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 The final evaluation is done by the organizers of the task who compare our predicted results with the unpublished ground truth.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

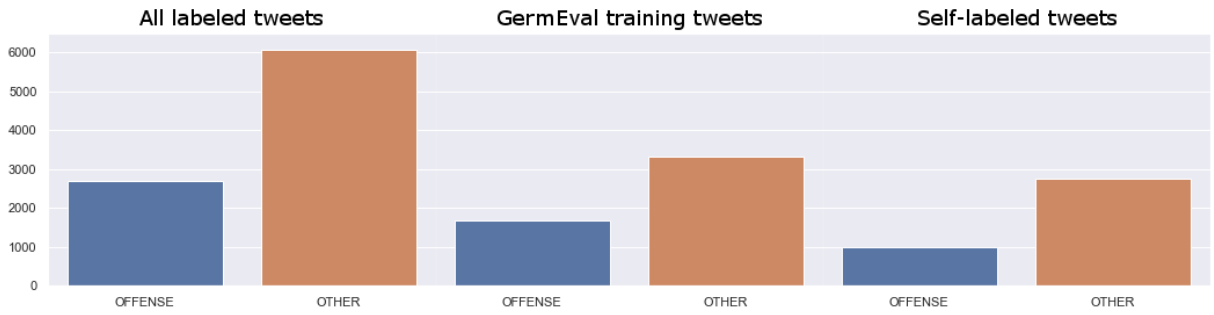

Figure 1: Distribution of offensive tweets per data set

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

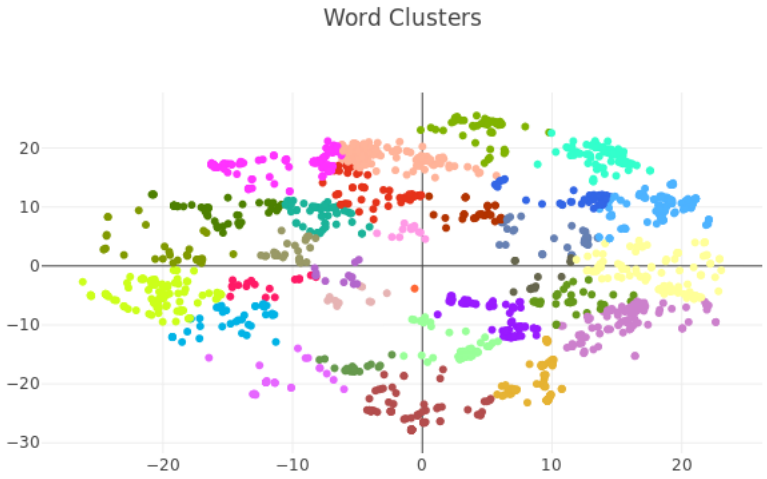

Figure 2: 30 word clusters with k-means

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.

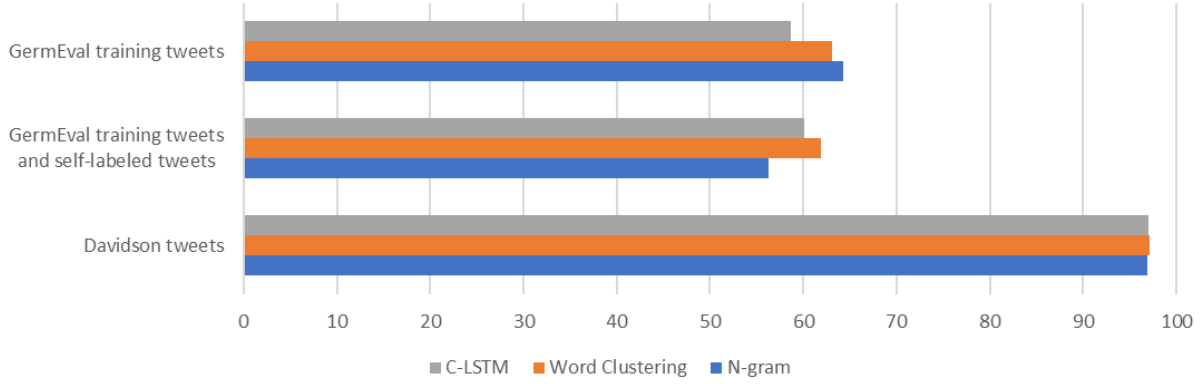

Figure 3: Results of different systems per data set

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

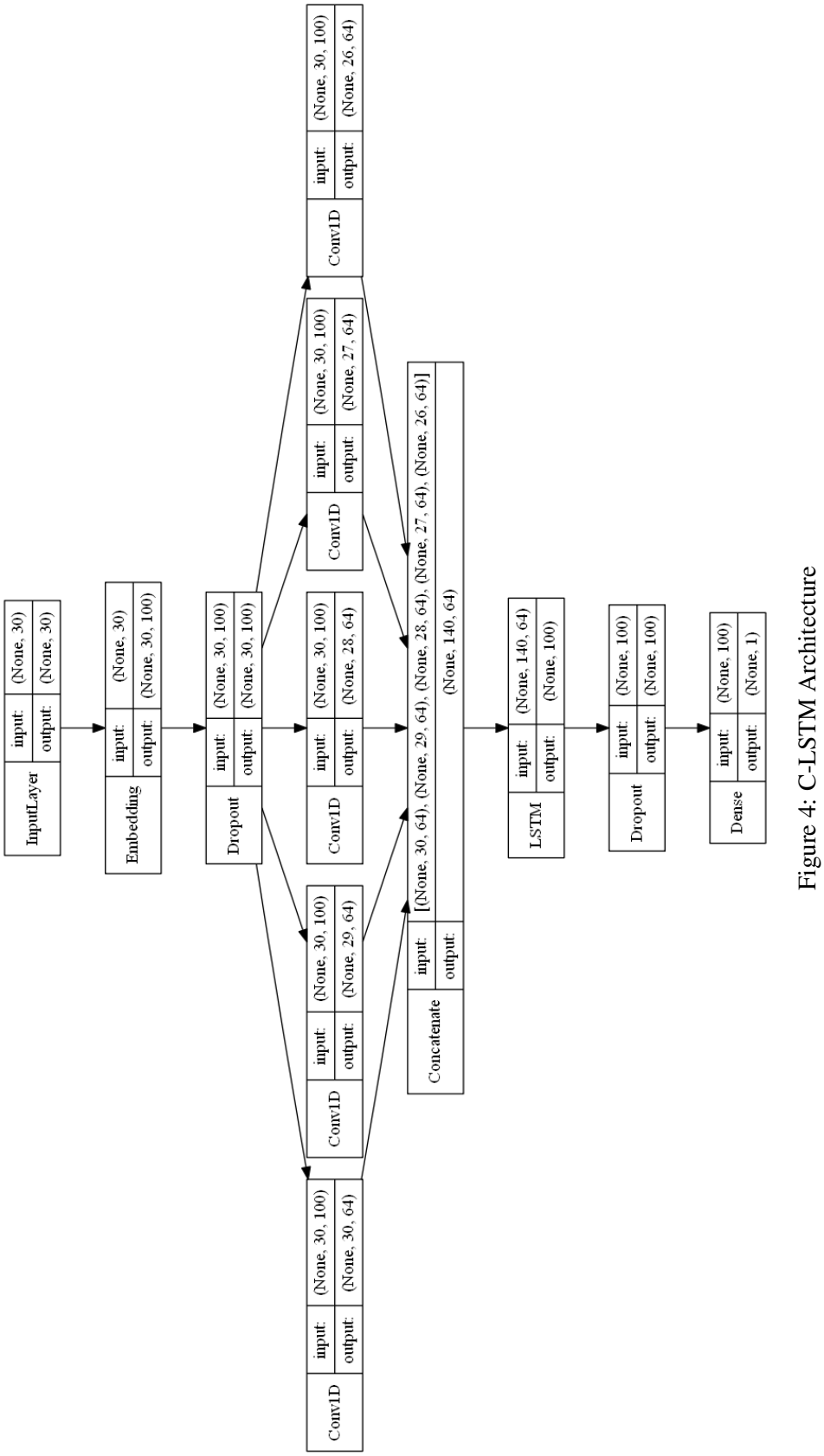

Figure 4: C-LSTM Architecture