

Hate Speech and Offensive Content Identification in Hindi and Marathi Language Tweets using Ensemble Techniques

Ratnavel Rajalakshmi¹, Faerie Mattins¹, S Srivarshan¹, L Preethi Reddy¹ and Anand Kumar M.²

¹*School of Computer Science and Engineering, Vellore Institute of Technology, Chennai*

²*Department of Information Technology, National Institute of Technology Karnataka (NITK), Surathkal*

Abstract

Hate Speech is described as any form of speech in which speakers attempt to ridicule, humiliate, or inculcate hatred in someone else's minds based on characteristics such as religion, the colour of skin, race, or sexual preference. In recent years, social networking sites have been a major source of excessive amounts of hate speech. If unaddressed, these might cause anxiety and despair in the affected individuals or groups. As a result, the above-mentioned social networks utilize an assortment of algorithms to identify such hate speech. Detecting Hate Speech in English texts has been one of the hottest topics in recent years, with multiple types of research being published. However, in regional and indigenous languages, hate speech detection is a recent area with not much research being conducted. It is difficult to perform hate speech detection using data in regional languages due to a lack of large enough training data and a lack of resources about that domain. The HASOC [1] 2021 Hate Speech Detection Task solves one of the problems. It provides a dataset containing Tweet data in English, Hindi [2] and Marathi [3] languages. There were two subtasks as part of the main task. The subtask was to classify the hate speech and offensive texts in the Hindi and Marathi tweet dataset as Hate Speech (HATE), Offensive (OFFN) or Profane (PRF). This work compares the performance of different models on both subtasks and provides a conclusion on the best performing model. The Random Forest Classifier reports the most remarkable accuracy on the first subtask with a macro F1 score of 75.19% and 73.12% on the Marathi and Hindi tweet datasets. The XGBoost algorithm is the best performing algorithm on the second subtask with a 46.5% macro F1 score. Overall any of these models can get satisfactory results when dealing with hate speech detection in regional language. This work has been submitted to the FIRE2021 shared task, Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages (HASOC-2021) by team DLRG.

Keywords

Hate speech detection, Multilingual tweets, Machine learning, XGBoost, Majority voting, Random forest

1. Introduction

Hate speech is defined as any communication, whether spoken, written, or physical, that criticises or uses discriminating, or disparaging terminology about a person (or group) based on

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

✉ rajalakshmi.r@vit.ac.in (R. Rajalakshmi); faeriemattins.r2019@vitstudent.ac.in (F. Mattins);


srivarshan.2019@vitstudent.ac.in (S. Srivarshan); preethireddi17@gmail.com (L. P. Reddy);

m1_anandkumar@nitk.edu.in (A. K. M.)

🆔 0000-0002-6570-483X (R. Rajalakshmi)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

who they are, such as colour, religion, nationality, ethnicity, race, sexuality, descent, or another identifying feature. Nowadays, social media has evolved into a popular forum for ordinary people to share their feelings on any topic. It has its own set of advantages and disadvantages. Everyone understands their point of view and perceives the world in a new light when they express themselves. However, this might also imply that people have the right to share whatever they wish, even offensive content that is harsh and upsetting to people. It impacts not only the feelings of that particular group but also their self-esteem and dignity. As a result, maintaining a courteous demeanour while portraying one's viewpoint on any social media site is essential. Unfortunately, preventing people from posting or tweeting hate speech on social media is quite tricky. Thence, every social media company globally is attempting to build a revolutionary approach for detecting and preventing hate speech. It has been done to some level in English, as English is a widely known and spoken language worldwide. However, this is not the case in Indian regional languages such as Hindi, Marathi, Tamil, and others. Therefore, it is critical to have a solution in place.

Machine learning is an application of data analysis that facilitates the creation of analytical models. Artificial intelligence is predicated on the idea that computers can recognise patterns, learn from data, and make decisions without the need for humans. Machine learning has been utilised to solve problems in many science and technology domains, including social media. Many academics are now working on multiple machine learning models to detect highly accurate hate speech and an optimised model. However, there has not been much advancement in the subject of detecting hate speech in regional languages. Because it is challenging to train a model in a language that isn't widely spoken and hence lacks the necessary dataset and procedures to train the model effectively. In this study, data embedding and multiple classifier models were employed to determine whether or not a tweet contains hate and offensive content in two Indian regional languages, namely Hindi and Marathi. This research is split into two sections: subtask A and subtask B. Binary classification models were used to determine if the tweet contained hate speech in Hindi and Marathi. The multiclass classification was used for subtask B to classify hate speech further, and they are: hate speech, profane or offensive content in Hindi. Our team DLRG participated in Subtask A-Marathi, Subtask-A and B for Hindi.

In section 2 of this paper, the related works have been analysed and discussed. In section 3, the dataset utilised in both subtasks is discussed. Section 4 includes an explanation of the proposed methodology. In section 5, the findings are presented and discussed, and in section 6, the conclusion and future work are presented.

2. Related Work

In earlier works, different term weighting methods have been applied for web page categorization using TF-IDF based methods [4]. In [5], the authors used unigrams and to detect the tweet is offensive or not(binary classification) ,patterns are automatically collected from the training set. These patterns and unigrams are later used, among others, as features to train a machine learning algorithm. In this method, they have not considered platforms like Facebook. In

[6], traditional machine learning techniques were applied with SVM for URL based web page classification. In many recent works, the deep learning algorithms such as Convolutional Neural Networks, Recurrent Neural Networks are explored for various applications [7, 8]. Sentiment analysis in English is a most common problem and many approaches are suggested by various researchers by applying both machine learning and deep learning techniques [9, 10]. This paper projects the various works submitted as part of the HASOC FIRE'20 track[11]. Ensemble based approaches were also reported in [12] for offensive speech detection. A novel relevance factor has been proposed in [13] to address the challenges in code-mixed Hindi-English tweets. In [14], Hindi English Code Mixed Hate Speech Detection is performed using Character Level Embedding. The authors have used different deep learning models, in which Attention Model with GRU Hybridisation performed the best. However, instead of word embeddings, the authors have used Character-Level Embeddings which made model adaptable to the common defect. In [15], the authors applied the Multi Input Multi Channel Transfer Learning based model to detect hate speech or abusive Hinglish tweets from the Hinglish Offensive Tweet (HOT) dataset. The authors used transfer learning with multiple feature inputs. They achieved peak performance using the SVM model. In [16], this paper explains an increment to the state of the art in hate speech detection for Hindi English code mixed tweets. The authors compared three typical deep learning models using domain-specific embeddings. However, the proposed method does not take consideration into code mixed data. In [17], the authors performed hate speech detection on datasets of two Dravidian languages that are Tamil and Malayalam. They obtained an F1 score 0.77, and similarly, for the Tamil language, the best performing model obtained an F1 score of 0.87.

3. Dataset

The dataset was obtained as part of the HASOC '21 competition at FIRE 2021. These datasets consist of tweets sampled from the Twitter social media platform. The data set consisted of English, Hindi and Marathi tweets. For this work, the English tweets were disregarded. The hashtags, URLs and keywords prevalent in regular tweets are also present in this data. For Subtask A, the dataset was broadly divided into two categories, NOT tweets that contained non-offensive content and HOF tweets that contained offensive or profane content. The HOF tweets in the Hindi tweet data were further divided into Hate Speech (HATE), Offensive (OFFN) and Profane (PRF) as part of Subtask B. The dataset was in a CSV file with each entry containing the ID, Tweet ID, Text and corresponding labels. The Hindi tweet data consisted of 4594 entries, while the Marathi data consisted of 1874 entries.

4. Methodology

The proposed methodology for all the subtasks are detailed in the following sections.

4.1. Preprocessing

In the preprocessing of the data, the first step performed was removing noise. The URL links and Twitter handle names were the primary sources of noise in the dataset. All the words are removed that followed @ symbol using regular expressions. After the above step, Twitter handle names are removed and to remove URL links, regular expressions were used to remove anything that followed HTTP/HTTPS. Then the stop words like a, an, the, is, etc., followed by punctuation marks and special characters, are removed. These stop words are not necessary to find the statement's sentiment as they do not carry any significant meaning. The punctuation and special characters cannot be used to identify the sentiment.

We removed these as they are unnecessary, and we can work solely on the essential characters and words for better performance. Further, Stemming is performed to reduce the inflection in words, and the words can be brought to their respective base forms. Stemming helps deal with spelt words and remove unwanted suffixes, which is very common in a social media text. We used a snowball stemmer imported from the NLTK package, and it is based on a programming language called "Snowball". Finally, we removed high-frequency words. Based on the term frequency method, we removed the most frequently occurred 1000 words.

4.2. Feature Extraction

The Scikit-Learn [18] library has several vectorizers to process and tokenize text in the same function, and it involves converting word characters into integers. After the preprocessing of data, the clean tweets are obtained. By treating these as a sequence of words, the features were extracted. We used a Count vectorizer for feature extraction and Machine Learning models.

Countvectorizer: We made a loop over n_gram to create unigram, bigram, trigram till the value of n to 5. The Countvectorizer tokenizes the text by breaking down a sentence into words, and that vocabulary is used to encode new texts. For the n-gram of 1 to 5, the texts are broken down into words with the integer count for the number of times it appeared and is returned in the encoded vector.

4.3. Classifiers

The process by which a group of data is divided into a set of categories is known as classification. This process can be performed on both unstructured and structured data. The most crucial step in this process is predicting the category by performing some calculations on the data points provided. This proposed model uses the following classifiers: Logistic Regression, Support Vector Machine, Stochastic Gradient Descent, Random Forest, Ensemble, and XGBoost.

4.3.1. Binary Classification

Classifying the labels of a collection into two groups using a classification model is known as binary classification. In most binary classification problems, one class represents the normal state, and the other represents the aberrant state. The regular state class is assigned the class

label 0, while the aberrant state class is labelled 1. Logistic Regression, k-Nearest Neighbours, Decision Trees, Support Vector Machines, and Naive Bayes are the most prominent binary classification techniques. In this research, subtask A consists of binary classification with the labels Hate and offensive (HOF) speech or Not hate speech (NONE).

4.3.2. Multiclass Classification

Classifying the labels of a collection into three or more groups using a classification algorithm is known as multiclass classification. In most multiclass classification problems, the different classes represent the various labels that are to be assigned to each set of input variables. Generally, when label encoding is done for multiclass problems, each class would be converted into a binary number. This method of having multiple output variables instead of multiple outputs simplifies the process of training. Logistic Regression, Support Vector Machines, K Nearest Neighbours, Naive Bayes and Decision Trees are some of the most prominent classification techniques that can also be extended to multiclass problems. In this research, subtask B consists of multiclass classification with the labels Hate Speech (HATE), Offensive (OFFN) and Profane (PRF).

4.4. Stochastic Gradient Descent

Gradient descent is an iterative procedure that begins at a random position on the slope of a function and gradually falls until it reaches its lowest point. Stochastic Gradient Descent is an optimisation approach that takes one data point at random from the entire data set at each iteration to minimise computations drastically. SGD is an easy algorithm to implement as well as an efficient method to find the minimum error point. Because of the randomness of SGD's descent, it usually takes more iterations to reach the minima. Even though it takes more rounds to reach the minima than normal Gradient Descent, it is computationally considerably less expensive. Stochastic Gradient Descent has been used in both subtasks A and B. This model was chosen due to the fact that, it takes very little amount of time to train the model. That and the reason that this model is more likely to reach the local minima of the error function than other models, makes it an interesting prospect. Python's Scikit-learn package loads the SGD model. After that, the model is trained on the training data, and the results are verified and documented.

4.5. Logistic Regression

This is a popular machine learning algorithm that has seen its fair share of usage as a classifier. The base for Logistic Regression can be traced to its conception from statistical mathematics. The model is trained on data where the dependent variables are categorical. It is derived from the Linear Regression model. The output obtained from the Linear Regression Model is fitted into a Logistic Function, which predicts the target variable. This model makes use of a decision boundary. It sets a threshold that differentiates one class of variables from another. The linearity or the non-linearity of the decision boundaries depend on the input variables. The activation function used is a Sigmoid activation function. This constraint the output obtained between the numbers 0 and 1. Hence, the output becomes an estimated probability, which gives the final

class prediction when subjected to the decision boundary.

This model was chosen due to the fact that it is one of the most simple and basic classifiers that can be implemented and it would provide a good base for comparison against the other approaches. The Scikit-Learn [19] library for Python provides a Class for initializing a Logistic Regression Model. It also provides functions for training the model and obtaining predictions from it. The vectorized input variables and the corresponding labels constitute the training data on which the Logistic Regression model is loaded and trained. After training the model, it is validated using the validation data set, and the results are recorded.

4.6. Support Vector Machine

A support vector machine (SVM) is a type of machine learning technique that evaluates data for categorization and regression analysis. A supervised learning algorithm called a support vector machine sorts data into two groups. An SVM algorithm's job is to figure out which category a new data point fits in. As a result, SVM is classified as a non-binary linear classifier. An SVM's output is a map of the sorted data with a separation between both sides. The SVM method represents each piece of data as a position in n-dimensional space, where each feature is the value of the specified coordinate. Later, the classification process is then completed by selecting the hyper-plane that best differentiates the 2 classes. In the SVM classifier, creating a linear hyper-plane between these two classes is easy. The SVM kernel is a function that transforms a low-dimensional input space into a higher-dimensional space to convert a non-separable problem into a separable one. SVM is efficient in high-dimensional spaces and works well with a clear separation margin. Here in this research, SVM is used in both subtasks. This model was chosen for the reason that it works extremely well in cases where there are a large number of features when compared to the training set. This comes in handy when dealing with data vectorized with TF-IDF where a large number of features are extracted. It also is highly memory efficient. Python's Scikit-learn package is used to load the SVM model. After that, the model is trained on the training data, and the results are verified and documented.

4.7. Ensemble - Majority Voting

When individual classifiers do not perform well on a given dataset, it is common practice to combine the models in a process called Ensembling. In most cases, this combined model would perform as well as or even better than the individual classifiers. For t One such Ensembling technique explored for this task was Majority Voting. It is an ensemble technique where the predictions from various weak classifiers for a given set of input variables are considered votes. These votes are tallied together, and the final outputs are obtained from the tallied votes. In the current Binary Classification problem, the same process occurs. The predictions of all the weak learners are considered as votes. The number of votes obtained by every output class are tallied. The output class that gather the highest number of votes will be considered the output for that particular set of input variables.

The combining classifiers used for this task are Support Vector Machine, Logistic Regression,

Stochastic Gradient Descent, K Nearest neighbours, Naive Bayes, Random Forest and Decision Tree. Instances of each of the above models are initially trained on the Vectorized input data and their corresponding labels. Then the validation data is passed to all seven models, and the predictions are obtained. Majority voting is performed on these predicted labels, and the final predicted labels from Ensembling are obtained. These predicted labels are compared with the actual labels to obtain the performance metrics of the model. A similar process is also carried out for the test dataset.

4.8. XGBoost

XGBoost is a popular Machine Learning algorithm which works as decision tree-based ensemble using a gradient boosting framework. The XGBoost algorithm takes a text string as input and loads an XGBoost model trained to predict its label. Bagging and boosting are the most commonly used ensemble learning techniques. The XGBoost classifier is robust and even in distributed environments XGBoost yields efficient results. In the Scikit-learn framework, the algorithm provides a wrapper class by allowing models to be treated like classifiers or regressors. With this we can use the entire sci-kit-learn library with XGBoost models. The ensemble tree methods such as XGBoost and Gradient Boosting Machines use gradient descent architecture to boost weak learners principles. This model was chosen as a fact that it works well with large number of texts in training dataset. As an advantage, At each iteration the XGBoost algorithm has a built in method, which is cross validation and it is made easy by not specifying the exact count of boosting iterations required during a single run.

4.9. Random Forest

Random Forest is an advanced machine learning technique that may be applied to various tasks, including regression and classification. It is an ensemble method, which means a random forest resulting in a combination of many small decision trees called estimators, each of which makes its predictions. To provide a more accurate prediction, the random forest model integrates the estimators' predictions. The problem of standard decision tree classifiers is that they are prone to overfitting the training set. The ensemble design of the random forest compensates for this and allows the random forest to generalise effectively to anonymous data, including data with missing values. Random forests can also handle enormous datasets with much dimensionality and several feature types. While expanding the trees, the random forest adds more randomness to the model. When splitting a node, it looks for the best feature from a random subset of features rather than the essential feature. As a result, there is much variety, leading to a better model. Random forest is used in both subtasks in this study. The Random forest model is loaded using Python's Scikit-learn package. The model is then trained using the training data, with the results being validated and reported.

5. Results and Discussion

5.1. Marathi - Subtask - A

A comparison was performed on various models that were trained on the Marathi dataset. The different classifiers used after cleaning and vectorising the Marathi tweet dataset are, Logistic Regression, XGBoost, Support Vector Machine, Stochastic Gradient Descent, Majority Voting and Random Forest. A comparison of the performance metrics of all the different classifiers used can be viewed in Table 2. As the table explains, the highest performance was obtained with the Random Forest classifier model. The model attained an F1 score of 0.7519. Out of all the classification models implemented, the Logistic regression model achieved minor performance with an F1 score of 0.6924. It can be explained because the Logistic regression algorithm does not work well with non-linear data. It also does not bode well with outliers in the training data. Since the data considered here is mostly text, it is valid to assume that it contains multiple cases of outliers being present in it.

The Logistic regression algorithm also finds it challenging to accommodate the distributed data, which might explain its diminished performance. As expected, the Majority Voting technique performs better than its weak constituent learners due to reasons explained in section 4.7. However, it can be observed that the usage of Random Forests improves upon this technique. It can be explained by the fact that Random Forest is by itself an Ensembling algorithm. Tree type algorithms are generally better than probabilistic models in handling outliers and noise in the input data. They are also less impacted by the noise. A unique feature of the Tree type models and incredibly Random Forests is the innate ability to handle missing data independently. The embedded data is also non-linear. This negatively affects all probabilistic models, as that is the area in which they thrive. Random Forests are not impacted by non-linear data as much as the probabilistic models. The comparison between all the different models used and the F1 score procured by them is illustrated in Figure 1.

5.2. Hindi - Subtask - A

A comparison of several models trained on the Hindi dataset was carried out. After cleaning and vectorising the Hindi tweet dataset, the following classifiers were used: XGBoost, Logistic Regression, Stochastic Gradient Descent, Support Vector Machine, Majority Voting, and Random Forest. Table 2 contains a comparison of the performance metrics of all the various classifiers utilised. The Random Forest classifier model produced the best results, as seen in the table. The model achieved an F1 score of 0.7312, and the XGBoost model performed the most unsatisfactory out of all the classification models tested, with an F1 score of 0.6628. This low performance could be explained because the boosting algorithm does not work well with distributed data. It's also bad news for anomalies in the training data. Given that the data in question is largely text, it's reasonable to presume that it contains several instances of outliers.

As predicted, the Majority Voting approach outperforms its weak learners, for reasons stated in section 4.7. However, it can be shown that using Random Forests improves on this method. It may be explained by the fact that Random Forest is an Ensembling algorithm in and of itself.

In general, tree-type algorithms outperform probabilistic models in dealing with outliers and noise in input data. They are also less influenced by noise. The capacity to manage missing data on its own is a specific property of Tree type models, particularly Random Forests. The embedded data is also non-linear and harms all probabilistic models because it is the domain in which they thrive. Random Forests are less affected by non-linear data than probabilistic models. Figure 2 depicts a comparison of all the different models employed and the F1 score obtained while using those classifiers.

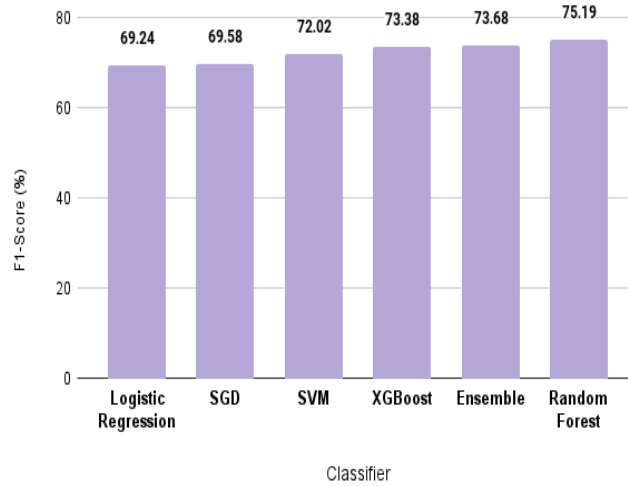


Figure 1: Performance of different classifiers for subtask A : Marathi

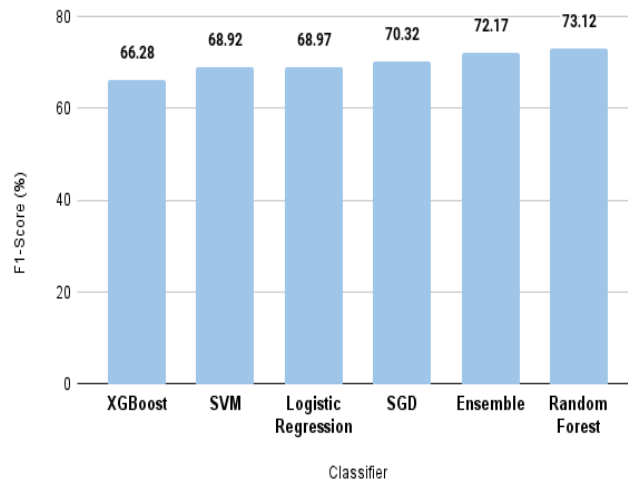


Figure 2: Performance of different classifiers for subtask A : Hindi

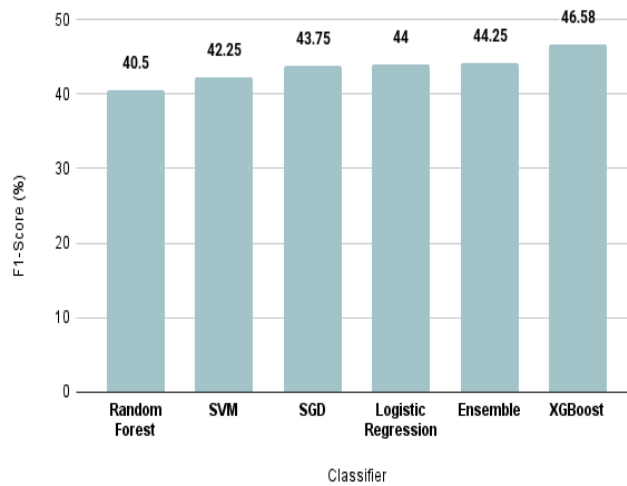


Figure 3: Performance of different classifiers for subtask B : Hindi

Table 1

The Submissions made for HASOC'21

Submission name	SubTask	Classifier	Macro F1-Score
dLRG_M_S1	Subtask A : Marathi	XGBoost	73.38%
HTA1	Subtask A : Hindi	XGBoost	66.28%
dLRG_HT2_S1	Subtask B : Hindi	XGBoost	46.58%

5.3. Hindi - Subtask - B

Several machine learning models were trained on the Hindi dataset and then compared. The input data was obtained after cleaning the tweets and vectorising them. The various machine learning algorithms used are XGBoost, Logistic Regression, Stochastic Gradient Descent, Support Vector Machine, Majority Voting and Random Forest. A comparison of performance metrics for all the above models is shown in Table 2. For the given subtask, XGBoost performed well with the highest F1 score of 0.4658, followed by the Ensemble model with an F1 score of 0.4425. The model with the lowest performance observed was the Random forest, with a macro F1 score of 0.405. This can be attributed to the fact that as the number of features increase, Random forest algorithm is more prone to overfitting. Following Random forest, the next least performance was showed by Support Vector machine with a macro F1 score of 0.4225. This performance achieved by Support Vector Machine can be explained because it does not perform well for large datasets or datasets with more noise. This is especially true with text data as it is prone to the presence of noise.

For multiclass classification, XGBoost performs well. At each iteration of the boosting process, the XGBoost algorithm runs cross validation, and in a single run the exact optimum number

Table 2

Performance Analysis - A comparative study of different classifiers for all the subtasks

SubTask	Classifier	Macro Precision	Macro Recall	Macro F1-Score
Subtask A : Marathi	SGD	70.43%	69.04%	69.58%
	Logistic Regression	78.17%	67.77%	69.24%
	SVM	77.33%	70.44%	72.02%
	Ensemble	74.69%	73.01%	73.68%
	XGBoost	74.84%	71.98%	73.38%
	Random Forest	76.55%	74.33%	75.19%
Subtask A : Hindi	SGD	72.40%	69.36%	70.32%
	Logistic Regression	76.08%	67.68%	68.97%
	SVM	76.37%	67.52%	68.92%
	Ensemble	74.90%	70.98%	72.17%
	XGBoost	67.96%	64.68%	66.28%
	Random Forest	76.98%	71.65%	73.12%
Subtask B : Hindi	SGD	45.25%	42.35%	43.75%
	Logistic Regression	52%	38.13%	44%
	SVM	43.5%	41.25%	42.25%
	Ensemble	51.75%	38.65%	44.25%
	Random Forest	52.25%	33.06%	40.5%
	XGBoost	54.92%	40.44%	46.58%

of boosting iterations are obtained. It leads to good results being obtained. The Majority Voting performed well after XGBoost due to the reasons explained in section 4.7. Therefore for the Hindi subtask B dataset, XGBoost produced the best performance and Random forest exhibited the least performance. Figure 3 illustrates the macro F1 scores obtained by different models in subtask B. Table 1 shows the various submissions made to the FIRE2021 shared task, Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages (HASOC-2021), along with their subtask names, classifiers and macro F1-scores. We have extended the experiments by applying other classifiers and tried to improve the performance, the final results are tabulated in Table 2.

6. Conclusion and Future Scope

This work was submitted to the FIRE2021 shared task, Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages (HASOC-2021). The challenge of recognising conversational hate-speech in Indian regional languages has been empirically studied in this research utilising binary and multiclass classifiers for Marathi and Hindi tweets. This research has projected a complete method from analysing the tweets to comprehending the results. In subtask A, Random forest showed the best Macro F1-score of 75.19% and 73.12% respectively for both Marathi and Hindi tweets. With a Macro F1-score of 46.58% in subtask B for Hindi tweets, it is evident that XGBoost produced the most significant results. Overall, it can be inferred that Random Forest works well for binary classification, whereas XGBoost is a better classifier

model for multiclass classification. In this work, only one type of feature extraction method has been implemented. In the future, multiple other types of embeddings can be implemented to get a better model by combining it with deep learning algorithms.

References

- [1] 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 Content Identification 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.
- [2] 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 FIRE 2021: 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/>.
- [3] S. Gaikwad, T. Ranasinghe, M. Zampieri, C. M. Homan, Cross-lingual offensive language identification for low resource languages: The case of marathi, in: Proceedings of RANLP, 2021.
- [4] R. Rajalakshmi, Supervised term weighting methods for url classification, Journal of Computer Science 10 (2014) 1969–1976. doi:10.3844/jcssp.2014.1969.1976.
- [5] H. Watanabe, M. Bouazizi, T. Ohtsuki, Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection, IEEE access 6 (2018) 13825–13835.
- [6] R. Rajalakshmi, C. Aravindan, An effective and discriminative feature learning for url based web page classification, 2018. doi:10.1109/SMC.2018.00240.
- [7] R. Rajalakshmi, H. Tiwari, J. Patel, A. Kumar, R. Karthik., Design of kids-specific url classifier using recurrent convolutional neural network, Procedia Computer Science 167 (2020) 2124–2131. doi:<https://doi.org/10.1016/j.procs.2020.03.260>.
- [8] V. Swaminathan, S. Arora, R. Bansal, R. Rajalakshmi, Autonomous driving system with road sign recognition using convolutional neural networks, in: 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), 2019, pp. 1–4. doi:10.1109/ICCIDS.2019.8862152.
- [9] V. Ganganwar, R. Rajalakshmi, Implicit aspect extraction for sentiment analysis: A survey of recent approaches, Procedia Computer Science 165 (2019) 485–491. URL: <https://www.sciencedirect.com/science/article/pii/S1877050920300181>. doi:<https://doi.org/10.1016/j.procs.2020.01.010>.
- [10] S. Sivakumar, R. R., Analysis of sentiment on movie reviews using word embedding self-attentive lstm, 2021. doi:10.4018/IJACI.2021040103.
- [11] T. Mandl, S. Modha, A. Kumar M, B. R. Chakravarthi, Overview of the hasoc track at fire 2020: Hate speech and offensive language identification in tamil, malayalam, hindi, english and german, in: Forum for Information Retrieval Evaluation, FIRE 2020, Association for

- Computing Machinery, New York, NY, USA, 2020, p. 29–32. URL: <https://doi.org/10.1145/3441501.3441517>. doi:10.1145/3441501.3441517.
- [12] R. Rajalakshmi, B. Y. Reddy, Dlrq@hasoc 2019: An enhanced ensemble classifier for hate and offensive content identification, in: FIRE, 2019.
- [13] R. Rajalakshmi, R. Agrawal, Borrowing likeliness ranking based on relevance factor, in: Proceedings of the Fourth ACM IKDD Conferences on Data Sciences, CODS '17, Association for Computing Machinery, New York, NY, USA, 2017. URL: <https://doi.org/10.1145/3041823.3067694>. doi:10.1145/3041823.3067694.
- [14] Rahul, V. Gupta, V. Sehra, Y. R. Vardhan, Hindi-english code mixed hate speech detection using character level embeddings, in: 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 1112–1118. doi:10.1109/ICCMC51019.2021.9418261.
- [15] P. Mathur, R. Sawhney, M. Ayyar, R. Shah, Did you offend me? classification of offensive tweets in hinglish language, in: Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), 2018, pp. 138–148.
- [16] S. Kamble, A. Joshi, Hate speech detection from code-mixed hindi-english tweets using deep learning models, arXiv preprint arXiv:1811.05145 (2018).
- [17] V. Pathak, M. Joshi, P. Joshi, M. Mundada, T. Joshi, Kbcnmujal@ hasoc-dravidian-codemix-fire2020: Using machine learning for detection of hate speech and offensive code-mixed social media text, arXiv preprint arXiv:2102.09866 (2021).
- [18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, CoRR abs/1201.0490 (2012). URL: <http://arxiv.org/abs/1201.0490>. arXiv:1201.0490.
- [19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.