JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Automated Detection of Hate Speech and Sentiment Analysis on Twitter using Machine Learning Techniques

Renu Kumari and Vijay Kumar
College of Commerce Arts and Science
Patliputra University, Patna, 800020 Bihar, India

Abstract. Sentiment analysis, or opinion mining, is a critical Natural Language Processing (NLP) technique that discerns the emotional tone in textual content, categorizing it as positive, negative, or neutral. It finds extensive application in understanding public sentiments on social media platforms like Twitter, Facebook, and Instagram. However, the proliferation of hate speech on these platforms poses a pressing issue, promoting violence, discrimination, and prejudice. This paper addresses the challenge of hate speech on Twitter, a widely utilized micro-blogging platform. Many methods have already been created to automate hate speech detection online. This process has two elements: identifying the qualities these terms utilize to target a specific group and classifying textual material as hate or non-hate speech. Detecting hate speech is more challenging, as our research of the language used in typical datasets reveals that hate speech lacks distinctive, discriminatory characteristics. In this paper, we present a novel approach that involves classifying tweets into three categories: "sexism," "racism," or "none." By doing so, we aim to detect and categorize instances of harmful content on Twitter. Our work contributes to sentiment analysis and offers a practical solution to identify and combat hate speech on a platform with significant societal influence. Machine learning methods are beneficial for capturing the meaning of hate speech and are thus proposed as feature extractors. Data from social media sites such as Twitter are used to test the effectiveness of these procedures, and they reveal a significant improvement in macro-average F1 and 9% improvement for content labeled as hateful, respectively.

Keywords: Hate speech · Machine learning · Online social networks · NLP · Text classification · Social media.

1 Introduction

Opinion mining, also known as sentiment analysis, is a fundamental aspect of Natural Language Processing (NLP). It enables the automated assessment of emotional tones conveyed in textual data, categorizing sentiments as positive, negative, or neutral, with the potential to capture nuanced emotions like happiness, sadness, or anger. Opinion mining finds widespread application in analyzing public sentiments on social media platforms, such as

Twitter, Facebook, and Instagram, providing valuable insights into public perceptions. However, the proliferation of hate speech on online platforms is a growing concern. Hate speech encompasses any form of communication promoting violence, discrimination,

hostility, or prejudice based on attributes like race, ethnicity, gender, religion, or sexual orientation.

In recent times, the proliferation of hate speech has become increasingly prevalent, both in face-to-face interactions and online communication. The influence of social media and various online platforms has significantly contributed to the dissemination of hateful content, with grave consequences often culminating in hate crimes. Notable instances, such as the impact of online hate speech surrounding the US Presidential election in 2016 [1], the terrorist attacks in Manchester and London in the UK [2], and the horrific events in New Zealand [3], have underscored the severe ramifications of this issue. To combat these detrimental effects, legislative measures have been implemented, with the European Union Commission compelling social media networks to adhere to an EU hate speech code, mandating the removal of such content within 24 hours [4]. However, the manual process involved in identifying and eradicating hate speech content is arduous and time-consuming. Given the concerning prevalence of hate speech on the internet, there is a compelling need for automated hate speech detection.

The automatic detection of hate speech presents a formidable challenge, primarily due to the absence of a universally accepted definition of hate speech. Consequently, what is perceived as hateful can vary among individuals, contingent on their particular interpretations. As proposed by Fortuna et al. [5], hate speech can be defined as content that advocates violence against individuals or groups based on attributes such as race or ethnic origin, religion, disability, gender,age, veteran status, and sexual orientation or gender identity. Despite the diversity in these definitions, recent research endeavors have yielded promising outcomes in the realm of automatic hate speech detection within textual content [6,7,8,9]. These pioneering solutions have employed an array of feature engineering techniques and machine learning (ML) algorithms to categorize content as hate speech. However, despite the substantial body of work in this domain, there remains a paucity of comparative analyses evaluating the performance of distinct feature engineering techniques and ML algorithms in classifying hate speech content. This paper centers on the application of sentiment analysis to Twitter tweets for the purposes of hate speech detection and monitoring. Leveraging the power of machine learning, with its capacity to process extensive data and unveil discernible patterns, is pivotal in this endeavor. The paper surveys various machine learning methodologies, encompassing both supervised and unsupervised learning, feature engineering, deep learning, and NLP. Additionally,it delves into the formidable challenges and constraints intrinsic to hate speech detection, including issues related to data scarcity and algorithmic biases. In today's digital landscape, the prevalence of hate speech underscores the need for effective detection methods. This paper contributes to addressing this issue by examining machine learning-based hate speech detection and its challenges, aiming to provide a comprehensive overview of this vital and evolving field of research. The paper's structure entails the following: Section 2 provides an overview of related research, Section 3 discusses the methodology, Section 4 discusses the results and its analysis, and finally, Section 5 addresses limitations, suggests future research directions, and concludes the paper.

2. Literature survey

Hate speech has become alarmingly prevalent in contemporary social media landscapes, prompting researchers to employ supervised machine learning (ML)- based text classification approaches for hate speech content identification. These endeavors have encompassed a wide array of feature representation techniques, including dictionary-based [10,11,12], Bag-of-Words (BOW) based [6,7,8], Ngrams- based [13,9,14], TF-IDF-based [15,16], and Deep Learning-based [16] methods. In [17], the authors utilized a dictionary-based approach coupled with Ngram feature engineering to detect cyber hate on Twitter, achieving a maximum F-score of 67% using support vector machines (SVM). Similarly, The authors in [11] applied a dictionary-based approach for racism detection in Dutch social media, yielding an F-score of 0.46 through SVM. Njagi Dennis et al. in their paper [10] classified web forum and blog hate speech using sentiment expressions and subjectivity features, obtaining 73% precision with a rule-based classifier. Despite these promising outcomes, the dependency on large dictionaries remains a limitation. BOW-based methods, despite ignoring word order, have demonstrated superior accuracy. The existing study [12] attained 87% accuracy by employing bi-gram feature extraction with SVM. Another existing study [6] achieved a maximum 76% accuracy by utilizing uni-grams with BOW. Sanjana Sharma et al. [7] reported a maximum accuracy of 73% using BOW features. To address the word order limitation, N-grams-based techniques [18] have been proposed. The existing article [9] employed character N-grams and logistic regression (LR) to achieve an F-score of 73%. Chikashi Nobata et al. [13] reached a 77% F-score with character N-grams and SVM. Shervin Malmasi et al. [8] used 4-grams with character grams and SVM, resulting in 78% accuracy. Recent endeavors have focused on multilingual hate speech detection, such as Danish [19], achieving an F1 score of 0.74 using deep learning. Schmidt et al. [20] conducted a comprehensive survey of hate speech detection using NLP techniques, although they lacked experimental results. Despite global efforts to combat hate speech in languages like German, Dutch, and English, a comparative study assessing various features and ML algorithms on standard datasets, essential as a baseline for future research, is conspicuously absent. Consequently, this study conducts a comprehensive evaluation of three feature engineering techniques and eight ML classifiers on hate speech datasets, as detailed in Section 3.

3 Proposed methodology

The prevalence of hate speech on platforms like Twitter poses a significant challenge. Detecting and categorizing such harmful content, particularly regarding sexism and racism, is a complex task. Hate speech often lacks easily identifiable characteristics, making its automated detection vital for a safer online environment. three distinct feature extraction approaches: Bigram with CountVectorizer, TF-IDF and Word2Vec. In this study SVM, Logistic Regression, Naïve Bayes and KNN classifiers are used.

3.1 Data collection

We used publicly available tweets1 for this study. There are a total of 11325 tweets in this dataset. The given data contains three unique classes namely ‘sexism’, ‘racism’ and ‘normal’. 2988 tweets show ‘sexism’, 20 tweets show ‘sexism’ and 8317 tweets are categorized as ‘none’.

3.2 Data Preprocessing

Preprocessing of data is a critical step in the data analysis and machine learning workflow. It involves cleaning and transforming raw data into a format that is suitable for further analysis and modeling. The preprocessing step aims to remove inconsistencies, noise, and irrelevant information from the data, making it more meaningful and easier to work with. All special symbols like $, #, !, @ etc. and emoticons are removed from the dataset. All tweets are converted into lower case and stopwords like is, and, a, the, etc. are removed. The classes assigned to each data namely 'none', 'racism', 'sexism' are labeled as 0, 1, 2 respectively.

3.3 Data Splitting

Data splitting is done to assess the performance of the model. Dataset is split into two subsets, a training set and a testing set. The purpose of this splitting is to train our machine learning model on one subset and evaluate its performance on another to know how well it performs on new (unseen) data. Dataset is split into training and testing data such that 20% of data is included in test set and 80% in train set. Test set contains 2265 tweets and train set contains 9060 tweets.

1 https://raw.githubusercontent.com/srishb28/Hate-Speech-Detection-on-Twitter-Data/master/final_dataset.csv

3.4 Feature Engineering

The choice of feature engineering techniques significantly impacts text classification outcomes [21]. Previous research has demonstrated the superiority of the TF-IDF representation technique over binary and term frequency representations [22]. Conversely, the lower performance of Word2Vec can be attributed to its limitations in handling out-of-vocabulary words, particularly in the domain of Twitter data. Furthermore, Word2Vec requires an extensive training set to effectively capture intricate word relationships [23]. In contrast, our dataset comprises approximately 15, 000 tweets, which may not be sufficient for Word2Vec to learn the nuanced word relationships effectively. Similarly, Doc2Vec demonstrated sub-optimal performance, likely due to its reduced efficacy with very short-length documents, such as the tweets in our dataset, often limited to 280 characters [24]. A word embedding is a learned representation for text where words that have the same meaning have a similar representation. It a type of NLP technique where individual words are represented as real-valued vectors in a predefined vector space. These methods expect to catch the semantic and relevant relationships between words on the basis of their distributional properties in a huge corpus of text. By addressing words as continuous vectors, word embeddings empower model to effectively process and understand language.

3.5 Word Embedding Techniques

CountVectorizer It is a text preprocessing tool used in NLP and machine learning. It converts a collection of text documents into a numerical format suitable for machine learning algorithms. It tokenizes the text, creating a vocabulary of unique terms, then counts how many times each term in the vocabulary appears in each document. It generates a matrix where rows correspond to documents, and columns correspond to terms in the vocabulary. The values in this matrix represent the frequency of each term in each document. The resulting matrix is used as input data for machine learning algorithms. Each row of the matrix represents a document, and each column represents a term. This numeric representation allows machine learning models to operate on text data, making it possible to train and evaluate models for various NLP tasks. The resulting feature matrix is interpretable since each column corresponds to a specific term, allowing us to understand which terms are important for our model. It's compatible with various machine learning algorithms, including Naive Bayes, logistic regression, which require numerical input data.

TF-IDF TF-IDF stands for Term Frequency (TF)-Inverse Document Frequency (IDF). It is a numerical statistic used in information retrieval and text mining to measure the significance of a term within a document or a collection of documents. TF-IDF combines two factors: TF and IDF. TF measures how frequently a term appears in a document. It is calculated by dividing frequency of a term by total number of terms within a document. The motive behind TF is that the more times a term appears in a document, the more likely it is to be important or significant to that document. It is calculated by the Eq. 1.

$$\text{TF}\ (t,d) = \frac{Number\ of\ term\ t\ appears\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d} \qquad (t,d) = \frac{Number\ of\ term\ t\ appears\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d} \qquad (1)$$

IDF measures the rarity or uniqueness of a term across a collection of documents. It is calculated by dividing the total number of documents in the collection by the number of documents that contain the term and then taking the logarithm of that ratio. The IDF value is higher for terms that are unique indicating that those terms are more important or carry more information. It is calculated by the Eq. 2.

$$\text{IDF} \qquad (t,D)\ log = \left(\frac{Total\ numbe\ of\ documents\ in\ corpus\ D}{Number\ of\ ducuments\ containing\ term\ t}\right)$$

$$(t,D)\ log = \left(\frac{Total\ numbe\ of\ documents\ in\ corpus\ D}{Number\ of\ ducuments\ containing\ term\ t}\right) \qquad (2)$$

TF-IDF is obtained by multiplying the term frequency (TF) of a term in a document by the inverse document frequency (IDF) of that term across the entire collection of documents. The higher the TF-IDF value, the more significant the term is to the document. It is calculated by the Eq. 3.

$$TF - IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

$$TF - IDF(t,d,D) = TF(t,d) \times IDF(t,D) \qquad (3)$$

TF-IDF represents the TF-IDF score. t represents the term (word) for which you are calculating TF-IDF. d represents the specific document in which you are calculating TF. D represents the entire document collection or corpus. It gives higher weights to terms that are unique in the document collection. TF-IDF helps to lessen the impact of common terms that appear frequently across documents.

Word2Vec It is a technique used in NLP to represent words as numerical vectors. It creates vectors of the words that are distributed numerical representations of word features. Words with similar meanings or usage have similar vector representations. This numerical representation allows machine to understand and work with words in a more effective way. There are two main architectures for training Word2Vec models i.e., Continuous Bag-of-Words (CBOW) and Skipgram. CBOW predicts a target word based on the context words surrounding it. Skip-gram predicts the context words given a target word. Word2Vec offers several advantages in NLP and text analysis: It captures the semantic meaning of words by representing them as dense vectors. Words with similar meanings or usage tend to have similar vector representations. This allows for measuring semantic similarity and capturing relationships between words. It maps high-dimensional word space into lower-dimensional vector space, making it computationally efficient and suitable for various NLP works. It considers the context in which words appear. This helps in understanding the meaning of words in different contexts.

3.6 Machine Learning Classifier

In machine learning, a classifier is a type of model that is used to assign categories or labels to input data based on patterns and features present within the data. The primary goal of a classifier is to learn from a labeled dataset (where data points are already assigned to specific categories) and then make predictions or decisions about the category or class of new, unseen data. The selection of the most suitable machine learning algorithm often depends on the dataset at hand, as various algorithms exhibit varying performance across different data types. In this study SVM, Logistic Regression, Naive Bayes and KNN classifiers are used.

Logistic Regression It is a statistical method used for binary classification tasks. It is a type of supervised learning algorithm that predicts the probability of a binary outcome (usually represented as 0 or 1) based on one or more input features [25]. It uses the sigmoid function to transform a linear combination of the input features into a value between 0 and 1. The sigmoid function has an S-shaped curve and maps any real-valued number to the range [0, 1]. Where z is the linear combination of input features and their corresponding weights, plus a bias term. It is expressed by the Eq. 4.

$$z = \omega_o + \omega_1 \cdot x_1 + \omega_2 \cdot x_2 + \ldots + \omega_r \cdot x_r$$

$$z = \omega_o + \omega_1 \cdot x_1 + \omega_2 \cdot x_2 + \ldots + \omega_r \cdot x_r \quad (4)$$

Where w0 is the bias term (intercept), $\omega_1\omega_1$, $\omega_2\omega_2$. . . wr are the coefficients (weights) for each input feature x_1x_1, x_2x_2 , . . . , x_rx_r, and r is the number of input features.

Naive Bayes Classifier It is a popular ML algorithm for classification tasks [25]. The classifier is called "naive" because it assumes that the features are conditionally independent given the class label. It describes the probability of an event based on prior knowledge of conditions related to that event. In the context of classification, it calculates

the probability of a particular class given some input data. It makes a prediction based on the class with the highest probability .Bayes 'Theorem is given by the Eq.5.

$$P(y|x) = \left(\frac{P(x|y).P(y)}{P(x)}\right) \qquad P(y|x) = \left(\frac{P(x|y).P(y)}{P(x)}\right) \tag{5}$$

K-Nearest Neighbors (K-NN) It is a popular ML algorithm used for both classification and regression tasks. It is a type of instance-based, lazy learning algorithm, which means it doesn't explicitly learn a model during training. Instead, it stores the entire training dataset in memory and uses it to make predictions when new data points are provided [26]. The "K" in K-NN represents the number of nearest neighbors to consider when making predictions. This is a hyperparameter that needs to specified before training the model. K-NN uses a distance metric to measure the similarity between data points. It calculates the distance between the new data point and all points in the training dataset and makes predictions by taking a majority vote among the K nearest neighbors. The class that occurs most frequently among the neighbors is assigned as the predicted class for the new data point.

Support Vector Machine (SVM) It is a supervised machine learning algorithm used for classification and regression tasks [18,27]. The goal of SVM is to separate data into classes based on their features. The data points that are closest to the hyperplane and influence its position are called support vectors. It aims to find the hyperplane with the maximum margin. Maximizing the margin often leads to better generalization to unseen data. SVMs can handle non-linear data by mapping the input data into a higher-dimensional space using a kernel function. This allows SVMs to find non-linear decision boundaries.

4. Result analysis and discussion

In the experimental work, we have evaluated four classifiers over three different feature engineering techniques, giving 12 different analyses over hate speech dataset containing three classes. Our experimental results showed that the SVM algorithm with the combination of bigram with TF-IDF Feature extraction techniques showed the best results. The theoretical analysis is discussed in subsequent sections.

Class wise Performance As discussed in Section 3. We have three classes name "hate speech", "offensive but not hate speech" and "neither hate speech nor offensive speech". The results show that all features and classifiers performed well for two classes (i.e. offensive but not hate speech, and neither hate speech nor offensive speech). Our experimental results showed that the 12 combinations performed lowest for class hate speech. According to the values in the Tables, the class "Hate Speech" has the lowest training instances as compared to other classes, but the major reason for misclassification of class "Hate Speech" might be overlapping of different bigram words with higher frequency in other classes than hate speech class. For example, bigrams like "lame nigga, white trash, bitch made" are more frequently appearing in class "Offensive but not Hate Speech" as compared to class "Hate Speech". Hence, it might be possible that the classifier learned weak learning rules.

4.1 Comparative study

In this study, we conducted a comparative analysis of three distinct feature extraction approaches: Bigram with CountVectorizer, TF-IDF and Word2Vec. The experimental results distinctly favored the Bigram with TF-IDF method, which outperformed its counterparts. This preference for Bigram with TF-IDF can be attributed to its ability to preserve word sequence information, a feature that

Table 1: Classification Report for CountVectorizer using SVM, NB, and KNN

SVS				NB				KNN		
Precision recal fl-score precision recall f1-score precision recal fl-score support										
0	0.89	0.92	0.91	0.88	0.95	0.91	0.79	0.99	0.88	1662
1	0	0	0	0	0	0.	0	0	0	3
2	0.77	0.69	0.73	0.81	0.65	0.72	0.88	0.28	0.43	600
accuracy			8.6			8.7			0.8	2265
macro avg	0.55	0.54	0.55	0.56	0.53	0.55	0.56	0.42	0.44	2265
weighted avg	0.86	0.86	0.86	0.86	0.87	0.86	0.81	0.8	0.8	2265

Table 2: Classification Report for TF-IDF using SVM and LR

SVS			NB			KNN	
Precision recal fl-score precision recall f1-score support							
0	0.91	0.87	0.89	0.92	0.85	0.88	1662
1	0	0	0	0	0	0	0
2	0.69	0.78	0.73	0.65	0.79	0.71	600
accuracy						0.83	2265
macro avg	0.53	0.55	0.54	0.52	0.54	0.53	2265
weighted avg	0.85	0.85	0.85	0.84	0.83	0.83	2265

Table 3: Classification Report for TF-IDF using Naive Bayes

SVS			NB			KNN	
Precision recal fl-score precision recall f1-score support							
0	0.85	0.97	0.91	0.84	0.96	0.9	1662
1	0	0	0	0	0	0	0
2	0.88	0.52	0.65	0.82	0.5	0.62	600
accuracy			0.82			0.84	2265
macro avg	0.58	0.5	0.52	0.55	0.49	0.51	2265
weighted avg	0.86	0.85	0.84	0.83	0.84	0.82	2265

Table 4: Classification Report for Word2Vec using SVM

SVS			NB			KNN	
Precision recal fl-score precision recall f1-score support							
0	0.75	1	0.86	0.85	0.68	0.76	1696
1	0	0	0	0	0	0	4
2	0	0	0	0.4	0.65	0.5	565
accuracy			0.75			0.67	2265
macro avg	0.25	0.33	0.29	0.42	0.44	0.42	2265
weighted avg	0.56	0.75	0.64	0.74	0.67	069	2265

Table 5: Classification Report for Word2Vec using Naive Bayes

SVS			NB			KNN	
Precision recal fl-score precision recall f1-score support							
0	0.75	1	0.86	0.8	0.91	0.85	1696
1	0	0	0	0	0	0	4
2	0	0	0	0.53	0.31	0.39	565
accuracy			0.75			0.76	2265
macro avg	0.25	0.33	0.29	0.44	0.41	0.41	2265
weighted avg	0.56	0.75	0.64	0.73	0.76	0.73	2265

Word2Vec lack. In our study, we evaluated four machine learning algorithms, as elaborated in 3. Notably, Support Vector Machine (SVM) emerged as top performers. SVM's strength lies in its use of threshold functions for data separation, a feature-independent approach that can effectively handle both linear and nonlinear data due to its kernel functions. LR, while versatile, is inherently linear in nature, making it less suitable for complex, non-linear data. Subsequently, we noted lower performance from Naïve Bayes (NB), K-Nearest Neighbors (KNN). NB's performance suffered due to its assumption of conditional independence among features, limiting its adaptability to intricate feature relationships. Lastly, KNN exhibited the poorest performance, primarily due to its lazy learning approach and susceptibility to noisy data, rendering it unsuitable for hate speech tweet detection.

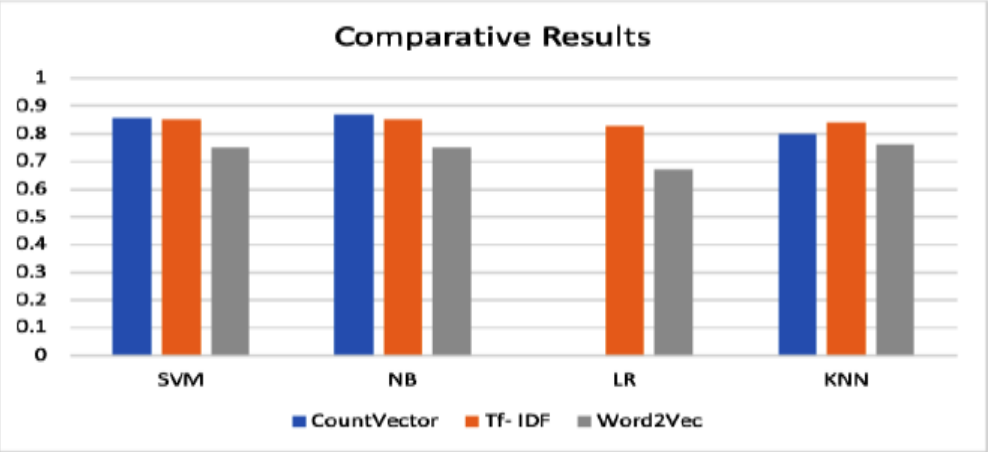

Fig.1 : Comparision of results.

5 Conclusions

The proposed work effectively harnessed automated text classification techniques to detect hate speech messages and conducted a comprehensive evaluation of three feature engineering methods and eight machine learning algorithms for hate speech classification. The results unequivocally SVM with CountVectorizer emerged as the leader in accuracy, while SVM and Logistic Regression with TFIDF excelled in precision. When it came to recall, SVM, KNN with Word2Vec, Naive Bayes demonstrated top-tier performance. These findings serve as a valuable benchmark for future research in the domain of automatic hate speech detection, offering practical significance. From a scientific perspective, this study contributes by presenting results using various scientific measures for text classification. Nevertheless, it is essential to acknowledge two notable limitations in the proposed ML model. Firstly, it faces challenges in terms of real-time prediction efficiency, which remains an area for improvement. Secondly, the model currently lacks the capability to assess the severity of hate speech messages. Looking ahead, future work will strive to enhance the model's real-time predictive abilities and its capacity to evaluate the severity of hate speech messages. This improvement journey will involve exploring lexicon-based techniques and expanding the dataset to facilitate more efficient rule learning. In doing so, we aim to bolster the model's performance and its applicability in addressing the complex challenges posed by hate speech in online communication.

References

1. Jonathan Rosa and Yarimar Bonilla. Deprovincializing trump, decolonizing diversity, and unsettling anthropology. American Ethnologist, 44(2):201–208, 2017.

2. Alan Travis. Anti-muslim hate crime surges after manchester and london bridge attacks. The Guardian, 20, 2017.

3. Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, and Ophir Frieder. Hate speech detection: Challenges and solutions. PloS one,
14(8):e0221152, 2019.

4. Alex Hern. Facebook, youtube, twitter and microsoft sign eu hate speech code. The Guardian, 31, 2016.

5. Paula Fortuna and Sérgio Nunes. A survey on automatic detection of hate speech in text. ACM Computing Surveys (CSUR), 51(4):1–30, 2018.

6. Irene Kwok and Yuzhou Wang. Locate the hate: Detecting tweets against blacks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 27, pages 1621–1622, 2013.

7. Sanjana Sharma, Saksham Agrawal, and Manish Shrivastava. Degree based classification of harmful speech using twitter data. arXiv preprint arXiv:1806.04197, 2018.

8. Shervin Malmasi and Marcos Zampieri. Detecting hate speech in social media. arXiv preprint arXiv:1712.06427, 2017.

9. Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In Proceedings of the NAACL student research workshop, pages 88–93, 2016.

10. Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. A lexicon-based approach for hate speech detection. International Journal of Multimedia and Ubiquitous Engineering, 10(4):215–230, 2015.

11. Stéphan Tulkens, Lisa Hilte, Elise Lodewyckx, Ben Verhoeven, and Walter Daelemans. A dictionary-based approach to racism detection in dutch social media. arXiv preprint arXiv:1608.08738, 2016.

12. Edel Greevy and Alan F Smeaton. Classifying racist texts using a support vector machine. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pages 468–469, 2004.

13. Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. In Proceedings of the 25th international conference on world wide web, pages 145–153, 2016.

14. Karthik Dinakar, Roi Reichart, and Henry Lieberman. Modeling the detection of textual cyberbullying. In Proceedings of the International AAAI Conference on Web and Social Media, volume 5, pages 11–17, 2011.

15. Shuhua Liu and Thomas Forss. Combining n-gram based similarity analysis with sentiment analysis in web content classification. In Special Session on Text Mining, volume 2, pages 530–537. SCITEPRESS, 2014.

16. Sebastian Köffer, Dennis M Riehle, Steffen Höhenberger, and Jörg Becker. Discussing the value of automatic hate speech detection in online debates. Multikonferenz Wirtschaftsinformatik (MKWI 2018): Data Driven X-Turning Data in Value, Leuphana, Germany, 2018.

17. Pete Burnap and Matthew L Williams. Us and them: identifying cyber hate on twitter across multiple protected characteristics. EPJ Data science, 5:1–15, 2016.

18. William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. In Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval, volume 161175, page 14. Las Vegas, NV, 1994.

19. Gudbjartur Ingi Sigurbergsson and Leon Derczynski. Offensive language and hate speech detection for danish. arXiv preprint arXiv:1908.04531, 2019.

20. Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In Proceedings of the fifth international workshop on natural language processing for social media, pages 1–10, 2017.

21. Upendra V Chaudhari and Michael Picheny. Matching criteria for vocabularyindependent search. IEEE transactions on audio, speech, and language processing, 20(5):1633–1643, 2012.

22. Ghulam Mujtaba, Liyana Shuib, Ram Gopal Raj, Retnagowri Rajandram, and Khairunisa Shaikh. Prediction of cause of death from forensic autopsy reports using text classification techniques: A comparative study. Journal of forensic and

legal medicine, 57:41–50, 2018.

23. Yang Li and Tao Yang. Word embedding for understanding natural language: a survey. Guide to big data applications, pages 83–104, 2018.

24. Ye Wang, Zhi Zhou, Shan Jin, Debin Liu, and Mi Lu. Comparisons and selections of features and classifiers for short text classification. In Iop conference series: Materials science and engineering, volume 261, page 012018. IOP Publishing, 2017.

25. David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In European conference on machine learning, pages 4–15. Springer, 1998.

26. Nitin Bhatia et al. Survey of nearest neighbor techniques. arXiv preprint arXiv:1007.0085, 2010.

27. Min-Ling Zhang and Zhi-Hua Zhou. A k-nearest neighbor based algorithm for multi-label classification. In 2005 IEEE international conference on granular computing, volume 2, pages 718–721. IEEE, 2005.