HHU at SemEval-2019 Task 6: Context Does Matter - Tackling Offensive Language Identification and Categorization with ELMo

Alexander Oberstrass[1] **Julia Romberg**[1] **Anke Stoll**[2] **Sefan Conrad**[1]

[1]Institute of Computer Science, Heinrich Heine University Düsseldorf, Germany
alexander.oberstrass@hhu.de
{romberg,conrad}@cs.uni-duesseldorf.de

[2]Department of Social Sciences, Heinrich Heine University Düsseldorf, Germany
anke.stoll@hhu.de

Abstract

We present our results for OffensEval: Identifying and Categorizing Offensive Language in Social Media (SemEval 2019 - Task 6). Our results show that context embeddings are important features for the three different subtasks in connection with classical machine and with deep learning. Our best model reached place 3 of 75 in sub-task B with a macro F_1 of 0.719. Our approaches for sub-task A and C perform less well but could also deliver promising results.

1 Introduction

User generated content in social media platforms such as Twitter often includes high levels of rude, offensive, or sometimes even hateful language. The increasing vulgarity in online discussions and user comment sections have recently been discussed as relevant issues in society as well as in science.

The identification of offensiveness, aggression, and hate speech in user-generated content has been addressed in recent research (Waseem et al., 2017; Davidson et al., 2017a; Malmasi and Zampieri, 2018) and previous shared tasks (Wiegand et al., 2018; Kumar et al., 2018). However, detecting such content automatically is still challenging. We developed classification models to identify *offensive language*, different *categories of offense types*, and *targets of offensive language* throughout the SemEval-2019 challenge on Identifying and Categorizing Offensive Language in Social Media (Zampieri et al., 2019b).

2 Methodology and Data

In this section, we introduce the datasets, the features and classifiers we use.

2.1 Datasets

The training dataset provided for this task is further described in Zampieri et al. (2019a). For sub-task A: Offensive Language Detection 4.400 offensive (*OFF*) and 8.840 not offensive (*NOT*) tweets are given for the training and 240 offensive plus 620 not offensive tweets are given as test data. Sub-task B: Categorization of Offensive Language is provided with a training set of 3876 targeted insult (*TIN*) and 524 untargeted (*UNT*) tweets and a test set of 213 targeted insult plus 27 untargeted tweets. The train data distribution for sub-task C: Offensive Language Target Identification are 2407 tweets targeting an individual (*IND*), 1074 targeting a group (*GRP*) and 395 targeting any other category (*OTH*). The test data given counts 100, 78 and 35, respectively.

We use Davidson et al. (2017b)'s dataset for hate speech detection as an additional source of knowledge and to address the problem of overfitting in our deep learning models. Tweets were collected and filtered using a lexicon of common hate speech terms. The remaining tweets were then each classified by at least 3 CrowdFlower users into the three categories *hate speech*, *offensive language* or *none*. Each tweet was assigned a label, which was chosen based on a majority vote. In order to match the labels to this task, we merged the categories: *hate speech* and *offensive language* into a single offensive label *OFF* and renamed the label *none* to *NOT*. This resulted in 20620 *OFF* and 4163 *NOT* labeled additional tweets. It should be noted that our assumption to equate hate speech and offensive language does not apply in general.

2.2 Features

Word and Char-Level Features We transformed the tweet texts into tf-idf weighted bag of words and bag of chars representations using scikit-learn

(Pedregosa et al., 2011). The feature *CHARS* is built from bigrams to 6-grams on character level. For the feature *WORDS* we use unigrams and bigrams on word level, excluding stop words. In order to reduce variance in the text data, we also built the feature *STEMS* with unigrams and bigrams. For this, we applied the TweetTokenizer from NLTK (Bird et al., 2009) that removes user name handles from a tweet and replaces repeated characters with a sequence of only 3. Afterwards, the word tokens are reduced to their stems using the Snowball Stemming algorithm (Porter et al., 2002).

Named Entities Named entities in a tweet can indicate whether this tweet is person-related or relates to e.g. religious or political groups or organizations. This information might help with the target identification. The feature *NE* is implemented using the named entity recognition of spacy[1], building a bag of named entities for every tweet preserving all OntoNotes5 (Weischedel et al., 2013) named enity types, which results in an 18-dimensional feature vector per tweet.

Grammatical Number of Nouns and Pronouns The grammatical number of a noun might also indicate if a tweet insults a specifiable target. We use spacy's part-of-speech tagger, which uses the OntoNotes5 tagging scheme. The feature *ND* models the noun distribution of singular and plural nouns and is implemented building a bag of tags for the tags NNS, NN, NNP and NNPS, normalized by the tweets' token count. Contrarily, feature *SPNR* describes the ratio of singular noun tags (NN and NNP) and plural noun tags (NNPS and NNS) per tweet. The first value is divided by the latter one, smoothing both by adding 1. Another feature to separate single person targets from groups is the absolute count of single pronoun words (he, she, him, her, his, hers, himself, herself) per tweet, hereinafter referred to as *SPC*. It should be noted that the corresponding procedure for plural pronouns could not bring any benefit in our evaluations and is therefore not further described.

Dependencies Another consideration was whether syntactic dependency relations could provide information for one of the three sub-tasks. For the feature *DEP*, every tweet is transformed into the corresponding list of dependency labels. These representations are subsequently used to build n-grams in the range (2,4) which are then vectorized using tf-idf.

Global Vectors for Word Representation (GloVe) Feeding words into machine learning models often requires a meaningful vector representation that captures syntactic and semantic information. We use GloVe word embeddings (Pennington et al., 2014), an unsupervised learning algorithm for obtaining vector representations of words. A set of pre-trained word vectors, trained on over 2 billion tweets, containing vector representations for over 1.2 million words in various dimensions (25, 50, 100, 200) is publicly available on the author's website[2]. In this work, we use 200 dimensions.

Embeddings from Language Models (ELMo) Traditional pre-trained word representations merely contain meaning based on statistical information and therefore struggle with word-sense disambiguation. Since offensive words change their meaning greatly depending on their context, a contextualizing method would help to tackle this task. ELMo (Peters et al., 2018) is a novel approach on creating word representations and shows a significant improvement of state of the art systems on many benchmarks. It models a function using character-based word representations and bidirectional long-short term memories of not only each single word, but also the entire input sentence. Thus, it can be useful for solving the problem of processing ambiguous words that are not offensive by themselves but could point to offensive language depending on the context they are used in. For these reasons, we expect ELMo to increase the results of this task. The pre-trained model can be downloaded from TensorFlow Hub[3].

χ^2 Feature Selection To quantify the contribution of a feature, we choose the χ^2 test statistic, that excludes features that are most likely to be independent from a class, and keep the k features with the highest values for χ^2.

2.3 Classifier

We use logistic regression (Nelder and Wedderburn, 1972), support vector machines (Cortes and Vapnik, 1995) and neural networks with long short-term memory units (Hochreiter and Schmidhuber, 1997) for the different classification prob-

[1] https://spacy.io/

[2] https://nlp.stanford.edu/projects/glove/

[3] https://tfhub.dev/google/elmo/2

lems. The classifiers are implemented with scikit-learn and TensorFlow (Abadi et al., 2015).

Logistic regression has been used by several teams of the GermEval shared task on offensive language identification (Wiegand et al., 2018) with promising results. We therefore decided to use it as a baseline approach. In the multiclass case the one-vs-rest scheme is used.

Support vector machines (SVM) are also known to perform well on a variety of classification tasks and have been used in the context of hate and offensive speech and abusive language detection in recent years (Malmasi and Zampieri, 2017; Wiegand et al., 2018). We use the rbf kernel. In the multiclass case the one-vs-one scheme is used.

To overcome class imbalances, class weights are adjusted inversely proportional to the train data class frequencies for both classifiers.

As seen in recent challenges focused on offensive language detection in social media like TRAC (Kumar et al., 2018), long-short term memories (LSTM) play an important role and are often used among the best classifiers. Their ability of sequential data iteration and of memorizing recent content across time provides them with a good opportunity not only to analyze individual words, but also to find problem-specific dependencies between them.

3 Models

This chapter describes the development of the models for the submissions and how they perform on the training data.

3.1 Sub-task A

The training data was split into a training set containing 3400 OFF and 7840 NOT labels and a validation set, which consists of the remaining 1000 OFF and 1000 NOT labels. To establish a baseline for this sub-task, we used the features *CHARS*, *WORDS* and *STEMS*, which were the features that were mainly used by the most powerful systems in the TRAC challenge that did not use deep learning. We opted for logistic regression, because it performed better than the SVM. C was set to the default value 1. Table 1 gives an overview of all model results for sub-task A. The baseline approach, denoted as *Baseline A*, reaches a macro F_1 of 0.712 on the validation set.

Our first deep learning model architecture *LSTM A1* uses ELMo context embeddings as inputs into a bidirectional LSTM layer with a size of 64 cells. For classification, we used the hidden states of the LSTM cells generated when passing the last token from each tweet to the LSTM. A fully connected layer was used to compute the two class scores that were normalized using softmax. Its highest macro F_1 value of 0.742 on the validation set was reached after only two epochs, which means that the network tends to overfit very quickly. To overcome this problem, we placed an additional fully connected layer with a dimensionality of 64 and L2 loss between the high-dimensional ELMo output (1000 dimensions) and the LSTM layer to insert a relatively strong regularization loss to the system. We expected this to force the system to look for more universal patterns in the embeddings before passing them to the LSTM layer. While improving the F_1 score by 1%, this architecture, named *LSTM A2*, still started overfitting after two epochs. As a further adjustment against overfitting we used an extra heavy dropout layer with a 70% dropout between the ELMo embeddings and the fully connected layer. We also tried other dropout rates (30%, 40%, 50%, 60%, 80%, 90%), but 70% worked best. A higher percentage did not yield meaningful results. Through this modification of the architecture, *LSTM A3*, the overfitting was delayed until the 16th epoch. The F_1 score also showed a slight improvement.

Comparing non-contextualized to contextualized word embeddings, the developed architecture was also evaluated with GloVe (*LSTM A4*) as well as with a vector concatenation of GloVe and ELMo Embeddings *LSTM A5*. It turned out that ELMo embedding inputs alone exceed the GloVe embedding inputs in this model and that they work together even better than each one on its own.

System	F_1 (macro)	Accuracy
Baseline A	0.712	0.718
LSTM A1	0.742	0.745
LSTM A2	0.753	0.753
LSTM A3	0.759	0.759
LSTM A4	0.729	0.731
LSTM A5	0.764	0.764
LSTM A6	**0.767**	**0.768**
Feature Union A	0.760	0.760

Table 1: Overview of the performance by different models for sub-task A

Due to the remaining problem of overfitting, we used the additional data described in Section 3.1. Pre-training the network with this additional data and then training it with our training set, a small but no significant improvement was achieved. With a macro F_1 of 0.767, this model (*LSTM A6*) shows the best performance on the validation set and was therefore used for our first submission. We used a learning rate of 0.001 and a batch size of 64 for all models. The number of epochs was chosen dependent on the stagnating progress on the validation set.

We used logistic regression with the default $C = 1$ for the second submission system. As feature representation for each tweet, we combined *CHARS*, *WORDS* and *STEMS* with the output of the already trained LSTM of *LSTM A6* as 128-dimensional fixed features. This system *Feature Union A* could not surpass *LSTM A6*, but achieves nevertheless a F_1 of 0.760. It should also be noted, that the χ^2 feature selection could reduce the features used in this feature union to only four of them without affecting the accuracy of the classification. All of them derived from the LSTM output features. The full architecture of this model, including all the models described earlier in this section, is illustrated in Figure 1.

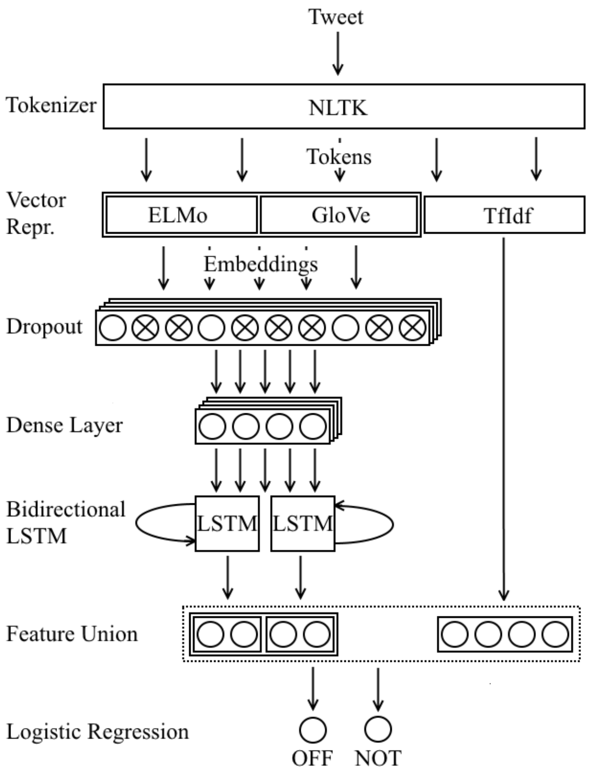

Figure 1: Model architecture for the *Feature Union A* system

3.2 Sub-task B

System	F_1 **(macro)**	**Accuracy**
Baseline B	0.615	**0.869**
LSTM B1	0.602	0.745
LSTM B2	0.628	0.809
Feature Union B	**0.653**	0.825

Table 2: Overview of the performance by different models for sub-task B

In sub-task B we split the data into 80% training and 20% validation set. We trained the same baseline model as in sub-task A, which reached a macro F_1 of 0.615 on the validation set and forms the baseline for the list of all models build for this sub-task shown in Table 2.

For our first deep learning approach on this sub-task, we used the same structure as in *LSTM A6*, but with smaller layer sizes as another way to reduce overfitting, because the training data given for this sub-task was even smaller than for sub-task A. The dense layer between the dropout layer and the bidirectional LSTM consists of 8 and the LSTM of 32 cells. This system named *LSTM B1* achieves 0.623 on the validation set.

Pseudo Labeling was used on the additional data described in Section 3.1 to generate the missing labels for this task. For this, we first labeled the additional data using *LSTM B1*, which had already been trained on the training data. Then the resulting labels are used to extend the training data. In addition, we have focused only on the labels which softmax class score was higher than 0.7 for the predicted label to reduce noisy labels. This results in about 2000 UNT and 17500 TIN labeled tweets. The resulting system named *LSTM B2* reaches a macro F_1 of 0.628 on the validation set.

As first classifier *Feature Union B*, we used the same structure as in sub-task A and combined *CHARS*, *WORDS* and *STEMS* and the output of *LSTM B2* as feature input. The value for C in the logistic regression was selected using grid search and was set to 0.51. Together these features improved the macro F_1 even further and reached our best result 0.653 for sub-task B on the validation set.

For the second submission, we trained additional four models of *LSTM B2* and we used the *Feature Union B* from the first submission as a fifth classifier. We then performed a majority vote on all five resulting labels.

All deep learning models in sub-task B are trained with a learning rate of 0.001 and a batch size of 32 for all models. The number of epochs was defined in the same way as in sub-task A.

3.3 Sub-task C

The evaluation for sub-task C is done on a 80%-20% train-validation split making sure that the test data set contains a 20% proportion of each class. Additionally, the train data was enriched with the trial data, where all occurrences of class ORG were replaced by class OTH. Deep learning was not used for the implementation of this sub-task due to the small amount of training data. To select the best feature sets for the used classifiers, logistic regression and SVM all combinations were tested including grid search to find the best parameter values for C and γ, respectively.

Class	Features	Prec.	Recall	F$_1$
GRP	ELMo	0.566	**0.679**	0.617
	ELMo+NE	0.573	0.674	0.62
	ELMo+ND	0.567	0.61	0.614
	ELMo+NE+ND	0.58	0.674	0.623
	all	**0.582**	0.674	**0.625**
IND	ELMo	0.820	0.789	0.804
	ELMo+NE	0.821	0.795	0.808
	ELMo+ND	0.567	0.67	0.614
	ELMo+NE+ND	0.823	0.795	0.809
	all	**0.825**	**0.798**	**0.811**
OTH	ELMo	0.360	0.237	0.286
	ELMo+NE	0.365	0.250	0.297
	ELMo+ND	0.370	0.263	0.308
	ELMo+NE+ND	0.375	0.276	0.318
	all	**0.386**	**0.29**	**0.331**
macro	ELMo	0.582	0.568	0.57
	ELMo+NE	0.587	0.573	0.575
	ELMo+ND	0.586	0.574	0.575
	ELMo+NE+ND	0.593	0.582	0.584
	all	**0.598**	**0.587**	**0.589**

Table 3: Overview of feature impact on the logistic regression system with $C = 0.011$

Logistic regression classifies best on the validation set using $C = 0.011$ and the combination of the features *WORDS*, *NE*, *SPC*, *DEP*, *SPNR*, *ND* and the ELMo embeddings trained for sub-task B achieving a macro F_1 of 0.589. Table 3 gives an overview of the model performance for the three classes. The features are broken down by influence, where *all* denotes the entire feature set used. ELMo is the most important feature here and gives a macro F_1 of 0.57. The further two percent are largely due to the use of named entities and the noun distribution. The additional features (*WORDS*, *SPC*, *DEP*, *SPNR*) have only little influence of the classification. A closer look reveals that the class performances differ widely depending on class size. Underrepresented classes benefit the most from adding other features to the embeddings, OTH in particular: Precision and recall increase by two and five percent, which leads to a rise in F_1 of about four percent.

Class	Features	Prec.	Recall	F$_1$
GRP	ELMo	0.537	**0.729**	0.618
	ELMo + SPNR	0.554	0.715	0.625
	all	**0.554**	0.715	**0.625**
IND	ELMo	0.871	0.706	0.78
	ELMo + SPNR	0.873	0.706	0.781
	all	**0.874**	**0.708**	**0.782**
OTH	ELMo	0.296	0.342	0.317
	ELMo + SPNR	0.317	0.434	0.367
	all	**0.32**	**0.434**	**0.369**
macro	ELMo	0.568	0.592	0.572
	ELMo+SPNR	0.582	0.618	0.591
	all	**0.583**	**0.619**	**0.592**

Table 4: Overview of feature impact on the SVM system with $C = 6$, $\gamma = 0.0001$

Table 4 depicts the results for the best SVM model, using a feature set of *WORDS*, *DEP*, *SPNR*, *ND* and the ELMo embeddings trained for sub-task B. The best parameter values we found are $C = 6$ and $\gamma = 0.0001$ and lead to a macro F_1 of 0.592 on the validation set. As in the previous setting, ELMo is the key feature and achieves 0.572 macro F_1. The union with feature *SPNR* leads to 0.591 macro F_1. As with logistic regression, the addition has a positive effect on the less common classes.

In addition, we use a majority vote as third submission. For this, we build a voting classifier out of the previously developed logistic regression and SVM classifiers and a further logistic regression classifier which uses *CHARS*, *WORDS* and *STEMS* and the output of *LSTM B2* as feature input. It should be noted, that the LSTM has not been re-trained on the sub-task C data. Subsequently, a majority decision is taken and in case of doubt, the label prediction of the latter logistic regression classifier is given preference.

4 Results

After the previous description of the submission models, an overview of the test results follows in this section.

4.1 Sub-task A

In order to prepare the *LSTM A6* system for submission, we re-trained it with the complete available training data. This classifier reached a macro F_1 of 0.768, as shown in Table 5 on the test set, which is comparable to the result reached on the validation set in Table 1.

As a second submission we chose the *Feature Union A* classifier, but took the output of the LSTM from the first submission for the LSTM output features in the feature union instead. With 0.742, it reached a somewhat lower macro F_1 on the test set than on our validation set as shown in comparison of Table 1 and Table 5.

System	F1 (macro)	Accuracy
All NOT baseline	0.419	0.721
All OFF baseline	0.218	0.279
LSTM A6	**0.768**	**0.807**
Feature Union A	0.742	0.788

Table 5: Test results for sub-task A

4.2 Sub-task B

The *Feature Union B* system reached a macro F_1 score of 0.671. Comparing the results from Table 2 and Table 6, the classifier works equally well on the train and on the test data.

The majority vote classifier achieved our best result with a macro F_1 score of 0.719 (Table 6). This result is remarkably good which we think is caused by a high robustness resulting from the different strengths of the incorporated classifiers.

System	F1 (macro)	Accuracy
All TIN baseline	0.470	0.888
All UNT baseline	0.101	0.113
Feature Union B	0.671	**0.871**
Majority Vote B	**0.719**	0.850

Table 6: Test results for sub-task B

4.3 Sub-task C

On the test data, all systems perform inferior than on the validation set. The best macro F_1 is obtained by the SVM having 0.571, followed by the logistic regression with 0.551 and then the voting classifier with 0.539. Overall, the performance is fairly stable compared to the validation set and the fact that no cross validation was used in the model selection process.

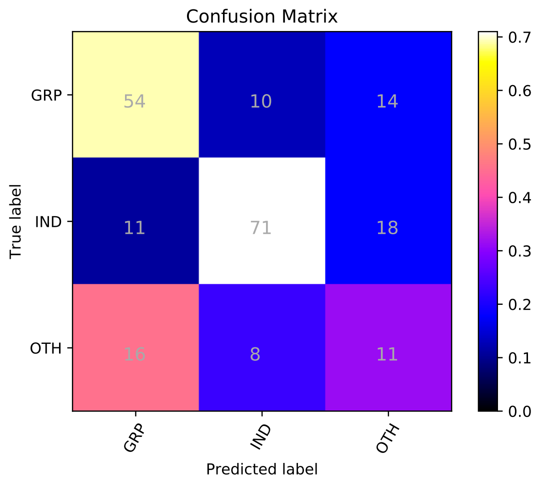

Figure 2: Confusion matrix for sub-task C and the best performing model SVM

Figure 2 shows the confusion matrix for the best performing system. As can be seen, most tweets of GRP and IND are classified correctly, whereas almost half of the OTH tweets are incorrectly recognized as GRP. On the one hand, the rare class occurence in the training seems to obstruct the classifier in learning. On the other hand, it was challenging for us as humans to differentiate between the classes GRP and OTH in the data set: For instance "Liberals are mentally ill!" is labeled as GRP, whereas "@USER republicans/conservatives are the most disgusting people" has label OTH.

5 Conclusion

We showed that contextualized embeddings work well in the context of offensive language identification and the two different categorization tasks. Other language and linguistic features could deliver only small improvements.

The main problems in using deep learning are the model size and the dimensionality and number of parameters resulting in fast overfitting. We plan to address this problem through strategies to reduce model size (primarily in reducing the high-dimensional ELMo output without loosing relevant information before connecting it to the less high-dimensional LSTMs using a fully connected layer), or to include more data when available. Another consideration for sub-task A would be the use of the majority vote model from sub-task B, which showed good results. For sub-task C, the development of deep learning models would be interesting.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1(2).

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017a. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017b. 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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

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

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)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

John Ashworth Nelder and Robert WM Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384.

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.

Jeffrey Pennington, Richard Socher, and Christoper Manning. 2014. Glove: Global vectors for word representation. volume 14, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Martin F Porter, Richard Boulton, and Andrew Macfarlane. 2002. The english (porter2) stemming algorithm. *Retrieved*, 18:2011.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.