which expresses the probability of the sample containing abusive language. The probabilistic prediction is transformed into a binary prediction using a 0.5 threshold.

For the fine-grained prediction we use the softmax activation function and as the number of output units the number of labels (4 in our dataset). The one label with the maximum probability is then selected as the predicted label for each sample.

2.2 Text & Metadata Model: Network Architecture

As second model we developed a neural network which combines textual sequence input in a sub-network similar to the above-mentioned baseline model with an additional metadata sub-network. An overview of the combined network is given in Figure 2. First, we describe in Section 2.2.1 the new metadata network and then in Section 2.2.2 we show how we include the metadata sub-network into the text sequence-based network.

[6] https://www.spinningbytes.com/resources/wordembeddings/

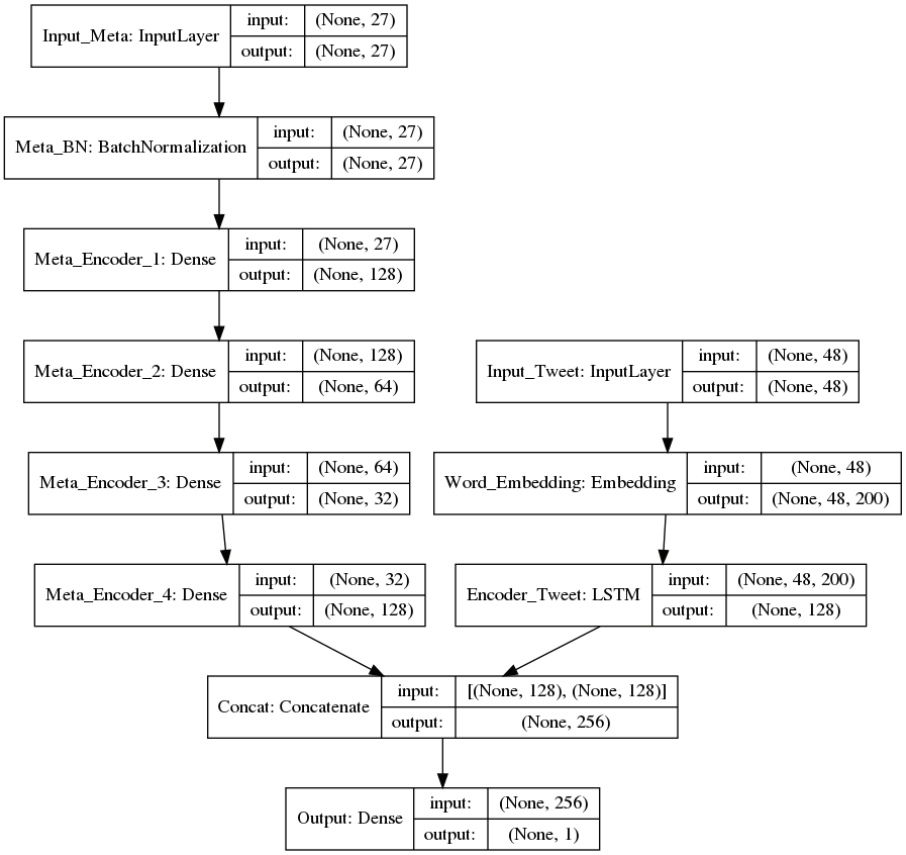


Figure 2: Text & metadata neural network, binary classification variant.

2.2.1 Metadata Network

In our setting we understand metadata as numerical data describing features of a single tweet going beyond its text, possibly with the aid of an external lexicon; however, not using its word sequence structure. For example, we extract the number of @-marked user name mentions and #-marked keywords from tweets, count special characters and attempt to match entries of pre-collected lexicons containing lists of profane words, words with known sentiment polarity or typical targets of abusive speech. The list of the 27 types of metadata considered in our experiments is given in Section 3.1.

The extracted numerical metadata features are fed into an input layer to instantiate the *Keras* tensor (*Input_Meta* in Figure 2). Next we apply a batch normalization layer (*Meta_BN*) transforming the values to a mean and unit variance of zero, which optimizes the neural network performance. To automatically compute a high-level encoding of our metadata features we utilize a technique known as bottleneck (Tishby and Zaslavsky, 2015; He et al., 2016) which consists of a sequence of several differently-sized fully-connected dense layers. We experimented with various configurations and found a sequence of three dense layers ((*Meta_Encoder_[1-3]* in Figure 2 with 128, 64 and 32 neurons respectively) to work best on our dataset. A forth dense layer (*Meta_Encoder_4*) is added which transforms the output to a tensor of the same dimensionality (128) as the output of the text sequence-based network. This supports the combination of the two sub-networks to a single neural network model which operates more efficiently on sub-networks of the same size.

For all dense layers in this network we use the *tanh* activation function as it works efficiently with standardized numerical data. An L2 regularization is also applied in all layers to reduce overfitting, however, with an relatively low $\lambda = 0.0001$. This can be justified since the input dimension of our metadata features is quite low (27), thus the model should not be overly complex to still be able to learn patterns.

2.2.2 Combined Text & Metadata Network

We combine the above-mentioned metadata network with the text sequence-based baseline net-

work described in Section 2.1 as follows. In front of the fully-connected output layer, we insert a concatenate layer which appends the output of the text encoder (*Encoder_Tweet* in Figure 2) to the output of the final layer of our metadata sub-network (*Meta_Encoder_4*). Finally, to predict the output probabilities we again use a fully-connected dense layer (*Output* in Figure 2 with the output dimension in the figure given as 1 for the binary classification). The configuration of this layer is unchanged from the baseline model with the exception that its input dimensionality is doubled (256) as we concatenate two tensors of size 128 (second dimension).

When training this network as a combined model at once, we struggled to achieve observable improvements in comparison to the baseline network. This is justified by the fact that the two paths might have a different convergence rate, which we observed in experiments. Thus, one path can dominate in predictions past a certain epoch and prevent the weights of the other path from getting significant updates. To avoid this problem, we use transfer learning. We first train both sub-networks separately, then freeze their weights and train the combined model using the pre-trained sub-networks. When training the sub-networks separately, we first remove the other respective sub-network and the concatenate layer as we can compute the output directly (e. g. when training the text sequence-based network we exactly train the baseline network as given in Figure 1). In a second step we remove the output layer of each sub-network, concatenate the output tensors and then add an output layer again to receive the network displayed in Figure 2. In the combined model, however, only the 257 weights of the final layer (*Output*) are trainable.

2.3 Text & Metadata Model extended with Linguistic Analyses

In this section we describe experimental methods where we integrate pre-computed linguistic analyses of the given text into our above-mentioned text & metadata neural network. We make use of the following three additional systems.

1. Compound splitter *COMPOST* (Cap, 2014).
2. Part-of-speech (POS) tag sequences of normalized tweets, computed using the Python module textblob-de[7].
3. Dependency parse trees for normalized tweets using the *mate* parser (Bohnet, 2010).

We use 1. to reduce the number of unknown words when importing the pre-trained word embeddings for our data. We first split all compounds in a tweet before running the word embedding look-up. Thus, we reduce the dimensionality of the embedding layer by reducing the number of distinct words as we consider components instead.

For integrating the POS tag sequences into our network, we simply add another LSTM-based sub-network with the same architecture as the text sequence-based network as described in Section 2.1 to our combined network. This new sub-network operates on trainable embeddings (initialized randomly) of the POS tag sequence as input.

We integrate the dependency structures with two different methods into our combined model. First, we also use the tag sequence similar to the above-mentioned POS tags in an additional LSTM-based sub-network. In a second approach we only consider the 1000 most frequent combinations of a word with its dependency arc label and encode these in a 1-hot vector for each tweet. This vector can then be fed into a network of fully-connected dense layers similarly to the metadata sub-network.

Finally, a top-level model combines the outputs of the sub-networks in a concatenate layer in the same fashion as we integrated the metadata sub-network.

3 Data processing

The training dataset of the *GermEval-2018 Shared Task on the Identification of Offensive Language* consists of 5,009 tweets with two annotation layers. No user metadata is given. The first annotation layer marks the binary classification of tweets, i. e. they are either marked when containing offensive language using the label "OFFENSE" or in any other case using the label "OTHER". The second annotation layer is used for the fine-grained classification tasks where three subcategories of offensive language are marked. Thus, this layer has four labels in total: "PROFANITY", "INSULT", "ABUSE" and "OTHER". Before training the neural networks on the raw tweets we apply certain pre-processing techniques which we describe in this chapter.

To be able to operate on words instead of the character sequence input, we apply tokenization using the *TweetTokenizer*[8] implemented in the

[7] http://textblob-de.readthedocs.io

[8] https://www.nltk.org/api/nltk.tokenize.html

nltk.tokenize Python module. This tokenizer is especially designed to process Twitter-specific expressions such as emoticons. It is introduced for English but, according to our observations, it also works fine on our German data.

We extract Twitter-specific mentions (marked by @) and #-marked keywords[9] from tweets using regular expressions. These lists are later used during the metadata extraction process.

To extract rather conventional linguistically structured sentences we apply a normalization step. Here, we remove *Twitter*-specific user name mention (@) and #-markers, line break markers, vertical bars inside words which were used to mark keywords when word affixes are present and replace xml-escaped symbols. Furthermore, we add missing whitespace characters after punctuation marks, replace ascii-emoticons with the corresponding Unicode characters and finally remove remaining special characters.

3.1 Metadata extraction

We extract metadata for each tweet from features going beyond its text sequence structure, however, only with the tweet text (and normalized text) as input. Typically used Twitter metadata features consider the author of a tweet, the number of his followers, the location, account age, etc. This group of metadata features are however not available in our dataset as the task focuses only on the linguistic content of microposts. All our metadata features are numerical while some take external source lexicons into account. We use the following external resources (all German language):

(i) List of strings matching typical targets of abusive speech, manually assembled according to the annotation guidelines of the shared task (Ruppenhofer et al., 2018) including: feminists, black people, muslims, jews, homosexuals (LGBT), refugees, members of political parties, etc.;
separate list for strings matching German public media names.

(ii) Lists of gender-specific names by the city of Cologne, available on the web[10].

(iii) Lists of positive, negative and neutral German polarity clues from the University of Bielefeld (Waltinger, 2010).

(iv) List of 1,782 swearwords from the web[11].

(v) Lists of words with positive and negative sentiment value from the University of Leipzig: SentiWS (Remus et al., 2010).

(vi) Lexicon of words with positive, negative and neutral sentiment values from the University of Zürich.[12]

Note that the latter two resources not only contain lists of words but additionally weights which we also consider for our metadata features.

In total we extract the following 27 metadata features for each tweet:

1. Length in number of characters;
2. Length of the normalized text in number of characters;
3. Number of words starting with an uppercase letter;
4. Number of user mentions (marked by @);
5. Number of user mentions in the first half of the tweet;
6. Number of user mentions in the second half of the tweet;
7. Number of matches of targets from list (i) in the normalized text;
8. Number of matches of public media-specific strings from list (i) in the list of mentions;
9. Number of matches of targets from list (i) in the list of mentions;
10. Number of female names in the mentions using list (ii);
11. Number of male names in the mentions using list (ii);
12. Number of #-marked keywords;
13. Number of matches of targets from list (i) in the list of keywords;
14. Number of matches of public media-specific strings from list (i) in the list of keywords;
15. Number of punctuation marks;
16. Number of reduplications of punctuation marks;
17. Number of special characters (mostly emoticons);
18. Number of words with uppercase letters only in the normalized text;

[9] According to `https://help.twitter.com/en/using-twitter/how-to-use-hashtags`, a hashtag written with a # symbol is used to index keywords or topics on Twitter while usernames are marked by the @ symbol. We assume that in our experiments these can be used specifically to find the target or typical topic of abusive comments.

[10] `https://offenedaten-koeln.de/dataset/vornamen`

[11] `http://www.insult.wiki/wiki/Schimpfwort-Liste`

[12] `http://bics.sentimental.li/files/8614/2462/8150/german.lex`

19. Number of matches of words with negative polarity according from list (iii) in the normalized text;
20. Number of matches of words with neutral polarity according from list (iii) in the normalized text;
21. Number of matches of words with positive polarity according from list (iii) in the normalized text;
22. Number of matches of swearwords from list (iv) in the normalized text;
23. Sum of negative sentiment values of matched words from list (v) in the normalized text;
24. Sum of positive sentiment values of matched words from list (v) in the normalized text;
25. Sum of negative sentiment values of matched words from lexicon (vi) in the normalized text;
26. Sum of positive sentiment values of matched words from lexicon (vi) in the normalized text;
27. Sum of neutral sentiment values of matched words from lexicon (vi) in the normalized text.

3.2 Linguistic Analyses

We run the external systems described in Section 2.3 on our data as follows. As system 1. computes word frequencies and determines split points according to a corpus, we add our tokenized tweets to the large German corpus *SdeWaC* (Faaß and Eckart, 2013) and input the combined corpus to the system. We finally use the output to map compounds of tweets to their components before computing word embeddings. The systems 2. and 3. are applied directly to the tokenized tweets.

4 Experiments

In this chapter we report on experiments using our different models on the *GermEval-2018* dataset. Additionally, we mention dataset-specific observations as well as configurations of the neural networks. We present results from a 10-fold cross-validation evaluation on the training data and describe our test runs with references to the filenames containing the respective test data predictions.

The vocabulary given our entire dataset (training and test data) consists of 9,812 distinct tokens with a frequency greater than one. As the embedding matrix contains all weights for all vocabulary entries, our embedding layer has 1,962,400 trainable weights (9,812 x 200 since the word vectors have 200 dimensions).

As the output predictions in the shared task are evaluated based on the macro-average F_1-score measure, which does not take the frequency of each class label into account, we optimized our model to predict all labels uniformly. We achieve this by adding class weights during training which we compute according to the inverse frequencies of the training data labels. Additionally, we smooth the weights by factor 5 to avoid the bias getting too strong, moving the values closer to the neutral weight (1). For binary classification this leads approximately to the following weights: 1.1 for the label "OTHER" and 1.4 for the label "OFFENSE". As the label frequencies are much more imbalanced for the fine-grained classification, we 'smooth' them using a factor of 0.5, effectively doubling the weights in comparison to their inverse frequency with: "PROFANITY": 144.4, "INSULT": 15.9, "ABUSE": 8.1 and "OTHER": 2.0. We want to note that using these weights does not lead to an optimized overall accuracy; however, it helps to balance the F_1-scores over the classes.

Baseline Model: In total the baseline network has 2,130,977 trainable weights for the binary classification (2,131,364 for the fine-grained classification). We find that the model converges during the binary classification experiments already after 4 epochs and during the fine-grained classification experiments after 10 epochs. The results of the 10-fold cross-validation on the training data are given in Table 1. The table shows the label-specific precision, recall and F_1-score values with the overall accuracy and macro-average F_1-score ($F_{1,\text{ macro-avg}}$). All these values are averaged over the 10 folds. The baseline model detects tweets marked to contain offensive language with an F_1-score of 61.88%. Overall the macro-average F_1-score is 71.93% for binary classification. Results for the prediction of the fine-grained classes are given in Table 2. Here, the macro-average F_1-score is considerably lower with 43.24%. Instances annotated with the label "PROFANITY" are most difficult to detect as the model only reaches an F_1-score of 13.21%. Considering all labels, the macro-average F_1-score during the 10-fold cross-validation on our training data of our baseline fine-grained classifier is 43.24%.

We produce the first two test runs using the exact same configurations as during the described cross-validation setting, however, training on the full training data and predicting the given test data (*HIIwiStJS_coarse_1.txt* using the baseline binary classifier and *HIIwiStJS_fine_1.txt* using the baseline fine-grained classifier).

Model	Label "OTHER" precision	recall	F_1-score	Label "OFFENSE" precision	recall	F_1-score	Acc.	$F_{1,\text{ macro-avg}}$
Baseline NN	80.45	83.82	81.94	65.86	59.32	61.88	75.64	71.91
Text&Meta	**81.49**	84.46	**82.92**	**67.09**	**62.19**	**64.45**	**76.98**	**73.69**
Text&Meta&POS	80.03	**85.55**	82.69	66.98	57.85	62.03	76.28	72.36

Table 1: 10-fold cross-validation result scores in % for the binary classification task.

Model	"OTHER" F_1-score	"ABUSE" F_1-score	"INSULT" F_1-score	"PROFANITY" F_1-score	Acc.	$F_{1,\text{ macro-avg}}$
Baseline NN	78.16	**48.24**	33.35	**13.21**	65.24	**43.24**
Text & Meta	**79.14**	48.04	**35.79**	7.57	**66.34**	42.63

Table 2: 10-fold cross-validation result scores in % for the fine-grained classification task.

Text & Metadata Model: The combined network consists of three models which are trained separately: 1. The text sequence-based sub-network which is trained exactly like the baseline network and converges after 4 epochs for binary classification (10 training epochs for the fine-grained classification). 2. The metadata sub-network (with 18,714 trainable weights) we find to converge after 40 epochs for the binary classification (100 training epochs for the fine-grained classification). 3. The combined model (1,028 trainable weights) which uses the pre-computed sub-networks converges after only 4 epochs for the binary classification and 6 epochs for the fine-grained classification.

The results for the binary prediction in the 10-fold cross-validation on the training data are given in Table 1. The evaluation shows that the extended model outperforms the baseline model by 1.78% macro-average F_1-score. Especially the prediction of tweets labeled as containing offensive language is improved with and F_1-score of 64.45% which is 2.57% more than to the baseline result.

The results for of the extended model the fine-grained prediction in the 10-fold cross-validation on the training data are given in Table 2. Here, the overall macro-average F_1-score is 0.61% lower than the score for the baseline while the accuracy is 1.10% higher. Observing the label-specific F_1-scores also shows an unclear pattern, as the values only improve for half of the labels.

We compute two test runs using the text & metadata network using the exact same configurations as during the described cross-validation setting, however, training on the full training data and predicting the given test data (*HIIwiStJS_coarse_2.txt* using the binary classifier and *HIIwiStJS_fine_2.txt* using the fine-grained classifier).

Text & Metadata Model extended with Linguistic Analyses: Early experiments on integrating the linguistic analyses in our described approach led to mixed results. We only report on a few experiments here as the research is still ongoing and most models still require fine-tuning.

Using the compound splitter to normalize tweets substantially reduces the size of vocabulary which speeds up training, however, the performance deteriorates. We assume that it might be necessary to train new word component embeddings using compound splits on a large corpus as initial weights.

Furthermore, simply integrating the sub-networks based on dependency parses does not seem to improve the performance of the model.

Finally, we report the results when using three sub-networks: the two sub-networks of our Text & Metadata model extended by and additional sub-network operating on sequences of POS tags. We train the POS-based sub-network separately for 50 epochs and the combined model for 6 epochs. The evaluation scores of our cross-validation are given in Table 1 for the binary classification. Note that this model reaches the highest recall for the label "OTHER" in comparison to the other models while it performs worse according to all other evaluation scores. The model seems to be overfitted to this label which might signal that this approach is not fully optimized. We compute two final test runs using this Text & Meta & POS-based NN configuration, training on the full training data and predicting the given test data (*HIIwiStJS_coarse_3.txt* using the binary classifier variant and *HIIwiStJS_fine_3.txt* using the fine-grained classifier variant[13]).

[13] We train the POS-based sub-network in 120 epochs.

5 Conclusion and Future Work

In this paper we described our system runs and methods for the identification of abusive language in microposts. When integrating further feature types we observed that fine-tuning the complex neural networks gets much more difficult and time-consuming. However, we manage to improve our baseline model macro-average F_1-score by 1.78% to 73.69% when adding metadata features. Furthermore, we presented early findings on using linguistic annotations in additional neural sub-networks, which requires more optimization steps. We plan to focus in future work on more in-depth analyses on the integration of linguistic annotations into the neural network to further improve the performance of the system in modeling offensive language.

Acknowledgments

We would like to thank Ulrich Heid for his valuable feedback and support.

References

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd international conference on computational linguistics*, pages 89–97. Association for Computational Linguistics.

Fabienne Cap. 2014. *Morphological processing of compounds for statistical machine translation*. Ph.D. thesis, Institute for Natural Language Processing (IMS), University of Stuttgart.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Mark Cieliebak, Jan Milan Deriu, Dominic Egger, and Fatih Uzdilli. 2017. A Twitter corpus and benchmark resources for German sentiment analysis. In *5th International Workshop on Natural Language Processing for Social Media, Boston, MA, USA, December 11, 2017*, pages 45–51. Association for Computational Linguistics.

Fabio Del Vigna, Andrea Cimino, Felice DellOrletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*, Venice, Italy.

Gertrud Faaß and Kerstin Eckart. 2013. SdeWaC–A corpus of parsable sentences from the web. In *Language processing and knowledge in the Web*, pages 61–68. Springer.

Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2018. A unified deep learning architecture for abuse detection. *CoRR*, abs/1802.00385.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sebastian Köffer, Dennis M Riehle, Steffen Höhenberger, and Jörg Becker. 2018. Discussing the value of automatic hate speech detection in online debates. In *Multikonferenz Wirtschaftsinformatik (MKWI 2018): Data Driven X - Turning Data in Value*, Leuphana, Germany.

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. International World Wide Web Conferences Steering Committee.

Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence*, pages 16–27. Springer.

Robert Remus, Uwe Quasthoff, and Gerhard Heyer. 2010. SentiWS – a Publicly Available German-language Resource for Sentiment Analysis. In *Proceedings of the 7th International Language Resources and Evaluation (LREC)*, pages 1168–1171.

Josef Ruppenhofer, Melanie Siegel, and Michael Wiegand. 2018. Guidelines for IGGSA Shared Task on the Identification of Offensive Language, March 12, 2018.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE.

Ulli Waltinger. 2010. GERMANPOLARITYCLUES: A Lexical Resource for German Sentiment Analysis. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta, May. electronic proceedings.

ULMFiT at GermEval-2018: A Deep Neural Language Model for the Classification of Hate Speech in German Tweets

Kristian Rother
Hochschule Hamm-Lippstadt
Marker Allee 76-78
59063 Hamm
kristian.rother@hshl.de

Achim Rettberg
Hochschule Hamm-Lippstadt
Marker Allee 76-78
59063 Hamm
achim.rettberg@hshl.de

Abstract

This paper describes the entry hshl_coarse_1.txt for *Task I (Binary Classification)* of the *Germeval Task 2018 - Shared Task on the Identification of Offensive Language*. For this task, German tweets were classified as either offensive or non-offensive. The entry employs a task-specific classifier built on top of a medium-specific language model which is built on top of a universal language model. The approach uses a deep recurrent neural network, specifically the AWD-LSTM architecture. The universal language model was trained on 100 million unlabeled articles from the German Wikipedia and the medium-specific language model was trained on 303,256 unlabeled tweets. The classifier was trained on the labeled tweets that were provided by the organizers of the shared task.

1 Introduction

Hate speech is on the rise in online communication and can come in different forms but usually follows certain patterns (Mondal et al., 2017). Additionally social media serves as a breeding ground for deviant behavior following real world incidents (Williams and Burnap, 2015).

Hate speech has psychological consequences for the victims such as fear, anger and vulnerability (Awan and Zempi, 2015) as well as the worry that online threats may become a reality (Awan and Zempi, 2016). Additionally, hate speech can be the harbinger of actual violence. Hate speech towards a group can serve as a predictor of violence towards that group (Müller and Schwarz, 2018a) and Twitter use can fuel hate-crimes (Müller and Schwarz, 2018b).

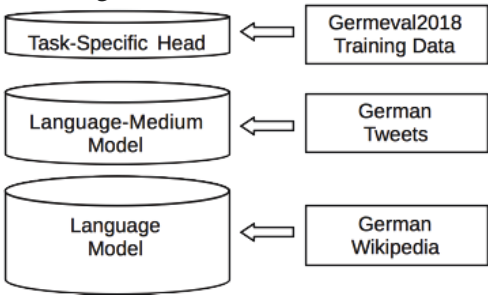
Institutions and legislators have reacted to this trend towards hate speech. The European Commission and multiple social media companies agreed to a code of conduct on countering illegal hate speech online (European Commission, 2016). Germany passed the *Network Enforcement Act* on September 1st 2017 to enforce fines of up to 50 million Euros against social media companies that fail to delete illegal content (German Bundestag, 2017). The law specifically includes hate speech (§§130, 166 and 185-187 of the Criminal Code).

Due to the negative impact of hate speech and the amount of social media data that is generated every day, automated detection and classification of hate speech has been studied widely. Recent overviews can be found in (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018). However, with some exceptions such as (Ross et al., 2017) and (Van Hee et al., 2015), the scope of the studies is often limited to the English language. Therefore, this paper tries to contribute to the improvement of the state of the art in German hate speech detection by describing the entry hshl_coarse_1.txt which participated in the binary classification task at Germeval 2018.

2 Experimental Setup

The overall setup closely follows the ULMFiT method (Howard and Ruder, 2018) as depicted in figure 1.

Figure 1: ULMFiT Overview.



The general idea is to split the training process into three parts. First, a *language model (LM)* is trained from a large corpus of unlabeled data.

This model is used as the basis to train a *language-medium model (LMM)* from unlabeled data that matches the desired medium of the task (e.g. forum posts, newspaper articles or tweets). Finally, a *task-specific head (TSH)* like a hate speech classifier or a sentiment classifier is trained on top of this model from a labeled dataset. This approach facilitates the reuse of pretrained models for the lower layers.

2.1 Technical Resources

All experiments were conducted in Jupyter Notebooks (Kluyver et al., 2016) running Python 3 (Python Software Foundation, 2018) kernels with the following libraries:

- pytorch (Paszke et al., 2017)
- fast.ai (Howard and others, 2018)
- pandas (McKinney, 2010)
- numpy (Oliphant, 2006)
- scikit-learn (Pedregosa et al., 2011)
- matplotlib (Hunter, 2007)
- spaCy (Honnibal and Montani, 2018)

All models were trained on a desktop computer with an Intel i7-6850 CPU, 32 GB of RAM and a GTX 1080 GPU with 8 GB of RAM. A fixed seed was used for the random number generators.[1]

2.2 Data

To train the entire model end to end, three data sources were used. The language model was trained on a dump of the entire German Wikipedia. Only the top 100 million articles (with a character length of at least 2,000) were kept and the vocabulary was limited to 50,000 tokens. Because this is the same approach as Wikitext-103 (Merity et al., 2016) the model will be called W103-DE-50k.

The language-medium model was trained on 303,256 unlabeled tweets[2] that were collected with a custom script using the Twitter-Streaming-API.

Finally, to train the task-specific head, the 5,009 labeled tweets that were provided by the Germeval 2018 competition organizers were used. The data is summarized in table 1 and the distribution of the labels for the binary classification task is shown in table 2

Model	Medium	Items	Type
TSH	Twitter	5,009	Labeled
LMM	Twitter	303,256	Unlabeled
LM	Wikipedia	100,000,000	Unlabeled

Table 1: Training Data.

Category	Items	Percent
Offensive	1,688	33.7
Other	3,321	66.3

Table 2: Frequencies of the Categories in the Training Set.

spaCy was used to tokenize the data and some additional preprocessing as in (McCann et al., 2017; Johnson and Zhang, 2017) was applied. Both the Wikipedia and Twitter data was sanitized with a custom function by replacing html-code and other unwanted characters with sensible ones (e.g., replacing nbsp; with a space or
 with a newline). For the Wikipedia data, labels for the beginning of an article and for the beginning of a paragraph were added. For the tweets only a beginning of tweet token was added and all @username occurrences were replaced with the label x_user_mention and all urls were replaced with x_url_mention. Finally, special tokens for upper-case words, elongation, repetition, unknown and padding were inserted. For the language model, the vocabulary was capped at the most frequent 50,002 tokens with a minimum frequency of 5. The medium-language model has a vocabulary of 33,191 tokens.

All datasets were split into a training and a validation set by randomly separating 10% of the data from the rest.

2.3 Architecture

Due to the sequential nature of the task, a recurrent neural network (RNN) architecture was employed. Specifically, the weight-dropped AWD-LSTM variant (Merity et al., 2017) of the long short-term memory network (Hochreiter and Schmidhuber, 1997) and (Gers et al., 1999) was used. The chosen embedding size was 400, the number of hidden activations per layer was 1150 and the number of layers was 3. For the classifier, two linear blocks with batch normalization and dropout were added

[1] With this setup, the training of one epoch of the language model took approximately 2 hours and 50 minutes.

[2] A rule of thumb from correspondence at the forums hosted by one of the ULMFiT-authors is to use between 5x and 10x of the available training data for this step.

to the model with rectified linear unit activations for the intermediate layer and a softmax activation at the last layer (Howard and Ruder, 2018).

2.4 Hyperparameters

The hyperparameters are similar across all stages of the ULMFiT method. The batch size was limited by the available GPU memory and always set to the highest possible value. Back propagation through time (BPTT) was set to 70 for all models. Apart from these parameters, the models used different configurations for the learning rate (LR), weight decay (WD), dropouts, cyclical learning rates (CLR) (Smith, 2017) and slanted triangular learning rates (STLR) (Howard and Ruder, 2018). Additionally, gradient clipping (Pascanu et al., 2013) was applied to some of the models.

For the dropouts, the two configurations that are summarized in table 3 were used. They are taken from the Github repository[3] corresponding to (Howard and Ruder, 2018) and the Github repository[4] corresponding to (Merity et al., 2017). The dropout multiplier, when configured, is applied to all dropouts. For the CLR the four parameters are maximum to minimum learning rate divisor, cooldown percentage, maximum momentum and minimum momentum in that order and for the STLR the parameters are maximum to minimum learning rate divisor and cut_fract.

Dropout	Howard	Merity
Input Layer	0.25	0.6
General	0.1	0.4
LSTM's Internal	0.2	0.5
Embedding Layer	0.02	0.1
Between LSTM Layers	0.15	0.2

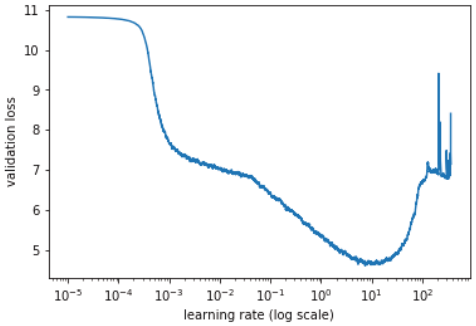
Table 3: Dropout configurations.

2.4.1 Language Model

To obtain a sensible learning rate, the learning rate finder (LRF) introduced by (Smith, 2017) was used. The graph for the LRF is depicted in figure 2.

The hyperparameters for model C are directly transfered from (Howard and Ruder, 2018)[5]. The hyperparameters for model D are a variation of

Figure 2: LRF - Language Model.



these parameters. The hyperparameters for model A, B and E were deduced from the learning rate finder and some short experiments.

Discriminative learning rates (Howard and Ruder, 2018) of [lr/6, lr/4, lr, lr] were used for models C and D for the four layer-groups. A fixed learning rate was used for all other models.

The batch size for all language models was set to 32 and a BPTT of 70 was used. A gradient clipping of 0.4 and 0.12 was applied to model B and D respectively. Model C used STLR with a ratio of 32 and a cut_fract of 0.1. Models A, B and E used CLR with the parameters 10, 10, 0.95, 0.85 and model C used CLR with the parameters 10, 33, 0.8, 0.7. Adam was used as the optimizer for models C and D and stochastic gradient descent was used for the other models. Table 4 summarizes the remaining hyperparameters.

Model	LR	WD	Dropout
A	2	1e-7	Howard * 0.5
B	1.4	1e-7	Howard * 0.4
C	3e-4	1e-7	Howard * 0.5
D	2e-3	1e-6	Merity * 0.2
E	5.12	1e-7	Merity * 0.5

Table 4: Language Model Hyperparameters.

2.4.2 Language-Medium Model

A learning rate finder was used to determine suitable candidate learning rates. The graph is depicted in figure 3.

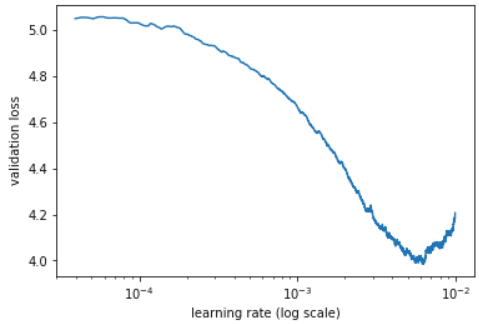
The batch size for all language-medium models was set to 32 and a BPTT of 70 was used. The weight decay was set to 1e-7 for all models and no gradient clipping was used. The model was gradually unfrozen (Howard and Ruder, 2018) by unfreezing the last layer first and then unfreezing all remaining layers. Slanted triangular learning

[3] https://github.com/fastai/fastai/tree/master/courses/dl2/imdb_scripts

[4] https://github.com/Smerity/awd-lstm-lm

[5] Specifically, these up to date parameters were used: https://github.com/fastai/fastai/tree/master/courses/dl2/imdb_scripts

Figure 3: LRF - Language-Medium Model.



rates (Howard and Ruder, 2018) with a ratio of 32 and a cut_fract of 0.5 were used after the last layer was unfrozen and a ratio of 20 and a cut_fract of 0.1 was used when all layers were unfrozen. The hyperparameters of all four models are summarized in table 5. The columns LR-Last and LR-All refer to the learning rates for the runs were only the last layer was unfrozen and where all layers were unfrozen.

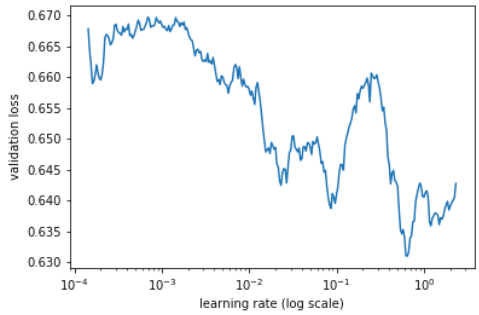
Model	LR-Last	LR-All	Dropout
LMM1	3e-3	3e-3	Howard * 0.7
LMM2	1e-3	1e-3	Howard * 0.7
LMM3	4e-3	3e-3	Howard * 0.3
LMM4	5e-3	1e-3	Howard * 0.3

Table 5: Language-Medium Model Hyperparameters.

2.4.3 Task-Specific Head

A learning rate finder (see figure 4) was used to find the learning rate of 3e-1. The batch size for the classifier was set to 52 and a BPTT of 70 was used.

Figure 4: LRF - Task-Specific Head.



The model was gradually unfrozen layer by layer with the same hyperparameters applied to each layer. The weight decay was set to 1e-7. Cyclical learning rates with the parameters 10,10,0.98 and 0.85 were used. The Howard dropouts were used with a multiplier of 1.8 and no gradient clipping was applied. The optimizer was stochastic gradient descent.

3 Experiments

The working hypothesis that an increased performance of the lower layers improves the results at the upper layers lead to the decision to try five different hyperparameter configurations for the language model. The models A-E were trained on unlabeled Wikipedia data for 25 epochs. Model A was trained for an additional 50 epochs and used as the basis for the language-medium model.

Four different hyperparameter configurations for the language-medium model (LMM1-LMM4) were trained on unlabeled tweets for 30 epochs each. Afterwards the best model (LMM1) was used as the basis for the hate speech classifier.

Lastly, The hate speech classifier was trained on the provided training data of labeled tweets. The hyperparameters were tuned during experimentation by picking a learning rate that lead to convergence with overfitting and regularizing via the other parameters until the model didn't overfit anymore.

4 Results

The perplexities for the language models are depicted in table 6. Models A, B and E outperformed the other models and converged while models C and D were underfitting. The perplexity of the best LM is 27.39 which is better than the best perplexity for a non-ensembled English language model on the One Billion Word benchmark (30.0) that was reported in the summary by (Jozefowicz et al., 2016) and the current state of the art (28.7) for the same corpus (Bakhtin et al., 2018). For comparison, the best published result for the English Wikitext-103 is 40.8 (Grave et al., 2016). To our knowledge, the best perplexity for a word level German language model that uses the Wikipedia is 36.95 (Van Hee et al., 2015).

After 30 epochs, the language-medium model LMM1 showed the best overall result with a perplexity of 17.64.

To get a feeling for the quality of the hate speech classifier, the labels for the validation set were predicted. Table 7 summarizes the results.

Model	Validation Loss	Perplexity
A	3.31	**27.39**
B	3.41	30.27
C	3.81	45.15
D	3.68	39.65
E	3.38	29.37

Table 6: Language Model Perplexities. Lower is better.

Class	Precision	Recall	F1	Support
Offensive	0.73	0.68	0.71	179
Other	0.83	0.86	0.85	322
Average	0.8	0.8	0.8	

Table 7: Results Binary Classification.

5 Conclusion

The paper presented the submission hshl_coarse_1.txt that was entered for the binary hate speech classification task of Germeval 2018. It used a deep recurrent neural net, specifically an AWD-LSTM architecture, to classify German tweets as offensive or non-offensive.

A three layered approach based on the ULMFiT method was used to train the classifier. First, a German language model was trained from unlabeled Wikipedia data. A language-medium model for German tweets was trained on top of this model from unlabeled tweets and served as the backbone to train the hate speech classifier on the provided labeled training data. This classifier achieved an average F1 score of 80 on the validation data.

All relevant code will be made available at one of the authors' Github repositories[6]. The German language model with a vocabulary size of 50,000 tokens achieved a perplexity of 27.39. It will be released as W103-DE-50k and a link will be added to the repository.

6 Outlook

The proposed approach towards hate speech classification can be improved in various ways. The working hypothesis that better lower layer results improve the classifier needs empirical support but assuming it holds, the overall results could be improved by improving the lower layers. Instead of relying on a single model at each stage an ensemble of models could be used. A good starting point would be turning all models into bidirectional models (Peters et al., 2017). Different architectures such as Quasi Recurrent Neural Networks (Bradbury et al., 2016) or Contextual LSTM (Ghosh et al., 2016) or general improvements like continuous caches (Grave et al., 2016) could improve the overall results further.

The idea of super-convergence (Smith and Topin, 2017) might also be worth investigating and some of the ideas outlined in the overview by (Schmidt and Wiegand, 2017) could be tried.

Lastly, hate speech dictionaries could be used to construct a keyword-filter for the Twitter-API to collect more data for the offensive category to improve the classifier by effectively increasing the size of the training set.

[6] https://github.com/rother

7 Acknowledgements

We thank the Behr-Hella Thermocontrol GmbH for supporting this research. We also thank all reviewers and the competition organizers.

References

Imran Awan and Irene Zempi. 2015. We fear for our lives: Offline and online experiences of anti-Muslim hostility. *Report,[online] available: http://tellmamauk. org/wp-content/uploads/resources/We% 20Fear% 20For% 20Our% 20Lives. pdf [accessed: 7 January, 2016]*.

Imran Awan and Irene Zempi. 2016. The affinity between online and offline anti-Muslim hate crime: Dynamics and impacts. *Aggression and violent behavior*, 27:1–8.

Anton Bakhtin, Arthur Szlam, Marc'Aurelio Ranzato, and Edouard Grave. 2018. Lightweight Adaptive Mixture of Neural and N-gram Language Models. *arXiv preprint arXiv:1804.07705*.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-Recurrent Neural Networks. *arXiv:1611.01576 [cs]*, November. arXiv: 1611.01576.

European Commission. 2016. Code of conduct on countering illegal hate speech online. http://ec.europa.eu/newsroom/document.cfm?doc_id=42985.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Comput. Surv.*, 51(4):85:1–85:30, July.

German Bundestag. 2017. Act to improve enforcement of the law in social networks (network enforcement act). https://www.bmjv.de/SharedDocs/Gesetzgebungsverfahren/Dokumente/NetzDG_engl.pdf?__blob=publicationFile&v=2.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM. In *9th International Conference on Artificial Neural Networks: ICANN '99*, pages 850–855.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual LSTM (CLSTM) models for Large scale NLP tasks. *arXiv:1602.06291 [cs]*, February. arXiv: 1602.06291.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew Honnibal and Ines Montani. 2018. spaCy library. `https://spacy.io`.

Jeremy Howard et al. 2018. fast.ai library. `https://github.com/fastai/fastai`.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. *arXiv:1801.06146 [cs, stat]*, January. arXiv: 1801.06146.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–570. Association for Computational Linguistics.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. 2016. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Wes McKinney. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Mainack Mondal, Leandro Arajo Silva, and Fabrício Benevenuto. 2017. A measurement study of hate speech in social media. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, pages 85–94. ACM.

Karsten Müller and Carlo Schwarz. 2018a. Fanning the Flames of Hate: Social Media and Hate Crime. CAGE Online Working Paper Series 373, Competitive Advantage in the Global Economy (CAGE).

Karsten Müller and Carlo Schwarz. 2018b. Making America Hate Again? Twitter and Hate Crime under Trump.

Travis E. Oliphant. 2006. *A guide to NumPy*, volume 1. Trelgol Publishing USA.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS 2017 Workshop Autodiff*.

Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv:1705.00108 [cs]*, April. arXiv: 1705.00108.

Python Software Foundation. 2018. Python programming language. `https://python.org`.

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*.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Leslie N. Smith and Nicholay Topin. 2017. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. *arXiv:1708.07120 [cs, stat]*, August. arXiv: 1708.07120.

Leslie N. Smith. 2017. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Vronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events. In *International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 672–680.

Matthew L. Williams and Pete Burnap. 2015. Cyberhate on social media in the aftermath of Woolwich: A case study in computational criminology and big data. *British Journal of Criminology*, 56(2):211–238.

German Hate Speech Detection on Twitter

Samantha Kent
Fraunhofer FKIE
Fraunhoferstraße 20
53343 Wachtberg
samantha.kent@
fkie.fraunhofer.de

Abstract

This paper describes our system submission for the GermEval 2018 shared task on the identification of German hate speech in Tweets at Konvens 2018. We trained and tested a Logistic Regression classifier with 10-fold cross validation using character n-grams as features. We achieved a macro F1 of 76.72 for the coarse-grained classification task and 47.17 for the fine-grained task when testing the classifiers on a small development set we created.

1 Introduction

Germany recently passed the Network Enforcement Act[1], a law stating that social media companies such as Twitter and Facebook are obliged to remove hate speech and other illegal activity from their websites. In light of this new law, hate speech on social media has been receiving more and more attention and raises the question of how to automatically detect it. Twitter's user guidelines define hateful conduct by stating "You may not promote violence against or directly attack or threaten other people on the basis of race, ethnicity, national origin, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease." [2] Tweets in violation of the hateful conduct policy, such as the example (1), must be removed.

(1) @welt Abschieben es sind doch nur Moslems!!

@welt Deport they are just Muslims!!

While the definition in itself might seem straightforward, actually agreeing upon what hate speech entails is much more complex. Ross et al. (2016) point out that defining hate speech is difficult and "a given statement may be considered hate speech or not depending on someone's cultural background and personal sensibilities". To test the reliability of annotated hate speech corpora, they conducted an experiment where 56 annotators categorized German Tweets according to whether they contained hate speech or not. They found a very low inter-annotator agreement, even when providing annotators with Twitter's hateful conduct definition. A similar observation was made by Waseem (2016), who reports that it seems to be difficult to annotate hate speech corpora without an intimate knowledge of hate speech.

Currently, most automatic hate speech detection research is conducted in English, and there is a need for research in other languages. Thus, the main aim in this paper is to contribute to German hate speech detection methods.

2 Related Work

Automatic hate speech detection research, in particular on Twitter, can be roughly divided into two separate areas. Namely, binary classification tasks where the goal is to identify whether a Tweet contains hate speech or not, and domain specific classification tasks where the Tweets contain, for example, racist and sexist remarks (Waseem and Hovy, 2016) or remarks pertaining to the German refugee crises (Ross et al., 2016).

There are a number of different methods used in hate speech detection. Malmasi and Zampieri (2017) use a linear Support Vector Machine with which they obtain an accuracy of 78.0%. This was achieved using character 4-grams only, as they found that the addition of further features did not improve overall performance. They also point out that one of the particular difficulties in the binary classification tasks is making a distinction between hate speech and Tweets containing profane or offensive language that should not be classified as

[1] https://www.gesetze-im-internet.de/netzdg/BJNR335210017.html

[2] https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy

such.

Waseem and Hovy (2016) created a hate speech corpus and used a logistic regression classifier to identify racist and sexist Tweets. They found the most predictive features are character bi- to fourgrams combined with the information about the gender of the person sending the Tweet and achieved an F1 of 0.739. Badjatiya et al.(2017) used a combination of deep learning methods and gradient boosted decision trees and tested their model on the dataset created by Waseem and Hovy (2016), and reported an F1 score of 0.93.

3 Classification

We used scikit-learn 0.19 (Pedregosa et al., 2011) to train and test a Logistic Regression classifier. The optimal parameter settings were found to be: C= 100.0, random_state = 2, tol = 10.0, dual = True. The classifier was evaluated using 10-fold cross validation and a development Tweet set (see section 3.1 for details). We used macro F1 as an evaluation metric and compared the results to a majority class baseline.

3.1 Data

The data was supplied as part of the GermEval 2018 shared task and consists of a total of 8541 German Tweets. 5009 Tweets are used for training and 3532 for testing. The Tweets have been annotated so that there are two separate classification tasks. Firstly, the coarse-grained (binary) classification task distinguishes between the classes "offense" and "other", where the former contains hate speech and the latter does not. Secondly, for the fine-grained classification task, the class "offense" has been been split into three subclasses. Thus, each tweet in the dataset is accompanied by a coarse and a fine grained tag. The different classes are as follows:

1. Offense - Tweets that contain hate speech.
   - 1 a) Profanity — The use of profane words without insulting someone.
   - 1 b) Insult — Profanity directed at an individual with the intention to insult them.
   - 1 c) Abuse — The most severe form of hate speech where negative characteristics are ascribed to a group of people.
2. Other — Tweets that do not contain hate speech.

To thoroughly test the classifier, the training data was split into a training and development set. This allows for the identification of potential overfitting or underfitting of the classifier on the training data. The last 500 Tweets, approximately 10%, were cut from the training data to form the development set. The distribution of classes in the training and development sets is reported in table 1. Both sets have an extremely similar distribution in terms of the occurrence of hate speech, as 33.6% of Tweets in the training data and 33.8% in the development data are annotated as hate speech.

Class	Tweets	%
Training set:		
Offense	1519	33.6
Other	2990	66.3
Total	4509	100
Development set:		
Offense	169	33.8
Other	331	66.2
Total	500	100

Table 1: The number of Tweets per class in the training and development data.

A similar distribution can also be found within the "offense" class. Table 2 shows the distribution of the different types of hate speech is almost identical in the training and development Tweets.

Offense	% Train Tweets	% Dev. Tweets
Profanity	4.2 (n=64)	4.1 (n=7)
Insult	35.3 (n=536)	34.9 (n=59)
Abuse	60.5 (n=919)	60.9 (n=103)
Total	100 (n=1519)	100 (n=169)

Table 2: The distribution of Tweets within the offense class in the training and development data sets.

3.2 Feature Description

Different types of features were tested while constructing the classifier. We performed a search to find the best feature combination and found that the best results were obtained using only character n-grams. The other features listed below were tested but did not contribute to overall performance. They consist of lexical lookups and basic linguistic features such as those suggested in Nobata et al. (2016).

- N-grams — These are the most basic features that contribute the most in terms of performance. We employ character n-grams ranging from 1 to 6 characters each weighted by their TF-IDF.

- The number of characters in a Tweet.
- The number of tokens in a Tweet.
- The number of non-alphanumeric characters
- The number of words in a Tweet containing an asterisk, a symbol which is often used to disguise swear words.
- The number of words present in a German swear word list. [3]
- The positive and negative sentiment score based on the presence in the positive or negative polarity lists created by Waltinger (2010).
- The emoji feature was used to process the sentiment contained in the emoji's in the Tweets. It consists of a list lookup in a positive and a negative emoji list[4]. Both lists are small and contain only 22 and 24 entries respectively, so that the focus lies on precision rather than recall (Davidson et al., 2017).

3.3 Pre-processing

All Tweets were pre-processed prior to classification. Firstly, the Tweets were anonymized by removing all user names. Secondly, all punctuation was removed. And finally, all Tweets were lemmatized using the Spacy [5] lemmatizer.

4 Results

The results section is split into two separate parts. Given that the results on the final test data are not yet available at the time of writing, the first part of the results section will focus on the preliminary results obtained by testing the classifiers on the development set. The description of the predictions submitted to the GermEval organizers can be found in section 4.2.

[3]The swear word list was retrieved from http://www.hyperhero.com/de/insults.htm

[4]The positive and negative emoji lists were obtained from https://unicode.org/emoji/charts/full-emoji-list.html

[5]The Tweets were lemmatized using https://spacy.io/api/lemmatizer

4.1 Preliminary results

The coarse-grained classification results are shown in table 3. The best performing classifier used character 1-5 grams as features and achieves a macro F1 of 76.26 on the 500 Tweet development set. None of the other features contribute to the performance. For example, the addition of the emoji feature or the sentiment scores seems to increase the cross validated F1, but decreases performance when tested on the development data.

The macro F1 is consistently slightly lower on the cross-validated training data. We suspect this is due to variance in the data. The results fluctuate depending on the instances provided to the classifier in the training and the development sets. During cross-validation on the training data, it became apparent that the range of different results is larger for the classifier with character 1-2 grams as a feature, than it is for character 1-5 grams. For character 1-2 grams, the difference between the highest and lowest f1 score is 1.01, compared to 0.08 for character 1-5 grams. The smaller the range in cross-validation during training, the better the results on the blind development set.

Feature	Macro F1 10-fold CV	Macro F1 Development
Majority Class Baseline	-	39.83
Character 1-2 grams	64.24	65.98
Character 1-3 grams	68.73	74.46
Character 1-4 grams	73.04	75.49
Character 1-5 grams	73.80	**76.26**
Character 1-6 grams	70.27	39.83
Best N-grams + emoji	73.27	39.76
Best N-grams + polarity	71.94	54.82

Table 3: Classification results for the coarse-grained task: Macro F1 10-fold cross validation on the training data and macro F1 on the development set.

The same classifier retrained on the fine-grained labels does not perform as well as it does on the binary classification task. Table 4 shows that the best result on the development set was an F1 of 47.17. Unlike in the coarse-grained task, lengthening the n-gram sequence does not increase performance, because the F1 decreases quite drastically after character trigrams.

The performance for the individual classes for the best feature combination, character 1-3 grams, is shown in table 5. The classifier performs best on

Features	Macro F1 10-fold CV	Macro F1 Development
Majority Class Baseline		19.92
Character 1-2 grams	33.27	40.10
Character 1-3 grams	39.70	**47.17**
Character 1-4 grams	40.09	23.85
Best N-grams + emoji	44.37	23.54
Best N-grams + polarity	42.15	23.33

Table 4: Classification results for the fine-grained task: Macro F1 10-fold cross validation on the training data and macro F1 on the development set.

the abuse subclass, which is the strongest form of hate speech, and the worst on the profanity class. This indicates that the classifier is sensitive to the most offensive hate speech, rather than the less offensive Tweets in the profanity class.

Class	**Precision**	**Recall**	**Macro F1**
Profanity	0.00	0.00	0.00
Insult	81.82	15.25	25.71
Abuse	70.27	50.49	58.76
Other	75.90	95.17	84.45

Table 5: Classification results for the fine-grained task: Macro F1 10-fold cross validation on the training data and macro F1 on the development set.

4.2 Final results

The following six files were submitted for evaluation:

1. fkieITF_coarse_1.txt — character 1-3 grams
2. fkieITF_coarse_2.txt — character 1-4 grams
3. fkieITF_coarse_3.txt — character 1-5 grams
4. fkieITF_fine_1.txt — character 1-3 grams
5. fkieITF_fine_2.txt — character 1-3 grams
6. fkieITF_fine_3.txt — character 1-3 grams

The final models were trained and tested on the full training and test set provided by the task organizers, not the reduced set that was used to achieve the results described above. For the coarse grained task, three separate classifiers were trained with character 1-3, 1-4 and 1-5 grams as features. For the fine-grained task, the three classifiers were trained using character 1-3 grams only. The parameters were the same as described in section 3 for all classifiers, except the random state parameter was not fixed for the three fine-grained classifiers. All data was pre-processed as described in section 3.3.

5 Discussion

The results for the fine-grained classification indicate that differentiating between different types of hate speech is more difficult than just detecting whether hate speech is present in a Tweet. While the basic character n-gram features perform reasonably well on the binary classification task, other features tailored specifically to differentiating between different types of hate speech are needed for the fine-grained task. For example, Tweet (2) is annotated and correctly identified as "abuse", but Tweet (3) is annotated as "profanity", and is incorrectly predicted to be "other". Both Tweets are a similar length, contain swear words, and attribute a negative quality to the subject of the Tweet. A human annotator knows that one Tweet is much more harmful than the other, but it is not so easy to define features to distinguish between the two.

(2) Wer die Grünen wählt ist entweder dumm oder ein Hurensohn

(3) Juhu, das morgige Wetter passt zum Tag SCHEIßWETTER

Another challenge stems from the fact that it is difficult for human annotators to consistently annotate hate speech. The Tweets in the examples below are both annotated as "abuse", the strongest type of hate speech in the corpus. While the Tweet in example (4) can be clearly identified as being abusive to a large group of people, the Tweet in example (5) is much less extreme, and an argument can be made that this is not the same type of hate speech, or perhaps not even hate speech at all. Better definitions of what hate speech exactly is would make it easier to automatically distinguish between different types.

(4) @diMGiulia1 Araber haben schon ekelhafte Fressen....!

(5) @BILD Warum lese ich nix ber Abgaswerten von ausländischen Autos. Werden diese Daten uns unterschlagen? Kann mir beim besten Willen nicht vorstellen, dass nur bei deutschen Automotoren geschummelt wurde!!!!!!

An issue that may also contribute is the fact that the subclasses within "offense" are relatively small.

In particular, "profanity" only has a total of 64 training instances and it occurs only 7 times in the development set. This leads to issues with the fine-grained classifiers, as Tweets are often not predicted to be "profanity" at all.

For the final results of the classifiers, there is an increase in both training and test data. There is a possibility that some of the challenges discussed above have been solved by the increase in data. Based on the learning rate provided in figure 1 below, a slight increase in the results could be expected. However, as discussed in the previous section, the results may fluctuate depending on the instances in the training and test data.

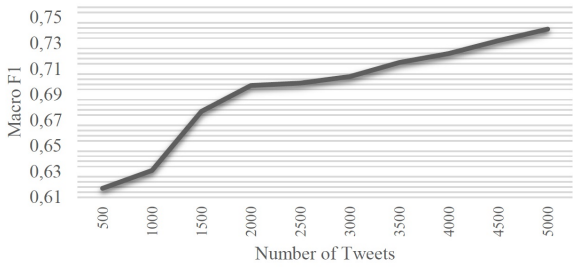


Figure 1: Learning rate for a coarse-grained classifier with character 1-5 grams as features.

6 Conclusion

In this paper we presented a logistic regression classifier based on character n-grams to automatically identify hate speech in Tweets. We participated in both tasks set out at GermEval 2018. We reported a macro F1 of 76.72 and 47.17 for the coarse-grained task and fine-grained task, respectively, after testing the classifiers on the self-created development set. Unsurprisingly, the binary classification task was simpler than trying to determine the degree of severity of hate speech in a Tweet. Nevertheless, in both cases, the task still remains challenging and highlights the fact that defining and annotating hate speech can indeed be problematic (Ross et al., 2016). A comprehensive error analysis would provide insight into how to differentiate between these types of hate speech and allow us to understand how to design features specifically tailored to the task at hand.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017*, pages 759–760.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017.*, pages 512–515.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting hate speech in social media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017, Varna, Bulgaria, September 2 - 8, 2017*, pages 467–472.

Chikashi Nobata, Joel R. 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, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 145–153.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2016. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *CNLP4CMC III: 3rd Workshop on Natural Language Processing for Computer-Mediated Communication, Bochum, Germany*.

Ulli Waltinger. 2010. Germanpolarityclues: A lexical resource for german sentiment analysis. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the Student Research Workshop, SRW@HLT-NAACL 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, 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, NLP+CSS@EMNLP 2016, Austin, TX, USA, November 5, 2016*, pages 138–142.

CNN-Based Offensive Language Detection

Jian Xi[⋆] and **Michael Spranger**

University of Applied Sciences Mittweida
Forensic Science Investigation Lab (FoSIL)
Mittweida, Germany
{xi, spranger}@hs-mittweida.de

Dirk Labudde

Fraunhofer SIT
Cyber Security
Darmstadt, Germany
labudde@hs-mittweida.de

Abstract

Sentiment analysis of short social media texts is a challenging task due to limited contextual information and noise in texts. We present a deep convolutional model that utilizes unsupervised pre-trained word embeddings to detect offensive texts. Unfortunately, the model cannot outperform the baseline model in task-1 of the Germeval Task 2018 in terms of the F_1-measure.

1 Introduction

Sentiment Analysis (SA) is a subtask in Text Classification (TC) that focuses on the contextual mining of texts that are related to some specific objects. SA has great potential for several different applications. For instance, for a recommender system it is critical to know the interests of the customers. Furthermore, SA is also useful to find out the public opinion concerning highly sensitive political topics, as was the case in the study by Ross et al. (2016), in which Twitter texts were used to detect hate speech in the European refugee crisis. Usually, SA includes methods from different disciplines such as natural language processing (NLP) and machine learning (ML) (Pang et al., 2002).

The detection of offensive language in the Germeval Task 2018 is a typical task in SA. The submitted models should be able to categorize tweets into offensive or neutral for task-1 and into more fine-grained categories, namely neutral, profanity, insult and abuse, in task-2. Both, basic features and deep learning features, were used and combined with a classical ML model and a deep model in order to find out how the best result for the task can be achieved.

The paper is organized as follows: in Section II the architecture for the task is presented. Section III details the experimental setup and results. Finally, Section IV gives a short conclusion and discusses future work.

2 Model Description

The deep learning model shows remarkable performance in SA tasks as was shown by Nogueira dos Santos and Gatti (2014) as well as in NLP sequential text generation (Sutskever et al., 2011). The former study used a Convolution Neural Network (CNN) that uses convolution filters to extract local features in order to classify texts. In the latter study, a Recurrent Neural Network (RNN) captures the dependencies of data in a time-sequential way. In our case, we used a CNN model due to its performance in NLP tasks.

2.1 Architecture

Our model is a variation of the CNN by Kim (2014) as depicted in Figure 1. For the model, two channels were used for static and non-static representations of inputs with word embeddings (Mikolov et al., 2013). After maximizing the *feature map* with a max pooling operator as was presented by Kim (2014) a dense layer was added and its output entered into a second convolution layer

$$\mathbf{c}_s = f(\mathbf{w} \cdot \max\{\mathbf{c}\} + \mathbf{b}), \tag{1}$$

where $\mathbf{c}$ is the *feature map*, $\mathbf{w}$ and $\mathbf{b}$ the weights connected to the dense layer. It was found that, without this structure, the results are even worse. The output of the second convolution layer is concatenated and used as the input for the last dense layers. The final predicted sentiment label is output by a softmax layer.

2.2 Network Training

In our task let $T = t_1, ..., t_m$ be a set of texts to be categorized, and $c = c_1, ..., c_n$ a set of sentiment classes, then the task of categorizing can be described as a surjective mapping $f : T \to C$, where $f(t) = c \in C$ yields the correct class for $t \in T$. Given a text, the model calculates a score for each sentiment class $c \in C$. The network is hence trained

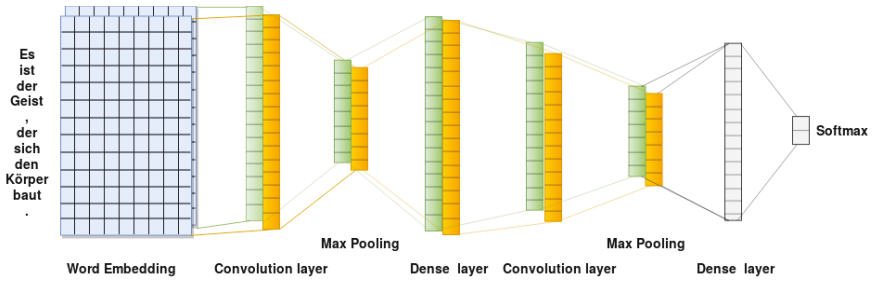


Figure 1: The architecture of the model with two input channels.

by minimizing the negative likelihood for the training set T defined in Equation 2.

$$\log L(c|t,\Theta) = \sum_{i=1}^{m} p(c|t_i,\Theta) - \log \sum_{j=1}^{n} e^{s_\Theta(t_i)_{c_j}} \quad (2)$$

For each input text t_i, the sentiment score $s_\Theta(t_i)_c$ for the sentiment label c is calculated by the network with the parameter Θ. The probability of a sentiment class c_k given the input t_i is the proportion of the sentiment class c over all sentiment classes $c_j \in C, j = 1,...,n$ and is calculated as shown in Equation 3.

$$p(c_k|t_i,\Theta) = \frac{e^{s_\Theta(t_i)_{c_k}}}{\sum_{j=1}^{n} e^{s_\Theta(t_i)_{c_j}}} \quad (3)$$

To predict a sentiment class it has to be determined which Θ maximizes the probability for a certain class as is shown in Equation 4.

$$\tilde{c} = \arg\max_{\Theta} p(c|t_i,\Theta) \quad (4)$$

In order to solve this optimization task ADADELTA, as proposed by Zeiler (2012), was applied.

2.3 Regularization

In order to regularize the parameters the L_2 norm was used in the convolution layers and a batch normalization (Ioffe and Szegedy, 2015) in the dense layers. The training does not stop until the validation accuracy does not improve any further within 25 epochs.

3 Experimental Setup and Results

The tasks are implemented with NLTK (Loper and Bird, 2002), Keras (Chollet, 2017), Scikit-learn (Pedregosa et al., 2011) and TreeTagger (Schmid, 1995). For task-1 four machine learning approaches were used: Naïve Bayes, SVM, a Multi-layer Perceptron (MLP) and our deep model. The basic models give a base-line performance for task-1. Afterwards, the deep model was build to upgrade the results for both tasks. All models are evaluated with respects of precision, recall and F_1-measure. Before the setup is explained in more detail, the features used are briefly introduced.

3.1 Feature Selection

In text classification tasks the selection of features is a critical step. On the one hand, well selected features are necessary to achieve highly accurate results. On the other hand, they help to reduce the feature space and as a consequence to minimize the time complexity (Yang and Pedersen, 1997).

Basic Features: Before the selection of features, all stop-words, repeated words and the punctuation were removed. Wang and Castanon (2015) showed that emoticons help in sentiment analysis tasks, however, this was not taken into account in our classification. The following three representations of text documents incorporating different features were compared:

- bag of words (BoW),
- TF-IDF of the BoW,
- Word $n-$grams (bi- and trigrams)

We also tried to select the top most common k $n-$grams to serve as a dictionary. However, due to an almost uniform distribution of $n-$grams in the corpus, this approach gives less informative feature representations.

Deep Learning Features: In order to use the similar contextual semantic of words, we used unsupervised pre-trained word embeddings (Mikolov et al., 2013) from the following resources:

- German twitter data between 2013 and 2017, with 100 dimensions and window size 5 provided by Ruppenhofer (2018),

- German Wikipedia and news articles, with 300 dimensions and window size 5 from Müller (2015)

3.2 Setup

Features: Table 1 shows the abbreviations for the features considered in both classification tasks.

Abbrv.	Feature
RAW	only raw texts
RAW*	with replacement of mention and hash tag
STM	BoW after stemming
LEM	BoW after lemmatizing
TFI	TF-IDF of BoW
STF	TF-IDF of BoW after stemming
LTF	TF-IDF of BoW after lemmatizing
BIG	word bigrams after stemming
TRG	word trigrams after stemming
MIG	mixture of BIG and TRG

Table 1: Features considered in the classification tasks.

In order to evaluate the fitting of the models for our data, a 10-fold cross validation was used. In each cross step, models with different features were evaluated regarding precision, recall and f-measure. After the best accuracy was achieved the most appropriate features and model was selected. The results will be given in 3.3.

Models: For the three basic models the default parameter settings from NLTK were used. In order to select the best version for the deep model, the following model variations were tested:

- Random: the word embeddings are initialized randomly and learned during training,
- Static: the word embeddings are initialized with previously pre-trained word embeddings and not changed during training,
- Non-static: one channel is set as static and the other as non-static. The static channel gives a basic word representation in the semantic space, while the other channel is adjusted during the learning process, so it can give a plausible representation of words in the given context.

3.3 Results

The results for the 10-fold cross-validation of three basic machine learning models for task-1 with different features are given in Table 2. As can be seen, unigram features lead to less information in the classification, while trigrams give the best precision results. Since the sequential and contextual information between words are encoded in trigrams, it enables a model to classify offensive texts better. Of all three basic models, the Naïve Bayes using BoW and stemmed texts performs best in terms of the F_1 measure.

Model	Feature	P	R	F_1
Naïve Bayes	RAW	0.542	**0.789**	0.623
	RAW*	0.536	0.756	0.627
	STM	0.556	0.784	**0.651**
	LEM	0.558	0.779	0.650
	BIG	0.570	0.225	0.323
	TRG	0.775	0.018	0.036
	MIG	0.565	0.222	0.319
MLP	RAW	0.654	0.473	0.549
	RAW*	0.651	0.439	0.524
	STM	0.661	0.493	0.565
	LEM	0.669	0.495	0.569
	TFI	0.629	0.511	0.564
	STF	0.626	0.509	0.561
	LTF	0.638	0.490	0.554
	BIG	0.748	0.069	0.126
	TRG	0.875	0.012	0.025
	MIG	0.836	0.033	0.064
SVM	TFI	0.663	0.513	0.579
	STF	0.677	0.524	0.591
	LTF	0.680	0.523	0.591
	BIG	0.777	0.056	0.104
	TRG	**0.917**	0.007	0.013
	MIG	0.857	0.025	0.048

Table 2: Evaluation results of the basic models for task-1.

Additionally, Table 3 shows stems of words that often occur in offensive twitter texts. They were selected by their informativeness which is based on the prior probability that features occur for each label. These may be useful in a later approach in order to set up a knowledge base.

Table 4 shows the best results for our deep model for task-1, achieved using word embeddings pre-trained on Twitter data, as suggested by Rezaeinia et al. (2017). The model performs best with a static

Stem	Informativeness
murksel	21.68
scheiss	19.09
pack	17.95
idiot	17.34
wand	14.20
deutschfeind	12.09
entsorgt	10.07
gehirn	8.31
hitl	7.18
altmai	6.65

Table 3: The 10 most informative features detected by the Naïve Bayes model.

Class	P	R	F_1
OTHER	0.778	0.918	0.840
OFFENSIVE	0.754	0.470	0.572

Table 4: Evaluation results of the CNN model for task-1.

initialization. However, the Naïve Bayes model performs better in this task. One possible explanation for the poor performance of our model is the lack in sufficient training data. For example Kim's (2014) training data set was on average of double the size. Another possible explanation is that the quality of the pre-trained word embeddings is not sufficient. As we have seen the word embeddings include a lot of noise. Subsequently, three runs of the static deep model using Twitter word embeddings were submitted as:

- `FoSIL_coarse_1.txt`,
- `FoSIL_coarse_2.txt`, and
- `FoSIL_coarse_3.txt`.

4 Conclusions and Future Work

In this paper we used basic ML methods and a deep CNN model in order to classify texts into different categories regarding offensive language. The results show that the Naïve Bayes model performs better in task-1 in comparison to our proposed CNN model. The reasons might be the small amount of training data as well as the poor quality of the provided word embeddings. Tai et al. (2015) showed that sequential models perform best in sentiment analysis tasks, which is why these models should be further tested. However, also further features should be considered. For instance, in order to distinguish texts including profanity from those, that include abuse and insults, it would be useful to take Part-of-Speech (POS) into account as Rezaeinia et al. (2017) suggest to use POS and word embeddings to improve classification accuracy. As emoticons occur in both, neutral texts and offensive texts, it should be analyzed how they might influence the classification results. Furthermore, Nogueira dos Santos and Gatti (2014) used word-level embeddings as well character-level embeddings to catch morphological information in order to classify short texts.

References

Francois Chollet. 2017. *Deep learning with python*. Manning Publications Co.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Andreas Mueller. 2015. German Wikipedia Embeddings. URL: `https://devmount.github.io/GermanWordEmbeddings/` [accessed: 2018-08-09].

Cicero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78. Dublin City University and Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November.

Seyed Mahdi Rezaeinia, Ali Ghodsi, and Rouhollah Rahmani. 2017. Improving the accuracy of pretrained word embeddings for sentiment analysis. *CoRR*, abs/1711.08609.

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 Michael Beißwenger, Michael Wojatzki, and Torsten Zesch, editors, *Proceedings of NLP4CMC III: 3rd Workshop on Natural Language Processing for Computer-Mediated Communication*, pages 6–9.

Josef Ruppenhofer. 2018. German Twitter Embeddings. URL: `http://www.cl.uni-heidelberg.de/english/research/downloads/resource_pages/GermanTwitterEmbeddings/GermanTwitterEmbeddings_data.shtml` [accessed: 2018-08-09].

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *In Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 1017–1024, USA. Omnipress.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075.

H. Wang and J. A. Castanon. 2015. Sentiment expression via emoticons on social media. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2404–2408, Oct.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701.

spMMMP at GermEval 2018 Shared Task: Classification of Offensive Content in Tweets using Convolutional Neural Networks and Gated Recurrent Units

Dirk von Grünigen*
Fernando Benites
Pius von Däniken
Mark Cieliebak
Zurich University of Applied Sciences (ZHAW)
CH-8400 Winterthur
dirk@vongruenigen.com
{benf,vode,ciel}@zhaw.ch

Ralf Grubenmann*
SpinningBytes AG
Albanistrasse 20
CH-8400 Winterthur
rg@spinningbytes.com

Abstract

In this paper, we propose two different systems for classifying offensive language in micro-blog messages from Twitter ("tweet"). The first system uses an ensemble of convolutional neural networks (CNN), whose outputs are then fed to a meta-classifier for the final prediction. The second system uses a combination of a CNN and a gated recurrent unit (GRU) together with a transfer-learning approach based on pretraining with a large, automatically translated dataset.

1 Introduction

Sentiment Analysis was a major focus for text analytics in the last few years. Recently it became clear that only differentiating between positive and negative opinions is insufficient for some practical applications. Nowadays many website maintainers are requested to remove offensive content and monitor the discussions on their websites and social networks. This creates an overwhelming need for automated classification and removal of posts which could cause legal issues.

Although there are resources and research on some languages, e.g. English (Davidson et al., 2017; Waseem and Hovy, 2016), most languages have little or no resources on the matter. The GermEval Shared Task 2018 aims to tackle the problem of offensive language within micro-blog posts from Twitter ("tweets") written in German.

In this report, we propose two classifiers for identifying offensive content in tweets. Our experiments show that using embeddings created from large amounts of unsupervised in-domain data has a beneficial impact on the results. We rely on state-of-the-art convolutional neural networks (CNNs) and ensemble strategies, which have shown to achieve competitive results on sentiment analysis (e.g. Deriu et al. (2016)).

2 Task Description

The organizers of the shared task provided a dataset with 5009 samples. Each sample contains a tweet and two types of labels, one for each sub-task: The first label is for the binary-classification task ("Task I") and hence only distinguishes between *offensive* and *non-offensive* content. The second label discriminates between four different classes, of which 3 are different types of offensive content: *abuse*, *insult* and *profanity* and the fourth label for *non-offensive*. The second subtask is very unbalanced, with the labels distributed as: 3321 non-offensive, 1022 abusive, 595 insult and 71 profanity.

3 System Descriptions

In the following two sections, we describe our two proposed systems. System I is built on an ensemble of convolutional neural networks (CNN) whose outputs are consumed by a meta-classifier for the final prediction. This system is optimized to work as a classifier for the binary classification task ("Task I"). System II is based on the CNN+GRU architecture proposed by Zhang and Luo (2018). An important component of both systems is the use of diversified and enriched word embeddings to grasp the semantic context of the words. Both approaches are cutting-edge for specific but related text classification tasks and are therefore well suited to the problem domain, although they have not been di-

*Equal Contribution

rectly compared to date.

4 System I

Deep learning models based on convolutional neural networks (CNN) are state-of-the-art for a number of text classification tasks, in particular in sentiment analysis (Kim, 2014; Kalchbrenner et al., 2014; Severyn and Moschitti, 2015a; Severyn and Moschitti, 2015b; Johnson and Zang, 2015), which is closely related to the domain of detecting offensive content in text. The system proposed by Mahata et al. (2018) has proven to perform exceptionally well in the domain of classifying medication intake from tweets. Based on this, we also trained multiple shallow CNNs and combine them into an ensemble in a similar fashion.

4.1 Preprocessing

The data is processed by lowercasing the tweet and normalizing numbers and removing "|LBR|" tokens, which signify a newline in a tweet. Depending on the embeddings used further down the process, as detailed in Section 4.3, we used different tokenization strategies. For vanilla word2vec and fastText embeddings, we used the NLTK `TweetTokenizer` (Bird et al., 2009). On the other hand, for the subword byte-pair embeddings (Sennrich et al., 2016), we used the Google `sentencepiece`[1] tool.

As the last step, we applied the hashtag splitting procedure described below to split up hashtags into their distinctive parts, since hashtags can convey a lot of the intention of a tweet. Finally, we converted the tokenized tweets into a list of indices, which was used to select the corresponding word embeddings. Furthermore, we enriched the word-embeddings with word-based polarity values.

Word Polarity Values: In offensive texts in tweets, often very polarising words are used (e.g. racial slurs or insults). To take advantage of this fact, we incorporated polarity values for each word in the used dataset. For that purpose, we employed three different resources: A multi-domain sentiment lexicon for German from the IGGSA website[2], the list of insults in German from the website hyperhero.com[3] and a list of racial slurs in German from the website hatebase.org[4]. The polarity values in the lexicon range from -1.0 (negatively polarising) to +1.0 (positively polarising). The average of all polarity values provided for each word in the lexicon provided to the system an additional feature. This sentiment lexicon was extended with the words from the list of German insults from the website hyperhero.com and from the list of racial slurs from hatebase.org to it. Further, we assigned a negative polarity (i.e. -1.0) value to these additional words. We then generated a one-hot encoded vector with 11 polarity-classes for each word in the dataset by discretizing the continuous polarity values. These vectors were stacked on top of each of the word embedding vectors before being passed to the convolutional network.

Hashtag Splitting: Hashtags are problematic in tweets, since sometimes they are composed of multiple words (e.g. "#ThisIsASingleHashtag") and hence would be out-of-vocabulary for the word embeddings most of the time. But they are crucial to understand the real meaning behind a tweet: For example the meaning of a tweet with the hashtag "#sarcasm" might be understood completely different without adding this hashtag. To tackle this problem, we implemented a hashtag splitting procedure using the CharCompound[5] tool (Tuggener, 2016). It is a simple but elegant solution, which uses ngram probabilities and returns different splits for each word with a certainty value for each split. We applied the splitting procedure recursively to the hashtags to ensure that we split all compounds. We set the certainty threshold to 0.8 and stopped when no split with a certainty greater or equal to this threshold could be found.

4.2 Base CNN

The base CNN for the ensemble consists of multiple, shallow convolutional layers. Each convolutional layer consists of the following components, in the listed order:

- Word embeddings layer that converts an indices-vector into a sentence-matrix.
- Dropout layer (Srivastava et al., 2014) as a regularization measure.
- Convolution operation for the feature extraction.
- Batch normalization layer (Ioffe and Szegedy, 2015) to speed up the training.

[1] https://github.com/google/sentencepiece
[2] https://sites.google.com/site/iggsahome/downloads
[3] http://hyperhero.com/de/insults.htm
[4] https://www.hatebase.org/search_results
[5] https://github.com/dtuggener/CharSplit

Hyperparameter	*Value*
Number of Conv. Kernel	200
Conv. Kernel Sizes	[2, 3, 4, 5, 6]
Conv. Kernel Stride	1
Conv Kernel Dilation	0
Number of Neurons in Hidden Layer	4096
Dropout Probability (after word-embeddings layer)	0.4
Dropout probability (after conv. operation)	0.3
Dropout probability (between fully-connected layers)	0.4
Max. Input Length	200

Table 1: Hyperparameters used for the base CNN in System I. Only one kernel size was used per convolutional operation, but we used 5 convolutional layers, each using one of the sizes for its kernels.

- Another dropout layer.
- Max-pooling layer to reduce the dimensionality of the output.
- ReLU activation function (Nair and Hinton, 2010) to squeeze the output values into the range $[0, +\infty)$.

In total there are five of these layers, all using the same hyperparameters (see Table 1), except for the kernel size in the convolution operation. The sentence-matrix is fed to each of these parallel convolutional layers and the resulting output vectors are concatenated, resulting in a vector with 1000 values. This vector is then forward propagated through two fully connected layers, which then output two logit values for the two classes (i.e. "not offensive" and "offensive"). A visualization of the base CNN model is depicted in Figure 1.

Hyperparameters: The hyperparameters used in the base CNN of System I can be seen in table 1. The max-pooling operation was performed as global max-pooling. This implies that each of the convolution operations outputs 200 distinct values, because we configured each convolution operation to use 200 different kernels. As a result of using 5 different convolutional layers having 200 output values each, the vector, which is forwarded to the fully-connect layer, contains 1000 values.

Initialization and Optimization of Parameters: All parameters, except for the biases, of the base CNN were initialized using the `Xavier Normal` initialization (Glorot and Bengio, 2010) with the gain value set to 1. The biases were initialized to 0. We used the `Adam` optimizer (Kingma and Ba, 2014) for the optimization of the network parameters, including the word embeddings. `Adam` dynamically adapts the learning rate for every parameter in the network by using first- and second-order information. We used a learning rate of 0.001, 0.9 and 0.999 as the beta coefficients for computing the running averages of the gradients, a weight decay value of 0.0005 and an epsilon value of 10^{-8}. As the loss function, we employed the cross-entropy loss between the expected, one-hot encoded label vector and the output of the CNN after being passed through a Softmax layer.

4.3 Word Embeddings

Word embeddings are omnipresent today when performing any natural language processing, especially with deep learning models. Due to our approach of using several of the previously described base CNNs, we decided that we would initialize each of these with another kind of word embeddings. We use different kind of word embeddings to get an diversified view of the data, which helps with our ensembling approach.

The following types of word embeddings were used:

- `fastText (SpinningBytes-FT)` embeddings (Bojanowski et al., 2017; Joulin et al., 2017) with 300 dimensions trained on a large corpus of German tweets ("sb-tweets") provided by SpinningBytes[6]. These are currently not publicly available.
- `fastText (fasttext-Wiki)` embeddings with 200 dimensions pretrained on the texts from the German Wikipedia corpus. These can be downloaded via the fastText GitHub page[7].
- `word2vec (SpinningBytes-W2V)` (Mikolov et al., 2013) embeddings with 200 dimensions, also trained with the "sb-tweets" corpus. These can also be downloaded from the SpinningBytes website.
- `fastText Byte-Pair Embeddings (Spinningbytes-BP)` embeddings with 100 dimensions for the case where subword tokenization (Sennrich et al., 2016) was performed, trained with the "sb-tweets" corpus. For the tokenization, we used the previously mentioned Google `sentencepiece` tool. These embeddings are not publicly available at the moment.

[6]http://spinningbytes.com
[7]https://github.com/facebookresearch/fastText/

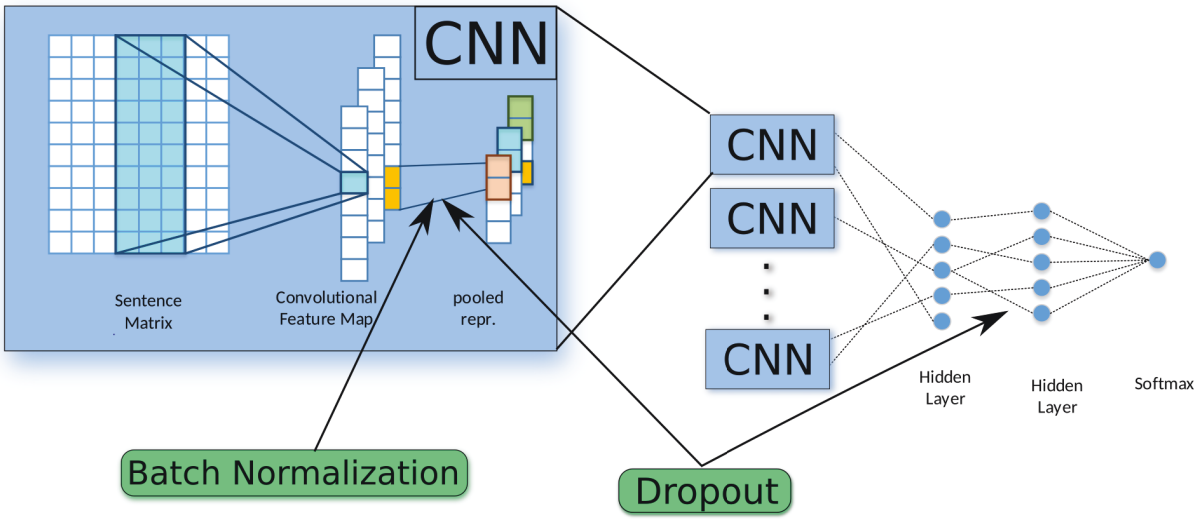


Figure 1: Visualization on the structure of the base CNN model.

4.4 Training Procedure and Ensembling of Classifiers

We decided to train our models in a similar fashion as Mahata et al. (2018): First, we split the data provided by the organizers randomly into a training and holdout dataset, where the training dataset contains 90% of the provided data and the other 10% is used as for the holdout dataset. We train each of the different models by doing k-fold cross-validation (with $k = 5$) over said training data and use the evaluation dataset for performing early stopping if the performance on it did not improve for more than 20 epochs with respect to the macro F1-score. Each of the models trained on each fold is then stored for later usage in the ensemble. This results in 20 base CNNs in total, 5 for each of the 4 different CNNs initialized with the word embeddings listed in Section 4.3.

Class Weights: Only 33.7% of the samples in the provided data contain offensive content, whereas 66.3% do not. We used class weights to counter this imbalance in the label distribution. For this we computed class weights, which are then used to rescale the loss function when performing the back-propagation. The following formulae were employed:

$$C_{\mathrm{O}} = \frac{|L_{\mathrm{N}}| + |L_{\mathrm{O}}|}{2 \cdot |L_{\mathrm{O}}|} \quad (1)$$

$$C_{\mathrm{N}} = \frac{|L_{\mathrm{N}}| + |L_{\mathrm{O}}|}{2 \cdot |L_{\mathrm{N}}|} \quad (2)$$

where $|L_{\mathrm{O}}|$ is the number of offensive samples, $|L_{\mathrm{N}}|$ the number of samples with non-offensive content in the provided dataset. C_{O} and C_{N} are the resulting class weights for offensive and non-offensive samples respectively.

4.5 Meta Classifiers

As described before, we trained the same base CNN with different word embeddings on different parts of the training data using k-fold cross-validation. Moreover, we concatenated the outputs of these 20 models on the training dataset and used them in conjunction with the labels to train different meta-classifiers. We experimented with different strategies for meta-classification (see Table 3 in Section 6) and used hyper-parameter optimization while training them.

5 System II

Following Zhang and Luo (2018), our second architecture utilizes both CNN and Gated Recurrent Units (GRU, Cho et al. (2014)). It uses three different embeddings and an attention layer, which are described in detail in the following.

5.1 Preprocessing

Additionally to the preprocessing of System I, user mentions (*@username*) were removed, words containing dots were split and special characters */ |: ; & * were removed. German stopwords[8] were also removed from the input string. Words not present in the embeddings were replaced with an **UNK** token.

[8] https://github.com/stopwords-iso/stopwords-de

5.2 CNN + GRU

The model consists of two CNN+GRU architectures, one for word-embeddings and one for subword embeddings, which are later concatenated together, along with a Smiley-feature vector, before being used by a fully connected Softmax layer to get predictions of the model. To prevent overfitting, dropout of 0.5 was added before every convolutional as well as the final layer. ReLU was used as activation function for all convolutional layers. An overview of the architecture is shown in Figure 2.

Word embeddings architecture: fastText embeddings of 200 dimensions each for uni- and bigrams in a tweet are concatenated to get a 100x400 feature matrix. Tweets are limited 100 tokens. 1d convolutions with 100 feature maps and kernel sizes of 3, 4 and 5, and kernel sizes 2 and 3 with dilations of 2 and 3, respectively, are then applied to the feature matrix separately. The dilated convolutions are meant to simulate the *skipped CNN* proposed in (Zhang and Luo, 2018). The results are max-pooled by a factor of 4 and concatenated along the feature axis. This is then passed to a bi-directional GRU unit. The hidden states at each time step of the GRU are then combined by an attention layer (Xu et al., 2014), yielding a feature vector containing 1000 values.

Subword embeddings architecture: This architecture largely mirrors the word embeddings architecture, but takes subword tokenized embeddings as input. Due to the smaller nature of subword tokens, a maximum sentence length of 150 is enforced. The architecture is adjusted to yield the same 1000 dimensional feature vector as in the word-embeddings architecture.

Emoji embeddings: A list of 751 Unicode emojis (Kralj et al., 2015) is used to count the occurrences of different emojis in the tweets. A linear transformation is applied to the emoji feature vector to reduce dimensionality to 200.

Final layer: The output of all three parts of the architecture is concatenated to yield a 2200 dimensional feature vector. A fully connected layer with Softmax is used to get the final output of the architecture, with 2 and 4 dimensions for the coarse and fine tasks, respectively.

5.3 Transfer Learning

Due to the relatively small amount of training data, the model was pretrained on a related task. To our knowledge, only one other hate speech corpus in German is available (Ross et al., 2016). But there are two large corpora for hate speech detection available in English, namely (Davidson et al., 2017) and one provided by Lukovnikov[9]. To get as close as possible to the target domain, the English hate speech corpora were automatically translated[10] to German. The model was jointly trained on the related German and English corpora until train scores stopped improving. Then the last layer of the network was discarded and retrained on the actual data provided for the Shared Task.

5.4 Semi-Supervised Retraining using the Test Dataset

To extend the training set, we used a similar semi-supervised approach to Jauhiainen (2018). For that purpose, our system is first trained on the training dataset and then used to classify the test dataset. Predictions on samples of the unlabeled test dataset with a confidence higher than 0.75 are then used as additional labeled data to augment the training set. We treat the output of the Softmax layer as the confidence score. The classifier is then trained again on the augmented training dataset. The results can be seen in Tables 2 and 3 for the systems labeled with *Semi*.

6 Experiments

We performed several tests on the labeled training data. As described above, we randomly selected 10% of the training data as test data. We then trained on the training data and evaluated the systems on the test data. This procedure was repeated five times in order to estimate an average and standard deviation of the performance.

We compared our results to a baseline which consisted of an SVM using TF-IDF feature weighting. The data preprocessing was performed by tokenizing the tweets with the mentioned `TweetTokenizer` and the `GermanStemmer` from the `stem.snowball` module of `NLTK`. We also compared the single classifiers of System I versus the results using different meta classifiers. We evaluated the results with the F-1 macro average measure. The results are depicted in Table 3.

In task I, the meta classifiers had a remarkable impact. Logit Averaging provided an advantage over the other approaches and improved the overall classification performance by more than 3 points

[9] https://github.com/lukovnikov/hatespeech

[10] https://translate.google.com/

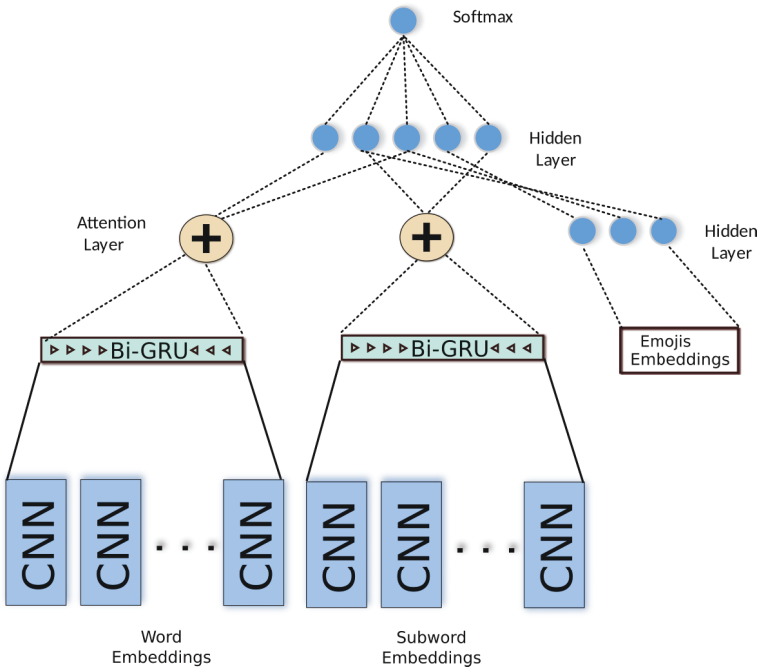


Figure 2: Visualization on the structure of the CNN + GRU model.

with respect to the F-1 macro score in comparison to the best performing single classifier (see Table 3). This confirms the results of Mahata et al. (2018). Other meta-classifiers, such as Random Forests, Logistic Regression and Linear SVM were close, though the single classifiers were also in this range. The System I results showed that the embeddings can have a decisive impact on the results of the classification systems. These systems had a big margin to the Multilayer Perceptron meta classifier, which performed last in the results and also has the largest variance in the performance. The SVM baseline performed worse comparing to the other single classifier approaches.

Using the semi-supervised routine can make a decisive difference on the performance, as can be seen from the System II results. Especially for task II, we see that the semi-supervised approach was 4 points better. Interestingly, the baseline SVM performed best in this task.

7 Submitted Runs

7.1 For Task I

The following runs were submitted to the GermEval organizers for Task I:

- **spMMMP_coarse_1**: System I, best model out of 15 runs.
- **spMMMP_coarse_2**: System I, second-best model out of 15 runs.
- **spMMMP_coarse_3**: System II with semi-supervised augmented training data, best model out of 5 training runs.

System II	*F-1 macro*
SpinningBytes-CNN+GRU	0.4100 ± 0.0363
SpinningBytes-CNN+GRU Semi	0.4549 ± 0.0324
SVM	**0.4797 ± 0.0346**

Table 2: Results for the CNN+GRU classifier on task 2. All reported scores are the performance on the holdout dataset from each specific run, measured in F1-score (macro) over the `OFFENSIVE`, `ABUSIVE`, `INSULTING` and `OTHER` labels for the 4-class classification task.

7.2 For Task II

The following runs were submitted to the GermEval organizers for Task II:

- **spMMMP_fine_1**: System II without semi-supervised augmented training data, best model out of 5 training runs.
- **spMMMP_fine_2**: System II with semi-supervised augmented training data, best model out of 5 training runs.

System	*F-1 macro*
SVM	0.7266 ±0.0212
System I	
Single Classifiers	
SpinningBytes-FT CNN	0.7547 ± 0.0160
SpinningBytes-W2V CNN	0.7656 ± 0.0143
fastText-Wiki CNN	**0.7703 ± 0.0102**
SpinningBytes-BP CNN	0.7354 ± 0.0188
Meta Classifiers	
Random Forest	0.7843 ± 0.0188
Majority Vote	0.6813 ± 0.0329
Logit Averaging	**0.8048 ± 0.0138**
One Trigger	0.6304 ± 0.0223
Logistic Regression	0.7762 ± 0.0308
Linear SVM	0.7686 ± 0.0334
Multilayer Perceptron	0.6638 ± 0.1299
System II	
SpinningBytes-CNN+GRU	0.7454 ± 0.0168
SpinningBytes-CNN+GRU Semi	0.7684 ± 0.0087

Table 3: Classification results on the task I training data. All reported scores are the performance measures in F1-score (macro) over 5 randomly different tests on the holdout set.

- **spMMMP_fine_3**: SVM with TF-IDF and semi-supervised augmented training data.

8 Conclusion

In this paper, we described our two different approaches to tackling the problem of detecting offensive content in micro-blog posts from Twitter in the context of the GermEval 2018 Competition.

The first system used an ensemble of the same CNN base model initialized with different types word embeddings. These models are then used in combination with an output-averaging approach to generate the final prediction. A preliminary evaluation of the system showed that it achieves an average F1-score (macro) of 80% on average on randomly chosen holdout datasets on the binary classification task.

The second system used a combination of a CNN and GRU architecture with two different type of word embeddings. The preliminary evaluation on a randomly chosen holdout set showed that it could achieve a performance of 45% with respect to the macro-averaged F1-score over all four labels from the multi-label classification task.

References

Bird Steven, Loper Edward and Klein Edward. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Bojanowski Piotr, Grave Edouard, Joulin Armand and Mikolov Tomas. 2017. *Enriching Word Vectors with Subword Information*. *Transactions of the Association for Computational Linguistics*. Association for Computational Linguistics.

Cho Kyunghyun, Van Merriënboer Bart, Gulcehre Caglar, Bahdanau Dzmitry, Bougares Fethi, Schwenk Holger, Bengio Yoshua. 2014. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. *arXiv preprint arXiv:1406.1078*. arXiv.org.

Davidson Thomas, Warmsley Dana, Macy Michael, Weber Ingmar. 2017. *Automated Hate Speech Detection and the Problem of Offensive Language*. *Proceedings of the 11th International AAAI Conference on Web and Social Media*. ICWSM.

Deriu Jan, Gonzenbach Maurice, Uzdilli Fatih, Lucchi Aurelien, De Luca Valeria and Jaggi Martin. 2016. *SwissCheese at SemEval-2016 Task 4: Sentiment Classification using an Ensemble of Convolutional neural networks with distant supervision*. *Proceedings of the 10th International Workshop on Semantic Evaluation*, 1124–1128. Association for Computational Linguistics.

Deriu Jan, Lucchi Aurelien, De Luca Valeria, Severyn Aliaksei, Müller Simon, Cieliebak Mark, Hoffmann Thomas and Jaggi Martin. 2017. *Leveraging Large Amounts of Weakly Supervised Data for Multi-Language Sentiment Classification*. *Proceedings of the 26th International Conference on World Wide Web*, pages 1045–1052. International World Wide Web Conferences Steering Committee.

Glorot Xavier and Bengio Yoshua. 2010. *Understanding the Difficulty of Training Deep Feedforward Neural Networks*. *Proceedings of the thirteenth international conference on artificial intelligence and statistics* 2010, pages 249–256. ACM.

Ioffe Sergey and Szegedy Christian. 2015. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. *Proceedings of the 32nd International Conference on International Conference on Machine Learning* volume 37. JMLR.org.

Jauhiainen Tommi, Linden Krister and Jauhiainen Heidi. 2018. *HeLI-based Experiments in Swiss Germ Dialect Identification* (in Press). *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*.

Johnson Rie and Zhang Tong. 2015. *Semi-Supervised Convolutional Neural Networks for Text Categorization via Region Embedding*. *NIPS 2015 - Advances in Neural Information Processing Systems*. Association for Computational Linguistics.

Joulin Armand, Grave Edouard, Bojanowski Piotr and Mikolov, Tomas. 2017. *Bag of Tricks for Efficient Text Classification*. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* volume 2, short papers. Association for Computational Linguistics.

Kalchbrenner Nal, Grefenstette Edward and Blunsom Phil. 2014. *A Convolutional Neural Network for Modelling Sentences*. *ACL - Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Kim Yoon. 2014. *Convolutional Neural Networks for Sentence Classification*. *EMNLP 2014 - Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Kingma Diederik and Ba Jimmy. 2014. *Adam: A Method for Stochastic Optimization*. *arXiv prepr.int arXiv:1412.6980*. arXiv.org.

Kralj Novak Petra, Smailović Jasmina, Sluban Borut, Mozetič, Igor. 2015. *Sentiment of emojis*. *PLoS ONE* Volume 10. PLoS ONE

Mahata Debanjan, Friedrichs Jasper, Shah Rajiv Ratn and Hitkul. 2018. *# phramacovigilance-Exploring Deep Learning Techniques for Identifying Mentions of Medication Intake from Twitter*. *arXiv preprint arXiv:1805.06375*. arXiv.

Mikolov Tomas, Sutskever Ilya, Chen Kai, Corrado Greg and Dean Jeff. 2013. *Distributed Representations of Words and Phrases and their Compositionality*. *NIPS 2013 - Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Nair Vinod and Hinton Geoffrey E. 2010. *Rectified linear units improve restricted boltzmann machines*. *Proceedings of the 27th international conference on machine learning (ICML-10)*. Omnipress.

Ross Björn, Rist Michael, Carbonell Guillermo, Cabrera Benjamin, Kurowsky Nils, Wojatzki Michael. 2016. *Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis*. *Proceedings of NLP4CMC III: 3rd Workshop on Natural Language Processing for Computer-Mediated Communication*. Bochumer Linguistische Arbeitsberichte.

Sennrich Rico, Haddow Barry and Birch Alexandra. 2016. *Neural Machine Translation of Rare Words with Subword Units*. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* volume 2, long papers. Association for Computational Linguistics.

Severyn Aliaksei and Moschitti Alessandro. 2015. *Twitter Sentiment Analysis with Deep Convolutional Neural Networks*. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Severyn Aliaksei and Moschitti Alessandro. 2015. *UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification*. *SemEval 2015 - Proceedings of the 9th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya and Salakhutdinov Ruslan. 2014. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. *Journal of Machine Learning Research* volume 15. JMLR.org.

Tuggener Don. 2016. *Incremental Coreference Resolution for German*. *PhD Thesis*. University of Zurich.

Tuggener Don. 2018. *Evaluating Neural Sequence Models for Splitting (Swiss) German Compounds* (in press). *Proceedings of the 3rd Swiss Text Analytics Conference - SwissText 2018*. ceur-ws.org.

Waseem Zeerak and Hovy Dirk, 2016. *Hateful symbols or hateful people? predictive features for hate speech detection on twitter*. *Proceedings of the NAACL student research workshop* 2016, pages 88–93. NAACL.

Xu Kelvin, Ba Jimmy, Kiros Ryan, Cho Kyunghyun, Courville Aaron, Salakhudinov Ruslan, Zemel Rich, Bengio Yoshua. 2014. *Show, attend and tell: Neural image caption generation with visual attention*. *International conference on machine learning* 2015, pages 2048–2057. icml.cc.

Zhang Ziqi, Luo Lei. 2018. *Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter*. *arXiv preprint arXiv:1803.03662* arXiv.org.

GermEval 2018 : Machine Learning and Neural Network Approaches for Offensive Language Identification

Pruthwik Mishra
IIIT-Hyderabad
Hyderabad, Telangana
India - 500032
pruthwik.mishra@
research.iiit.ac.in

Vandan Mujadia
i.am+ LLC.
Bangalore, Karnataka
India - 560071
vandan.mujadia@
iamplus.com

Soujanya Lanka
i.am+ LLC.
Mapex Building
Singapore - 577177
soujanya@
iamplus.com

Abstract

Social media has been an effective carrier of information from the day of its inception. People worldwide are able to interact and communicate freely without much of a hassle due to the wide reach of the social media. Though the advantages of this mode of communication are many, the severe drawbacks can not be ignored. One such instance is the rampant use of offensive language in the form of hurtful, derogatory or obscene comments. There is a greater need to employ checks on social media websites to curb the menace of the offensive languages. GermEval Task 2018 [1] is an initiative in this direction to automatically identify offensive language in German Twitter posts.

In this paper, we describe our approaches for different subtasks in the GermEval Task 2018. Two different kinds of approaches - machine learning and neural network approaches were explored for these subtasks. We observed that character n-grams in Support Vector Machine (SVM) approaches outperformed their neural network counterparts most of the times. The machine learning approaches used TF-IDF features for character n-grams and the neural networks made use of the word embeddings. We submitted the outputs of three runs, all using **SVM** - one run for Task 1 and two for Task 2.

1 Introduction

Automatic identification of offensive language in social media micro posts has become paramount in order to tackle dangerous phenomena like cyber bullying, trolling, hateful comments related to ethnicity and gender discrimination. The GermEval 2018 shared task is an attempt to automatically identify offensive language with a training dataset containing around 5000 twitter posts. This shared task is divided into two subtasks.

- Task 1 - it is a binary classification task where the tweet is either predicted as Offensive or Other or Non-Offensive.
- Task 2 - it is a fine grained classification on offensive tweets - either Profanity, Insult or Abuse, non offensive tweets are classified as Other similar to task 1

We used different machine learning and neural network approaches for both the tasks which are explained in the subsequent sections. The paper is organized as follows: Section 2 lists down the related work and Section 3 describes our approach. Section 4 presents the experiments and results on the development set. Section 5 discusses about the evaluation metric and Section 6 details about the error analysis. Section 7 concludes the paper with possible future work.

2 Related Work

There has been a renewed interest among academia and industry recently for identification of online offensive posts due to their unabated use mostly related to racism and sexism. Most of the previous works have been done with respect to hate speech detection in English. According to Waseem and Hovy (2016), a logistic regression classifier with character n-grams performed the best for detecting hateful speeches in English twitter data. They used various extra-linguistic features like gender, word length and location information. Schmidt and Wiegand (2017) outlined the approaches for hate speech detection ranging from surface level features like character n-grams, linguistic features

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

like POS Tags, typed dependency labels, word clustering, to customized lists comprising of hateful utterances, hate verbs, and meta information like gender, location, history of a user. Most of the approaches in Schmidt and Wiegand (2017) used support vector machines (Cortes and Vapnik, 1995) for achieving optimal results. Microsoft used Smokey (Spertus, 1997) for identifying abusive content in their commercial applications. Smokey was implemented using a C 4.5 decision tree classifier (Quinlan, 1986) for flame or abuse identification. Nobata et al. (2016) used a regression model taking into account word and character n-grams, syntactic features, and word embeddings to detect abusive language in online comments found from Yahoo! finance and news.

There has been very few attempts on offensive language detection in social media content in German. So this shared task is the first such initiative in this regard. The released data was annotated by human annotators following the guidelines prepared by the organizers [2].

3 Approach

Most of the machine learning (ML) algorithms are heavily reliant on hand crafted features designed by experts. This makes ML algorithms less generalizable and cost inefficient. So we did not use any language specific features like part-of-speech tags, morph features, dependency labels etc. for any of the tasks. We used publicly available sentiment lexicons as the only external resource. We describe our approaches in the following subsections.

3.1 Machine Learning Approaches

Three different machine learning algorithms were implemented for Task 1 and 2.

1. Linear SVM (Cortes and Vapnik, 1995)
2. Stochastic Gradient Descent (Bottou, 2010)
3. Logistic regression

In the above ML algorithms, we used TF-IDF (Sparck Jones, 1972) vectors for the character n-grams present in the training corpus. Two different feature sets were passed to the ML algorithms to create models for prediction of Task 1 and Task 2. The first feature set only contained a bag-of-words representation with TF-IDF vectors. The second set made use of the publicly available sentiment datasets German Polarity clues (Waltinger, 2010) and German slur dictionary [3] along with the TF-IDF representations. From the German Polarity clues repository, we created lists belonging to individual sentiments. Three sentiment labels were used namely positive, negative and neutral. The list specific features were computed as the counts of words pertaining to a specific sentiment and the number of slur words present in the post. All these features were appended to the TF-IDF vector representation of a post. We also tried a variation for the fine-grained classification task where the predicted output from task 1 was also added as a feature to the TF-IDF and list specific features.

3.2 Neural Networks

Two neural network architectures, bi-directional LSTM(bi-LSTM) (Graves and Schmidhuber, 2005), multi-layer perceptron (MLP) (Sparck Jones, 1972) were used. These neural network models used word embeddings as features. Publicly available German Word2vec embeddings [4] were used for these experiments. For encoding a sequence, bidirectional LSTM uses both past and future contextual information. We used word embeddings of size 200 trained on 200 million tweets. We have experimented only with a single hidden layer, with the number of hidden units same as the word embedding dimension for the MLP architecture. For the bidirectional LSTM model, we only use word embeddings as features. The maximum length of the sample was set to 50 based on the property of the training data. Each sample is represented as a vector of size $50 * d$ where d =word embedding dimension. In case of MLP, all the vectors present in a sample are concatenated and presented as a single input vector whereas all the vectors are given as a sequence of vectors for a bidirectional LSTM. The representation of a post is the representation learned after passing the whole sequence of tokens through the biLSTM. There is no sequence or ordering information in case of an MLP, so the hidden layer representation is the learned representation of the post. For both tasks, the output layer contained nodes equal to the number of class labels (2 for Task 1 and 4 for Task

[2] http://www.coli.uni-saarland.de/~miwieg/Germeval/guidelines-iggsa-shared.pdf

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

[4] https://www.spinningbytes.com/resources/wordembeddings/

2).

4 Experiment Setup

The corpus details and the model description are explained in the following subsections.

4.1 Corpus Details

The released data contained 5009 twitter posts. The frequency analysis for coarse and fine classes with respect to Task 1 and Task 2 is detailed in tables 1 and 2. We split the data into 9:1 ratio for designing the training and development sets.

Label	No Of Samples
OFFENSIVE	1688
OTHER	3421
Total	**5009**

Table 1: Label Frequencies for Task 1

4.2 Tokenization

As the training data provided for this task was from twitter posts, we tokenized the data as a preprocessing step. We considered punctuations except the @, #, as individual tokens. @ and # are associated with twitter handles and user ids, appeared in some twitter handles, so we did not tinker these punctuations.

4.3 Model Description

The model parameters of both approaches are explained in detail in the subsections below.

4.3.1 Machine Learning Algorithms

The machine learning algorithms described in 3.1 were implemented using sklearn library (Pedregosa et al., 2011). sklearn uses count vectorizers to convert a text input into a sparse collection of tokens. It provides the flexibility of including higher order n-grams in the vocabulary. The SVM algorithm used linear kernel with penalty parameter of 1.0 for error term and tolerance level of 0.001 for classification. For the logistic regression classifier (Fan et al., 2008), the parameters were set to L2 regularization, tolerance level of 0.0001, and penalty parameter of 1.0. The SGD classifier was implemented using hinge loss with L2 regularization.

Label	**No Of Samples**
ABUSE	1022
INSULT	545
OTHER	3321
PROFANITY	71
Total	**5009**

Table 2: Label Frequencies for Task 2

Type	No Of Samples
Train	4505
Dev	504
Total	**5009**

Table 3: Corpus Split

4.3.2 Neural Networks

We implemented two neural network models, bi-directional LSTM(bi-LSTM) (Graves and Schmidhuber, 2005), multi-layer perceptron (MLP) (Sparck Jones, 1972) using the framework designed by Chollet et al. (2015). The accuracies of these models are reported in the subsequent sections. The word embedding for a word was arrived at by concatenating the pre-trained word vector and the learned representation of the word from the training data. The word representation is learned using an embedding layer. The size of the vector representation for a word has been fixed at 200. We have experimented only with a single hidden layer, with 200 hidden units for the MLP architecture. The activation of each hidden node was chosen to be tanh. The maximum length of a post has been fixed at 50. Each sample was represented as a vector of size $200 * 50 = 10000$ for bi-LSTM and MLP experiments when only word vectors were used as features. For a post containing more than 50 words, only the first 50 words were used and rest all are ignored. When the number of tokens in a post is less than 50 words, it has to be padded with zero vectors. The Adam (Kingma and Ba, 2014) optimizer was used with categorical cross entropy as the loss function, batch size 16 and 10 epochs. The activation at the output layer was "softmax" for both the tasks.

4.4 Experimental Results on Development Set

The results of the experiments on the development set are detailed in this section. The features explained in table 6 were used in SVM for the final submission.

As the linear SVM was the best performing sys-

Model	Features	Category	Precision	Recall	F1-Score
Bi-LSTM	Word Embeddings	OFFENSE	65.27	63.74	64.50
		OTHER	81.60	82.58	82.09
		AVERAGE	73.44	**73.16**	73.30
MLP	Word Embeddings	OFFENSE	64.23	51.46	67.14
		OTHER	77.38	85.29	81.14
		AVERAGE	70.81	68.37	69.57
Linear SVM	TF-IDF	OFFENSE	80.00	52.07	63.08
		OTHER	79.44	93.43	85.87
		AVERAGE	**79.72**	72.75	**76.08**
SGD	TF-IDF	OFFENSE	61.54	14.20	23.08
		OTHER	68.82	95.52	80.00
		AVERAGE	65.18	54.86	59.58
Log Reg	TF-IDF	OFFENSE	79.71	32.54	46.22
		OTHER	73.79	95.82	83.38
		AVERAGE	76.75	64.18	69.91

Table 4: Task 1 Results on Dev-Set for All Models

Model	Features	Category	Precision	Recall	F1-Score
Bi-LSTM	Word Embeddings	ABUSE	13.59	13.59	13.59
		INSULT	40.82	33.33	36.70
		OTHER	66.95	69.97	68.43
		PROFANITY	0.00	0.00	0.00
		AVERAGE	30.34	29.22	29.77
MLP	Word Embeddings	ABUSE	9.09	6.80	7.78
		INSULT	45.83	18.33	26.19
		OTHER	65.51	79.28	71.74
		PROFANITY	0.00	0.00	0.00
		AVERAGE	30.11	26.10	27.96
Linear SVM	TF-IDF	ABUSE	74.00	35.92	48.37
		INSULT	66.67	10.17	17.65
		OTHER	73.26	97.31	83.59
		PROFANITY	0.00	0.00	0.00
		AVERAGE	53.48	**35.85**	**42.93**
SGD	TF-IDF	ABUSE	62.96	16.50	26.15
		INSULT	25.00	37.29	29.93
		OTHER	72.24	83.88	77.62
		PROFANITY	0.00	0.00	0.00
		AVERAGE	40.05	34.42	37.02
Log Reg	TF-IDF	ABUSE	75.00	23.30	35.56
		INSULT	100.00	5.08	9.68
		OTHER	69.94	97.91	81.59
		PROFANITY	0.00	0.00	0.00
		AVERAGE	**61.23**	31.57	41.66

Table 5: Task 2 Results on Dev-Set for All Models

Features	Category	Precision	Recall	F1-Score
TF-IDF+Sentiment Lexicon+Slur Dictionary	OFFENSE	76.99	51.48	61.70
	OTHER	79.03	92.24	85.12
	AVERAGE	78.01	71.86	74.81
	ABUSE	78.85	39.81	52.90
	INSULT	66.67	10.17	17.65
	OTHER	74.04	97.91	84.32
	PROFANITY	0.00	0.00	0.00
	AVERAGE	**54.89**	36.97	44.18
TF-IDF +Sentiment Lexicon +Slur Dictionary +Task1 Predictions	ABUSE	59.55	51.46	55.21
	INSULT	66.67	27.12	38.55
	OTHER	79.03	92.24	85.12
	PROFANITY	0.00	0.00	0.00
	AVERAGE	51.31	**42.70**	**46.61**

Table 6: Task 2 Results on Dev-Set for All Models

tem in basic TF-IDF vectors of character 2-6 grams, the variation of using a sentiment lexicon was implemented only with SVM. The results are shown in table 6.

5 Evaluation

Evaluation code was provided by the organizers [5]. The evaluation metric included the overall accuracy, class or category wise precision, recall, and F1-scores for both the tasks. The average precision, recall and F1-scores for all the categories are also returned after running the evaluation tool.

5.1 Observation

From the tables 4 and 5, we can see that SVM consistently outperforms all other models. The Bi-LSTM model outperforms MLP by a significant margin while linear SVM comfortably beats SGD and logistic regression. All the machine learning models outperform the neural models. It was quite intuitive that the use of sentiment lexicon and slur dictionary improved the results. All the models failed to predict samples with 'PROFANITY' labels owing to fewer class samples in the training data.

5.2 Submitted Runs

For the GermEval Shared Task, we submitted three outputs using linear SVM classifier. The features used for the submissions are explained below in detail.

- We used TF-IDF features along with the counts of sentiment words using the German polarity clues lexicon, and count of slur words using the slur dictionary appearing in posts for coarse_1 and fine_1.
- The output predicted for Task 1 was appended to the features explained above for fine_2.

6 Error Analysis

The neural network models did not perform well because of the limitation of pre-trained word embeddings. The words present in the pre-trained word2vec model were not normalized, so different spelling variations of a single word can have multiple representations. Overall the dataset was imbalanced, many classes had fewer samples to create robust models. The higher number of samples from the 'OTHER' class created a lot of false positives.

7 Conclusion & Future Work

In this paper, we describe our systems for both subtasks. We showed that linear SVM with character n-grams outperformed all other machine learning and neural network models.

It is intuitive that specific words in a text influence its overall classification. We can explore attention mechanism with bi-LSTM models to model this context vector in a better way. We plan to use character embeddings for out-of-vocabulary (OOV) words. This might help us improve the overall models and make it robust to spelling variations of words. Some normalization tools can be added as a preprocessing step for twitter text which will help us learning better word representations.

[5] https://projects.fzai.h-da.de/iggsa/evaluation-tool/

References

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Corinna Cortes and Vladimir Vapnik. 1995. Support vector machine. *Machine learning*, 20(3):273–297.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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. International World Wide Web Conferences Steering Committee.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011.

Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Ellen Spertus. 1997. Smokey: Automatic recognition of hostile messages. In *AAAI/IAAI*, pages 1058–1065.

Ulli Waltinger. 2010. Germanpolarityclues: A lexical resource for german sentiment analysis. In *LREC*.

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.