Offensive language identification in Twitter using part of speech

Pooja Gowda and Leire Varela
S4410963, S5418658
{p.gowda, l.varela.aranguren}@student.rug.nl

Abstract

In Twitter, there are users writing tweets that incite hatred. In this paper, we create different NLP models that, given a dataset of tweets, classify them as offensive or not. The aim of this experiment is to create the best model possible that learns to determine whether tweets contain insults or other type of offensive words, and to evaluate if part of speech feature tagging helps in improving the SVM classification. The developed code is available in GitHub.[1]

1 Introduction

With the growth of social media, cases of cyberbullying have increased. In Twitter, insults or bad words are not censored, what makes it easy to attack someone from the other side of the screen. This might cause negative effects on the mental health of the person being harassed, especially on the youngest ones. For trying to make things change and protect people from bullying, we want to analyse tweets and, with the help of NLP tools, detect offensiveness in Twitter and check how much there is.

Also, in real-time, the part of the tweets' speech helps the person identify if the tweet is offensive or not. One of the main reasons part of speech finds a significant research direction is to find if part of speech affects the classification model's performance and to research if the model's classification is analogous to the human's behaviour to predict the offensiveness of tweets. Most of the offensive tweets are hate speech or kind of sarcasm-oriented speech.

In this experiment, we have two main goals: *(i)* to create a model that performs as good as possible on classifying tweets as offensive or not, and *(ii)* to evaluate if part of speech feature tagging helps in improving the Support Vector Machine (SVM) classification. To reach our first goal, we experiment with different NLP models that, given a set of tweets, are able to predict if they are offensive. This is achieved by training them and comparing results to see which one predicts better the offensiveness of the tweets. In this paper, we describe and provide scores for four NLP models: a baseline classic model using bag-of-words and TFIDF vectorization, a classic model with optimized feature set, an optimized LSTM model with pretrained static embeddings, and a fine-tuned pretrained language model. For the second goal, we experiment with different SVM features to improve the classifier's performance. We add part of speech tag features which segments each word of the tweet, we train the tweet data using the SVM baseline classifier with POS-tag, and we evaluate the classifier's performance.

On experimenting with several models, we explored that the BERT and RoBERTa model's performance was better than the one of LSTM and SVM baseline model with and without features. The BERT and RoBERTa models successfully achieve 79 F1 macro average score on test set. On another hand, using part of speech tags did lead to slight increase in accuracy of SVM classification's accuracy, what helped in identifying non-offensive tweets better.

2 Related Work

There have already been some experiments on the task of identifying offensiveness in social media using NLP tools. The main contribution is SemEval-2019 Task 6: OffensEval-2019 (Zampieri et al., 2019b), in which they also identified offensive language in Twitter. In their ex-

[1] `https://github.com/poojagowda7818/LFD_group9_Final-Project`

periment, they had three main objectives: identifying offensive language, automatic categorization of offense types, and offense target identification. They studied both the type and the target of the offensive language, and they are the creators of the OLID dataset (Zampieri et al., 2019a), the one we are using in our experiment. Their paper describes a shared task in which 115 teams participated in at least one of the three tasks. The evaluation results showed that the best performing systems used BERT, but there were no newer Language Models (LMs) used.

After seeing the interest of researchers in OffensEval-2019, they published SemEval-2020 Task 12: OffensEval-2020 (Zampieri et al., 2020). They had the same objectives, but improved the previous task by incrementing the semi-supervised training dataset, providing the datasat in four languages (Arabic, Danish, Greek and Turkish), and using larger test datasets for all subtasks. 145 teams submitted results this time, and the winning team reaching the highest F1-score used BERT followed by a sigmoid classifier. They also performed translation of the meaning of emojis.

Apart from those, there have been other shared tasks following the same goal such as GermEval-2019 (Wiegand et al., 2018) with German tweets, HASOC-2019 (Mandl et al., 2019) for English, German, and Hindi, HatEval-2019 (Basile et al., 2019) for English and Spanish, and TRAC-2020 (Kumar et al., 2020) for English, Bengali, and Hindi.

The work of Mladenovic et al. (2017) integrates features like antonymous pairs, positive sentiment polarity of words, ordered sequence of sentiment tags in a tweet, POS-tags of words, and irony markers, and achieves improved classification compared to the case when that feature is not used. This work also provides an excellent motivation for us to work on this research direction.

Some of the participants in SemEval-2019 (Zampieri et al., 2019b) and SemEval-2020 (Zampieri et al., 2020) contributed to the task by integrating lexical features in the classification using the SVM algorithm (Plaza-del Arco et al., 2019), using a linear SVM with character n-gram features to identify the target of hate speech (Zhu et al., 2019), and by training a linear SVM classifier with a TF-IDF feature model to detect offensive tweets and messages (Saroj et al., 2020).

3 Method

3.1 Data

The dataset we are using in our experiment is the Offensive Language Identification Dataset (OLID)[2] (Zampieri et al., 2019a), which is publicly available online. It consists of a large dataset of English tweets annotated on three tasks: *(i)* Offensive Language Detection, *(ii)* Categorization of Offensive Language, and *(iii)* Offensive Language Target Identification. All the example tweets were retrieved from Twitter using its API and searching for keywords that are usually used in offensive tweets. As the majority of content on Twitter is non-offensive, Zampieri et al. (2019a) tried different strategies to have around 30% of offensive tweets in their dataset.

The corpus is divided into train, development, and test sets as shown in Table 1. The test set we are using is the same as the one of Zampieri et al. (2019a) in order to be able to compare results at the end. There is a total of 14,097 tweets divided into 86.82% train, 7.09% development, and 6.09% test, and from which 32.9% are considered offensive.

	OFF	**NOT**	**Total**
Train	4,047	8,192	12,239
Dev	352	647	999
Test	239	620	859
Total	4,638	9,459	**14,097**

Table 1: Total number of tweets in the dataset and their division into train, dev and test set and offensiveness.

3.1.1 Data preprocessing

For our experiment, we are only interested in the first category, the one about Offensive Language Detection. Therefore, our dataset has been preprocessed and consists of a tweet per line and a single tag at the end indicating whether the tweet is offensive (OFF) or non-offensive (NOT). Table 2 shows some examples of how the tweets are presented.

Apart from getting rid of the labels we are not interested into, we added some further preprocessing to the tweets themselves. The steps included converting each letter to lowercase; removing unlabeled data, unnecessary punctuation, some sym-

[2]`https://scholar.harvard.edu/malmasi/olid`

Original tweet	Preprocessed tweet	Tag
@USER Fuck off	fuck off	OFF
@USER He is a DUMBASS !!!!!	he is a dumbass ! ! ! ! !	OFF
@USER Liberals are all Kookoo !!!	liberals are all kookoo ! ! !	OFF
@USER Buy more icecream!!!	buy more icecream ! ! !	NOT
@USER @USER You are correct.	@USER you are correct.	NOT

Table 2: Five example tweets before and after applying further preprocessing steps and their corresponding tag indicating if they are offensive (OFF) or not (NOT).

bols like '#' and '@' and the repetitive '@USER' at the beginning of each line; adding spaces between words and punctuation, POS-tags, noun tags, gpe count, sentence count and org count using Spacy (Honnibal and Montani, 2017); renaming the column names to *Tweet* and *Task*; and up-down sampling the training data to go from 33.1% offensive and 66.9% non-offensive to 50.0% of both. The resulting data after applying all the preprocessing steps can be seen at Table 2.

3.2 Experiments

3.2.1 Baseline classic SVM model using bag-of-words and TFIDF vectorizer

As per the first research objective, we will experiment with various classification models throughout the research to evaluate various models' performance in detecting whether the tweets are offensive or not. Hence, we first set a classic baseline model using a kit from the scikit-learn library (Pedregosa et al., 2011) with a bag-of-words (BoW) or term frequency-inverse document frequency (TFIDF) representation of tweets, and a linear Support Vector Machine (SVM) classifier. We tune the hyperparameters manually using the development (dev) set and run the model on the test set once we have successfully tuned the model.

We experiment with BoW and TFIDF to evaluate the performance corresponding to these word representations (vector). BoW words use only co-occurrences of terms, whereas TFIDF also contains information on much more significant and lesser significant ones (overall document weightage of a word).

We have chosen an SVM linear kernel classifier. Kernel is a function that takes low dimensional input space, transforms it to a higher dimensional space, and helps convert non-separable problems to separable ones. In addition, we changed the C value, where this value controls the trade-off between smooth decision boundaries and classifying training points correctly. If the C value is increased, the probability of accurately getting more training points is high. We set C value to 7, and obtained a good performing classification model. The Table 3 shows the final result of classic SVM baseline model after fine-tuning.

Vectorizers	Accuracy	F1 macro avg.
Bow	69%	67
TFIDF	71%	69

Table 3: Accuracy and F1 macro average attained on preprocessed dev set with classic SVM classification model with c=7, random seed=32, and kernel=linear and using BoW and TFIDF vectorizers.

For the tokenizer, we use Spacy tokenizer (Honnibal and Montani, 2017) using the English *en_core_web_sm* pipeline[3]. Here the text is segmented, and we create doc objects with the discovered segment boundaries we return as tokens. While preprocessing the data, we merged the processed and unprocessed tweets. Consequently, we train the SVM classification model here using the preprocessed tweets and their corresponding label (*Task*).

3.2.2 Classic model with optimised feature set

In this experiment, we evaluated the classic SVM baseline model with added custom features. As stated during data preprocessing, we added some custom features, including the geo-positional entities (gpe), number of named entities, and number of organisations (org). Also, we add a calculated count of organisations, count of gpe, and count of sentences for every textual data. Here, we combine them with the other feature vectors

[3]`https://spacy.io/models/en#en_core_web_sm`

to experiment and evaluate the SVM classification model's performance. Preprocessed custom test set is created by adding custom features that we mention above and running the classic SVM baseline model on this test set. We do not use tokenizer, we directly convert the fitted text to vectors representation instead (BoW or TFIDF) .

3.2.3 An optimised LSTM model with static embeddings

Our third model consists on a Long Short-Term Memory (LSTM) with static embeddings. To evaluate our LSTM model, we tried different hyperparameters to reach a high degree of accuracy. We experiment with the number of epochs, the batch size, the optimizer, the learn- ing rate, the loss function, the model, the LSTM layer, and we freeze and unfreeze the training of the layer by setting the trainable value.

The number of epochs indicates how many total interactions of the dataset we want to run. The batch size defines the number of sam- ples to work on before the model's internal param- eters are updated. The optimizers are applied to improve the training dynamics of RNNs, whereas the learning rate defines how quickly the network updates its parameters. The loss function helps us calculate the gradients to update the weights of the RNN, and the model is defined to decide the type of LSTM model we want to use. Finally, the LSTM layer is accompanied by the dropout layer, which avoids overfitting.

Bidirectional LSTM (BiLSTM) is a sequence processing model that will consist of two LSTMs: one which takes the input in a forward direction, and another in a backward direction. BiLSTMs effectively increases the amount of information available to the network and improves the context available to the algorithm (eg. knows which words follow and precede a word in a sentence). The GloVe embeddings (Pennington et al., 2014) we have used consists of 100 dimensions. First we read the GloVe vector, then we create the embedding matrix and return the embedding layer. This embedding layer is used in Bidirectional LSTM model, and the hyperparameters are tuned manually by running the model on dev set vigorously.

Finally, we set the epochs to 20 with an early stopping functionality, batch size 32, learning rate 0.001, Adam optimiser, binary crossentropy, loss function, embedding dimension 100, and maximum length for padding 100. We apply Dense layer with sigmoid activation function. Overall, the application of the concept was pretty much right, but we did not observe great results of classification model. The Table 4 shows the results we obtained on dev set by fine-tuning the hyperparameters with "unfreezing" embedding layer.

Model	Accuracy	F1 macro avg.
LSTM	65%	39

Table 4: Accuracy and F1 macro average attained on LSTM bidirectional model with GloVe after fine-tuning and setting the tuned hyperparamaters values as: epoch=20, batch size=32, learning rate=0.001, optimiser=Adam, and loss function=BinaryCrossentropy[logits =True].

3.2.4 A fine-tuned pretrained language model

In this experiment, the model is trained on unprocessed data (with punctuations). We evaluate the model's performance by looking at the F1 macro average score (F1 macro avg.) and accuracy on the test set. We use a Pretrained Language Model (PLM) to classify tweets as offensive or not, i.e. a binary classification. We focus on using transformers architecture, which depends on a self-attention mechanism without utilising any Recurrent Neural Network (RNN) in the encoder and decoder. PLMs aim to predict the next word given a previous sequence of words. To do so, the model needs to know the language quite well and must have a vast knowledge of words in that language. They achieve this by analysing text data and interpreting it with an algorithm that establishes rules. In this process, the model learns the language's characteristics and can then apply the previously learned rules to produce and understand new sentences.

We experimented with different PLMs to evaluate their performance in detecting offensive language. The settings of the PLMs that we experimented with are the number of epochs, the batch size, the optimiser, the learning rate, the loss function, and the model.

We run the model with different PLMs on 10 epochs with an early stopping function as a callback while training the model. This function stops the training when validation loss values are not improving. Most of the time, training ended before 10 epochs. Hence, we set the epochs value to 10. We tune the model first by changing the learning rate, and we observe a drop in the model's performance on decreasing the learning rate from 5e-5. Also on, since increasing the learning rate led to a gradual decline, we set the value of the learning rate to 5e-5.

On the other hand, on changing batch size, we observed that there was a decline in the model's macro average score on decreasing the batch size. On increasing, batch size performance improved. At size 32, the macro avg score was stable and improved substantially as we increased to 64. The loss function we have used is binary crossentropy, as we wish to identify if the tweet is OFF or NOT, so there can be two possibilities (true or false). We used Adam optimiser from SGD and set the maximum length to 100, which also affected an increase of macro average. After increasing the maximum length value, the macro average score dropped instantly. The Table 5 shows the performance obtained on specific PLM after fine-tuning on dev set.

PLMs	Accuracy	F1 macro avg.
BERT	77%	75
RoBERTa	77%	75
DistilBERT	78%	76

Table 5: Accuracy and F1 macro average attained on three PLMs after fine-tuning and setting the tuned hyperparamaters values as: epochs=10, batch size=64, learning rate= 5e-5, optimizer=Adam, and loss function=BinaryCrossentropy[logits =True].

BERT-base uncased (Devlin et al., 2018) is a transformers model that has been pretrained on a large corpus of English data on the raw texts, with no humans labelling them in any way, allowing it to use lots of publicly available data. We first experiment using this model and we fine-tune this model manually by running it on dev set vigorously. DistilBERT (Sanh et al., 2019) is a transformers model, 40% smaller and 60% faster than BERT, that retains 97% of the language understanding capabilities. Finally, RoBERTa (Liu et al., 2019) is replication study of BERT pretraining, which includes a careful evaluation of the effects of hyperparmeter tuning and training set size. We find that BERT was significantly under trained and propose an improved recipe for training BERT models, which should approximately match the performance of BERT model.

3.2.5 Evaluation of SVM model with POS-tags

This experiment aims to evaluate the SVM classifier's performance with POS-tagged tweets. During Data preprocessing, we tag each tweet text with a POS-tag (NOUN, PROPER NOUN, VERB, ADJ, and PRONOUN) using Spacy. We run the classic SVM baseline model with POS-tagged tweet, and train the SVM model keeping the hyperparameters the same. We run the model with different vectorizers (BoW and TFIDF) here to observe the changes in the performance.

4 Results

4.1 Baseline classic SVM model using bag-of-words and TFIDF

In previous section we discussed briefly on how the baseline SVM model performs after fine-tuning on dev set. Table 6 shows the performance of the model on processed test set without POS-tag with respect to dev set, which looks overall better than on dev set. As per the results obtained, the performance of SVM model on test set was much better with around 73% accuracy in both BoW and TFIDF vectorizer.

	Dev		Test	
Vec	Acc.	F1mac	Acc.	F1mac
BoW	69%	67	73%	69
TFIDF	**71%**	69	**73%**	70

Table 6: Results for offensive language detection with baseline classic model using bag-of-words (BoW) and TFIDF on test set and dev set. We report the accuracy obtained and F1 macro average (F1mac) score on dev and test set.

4.2 Classic model with optimised feature set

We add custom features geo-positional count, org count, sentence count, note that we don't use Pos tagging here. We create a customised test data set using these features we run the baseline model with matrix of custom features surprisingly we observe a decline in accuracy. Table 7 shows decline in accuracy of classic model after adding the custom features that we mention. The classic model's accuracy declines from 73% to 70%.

But in Table 8 when you look at the F1 scores with respect to each label (OFF and NOT), there is slight increase in F1 score while classifying the OFF labels. This gives a very interesting observation, as the results showed us that, with features, the classification performance of model in identifying tweets to be offensive gets better. The features org count, geo-positional features turns out to be helping the model classify better.

	Baseline		W/features	
Vec	**Acc.**	**F1mac**	**Acc.**	**F1mac**
BoW	73%	68	70%	68
TFIDF	**73%**	70	**70%**	69

Table 7: Comparison of results for offensive language detection with respect to baseline SVM model with and without features using bag-of-words (BoW) and TFIDF vectorizers on custom test set. We report the accuracy and F1 macro average score (F1mac) obtained on test set.

	Baseline		W/features	
Vec	**NOT**	**OFF**	**NOT**	**OFF**
BoW	80	58	75	61
TFIDF	**80**	59	**75**	62

Table 8: F1 macro average score results with respect to classes for offensive language detection with baseline classic model with and without features using bag-of-words (BoW) and TFIDF on the test set and dev set.

4.3 An optimised LSTM model with static embeddings

Table 9 shows results of LSTM with unfreezing the embedding layer on test set With the GloVe embeddings that we use of 100 dimension to generate better results. The results clearly shows the weak performance of LSTM model.

Model	Accuracy	F1 macro avg.
LSTM	72%	42

Table 9: Performance attained after fine tuning and setting the tuned hyper paramaters values, epoch =20, batch size =32, learning rate = 0.001, optimizer = Adam, loss function= BinaryCrossentropy[logits =True] on test set

We tried to tune the model several times, but freezing the embedding layer does not seem to improve the results. Table 10 shows results of classification, which helps in further analysing of the classification of labels (OFF or NOT).

Model	NOT	OFF
LSTM	0	84

Table 10: Performance attained after fine tuning and setting the tuned hyper paramaters values, epoch =20, batch size =32, learning rate = 0.001, optimizer = Adam, loss function= BinaryCrossentropy[logits =True] on test set

4.4 A fine-tuned pretrained language model

We train the unprocessed tweet in PLMs, we use BERT, RoBERTa, DistillBERT PLMs to classify the tweets to target classes as offensive or non-offensive. Table 11 shows the results on test data.

PLMs	Accuracy	F1 macro avg.
BERT	83%	79
RoBERTa	83%	79
DistilBERT	81%	76

Table 11: Performance attained after fine tuning and setting the tuned hyper paramaters values, epoch =10, batch size =32, learning rate = 5e-5, optimizer = Adam, loss function= BinaryCrossentropy[logits =True]

By looking at the macro average score of PLMsm we can see the accuracy of classification has improved in comparison to that of other models. Especially BERT and RoBERta give almost same results as Distilbert. It was expected from DistilBERT to have lesser performance results from that of BERT and RoBERTa, as it uses only 40% lesser parameters than BERT uncased (Sanh et al., 2019).

4.5 Evaluation of SVM model with POS-tags

We train the POS-tagged tweet for this experiment instead of the processed tweet to evaluate the performance of the classic baseline SVM model. Table 12 shows the results obtained by comparing the POS-tags concerning the Classic SVM baseline model's performance without the POS-tags.

We can see from the results that using the POS-tags slightly increases the SVM classification's performance. The accuracy increases slightly from 73% to 76% using BoW vectorization, and when we apply TFIDF, the accuracy of the SVM classification increases here as well from 73% to 75%.

	W/o POS-tags		W/POS-tags	
Vec	**Acc.**	**F1mac**	**Acc.**	**F1mac**
BoW	**73%**	69	**76%**	71
TFIDF	**73%**	70	**75%**	71

Table 12: Comparison of results for offensive language detection with respect to baseline SVM model with and without POS-tags using bag-of-words (BoW) and TFIDF vectorizers on test set. We report the accuracy and F1 macro average score (F1mac) obtained on test set.

5 Discussion

On observing the results of every model's performance, we observe the classification eventually increases when we apply Pretrained Language Models, especially with BERT and RoBERTa. Using a pretrained model requires less training and requires less effort in building the model's architecture, which makes a better model to use for classifying the offensive language. On another hand, using POS-tags did show a significant increase, to be precise, this helped the classifier to identify the non-offensive tweets better than SVM baseline classifier which trains of tweet without POS-tags.

For example, during error analysis, it was observed that out of 858 tweets in the test set using a POS-tag, the classifier predicted 651 tweets accurately. Without a POS-tag, the classifier predicted 624 tweets accurately. Mostly, the POS-tag has helped the classifier predict the non-offensive tweets better than the baseline model using TFIDF. Along with the overall weight of the specific words, the part of speech also improves the classifier and helps it to learn better to identify the non-offensiveness comparatively. One of the example tweet's prediction which we would like to discuss about is the following:

> "#BREAKING: After a week-long trial, a Linn County jury convicted Gregory Davis of First-Degree Murder in connection with the Carrie Davis homicide that took place almost a year ago. URL URL".

This was classified as offensive by the classifier without POS-tags, but it was not offensive in test set, and the classifier with POS-tags predicted it as non-offensive. This example tweet gave an insight into the importance of using POS-tags, as this feature helps the classifier to learn and to identify the context of the tweets. This tweet consists of negative or potentially offensive words like "murder" and "homicide", which without POS-tags are considered to be offensive due to the occurrences of these words with respect to label, but with POS-tags the classifier is enabled to infer the context and predict the sentence to be non-offensive.

6 Conclusion

Throughout this research our aim was to improve and evaluate the detection of offensiveness in Twitter by creating a model that performed as good as possible on classifying tweets as offensive or not, and by evaluating if part of speech feature tagging helps in improving the SVM classification to protect people from cyberbullying. We experimented with several models to evaluate which model classifies the offensiveness present in a tweet more accurately. Also, we applied part of speech tagging to explore its implications on the SVM classifier's performance using TFIDF or BoW vectorizers. On observing the results of every model's performance, we can clearly state that BERT and RoBERTa models perform better in classification, and, on applying the POS-tags, that there is an increase in the accuracy of the SVM classifier.

Without a POS-tag, the classifier predicted 624 tweets accurately, and with POS-tag, as we stated in previous section, the performance of prediction accuracy increased to 651. Significantly POS-tag inclusion to tweets led the classifier to predict the offensive tweets better than the baseline model using TFIDF. Along with the overall weight of the specific words, the part of speech also improved the classifier and helped it to learn better to identify the non-offensiveness comparatively to model trained on tweet without POS-tag. As a part of future work, one can work on BERT model by using tweets with POS-tags and other features which focus to explore the context and tone of the tweets.

To conclude, we can state that we succeeded with our main objectives, what could help avoiding hate speech in social media.

7 Work distribution

Since we were only two people in the team instead of three, the work load was higher. Due to this, we could not work efficiently towards the additional research question in comparison to other teams.

7.1 Pooja Gowda

Code & Research: I have worked on the code writing, formatting, fine tuning, experimentation of every model (research question 1 and research question 2) and literature study with respect to research question 2, error analysis, data preprocessing, GitHub management.

Report: I wrote Experiments, Results, Discussion with error analysis, and Conclusion sections.

7.2 Leire Varela

Code & Research: I worked on research and literature study of research question 1 and on analysis and exploration of provided data.

Report: I wrote Abstract, Introduction, Related Work, Data, and I formatted the report overall.

References

[Basile et al.2019] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th international workshop on semantic evaluation*, pages 54–63.

[Devlin et al.2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Honnibal and Montani2017] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

[Kumar et al.2020] Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2020. Evaluating aggression identification in social media. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 1–5, Marseille, France, May. European Language Resources Association (ELRA).

[Liu et al.2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

[Mandl et al.2019] Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of the 11th forum for information retrieval evaluation*, pages 14–17.

[Mladenovic et al.2017] Miljana Mladenovic, Cvetana Krstev, Jelena Mitrović, and Ranka Stanković. 2017. Using lexical resources for irony and sarcasm classification. 09.

[Pedregosa et al.2011] 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.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Plaza-del Arco et al.2019] Flor Miriam Plaza-del Arco, M. Dolores Molina-González, Maite Martin, and L. Alfonso Ureña-López. 2019. SINAI at SemEval-2019 task 6: Incorporating lexicon knowledge into SVM learning to identify and categorize offensive language in social media. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 735–738, Minneapolis, Minnesota, USA, June. Association for Computational Linguistics.

[Sanh et al.2019] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

[Saroj et al.2020] Anita Saroj, Supriya Chanda, and Sukomal Pal. 2020. Irlab@ iitv at semeval-2020 task 12: multilingual offensive language identification in social media using svm. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2012–2016.

[Wiegand et al.2018] Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.

[Zampieri et al.2019a] 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. *arXiv preprint arXiv:1902.09666*.

[Zampieri et al.2019b] 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). *arXiv preprint arXiv:1903.08983*.

[Zampieri et al.2020] Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1425–1447, Barcelona (online), December. International Committee for Computational Linguistics.

[Zhu et al.2019] Jian Zhu, Zuoyu Tian, and Sandra Kübler. 2019. Um-iu@ling at semeval-2019 task 6: Identifying offensive tweets using bert and svms.