Computer Speech and Language 78 (2023) 101464

Contents lists available at ScienceDirect

Computer Speech & Language

journal homepage: www.elsevier.com/locate/csl

HOTTEST: Hate and Offensive content identification in Tamil using Transformers and Enhanced STemming

Ratnavel Rajalakshmi [a,*], Srivarshan Selvaraj [a], Faerie Mattins R. [a], Pavitra Vasudevan [a], Anand Kumar M. [b]

[a] School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, Tamil Nadu, India
[b] Department of Information Technology, National Institute of Technology, Surathkal, Karnataka, India

ARTICLE INFO

Keywords:
Offensive content identification in Tamil
Hate Speech in Tamil YouTube comments
Stop word removal and enhanced stemming for Tamil text
Tamil text data pre-processing
Deep learning
MuRIL
Transformers for Tamil

ABSTRACT

Offensive content or hate speech is defined as any form of communication that aims to annoy, harass, disturb, or anger an individual or community based on factors such as faith, ethnicity, appearance, or sexuality. Nowadays, offensive content posted in regional languages increased due to the popularity of social networks and other apps usage by common people. This work proposes a method to detect and identify hate speech or offensive content in Tamil. We have used the HASOC 2021 data set that contains YouTube comments in Tamil language and written in Tamil script. In this research work, an attempt is made to find suitable embedding techniques for Tamil text representation by applying TF-IDF and pre-trained transformer models like BERT, XLM-RoBERTa, IndicBERT, mBERT, TaMillion, and MuRIL. As Tamil is a morphologically rich language, a detailed analysis is made to study the performance of hate speech detection in Tamil by applying enhanced stemming algorithms. An extensive experimental study was performed with different classifiers such as logistic regression, SVM, stochastic Gradient Descent, decision tree, and ensemble learning models in combination with the above techniques. The results of this detailed experimental study show that stop word removal produces mixed results and does not guarantee improvement in the performance of the classifier to detect offensive content for Tamil data. However, the performance on stemmed data shows a significant improvement over un-stemmed data in Tamil texts. As the data is highly imbalanced, we also combined an oversampling/downsampling technique to analyze its role in designing the best offensive classifier for Tamil text. The highest performance was achieved by a combination of stemming the text data, embedding it with the multi-lingual model MuRIL and using a majority voting ensemble as the downstream classifier. We have achieved the F_1-score of 84% and accuracy of 86% for detecting offensive content in Tamil YouTube comments.

1. Introduction

The term offensive refers to behavior that irritates, resents, or upsets a person or a group of people. Offensive content can be spread through social media channels such as tweets, comments, and posts. According to Aleksandra (2016), YouTube has the largest number of offensive comments. The anonymity of the user is a crucial factor in the prevalence of disgusting comments on the internet. Since the user's identity is unknown, they share their personal views without any restrictions. Another reason for this is that, when people comment on something, they believe they are conversing with void rather than another human. The offensive

* Corresponding author.
E-mail address: rajalakshmi.r@vit.ac.in (R. Rajalakshmi).

https://doi.org/10.1016/j.csl.2022.101464
Received 10 February 2022; Received in revised form 8 October 2022; Accepted 9 October 2022
Available online 22 October 2022

Word	Transliteration	Translation
படி	padi	step, study
பொருள்	porul	meaning, object
இடம்	idam	spot, rank, place
ஆவி	aavi	vapour, spirit of the dead

Fig. 1. Polysemy words in Tamil.

comments will impact the concerned person/group and disturb their mental health (Guntuku et al., 2019). In their daily life, the victim suffers from depression, anxiety, insecurity, and negativity (Sharen and Rajalakshmi, 2022). This is a significant problem that requires immediate attention. However, locating and eliminating the objectionable comments can be time-consuming, if it is done manually or applying simple methods. Machine learning and deep learning techniques play a critical role in achieving this goal with automated systems to provide more better solutions. By automatically identifying and removing offensive texts, social media platforms can become safe for users where none of them get offended by any kind of comments. To classify offensive and non-offensive comments on YouTube, a variety of machine learning techniques are used for English, Spanish Basile et al. (2019), German Risch et al. (2021), Reddy and Rajalakshmi (2020), and other languages too, but limited works Kumaresan et al. (2021) are reported in the literature for low resource languages like Tamil.

Dravidian languages include languages like Tamil, Malayalam, Telugu, and Kannada. Despite being the official language of the Indian state of Tamil Nadu and the union territory of Puducherry, Tamil is also widely spoken in Malaysia, Mauritius, Fiji, and South Africa. It is also one of the official languages of Singapore and Sri Lanka. Many offensive comments can also be found in these languages on social media and there is a high demand for automated systems for categorizing the offensive and non-offensive remarks on YouTube comments in regional languages. The best strategy to identify the offensive/non-offensive content has been studied in this work by focusing only on Tamil YouTube comments. Few researchers Kedia and Nandy (2021), Garain et al. (2021a), Jayanthi and Gupta (2021) have tried to study the problem of identifying offensive content in code-mixed Tamil language, in which the tweets/comments are written in English/Roman script.

However, very few works are reported for identifying the offensive content in Tamil YouTube comments that are written in native language/Tamil script (Rajalakshmi et al., 2022b). The challenges of this low-resourced language include the limited corpus, non-availability of standard/benchmarked stop words, and proper stemming algorithms exclusively for Tamil scripts. Tamil language, one of the oldest surviving languages, is an official language in Tamil Nadu and Pondicherry of India and also other countries such as Singapore, Malaysia and Srilanka. Tamil language is rich in morphology, linguistics and has a complex grammar with many polysemy words as seen in Mohanty and Arulmozi (2010). Fig. 1 shows some examples of polysemy words in Tamil. The Tamil language is spoken by many people, and a colloquial language, they use different words that do not follow the grammar of the language and combine some other non-Tamil terms also. So, it is highly challenging to get the formal representation of these kinds of terms, which is purely region-specific. With Recent technical advancements and high use of social media by all people belonging to different socioeconomic statuses and education levels, many offensive tweets are posted that may target people of a particular caste/gender/religion. The task of determining whether the content is offensive or not-offensive is highly challenging due to its complex nature.

In this paper, we try to address the above problems by applying NLP techniques in combination with machine learning and deep learning models. For this study, we have considered the data set that contains Tamil comments from YouTube, which was released as part of the shared task HASOC 2021 (Chakravarthi et al., 2021a). It is an annotated corpus, in which every comment is labeled as offensive or not-offensive. We investigate an effective method to determine offensive comments, by designing different classifiers and applying various embedding techniques for better representation of Tamil native script. We have studied in detail by applying traditional TF-IDF-based representation, Tamil-specific language model TaMillion, multi-lingual language models such mBERT, IndicBERT (specific to Indian Languages), MuRIL in addition to other widely used popular transformer models viz., BERT, XLM-RoBERTa to find the best representation for Tamil text. For developing any predictive models for NLP tasks, data cleaning and pre-processing steps play a crucial role, as they may directly/indirectly impact the performance. In this research work, we have analyzed the effects of stop-word removal and existing Stemming algorithms in identifying the offensive contents in Tamil text with various classification algorithms including logistic regression, Support Vector Machine, Naive Bayes classifier, Stochastic Gradient Descent, decision tree, random forest, and majority voting ensemble classifier. We have applied a suitable stemming algorithm that can address the problems in a morphologically rich language like Tamil. The data set is highly imbalanced with more non-offensive (3774) tweets and very less offensive tweets (926). So, we have also applied an oversampling/under-sampling technique and studied the effect of sampling techniques in designing the best ensemble-based classifier that could identify offensive tweets with better performance. The highest performance was achieved by a combination of stemming the text data, embedding it with the multi-lingual language model, MuRIL, and using a majority voting ensemble for the classification purpose. We have achieved the F_1 score of 84% and accuracy of 86% for detecting offensive content in Tamil YouTube comments. The contributions of our research work are highlighted below:

- An effective and suitable text embedding method is identified for representing Tamil language YouTube comments, which are written in Tamil script
- The impact of Tamil stop words in feature representation of Tamil text is analyzed with an extensive experimental study
- The effectiveness of applying Stemming for Tamil text representation is studied in depth
- A detailed comparative analysis is carried out by combining the best pre-processing techniques with the best feature representation method and applied machine learning models to determine the best ensemble-based classifier for identifying offensive contents in the Tamil language

The paper is organized as follows. The overview of the existing literature is briefed in Section 2. The data set used for this research work is described in Section 3. The proposed methodology is presented in Section 4. Extensive experiments and detailed result analysis are elaborated in Section 5. The comparative study of the proposed system with the existing works is outlined in Section 6 followed by the concluding remarks in Section 7.

2. Related works

Offensive content detection in social media comments is not new for the English language (Mandl et al., 2021) and some systems are also developed for other languages like Hindi (Rajalakshmi et al., 2022), German. (Rajalakshmi and Reddy, 2019), etc. But only limited work is carried out for identifying the offensive content in low-resourced Dravidian languages like Tamil, Malayalam, and Kannada. Garain et al. (2021b) proposed a method for detecting offensive language in Kannada-English, Malayalam-English, and Tamil-English code-mixed language pairs extracted from social media. They proposed a multi-label classification approach based on the ensemble of IndicBERT and generic BERT models, intending to identify and minimize offensive content in social media. It was a multi-label classification problem with many sub-categories. The system tokenizes the data before feeding it into the model for training. The class-wise confidence score was calculated and then combined to form an output vector that aids in classification. In the Malayalam–English, Tamil-English, and Kannada-English datasets, they have obtained F_1-scores of 0.54, 0.72, and 0.66, respectively. The task described in the study by Kedia and Nandy (2021) is similar to the previous study, as it aims to classify code-mixed offensive language material acquired from social media. Their data set comprised six labels, and multi-label classifiers such as multinomial NB, linear SVM, and random forests were used to perform the classification. The use of a trained from scratch vanilla LSTM model and ULMFiT transfer learning framework resulted in considerable improvements in outcomes. However, the suggested system is an ensemble of an AWD-LSTM-based model and two distinct transformer model architectures based on BERT and RoBERTa. In the Malayalam–English, Tamil-English, and Kannada–English datasets, weighted-average F_1-scores of 0.97, 0.77, and 0.72, respectively, were obtained.

Models based on transformer architecture were employed by Jayanthi and Gupta (2021) including a cased version of the multilingual BERT and XLM-RoBERTa. Following tokenization, BERT is used to represent sentences at the sentence level. By averaging the representations of sub-tokens of a particular word, the sub-word level representations obtained from the BERT models are translated back to word-level representations. This allows room for a novel fusion architecture to better the performance of classification. A Bidirectional LSTM model is utilized to capture the variations in word patterns for accurate classification, backing up the novel architecture. The ensemble results in an accuracy of 79.67% on Tamil tweets. Another multilingual BERT cased model, presented by Ghanghor et al. (2021), was employed in offensive language detection and troll meme classification for Dravidian languages like Tamil, Malayalam, and Kannada. Loss functions such as negative log-likelihood with class weights and Sadice were employed to tackle the class imbalance issues. mBERT-cased outperformed the other multilingual models used, namely, XLM-RoBERTa and IndicBERT; the meme classification task used this model for text modality. The mBERT-cased achieved a weighted average F_1-score of 0.55 in Tamil troll meme classification and 0.75 for Tamil offensive language identification.

Dave et al. (2021a) proposed a system that uses Google's MuRIL pre-trained transformer model to detect hope speech in a code-mixed dataset. In their approach, TF-IDF character n-grams and the MuRIL model have been employed to extract the features from the text. Then a logistic regression model and a linear support vector machine are used as downstream classifiers to train on the embedded data and perform predictions. The model achieved a weighted average F_1-score of 0.92 on English data, 0.75 on English-Malayalam code-mixed data, and 0.57 on English-Tamil code-mixed data. The same authors, Dave et al. (2021b), have worked on another similar research that used the same system to identify the offensive language in a code-mixed data set obtained from YouTube comments in which their model achieved a weighted average F_1-score of 0.64 on Kannada data, 0.95 on Malayalam data and 0.71 on Tamil data. Earlier work such as Loria (2018) applied machine learning techniques, for automatically detecting the language of the web documents/web pages in addition to the works that purely focus on classification of web pages (Rajalakshmi et al., 2020). Chinnappa (2021) analyzed the usage of language models and BERT-based models to detect hope speech in code-mixed and transliterated text datasets in her work. The system used a two-phase approach where various language models are used to predict the language of the text and a pre-trained BERT model for each language to detect the presence of hope speech. To identify the offensive contents in Dravidian languages Yasaswini et al. (2021) explored transformer models such as ULMFiT, which produced F_1 scores of 0.9603, 0.7895 on Malayalam and Tamil, and DistilmBERT, which obtained a score of 0.7277 on Kannada. Sivalingam and Thavareesan (2021a) focussed on Tamil language alone and applied 4 classifiers viz., Support Vector Machine, Random Forest, K - Nearest Neighbor and Naive Bayes algorithm, with χ^2 feature selection technique, TF-IDF and BoW feature representation strategies. Linear SVM achieved the highest accuracy of 76.96% when TF-IDF was used as the feature representation technique.

The study from Hande et al. (2021) shows the usage of pseudo-labeling to find offensive content in code-mixed Dravidian languages. The method first categorized code-mixed social media comments and posts into Tamil, Kannada, or Malayalam. All

the code-mixed texts were transliterated into the appropriate Dravidian languages Kannada, Malayalam, or Tamiland then pseudo-labels were created for the transliterated dataset. A unique dataset termed CMTRA was produced by combining the two datasets with the help of the created pseudo-labels. The authors assert that this strategy enhances the amount of training data for the language models because Dravidian languages are under-resourced. According to the authors, fine-tuning ULMFiT on the custom dataset yields the best results among the benchmarked models on Tamil-English, obtaining a weighted F1-Score of 0.7934 while achieving competitive weighted F1-Scores of 0.9624 and 0.7306 on the code-mixed test sets of Malayalam–English and Kannada-English, respectively. Thavareesan and Mahesan (2021b) establish a system that predicts the sentiments expressed in Tamil texts using K-means clustering and K-nearest neighbor classifier. The authors used the UJ_MovieReviews corpus from numerous sources in order to test this strategy. Clustering the corpus using class-wise information and clustering without class-wise information were two separate methods that were taken into consideration. These two clustering-based methods were tested by utilizing m-folds of training samples. Word embeddings from BoW and fastText are used in these studies. Each method is put to the test with a different number of centroids in order to investigate how it affects accuracy. The strategy with fastText and class-wise clustering with m-folds of the training set, according to the authors, produced the best results with an accuracy of 89.87%. The research by Sivalingam and Thavareesan (2021) focused on the prediction of offensive language in code-mixed Tamil text. Four different classifiers have been used for this purpose: Support Vector Machine, Random Forest, K-Nearest Neighbor, and Naive Bayes. BoW and TF-IDF have been used along with different combinations of n-grams to represent the text. The most relevant features are then selected using the χ^2 feature selection method. These selected features are then used to train the different classifiers. The authors claim that using this method of feature representation when used along with the Support Vector Machine and TF-IDF produced an accuracy of 76.96% on the corpus.

The study by Roy et al. (2022) examines the application of an ensemble model to identify hate speech and offensive language on social media sites by incorporating the results of transformer and deep learning-based models. For Malayalam and Tamil code-mixed data sets, the experimental results of the proposed weighted ensemble system surpassed state-of-the-art models by reaching weighted F_1-scores of 0.802 and 0.933, respectively. The authors of Bharathi and Silvia (2021) described an automatic detection of offensive language from Dravidian languages using a variety of machine learning techniques and they tried locating inappropriate language in a code-mixed data set of social media comments and posts written in Dravidian languages (Tamil, Malayalam, and Kannada). According to the authors, the system was able to achieve F_1-scores of 0.95 for Malayalam, 0.7 for Kannada, and 0.73 for Tamil on the testing data. When typing a text or converting printed or handwritten documents into editable formats, sentences may contain incorrect words, grammatical errors, or missing words. This study by Sakuntharaj and Mahesan (2021) proposes an effective method for identifying missing words in Tamil sentences and offering alternatives for them. The suggested method makes use of uni-gram, bi-gram, and tri-gram approaches at the word level. According to test findings, the suggested method finds every word that is absent from Tamil sentences. Additionally, according to test findings that were validated by a Tamil scholar, the generated suggestions are more than 99% accurate.

Three techniques were employed by Rajalakshmi et al. (2022a) to locate abusive comments in Tamil and Tamil–English data. They implemented this by making use of deep learning, machine learning, and transformer-based modeling. The Random Forest classifier, which has an F_1-score of 0.78 for Tamil data, produced the best results. They employed IndicBERT and mBERT, and for the Tamil dataset, mBERT produced the best results, with an F1 score of 0.70. They suggested that contextualized embeddings could improve the system's overall performance. To create feature vectors, Prasanth et al. (2022) applied the Random Kitchen Sink algorithm with the well-known TF-IDF with char-wb analyzers. Furthermore, they examined abusive comments in Tamil and Tamil–English data using Support Vector Machine as their downstream classifier. Concerning Tamil and Tamil-English comments, they obtained F1 scores of 0.32 and 0.25, respectively. To achieve better outcomes, they suggested adopting transformer-based models like BERT and LaBSE. Prasad et al. (2022) classified offensive comments in Tamil using transformer models like XML-RoBERTa and DeBERTA. They achieved the best results with XML-RoBERTa, which had an F1 score of 0.65, and a comparable outcome with DeBERTA, which had an F1 score of 0.57. To achieve better results for the Tamil dataset, they recommended experimenting with pre-trained models in Tamil and investing more effort into developing custom model architectures. LekshmiAmmal et al. (2022c) used multiple transformer-based models like mBERT, MuRIL, and Electra to identify offensive spam in Tamil. Out of the three models, their fine-tuned MuRIL gave the best F1 score of 0.44.

Sai and Sharma (2021) show how offensive speech identification in code-mixed languages can be performed using multilingual transformers. The paper discusses the usage of a novel technique for the feature representation of the code-mixed text. This algorithm called STT (Selective translation and transliteration) transliterates romanized native words into Tamil and then performs selective translation of some English text. Transformer models like XLM-RoBERTa are then used to obtain the word embeddings. A logistic regression model has then trained on the word embeddings and performs the detection. The authors claim that this setup enables them to achieve a weighted F1 score of 85%. Rajalakshmi et al. (2021) use a cosine similarity metric to identify the most similar terms in a phrase that enables to isolate the rare and important terms in a code-mixed Tamil corpus. This method is used to vectorize the text data. The work also discusses the usage of a data balancing technique called SOUP (Similarity-based Oversampling and Undersampling Preprocessing) to balance the dataset. The BERT model is then trained on these balanced vectors to detect the presence of offensive content. The authors claim that this system achieves an F1 score of 64%. Dowlagar and Mamidi (2021) use pre-trained transformer models to train on transliterated code-mixed data to detect offensive content. The system uses the NLTK English word corpus to detect English words in the code-mixed data. These words are then translated into the native script. The authors then use a class-balanced loss to balance the dataset. A pre-trained multilingual BERT model is then trained on this data. The authors claim that this system achieves a weighted F1 score of 80% on the Tamil–English code-mixed data. The work by Que (2021) discusses the detection of offensive content in Kannada using transformer models. The author uses XLM-RoBERTa to obtain

Label	Text	Transliteration	Translation
NOT	கண்டிப்பாக இந்த படத்தை திரையரங்கில் தான் பார்ப்பேன்	kandippaga intha padathai thiraiyarangil dhaan paarpen	I will definitely watch this movie in theater
OFF	உண்மையாகவே இது சைக்கோ படம் தான் ஒன்னும் புரியல	unmayagave idhu psycho padam dhaan onnum puriyala	Really, this is a psychotic movie, I do not understand anything

Fig. 2. Sample YouTube comments from the data set.

Table 1
Data set statistics.

Label	Total	Number of not-offensive	Number of offensive
Training set	4701	3774	927
Testing set	664	536	118
Validation set	1176	950	226

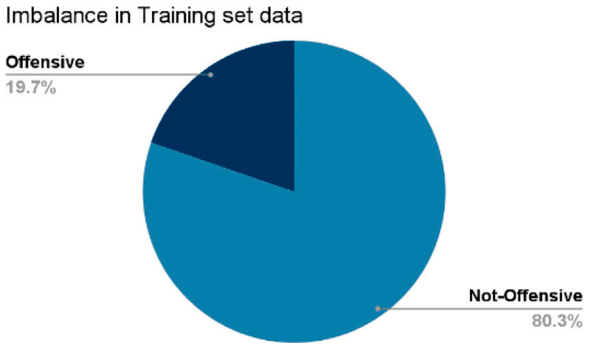

Fig. 3. Pie chart illustrating the imbalance of training dataset.

the word embeddings from the text data. These embeddings are then passed to an LSTM. The output from the LSTM is then passed to a classifier to obtain the prediction. This setup achieves a weighted F_1 score of 33% on the Kannada text data. The features extracted from BERT models were also used for the sentiment classification tasks (Sivakumar and Rajalakshmi, 2022). Also there are few works that focus on new metrics for determining the borrowing likeliness in the code-mixed language ?, which is useful for multi-lingual query processing.

3. Data set

The data set used for this research work is collected from HASOC-DravidianCodeMix Shared Task on Offensive Language Detection in Tamil and Malayalam, (Chakravarthi et al., 2021b). The data set contains Tamil YouTube comments, along with the labels and text IDs. The labels indicate whether the comment is offensive, not-offensive, or not-Tamil. Since there were only 3 not-Tamil category comments in the training set and no such labeled comments found in the test set, we have discarded those non-Tamil labeled samples from our corpus. Hence, we have viewed this as a binary classification problem, with offensive (labeled as OFF) and Not-Offensive categories (labeled as NOT) alone. Fig. 2 shows two examples of YouTube comments in the Tamil language taken from the considered data set. The first sentence with a NOT label (denotes a sample from the Not-offensive category) does not cause or express any displeasure to the intended audience. However, the example for the OFF label looks to be pejorative, conveying displeasure toward a movie simply because the speaker did not understand it.

In this dataset, we have a total of 6451 Tamil YouTube comments, in which 4701 comments were used for training and 1176 were used for validation. The remaining 664 samples were part of the testing set. This has been clearly illustrated in Table 1 with the corresponding statistics of offensive and Not-offensive labeled comments. It can be noticed that the number of NOT-offensive labeled data exceeds the number of Offensive labeled data significantly. As has been observed in Fig. 3, 80.3% of the training set contains NOT labels (not-offensive) and only 19.7% contains OFF labels (offensive). The class imbalance is present in the testing set with 82% NOT and 18% OFF samples, as well as in the validation set with 80.8% NOT and 19.2% OFF samples.

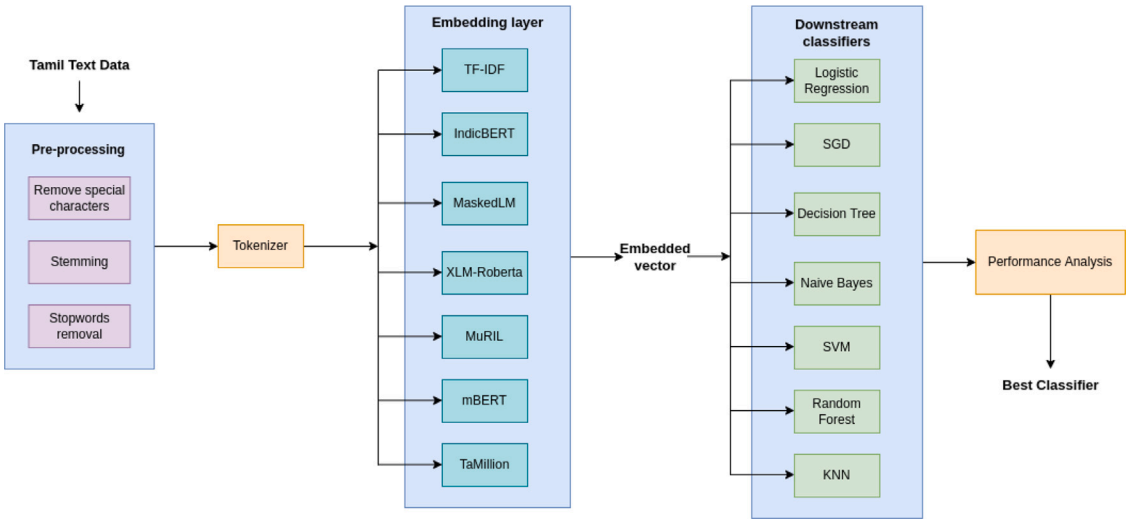

Fig. 4. Architecture of the proposed method.

Using the above data set, we have tried to find a suitable representation for Tamil language text data and also studied the effect of stemming and stop word removal by applying various methods. The details of the proposed methodology are presented in the following sections.

4. Proposed methodology

The proposed system is designed to automatically detect and identify hate speech or offensive content in YouTube comments in the Dravidian language, Tamil. The YouTube comments in the text form are pre-processed by removing the unnecessary special characters. The cleaned data is then subjected to different combinations of stemming and stop word removal, with two types of stemming algorithms and by varying the number of stop words used. The features are extracted by embedding this enhanced pre-processed (stemmed) data using the transformer-based pre-trained models such as mBERT, IndicBERT, base BERT model, XLM-RoBERTa, MuRIL, TaMillion and also using conventional TF-IDF. Then, the embedded data is used to train multiple binary classifiers and their performances are evaluated. Fig. 4 illustrates the architecture of the proposed methodology.

4.1. Pre-processing

This section gives a brief description of the pre-processing techniques used on the Tamil text data. The pre-processing methods utilized in this work include stop word removal and two variants of stemming. As Tamil is rich in morphology, we have enhanced the stemming process with a suitable morphological analyzer algorithm to study its effect on text representation and classification performance.

4.1.1. Stop word removal

Stop words are the words that are frequently used in a document. These words do not have much significance in the document and are hence removed during the pre-processing steps in any Natural Language Processing (NLP) task. NLP tasks generally involve a lot of texts and to avoid high-dimensional data, only the important words are considered while training a machine learning model. So, the stop words that do not convey any meaning are removed in the pre-processing stage itself independent of language. Having a little over 100 stop words in Tamil proves to be a major setback as in common languages like English and Hindi, there are more than 400 stop words and 200 plus stop words respectively. In this work, 125 Tamil stop words were used (Ashok, 2016). The total number of stop words varies from language to language. Fox (1989) has proposed a total of 421 stop words in English. However, Saini and Rakholia (2016) has mentioned that English has a total of 174 stop words. This is evident that there is no universally accepted fixed number of stop words for any language for now. Similarly, Ashok (2016) has proposed 125 stop words and Fayaza and Fathima Farhath (2021) has proposed 93 stop words in Tamil along with domain-specific stop words. The reason for this may be the number of corpus and algorithms in determining the stop words used by the researchers. Moreover, it can be noticed that the average number of stop-words for any given language could be estimated to be 200 (Saini and Rakholia, 2016). Hence, in this study, a total of 170 stop words have been used by combining the unique words proposed by both Ashok (2016) and Fayaza and Fathima Farhath (2021).

In some of the experiments, these stop words were removed and their performance was evaluated. A few examples are shown in Fig. 5 depicting the need for removing the stop words. The highlighted words in the first column (named Before stop word removal)

Before stopword removal	After stopword removal	Label
இது போன்ற திரைப்படங்கள் தொடர்ச்சியாக வர வேண்டும். அப்போதுதான் இளம் பெண்களும் அவர்களது பெற்றோர்களும் விழிப்புணர்வுடன் இருக்க உதவும். (Movies like this should keep coming. Only then young women and their parents will be aware.)	திரைப்படங்கள் தொடர்ச்சியாக வர வேண்டும். அப்போதுதான் இளம் பெண்களும் அவர்களது பெற்றோர்களும் விழிப்புணர்வுடன் இருக்க உதவும். (Movies should keep coming. Only then young women and their parents will be aware.)	NOT
நம்ம கிட்ட காடு இருந்த எழுதிகிடுவானுக ரூவா இருந்த புடுங்கிக்கிடுவானுக ஆனா படிப்ப மட்டும் எடுத்துக்கவே முடியாது (They'll take our forests if we have them, they'll snatch our money if we have it, but education is the only thing they can never take away)	நம்ம கிட்ட காடு எழுதிகிடுவானுக ரூவா புடுங்கிக்கிடுவானுக ஆனா படிப்ப எடுத்துக்கவே முடியாது (They'll take our forests if we them, they'll snatch our money if we it, but education is the thing they can never take away)	OFF

Fig. 5. Before and after stop words removal.

indicate the stop words in Tamil. In the first example, four (highlighted) words have been removed. These words can be translated to 'this' and 'like' respectively. These phrases do not contribute much meaning to the sentence and their removal does not alter the meaning. These words were correctly identified as stop words and are removed from the text. Similarly in the second example, the three phrases that are removed are 'have'("irunta") and 'only' ("maṭṭum") respectively. These words can also be removed as they do not contribute much to the meaning of the entire comment. The label 'OFF' represents offensive content and 'NOT' represents not offensive content.

4.1.2. Stemming

Stemming is a technique for lowering the inflections in a word, in order to extract their base form. Inflection is the process by which a word is altered to communicate a variety of grammatical categories such as tense, case, voice, aspect, person, number, gender, and mood. As a result, while a word may have multiple inflected forms, its existence within the same text adds redundancy to the NLP process. Storing only the stems reduces index size and improves retrieval accuracy (Frakes and Baeza-Yates, 1992). As a result, this method accounts for a significant portion of information extraction and retrieval. This pre-processing technique helps to build an effective model by normalizing the text. However, stemming in Tamil NLP tasks is quite uncommon and rarely implemented despite its potential advantages. There are various works that compare the performance of multiple Tamil Stemmers on Tamil text data like Gurusamy and Nandhini (2017), Lakshmi and Kumar (2014) and Thangarasu and Manavalan (2013). However, there are no works that compare the effect of stemming on the performance of a model classifying Tamil text. An affix stripping iterative stemming algorithm for Tamil, implemented using Snowball is used in this study, to reduce inflected words to their root form (Porter, 2001). Another stemming algorithm developed using the IndicNLP library's morphological analyzer (Kunchukuttan, 2020) was also used to stem the data. These stemmed texts were used later, to categorize the data as "offensive" or "not-offensive". 'OFF' indicates offensive content and 'NOT' represents not-offensive content. Fig. 6 shows some examples of how the text looks like before and after stemming. The highlighted words indicate the stemmed words. As Tamil is rich in morphology, we tried to apply these algorithms to study the effect of stemming on this dataset.

4.2. Feature representation

This section gives a brief description about the traditional feature extraction method used and the embedding techniques used. The traditional methods include TF-IDF and the embedding techniques include IndicBERT, XLM-RoBERTa, BERT, mBERT, MuRIL and TaMillion.

4.2.1. Traditional methods

TF-IDF is an abbreviation that stands for Term Frequency -Inverse Document Frequency. It is a traditional way of determining how relevant a word is to a text, in a sentence or corpus. TF or term frequency indicates the word's frequency in each document in the corpus. A term's specificity can be measured as an inverse function of the number of documents where it appears, and is given by Inverse Document Frequency or, IDF. The product of the two statistics, TF and IDF will give the TF-IDF score of the word in that sentence or document; the higher the score, the more notable that word is in that specific document. This method is used to vectorize the text for ease of training, and in this study, is used as a feature extraction technique.

Before stemming	After stemming	Label
எங்களுக்கு மண்ணு பொண்ணு இரண்டுமே முக்கியம் அதில் யாரு கையை வச்சாலும் கையை வெட்டுவோம் (Both the soil and gold are important to us and we will cut off the hand of whoever touches it)	எங்கள் மண்ணு பொண்ணு இரண்டும் முக்கியம் அதில் யாரு கையை வச்சா கையை வெட்டு	OFF
படம் அழகாக இருக்குங்க அத விட எதார்த்தமாயிருக்குங்க (The movie is beautiful more than that it is realistic)	படம் அழகாக இருக்கு அத விட எதார்த்தம்	NOT

Fig. 6. Before and after Stemming.

4.2.2. Embedding techniques

BERT by Devlin et al. (2018) based embedding model is used in this study. It accepts incomplete input text with a mask and predicts the masked word using the context words surrounding the mask. This model works well for self-supervised tasks, after which it can be fine-tuned for supervised NLP tasks. This model first performs text tokenization. One of the crucial tensors obtained from this procedure is input id. The label tensor is then generated by cloning this input id. The tokens can now be masked at random. The model is trained by randomly masking around 15% of the text and predicting the masked tokens using the next sentence prediction. Even though the model has not been trained on Tamil data, since it is a Unicode model it can be directly used on Tamil text.

The Tamil script was fed as it is, to the BERT model without transliterating. The model trains on the Tamil text and makes its predictions. The reason behind this choice was that this model was used as a baseline to test the other BERT-based models. The BERT model supports a Unicode model so it is possible to use it on Tamil text directly. And according to the BERT documentation, BERT (2018), it is an effective method to train on textual data.

The BERT multilingual base model (mBERT) from Devlin et al. (2018) is a BERT model pre-trained on Wikipedia data of multiple languages. The model is trained on the top 104 most spoken languages in the world using the masked language modeling objective. It is trained in a self-supervised method without using any labels to validate the model. The model is trained in a similar process to the BERT model. This allowed the model to learn representations for all the languages that it is trained on. The model can be used as an embedding layer by using the representations learned by the model for the intended language and training on them using any downstream classifier.

XLM-RoBERTa by Conneau et al. (2019), is a multilingual version of the RoBERTa model. It is a transformer-based model which is pre-trained on about 2500 Gigabytes of filtered CommonCrawl data. This data consists of texts from about 100 languages. The model is trained in a self-supervised manner which makes the model more efficient since this can be utilized in many scenarios and datasets. The masked language modeling (MLM) objective is used to train the model.

IndicBERT, by Kakwani et al. (2020) is a multilingual BERT model. It is a derivative of the ALBERT model (Lan et al., 2019). The model has been trained on a corpus containing about 9 million tokens. The authors have subjected the model to multiple tests on diverse natural language processing (NLP) tasks. They claim that IndicBERT is better than the other models as it uses around a tenth of the parameters, enabling it to process and learn at a rapid pace when compared to other models. This shows no effect in its results as it performs as well as or in some cases even better than comparable models used for Indian languages like DistilBERT, XLM-RoBERTa, etc. The ALBERT model from which IndicBERT is derived is originally derived from the BERT model. The model is trained on 12 major Indian languages. They are English, Bengali, Assamese, Hindi, Gujarati, Malayalam, Kannada, Oriya, Marathi, Telugu, Tamil, and Punjabi.

TaMillion by Doiron (2020) is a monolingual BERT model that is only pre-trained on Tamil corpus. As of October 1, 2020, it has been trained on 482MB of Wikipedia dumps and 11 GB of IndicCorp Tamil. With 224,000 steps, this model has been pre-trained on TPU. This approach is comparable to the Encoder that Classifies Token Replacements Accurately (ELECTRA) from Google Research.

Multilingual Representations for Indian Languages or MuRIL is a pre-trained BERT model from Google's Indian research unit, (Khanuja et al., 2021). It is a multilingual language model trained exclusively on Indian text corpora. The corpora used by the authors is also subject to augmentation by translation and transliteration. The model is trained on 17 languages containing English and 16 other Indian languages. It is trained in two phases: masked language modeling and translation language modeling. By testing the model on Indian language tasks and comparing it against the mBERT model, the authors report that MuRIL performs better than mBERT across all objectives. The MuRIL model can also be used as an embedding layer.

Table 2
Classifiers and their abbreviations.

Classifier	Abbreviation
Logistic Regression	LR
Support Vector Machine	SVM
Naive Bayes	NB
Stochastic Gradient Descent	SGD
Decision Tree	DT
Random Forest	RF
Majority Voting Ensemble	Ens

4.3. Classifiers

This study employs the following downstream models to train on the embedded data, acting as binary classifiers to take the embedded data as input and classify them into 'OFF' or 'NOT' (predicts whether or not the comment is offensive). The training data is used to train this model, and the testing data is used to evaluate its performance. The collected outcomes are later documented.

Decision tree is a simple, powerful, and useful tool used for the prediction and classification of data. The basic goal of this model is to forecast the values of an instance by learning decision rules based on data attributes. An instance starts from the root node, then goes down to the next corresponding node according to the classification result by the test attribute. This instance further moves down the tree branch and continues the same process with the next sub-tree.

The gradient descent algorithm is a repetitive algorithm. It takes a random value of a prediction function and calculates the error function. Then it tries to minimize the error by iteratively modifying the prediction function. This process is continued until the error function reaches its minimum value. The stochastic gradient descent algorithm optimizes this approach by calculating the error for any one random data point at iteration and tries to reduce the value of the error function accordingly, proving to be a less intensive task as compared to the usual gradient descent model.

Naive Bayes, a Bayes theorem-based, probabilistic machine learning algorithm, is used in a wide range of classification application domains. It is very good at dealing with Natural Language Processing (NLP) problems. The underlying concept of Naive Bayes is that each feature contributes equally and independently to the result. It presents itself as a rapid, precise, and easily scalable algorithm even with large datasets.

The logistic regression model is a simple statistical algorithm that is used as a classification model. The model is generally trained on data with categorical labels. This model is a modification of the Linear Regression model. The linear regression model is modified by passing the output obtained as input to a logistic function. The output of this function is the required predicted label. This is done by making use of a technique called a decision boundary. The decision boundary acts as a cut-off point, which helps in separating one label from another. The output from the regression model is passed through a sigmoid activation function. This function takes any number as input, normalizes it, and gives an output between 0 and 1.

Support vector machine is a supervised machine learning model that can be used for both regression and classification tasks. However, it is primarily used for classification problems. It generates a hyperplane or set of hyper-planes in a high or infinite-dimensional space, which can then be used for classification, regression, and other purposes. Numerous hyper-planes could be used to separate the two classes of data points. The goal is to find a plane with the greatest margin, i.e. the greatest distance between data points from both classes. Maximizing the margin distance provides some reinforcement, allowing future data points to be classified with greater certainty. For many applications Rajalakshmi and Aravindan (2018), it is found to be suitable.

Ensemble learning is the process of systematically generating and combining many models, such as classifiers, to tackle a specific computational intelligence problem. Ensemble learning is commonly used to improve the performance of a model (classification, prediction, function approximation, etc.) or to reduce the risk of an unintentional poor model selection. In a nutshell, it makes the resulting model less data sensitive and more flexible. Ensemble methods are of two types, depending on how it is trained: bagging if parallel, and boosting if sequential.

The Random Forest is one such ensemble model, that uses bagging and decision tree as the individual models. A significant number of relatively uncorrelated models (decision trees) acting together will surpass any of the constituent models individually. Here, the main element is the low correlation between models. Even though some trees may predict incorrectly, many others will be correct, allowing the trees to move in the right direction as a group.

Yet another ensemble learning technique is the majority voting ensemble. In its most basic form, majority voting is a technique in which a model makes a prediction that is heavily predicted by multiple other weak models. In a classification task, the estimates of the weak models are tallied as votes, and the prediction with the most votes is the final prediction for that specific set of input variables. Various models make up the weak learners who participate in voting. The learners used in this task are Logistic Regression, Naive Bayes, Stochastic Gradient Descent, Decision Tree, and Random Forest.

For ease of representation, the above-mentioned models are abbreviated as given in Table 2.

4.4. Evaluation metrics

Performance metrics like accuracy, precision, recall, and F_1-score are utilized to analyze the downstream classifier's effectiveness. The ratio of true outcomes to the total number of cases studied is referred to as accuracy, Accuracy is used as the evaluation metric in this case since it is a viable choice for classification issues that are well balanced and not skewed or have no class imbalance. The formula is given in Eq. (1) where True Positive is indicated by TP, True Negative is indicated by TN, False Positive is indicated by FP and False Negative is indicated by FN. As the accuracy alone is not sufficient to measure the true performance, we have also used precision, recall, and F_1 as the performance metrics.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F_1-score is a harmonic mean of precision and recall. The formula for F_1-score is given in Eq. (4).

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

Since the data set is imbalanced one with more non-offensive content and 1/4 of offensive content, the weighted average is taken for precision, recall, and F_1-score. The precision, recall, and F_1-scores for each label are calculated here, and the average is then weighted by support. It can lead to an F_1-score that is not in the center of precision and recall. It is meant to be used to emphasize the value of certain samples in comparison to others. The competition that the dataset is part of, FIRE 2021's Dravidian CodeMix shared task (Chakravarthi et al., 2021b), used the weighted F_1-scores to rank the submissions made by different teams. This can be due to the fact that it is an imbalanced dataset with lesser OFF labels (offensive category) than NOT labels (not-offensive category). Focusing on the performance of the model on any one label may not be sufficient and the F1 score does not reflect the contribution of each class weighted by the number of samples in every class. So, to appropriately summarize the performance of the model on both OFF and NOT labels, the authors chose a weighted F-score. Other related works that used similar imbalance data sets (Ghanghor et al., 2021) have also preferred weighted F_1-score to showcase the performance of their respective systems.

5. Experiments and results

All of the models used for detecting offensive content in English are trained using a large corpus of English text. As a result, these models are not appropriate for identifying offensive content in other languages, and this holds for Tamil as well because the data is written in native/Tamil script and Tamil is a challenging language. We have carried out various experiments to identify the offensive contents in the Tamil YouTube comments by employing different methods. We present below, the details of the experiments and discussed the results of each experiment along with the inferences that we observed.

We conducted this experimental study with the following objectives:

1. To study the need for better feature representation for Tamil YouTube comments using machine learning and deep learning methods
2. To study the impact of stop words in feature representation
3. To study the effectiveness of stemming using existing algorithms
4. To find the best feature embedding technique for Tamil YouTube comments using monolingual and multi-lingual models.
5. To determine the best classification technique and to study the effect of ensemble learning

5.1. Baseline experiments with machine learning techniques

In this experiment, we studied the need for a suitable feature representation of Tamil text. The data considered for this study is in the form of text in Tamil script along with the associated labels viz., offensive and not-offensive. The data is pre-processed by removing the special characters like [, +, /, #, @, &, etc. Now, the TF-IDF feature extraction method is introduced. For the TF-IDF, the input is a sequence of items, with a decode error parameter set to Unicode Decode Error which is represented as a 'strict' parameter. This process, before tokenization, converts all characters into lower case letters and the features are made from words with no stop words. The maximum and minimum n-gram ranges are both set to 1,and the minimum document frequency is set to 5, so terms below this threshold are ignored. Here, a single YouTube comment refers to a document. With a validation size of 20% and a random state of 42, this data is split into training and validation sets. After converting the data into features using TF-IDF representation, various classifiers are employed. The testing data set uses the same set of pre-processing and feature extraction techniques as the training data set. Finally, all of the predictions are examined. The results of all the classifiers with TF-IDF as the feature extraction technique are shown in Table 3.

Here, ensemble learning is done through a majority voting ensemble, in which the three best models are chosen for the ensemble. In this case, the majority voting methods are Random Forest (RF), Decision Tree (DT), and Naive Bayes (NB) algorithm. The label NOT denotes non-offensive content, while OFF denotes offensive content. It is noticed that the models Logistic Regression (LR) and

Table 3
Performance analysis of TF-IDF based feature representation.

Metrics	Label	LR	SVM	SGD	RF	DT	Ens	NB
Accuracy		0.81	0.81	0.78	0.80	0.72	0.70	0.65
Precision	NOT	0.82	0.82	0.81	0.82	0.81	0.81	0.82
	OFF	0.00	0.00	0.09	0.07	0.11	0.15	0.17
	Weighted avg	0.67	0.67	0.68	0.68	0.68	0.69	0.70
Recall	NOT	0.99	0.99	0.94	0.97	0.87	0.82	0.75
	OFF	0.00	0.00	0.03	0.01	0.08	0.14	0.23
	Weighted avg	0.81	0.81	0.78	0.80	0.72	0.70	0.65
F_1-score	NOT	0.90	0.90	0.87	0.89	0.84	0.82	0.78
	OFF	0.00	0.00	0.04	0.02	0.09	0.14	0.19
	Weighted avg	0.74	0.74	0.72	0.73	0.70	0.70	0.67

SVM fail to classify offensive content, resulting in zero precision and recall leading to F_1 score of 0. On the other hand, despite having a lower F_1-score than the other classifiers, NB exhibits a higher overall classification for both offensive and non-offensive content. This is due to the fact that NB has higher precision and recall for both labels than other models. Following NB, the ensemble classifier achieved a strong overall outcome in terms of precision, recall, and F_1-score. To get even better performance, transformer-based models can be implemented.

From the above results, we conclude that the traditional TF-IDF-based representation is not sufficient for Tamil text. To get even better performance, we experimented with various embedding techniques applying transformer models and it is presented in the subsequent sections.

5.2. Identifying the appropriate embedding technique

Text embedding is a technique where mathematical functions are used to represent text as numeric vectors. These vectors can be used to train machine learning models to perform NLP tasks. TF-IDF is one such form of text embedding. It is applied for many NLP tasks such as sentiment analysis (Soubraylu and Rajalakshmi, 2021) and short text classification (Rajalakshmi, 2014). However, classic embedding models have one drawback. They are subjected to polysemy disambiguation, where a token or word which may have more than one meaning is represented by the same vector. This leads to the tokens which specify different meanings in the text having the same representation. BERT-based deep learning models overcome this by utilizing ELMO context embedding (Peters et al., 2018) and multiple bi-directional transformers. The value of the vector assigned to a token is calculated contextually thereby assigning a different vector to the same token. Pre-trained BERT models take this one step further by training on similar data, thus ensuring a better representation of the language. As the Tamil language also contains polysemy words as shown in Fig. 1, instead of simple TF-IDF methods, we preferred BERT-based models. The BERT-based models used in this work for text embedding are IndicBERT, XLM-RoBERTa, mBERT, MuRIL, TaMillion, and base BERT. The models are imported using the Hugging Face Transformer library for Python. The texts after pre-processing are tokenized using the tokenizer provided for each BERT model. The maximum token length is set to 512. Tokens larger than 512 characters are truncated. The tokenized text is fed to the models which perform text embedding and give an output in the form of numeric vectors. The model performs max pooling on the vectors and a final vector of length 768 is obtained from the model. The various downstream classifiers are then trained on these vectors. The performance of the models is analyzed using the testing data set. Fig. 7 shows the results of the models when validated using the testing data.

From Fig. 7, it can be inferred that the most reliable and best-performing classifier is the majority voting ensemble. This is due to the fact that it is the best for MuRIL and the joint best for BERT, XLM-RoBERTa, and TaMillion. The second most reliably performing classifier is the Random Forest classifier. This high and consistent performance can be attributed to the fact that both models are ensemble models. Ensemble models generally are more robust and perform better than probability-based or distance-based models. They increase the performance by reducing the variance of the prediction. This is done by introducing bias as they combine multiple weak learners into a single model. Each learner's prediction is taken into account and the final prediction is a combination of all the predictions. The performance of the Naive Bayes classifier varies across all the embedding models. This is due to the fact that being a probabilistic model, its performance is highly dependent on the features it trains on. The more the embedding technique is able to properly represent the text, the higher the Naive Bayes classifier performs. The Logistic Regression, Support Vector Machine, and Stochastic Gradient Descent (SGD) classifiers mostly stay consistent across all the embedding models. The Logistic Regression and SVM are boundary-based models. They are ineffective in a situation where a clear geometric boundary does not exist between data pertaining to both labels. Text embedding algorithms do not normally produce such data and thus the models find it difficult to train on them. The SGD classifier uses the SGD optimizer to learn the parameters. However, a drawback of the algorithm is that it uses the same learning rate for all the parameters. This is not an issue when the model is limited to a few parameters, but text embedding results in a vector of length 768. Thus the classifier has to learn all the 768 parameters. When all the parameters have the same learning rate, the descent sometimes overshoots the required trajectory for a parameter. This leads to poor performance. The decision tree classifier performs reasonably well when compared to the other classifiers, but due to it being a tree-based classifier, it is susceptible to over-fitting. This can be the reason why it is not able to match the performance of the ensemble models. The different embedding techniques too play a role in the performance of the entire system. Fig. 8 displays the comparison between the various embedding techniques with the majority voting ensemble as the downstream classifier.

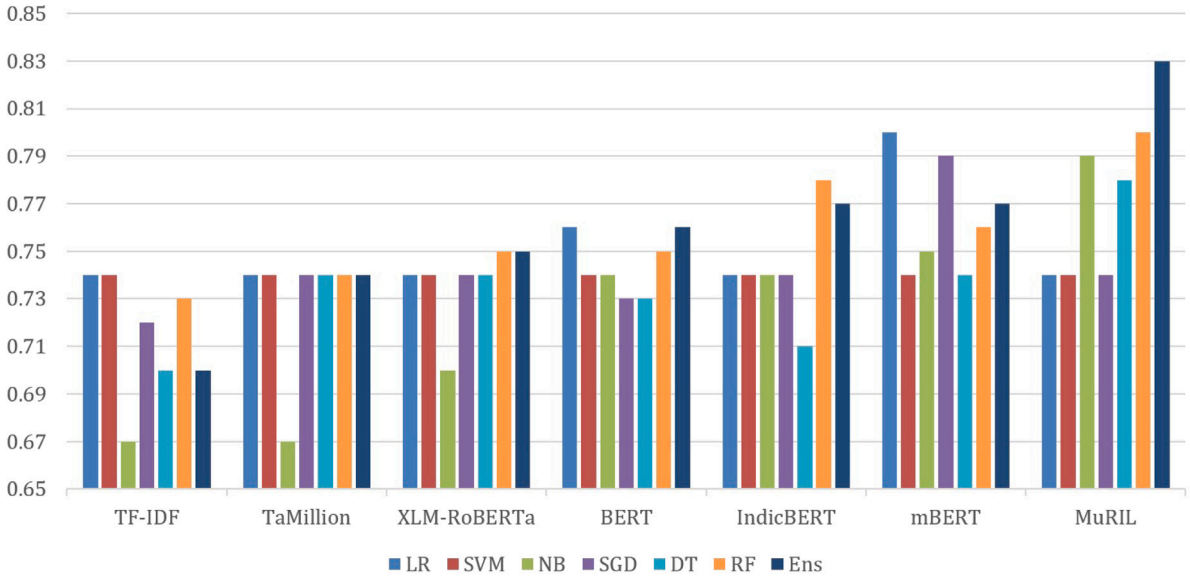

Fig. 7. Weighted average F_1-score of the text embedding techniques along with the various classifiers.

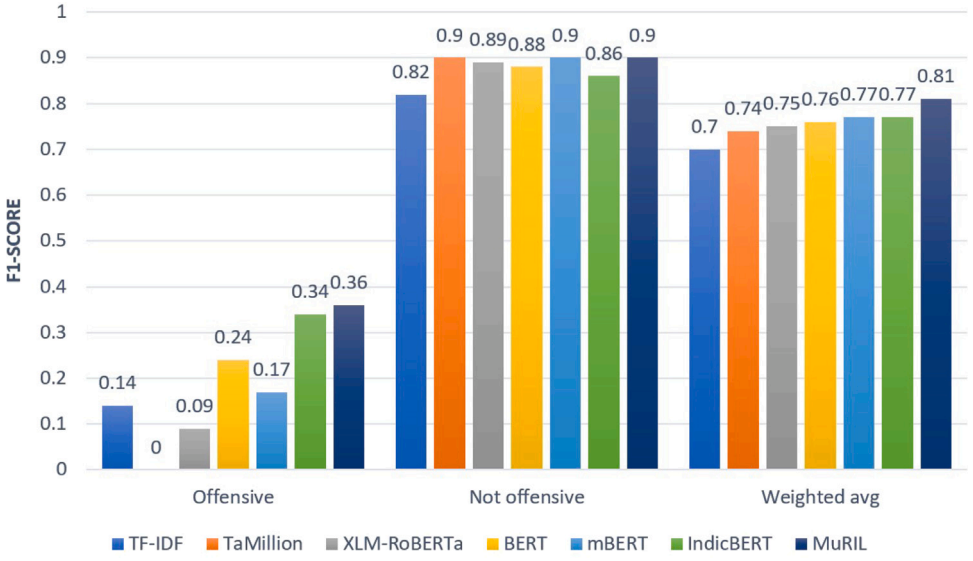

Fig. 8. F_1-score comparison of the text embedding techniques Ensemble classifier.

From Fig. 8, it is clear that both the MuRIL and the IndicBERT models provide the best performance with the majority voting ensemble, with MuRIL performing a little better than IndicBERT. Both models have been exclusively pre-trained on Indian language data and thus have a better representation of Tamil than XLM-RoBERTa, BERT, and mBERT. MuRIL especially is also trained on translated and transliterated data, which attributes to its robustness and high performance. The mBERT model performs better than the base BERT model as it has been pre-trained on a large corpus of the top 104 most spoken languages in the world including Tamil. Theoretically, XLM-RoBERTa and TaMillion are supposed to achieve better results as the former has been trained in about 100 languages including Tamil while the latter has been trained exclusively on Tamil data. However, this can be explained since XLM-RoBERTa is based on the RoBERTa model by Liu et al. (2019) and TaMillion is a pre-trained version of Google's ELECTRA from Clark et al. (2020), whereas the other models are derived from BERT (other than IndicBERT). The RoBERTa model introduced dynamic masking while removing Next Sentence Prediction (NSP) from the BERT model. However, in offensive content identification tasks, NSP is required to learn the context of a sentence (Liu et al., 2019). The context of a sentence indicates the presence or absence of offensive content in the text. Therefore XLM-RoBERTa model does not perform as well as the other models. The ELECTRA model removes the MLM objective from the BERT model and performs Replaced Token Detection (RTD). The RTD works by corrupting

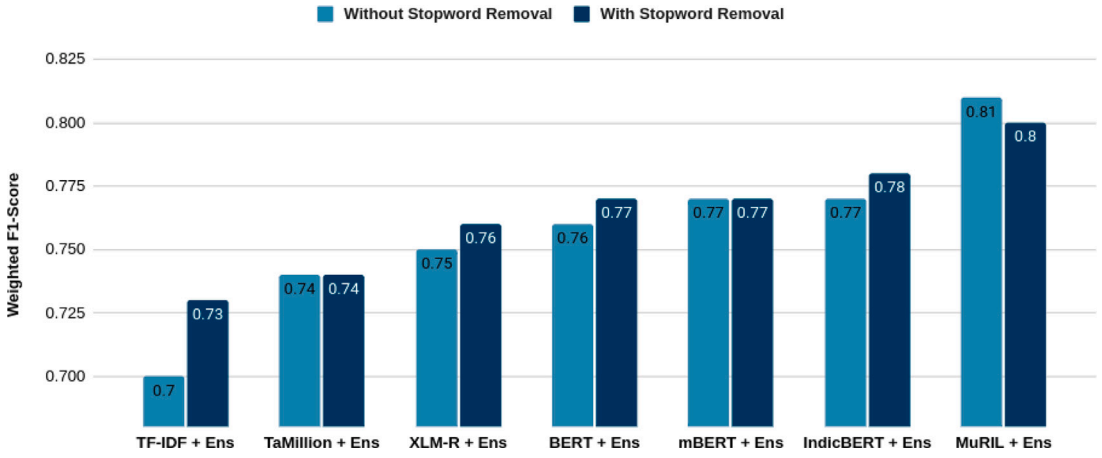

Fig. 9. Effects of stop word removal using 125 stop words.

one of the tokens instead of masking it and predicting whether each token is corrupted or not. Similar to the RoBERTa model the removal of the masking objective inhibits the model from learning the context of the sentence. The context is crucial in identifying the presence of offensive content in the sentence thus leading to poor performance of the TaMillion model. Overall the best combination of the embedding model and the downstream classifier can be concluded as MuRIL with a majority voting ensemble classifier.

It is to be noted that the poor performance of every model on classifying the Offensive label when compared to the Not-Offensive label is because there are more Not-Offensive data in the data set than Offensive labels. This imbalance causes the models to prefer the performance of the Not-Offensive labels during the learning process and the error in the Offensive labels contributes little to the total error.

5.3. Effects of stop word removal and stemming

This section shows the effect of how the results in the models using Ensemble as their downstream classifiers change by removing stop words, applying stemming, and using a combination of both stop words removal and stemming.

5.3.1. Effects of stop word removal

Fig. 9 illustrates a bar graph that shows the weighted F_1-score of all the embeddings and TF-IDF with the classifier as an ensemble. It can be seen that TaMillion and mBERT do not show any difference in F_1-score after removing stop words. However, three embedding models, namely, XML-RoBERTa, BERT, and IndicBERT show a slight increase when stop words are removed. TF-IDF shows a good increase in F_1-score by 3%. It can also be inferred from the graph that MuRIL shows a slight drop in F_1-score when stop words are removed. Overall, it can be concluded that the change in F_1-score with and without stop word removal is inconsistent. The reason may be due to the less number of stop words, we used only 125 stop words from Ashok (2016). Hence, another collection of 45 stop words from Fayaza and Fathima Farhath (2021) is also added, to study the significance of stop words. The outcomes of the models generated did not significantly alter after applying the new set of stop words, which consisted of 170 stop words in total. However, there was a significant change in the TaMillion model where the F_1-score of Offensive text was increased to 0.34. The results of the Offensive texts are better using more stop words even if the overall weighted F_1-score has been reduced. The results of TaMillion using ensembles are shown using 170 stop words in Table 4. It is clear that adding more stop words did not significantly enhance the results of all the other models but showed some improvements for Offensive text in TaMillion. This may be because some of the stop words used help the model comprehend the context of the phrase, and by deleting them, the model may lose that understanding. Stop word removal also suppresses the transformation and masking strategies utilized by the multilingual feature models that were used in this study which were trained in multiple languages. On the other hand, TaMillion is trained only on one language, Tamil, which ultimately shows some improvement upon using more stop words in the pre-processing step. Also, there is no clear evidence that every stop word does not contribute to classification. In Section 5.3.3, 170 stop words were used.

5.3.2. Effect of stemming

Stemming in Tamil NLP tasks is quite uncommon and rarely implemented despite its potential advantages. Various works compare the performance of multiple Tamil Stemmers on Tamil text data like Gurusamy and Nandhini (2017), Lakshmi and Kumar (2014) and Thangarasu and Manavalan (2013). We compare the effect of stemming on the performance of a model classifying Hate speech and offensive content in Tamil text written in native script. Table 5 shows the performance of all the embedding models without stemming with the majority voting ensemble as the downstream classifier. It can be noticed that MuRIL + Ensemble, without stemming, gives the best weighted average F_1-score of 0.81. We have analyzed the importance of stemming by applying two methods viz., Affix Stripping Iterative Stemmer and IndicNLP Morphological Analyzer.

Table 4
Performance after adding more stop words to TaMillion + Ensemble.

Labels	Using 170 stop words			
	Accuracy	Precision	Recall	F_1-score
Not-offensive	0.52	0.87	0.48	0.62
Offensive	0.52	0.22	0.69	0.34
Weighted average	0.52	0.76	0.52	0.57

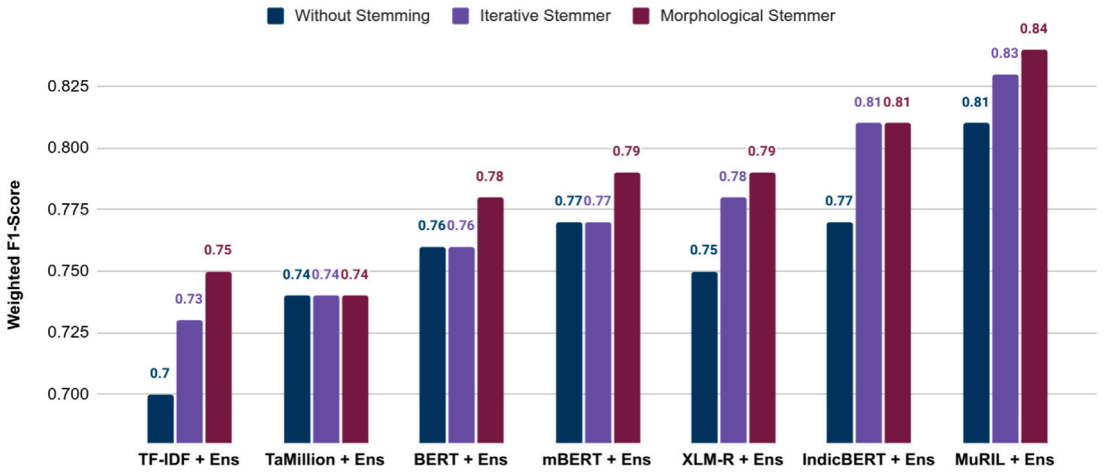

Fig. 10. Effects of Stemming.

Table 5
Performance of all embedding models without applying stemming techniques.

Embedding with ensemble	Accuracy	Precision	Recall	F_1-score
TF-IDF	0.70	0.69	0.70	0.70
TaMillion	0.82	0.67	0.82	0.74
XLM-R	0.81	0.73	0.81	0.75
mBERT	0.83	0.79	0.83	0.77
BERT	0.79	0.75	0.79	0.76
IndicBERT	0.77	0.76	0.77	0.77
MuRIL	0.83	0.81	0.83	0.81

The performance of the models when the text is stemmed using Affix Stripping Iterative Stemmer and IndicNLP's Morphological Analyzer is given in Table 6. The general trend from the results is that the system performs marginally better when the Tamil text data is stemmed. Using Affix Stripping Iterative Stemmer, the results for XLM-RoBERTa, IndicBERT and MuRIL drastically improve, while the rest stay the same. It can be noticed that MuRIL + Ensemble, without stemming, gives the best weighted average F_1-score of 0.81 which is comparatively less than that of the same model with stemming. This trend can also be seen in Fig. 10 which graphically displays the effect of stemming on the weighted average F_1-score of MuRIL with various downstream classifiers. Therefore, a conclusion can be reached that pre-processing Tamil text data by stemming it, does increase the performance of the system. IndicNLP's morphological analyzer completely outperforms the previous Affix Stripping Iterative Stemmer across all models. All results are a little better or at least equal to the Iterative Stemmer. It may be since Tamil is rich in morphology. The highest performance is obtained with the MuRIL model with the majority voting ensemble classifier. This model manages to achieve accuracy, precision, recall and weighted F_1-score of 0.86, 0.84, 0.86, and 0.84 respectively. The conclusion from the analysis is that the best system for the identification of offensive content in Tamil text data with the current setup involves stemming (with IndicNLP Morphological analyzer) the text, embedding it using the MuRIL model and using a majority voting ensemble as a downstream classifier to train on the embedded data.

As the IndicNLP morphological analyzer performed better than the affix stripping iterative Stemmer, we have preferred this method for all the subsequent experiments in our study. Another reason for choosing the IndicNLP approach is that they (Kunchukuttan et al., 2020) collected a large Tamil Corpus with a wide range of topics with CommonCrawl. Their Tamil corpus statistics reveal that they used 20,966,284 Tamil sentences and 362,880,008 tokens. They have also shown that their unsupervised morphanalyser outperforms other algorithms and it is suitable for downstream tasks with their detailed experimental study.

Table 6
Effects of stemming using affix stripping iterative Stemmer and IndicNLP morphological analyzer.

Embedding with ensemble	With affix stripping iterative Stemmer				With IndicNLP morphological analyzer			
	Accuracy	Precision	Recall	F_1-score	Accuracy	Precision	Recall	F_1-score
TF-IDF	0.81	0.67	0.81	0.73	0.80	0.73	0.80	0.75
TaMillion	0.82	0.67	0.82	0.74	0.78	0.72	0.78	0.74
XLM-R	0.83	0.80	0.83	0.78	0.81	0.77	0.81	0.78
mBERT	0.82	0.77	0.82	0.77	0.81	0.78	0.81	0.79
BERT	0.81	0.76	0.81	0.76	0.82	0.78	0.82	0.79
IndicBERT	0.81	0.80	0.81	0.81	0.83	0.81	0.83	0.81
MuRIL	0.80	0.82	0.83	0.83	0.86	0.84	0.86	0.84

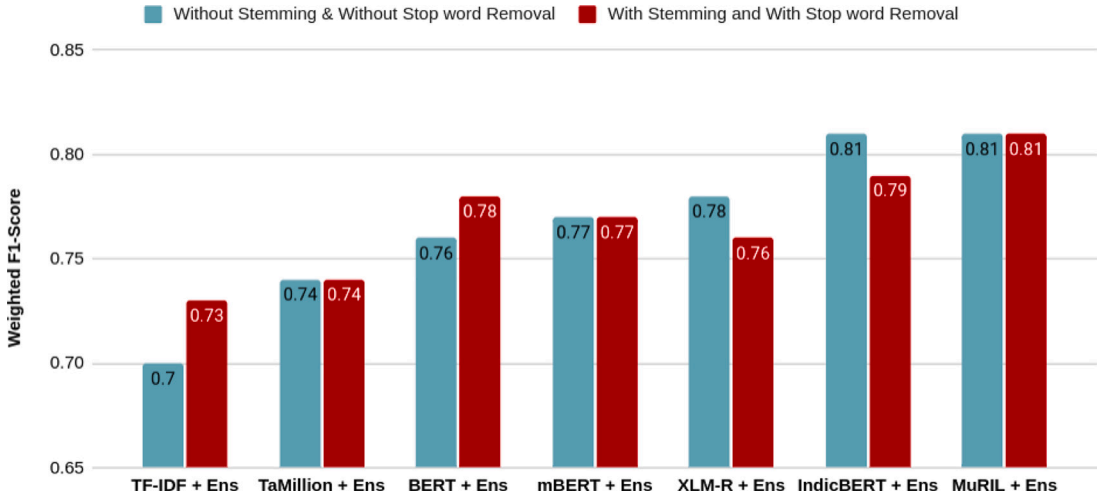

Fig. 11. Effects of Stemming and stop word removal.

5.3.3. Effect of stemming and stop word removal

An interesting analysis would be performing both stop word removal and stemming on the texts before processing them and comparing the performance with the results of the unprocessed text (no stop word removal and no Stemming). Fig. 11 compares the performance of text pre-processed with both methods and text without the pre-processing. From the past analysis, the conclusion is that the stemming process increases the performance of the model while removing stop words provides inconsistent results. This trend is also seen in Table 7. However, the comparison brings to light an interesting observation. The performances of BERT-based models pre-trained on Tamil data either stay the same or decrease. The performances of TaMillion, mBERT, and MuRIL stay the same, whereas the performances of XLM-RoBERTa and IndicBERT decrease. The reason for this trend can be traced backed to the BERT model. BERT models work by learning the context of a token in a particular sentence. Some of the stop words chosen for this task are polysemous words that can have multiple meanings and in fact, can contribute to understanding the context of the sentence. Identifying offensive content relies majorly on the context of a sentence and removing such words that contribute to understanding the context should not be performed. There might also be another factor to explain the trend seen. The models pre-trained on Tamil data might not have removed the stop words before the training process. Therefore they might have trained on those words and may provide some value to the words which help in identifying offensive content. The stop words taken in this work is a general list of tokens considered to be stop words in the Tamil language. Stricter restrictions on the stop words taken by choosing tokens that do not provide any insight into the context might have led to better performance, but for the stop words taken, the results are inconsistent. So, we believe that there is a scope for further improvement, in this research direction.

5.4. Analysis of sampling techniques for handling the data imbalance

As discussed in Section 3, the considered data set is a highly imbalanced one with more not-offensive comments. This section shows the effect of how the results change by applying sampling techniques such as Oversampling and Downsampling methods to address the issue of imbalanced data. All the models used Ensemble as their downstream classifier and stemming is applied since it showed the best results as discussed in the previous sections.

5.4.1. Effect of oversampling method

The dataset used in this experimental study contains only 926 offensive comments but has 3774 non-offensive comments. To address the skewed distribution of the considered dataset, the SMOTE oversampling technique has been applied. This technique

Table 7
Effect of combining stemming and stop word removal.

Embedding with ensemble	Without stop word removal and stemming				With stop word removal and with stemming			
	Accuracy	Precision	Recall	F_1-score	Accuracy	Precision	Recall	F_1-score
TF-IDF	0.70	0.69	0.70	0.70	0.81	0.67	0.81	0.73
TaMillion	0.82	0.67	0.82	0.74	0.82	0.67	0.82	0.74
XLM-R	0.81	0.73	0.81	0.75	0.82	0.76	0.82	0.76
mBERT	0.83	0.79	0.83	0.77	0.82	0.77	0.82	0.77
BERT	0.79	0.75	0.79	0.76	0.80	0.77	0.80	0.78
IndicBERT	0.77	0.76	0.77	0.77	0.79	0.79	0.79	0.79
MuRIL	0.83	0.81	0.83	0.81	0.83	0.81	0.83	0.81

Table 8
Performance of all embedding models without applying sampling techniques.

Embedding with ensemble	Accuracy	Precision	Recall	F_1-score
TF-IDF	0.80	0.73	0.80	0.75
TaMillion	0.78	0.72	0.78	0.74
XLM-R	0.81	0.77	0.81	0.78
mBERT	0.81	0.78	0.81	0.79
BERT	0.82	0.78	0.82	0.79
IndicBERT	0.83	0.81	0.83	0.81
MuRIL	0.86	0.84	0.86	0.84

Table 9
Effect of applying oversampling and down sampling techniques.

Embedding with ensemble	With oversampling				With downsampling			
	Accuracy	Precision	Recall	F_1-score	Accuracy	Precision	Recall	F_1-score
TF-IDF	0.79	0.70	0.79	0.74	0.57	0.72	0.57	0.62
TaMillion	0.46	0.79	0.46	0.50	0.70	0.73	0.70	0.72
XLM-R	0.74	0.78	0.74	0.76	0.63	0.80	0.63	0.67
mBERT	0.73	0.78	0.73	0.75	0.70	0.80	0.70	0.73
BERT	0.71	0.74	0.71	0.72	0.59	0.59	0.64	0.64
IndicBERT	0.73	0.79	0.73	0.75	0.68	0.79	0.68	0.71
MuRIL	0.75	0.80	0.75	0.77	0.75	0.83	0.75	0.78

produces a balanced dataset by increasing the size of offensive comments equal to the non-offensive comments. Then, the same previously discussed methods have been applied for classifying the comments. Among all the combinations that were tried, MuRIL with a majority voting ensemble classifier performed the best with precision, recall, and F_1 of 0.80, 0.75, and 0.77 respectively. However, the system before applying the SMOTE technique can achieve better performance with a precision, recall, and F_1 score of 0.84, 0.86, and 0.84 respectively. It is evident from the results in Tables 8 and 9 that, this trend is present in every embedding model that was taken. Thus, it can be concluded that oversampling does not improve the overall performance of the system.

5.4.2. Effect of downsampling method

Another technique to balance an imbalanced dataset is downsampling. In this technique, a balanced dataset is produced by decreasing the size of the larger class (non-offensive comments), and making it equal to the smaller class (the offensive comments). This is done by selecting a subset of the larger class in a random process to avoid any source of bias. Then, the same previous methods for classifying the comments have been applied. Table 9 presents the results obtained for various experiments. It is evident from the results that performing downsampling leads to reduced performance across all embedding models. Thus, it can be concluded that downsampling does not improve the overall performance of the system.

Hence, we conclude that, there is still scope for future research in this direction too.

6. Comparative study

As discussed in Section 3, the dataset used in this study was collected from the HASOC'21 competition hosted by FIRE 2021's Dravidian CodeMix shared task. We compared the results of the proposed system with the other teams and it is shown in Table 10. Even though all the evaluation metrics precision, recall, and F_1-score are listed, the main ranking criteria is F_1-score. It could be noticed that Team KonguCSE has secured a decent F_1-score of 0.76. Following them comes Pegasus with a good F_1-score of 0.81. The next best team is AIML with good precision and recall scores of 0.823 and 0.843 respectively. We have achieved better results with a precision and recall of 0.84 and 0.86 respectively. Also, with an F_1-score of 0.84, the proposed approach provides a more promising result. As shown in Table 10, the proposed system performs better than the other reported outcomes.

Table 10
Comparison of proposed system (MuRIL + Ensemble) with existing approaches.

Team name	Precision	Recall	F_1-score
KonguCSE	0.74	0.79	0.76
Pegasus	0.81	0.80	0.81
AIML	0.82	0.84	0.82
Proposed (MuRIL + Ensemble)	0.84	0.86	0.84

7. Conclusion and future work

We have proposed a deep learning-based approach for identifying hate and offensive content identification in Tamil text, written in native script. The proposed approach identifies offensive content in Tamil text data by simple pre-processing techniques with an emphasis on performing stemming on the data. The inclusion of stop word removal shows inconsistent results, models pre-trained on Tamil show poor performance while the other models report an increase in performance. This may be since some of the stop words taken to contribute to the model understanding the context of the sentence and removing it may strip the model of that learning. Stemming normalizes the tokens which in turn results in a better representation of the various tokens that make up the entire text. To enhance stemming, two different stemming algorithms were applied. As the Tamil language is rich in morphology, the performance of the morphological analyzer-based method is found to be more significant than the simple stemming algorithms. With an extensive experimental study, it is shown that MuRIL-based text representation is better than other language models. The combination of the MuRIL model for performing text embedding and majority voting ensemble as the downstream classifier for training on the data is found to be the most effective resulting in a weighted F_1 score of 0.84. As the majority voting ensemble combines the predictions from multiple weak learners, the model is more robust and performs better than the other classifiers for the given task. Also, SMOTE-based oversampling and down sampling techniques were applied to balance the highly imbalanced data set. However, applying these sampling techniques does not seem to improve the performance of the system.

In conclusion, the best results using the current setup are achieved by using a combination of stemming the data, embedding it using MuRIL, and learning using a majority voting ensemble. This study is unique, as it is the first work that analyses the effect of stemming and stop-words for Tamil script with an emphasis on suitable text representation. The proposed system is shown to be effective in the offensive content identification task with precision, recall, and an F_1 score of 0.84, 0.86, and 0.84 respectively which is significantly better than the existing approaches. We hope to extend this further in our future work by combining more specific linguistic-based approaches and also strengthening the methods of stemming, especially for Tamil text, written in native script.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data was collected from the organizers of a shared task HASOC 2021 and the researchers with an agreement to use it for research purpose

References

Aleksandra, 2016. How reading online comments affects us. https://socialmediapsychology.eu/2016/10/05/onlineandsocialmediacomments/.

Ashok, R., 2016. TamilNLP. https://github.com/AshokR/TamilNLP/.

Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Rangel Pardo, F.M., Rosso, P., Sanguinetti, M., 2019. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In: Proceedings of the 13th International Workshop on Semantic Evaluation. Association for Computational Linguistics, Minneapolis, Minnesota, USA, pp. 54–63. http://dx.doi.org/10.18653/v1/S19-2007, URL https://aclanthology.org/S19-2007.

BERT, 2018. BERT Documentation. https://github.com/google-research/bert/blob/master/multilingual.md.

Bharathi, B., Silvia, A.A., 2021. SSNCSE_NLP@DravidianLangTech-EACL2021: Offensive language identification on multilingual code mixing text. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Kyiv, pp. 313–318, URL https://aclanthology.org/2021.dravidianlangtech-1.45.

Chakravarthi, B.R., Kumaresan, P.K., Sakuntharaj, R., Madasamy, A.K., Thavareesan, S., Chinnaudayar Navaneethakrishnan, P.B.S., McCrae, J.P., Mandl, T., 2021a. Overview of the HASOC-DravidianCodeMix Shared Task on Offensive Language Detection in Tamil and Malayalam. In: Working Notes of FIRE 2021 - Forum for Information Retrieval Evaluation. CEUR.

Chakravarthi, B.R., Kumaresan, P.K., Sakuntharaj, R., Madasamy, A.K., Thavareesan, S., Chinnaudayar Navaneethakrishnan, P.B.S., McCrae, J.P., Mandl, T., 2021b. Overview of the HASOC-DravidianCodeMix Shared Task on Offensive Language Detection in Tamil and Malayalam. In: Working Notes of FIRE 2021 - Forum for Information Retrieval Evaluation. CEUR.

Chinnappa, D., 2021. Dhivya-hope-detection@LT-EDI-EACL2021: Multilingual hope speech detection for code-mixed and transliterated texts. In: Proceedings of the First Workshop on Language Technology for Equality, Diversity and Inclusion. pp. 73–78.

Clark, K., Luong, M.-T., Le, Q.V., Manning, C.D., 2020. Electra: Pre-training text encoders as discriminators rather than generators. arXiv preprint arXiv:2003.10555.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V., 2019. Unsupervised cross-lingual representation learning at scale. CoRR URL http://arxiv.org/abs/1911.02116.

Dave, B., Bhat, S., Majumder, P., 2021a. IRNLP_DAIICT@LT-EDI-EACL2021: Hope speech detection in code mixed text using TF-IDF Char N-grams and MuRIL. In: Proceedings of the First Workshop on Language Technology for Equality, Diversity and Inclusion. pp. 114–117.

Dave, B., Bhat, S., Majumder, P., 2021b. IRNLP_DAIICT@ DravidianLangTech-EACL2021: Offensive language identification in Dravidian languages using TF-IDF Char N-grams and MuRIL. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. pp. 266–269.

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Doiron, N., 2020. Tamillion bert.

Dowlagar, S., Mamidi, R., 2021. OFFLangone@DravidianLangTech-EACL2021: Transformers with the class balanced loss for offensive language identification in Dravidian code-mixed text. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Kyiv, pp. 154–159, URL https://aclanthology.org/2021.dravidianlangtech-1.19.

Fayaza, F., Fathima Farhath, F., 2021. Towards stop words identification in Tamil text clustering. The Science and Information Organization.

Fox, C., 1989. A stop list for general text. In: Acm Sigir Forum. 24, (1–2), ACM, New York, NY, USA, pp. 19–21.

Frakes, W.B., Baeza-Yates, R., 1992. Information Retrieval: Data Structures and Algorithms. Prentice-Hall, Inc..

Garain, A., Mandal, A., Naskar, S.K., 2021a. JUNLP@DravidianLangTech-EACL2021: Offensive language identification in Dravidian langauges. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Kyiv, pp. 319–322, URL https://aclanthology.org/2021.dravidianlangtech-1.46.

Garain, A., Mandal, A., Naskar, S.K., 2021b. JUNLP@DravidianLangTech-EACL2021: Offensive language identification in Dravidian langauges. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Kyiv, pp. 319–322, URL https://aclanthology.org/2021.dravidianlangtech-1.46.

Ghanghor, N., Krishnamurthy, P., Thavareesan, S., Priyadharshini, R., Chakravarthi, B.R., 2021. IIITK@DravidianLangTech-EACL2021: Offensive language identification and meme classification in Tamil, Malayalam and Kannada. In: DRAVIDIANLANGTECH.

Guntuku, S.C., Buffone, A., Jaidka, K., Eichstaedt, J.C., Ungar, L.H., 2019. Understanding and measuring psychological stress using social media. In: Proceedings of the International AAAI Conference on Web and Social Media. 13, pp. 214–225.

Gurusamy, V., Nandhini, K., 2017. Stemming techniques for Tamil language. Int. J. Comput. Sci. Eng. Technol. 8 (6), 225–231.

Hande, A., Puranik, K., Yasaswini, K., Priyadharshini, R., Thavareesan, S., Sampath, A., Shanmugavadivel, K., Thenmozhi, D., Chakravarthi, B.R., 2021. Offensive language identification in low-resourced code-mixed Dravidian languages using pseudo-labeling. arXiv preprint arXiv:2108.12177.

Jayanthi, S.M., Gupta, A., 2021. SJ_AJ@DravidianLangTech-EACL2021: Task-adaptive pre-training of multilingual BERT models for offensive language identification. CoRR URL https://arxiv.org/abs/2102.01051.

Kakwani, D., Kunchukuttan, A., Golla, S., N.C., G., Bhattacharyya, A., Khapra, M.M., Kumar, P., 2020. IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In: Findings of EMNLP.

Kedia, K., Nandy, A., 2021. indicnlp@kgp at DravidianLangTech-EACL2021: Offensive language identification in Dravidian languages. CoRR URL https://arxiv.org/abs/2102.07150.

Khanuja, S., Bansal, D., Mehtani, S., Khosla, S., Dey, A., Gopalan, B., Margam, D.K., Aggarwal, P., Nagipogu, R.T., Dave, S., 2021. Muril: Multilingual representations for Indian languages. arXiv preprint arXiv:2103.10730.

Kumaresan, P.K., Premjith, R., Sakuntharaj, R., Thavareesan, S., Navaneethakrishnan, S., Madasamy, A.K., Chakravarthi, B.R., McCrae, J.P., 2021. Findings of shared task on offensive language identification in Tamil and Malayalam. In: FIRE 2021, Association for Computing Machinery, New York, NY, USA, pp. 16–18. http://dx.doi.org/10.1145/3503162.3503179.

Kunchukuttan, A., 2020. The IndicNLP Library. https://github.com/anoopkunchukuttan/indic_nl_library/blob/master/docs/indicnlp.pdf.

Kunchukuttan, A., Kakwani, D., Golla, S., C.G.N., Bhattacharyya, A., Khapra, M.M., Kumar, P., 2020. AI4Bharat-IndicNLP Corpus: Monolingual corpora and word embeddings for indic languages. http://dx.doi.org/10.48550/ARXIV.2005.00085, URL https://arxiv.org/abs/2005.00085.

Lakshmi, R.V., Kumar, S.B.R., 2014. Literature review: stemming algorithms for Indian and non-Indian languages. Int. J. Adv. Rese. Comput. Sci. Technol. 4, 2582.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R., 2019. Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.

LekshmiAmmal, H., Ravikiran, M., Madasamy, A.K., 2022c. NITK-IT_NLP@TamilNLP-ACL2022: Transformer based model for toxic span identification in Tamil. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Dublin, Ireland, pp. 75–78. http://dx.doi.org/10.18653/v1/2022.dravidianlangtech-1.12, URL https://aclanthology.org/2022.dravidianlangtech-1.12.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Loria, S., 2018. Textblob documentation. 2, (8), Release 0.15.

Mandl, T., Modha, S., Shahi, G.K., Madhu, H., Satapara, S., Majumder, P., Schäfer, J., Ranasinghe, T., Zampieri, M., Nandini, D., et al., 2021. Overview of the HASOC subtrack at FIRE 2021: Hate speech and offensive content identification in English and Indo-Aryan languages. arXiv preprint arXiv:2112.09301.

Mohanty, P., Arulmozi, S., 2010. On polysemy in Tamil and other Indian languages. In: Global Wordnet Conference. pp. 133–140.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L., 2018. Deep contextualized word representations. arxiv 2018. 12, arXiv preprint arXiv:1802.05365.

Porter, M.F., 2001. Snowball: A language for stemming algorithms. Published online URL http://snowball.tartarus.org/texts/introduction.html. Accessed 11.03.2008, 15.00h.

Prasad, G., Prasad, J., C, G., 2022. GJG@TamilNLP-ACL2022: Using transformers for abusive comment classification in Tamil. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Dublin, Ireland, pp. 93–99. http://dx.doi.org/10.18653/v1/2022.dravidianlangtech-1.15, URL https://aclanthology.org/2022.dravidianlangtech-1.15.

Prasanth, S.N., Aswin Raj, R., P, A., B, P., Kp, S., 2022. CEN-Tamil@DravidianLangTech-ACL2022: Abusive comment detection in Tamil using TF-IDF and random kitchen sink algorithm. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Dublin, Ireland, pp. 70–74. http://dx.doi.org/10.18653/v1/2022.dravidianlangtech-1.11, URL https://aclanthology.org/2022.dravidianlangtech-1.11.

Que, Q., 2021. Simon @ DravidianLangTech-EACL2021: Detecting offensive content in Kannada Language. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Kyiv, pp. 160–163, URL https://aclanthology.org/2021.dravidianlangtech-1.20.

Rajalakshmi, R., 2014. Supervised term weighting methods for URL classification. J. Comput. Sci. 10 (10), 1969.

Rajalakshmi, R., Aravindan, C., 2018. An effective and discriminative feature learning for URL based web page classification. In: 2018 IEEE International Conference on Systems, Man, and Cybernetics. SMC, IEEE, pp. 1374–1379.

Rajalakshmi, R., Duraphe, A., Shibani, A., 2022a. DLRG@DravidianLangTech-ACL2022: Abusive comment detection in Tamil using multilingual transformer models. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Dublin, Ireland, pp. 207–213. http://dx.doi.org/10.18653/v1/2022.dravidianlangtech-1.32, URL https://aclanthology.org/2022.dravidianlangtech-1.32.

Rajalakshmi, R., More, M., Shrikriti, B., Saharan, G., Samyuktha, H., Nandy, S., 2022b. DLRG@TamilNLP-ACL2022: Offensive span identification in Tamil usingBiLSTM-CRF approach. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Dublin, Ireland, pp. 248–253. http://dx.doi.org/10.18653/v1/2022.dravidianlangtech-1.38, URL https://aclanthology.org/2022.dravidianlangtech-1.38.

Rajalakshmi, R., Reddy, B.Y., 2019. DLRG@HASOC 2019: An enhanced ensemble classifier for hate and offensive content identification. In: Mehta, P., Rosso, P., Majumder, P., Mitra, M. (Eds.), Working Notes of FIRE 2019 - Forum for Information Retrieval Evaluation, Kolkata, India, December 12-15, 2019. In: CEUR Workshop Proceedings, 2517, CEUR-WS.org, pp. 370–379, URL http://ceur-ws.org/Vol-2517/T3-26.pdf.

Rajalakshmi, R., Reddy, P., Khare, S., Ganganwar, V., 2022. Sentimental analysis of code-mixed Hindi language. In: Congress on Intelligent Systems. Springer, pp. 739–751.

Rajalakshmi, R., Reddy, Y., Kumar, L., 2021. DLRG@DravidianLangTech-EACL2021: Transformer based approachfor offensive language identification on code-mixed Tamil. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Kyiv, pp. 357–362, URL https://aclanthology.org/2021.dravidianlangtech-1.53.

Rajalakshmi, R., Tiwari, H., Patel, J., Kumar, A., Karthik, R., 2020. Design of kids-specific URL classifier using recurrent convolutional neural network. Procedia Comput. Sci. 167, 2124–2131.

Reddy, B.Y., Rajalakshmi, R., 2020. DLRG@ HASOC 2020: A hybrid approach for hate and offensive content identification in multilingual tweets. In: FIRE (Working Notes). pp. 304–310.

Risch, J., Stoll, A., Wilms, L., Wiegand, M., 2021. Overview of the GermEval 2021 shared task on the identification of toxic, engaging, and fact-claiming comments. In: Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments. Association for Computational Linguistics, Duesseldorf, Germany, pp. 1–12, URL https://aclanthology.org/2021.germeval-1.1.

Roy, P.K., Bhawal, S., Subalalitha, C.N., 2022. Hate speech and offensive language detection in Dravidian languages using deep ensemble framework. Comput. Speech Lang. 75, 101386. http://dx.doi.org/10.1016/j.csl.2022.101386, URL https://www.sciencedirect.com/science/article/pii/S0885230822000250.

Sai, S., Sharma, Y., 2021. Towards offensive language identification for Dravidian languages. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Kyiv, pp. 18–27, URL https://aclanthology.org/2021.dravidianlangtech-1.3.

Saini, J.R., Rakholia, R.M., 2016. On continent and script-wise divisions-based statistical measures for stop-words lists of international languages. Procedia Comput. Sci. 89, 313–319.

Sakuntharaj, R., Mahesan, S., 2021. Missing word detection and correction based on context of Tamil sentences using N-grams. In: 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS). pp. 42–47. http://dx.doi.org/10.1109/ICIAfS52090.2021.9606025.

Sharen, H., Rajalakshmi, R., 2022. DLRG@LT-EDI-ACL2022:Detecting signs of depression from social media using XGBoost method. In: Proceedings of the Second Workshop on Language Technology for Equality, Diversity and Inclusion. Association for Computational Linguistics, Dublin, Ireland, pp. 346–349. http://dx.doi.org/10.18653/v1/2022.ltedi-1.53, URL https://aclanthology.org/2022.ltedi-1.53.

Sivakumar, S., Rajalakshmi, R., 2022. Context-aware sentiment analysis with attention-enhanced features from bidirectional transformers. Soc. Netw. Anal. Min. 12 (1), 1–23.

Sivalingam, D., Thavareesan, S., 2021. OffTamil@DravideanLangTech-EASL2021: Offensive language identification in Tamil text. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. pp. 346–351.

Sivalingam, D., Thavareesan, S., 2021a. OffTamil@DravideanLangTech-EASL2021: Offensive language identification in Tamil text. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Kyiv, pp. 346–351, URL https://aclanthology.org/2021.dravidianlangtech-1.51.

Soubraylu, S., Rajalakshmi, R., 2021. Hybrid convolutional bidirectional recurrent neural network based sentiment analysis on movie reviews. Comput. Intell. 37 (2), 735–757.

Thangarasu, M., Manavalan, R., 2013. Stemmers for Tamil language: performance analysis. arXiv preprint arXiv:1310.0754.

Thavareesan, S., Mahesan, S., 2021b. Sentiment analysis in Tamil texts using k-means and k-nearest neighbour. In: 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS). IEEE, pp. 48–53.

Yasaswini, K., Puranik, K., Hande, A., Priyadharshini, R., Thavareesan, S., Chakravarthi, B.R., 2021. IIITT@DravidianLangTech-EACL2021: Transfer learning for offensive language detection in Dravidian languages. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics, Kyiv, pp. 187–194, URL https://aclanthology.org/2021.dravidianlangtech-1.25.