2022 International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN) | 978-1-6654-2111-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICSTSN53084.2022.9761335

Detecting Offensive Tamil Texts Using Machine Learning And Multilingual Transformer Models

Malliga Subramanian
Department of Computer Science and Engineering,
Kongu Engineering College,
Erode, Tamil Nadu, India

Adhithiya G J
Department of Computer Science and Engineering,
Kongu Engineering College,
Erode, Tamil Nadu, India

Gowthamkrishnan S
Department of Computer Science and Engineering,
Kongu Engineering College,
Erode, Tamil Nadu, India

Deepti R
Department of Computer Science and Engineering,
Kongu Engineering College,
Erode, Tamil Nadu, India

***Abstract*—Nowadays, social media has permitted an exponential increase in the circulation of hostile and toxic content, which has resulted in an increase in the number of people exposed to it. Many members of the Natural Language Processing community have recently expressed an interest in automated identification of such harmful content as hate speech, provocative language, and abusive language as a means of addressing this problem. Machine learning and multilingual transformer models are used in this study to automatically identify Tamil language comments as either offensive or not offensive messages. The dataset is collected from YouTube and Kaggle. BERT tops the competition when it comes to offensive language identification models, with an accuracy of 82% compared to the others.**

Keywords—Hate Speech, Offensive, Toxicity, Social Media, Youtube, Tamil, Machine Learning, BERT, HASOC

I. INTRODUCTION

They are increasingly using social media platforms to exchange information and communicate with each other in their daily lives. Social media and microblogging platforms' widespread adoption increases people's ability to connect, but it can also have a negative impact on their daily life. There has been an increase in hateful and insulting remarks on social media because of the anonymity capabilities of these platforms, which allow toxic people to evade editorial supervision [1–2]. Toxic conduct can have a negative impact on the community if it is not addressed in a timely manner. The dread of being abused or harassed can prevent someone from expressing themselves freely in a hostile atmosphere. As a result, offensive language, hate speech, and other unpleasant content on the internet represent a danger to our society.

It is widely accepted that the propagation of abusive language on social media has a negative impact on the general population. Because of its size and lack of regulation, however, this is a challenge to solve. Hate speech on social media is illegal in many countries, but only if it does not target a specific group or inspire criminal acts. YouTube, Facebook, and Twitter all have policies and methods to control hate speech content and other harmful behaviour since it has a significant impact on society [5]. Automated approaches for identifying objectionable language and removing/hiding it from other users are prevalent [6].

Generally speaking, systems for the automatic identification of hatred and offensive words in Natural Language Processing (NLP) fall into one of three categories: In addition to traditional linear classifiers [7, 8], neural network architectures such as CNN and RNN [9-11] have been developed, as have fine-tuned pre-trained language model such as BERT (Bidirectional Encoder Representation from Transformer among others [12, 13]. Aiming to build on this previous work, we provide machine learning and multilingual transformer models for automatically classifying YouTube user comments as offensive or non-offensive texts. We specifically concentrate on Tamil, a classical language that originated in India and for which there is a paucity of data and models in prior research.

The remaining sections of the paper are organized as follows: Using a survey of existing research on the topic, Section 2 summarizes the background work on hate speech and offensive language identification that has been done so far. Our proposed models are explained in Section 4, which is followed by findings and discussion. Section 3 contains a task description as well as an overview of the dataset that was used in the study. Finally, in Section 5, we come to a conclusion to the study by summarizing the findings and implications of the research.

II. LITERATURE SURVEY

The sheer volume and expanding popularity of social media make manual detection and removal of undesirable information by moderators impossible. There is a great demand for tools and procedures that automatically detect and prevent the spread of offensive content on the internet. The development of automated algorithms to filter offensive language on social media platforms has accelerated in recent years, despite documented difficulties in categorizing nuances due to the subjective and context-dependent nature of texts [6, 14]. Additional difficulties arise when working with texts that comprise many languages – henceforth referred to as code-mixed data – as a result of multilingual consumers. Numerous prior systems omit data in languages

other than English [14, 15], resulting in a dearth of study in this area for languages with limited resources.

In light of recent research on Dravidian languages, particularly Dravidian code-mixed text, [16] and [17] provided a common goal for sentiment analysis of YouTube comments in Dravidian code-mixed text. Due to the fact that user-generated content on social media is frequently code-mixed and little studied for under-resourced languages, recent work has focused on developing methods for detecting inappropriate text in such languages [18-23]. The authors of [18] presented a multi-task learning dataset for sentiment analysis and offensive language identification in the under-resourced Kannada language in order to stimulate further research into multi-task learning and to improve performance for under-resourced languages. Bharathi Raja et al. [19] introduced a code-mixed Malayalam-English dataset of YouTube comments annotated for sentiment analysis that enabled code-mixed sentiment analysis and provided free data for code-mixed research. The authors of [20] constructed a gold standard Tamil-English code-mixed, sentiment-annotated corpus of 15,744 comment posts using YouTube comments, including details of corpus annotations for polarity and the results of inter-annotator agreement as a benchmark. Additionally, [21] published the outcomes of the first collaborative job on identifying offensive languages in Tamil, Malayalam, and Kannada, which relied on a thoroughly annotated data set based on human assessments. The job setup enabled us to evaluate the model in multilingual environments and to investigate the phenomena of code mixing.

Multimodal classification of "Troll Meme Classification in Tamil" was attempted in a novel way in [22] by offering an updated TamilMeme dataset that now included Tamil text from memes. This research, although providing a multimodal classification challenge, also explored difficulties in natural language processing of a low-resourced language and code-mixed dataset. Work by [23] describes the shared task of machine translation for the Dravidian (Tamil) languages presented at the first workshop on Speech and Language Technologies for Dravidian Technologies, where the best performing systems had a high Bilingual Evaluation Understudy Score despite the lack of training data. There is still a lot of effort to be done in detecting abusive language in under-resourced Tamil, Malayalam and Kannada languages to add to this burgeoning area.

Next, we'll focus on approaches that use machine learning and natural language processing to automatically identify objectionable language on web sites. For example, n-grams (word- and character level) are commonly used in traditional machine learning techniques. With 5-fold cross validation, Davidson et al.[24] categorised tweets as hate speech, offensive, and neither hate nor offensive using a multi-class classifier that used Naive Bayes, Decision Trees, Random Forests, etc. with precision 0.91, recall 0.90, and F1 score of 0.90. Gibert et al. [25] generated a manually annotated hate speech dataset from Stormfront, a white supremacist internet community, which contains both hateful and non-hateful utterances. It has been used to annotate test data based on hand-annotated training data using Support Vector Machines (SVM), Convolutional Neural Networks (CNN) and Recurrent Neural Networks.

Linear classifiers proved to be competitive, if not superior, when compared to neural networks when results were inconsistent across datasets and designs. On the other hand, systems that use pre-trained language models have shown to be the most effective in this field, hitting new levels of performance. However, because of the wide range of languages used in their training, these pre-trained models can only be used for general-purpose language understanding tasks. The authors have supplied benchmark datasets for testing machine learning models for hate speech classification [26]. There is also discussion of the pros and downsides of single and hybrid machine learning algorithms in the classification of hate speech (see [26]).

Despite the fact that a variety of feature extraction methods have been employed in machine learning-based approaches, a clear feature extraction strategy is always required. CNNs, Bi-directional Long Short-Term Memory Networks (LSTM), and BERT [29] are currently being used to improve the performance of neural network models that detect hate speech and objectionable content. Pre-trained BERT and multilingual BERT were used in a study in [29] to detect hate speech and objectionable content in English, German and Hindi. A linear model incorporating features from word unigrams, word2vec, and Hatebase, as well as word-based LSTM and a fine-tuned BERT were tested by the authors of [12]. As a part of this project, Twitter API searches are used to gather the Offensive Language Identification Dataset (OLID). Transfer learning based on pre-trained language bidirectional encoder representation models was developed in another study [30]. On the basis of transfer learning, this study examined BERT's ability to capture unpleasant circumstances in social media content using novel fine-tuning methodologies and two publicly available datasets annotated for racism, sexism and hatred on Twitter to evaluate the suggested approach.

While multilingual transformer models were shown to be successful for the shared objective in [21], a few machine learning models also performed well employing hybrid approaches for feature selection/extraction from texts rather than single feature selection/extraction algorithms. This paper's research relies on machine learning models with feature selection approaches rather than a transformer since they keep the model basic for simpler comprehension. BERT is one of several machine learning algorithms used to classify YouTube comments as hostile and non-offensive. There is no free lunch theorem [31] and current works with disparity in datasets, thus we emphasize that a solution which works well for one dataset may not be suitable for another. There may not be a single classifier that performs best on all datasets, thus we apply multiple classifiers to a feature vector in order to see which one produces the best results. In the following part, we go into greater depth about our methodology.

III. MATERIALS AND METHODS

A. DATASET DESCRIPTION

To define offensive language, which is commonly referred to as "flames," the texts in this study use the common definition, which refers to "offensive messages or remarks that in some circumstances are inappropriate, demonstrate a lack of respect towards certain groups of people, or are just rude in general. The comments are written

in code-mixed style (Mixture of Native and Roman Script), and they are in two different languages: Tamil and English.

Throughout the dataset, there are comments in Tamil, English, and Malayalam, and none of them are entirely in Tamil or English or Malayalam, but are instead a mixture of Tamil and English or Malayalam. Although the comments in the dataset contain more than one sentence, the average sentence length in the corpora is one sentence. In the dataset, each comment was marked as either Offensive (OFF), Not-Offensive (NOT), or Non-Tamil, depending on its content. To train the model, the dataset had 351339 Tamil and English texts, while the dataset contained 4392 Tamil and English texts for testing, with the texts being divided into three categories: Offensive and Not-Offensive. The training set has 281,345 texts classified as Offensive and 69994 texts classified as Non-Offensive. Because the dataset was unbalanced, we supplemented the available data by rearranging the texts in order to generate new ones. To do this, the words in each phrase were shuffled in order to construct a new one [32]. The use of Synthetic Minority Oversampling Technique (SMOTE) [33], which is an oversampling strategy that creates synthetic samples from the minority class, also helped to reduce imbalance. SMOTE is used to build a synthetically or virtually class-balanced training set, which is then used to train the classifier. SMOTE works by selecting instances in the feature space that are close together, which is why it is sometimes referred to as "neighbourhood selection. Table I contains examples of offensive and non-offensive texts drawn from the dataset, respectively.

TABLE I. SAMPLE TEXTS FROM THE DATASET

Documents	Texts	Label
Doc[10]	இதுசாதிபடம்அல்லபெண்களின்படம் ஒருசிலபேர்பண்ணும்தவறால்சாதிக்கு கெட்டபேருஅதான்படம்படம் கண்டிப்பாகவெற்றிபெறவேண்டும் செம்ம...	NOT
Doc[11]	இந்தபடம்மாபெரும்வெற்றிபெற்றுதமிழ் சினிமாவில்ஆதிக்கம்செலுத்தும்இது போன்றமேலும்பலபடங்களைஇயக்க வாழ்த்துக்கள்படம்கண்டிப்பாகவெற்றி பெறவேண்டும்செம்ம...	NOT
Doc[13]	தமிழகமக்கள்சார்பாகவாழ்த்துக்கள் சாதிகள்இல்லையடிபாப்பாசலுகைகள் நிறையவேண்டுமடிஓட்டுக்காகபாப்பா	NOT
Doc[35]	இதுவரையிலும்ஜெயம்ரவியாகஇருந்த நான்இனிமேல்கோமாளிரவியாக அழைக்கப்படுகிறேறன்இவன் கோமாளிரவி	OFF
Doc[62]	இப்படியேஇன்னும்வருடம்சாதி கொண்டுவாங்க...அப்புறம்என்னடா நடக்கும்உலகம்அழிச்சுடும்	OFF

B. PRE-PROCESSING AND FEATURE EXTRACTION

The initial step in building any classifier is preprocessing the corpus (or data cleaning). Emojis, punctuation letters, and other non-Tamil text were preprocessed out of the corpus. The Emoji for Python module was used to convert the emojis into text, and CountVectorizer was used to transform the text corpus to a vector of term/token count. In the scikit-learn toolkit, the CountVectorizer module takes care of managing n-gram size, custom preprocessing and tokenization as well as stop words and vocabulary size. Each remark was tokenized using Count Vectorizer and TF-IDF Vectorizer. Text data can be used to extract and represent features. Custom preprocessing, tokenization, stop words and vocabulary size may all be controlled using this tool. The Count Vectorizer in Python's scikit-learn module is an excellent utility. Term Frequency Inverse Document Frequency is referred to as TF-IDF. In order to match a machine algorithm for prediction, this is a standard approach for transforming text into meaningful numbers. Based on the frequency of each word in the text, it creates a vector representation of the text. It's helpful when working with a large number of texts and creating a vector for each word. The feature map is generated for both vectorizers. The common data extracted from the two feature maps is then fed into the multiple models for training purposes. It's important to note that CountVectorizer does not record these words as strings in its database. As an alternative, an index number is assigned to each of them. " சாதி " has index zero; " படம் " has one and so on. Classifiers are trained using the fit() and transform() methods, where each vector's value is calculated for each comment in the training set.

TABLE II. RESULTS FROM COUNT VECTORIZER

Text	சாதி	படம்	பெண்களின்	தவறால்
Doc[10]	2	1	4	1
Doc[11]	0	0	3	0
Doc[13]	1	0	0	0

C. PROPOSED CLASSIFIERS

Following a description of the text-to-feature transformation, we give the classification methods that were employed for offensive text identification in our research. We offer five classifiers based on machine learning, namely, the Naive Bayes Multinomial Classifier, the SVM, the Logistic Regression, and the KNN, as well as a transformer model, called the BERT (Bayesian Evolutionary Regression Transformer). The output of the count vectorizer is fed as input to these classifiers, and the output is the classification that has been identified for each text in the count vectorizer. The proposed models have been implemented using the scikit-learn Python libraries, which are available here.

We will now go into the reasoning behind the selection of the models mentioned above. An example of a probabilistic model is the Naive Bayes Multinomial Classifier, which is an advanced variation on the Naive Bayes method. Simple Naive Bayes models a document for the presence or absence of specific words, whereas Multinomial classifier explicitly models the word counts and works well on small amounts of training data and trains relatively quickly when compared to other models [34]. Simple Naive Bayes models a document for the presence or absence of specific words. supervised classification techniques that learn from training data to generate an ideal hyperplane that separates the categories while categorising fresh data. Support-vector machines (SVM) are one type of supervised classification algorithm. Because SVMs are meant to find a hyperplane that maximises the marginal distance between classes, they are capable of dealing with enormous numbers of samples. Binary classification is achieved by generating a hyperplane out of the support vectors that splits the cases into two non-overlapping

classes. Text classification tasks are particularly well suited for SVM classifiers [34]. In our studies, we employed the SVC() function from the scikit-learn library, with the Radial Basis Function (RBF) serving as the kernel function.

When analysing the relationship between one independent variable and one or more independent variables, another model known as Logistic regression makes use of a sigmoid function to describe the association. We used L2 regularisation for this model in order to deal with data that had a binary dependent variable, such as offensive or not offensive, in order to cope with the data. Through the use of a linear combination of the independent and prediction variables, it calculates the likelihood of an output. KNN is a straightforward text classification method that categorises new data by comparing it to all previously collected data using some measure of similarity to the new data. However, despite the fact that KNN is a straightforward technique that can be applied to both classification and regression tasks without making any assumptions about the data, it is memory and time heavy due to the fact that all training data must be saved. All of the classifiers provide different values for performance measures since they function in a variety of ways depending on the characteristics outlined above.

Bidirectional Encoder Representations from Transformers (BERT) is the acronym. It is aimed to pre-train deep bidirectional representations from unlabeled text by conditioning on both the left and right context. When applied to NLP, the BERT model can be refined with just one additional output layer, resulting in models that are among the most advanced in the industry. BERT is built on Transformers, a deep learning model in which each output element is connected to each input element, and the weightings between them are dynamically computed based on their connection. BERT is unique in that it can read in both directions at the same time, unlike any other device. The term "bidirectionality" refers to this new ability made possible by the invention of Transformers.

D. MODEL IMPLEMENTATION

Python programming language and its Sci-kit learn library have been used to implement the classification models. Google Colaboratory (Colab) was used to train and test the proposed models. Colab is a fully cloud-based Jupyter notebook environment that doesn't require any desktop setup, hence it can run on any browser. The code-base is released open-source on Github.

IV. RESULTS AND DISCUSSION

In this section, we discuss the results of the classification models built for automatic detection of Offensive and Not-Offensive Tamil texts collected from social media. We train each classifier using the features extracted from the training set and test the models using the test dataset provided.

A. PERFORMANCE METRICS

The performance of the different models used for the classification problem have been evaluated using the following metrics: Accuracy, Precision, Recall and F1-Score. These metrics commonly used for the evaluation of classifiers are defined as follows.

Accuracy is defined as the number of texts correctly classified as belonging to a specific class divided by the total number of texts in that class and is calculated by Equation (1).

$$\text{Accuracy} = (TP+TN) / (TP+TN+FP+FN) \quad (1)$$

Where, TP = True positive (the number of correctly classified texts for each class), TN = True Negative (the number of texts correctly classified in other class except the correct class), FP = False Positive (number of texts misclassified in other class except the right class) and FN = False Negative (the number of texts misclassified in the relevant class) in the confusion matrix.

The number of texts correctly categorized as a certain class out of the total number of actual texts in that class is defined as Recall (also known as Sensitivity or True Positive Rate) and is computed using Equation (2).

$$\text{Recall} = TP / (TP+FN) \quad (2)$$

Precision (Positive Predictive Value) is defined as the number of texts accurately categorized as a specific class out of the total number of texts categorized as that class, and is given by Equation (3).

$$\text{Precision} = TP / (TP+FP) \quad (3)$$

F1-Score is defined as the harmonic average of the Precision and Recall, that is, the weighted average of Precision and Recall. It is calculated as in Equation (4).

$$\text{F1-Score} = (2*\text{Precision}*\text{Recall}) / (\text{Precision}+\text{Recall}) \quad (4)$$

Table III shows the values of the performance metrics acquired for the test dataset for each of the classifiers, as calculated for each of the classifiers. We provide the created confusion matrix, which is a model matrix that compares the actual target values with those predicted by a machine learning model, and which assists us in doing error analysis of the proposed models. Fig 1 depicts the confusion matrix for the models that have been proposed. The correct classifications are represented by the diagonal elements of the confusion matrix. The X axis represents the expected classes, and the Y axis represents the actual classes. Consider the following: the naive bayes model properly identified 50 offensive texts as such, while mistakenly categorising an additional 83 of these texts as Not-Offensive (see Fig 1).

The results of classifiers are presented in Table III in terms of the performance measures used. When compared to other classifiers, BERT has the highest level of accuracy to offer. This high accuracy from BERT is due to the significant improvement in the model achieved by training it longer with larger batches over more data, removing the next sentence prediction objective, training on longer sequences dynamically changing the masking pattern applied to the training and training on longer sequences. However, this model is much more complex and requires a large amount of time to run. But, recognizes that BERT has

a bigger vocabulary, which allows the model to represent any word resulting in more parameters, and the increase in complexity is justified by gain in performance.

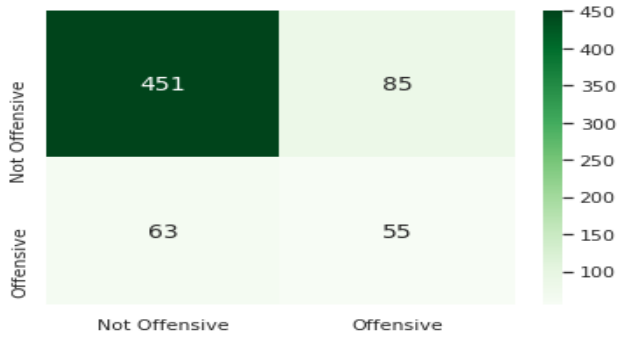

Fig 1. Confusion Matrix of NB

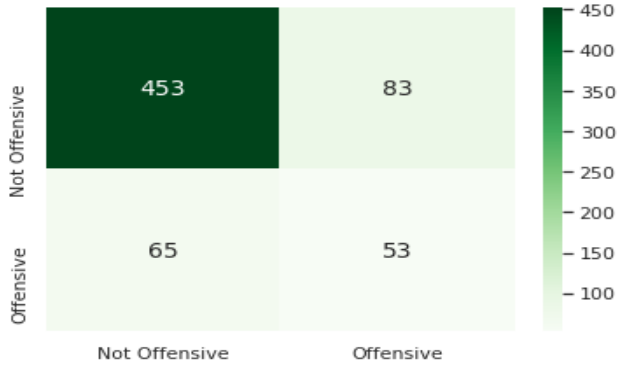

Fig 2. Confusion Matrix of LR

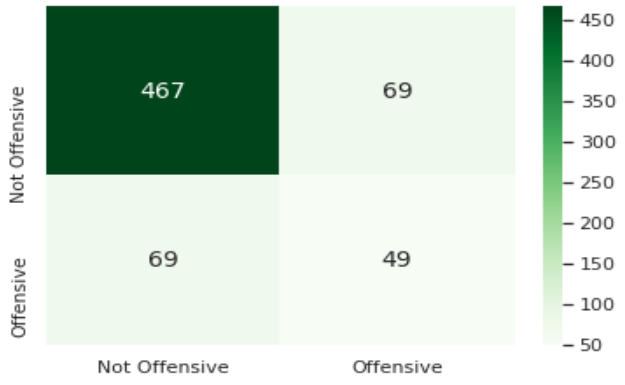

Fig 3. Confusion Matrix of SVM

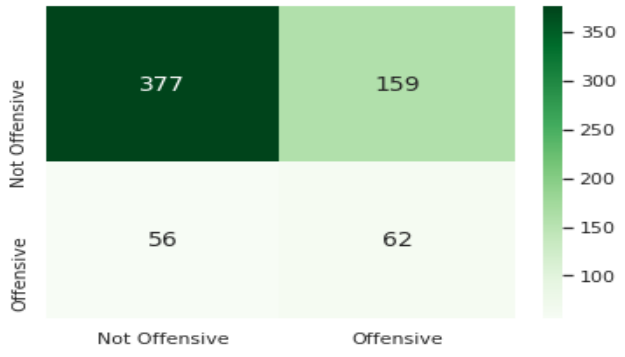

Fig 4. Confusion Matrix of KNN

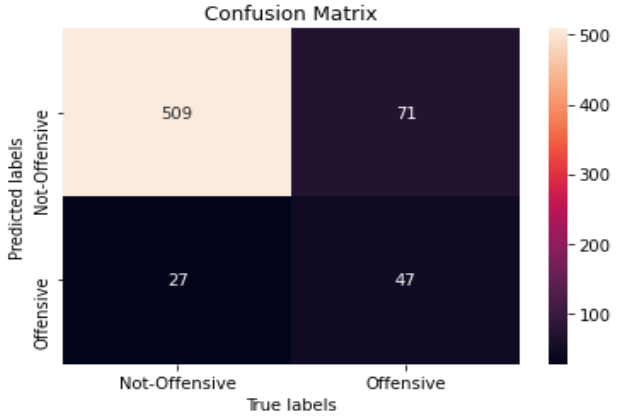

Fig 5. Confusion Matrix of BERT

For offensive class, the Recall and F1-score had low values compared to Not-Offensive classes. We understand that this is due to the smaller number of texts under this class. We have augmented texts for this class using SMOTE techniques, however, the low values for Not-Offensive class persisted.

TABLE III. PERFORMANCE OF CLASSIFIERS

Classifiers	Class Label	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
NB Multinomial	NOT	77.37	84.142	87.74	85.91
	OFF		46.610	39.28	42.64
SVM	NOT	78.89	87.127	87.12	87.13
	OFF		41.525	41.52	41.53
KNN	NOT	67.12	70.336	87.06	77.81
	OFF		52.542	28.05	36.57
LR	NOT	77.37	84.515	87.45	85.95
	OFF		44.915	38.97	41.73
BERT	NOT	82 .71	85.312	91.15	79.85
	OFF		57.430	33.95	44.35

In this paper used count vectors to extract features from the texts in this work - one issue with simple counts is that frequently used words, such as "an" and "the," make their high counts meaningless in the encoded vectors. An alternate option is to use TF-IDF to calculate word frequencies, which might be superior to Count Vectorizers because it not only considers the frequency of words in the corpus but also their importance. Future work can remove the words that are less important for analysis, reducing the input dimensions and making the model building less complex.

V. CONCLUSION

On the topic of offensive language in social media, this study presented experimental work and the results of the task to detect offensive content in a code-mixed dataset of Dravidian languages in order to address the issue of offensive language in social media. A comprehensive review of previous strategies in offensive text categorization was offered, as was a look at new models for automatically detecting offensive texts in Tamil. For this assignment, a count vector was used to extract features from the dataset,

and various classifiers were created, including the Nave Bayes Multinomial model, SVM, KNN, Logistic Regression, BERT to perform the classification. BERT was the most accurate of these models when compared to the other models. It is feasible that our approach will be further developed by incorporating various possible numerical/vectorial representations of texts as well as new neural network-based classifiers with advanced linguistic properties. The study contributes to studies in offensive text identification for under-resourced languages such as Tamil, and we hope that it will serve as a model for future work in this field.

REFERENCES

[1] J. Blair, "New breed of bullies torment their peers on the Internet," Education Week, vol. 22, p. 6, 2003.

[2] S.-H. Lee and H.-W. Kim, "Why people post benevolent and malicious comments online," Communications of the ACM, vol. 58, pp. 74-79, 2015.

[3] A. M. Obadimu, "Assessing the Role of Social Media Platforms in the Propagation of Toxicity," University of Arkansas at Little Rock, 2020.

[4] T. De Smedt, S. Jaki, E. Kotzé, L. Saoud, M. Gwóźdź, G. De Pauw, et al., "Multilingual cross-domain perspectives on online hate speech," arXiv preprint arXiv:1809.03944, 2018.

[5] N. Alkiviadou, "Hate speech on social media networks: towards a regulatory framework?," Information & Communications Technology Law, vol. 28, pp. 19-35, 2019/01/02 2019.

[6] B. Vandersmissen, "Automated detection of offensive language behavior on social networking sites," IEEE Transaction, 2012.

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

[8] M. H. Ribeiro, P. H. Calais, Y. A. Santos, V. A. Almeida, and W. Meira Jr, "Characterizing and detecting hateful users on twitter," in Twelfth international AAAI conference on web and social media, 2018.

[9] R. Kshirsagar, T. Cukuvac, K. McKeown, and S. McGregor, "Predictive embeddings for hate speech detection on twitter," arXiv preprint arXiv:1809.10644, 2018.

[10] P. Mishra, H. Yannakoudakis, and E. Shutova, "Neural character-based composition models for abuse detection," arXiv preprint arXiv: 1809.00378, 2018.

[11] J. Mitrović, B. Birkeneder, and M. Granitzer, "nlpUP at SemEval-2019 task 6: A deep neural language model for offensive language detection," in Proceedings of the 13th International Workshop on Semantic Evaluation, 2019, pp. 722-726.

[12] P. Liu, W. Li, and L. Zou, "NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers," in Proceedings of the 13th international workshop on semantic evaluation, 2019, pp. 87-91.

[13] S. D. Swamy, A. Jamatia, and B. Gambäck, "Studying generalisability across abusive language detection datasets," in Proceedings of the 23rd conference on computational natural language learning (CoNLL), 2019, pp. 940-950.

[14] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in Proceedings of the fifth international workshop on natural language processing for social media, 2017, pp. 1-10.

[15] C. Cieri, M. Maxwell, S. Strassel, and J. Tracey, "Selection criteria for low resource language programs," in Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), 2016, pp. 4543-4549.

[16] B. R. Chakravarthi, R. Priyadharshini, R. Ponnusamy, P. K. Kumaresan, K. Sampath, D. Thenmozhi, et al., "Dataset for Identification of Homophobia and Transophobia in Multilingual YouTube Comments," arXiv preprint arXiv:2109.00227, 2021.

[17] B. R. a. P. Chakravarthi, Ruba and Thavareesan, Sajeetha and Chinnappa, Dhivya and Durairaj, Thenmozhi and Sherly, Elizabeth and McCrae, John P. and Hande, Adeep and Ponnusamy, Rahul and Banerjee, Shubhanker and Vasantharajan, Charangan, "Findings of the Sentiment Analysis of Dravidian Languages in Code-Mixed Text 2021," in FIRE 2021, 2019.

[18] A. Hande, R. Priyadharshini, and B. R. Chakravarthi, "KanCMD: Kannada Code-Mixed dataset for sentiment analysis and offensive language detection," in Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media, 2020, pp. 54-63.

[19] B. R. Chakravarthi, N. Jose, S. Suryawanshi, E. Sherly, and J. P. McCrae, "A sentiment analysis dataset for code-mixed Malayalam-English," arXiv preprint arXiv:2006.00210, 2020.

[20] B. R. Chakravarthi, V. Muralidaran, R. Priyadharshini, and J. P. McCrae, "Corpus creation for sentiment analysis in code-mixed Tamil-English text," arXiv preprint arXiv: 2006.00206, 2020.

[21] B. R. Chakravarthi, R. Priyadharshini, N. Jose, T. Mandl, P. K. Kumaresan, R. Ponnusamy, et al., "Findings of the shared task on offensive language identification in Tamil, Malayalam, and Kannada," in Proceedings of the Speech and Language Technologies for Dravidian Languages, 2021, pp. 133-145.

[22] S. Suryawanshi and B. R. Chakravarthi, "Findings of the shared task on Troll Meme Classification in Tamil," in Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, 2021, pp. 126-132.

[23] B. R. Chakravarthi, R. Priyadharshini, S. Banerjee, R. Saldanha, J. P. McCrae, P. Krishnamurthy, et al., "Findings of the Shared Task on Machine Translation in Dravidian languages," in Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, 2021, pp. 119-125.

[24] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in Proceedings of the International AAAI Conference on Web and Social Media, 2017.

[25] 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.

[26] F. E. Ayo, O. Folorunso, F. T. Ibharalu, and I. A. Osinuga, "Machine learning techniques for hate speech classification of twitter data: State-of-the-art, future challenges and research directions," Computer Science Review, vol. 38, p. 100311, 2020.

[27] S. MacAvaney, H.-R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, "Hate speech detection: Challenges and solutions," PloS one, vol. 14, p. e0221152, 2019.

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

[29] S. Dowlagar and R. Mamidi, "HASOCOne@ FIRE-HASOC2020: Using BERT and Multilingual BERT models for Hate Speech Detection," arXiv preprint arXiv: 2101.09007, 2021.

[30] M. Mozafari, R. Farahbakhsh, and N. Crespi, "A BERT-based transfer learning approach for hate speech detection in online social media," in International Conference on Complex Networks and Their Applications, 2019, pp. 928-940.

[31] Y.-C. Ho and D. L. Pepyne, "Simple explanation of the no-free-lunch theorem and its implications," Journal of optimization theory and applications, vol. 115, pp. 549-570, 2002.

[32] (2021). Data Augmentation in NLP. Available: https://neptune.ai/blog/data-augmentation-nlp

[33] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, vol. 16, pp. 321-357, 2002.

[34] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," Emerging artificial intelligence applications in computer engineering, vol. 160, pp. 3-24, 2007.

[35] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, et al., "Roberta: A robustly optimized bert pretraining approach," arXiv preprint arXiv:1907.11692, 2019.