International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 5, October 2021, pp. 4431~4438
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i5.pp4431-4438

Smart detection of offensive words in social media using the soundex algorithm and permuterm index

Malek Z. Alksasbeh, Bassam A. Y. Alqaralleh, Tamer Abukhalil, Anas Abukaraki, Tawfiq Al Rawashdeh, Moha'med Al-Jaafreh
Faculty of Information Technology, Al Hussein Bin Talal University, Jordan

Article Info

Article history:

Received Jun 5, 2020
Revised Mar 27, 2021
Accepted Apr 7, 2021

Keywords:

Offensive words
Permuterm index
Smart information systems
Social media
Soundex algorithm

ABSTRACT

Offensive posts in the social media that are inappropriate for a specific age, level of maturity, or impression are quite often destined more to unadult than adult participants. Nowadays, the growth in the number of the masked offensive words in the social media is one of the ethically challenging problems. Thus, there has been growing interest in development of methods that can automatically detect posts with such words. This study aimed at developing a method that can detect the masked offensive words in which partial alteration of the word may trick the conventional monitoring systems when being posted on social media. The proposed method progresses in a series of phases that can be broken down into a pre-processing phase, which includes filtering, tokenization, and stemming; offensive word extraction phase, which relies on using the soundex algorithm and permuterm index; and a post-processing phase that classifies the users' posts in order to highlight the offensive content. Accordingly, the method detects the masked offensive words in the written text, thus forbidding certain types of offensive words from being published. Results of evaluation of performance of the proposed method indicate a 99% accuracy of detection of offensive words.

Corresponding Author:

Malek Zakarya Alksasbeh
Faculty of Information Technology
Al Hussein Bin Talal University
P.O. Box 20 Ma'an, Jordan
Email: malksasbeh@ahu.edu.jo

1. INTRODUCTION

The World Wide Web and the Internet have been making constant changes in people's everyday life [1]. Social networks became the most popular platforms on the Internet. They are currently used in various sectors and their users interact on them to realize several benefits. However, those users usually come from diverse cultures and educational backgrounds [2]. Therefore, offensive, user-generated content that is published on the various social netwroks may make the users' online experiences inconvenient since offensive words may insult and annoy them; they may show aggression against some cultures, societies, races, and/or ideologies [3]. In addition, cyberbullying is a form of offensive language and is one of the major reasons behind suicide [4].

There are cases when legal actions were taken against social media companies such as Twitter and Facebook because they did not prevent users from posting offensive words and/or hate speech [5]. However, many authors of offensive content use different variations of the same word to mask the cursing words in their attempts to avoid the automated tools which are designed to detect such content. Specifically, they replace some alphabetical letters by symbols or numbers with similar shape or sound and insert those

symbols between letters, for instance, replacing the word *fuck* with f&ck, f%ck, f$ck, f#ck, f'ck, f2k, ffuucckk, or phuck [6]. This makes manual detection of the offensive content and its removal from social networks a boring task. Consequently, there is a need for development of an effective automated tool that can detect such offensive content in social networks.

Several studies have been conducted on offensive language detection in user-generated online content. In general, they aimed at eliminating the posting of offensive language and focused on detecting different types of offenses such as cyberbullying [7]-[9], profanity or curse [10], [11], harassment [12], [13], and offensive language in general [14]-[16]. Most of these studies were based on feature extraction of offensive words from the text. Some of these studies employed the bag-of-words model [17] and some others used lexical features [18]. However, evidence supports that the various approaches followd thus far fail to understand the context of the words and sentences.

Use of lexical features can make success in easy detection of offensive entities without the need for consideration of the syntactical structure of the whole sentence. But, they fail to distinguish the sentences that contain the same offensive words but in a different order or the words which have some of their characters replaced with symbols and numbers of similar shape or sound [19]. Parts-of-speech (POS) features have also been used in offensive speech detection problems as explained in [20].

In other respects, text classifiers have been used to solve the problem of detecting offensive content. So far, the support vector machines [9], [17], [21]-[24] and the naïve bayes classifier [21]-[23] are the most popular classifiers that have been employed for this purpose. A multi-level text classifier for offensive content detection was proposed in [3] as an automated offensive content detection method. This method extracts features at different conceptual levels. Moreover, offensive detection software was designed and run at a high level of accuracy with both normal and offensive text. Furthermore, the researchers in [2] introduced a new approach that automatically classifies tweets on Twitter into three categories: offensive, hateful, and clean tweets. Experiments on this approach were performed using a Twitter dataset, considering n-grams as features and passing their term frequency-inverse document frequency (TFIDF) values to multiple machine learning models. Performance evaluation uncovered that this approach has a detection accuracy of 95.6%.

Recently, several studies highlighted the importance of using sentiment-based methods to detect offensive language. For example, the researchers in [19] applied sentiment analysis to detect bullying in tweets and used latent dirichlet allocation (LDA) topic models to recognize relevant topics in these texts. Moreover, several studies discussed different approaches to harassment detection on Web 2.0 [25], [26]. Additionally, logistic regression [16], [21], [23], random forests [21], decision trees [21], long short-term memory networks [9], and convolutional neural networks [27] were used to solve the offensive text detection problem. However, none of the efforts reported in the literature has been designed specifically to solve the problem of detecting and removing the offensive text that is masked by replacement of some of the alphabetical letters of the offensive terms with symbols and numbers that have similar shape or sound, or by insertion of symbols and special characters between letters. Thus, this study was intended to overcome the problem of automatically detecting the offensive text that has been masked. Our approach to solving this problem is based on development of a new method that integrates the Soundex algorithm with the permuterm index.

2. RESEARCH METHOD AND THE PROPOSED APPROACH

Detection of masked offensive content in short time and with low effort is an essential requirement and there is bad need for provision of robust solutions to meet it. Therefore, the method we propose in this study aims at improving the capability to automatically detect the masked offensive words. This method incorporates the Soundex algorithm and permuterm index for the purpose of extracting the offensive terms in an elegant, logical, and accurate way.

Soundex is one of the phonetic algorithms widely used for English language text detection. The main goal of this algorithm is to match homophone names, regardless of minor differences in spelling or pronunciation, by encoding them with the same representation [28]. The permuterm index, on the other hand, is a smart and time-efficient approach to solving the string-matching problem in which pattern queries may include one wildcard symbol [29].

As shown in Figure 1, our proposed method begins with importing the user's post as a tweet. Then, it starts the pre-processing phase wherein it performs the following processes: filtering, tokenization, and stemming. Subsequently, it begins the extractions phase, in which it applies the Soundex algorithm and the permuterm index together for data training and extraction of the offensive words. Thereafter, the post-processing phase is started. In this phase, the method shows the detected offensive terms, if any. This method was implemented using the Python programming language and its graphical user interface; Tkinter.

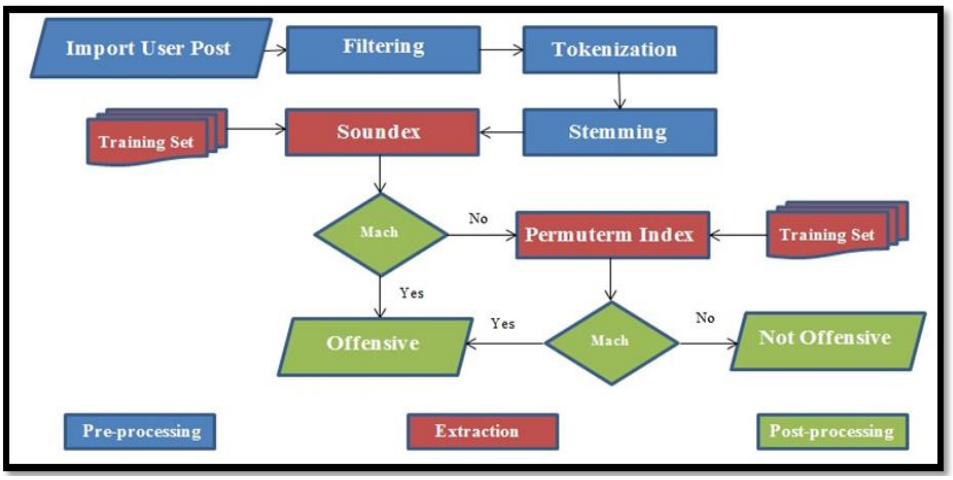

Figure 1. Architecture of the proposed offensive word detection method

2.1. Pre-processing

In the pre-processing phase (Algorithm 1), our proposed method imports the user's posts which are due to be processed. Basically, this phase consists of three processes: filtering, tokenization, and stemming. A briefing on each follow.

2.1.1. Filtering

The filtering process improves the efficiency of the Soundex algorithm and the permuterm index by rationally reducing the size of the imported text file. It removes the repetitive words that do not change the meaning of the sentence and do not have any value (e.g., prepositions and stop words). Furthermore, it removes all hyperlinks, images, audio records, and videos.

2.1.2. Tokenization

Tokenization splits the input text into tokens and generates a series of them. It also eliminates the spaces so that each word can be separated by only one white space. This step is necessary to transform the input text that has unstructured form into a suitable form for processing. To perform these tasks and ensure the splitting of the input text into tokens, the tokenization process uses the String.Split() method [30].

2.1.3. Stemming

Stemming is designed to obtain the token root using a data set that contains the roots of offensive words. This data set is used to classify the user's posts. It is trained using the Soundex algorithm and Permutrm Index. In order to obtain the root of every token, our approach uses the Porter stemming algorithm, which returns the English words to their roots [31].

Algorithm 1: Pre-processing

```
: procedure Preprocessing (P)
: ▶Input: P                                                    ∴ Post
: ◀Output: T = {t1, t2, …, tm}                                 ∴ Term
:    Filter list ← {URL, Images, Videos, Frequent words, audio, Prepositions, Stop words}
:    IF P ∈ Filter list
:        P ←remove the items that are found in the filter list from the post
:    else
:        P← P
:    End if
:    For each pi in P do
:       tokeni ← tokenizing(pi)                                ∴ Split the post
:       (termi) ← stemming(tokeni)                             ∴ Stemming the token
:       ti ← add termi to list
:    End for
: Return T
: End procedure
```

2.2. Extraction of offensive terms

In the extraction phase, we employ the Soundex algorithm and permuterm index in order to detect the offensive content that matches the offensive data existing in the training data sub-sets.

2.2.1. The soundex algorithm

The Soundex algorithm assigns values to terms in such a manner that similar-sounding terms get the same value [32]. These values are known as Soundex encodings. If Soundex encoding of any word in the post matches any Soundex encoding in the data set, then it is concluded that this post contains offensive content [33]. Table 1 presents the Soundex phonetic code for each English language letter.

In this study, the Soundex algorithm (Algorithm 2) begins with replacement of the first two letters as shown in Table 2. Then, the remaining letters of the term are replaced with phonetic code. Afterwards, any adjacent repetitions of codes and all occurrences of the 0 code are removed. Thereafter, the procedure returns the first four characters right-padding with zeroes if there are fewer than four letters. Table 3 illustrates how the Soundex algorithm works with one of the most popular offensive words, namely, *fuck*. This word is written in a variety of different ways but with a similar phoneme.

Table 1. The soundex phonetic codes of the english language letters

Letter(s)	Code
a, e, h, i, o, u, w, y	0
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m, n	5
r	6

Table 2. Suspicious letter replacement

Before	After
PH	F
TH	T
DH	D
SH	S
CK	K
GH	G
KH	K
CH	C

Algorithm 2: The soundex algorithm

```
: procedure Soundex (T)
: ►Input:T                                          ∴ Term
: ◄Output:S = {s1, s2, …, sm}                       ∴ Soundex encodings
:    For each ti in T do
:      IF first two letters (ti) ='PH'
:            ti ← replace first two letters with "F"
:      else IF first two letters (ti) ='TH'
:            ti ← replace first two letters with "T"
:      else IF first two letters (ti) ='DH'
:            ti ← replace first two letters with "D"
:      else IF first two letters (ti) ='SH'
:            ti ← replace firstwo letters with "S"
:      else IF first two letters (ti) ='CK'
:            ti ← replace first two letters with "C"
:      else IF first two letters (ti) ='GH'
:            ti ← replace first two letters with "G"
:      else IF first two letters (ti) ='KH'
:            ti ← replace two letters with "K"
:      else IF first two letters (ti) ='CH'
:            ti ← replace two letters with "C"
:      End if
:      Si ← soundex(ti)
:    End for
: Return S
: End procedure
```

Table 3. Examples on functioning of the Soundex algorithm step-by-step

Example	Step 1	Step 2	Step 3	Step 4	Final code
Fuck	Fuck	F022	F2	F2	F200
Phuck	Fuck	F022	F22	F2	F200
Fk	Fk	F2	F2	F2	F200
fuukkkkkk	Fuukkkkk	F0020000	F2	F2	F200
phuc	Fuk	F02	F2	F2	F200

2.2.2. The permuterm index

The permuterm index (Algorithm 3) is used to support wildcard querying over normal language text [34]. It adds a special character "$" to words. As an example, the word *fuck* was represented in the training sub-set as *fuck$, uck$f, ck$fu, or k$fuc*. Let us assume that the user posts '*fu*k*' or '*fu2k*'. This index will look for *fu* and *k*, ending up with *k$fu*. It simply rotates the wildcard so that it will appear at the end only.

Algorithm 3: The permuterm index

```
: procedure Permuterm(T)
: ▶Input:T                                   ∴ Term
: ◀ Output: PT = {pt1, pt2, …, ptm}
:    For each t_i in T do
:       pt_i ← permuterm(t_i)
:    End for
:  Return PT
: End procedure
```

2.3. Post-processing

In this phase (Algorithm 4), our approach classifies the user's post into offensive or non-offensive text based on the results of matching between the value returned by the soundex algorithm or the permuterm index and the content of the training sub-sets.

Algorithm 4: Post-processing

```
: procedure Post-processing (S,PT)
: ◀Output: O = {o1, o2, …, om}              ∴ offensive or not offensive
:  F ← false
: For each s_i in S do
:  IF s_i∈training sub-set of Soundex
:     F ← true
:          Print "Found Offensive:" + si
:   else
:     For each pt_j in PT do
:               IF pt_j ∈training sub-set of Permuterm
:                  F ← true
:                  Print "Found Offensive:" + pt_j
:               End if
:     End for
:  End if
: End for
: End procedure
```

3. PERFORMANCE EVALUATION

The data used in this study were collected during the period 1 October 2019 to 28 February 2020. As we are focusing on English tweets, this study picked the most popular 20 offensive English words according to [6]. Subsequently, five-hundred English tweets were manually selected as a sample for this study. Half of the instances in this sample were identified as offensive tweets while the other half were non-offensive tweets.

To fully evaluate performance of our proposed approach, precision (P), recall (R), the F-measure (F), and accuracy (A) were adopted as the performance evaluation measures, computed, and presented in a confusion matrix as shown in Table 4. In the confusion matrix, the acronyms TP, FP, TN, and FN stand for the numbers of correctly-classified offensive words, incorrectly-classified offensive words, correctly classified non-offensive words, and incorrectly classified non-offensive words, respectively. Outcomes of simulation experiments of the proposed approach are given by Table 4. The test values in the confusion matrix see in Table 4 confirm validity of the proposed method as portrayed in Figure 2.

Values of the parameters of the confusion matrix see in Table 4 were used to calculate the values of the aforementioned four performance evaluation metrics, i.e., P, R, A, and the F-measure. Precision (P) is a measure of completeness. It is calculated using as (1):

$$P = \frac{TP}{TP+FP} \quad (1)$$

Recall (R) is a measure of exactness. It is computed according to (2):

$$R = \frac{TP}{TP+FN} \quad (2)$$

Accuracy (A) is calculated as the ratio of the number of all correct predictions divided by the total number of predictions (3). Mathematically, it is expressed as (3):

$$A = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

The F-measure (F) is a weighted harmonic mean of precision and recall values. It is computed (4):

$$F = \frac{2PR}{P+R} \quad (4)$$

The values of these four metrics that are associated with the proposed method are listed in Table 5 and depicted in Figure 3. These performance assessment outcomes point out that the proposed method has an offensive word detection accuracy of 99% and precision of 98%. Further, it has a recall of 100% and an F-measure value of 99%. These values confirm that this method is highly efficient in detection of the masked offensive text that is posted on social media.

Table 4. Confusion matrix

Category	Prediction	
	Offensive	Normal
Offensive	TP= 245	FN= 0
Normal	FP= 5	TN= 250

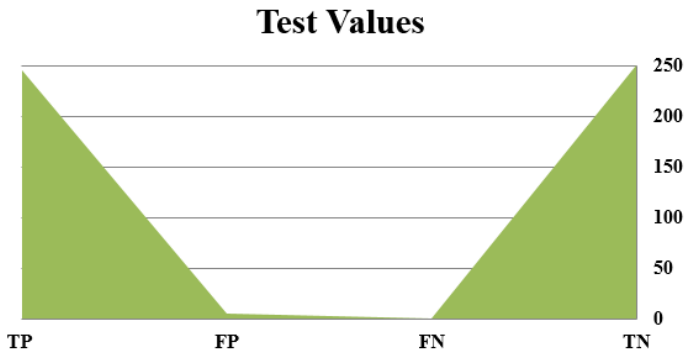

Figure 2. Values of the confusion matrix variables

Table 5. Evaluation results

	Precision	Recall	F-measure	Accuracy
Our method	98%	100%	99%	99%

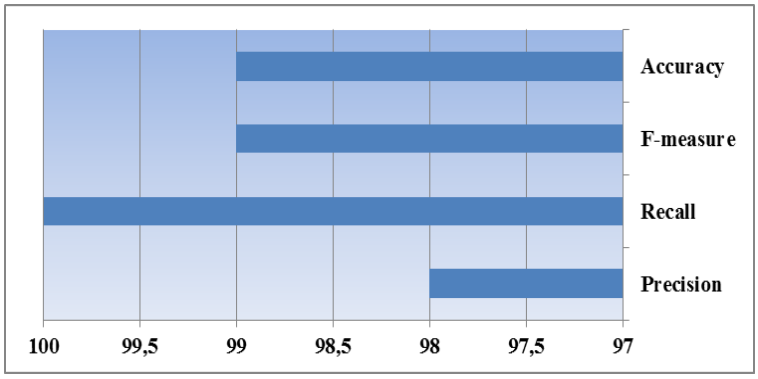

Figure 3. Values of the accuracy, precision, recall, and the f-measure associated with the proposed offensive text detection method

4. CONCLUSION

This study handled the problem of detection of the masked offensive content of English posts on social media. The main contribution of this study is that it developed a new method that integrates the Soundex algorithm with the Permuterm Index for detecting the prohibited words that are being posted on social media. Theoretically, our proposed method combines phonetic codes with indexed special tokens extracted from the textual input in order to efficiently detect the English terms that are commonly used for cursing and cussing. Practically, the proposed method can be used as a parental control tool that helps the parents in censoring the content being viewed by their children when surfing the Internet. The experimental results show that the proposed method has the ability to automatically detect restricted offensive content on social media with a very high accuracy (99%). In future work, we will attempt to develop a detection method that can identify offensive texts posted in images by converting the image to text and applying the herein proposed method on it.

REFERENCES

[1] R. Pelle, C. Alcântara, and V. P. Moreira, "A Classifier Ensemble for Offensive Text Detection," in *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, 2018, pp. 237–243, doi: 10.1145/3243082.3243111.

[2] A. Gaydhani, V. Doma, S. Kendre, and L. Bhagwat, "Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach," *arXiv preprint arXiv:1809.08651*, 2018.

[3] A. H. Razavi, D. Inkpen, S. Uritsky, and S. Matwin, "Offensive language detection using multi-level classification," in *Canadian Conference on Artificial Intelligence*, Berlin, Heidelberg, 2010, pp. 16–27, doi: 10.1007/978-3-642-13059-5_5.

[4] S. Hinduja and J. W. Patchin, "Bullying, cyberbullying, and suicide," *Archives of suicide research*, vol. 14, no. 3, pp. 206–221, 2010.

[5] S. Ullmann and M. Tomalin, "Quarantining online hate speech: technical and ethical perspectives," *Ethics and Information Technology*, vol. 22, no. 1, pp. 69–80, 2020.

[6] W. Wang, L. Chen, K. Thirunarayan, and A. P. Sheth, "Cursing in English on twitter," in *Proceedings of the 17th ACM conference on Computer supported cooperative work and social computing*, 2014, pp. 415–425, doi: 10.1145/2531602.2531734.

[7] D. Chatzakou, N. Kourtellis, J. Blackburn, E. De Cristofaro, G. Stringhini, and A. Vakali, "Mean birds: Detecting aggression and bullying on twitter," in *Proceedings of the 2017 ACM on web science conference*, 2017, pp. 13–22, doi: 10.1145/3091478.3091487.

[8] B. Ross, M. Rist, G. Carbonell, B. Cabrera, N. Kurowsky, and M. Wojatzki, "Measuring the reliability of hate speech annotations: The case of the European refugee crisis," in *Proceedings of the 3rd Workshop on Natural Language Processing for Computer-Mediated Communication*, Bochum, Germany, 2017, pp. 6–9.

[9] F. Del Vigna, A. Cimino, F. Dell'Orletta, M. Petrocchi, and M. Tesconi, "Hate me, hate me not: Hate speech detection on Facebook," in *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*, Venice, Italy, 2017, pp. 86–95.

[10] D. Fišer, T. Erjavec, and N. Ljubešić, "Legal framework, dataset and annotation schema for socially unacceptable online discourse practices in Slovene," in *Proceedings of the first workshop on abusive language online*, Vancouver, BC, Canada, 2017, pp. 46–51.

[11] S. Sood, J. Antin, and E. Churchill, "Profanity use in online communities," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2012, pp. 1481–1490, doi: 10.1145/2207676.2208610

[12] G. Kennedy *et al.*, "Technology solutions to combat online harassment," in *Proceedings of the first workshop on abusive language online, Vancouver*, BC, Canada, 2017, pp. 73–77.

[13] D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards, "Detection of harassment on web 2.0," in *Proceedings of the Content Analysis in the WEB*, Madrid, Spain, 2009, pp. 1–7.

[14] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Eleventh International Conference on Web and Social Media*, Montréal, Québec, Canada, 2017, pp. 512–515.

[15] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," in *Proceedings of the 25th international conference on world wide web*, Geneva, Switzerland, 2016, pp. 145–153.

[16] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," in *Proceedings of the 26th International Conference on World Wide Web*, Geneva, Switzerland, 2017, pp. 1391–1399, doi: 10.1145/3038912.3052591

[17] P. Burnap and M. L. Williams, "Us and them: identifying cyber hate on Twitter across multiple protected characteristics," *EPJ Data science*, vol. 5, pp. 1–15, 2016.

[18] S. Liu and T. Forss, "New classification models for detecting Hate and Violence web content," in *2015 7th international joint conference on knowledge discovery, knowledge engineering and knowledge management (IC3K)*, Lisbon, 2015, pp. 487–495.

[19] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, "Detecting offensive language in social media to protect adolescent online safety," in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing*, Amsterdam, 2012, pp. 71–80.
[20] E. Greevy, and A. 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*, 2004, pp. 468–469.
[21] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Eleventh international AAAI conference on web and social media*, Montréal, Québec, Canada, 2017, pp. 512–515.
[22] R. P. de Pelle and V. P. Moreira, "Offensive Comments in the Brazilian Web: a dataset and baseline results," in *Brazilian Workshop on Social Network Analysis and Mining*, São Paulo, SP, Brazil, 2017, pp. 510–519, doi: 10.5753/brasnam.2017.3260.
[23] H. M. Saleem, K. P. Dillon, S. Benesch, and D. Ruths, "A web of hate: Tackling hateful speech in online social spaces," in *Proceedings of the LREC 2016 Workshop on Text Analytics for Cybersecurity and Online Safety*, Portorož, Slovenia, 2017, pp. 1–10.
[24] C. Van Hee, E. Lefever, B. Verhoeven, J. Mennes, B. Desmet, G. De Pauw *et al.,* "Detection and fine-grained classification of cyberbullying events," in *International Conference Recent Advances in Natural Language Processing (RANLP)*, Hissar, Bulgaria, 2015, pp. 672–680.
[25] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, pp. 993–1022, Jan. 2003.
[26] D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards, "Detection of harassment on web 2.0," *Proceedings of the Content Analysis in the WEB*, vol. 2, 2009, pp. 1–7.
[27] B. Gambäck and U. K. Sikdar, "Using convolutional neural networks to classify hate-speech," in *Proceedings of the first workshop on abusive language online*, Vancouver, BC, Canada, 2017, pp. 85–90.
[28] D. Pinto, D. Vilariño, Y. Alemán, H. Gómez, N. Loya, and H. Jiménez-Salazar, "The Soundex phonetic algorithm revisited for SMS text representation," in *International Conference on Text, Speech and Dialogue*, Berlin, Heidelberg, 2012, pp. 47–55.
[29] P. Ferragina and R. Venturini, "The compressed permuterm index," *ACM Transactions on Algorithms (TALG)*, vol. 7, no. 1, pp. 1–21, 2010.
[30] R. E. Beasley, "String Operations," *Essential ASP. NET Web Forms Development*, Springer, pp. 183–192, 2020.
[31] F. Yamout, R. Demachkieh, G. Hamdan, and R. Sabra, "Further Enhancement to the Porter's Stemming Algorithm," in *Machine Learning and Interaction for Text based Information Retrieval*, Germany, 2004, pp. 7–23.
[32] R. Shah, and D. K. Singh, "Analysis and comparative study on phonetic matching techniques," *International Journal of Computer Applications*, vol. 87, no. 9, pp. 14–17, Feb. 2014.
[33] G. P. Hettiarachchi and D. Attygalle, "SPARCL: An improved approach for matching Sinhalese words and names in record clustering and linkage," in *2012 IEEE Global Humanitarian Technology Conference*, Seattle, WA, 2012, pp. 423–428.
[34] P. Chubak and D. Rafiei, "Index structures for efficiently searching natural language text," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 689–698, doi: 10.1145/1871437.1871527