Automatic offensive language detection from Twitter data using machine learning and feature selection of metadata

1st Gabriel Araújo De Souza
Federal University of Rio Grande do Norte (UFRN)
Natal, Brazil
gabriel_feg@hotmail.com

2nd Márjory Da Costa-Abreu
Sheffield Hallam University, Sheffield, UK
m.da-costa-abreu@shu.ac.uk
https://orcid.org/0000-0001-7461-7570

***Abstract*—The popularity of social networks has only increased in recent years. In theory, the use of social media was proposed so we could share our views online, keep in contact with loved ones or share good moments of life. However, the reality is not so perfect, so you have people sharing hate speech-related messages, or using it to bully specific individuals, for instance, or even creating robots where their only goal is to target specific situations or people. Identifying who wrote such text is not easy and there are several possible ways of doing it, such as using natural language processing or machine learning algorithms that can investigate and perform predictions using the meta-data associated with it. In this work, we present an initial investigation of which are the best machine learning techniques to detect offensive language in tweets. After an analysis of the current trend in the literature about the recent text classification techniques, we have selected Linear SVM and Naive Bayes algorithms for our initial tests. For the preprocessing of data, we have used different techniques for attribute selection that will be justified in the literature section. After our experiments, we have obtained 92% of accuracy and 95% of recall to detect offensive language with Naive Bayes and 90% of accuracy and 92% of recall with Linear SVM. From our understanding, these results overcome our related literature and are a good indicator of the importance of the data description approach we have used.**

***Index Terms*—Offensive Language Detection, Naive Bayes, Linear SVM, Attribute Selection, Twitter**

I. Introduction

In the face of popularisation of social media such as Facebook, Twitter, Instagram and Tik Tok, the communication between people has become faster and easier. In these communication mechanisms, people can express their feelings, criticism, opinions, achievements, etc. However, many times these networks are used to publicise hate speech though offensive words. The types of the offense can be directed to most diverse aspects including Ethnicity, Sexism, Economic status, Religion, Sexual Orientation, and so on. Thus, the big problem with this relies on the fact that the offense is presented to individuals or groups and this can be potentially harmful to them [1], [2].

The challenge in automated detection of offensive language lies in the fact that, in social networks, the used language contains a specific format that belongs to the environment. In this context, several word abbreviations are used and various forms of expression intensification and word modification, such as several letters repetitions (e.g.: loooooved, gooood) and excessive use of punctuation (e.g.: i loved!!!!, what????). Thus, the original text must be modified through a very important preprocessing stage to get a form that keeps the original sense and can somehow coincide with other similar posts [3].

For this, a software that is able to detect this type of offenses in a social network is a very important advancement to ensure the security and mental health of users [4]. Another considered application is detecting users that commonly perform cyber-bully acts and aggression. From this, measures must be taken to punish and block aggressive users [5].

Thus, this paper proposes an investigation to detect offensive language in twitter data. After a review of the literature, two techniques were chosen for this task: Linear SVM and Naive Bayes. The main objective is to improve these algorithms when compared with similar experiments found in the literature. For this, some results already published are presented and a comparison of implemented techniques will be discussed.

This work is organised as: Section II presents a review of the literature with the recent works about automatic detection of offensive language or similar. Section III presents the database structure and the process of data normalisation. Section IV introduces the concepts of chosen machine learning algorithms to describe the formulas and steps of the algorithm. Section V shows and explains the results obtained in our experiments and Section VI reports the conclusions obtained from the work.

II. Related Works

The investigation of hate speech in social media is a relatively new research area, but, despite that, it has created a lot of attention and have already whole events dedicated to the subject as well as an increase of related publications. This session will present and discuss some of the main findings of the most recent and relevant publications.

As an example, the International Workshop on Semantic Evaluation SemEval-2019[1] [6] presented tasks that focused on

[1]http://alt.qcri.org/semeval2019

detection and categorising of offensive language in social media. The three main sub-task were: offensive language identification, automatic categorisation of offense types, and offense target identification. For the first sub-task, the messages were classified as offensive or not offensive. A twitter post was label as offensive if it contained any offensive or profane language. The Deep Learning BERT [7] presented better results for this task [8]. In the second sub-task, the goal was to predict the type of offense, and for this, two classes were used: Insult and Untargeted. A twitter post was classified as 'Insult' when it contained an insult to an individual or group; and, a twitter post was labeled as 'Untargeted' when that the post contained non-acceptable language (swearing). The best solution found for this problem used a rule-based approach with a keyword filter, such as hashtags, signs, emoticons, and other features [9]. The last sub-task focused on the target of offenses. The used classes were 'Individual' for an offense to a unique user, 'Group' for an offense to a group of people, and 'Other' for an offense to an organisation, a situation, an event, or an issue. The team with better results also used BERT for this problem [10]. This Deep Learning model also was used to detect offensive language in German texts [11].

In [12], a similar application was explored, where the authors created a new dataset using the Twitter API for twitter data classification as hate speech, offensive language or neither. In their dataset, they collected a set of 85.4 million twitter samples from about 33 thousand Twitter users. From there, they built a set of 24k labeled twitter samples with features, such as bigram, unigram, and trigram, which were weighted by their TF-IDF and were used for the classification task. Other features included binary and count indicators for hashtags, mentions, retweets, and URLs. They tested a large number of classifiers: logistic regression, Naive Bayes, decision trees, random forests, and linear SVMs (Support Vector Machine). Through their experiments, it was found that Logistic Regression and Linear SVM tended to perform better results. The best model obtained an overall precision of 0.91, recall of 0.90, and F1-score of 0.9. However, the classifier did not present good results to detect hate speech, for this class, the precision and recall were 0.44 and 0.61, respectively.

A Deep Learning model for classification of messages in social media was proposed in [13], with the labels considered were racism, sexism or neutral. For their experiments, they joined various Long Short-Term Memory (LSTM) models, and in their classification, the features defined a user's tendency towards posting messages in any used classes, the set of messages posted by a user, and subsets that contained labeled messages. The method used was independent of the language and obtained better results to detect sexists messages (around 0.99 of precision and F1-Score). Neutral messages also presented good results (0.94 for precision), but racist messages obtained inferior results of about 0.75 and 0.70 of precision and F1-Score respectively. The best value obtained a average accuracy and precision around of 93% to all classes.

The problem of detecting hate speech was expanded in [14] for vulnerable community identification. The features extracted from the messages was performed using techniques such as convert words to vector and n-grams. For the process of hate speech detection, they used the Gated Recurrent Unit (GRU) and a variety of RNNs. These classifiers provided an accuracy of about 0.92, and thus, a Convolutional Neural Networks (CNN) was proposed in [15] for classifying offensive tweets written in English. The labels used in this work were offensive, abusive or hate-inducing, and the best results obtained an accuracy of 0.83 and a precision of 0.80.

In [16], the models SVM, bidirectional Long Short-Term-Memory (BiLSTM), and CNN were used to classify messages as offensive or not offensive. In the experiments, the BiLSTM obtained a better precision to detect offensive messages (0.81). For detection of not offensive language, the precision was 0.83, and, SVM and CNN model obtained a precision of 0.66 and 0.78 respectively to detect offensive messages and 0.80 and 0.87 to detect not offensive messages.

Table I presented a summary of all the relevant works for this paper and based on it, we have identified a lack of investigation regarding the type of features used in order to investigate offensive language. Also, a better fine tuning approach to the standard classical classification techniques was not fully explored, which can lead to poorer results. Thus, we reinstate that our main goal with this work is to investigate the quality of used features for this problem as well as the fine tuning approach of classical classification techniques.

III. Hate speech on twitter posts: Database

Since we have spotted a gap in the research area to explore fine tuning and feature selection, in order to validate our proposed ideas, we needed to, preferably, select a public dataset, so we would be able to compare our results with the state of the art. Thus, the used dataset was collected by [12] and contains 24783 tweets. From these, 1430 are classified as hate speech, 19190 as offensive language, and 4163 as normal language. Due to the low amount of messages classified as hate speech in comparison with the other labels and the low performance to detect hate speech with this dataset describe in [12], for this work, we have chosen to use only the labels of offensive and normal language.

The first reason for the choice of this dataset is due to the variety of examples of offensive twitters. The organisation of data in this dataset was another important reason for your choice because it facilitated data processing and manipulation. The good results obtained with machine learning algorithms present in [12] for offensive language detection was the final reason for us to decide to use it.

In order to select a larger number of messages tagged as an offensive language than normal, it is necessary to balance the dataset [19]. For this, we have chosen randomly a subset of messages marked as offensive with the same size as the set of normal messages. Thus, we selected a set of 4163 normal and 4163 offensive tweets, creating a dataset with a total size of 8326 messages. We have selected randomly offensive messages to provide more diversity in the dataset and a variety of seeds that will be used to test different datasets

TABLE I
RESUME OF RELATED WORKS

Reference	**Database**	**Objective**	**Feature Extraction**	**Classification**	**Results**
[8]	Offensive Language Identification Dataset (OLID)	Categorisation of offensive language in social media	Emoji substitution, HashTag segmentation and convert all the text into lowercase (Misc)	Logistic Regression (LR), LSTM and BERT	LR: 72% Acc LSTM: 76% Acc BERT: 84% Acc
[9]	Offensive Language Identification Dataset (OLID)	Automatic categorisation of offense types	Rule-based approach with a keyword filter based on a Twitter language	Modified sentence offensiveness calculation (MSOC) and RNN	RNN: 86% Acc MSOC: 92% Acc
[10]	Offensive Language Identification Dataset (OLID)	Offense target identification	Pre-processing and pre-trained word embedding based on GloVe.	BERT-Base, Multilayer Perceptron Network (MLP) and Soft Voting Classifier (SVC)	BERT-Base: 72% Acc MLP: 68% Acc SVC: 69% Acc
[11]	Dataset of German-language tweets provided in context of the GermEval Shared Task 2 (2019)	Offensive language identification for German-language texts	Replace all user mentions to a token Name	BERT	76% F1-Score
[12]	They created their own dataset described in [12]	Automatic hate-speech detection on social media is the separation of hate speech from other instances of offensive language.	Lower case, create bigram, unigram, and trigram features, TF-iDF and others	LR e SVM	Precision of 0.91 and Recall of 0.90 for offensive language detection and precision of 0.44 and recall for hate speech
[13]	A dataset of approximately 16k short messages from Twitter, that was made available by [17]	Detecting Offensive Language	Define the three features representing a user's tendency towards posting Neutral, Racist and Sexist content	An ensemble of Recurrent Neural Network (RNN) classifiers	Precision: 93% and Recall: 93%
[14]	Public hate speech datasets available in [18]	Vulnerable community identification using hate speech detection on social media	Convert words to vector and n-grams	Gated Recurrent Unit (GRU) and a variety of RNNs	RNN-GRU: 92% Acc GBT: 92% Acc RNN-LSTM: 91% Acc
[15]	Hinglish dataset HEOT and [12]	Detecting Offensive Tweets in Hindi-English	Removal of punctuations, URLs and user mentions; lower case; remove stop words and others	CNN	Accuracy of 83% and recall of 71%
[16]	Offensive Language Identi-fication Dataset (OLID)	Predicting the Type and Target of Offensive Posts in Social Media	Pre-processing of text	SVM, bidirectional Long Short-Term-Memory (BiLSTM), and CNN	SVM: 76% precision and 78% recall BiLSTM: 82% precision and 82% recall CNN: 82% precision and 82% recall

configurations. Based on related work [8], [13], [15], [16], we have decided to divide the dataset with more messages in the training dataset. Thus, we have allocated 60% of data to train and 40% to test, where the choice of training and test dataset was made at random as well and both sets contained the same number of offensive and normal messages.

A. Data processing

A tweet contains a diversity of elements that can confuse a text classifier, for example, user names, hashtags, URLs and emojis. This occurs because there exists a variety of different shapes in these texts which can be very complex to find patterns. Grammar errors and excessive use of repeat letters are other problems because these end up generating several different forms of the same term or word. Thus, before the data can be analysed by any model, it is necessary to perform a preprocessing in the text to remove or reduce the mentioned problems without losing the semantic meaning of the message. It is possible to find in the literature techniques to perform normalisation [10], [15], convert to lower case [10], [15] and removal of stop words [10], [15] with the propose to generate a new text with the same sense, but in a way that provides better performance to the text classification algorithms.

1) Data Normalisation: This process consists of mapping divergent text that belongs to the same class in a label. In this work, we have converted all the hashtags, user names, emojis, URLs and retweets to a tag that represents each information. Thus, every time that a hashtag is found in the text, it is replaced to tag $< hashtag >$, emojis are replaced to $< emojis >$ and so on [10]. Other important point in the normalisation processing is the removal of all text punctuation. This is a very common process and contributed for a clean text with focus in the words [10], [15].

For an algorithm, the word "Car" is different from "car". In a tweet, the words can be written with different uses of upper case and lower case, the words can also be written completely in lower case or completely in upper case or even contain upper and lower case occurrences simultaneously. For reducing this divergence, we have converted all text to lower case.

In a text, words such as articles, pronouns, connectors, etc are considered irrelevant to the process of classification [20]. This type of text appears frequently and so can hamper the training process. Thus, these words are known as stop words. A common method in a preprocessing of text stands is to remove all stop words to create a clean text with just what is relevant. For this work, we have created a list of words that are considered stop words, and we have removed all the words in a tweet that were present in the list.

Two forms of preprocessing were used, the first, considering the tags in a process of data normalisation which we called 'Data Type A'. The second form which removed all the hashtags, emojis, URLs, Retweets, User names and stop words leaving a very clean text, which we called 'Data Type B'. All forms will be used in each proposed technique. And in the next section, we will present the techniques and methods that were used in our experiments.

IV. Methodology

After a review of the literature, we have found many techniques to detect and classify offensive language. Deep learning and neural networks have been used for this purpose and each paper presented different results regarding performance. The SVM-based solution has not appeared, nevertheless, the authors of the used dataset obtained good results [12], [16]. The Naive Bayes classifier was also not mentioned in the recent literature, but it has presented good results with similar problems [21]. These classifiers are easy to implement and have a low computational cost, mainly the Naive Bayes. Based on that, these classifiers can present a cheaper, faster and better alternative to use in automatic language classification and as alternative to Neural Network and Deep Learning models. Due to this, we have decided to implement these techniques and evaluate their results. A brief description of these techniques will be presented in Sections IV-A and IV-B.

A. Naive Bayes Classifier

In this classifier, a table of words occurrences, also named bag of words, is created in the training process. This table contains words and a number of occurrences for this word in each class. Another information expressed in the table is the total of words for each class and the total words on the training set. The stop words are not considered [21].

After the training step, the mentioned table is ready to be used by the classifier. For this, the classifier receives a preprocessing twitter and a list of words is created, thus, for each class is calculated a score for this message that informs how pertinent this message is for this class. The class that obtained the biggest score is returned by the classifier. The score is calculated by the following equation: $score(y_i, W) = log\frac{y_i^t}{t} + \sum_{i=1}^{n} log\frac{w_i^{y_i}+mp}{y_i^t+m}$, where, y_i is the i^{th} class represents in the model, W is a set of words, y_i^t is a total of words classified as y_i class in the training data, t is the total of words in the training data, w_i^y is the amount of times that the i^{th} word in the W set is classified as y_i class in the training data, and m and p are parameters: $p = 0.5$ and $m = 1$

The formula $score(y_i, W)$ measures the score of a tweet W to the class y_i, and thus, W is a set of words resulting from the tweet prior processing and y_i is one of the classes (relevant or irrelevant). In this formula, for each word in W is calculated a value that represents the significance of this word in class y_i through on the parameters p and m. The value of all words is summed and the result is added with the logarithm of the total of words tagged as the class y_i divided by the total of words. The classifier calculates the score of W to all classes and the class with a better score is returned by the classifier.

B. Support Vector Machine

The objective of this classifier is to find a hyper-plane in a space with n dimensions, where n is a number of features that distinctly classifies the data points. In the training process, the principal goal is to find limitation points that can separate objects of distinct classes. In order to maximise the margin of the classifier, support vectors are used. These are data points that are closer to the hyper-plane and influence the orientation of the hyper-plane [22].

The first step to run the SVM is to prepare the data by transforming each message in a numerical feature vector. A common technique in the literature is to use TF-IDF (Term Frequency - Inverse Document Frequency) [23]. These metrics inform how interesting a word is for a type of document. However, in our experiments, we noticed a better result using the technique 'bag of words', also used for the Naive Bayes classifier and described in Section IV-A. After a construction of table with examples of offensive and normal texts, each phase is associate with a score for a offensive message and other score to normal language, thus, each phase receive two features demarcating how much belongs to the set of offensive language and the set of normal language respectively.

With the data, a default gradient for each class is generated and in the process of the train they are updated for each wrong classification or correct classification by the equations: $\omega = \omega - \alpha \cdot (2\lambda\omega)$ when a correct classification is realised and $\omega = \omega + \alpha \cdot (y_i \cdot x_i - 2\lambda\omega)$ when an incorrect classification is realised.

The ω is a gradient, α is the learning rate and λ is a regularisation parameter defined as $1/epochs$. After the training process, the product is to have the weights w_1 and w_2 and these are used in the classification process. For this step, the same process of converting text into two numeric features is executed. By applying the weights in the values the next step, we can verify if the result is great than one (1), so the text is classified as offensive, otherwise, it is normal.

V. Results

In this section, we will present the results of our experiments. The two algorithms (Linear Support Vector Machine and Naive Bayes) presented previously were fully implemented by the authors. For each algorithm, we tested the two configurations of data describes in section III-A: Data Type A and Data Type B.

For the Data Type A, the Naive Bayes achieved a gain of performance of around 1.5% when compared with Data Type B. For the Data Type A, the Linear SVM demonstrated a greater difficulty in parameter setting to find satisfactory results. We tested to change the configuration of the alpha parameter with the values 0.1, 0.01 and 0.001 besides configuring the max epochs to 100 and 500, but in all possible combinations of theses parameters, the algorithm always classified any text as offensive. However, with the Data Type B, it was possible to find good results configuring the parameter values to alpha equals 0.01 and epochs equals 100.

Regarding the dataset, it was previously mentioned that the number of examples of offensive messages is greater than the number of normal messages and that to create the dataset, we have selected randomly a subset of offensive messages with the same size as the set of normal messages. For the Linear SVM, this process was generated a unique time because another process of randomisation is performed to define the order of messages evaluation in the training process. However, due to the process of classification with Naive Bayes being simpler, we tested different seeds to the process of selecting the set of offensive messages. In the LSVM classifier, we also tested different seeds for randomising the evaluation order of messages. Thus, we have tested 50 seeds ranging from 0 to 49 to each random describe process.

Table II shows the results to tests using Linear SVM Classifier. During each of 50 seeds, we have collected values of accuracy, precision, recall, and F1-Score. Three values are calculated to each mentioned metric based on all executions: The best value found to all executions that can be visualised in the second column, the average value of executions (third column), and the standard derivation of executions (fourth column). The seed with the best value of accuracy and precision simultaneously is highlighted soon after.

The seed with the best result went to the seed 6 and the results for this are 90% of accuracy, 88% of precision, 92% of recall and an F1-score of 90%.

Table III presents the results to the tests using the Naive Bayes classifier. Such as the Linear SVM classifier, we have presented the best and average values of accuracy, precision, recall, and F1-Score to all executions, besides the standard derivation. The table distribution is similar to the SVM table, and the seed with the best result went to the 8 and the values are 92% of accuracy, 89% of precision, 95% of recall and an F1-score of 92%.

TABLE II
RESULTS OF LINEAR SVM CLASSIFIER WITH AVERAGE (AV) AND STANDARD DEVIATIONS (SD)

Metric	Best Value	AV	SD
Accuracy	0.900	0.523	0.136
Precision	0.883	0.272	0.260
Recall	1.0	0.300	0.442
F1-Score	0.902	0.242	0.345

TABLE III
RESULTS OF NAIVE BAYES CLASSIFIER WITH AVERAGE (AV) AND STANDARD DEVIATIONS (SD)

Metric	Best Value	AV	SD
Accuracy	0.922	0.912	0.004
Precision	0.899	0.882	0.006
Recall	0.964	0.950	0.005
F1-Score	0.924	0.915	0.004

We also performed a statistic test (t-test) to decide if the generated values for each algorithm have had a significant statistical difference between them. For each metric, we would use all the generated values by both algorithms and apply the t-test with the one-tailed hypothesis and a significance level of 0.05.

Based on the results, all calculated metrics have values with significant statistical differences between the algorithms. Therefore, it is concluded that the Naive Bayes classifier has a better performance to detect offensive language with an accuracy of 92% and a recall of 95% for the best dataset configuration.

Comparing the Naive Bayes performance with the researched literature, our results are better than or equal to all works except an RNN configuration presented in [13] that obtained 93% of precision and recall to detect offensive language. It is interesting to note that the developed Naive Bayes performed better than Deep Learning BERT in two different datasets.

In the dataset OLID, the BERT performed with an accuracy of 84% [8] and accuracy of 72% in [10]. For German language tweets, the performance of BERT is an F1-Score of 71% [11]. In both uses of BERT, the authors also using feature extractions such as emoji substitution, lowercase, change names token format, and others (for more details see Table I).

Our classifier also presented better results than CNN models. In [15], the dataset used was the same as the one we used and the CNN model showed lower performance than our with an accuracy of 83% and recall of 71%. Another model of CNN for a different dataset (OLID) [16] presented lower performance than Naive Bayes with a precision of 82% and a recall of 82%. In both CNN usage, a text preprocessing with similar to our feature extractions are mentioned.

In the RNN presented by [13], the authors used a different dataset of ours. The dataset contains short tweets classified as racism, sexism, and neutral while the dataset used by us contains the neutral, offensive and hate speech classes. The authors use features such as remove punctuation, name token format, and tags in all tweets to express the user tendency. This last is expressed through three features that represent the tendency to post neutral, racist and sexist tweets. Each tweet can contain a combination of these tags. The authors tested many models and in the best model, the performance to detect racist tweets stayed around 75% to precision and 66% to recall, but on average the precision and recall were 93%. The RNN precision is bigger than Naive Bayes precision, but the recall is not. Although the datasets are different, they contain the same data type (offensive language in tweets). Therefore, it can be said that Naive Bayes is a viable alternative to RNN.

Thus, we have a simpler and cheaper implementation of Linear SVM and Naive Bayes that obtained better results than a related work to the same goal.

VI. Conclusion

In this work, we have explored the use of Linear SVM and Naive Bayes classifiers to detect the offensive language in tweets. During our test phase, we have noted that the Linear SVM is very sensitive to data type used in the training process. It was also detected that the data normalisation with tags made it difficult for the process of parameter regulation. The tests also showed that the evaluation order of messages strongly influences the final result of the classifier, this is noted due to the high standard derivation for the tests with different seeds. This is a normal process because if big sequences of messages with the same label are given as input, for example, the weight regulation and the learning coefficient (alpha) make the learning arranged by the other inputs causing an imbalance of the weights. Thus, the Linear SVM needed a balanced input to obtain good results. The setting of the parameters for this algorithm proved to be a bit tricky task.

In contrast, the Naive Bayes classifier proved to be a good text classifier. One of your positive points is its simplicity and easiness of implementation that made this algorithm very fast. This algorithm showed to be better of many techniques demonstrated in the literature.

For future works, we aim to implement other text classification algorithms and evaluate the performance in comparison with our Linear SVM and Naive Bayes. An architecture to real-time tweet classification can be developed with these algorithms, mainly the Naive Bayes which obtained good results and is very fast as well as it has the simplicity of probabilistic-based analysis for classification. And finally, we are already planning to explore the use of these algorithms for classifying other types of text.

References

[1] F. Del-Vigna, A. Cimino, F. Dell-Orletta, M. Petrocchi, and M. Tesconi, "Hate me, hate me not: Hate speech detection on facebook," in *First Italian Conference on Cybersecurity*, 2017.

[2] J. Jacobs and K. Potter, *Hate crimes: Criminal law & identity politics*. Oxford University Press on Demand, 1998.

[3] M. Bouazizi and T. Ohtsuki, "Multi-class sentiment analysis on twitter: Classification performance and challenges," *Big Data Mining and Analytics*, vol. 2, no. 3, pp. 181–194, Sep. 2019.

[4] G. Jalaja and C. Kavitha, *Sentiment Analysis for Text Extracted from Twitter*. Singapore: Springer Singapore, 2019, pp. 693–700.

[5] S. Sharma and A. Jain, "Cyber social media analytics and issues: A pragmatic approach for twitter sentiment analysis," in *Advances in Computer Communication and Computational Sciences*, S. K. Bhatia, S. Tiwari, K. K. Mishra, and M. C. Trivedi, Eds. Singapore: Springer Singapore, 2019, pp. 473–484.

[6] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Identifying and categorizing offensive language in social media (offenseval)," *arXiv preprint arXiv:1903.08983*, 2019.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] P. Liu, W. Li, and L. Zou, "Transfer learning for offensive language detection using bidirectional transformers," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019, pp. 87–91.

[9] J. Han, S. Wu, and X. Liu, "Identifying and categorizing offensive language in social media," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019, pp. 652–656.

[10] A. Nikolov and V. Radivchev, "Offensive tweet classification with bert and ensembles," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019, pp. 691–695.

[11] J. Risch, A. Stoll, M. Ziegele, and R. Krestel, "hpidedis at germeval 2019: Offensive language identification using a german bert model," in *Preliminary proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019). Erlangen, Germany: German Society for Computational Linguistics & Language Technology*, 2019, pp. 403–408.

[12] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proceedings of the 11th International AAAI Conference on Weblogs and Social Media*, ser. ICWSM '17, 2017.

[13] G. Pitsilis, H. Ramampiaro, and H. Langseth, "Detecting offensive language in tweets using deep learning," *arXiv preprint arXiv:1801.04433*, 2018.

[14] Z. Mossie and J.-H. Wang, "Vulnerable community identification using hate speech detection on social media," *Information Processing & Management*, p. 102087, 2019.

[15] P. Mathur, R. Shah, R. Sawhney, and D. Mahata, "Detecting offensive tweets in hindi-english code-switched language," in *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, 2018, pp. 18–26.

[16] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Predicting the type and target of offensive posts in social media," *arXiv preprint arXiv:1902.09666*, 2019.

[17] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? predictive features for hate speech detection on twitter," in *Proceedings of the NAACL student research workshop*, 2016, pp. 88–93.

[18] H. Watanabe, M. Bouazizi, and T. Ohtsuki, "Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection," *IEEE Access*, vol. 6, pp. 13 825–13 835, 2018.

[19] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25–36, 2006.

[20] J. Wilbur and K. Sirotkin, "The automatic identification of stop words," *Journal of information science*, vol. 18, no. 1, pp. 45–55, 1992.

[21] I. Rish, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[22] Y.-W. Chang and C.-J. Lin, "Feature ranking using linear svm," in *Causation and Prediction Challenge*, 2008, pp. 53–64.

[23] G. Forman, "Bns feature scaling: an improved representation over tf-idf for svm text classification," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 263–270.