Hate and Offensive Speech Detection in Hindi Twitter Corpus

Ishali Jadhav, Aditi Kanade, Vishesh Waghmare and Deptii Chaudhari

Hope Foundation's International Institute of Information Technology, Pune, India

Abstract

Nowadays, social media sites like Twitter and Facebook emerge as user-friendly and accessible sources for people to express their voice. Everybody, irrespective of their age group, uses these sites to share every moment of their life, making these sites flooded with data. This has led to many positive outcomes. At the same time, it has brought risks and harms as these sites set no restrictions. The volume of hate speech is not manageable by humans. As part of the HASOC-2021 shared task on information retrieval, we, Team Ignite, address the problem of hate speech identification in the Hindi corpus. Subtask A aims to identify binary hate or non-hate speech. This work was further extended with subtask B to determine the result of subtask A into three categories: profane, offensive, and hate. Hence, this paper compares the performance of three feature engineering techniques and four machine learning algorithms to evaluate their performance on a publicly available dataset with two distinct classes. With these two classes of hate and non-hate, we create a baseline model and improve model performance scores using various optimization techniques. Moreover, the output of different comparisons can be used further for text classification techniques.

Keywords

Hate speech detection; natural language processing; text classification; social media text; machine learning

1. Introduction

India is a very diverse country in terms of its culture and language variations. It has around 44% of the population speaking Hindi. In addition, the number of people using social media has increased extensively in the past decade. The exponential growth of social media has revolutionized communication and content publishing. Most youths are inclined towards it for news consumption and social interaction. However, the freedom to post any content on the internet has resulted in problems like offensive and inappropriate posts. Users quickly take advantage of this liberty to promote abuse and hatred through posts and comments.

Hate speech is a form of non-verbal or verbal communication expressing aggression or speech intended to provoke hatred or violence against a group. It is defined as an act of deprecating a person or community based on their gender, age, race, religion, nationality, etc. These hate speech posts on social media lead to a lot of violence. They face a lot of criticism for the increasingly offensive content on their sites.

Even though there has been significant research into the analysis of hate speech in numerous languages, the majority of the work has been done for English only. For many other languages, especially Indian languages, there is limited research on this recent and significant topic. Therefore, our study contributes to solving this problem by comparing three feature engineering and four machine learning classifiers on the given Hindi tweets dataset. We have created models which classify the content of a text as hateful or not. The vectors obtained from the different feature extraction techniques are fed to various classical machine learning models to predict the target class.

[1] Forum for Information Retrieval Evaluation, December 13-17, 2021, India

EMAIL: ishalijadhav20@gmail.com (I. Jadhav); aditikanade2110@gmail.com (A. Kanade); waghmarevishesh810@gmail.com (V. Waghmare); deptiic@isquareit.edu.in (D. Chaudhari.)

CEUR Workshop Proceedings (CEUR-WS.org)

The different experiments with the various models yielded different results, which will be discussed. The highest Macro F1 score obtained for Task A was 69%, and that for Task B was 63%.

2. Related Work

Classifying hate speech can become tedious considering the different ways a user can use a word. For example, the term "gay" can be used derogatorily and in the circumstances unrelated to hate speech.

To identify cyber hatred on Twitter, Peter Burnap et al. [13] used a dictionary-based technique. This study used an N-gram feature engineering technique to construct numeric vectors from a predetermined vocabulary of hostile phrases. The authors used an ML classifier called SVM to feed the obtained numeric vector and got a maximum F-score of 67 percent. A dictionary-based technique was also employed by Stéphan Tulkens et al. [14] to detect racism in Dutch social media. The authors used the distribution of words among three dictionaries as a characteristic in this investigation. The SVM classifier was given the generated features. Their experimental results yielded an F-Score of 0.46. Njagi Dennis et al. [3] classified hate speech in web forums and blogs using a machine learning-based classifier. To create a master feature vector, the authors used a dictionary-based technique. The characteristics were created using sentiment expressions, semantic and subjective features, and a bias toward hate speech. The resulting feature vector was then supplied to a rule-based classifier by the authors. The scientists used a precision performance metric to evaluate their classifier in the lab and attained a precision of 73%.

Few researchers have used machine learning to detect hate speech automatically in recent years. To construct the numeric feature vectors, they used unigram in conjunction with the TFIDF feature representation technique. The features were input into four machine learning classifiers: Naïve Bayes (NB), rule-based, J48, and Support Vector Machine (SVM). The rule-based classifier beat the NB, J48, and SVM classifiers in their experiments with 73% accuracy.

To classify hate speech communications, some researchers compared three feature engineering techniques and eight machine learning algorithms. The experimental results revealed that when bigram features were represented using TF-IDF, they performed better than word2Vec and Doc2Vec features engineering techniques.

Several teams submitted runs utilizing classical classifiers such as SVM, Logistic Regression, and Random Forest employing word or character n-gram representations weighted by some TF-IDF score in the first edition of the HASOC [18]. The majority of HASOC 2019 submissions use deep neural models such as LSTM, CNN, On-LSTM, and capsule GRU. The most popular text representation approaches utilized by the participants were Word2Vec, Glove, and fastText pre-trained vectors. Contextual language models, such as BERT [20] and ELMo [21], have shown promising results at different evaluation forums such as Offenseval [15] or HateEval [20]. The results and findings of HASOC 2021 can be found in the overview paper by Mandl et al. [22].

3. Task Description

The format of the text in the training set shows the pattern as follows:

_id, tweet_id, text, task_1, task_2

Where _id is a progressive number denoting the text within the dataset, tweet_id is the id of a particular tweet on Twitter, the text is the given post, and the task_1, task_2 columns contain the labels of type of speech. And the test set only includes _id, tweet_id, and the text columns.

3.1. Subtask A

Subtask A offered for English, German, and Hindi datasets focus on hate speech and offensive language identification. It is a coarse-grained binary classification in which teams must categorize the given tweets into two classes, as follows:

1. Hate and Offensive (HOF) - This text contains hateful, offensive, or profane language.
2. Non-Hate and Offensive (NOT) - There is no hate speech or offensive content in this text.

3.2. Subtask B

Subtask B involves fine-grained classification. Posts tagged as Hate and Offensive in Subtask A are further classified into three categories as follows:

1. Hate Speech (HATE) - Posts that make hateful remarks about someone based on their race, political beliefs, sexual orientation, gender, socioeconomic standing, health condition, or similar.

   Example: अब गांडीव उठाने का समय आ गया है प्रभुजी अधर्मियों का विनाश का समय आगया है अभी नही तो कभी नही
   Translation: Now the time has come to lift the Gandiva, Lord, the time has come for the destruction of the unrighteous, now or never.

2. Profanity (PRFN) - Posts that contain unacceptable language that isn't always insulting. This usually refers to the use of abusive language and cursing.

   Example: मीडिया दोगला है देश विरोधी है और अपने मालिक के सिर्फ तालवें चाटना जनता है ।
   Translation: The media is crazy, it is anti-national, and people only lick the palate of their master.

3. Offensive speech (OFFN) - This category includes posts that are humiliating, dehumanizing, insulting an individual, or threatening with violent acts.

   Example: गांधी खानदान को गांधी की पड़पोती ने सिद्ध किया बस सात साल जेल,यहां क्यों बेल पर घूम रहे है कमीने?
   Translation: Gandhi's great-granddaughter proved the Gandhi family, just seven years in jail, why are bastards roaming here on bail?

The posts that do not contain any Hate and Offensive (HOF) content are labeled as NONE. In the given dataset, most of the posts belong to the NONE category; some are HATE, whereas the other two classes are less frequently seen.

4. Methodology

This section discusses the proposed system that we have implemented to classify the given Hindi tweets into two different classes for Subtask A and four classes for Subtask B. We have used various feature extraction techniques such as TF-IDF, Bag of words and Word2Vec in combination with four different classifier algorithms. Figure 1 shows the complete system overview of the implemented system implemented for the experiments.

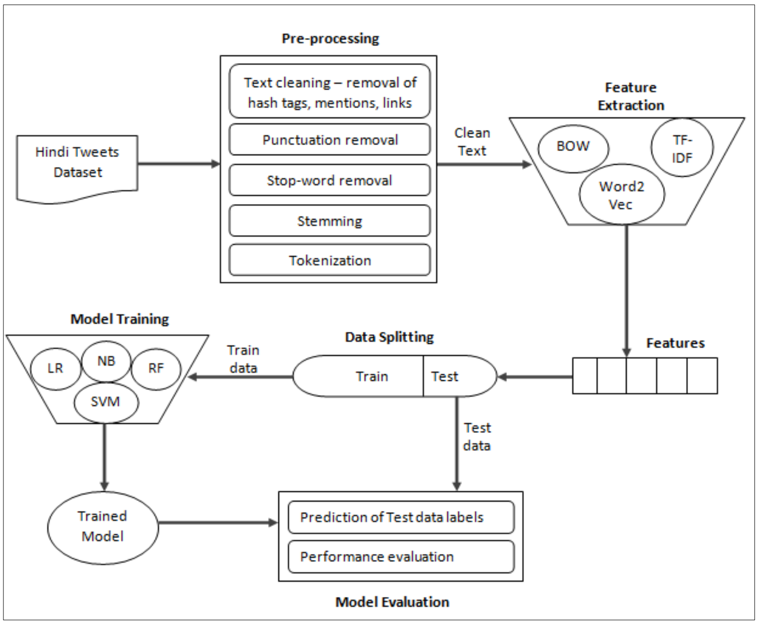

Figure 1: System Overview

4.1. Dataset Description

The dataset used in this research study was provided by HASOC for the shared task of hate speech detection in Indo-Aryan Language tweets. This dataset comprised monolingual Hindi tweets, collected from Twitter, labeled into two categories, namely, 'Hate and Offensive (HOF)' and 'Not-Hate-Offensive (NOT).' The dataset consisted of a total of 4594 tweets, of which 3161, i.e., about 69% belonged to the class NOT, and 1433, which is about 31%, were of the class (HOF). Out of the tweets labeled as 'HOF', about 15% were of the class' Offensive (OFFN)', around 13% of class 'HATE', and only 5% were labeled 'Profane (PRFN)'. A pictorial representation of this is given in Figure 2.

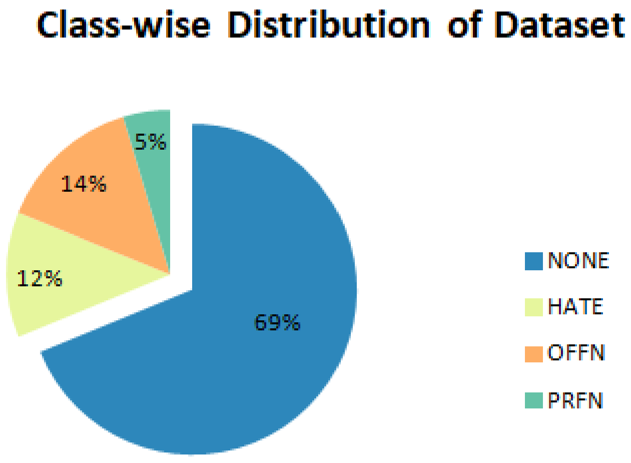

Figure 2: Class-wise Data Distribution

4.2. Pre-Processing

It has been proven in various studies that using text preprocessing makes natural language processing easier and provides better results. Therefore, we have implemented different preprocessing techniques step-by-step to filter out the unnecessary non-informative data. Since the dataset is obtained from Twitter, we have removed all the hashtags, URLs, and user mentions. Also, we have removed punctuations and stop-words, which are not very useful while analyzing a corpus. Since the dataset is entirely in Hindi, we have used regular expressions and pattern matching for most of the cleaning process. Besides cleaning, we have also performed stemming and tokenization of the individual tweets to get a clearer corpus. Stemming removes suffixes and converts individual words to their root forms. In contrast, tokenization breaks down the sentences into various tokens or words. Lastly, to ensure a smooth training process, null values have been removed from the dataset.

4.3. Feature Extraction

Working with datasets containing hundreds (or even thousands) of characteristics is becoming increasingly prevalent. When the number of features in the dataset approaches (or exceeds!) the number of observations contained in the dataset, a Machine Learning model is likely to suffer from overfitting. It is vital to use either normalization or dimensionality reduction techniques to avoid this type of difficulty.

Feature Extraction is a technique for reducing the number of features in a dataset by generating new ones from existing ones (and then discarding the original features). The original set of features should then summarize the majority of the information in the new reduced set of features. From a combination of the original set, a summarized version of the original features can be generated. It can also help to reduce the amount of redundant data in a study. Furthermore, the reduction of data and the machine's efforts in constructing variable combinations (features) speed up the learning and generalization stages of the machine learning process.

For our experiments, we have vectorized the text column of the dataset using different techniques. Also, for Subtask B, we have given the output labels of Subtask A as an input feature to the classifier along with the featured text for better learning of the classification model. These two features, for Subtask B, have been concatenated and passed as a single input to the machine learning models. The three feature extracting approaches that we have used are discussed below.

4.3.1. Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency was the first feature extraction method we used. This strategy is based on frequency. When a corpus is considered, this strategy works by reducing the weight of words that frequently appear in all documents (these words are also known as stop words) and raising the weight of terms that frequently occur only in a subset of the documents. The weight of TF-IDF is determined by the two components, TF (Term Frequency) and IDF (Inverse Document Frequency). Term Frequency is a metric for determining how frequently a term appears in a document.

Take a look at a set of N documents. Define f_{ij} as the number of times the word or term i appears in the document j. The term frequency tf_{ij} is thus defined as follows:

$$tf_{ij} = \frac{f_{ij}}{max_k f_{kj}} \tag{1}$$

The term frequency of word I in document j is tf_{ij} normalized by dividing it by the maximum number of occurrences of any term, excluding stop words, as shown in the above equation. The IDF for a given term is as follows:

$$idf_i = log(\frac{N}{n_i}) \tag{2}$$

Therefore, the TF-IDF score for term i in document j is calculated as:

$$tf - idf = \ tf_{ij} * \ idf_i \tag{3}$$

4.3.2. Bag of Words (BOW)

Bag of words is a text modeling technique. In technical terms, we can call it a method for extracting features from text data. This method of extracting features from documents is easy and adaptable. It is a text representation that describes the frequency of words appearing in a document. We only keep track of word counts and don't pay attention to grammatical subtleties or word arrangement because any information about the sequence or structure of words in the document is deleted. It is referred to as a "bag" of words. The model simply cares about whether or not recognized terms appear in the document, not where they appear.

One of the most significant issues with text is that it is messy and unstructured; machine learning algorithms prefer structured, well-defined fixed-length inputs. We can turn variable-length texts into fixed-length vectors using the Bag-of-Words technique.

4.3.3. Word2Vec

Word2Vec is a statistical method for learning a solitary word embedding from a text corpus quickly and efficiently. It was created by Tomas Mikolov et al. at Google in 2013 as a reaction to improving the efficiency of neural-network-based embedding training. It has since become the de facto standard for generating pre-trained word embedding. It's a hybrid of two models for grouping related words-: the Continuous Bag-of-Words and the Skip-gram models. A layer of multi-set of words, also known as a bag of words, is used as input to the CBOW model. In the concealed layer, these words are projected in a linear pattern. Because the hidden layer projects all of the words, the output of the hidden layer is the average of vectors.

For our experiments, we have custom-trained the Word2vec model for the given Hindi dataset. We have used the Skip-gram model with the hyperparameters such as vector length set to 200, context window set to 10, and the min_count to 2 to preserve the sentences' contextual meaning during the training process. This gave us a vocabulary of 7078 words which were then used to generate sentence embeddings based on the word embedding provided by the word2vec model and the total number of vocabulary words present in the tokenized sentence.

4.4. Train-Test Split

For training and testing purposes for both tasks A and B, we have split the cleaned dataset into a 4:1 ratio, i.e. 80% for Train Data and 20% for Test Data. The classification model is trained to learn classification rules using the training data. In contrast, the test data is used to validate the learning of the classification model. Table I presents the class-wise distribution of the dataset before and after splitting it into train and test sets.

Table 1
Data Split Task 1

	Class	Total instances	Train instances	Test instances
0	HOF	1426	995	431
1	NOT	2937	2059	878
	Total	**4363**	**3054**	**1309**

Table 2
Data Split Task 2

	Class	Total instances	Train instances	Test instances
0	HATE	563	455	108
1	NONE	2937	2346	591
2	OFFN	650	519	131
3	PRFN	213	170	43
	Total	**4363**	**3490**	**873**

4.5. Classification Models

It is seen that there is no single classifier that performs best on different kinds of datasets. Hence, it is always suggested that one should apply several different classifier algorithms on different feature vectors obtained using various feature extraction techniques and observe which combination gives the best results. Therefore, we have selected four different classifiers, namely, Logistic Regression (LR), Naïve Bayes (NB), Support Vector Machine (SVM), and Random Forest (RF), to go with the three feature vectorization methods discussed previously.

4.5.1. Logistic Regression

Logistic Regression is a binary classification probabilistic model of machine learning. It predicts independent data variables by analyzing the relationship between one or more existing independent variables. It uses the sigmoid function with the Bernoulli distribution as follows:

$$F(x) = \frac{1}{1+ e^{-(\beta_0+\beta_1 x)}} \tag{4}$$

4.5.2. Naïve Bayes

It is a classification algorithm that uses the "Bayes theorem" to calculate probabilities to predict the class. It works on conditional independence among features. The conditional probability of the Bayes theorem is as follows:

$$p(C_k \mid x) = \frac{p(C_k)p(x \mid C_k)}{p(x)} \tag{5}$$

4.5.3. Support Vector Machine (SVM)

It is a supervised learning algorithm used for classification and regression. It creates a hyperplane in multidimensional space by choosing the extreme points to separate different classes. These extreme points are called support vectors.

4.5.4. Random Forest

It is a type of ensemble supervised learning algorithm consisting of many decision trees. It is used for regression as well as classification. It constructs a decision tree and predicts the result, and performs a vote on each predicted result. The prediction with the most votes is the final prediction.

4.6. Evaluation Measures

A classifier's performance is calculated by evaluating four factors – True Positives (TP), False Positives (FP), True Negatives (TN), and, lastly, False Negatives (FN). A few commonly used metrics calculate the performance of a classifier based on these factors, namely Precision, Recall, Accuracy, and F1 score.

- **Precision**
  Precision is the percentage of predicted positives that turn out to be true positives. It is also known as the positive predictive value.
  $$Precision = \frac{TP}{(TP+FP)} \tag{6}$$

- **Recall**
  Recall gives the percentage of all positive cases, which measures the classifier's ability to detect positive samples.
  $$Recall = \frac{TP}{(TP+FN)} \tag{7}$$

- **Accuracy**
  It's the total number of instances that have been correctly categorized.
  $$Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)} \tag{8}$$

- **F1 Score**
  The F1 Score is the harmonic mean of precision and recall. The classical F1 score gives equal weight to both precision and recall.
  $$F - score = 2 * (\frac{Precision*Recall}{Precision+Recall}) \tag{9}$$

Since Task B is a multi-classification problem, we have used the **Macro-averaged F1 Score** for evaluation. For single-label multi-class situations, this is the default aggregation technique for the F1-score. In this report, both the subtasks, A and B, have the same evaluation metrics.

$$Macro_P = \frac{\sum Precision}{n} \qquad Macro_R = \frac{\sum Recall}{n} \tag{10}$$

$$Macro_F = \frac{Two \times Macro_P \times Macro_R}{Macro_P + Macro_R} \tag{11}$$

5. Results

The results of the various analyses performed for this report are presented in Table 3 to Table 6. Most of the models performed well and gave similar results with very slight differences between them. For Task A, it is observed that the LR model performed almost similarly with both TFIDF and BOW features. Logistic Regression, by default, is a two-class classification method. Therefore it has been excluded for the Subtask B experiments, which consists of multiple classes. NB, however, shows a deficient performance with Word2Vec embeddings. On the other hand, the TFIDF vectorizer performs similarly with both SVM and RF classifiers for Task B. The high accuracy scores and low macro F1 scores result from the skewness in the given dataset, as seen in sections 4.1 and 4.4.

Table 3
Macro F1 score for Task A

	Logistic Regression	Naïve Bayes	SVM	Random Forest
TFIDF	0.67	0.68	0.68	0.65
BOW	0.68	0.67	0.67	0.67
Word2Vec	0.68	0.64	0.69	0.67

Table 4
Accuracy for Task A

	Logistic Regression	Naïve Bayes	SVM	Random Forest
TFIDF	0.75	0.73	0.74	0.74
BOW	0.74	0.73	0.72	0.73
Word2Vec	0.75	0.65	0.75	0.75

Table 5
Macro F1 score for Task B

	Naïve Bayes	SVM	Random Forest
TFIDF	0.61	0.63	0.63
BOW	0.63	0.63	0.61
Word2Vec	0.56	0.61	0.59

Table 6
Accuracy for Task B

	Naïve Bayes	SVM	Random Forest
TFIDF	0.86	0.86	0.87
BOW	0.85	0.85	0.85
Word2Vec	0.84	0.85	0.87

6. Conclusion

Hate speech has become much more common on social media in recent years, which might significantly impact society in the coming years. Various studies have been done on automatic hate speech detection in social media text. However, most of the studies are limited to the English language. To monitor Indian social media feed, Hindi hate speech detection must be explored. Hence, to address this problem, we have conducted a study and explored the classification of Hindi text using some statistical models. We have used Feature Extraction techniques like TF-IDF, Word2Vec, and Bag-of-Words (BOW) and different classification models like Logistic Regression, Naïve Bayes, Support Vector Machine, and Random Forest for Subtask A, which was to classify tweets into Non-Hate-Offensive, Hate and Offensive tweets.

Then, using the labels obtained from Subtask A, we used a similar approach to do the further fine-grained classification of Hate-speech and Offensive posts from Subtask A to three specific classes namely, Offensive, Profane, and Hate, as a part of Subtask B.

Our findings show a good performance in the scores of Subtask A. However, Subtask B requires some more study using more experiments based on specific classes using different machine learning approaches that can analyze text in more depth.

References

[1] S. Abro, Z. S. Shaikh, S. Khan, G. Mujtaba, Z. H. Khand, Automatic Hate Speech Detection using Machine Learning: A Comparative Study, in: Machine Learning vol. 10 no. 6, 2020.

[2] N. D. Gitari, Z. Zuping, H. Damien, J. Long, A lexicon-based approach for hate speech detection, in: International Journal of Multimedia and Ubiquitous Engineering vol. 10 no. 4, 2015.

[3] S. Tulkens, L. Hilte, E. Lodewyckx, B. Verhoeven, W. Daelemans, A dictionary-based approach to racism detection in dutch social media, in: arXiv preprint arXiv:1608.08738, 2016.

[4] E. Greevy, 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).

[5] I. Kwok, Y. Wang, Locate the hate: Detecting tweets against blacks, in: Twenty-seventh AAAI conference on artificial intelligence, 2013.

[6] S. Sharma, S. Agrawal, M. Shrivastava, Degree based classification of harmful speech using Twitter data, in: arXiv preprint arXiv:1806.04197, 2018.

[7] S. Malmasi, M. Zampieri, Detecting hate speech in social media, in: arXiv preprint arXiv: 1712.06427, 2017.

[8] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, Y. Chang, Abusive language detection in online user content, in: Proceedings of the 25th international conference on world wide web, 2016 (pp. 145-153).

[9] Z. Waseem, 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).

[10] K. Dinakar, R. Reichart, H. Lieberman, Modeling the detection of textual cyberbullying, in: Fifth international AAAI conference on weblogs and social media, 2011.

[11] S. Liu, T. Forss, Combining N-gram based Similarity Analysis with Sentiment Analysis in Web Content Classification, in: KDIR, 2014 (pp. 530-537).

[12] S. Köffer, D. M. Riehle, S. Höhenberger, J. Becker, Discussing the value of automatic hate speech detection in online debates, in: Multikonferenz Wirtschaftsinformatik (MKWI 2018): Data Driven X-Turning Data in Value, Leuphana, Germany, 2018.

[13] S. Modha, T. Mandl, G. K. Shahi, H. Madhu, S. Satapara, T. Ranasinghe, M. Zampieri,Overview of the HASOC Subtrack at FIRE 2021: Hate Speech and Offensive ContentIdentification in English and Indo-Aryan Languages and Conversational Hate Speech, in: FIRE 2021: Forum for Information Retrieval Evaluation, Virtual Event, 13th-17th December 2021, ACM, 2021.

[14] P. Burnap, M. L. Williams, Us and them: identifying cyber hate on Twitter across multiple protected characteristics, in: EPJ Data Science vol. 5, Springer, 2016 (pp. 1-15).

[15] A. Jiang, QMUL-NLP at HASOC 2019: offensive content detection and classification in social media, in: Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation, 2019.

[16] A. Aggarwal, A. Wadhawan, A. Chaudhary, K. Maurya, “Did you really mean what you said?": Sarcasm Detection in Hindi-English Code-Mixed Data using Bilingual Word Embeddings, in: arXiv preprint arXiv:2010.00310, 2020.

[17] S. G. Roy, U. Narayan, T. Raha, Z. Abid, V. Varma, Leveraging multilingual transformers for hate speech detection, 202.

[18] S. Modha, T. Mandl, P. Majumder, D. Patel, Tracking hate in social media: Evaluation, challenges and approaches, in: SN Computer Science vol. 1 no. 2, Springer, 2020 (pp.1-16).

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

[20] V. Basile, C. Bosco, E. Fersini, N. Debora, V. Patti, F. M. Pardo, P. Rosso, M. Sanguinetti et al., Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter, in: 13th International Workshop on Semantic Evaluation, ACL, 2019 (pp. 54-63).

[21] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: arXiv preprint arXiv:1802.05365, 2018.

[22] T. Mandl, S. Modha, G. K. Shahi, H. Madhu, S. Satapara, P. Majumder, J. Schäfer, T. Ranasinghe, M. Zampieri, D. Nandini, A. K. Jaiswal, Overview of the HASOC subtrack at FIRE2021: Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages, in: Working Notes of FIRE 2021 - Forum for Information Retrieval Evaluation, CEUR, 2021. URL:http://ceur-ws.org/.