Computer Speech and Language 76 (2022) 101407

Contents lists available at ScienceDirect

Computer Speech & Language

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

An analysis of machine learning models for sentiment analysis of Tamil code-mixed data

Kogilavani Shanmugavadivel [a,*], Sai Haritha Sampath [a], Pramod Nandhakumar [a], Prasath Mahalingam [a], Malliga Subramanian [a], Prasanna Kumar Kumaresan [b], Ruba Priyadharshini [c]

[a] *Department of Computer Science and Engineering, Kongu Engineering College, Tamil Nadu, India*
[b] *Indian Institute of Information Technology and Management, Kerala, India*
[c] *ULTRA Arts and Science College, Tamil Nadu, India*

ARTICLE INFO

Keywords:
Code-mixing
Natural Language Processing
Sequence classification

ABSTRACT

Nowadays, more and more people are sharing and expressing their feelings through social media platforms such as Twitter, Facebook and YouTube. Sentiment analysis is a process that explores, identifies and categorizes content. People that belong to multilingual communities tend to communicate through multiple regional languages. This type of text is represented using different languages and is known as code-mixed data. The proposed system utilizes a code-mixed data set of Tamil–English languages from FIRE 2021. To handle the class imbalance problem, re-sampling is performed and the impact is analyzed. Pre-processing of input text data can play a vital role in code-mixed data classification by removing unnecessary content. This research work aims to explore the impact of pre-processing on Tamil code-mixed data by employing various pre-processing steps such as emojis removal; repeated characters removal; and punctuations, symbols and number removal. The pre-processed text is applied to traditional machine learning, deep learning, transfer learning and hybrid deep learning models, and the accuracy of all these models before and after pre-processing is compared. Traditional machine learning models depend on various weighting schemes for the feature selection process. The main objective of this research work is to build hybrid deep learning models combining Convolutional Neural Network (CNN) with Long–Short Term Memory (LSTM) and Convolutional Neural Network (CNN) with Bi-Long–Short Term Memory (LSTM) in order to capture the local and global features implicitly from the code-mixed data for conducting sentiment analysis, and then classify the Tamil code-mixed data into positive, negative, mixed_feelings and unknown_state. The performance of hybrid deep learning models were evaluated by comparing them with state-of-art methods that include various traditional machine learning techniques such as random forest, multinomial Naive Bayes, logistic regression and linear Support Vector Classification (SVC); deep learning techniques such as LSTM, BiLSTM, BiGRU (Bidirectional Gated Recurrent Unit) and CNN; and a transfer learning method, IndicBERT. This research work also summarizes the precision, recall, F1-score, accuracy, macro-average, weighted-average and confusion matrix for all mentioned models. The result indicates that among all the different models employed,

* Corresponding author.
E-mail addresses: kogilavani.sv@gmail.com (K. Shanmugavadivel), Saiharitha69@gmail.com (S.H. Sampath), pramodarjunan47@gmail.com (P. Nandhakumar), prasathmk1401@gmail.com (P. Mahalingam), mallinishanth72@gmail.com (M. Subramanian), prasanna.mi20@iiitmk.ac.in (P.K. Kumaresan), rubapriyadharshini.a@gmail.com (R. Priyadharshini).

https://doi.org/10.1016/j.csl.2022.101407
Received 27 January 2022; Received in revised form 19 May 2022; Accepted 19 May 2022
Available online 28 May 2022
0885-2308/© 2022 Elsevier Ltd. All rights reserved.

the hybrid deep learning model, especially the CNN+BiLSTM model performs better, with an accuracy of **0.66** with preprocessed Tamil code-mixed data.

1. Introduction

Social networks are popular platforms that play an indispensable role in our life, and that can provide a convenient service for the users for transmitting and receiving messages. Language is a powerful tool to interact with or communicate thoughts, ideas, concepts or feelings (Sampath et al., 2022; Ravikiran et al., 2022; Chakravarthi et al., 2022a; Bharathi et al., 2022; Priyadharshini et al., 2022). With the increased use of Facebook, Twitter and YouTube, code-mixed data is normally available in abundance, but processing and analyzing such code-mixed data still seems like a tedious task (Chakravarthi, 2020; Chakravarthi and Muralidaran, 2021). With the Tamil code-mixed data in this study, we aim to perform sentiment analysis and classify them into four classes: positive, negative, mixed-feelings and unknown-state, using state-of-the-art techniques that include traditional machine learning, deep learning, transfer learning and hybrid deep learning models. In general, machine learning models are able to perform various tasks without being explicitly programmed, but selecting features from a data set requires an explicit weighting scheme. Deep learning methods are sophisticated approaches to machine learning that employ a neural network that learns the features dynamically.

Mining the text by user helps to understand the meaning that the sender wants to convey. For example, residents living in multilingual regions tend to use English-based speech types and insert English into their primary language when communicating through text on digital mediums, for example, tweeting, posting, commenting, etc. (Sakuntharaj and Mahesan, 2021, 2017, 2016; Thavareesan and Mahesan, 2019, 2020a,b, 2021). Due to the advantage of ease of communication, code-mixed language has been widely used and popularized in multilingual regions (Chakravarthi et al., 2020d). People can express their feelings through social medias posts in various ways. The polarity identification of such social media posts is required in order to classify them into different categories of feelings (Chakravarthi et al., 2021). A significant amount of research work has been already carried out with respect to the English language, using tweets in English. However, sentiment identification in the context of regional languages, particularly Tamil code-mixed, is yet uncharted territory. Thus, performing sentiment analysis on tweets in Tamil code-mixed form will be quite necessary.

Data pre-processing is one of the significant steps in social media data analysis. The main aim of pre-processing data is to prepare the data set for accurate analysis by using different computational techniques. This research work analyzes the impact of pre-processing by applying the raw data set and pre-processed data set to multiple classifiers in order to categorize the sentiment of the Tamil code-mixed data into positive, negative, mixed_feelings and unknown_state. In general, traditional machine learning models provide better results when applied to finance, marketing and medical data analysis with explicit feature extraction. For social media data analysis, however, sentiment polarity-based deep learning models are best suited. Deep learning models have the ability to utilize intermediate hidden layers for the extraction of the best features. Earlier findings also suggest that every deep learning model has its own advantages and disadvantages. For example, LSTM provides better results but its processing time is high when compared to CNN. Similarly, CNN considers less parameters than LSTM. Thus, there is a need to combine two or more models to get the benefits of both the models and minimize the shortcomings of each. Transfer learning has emerged as a new machine learning technique in which a model developed for one task applies the learned knowledge for another.

The objective of this research work is to investigate the impact of applying pre-processing techniques to Tamil code-mixed data, and apply the pre-processed data to state-of-the-art classifiers and hybrid deep learning models in order to detect the sentiment of the Tamil code-mixed data and classify them into positive, negative, mixed_feelings and unknown_state. The data set utilized in this study is taken from the FIRE 2021 shared task and contains YouTube comments. In order to classify these comments, they need to be pre-processed. Various pre-processing techniques were applied while preprocessing the data. Finally, the pre-processed messages were fed into various machine learning and deep learning algorithms to generate respective models. When the test data is applied to these models, they have to classify the messages into positive, negative, mixed_feelings and unknown_state. For example, consider the following code-mixed message:

“His acting was not good ava Eno Konjam nala pannier kalam...!!”

“His acting was not good; he could do a lot better..!!”

Several words in the above sentence belong to the English vocabulary, while others are from Tamil, yet they are all written in the Roman script that is used in the English language. Natural Language Processing (NLP) tasks such as word-level language recognition, word or phrase tagging, dependency interpretation, translation software and semantic processing are all hampered by code-mixing. In this study, the proposed methodology called An Analysis of Machine Learning Models for Sentiment Analysis of Tamil Code-Mixed Data is used to detect and classify the messages into their corresponding sentiment polarity. The sample code-mixed messages and their corresponding class labels are represented in Table 1.

Earlier approaches as mentioned in Myers-Scotton (1993) used syntactic rules and lexicons to extract features, followed by traditional machine learning classifiers for sentiment polarity on code-mixed text. The procedure of rule extraction and defining lexicons is a time-consuming and laborious process, and it is domain-dependent.

Table 1
Messages and class labels.

Messages	Class labels
Str and Arun vj Vera level (Str and Arjun vj are awesome)	Positive
Ivlo mokayana traileraa (Worst Trailer ever)	Negative
Dislike pandravanga elam rajini sir ah parthu poramai pidichavanga. (Those who are disliking are envious on Rajani Sir	Mixed_feelings)
Wow Surya,kv Anand, Harris jayaraj best combo	Unknown_state

In order to avoid this, the proposed research work utilizes various pre-processing techniques to prepare the data for sentiment analysis. These pre-processed data are applied to traditional machine learning techniques such as random forest, linear support vector classifier and multinomial Naive Bayes. Different deep learning techniques such as LSTM, BiLSTM (Bidirectional Long–Short Term Memory), CNN and BiGRU (Bidirectional Gated Recurrent Unit) are also used to classify the Tamil code-mixed messages. Then, a transfer learning method named IndicBERT, which is a multilingual ALBERT model trained on 12 Indian languages including Tamil, is utilized to detect the polarity level of the Tamil code-mixed data. The novelty of this research work is that it investigates the impact of pre-processing and develops hybrid deep learning models such as CNN+LSTM (Convolutional Neural Network with Long–Short Term Memory),LSTM+CNN (Long–Short Term Memory with Convolutional Neural Network), CNN+BiLSTM (Convolutional Neural Network with Bidirectional Long–Short Term Memory) and BiLSTM+CNN (Bidirectional Long–Short Term Memory with Convolutional Neural Network) techniques in order to improve the accuracy of the models. Finally, the performance of all these models is evaluated based on precision, recall, F1-score, accuracy, macro-average, weighted-average and confusion matrix.

1.1. Research questions

In this research work, we address the following research questions:

RQ1 *Does pre-processing affect machine learning approaches for Tamil code-mixed data?*
The findings from this research work indicate that three pre-processing techniques: (1) emojis removal, (2) repeated characters removal and (3) punctuations, symbols and number removal, employed in this study improved the accuracy of all machine learning approaches for sentiment analysis in Tamil code-mixed data.

RQ2 *Which model works well to perform sentiment analysis on Tamil code-mixed data?*
We implemented various state-of-the-art approaches such as traditional machine learning, deep learning, transfer learning and hybrid deep learning models on a Tamil code-mixed data set. We found that the hybrid CNN+BiLSTM learning model works well to perform sentiment analysis by capturing local and global features of the Tamil code-mixed data.

RQ3 *Does class re-sampling improve the accuracy of the machine learning approaches for Tamil code-mixed Data?*
In order to deal with the class imbalance problem, this research work performed class re-sampling to make all the classes in the Tamil code-mixed data of the same size. However, the results show that the accuracy did not improve after employing the class re-sampling technique. Therefore, we proceed with the data set without re-sampling them for all models.

1.2. Contribution

The contributions of this research work can be summarized as follows:

1. We investigated the effect of significant re-sampling and data pre-possessing techniques to prepare the data for sentiment analysis.
2. We adopted the Term Frequency-Inverse Document Frequency (TF–IDF) weighting scheme to extract features from the Tamil code-mixed data set for traditional machine learning techniques.
3. We implemented state-of-the art machine learning, deep learning, transfer learning and hybrid deep learning models for sentiment analysis of the Tamil code-mixed data.
4. We performed sentiment analysis on Tamil code-mixed data and classified them into four classes, namely positive, negative, mixed_feelings and unknown_state.
5. We presented the results of all state-of-the-art approaches and hybrid deep learning models using precision, recall, F1-score, accuracy, macro-average, weighted-average and confusion matrix.

The contents of this paper are organized as follows: Section 2 discusses the literature review of recent research on Tamil code-mixed data. Section 3 describes the class imbalance problem, data pre-processing techniques and feature extraction. Section 4 discusses the various state-of-the-art classifiers such as traditional machine learning, deep learning, transfer learning and hybrid deep learning models. Section 5 discuss the results obtained by applying various classifiers to the Tamil code-mixed data. Section 6 provides the conclusion as well as the future scope.

2. Literature survey

Chakravarthi et al. (2020d) categorized text based on its polarity using BiLSTM and Recurrent Neural Networks (RNNs) with sub-word level representation. Chakravarthi et al. (2020b) presented three sentiment assessment models, BERT and logistic regression classifier, DistilBERT, and rapid Text-mod, for code-mixed TA–EN (Tamil–English) and MA–EN (Malayalam–English) data sets. For both data sets, TA–EN and MA–EN, the rapid text approach outperformed other approaches. On the Dravidian code-mixed data set, Chakravarthi et al. (2020d) offers a transformer-based approach for meta embedding sentiment detection with a rich representation of the data. Barman et al. (2014) developed an ensemble model that makes use of intricate sequential pattern information. More specifically, their model incorporates self-attention based on BiLSTM and sub-word representation learning to classify Video remarks in blended Malayalam language as offensive or not-offensive, as well as another label categorization at the message level task that classifies a tweet or YouTube comment in Tanglish.

Jose et al. (2020) developed an auto-regressive XLNet model to perform text analysis using blended TAMIL–ENGLISH and MALAYALAM–ENGLISH data sets. Priyadharshini et al. (2020) utilized network shared parameters to create a shared emotion space and connect statements from code-mixed and conventional languages. Furthermore, they presented a simple cluster-based pre-processing method for capturing variants in code-mixed transliterated words. Onan and Toçoğlu (2021) and Chakravarthi et al. (2020c) focused on the sentiment analysis of tweets using Hindi and English words as well as other symbols. They produced various representations of tweets at the word, character, and sub-word level that are given as input to various models such as CNN, LSTM and BiLSTM, and identified that the performance of the BiLSTM model is superior. Chakravarthi et al. (2020a) utilized a traditional machine-learning approach for classification that included Support Vector Machine (SVM), perceptron and logistic classifier. Myers-Scotton (1993) developed a survey article to provide an overview of the most trending developments in the sentiment categorization of code-switching text. They also emphasized NLP methods, machine learning, lexicon and hybrid methods to process the code-switching data. The implications and obstacles that researchers faced in this subject are also discussed in this section. Devlin et al. (2019) presented a method for determining the emotions of tweets written in Hindi, Bengali and Tamil. They developed 39 models with optimal parameter values utilizing three distinct neural network layers. Sanh et al. (2019) proposed Distil BERT, a method for pre-training a smaller general-purpose language representation model that can eventually be fine-tuned to perform well within a range of tasks, identical to its wider counterparts. The authors utilize knowledge distillation during the pre-training phase to demonstrate that a BERT model may be reduced in size by 40 percent while reserving 97 percent of its language comprehension skills and being 60 percent quicker. Bali et al. (2014) developed a quality Hindi–English code-switched data-set including 15,744 YouTube remarks. In their study, they spoke about how they built the collection and included the results of the trend analysis using the sample as a reference.

Onan (2021) and Kudo (2018) utilized sentiment polarity of blended text as a contrastive learning technique to classify sentences according to their emotional connotations such as positive, negative or neutral. They used Siamese network shared parameters to map phrases from code-switching and ordinal languages to a single emotion. In addition, they presented a simple clustering-based pre-processing approach for capturing variants in code-mixed transliterated words. Pennington et al. (2014) learned the emotional value of important phonetic symbols. As demonstrated by the phonetic transcription, feature maps generated by our model look to perform admirably in extremely crowded texts that are crammed with incorrect words. They also believe that using this approach in the sub-word design will increase performance. Cambria et al. (2013) utilized a set of 28 emotions based on linguistic representations of the text and derived using an integrative perspective of emotion. In the first phase, inductive coding was utilized to extract a set of emotion categories from 5,553 tweets. Card sorting and emotion word rating were used to further combine and refine the categories. In Phase 2, they used crowd sourcing to assess the representatives of the emotion categories on a bigger data set of 10,000 tweets. Phase 2 produced no new emotion categories, demonstrating that the 28 emotion categories mentioned earlier are adequate for capturing the breadth of emotional experiences conveyed in tweets.

Farrugia (2004) performed a comprehensive literature search on many state-of-the-art sentiment analyses of code-mixed data. This study of the literature indicated that the majority of the research efforts with respect to sentiment polarity for code-mixed data have focused on pre-processing, language identification, lexicon development and sentiment classification tasks. Bojanowski et al. (2017) worked on sentiment analysis in agglutinative Dravidian languages. This method generates a more concentrated and accurate sentiment summary of a given Telugu text, which is beneficial in detecting emotions. In addition, this method is not bound by any domain. Small changes to pre-processing, on the other hand, would be sufficient to apply this algorithmic formulation in various domains or languages.

Abdelwahab et al. (2015) evaluated the influence of retraining set size on SVM and Naive Bayes. When SVM and Naive Bayes was employed in Twitter sentiment segmentation, the researcher looked at the impact of changing the training response rate on the learning slopes of both SVM and Naive Bayes. The influence of the training set on various ensemble fusion types was also investigated by the author. Ensemble 1 performed better than Ensemble 2, which included SVM and Naive Bayes classifiers. Parveen and Pandey (2016) presented an Hadoop Distributed File System (HDFS) model and a MapReduce technique. The data set is first pre-processed, and then a supervised learning method called Naive Bayes is used to implement it. A trained SentiWordNet vocabulary is required to implement the Naive Bayes method. The method is implemented using two approaches: Method 1 is centered on the modeling phase, which converts the information of the Word embedding vocabulary from a disk to a hashing map for faster symmetric key orientation retrieval of words; and Method 2 is a reduction phase that gathers each tweet's overall polarity and divides it into 5 categories: extreme positive, negative, mixed_feelings, unknown_State and not_Tamil.

Sahni et al. (2017) used the subjectivity of tweets to classify data sets. An objective phrase usually does not communicate any emotion. As a result, only pure subjective phrases are examined to decrease the amount of training data sets. One of the pre-processing approach, such as Text Blob, are used to filter subjective phrases before implementing classification algorithms. N-grams and Part Of Speech (POS) are two of the feature extraction approach examined. Trupthi et al. (2017) illustrated the importance of pre-processing the training set. NLTK is used to eliminate words with POS tags that are not relevant for classifier construction. In addition, Hadoop is used to extract data, while Map Reduce is utilized to quickly extract many words together with their positive and negative probability. Map Reducer produces a large number of words, each with a good positive and negative score.

Tsapatsoulis and Djouvas (2017) wanted to see if tokens, which people manually specify in Twitter annotations, can be utilized to create a word index that can be used to train successful tweet categorization algorithms. To back up his claim, the author used a machine learning framework and three distinct classifiers to compare a human-created index of words with various automatically derived feature sets for tweet categorization. The three techniques identified by the author include a lexicon-based method, a machine learning approach and a deep learning approach. Among all other feature extraction approaches, the author found that tokens explicitly recognized by humans performed the best. However, the author did not take into account a mix of feature sets.

Gupta and Singal (2017) devised a unique method for filtering tweets based on their location and comparing the results in the outcome of accuracy and efficiency, to categorize a Twitter data set. Pre-processing of the data is accomplished using NLTK and Scikit-learn on the particular data set, which is then exposed to computational executions such as Naive Bayes, logistic regression and support vector machines. In terms of precision, recall, accuracy and F1-score, this execution is compared to see which method performs better for a particular data set. This helps to comprehend the feelings independently. The effect of pre-processing on data is also demonstrated by the author. Kamps et al. (2004) have analyzed the data using a lexical database. A lexical database is a collection of lexemes. Word-Net-based distance or similarity estimates are nearly focused entirely on taxonomic correlations. The semantic polarity of adjectives is calculated using the word distance metric.

Barbosa and Feng (2010) analyzed the sentiment of a Twitter collection using machine and deep learning methods. They collected Twitter test data and used syntactic elements such as symbols, retweets, emoticons, tags, links, punctuation and exclamation marks to analyze tweets. They used two types of classifiers, polarity classifier and subjectivity classifier. Semicolons are used in conjunction with structures to detect the polarity of words. Shobana et al. (2018) have analyzed that Twitter sentiment analysis falls under the category of text and opinion mining. This study area has progressed over the previous decade, with models attaining efficiencies of up to 85% to 90%. However, it still lacks data variety. In addition, it also has several difficulties with respect to application due to the slang and abbreviated versions of words utilized in the data. Many analyzers perform poorly when the number of classes is raised. Furthermore, the model's accuracy for topics other than the one under discussion has yet to be proven.

In their study, Patel et al. (2017) explored strategies for pre-processing data and retrieving information from tweets via Twitter. They also looked at a supervised learning approach called Support Vector Machine for document classification, but it may be used to identify the polarities of textual tweets. This finding suggests that SVM can detect some text characteristics, such as a strong feature set, a small number of irrelevant features and a sparse instance vector. Furthermore, precision and recall can be used to assess SVM performance. SVM avoids the necessity for feature extraction by being able to generalize a strong feature set.

Ganie and Dadvandipour (2021) performed a sentiment analysis of tweets. According to the emotional analysis classifiers, the majority of the tweets were detrimental in context. With over 82% accuracy and reduced false-positive and false-negative ratios, Naive Bayes is the most successful classifier. In the data set, two layers of neural networks were employed. Furthermore, increasing the number of epochs enhanced the accuracy to a certain extent, resulting in a 70.58 percent accuracy. The accuracy of both random forest and SVM classifiers was 73.52%, with the same false-positive and false-negative ratios. Syrien et al. (29) evaluated tweets using R programming and machine learning methods such as Naive Bayes and SVM. For sentiment analysis on the data, the SVM model had greater precision, whereas Naive Bayes had higher accuracy.

Hasan et al. (2018) developed a hybrid technique that combines a sentiment analyzer with machine learning. Opinion mining and sentiment analysis have seen significant growth in terms of exploring the views or text available on various social media platforms using machine learning methods. Furthermore, utilizing supervised methods such as Naive Bayes and support vector machines, this research assessed text analytical methodologies in the study of strategic views. The polarity of English tweets may be detected using a Naive Bayes classifier, as described in this article. Gamallo and Garcia (2014) detected the polarity of English tweets using a Naive Bayes classifier. To distinguish between tweets with and without polarity, the system employs a simple rule that looks for polarity terms in the tweets/texts being analyzed. The classification process includes factors such as lemmas, multi-words, polarity lexicons and valence shifters.

Kanakaraj and Guddeti (2015) revealed that experiments combining a linguistic feature vector with only a classification technique outperform a traditional backpack strategy with a single machine-learned classification by 3%–5%. Gautam and Yadav (2014) used different classifiers such as Naive Bayes, gradient boosting and SVM, as well as the conceptual direction word vector for sentiment analysis. According to Hegde et al. (2018), support vector machines provided improved accuracy and Naive Bayes worked admirably but fell short of expectations; however, their long execution time violated the aim of a fast classifier. Ramadhani and Goo (2017) developed RNNs for the emotive analysis of Twitter data sets. Twitter is a social networking service that collects a large quantity of data from users. Because of this large amount of data, it offers the potential for text mining research and can be exposed to sentiment analysis. To manage massive amount of unstructured data, machine learning techniques are utilized. In terms of neural networks, deep learning is a computational intelligence category that uses a deep-feed forward network of neurons with numerous hidden layers, with a success rate of 75%.

Alessia et al. (2015) computed two approaches for handling data in sentiment analysis. Method 1 categorizes data based on features and Method 2 categorizes data based on various techniques used for sentiment analysis. In addition, the study discusses many application domains of sentiment analysis based on the tweets. Zhang et al. (2009) developed Mlnb, a technique that extends standard Naive Bayes classifications to deal with multi-label cases to solve a learning challenge. Mlnb include built-in feature selection strategies to improve its performance. To begin feature selection, first, unnecessary and redundant traits are eliminated

Table 2
Code-mixed data set.

Class	Train	Dev	Test
Mixed_feelings	4020	438	470
Negative	4271	480	477
Positive	20070	2257	2546
Unknown_state	5828	611	665
Total	34189	3786	4158

using covariance analysis-based feature extraction approaches. Then, by employing attribute selection methodologies based on genetic algorithms, the more relevant subset of traits for prediction is identified. In terms of reliability, experiments on synthetic and genuine data show that Mlnb is comparable to other well-known cross machine learning algorithms.

Bhuvan et al. (2015) suggested a paradigm that is designed to be adaptable, allowing it to be used in any domain by simply rewriting the semantics for that topic. Despite its reduced reliability, their model uses a cognitive method of capturing different perceptions by learning from a mix of favorable and unfavorable tweets. They observed that regression analysis on Spark with Stochastic Gradient Descent (SGD) performed better, with an accuracy of 64.12, and that it is exceptionally salable, after analyzing the results of several testing. Le and Nguyen (2015) used machine learning approaches to create a model that analyzes Twitter sentiment using an effective feature set that improves accuracy, such as bigram, unigram and object-oriented features. The accuracy of two algorithms, i.e., the Naive Bayes followed by SVM, is examined by calculating precision, recall and F1-score, and both approaches exhibit the same accuracy. Mullen and Collier (2004) addressed the learning problem with support vector machine to bring together the diverse source of pertinent information. Experiments on Epinions.com movie review data show that hybrid SVMs with blend unigram-style outperforms than feature-based SVMs.

3. Methodology

The main objective of this research work is to perform sentiment analysis on Tamil code-mixed data by applying various state-of-the-art learning and hybrid deep learning techniques in order to classify them into four categories: positive, negative, mixed-feelings and unknown state. In the data exploration phase, various pre-processing steps are carried out to clean the data set. For feature selection, the TF_IDF method is used. For model building, traditional machine learning models such as random forest, linear support vector classifier, multinomial Naive Bayes and logistic regression are utilized. To further improve the performance, deep learning models such as LSTM, BiLSTM, BiGRU and CNN have been utilized. Then, a transfer learning method called IndicBERT is implemented to check the reusability of an existing model. Finally, hybrid deep learning models are constructed by combining CNN+LSTM, LSTM+CNN, CNN+BiLSTM and BiLSTM+CNN. All the developed models are evaluated using measures such as precision, recall, F1-score and accuracy.

3.1. Data set description

The data set contains code-mixed utterances in Tamil–English language. In all, there are three files in the data set, namely training, validation and test data files. The training and validation data set comprises of three columns: an id column, a text column with comment/post information, and a third column or feature represents sentiment polarity information at the message level. All of the language sentiment was divided into four categories, positive, negative, mixed-feelings and unknown_state. The data set's details include the total number of comments from each of the four classes (Chakravarthi et al., 2022b). The training, development and test data set description is specified in Table 2.

The positive label is made up of a positive message from the supplied remarks. The negative label contains a negative message from the given comments. The unknown_state indicates that the comments are neither positive nor negative. Lastly, mixed-feelings indicates that the label contains both positive and negative comments. The sentences are brief, with few well-defined grammatical structures and several spelling errors.

3.2. Class re-sampling

In both binary and multi-class classification problems, imbalanced classes prove a significant barrier in the development of an effective classifier. Because the classes are so unbalanced, a majority classifier would provide solid accurate results, identifying all occurrences with the class that has the most representation. Failure of all items in the other classes, on the other hand, would result in low accuracy, recall and F1-score, our primary evaluation metrics. Different alternate strategies to prevent this difficulty are provided by class re-sampling approaches. The objective is not to have a perfectly balanced distribution, but rather to have a distribution that classical classifiers can handle well. Thus, we proceeded with widely used methodologies (synthetic minority oversampling (SMOTE)) for our research.

A contrasting process is carried out in the oversampling approach. The minority class's size is expanded to match that of the majority class by copying its instances numerous times to achieve the necessary size. This technique offers the advantage of keeping all of the data set's relevant information.

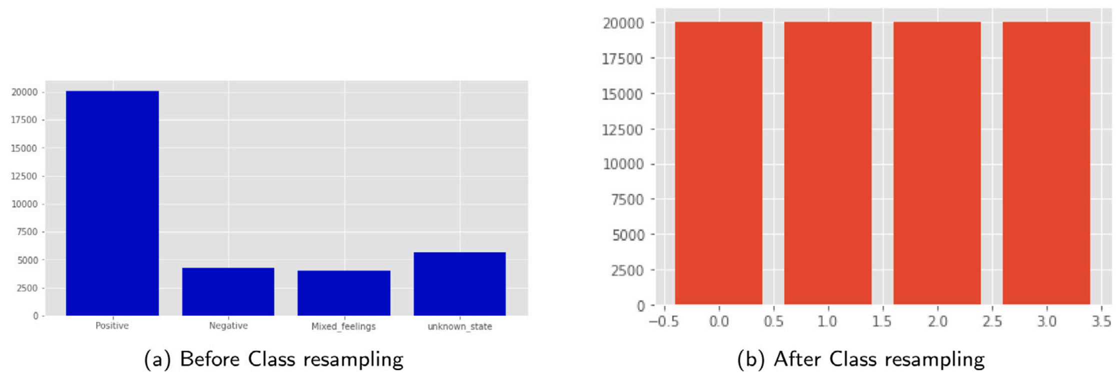

Fig. 1. Random Oversampling.

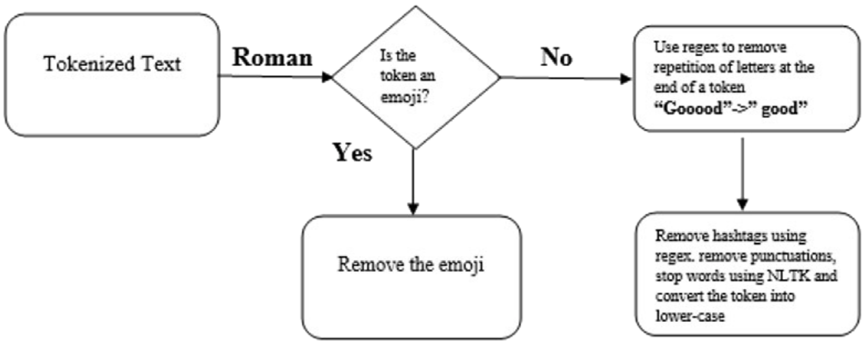
Tokenized Text
Roman
Is the token an emoji?
No
Use regex to remove repetition of letters at the end of a token "Gooood"->" good"
Yes
Remove the emoji
Remove hashtags using regex. remove punctuations, stop words using NLTK and convert the token into lower-case

Fig. 2. Pre-Processing of Code-Mixed Data.

Another common oversampling approach is **SMOTE**, in which new points are synthesized between old ones. The technique is usually pictured as a hypercube between each point in the minority class and its k closest neighbor. Inside the hypercube, new artificial points are produced. This technique has the obvious benefit of retaining as well as expanding the quantity of valuable information.

In Fig. 1 Label (a), the training data sets are classified into positive with 20,068 class labels, unknown_State with 5,628 class labels, negative with 4,2710 class labels and mixed_feelings with 4,020 class labels. After re-sampling, the class labels are equally distributed as 20,068 for all categories, as illustrated in Fig. 1 Label (b).

In the proposed system, initially, we used class re-sampling for machine learning methods. The accuracy of random forest before class re-sampling was 0.63 and after applying SMOTE is 0.58. Similarly, other machine learning models also produced lower accuracy than models without class re-sampling. The reason for this is that SMOTE does not consider neighbor data. This problem may be solved by combining SMOTE with techniques such as a nearest neighbor algorithm. Since the proposed work concentrates on the analysis of various machine learning, deep learning and transfer learning models, we proceed in this research work without employing class re-sampling for all traditional machine learning, deep learning, transfer learning and hybrid deep learning models.

3.3. Data pre-processing

Code-mixed data is gathered from several sources, and consists of excessive noise in the form of punctuation, white spaces, numerals, special characters, additional spaces, emojis and Stop-words. The code-mixed data, in its raw format, makes the analysis process difficult. As a result, data pre-processing is required to convert raw data into a clean one. Data pre-processing is applied to prepare the data in an appropriate format so that better results can be obtained after applying these pre-processed data into various models.

In the proposed system, the following pre-processing techniques were applied for code-mixed data. A detailed diagram is sketched in Fig. 2.

1. Tokenization.
2. Removal of emojis.
3. Handling of the sequence of repeated characters.
4. Removal of punctuations, symbols and numbers

Table 3
Sample code-mixed message and their tokens.

Message	Token
Arjun sir kaga padam pakalam	’Arjun’,’sir’,’kaga’,’padam’,’pakalam’
Vijay Anna iku like pannunga	’Vijay’,’Anna’,’like’,’pannunga’

Table 4
Removal of emojis for sample code-mixed message.

Message	Token
Surya and Jothika best combo ever	’Surya and Jothika best combo ever’
Vijay Anna iku like pannunga	’Vijay Anna like pannunga’

Table 5
Sample code-mixed messages after handling sequence of repeated characters.

Message	Token
Vijay anna ur masssssssssss	‘Vijay anna ur mass‘
Semmaaaaaaa vera level thaliva	’Semma vera level thaliva’

Table 6
Sample code-mixed messages after removal of punctuations, symbols and numbers.

Message	Token
Semma Semma waiting.....	’Semma semma waiting’
Wow..!! Super bro	’Wow Super bro’

3.3.1. Tokenization

Tokenization is the process of attempting to break down a sentence into a list of token or words. This pre-processing step is a very crucial one as it provides a wealth of information about a particular text’s meaning. The sample tokens obtained from the given code-mixed messages are shown in Table 3.

3.3.2. Removal of emojis

Emojis are visual representations of an idea or emotion that can be used in a text message. There are two ways to deal with them. The first is to replace an emoji with a textual term, while the second option is to eliminate it altogether.

In this work, all emojis are removed because they do not add anything to the text data. For example, ‘:)’ is an emoticon that represents a happy face. Hence, these emojis are removed because they do not provide any useful information. The sample messages for removal of emojis in a sentence acquired from the given code-mixed data set are described in Table 4.

3.3.3. Handling of repeated characters

In social media writing, certain characters are frequently encountered and repeated several times. These non-conforming spellings are extremely difficult to deal with, as they cannot be matched to any dictionary. For example, lol (short for ‘‘laughing out loud’’), can be written as loool, looool or loooooool. To reduce all of these instances to lol, pre-processing is applied. Any character that appears more than twice in a row is replaced with two of the same character. The sample messages after handling the sequence of repeated characters acquired from the given code-mixed messages are shown in Table 5.

3.3.4. Removal of punctuations, symbols and numbers

Punctuations are symbols that are used in writing to separate sentences and their constituents in order to explain the meaning of the text. Most social media texts include a lot of punctuation, and their placement is sometimes random, defying grammatical conventions. We will eliminate them from our data set because they do not help convey the meaning of the text and are simply utilized for good readability. In this research work, an attempt is made to construct our content as noiseless as possible by eliminating various punctuation symbols such as period, question mark and exclamation mark that are frequently used in sequence. The sample message after removal of punctuation, symbols and numbers acquired from the given code-mixed messages are shown in Table 6.

In the proposed system, all the previously mentioned pre-processing techniques are applied to the code-mixed data, and the result shows that these pre-processing techniques improve the accuracy of the models. The accuracy of all the models before and after applying pre-processing are reported in Section 5.2 - Table 10.

3.4. Feature extraction

In the context of NLP activities, feature extraction entails extracting features and producing numerical representations of text input so that machine learning and deep learning algorithms can use it to make predictions. When we generate a vector from a text based on word frequency, the most common words/terms, which contain relatively little information about the content, begin to take precedence over the rarer words, which are more significant and intriguing. As a result, rather than evaluating terms in documents based on their basic frequency in one document, we rescale it to the frequency of words appearing in all documents. This will result in a penalty for words that are common not only in one document but also across all texts. The score in TF–IDF is determined by the frequency of the term in the original file, while that of IDF is based on the term's rarity across all texts. Textual information has been successfully encoded into real-valued vectors using TF–IDF vectors. Count vectorizers in the Sklearn (Pedregosa et al. 2011) is the library that is used to transform text input into a collection of tokens. It also allows for the inclusion of greater n-grams in the lexicon. This may be beneficial in the categorization process. For words (unigrams), bigrams and trigrams in the training corpus, we experimented with TF–IDF vectors. TF–IDF vectors for character n-grams with n values ranging from 2 to 6 were also tested. For the Tamil–English corpus, we utilized several TF–IDF vectors.

Term Frequency: It refers to the multitude of times a word appears in the source text and can be calculated as in Eq. (1) below:

$$TF = (Number\ of repetitions\ of\ a\ word\ in\ a\ document)/(number\ of\ words\ in\ a\ document) \tag{1}$$

Inverse Document Frequency: The overall number of documents divided by the number of documents that include that specific term equals the set of files that contain that particular phrase. To reduce the exponential growth during frequency calculation, the log of the entire term is computed. It is calculated as in Eq. (2) below:

$$IDF = log((Number\ of\ documents)/(Number\ of\ documents\ containing\ the\ word)) \tag{2}$$

$$TF--IDF = TermFrequency\ (TF) * InverseDocumentFrequency\ (IDF) \tag{3}$$

4. Algorithmic modeling techniques

In this algorithmic modeling, various state-of-the-art traditional machine learning, deep learning, transfer learning and hybrid deep learning models are built to carry out the classification process. The data sets are noisy by nature as they are gathered through social media. As a result, pre-processing is essential and was completed at the start of this research work. TF–IDF and Keras Tokenizer API are used to extract significant features from the data sets (training and validation). Thus, our proposed research work constructs the following four approaches listed below.

1. An approach based on Machine Learning
2. An approach based on Deep Learning
3. An approach based on Transfer Learning
4. An approach based on a Hybrid Model

4.1. Machine learning approaches

4.1.1. Random forest

A random forest is a computational model for dealing with classification and regression problems. It makes use of ensemble learning, which is a technique for solving difficult problems by combining several classifiers. Many decision trees make up a random forest algorithm. This algorithm produces a 'forest,' which is trained via bagging or bootstrap aggregation. Bagging is a meta-algorithm that enhances the accuracy of machine learning methods by grouping them together.

4.1.2. Linear support vector classifier

SVMs are a machine learning classification approach that uses a kernel function to transfer a space of data points that are not linearly separable onto a new space, with allowances for incorrect classification. Thus, SVM seeks to create a decision boundary with as much separation between the two groups as feasible.

4.1.3. Multinomial naive Bayes

Dictionary entries, URLs, email addresses and other characteristics may all be used by Naive Bayes classifiers. However, if we only utilize based on word characteristics and use the entire text (not a selection), Naive Bayes is extremely similar to representation learning. The Naive Bayes model may be thought of as a collection of discrete category unigram word embeddings, each of which represents a single unigram word embedding. Because the likely characteristics of the Naive Bayes model provide a probability to each word P(word|c).

4.1.4. Logistic regression

By fitting data to a logistic function, logistic regression can help determine the likelihood of an event occurring. In NLP, logistic regression is an important supervised machine learning algorithm for performing text-related classification tasks. It identifies the probability of the occurrence of an event by fitting data to a logistic function. Logistic regression is used to predict categorical dependent variables using a set of independent variables. Here, the logistic regression technique is applied to Tamil code-mixed data in order to classify them into four different classes.

4.2. Deep learning approaches

Deep learning is a powerful machine learning technique that analyzes several degrees of data models or characteristics to produce cutting-edge prediction results. Deep learning has gained popularity in sentiment analysis in recent years, owing to its success in a variety of other application domains.

4.2.1. Long–short-term memory

The LSTM may learn long-term dependencies. RNNs are made up of a succession of modules that repeat. This recurring module is usually of a simple form in RNNs. The LSTM repeating module, on the other hand, is far more advanced. Instead of a single fully linked layer, there are four discrete layers that interact in different ways. In addition, there are two states: concealed state and cell state.

LSTM is most typically employed with sequential data. It combines the "forget" and "input" barriers into a single number that may be implemented. It also hides the cell's state and modifies the memory cell. The resulting model is more user-friendly than typical LSTM models, and its popularity is expanding.

For implementing LSTM in this study, different parameters are chosen for the Tamil language. The proposed system extracted total unique wordsvocab_size and length of longest sentencemax_length from the data set. The system found 69,675 unique words and 124 as the maximum length of the longest sentence for the Tamil language. Every input comment is padded with a maximum length max_length of the language. For sequential_10 model, maximum_features is considered as 20000, embedding dimension as 128, sequential length as 40 and lstm_out as 196. The output of the embedding layer (40,128) is given as input to the Spatial_dropout1d layer, the output of Spatial_dropout1d layer with the shape (40,128) is given as input to LSTM layer and the output of the LSTM layer is given as input to the dense layer. The dense layer has 5 nodes with a sigmoid as an activation function. LSTM is trained with the categorical_crossentropy loss function and a Nadam optimizer.

4.2.2. Bidirectional gated recurrent unit

BiGRU is similar to LSTM, in that it can deal with long-term dependencies, but it does it in a different way. They have a smaller number of parameters, except for memory cells, GRU has gating units similar to LSTM. GRU's mission is by using an update and reset gate, the vanishing gradient problem can be solved. The update gate aids the user. Using models, past knowledge from previous phases should be passed on to the future. The reset gate of the design is used to determine the amount of prior data that will be wiped.

For the sequential model, the input for maximum_features is given as 20,000, for embedding dimension as 128, for sequential length as 40 and for lstm_out as 196. The output of the embedding layer with the shape (40,128) is given as input to the Spatial_dropout1d layer, the output of Spatial_dropout1d layer with the shape (40,128) is given as input to bidirectional layer and the output of the bidirectional layer 392 is given as input to the dense layer. The dense layer has 5 nodes with a sigmoid as an activation function. Bi-GRU is trained with the categorical_crossentropy loss function and a Nadam optimizer. The parameter trained for the embedding layer is 2,560,128, for the bidirectional layer is 383,376 and for dense layer is 1572.

4.2.3. Convolutional neural network

CNN is a deep learning technology based on multi-layer perceptron (MLP). Convolution, pooling and direct connection layers are the three major attributes of CNN. The convolution layer, which consists of a series of independent filters, is the most basic component of a CNN. The pooling layer aims to make the depiction of complicated layers easier. Finally, fully connected layers are vectors resulting from the many convolutions and pooling processes of a multi-layer perceptron and are used to execute a specific job, such as classification. A cross neural network is also referred to as a CNN. An input layer, a convolutional layer, a pooling layer, a fully linked surface and output units together make up the fundamental structure of a CNN.

In the Sequential_3 model, the input for maximum_features is given as 20,000, for embedding dimension as 64, for sequential length as 40 and for lstm_out as 196. Conv1d is provided with output filters as 128, pooling size as 2 and kernel size as 3, with relu as activation function. The output of the embedding layer with the shape (4064) is given as input to the conv1d layer, the output of conv1d layer with the shape (38,128) is given as input to global_max_pooling1d layer, the output of global_max_pooling1d layer 128 is given as input to the dropout layer and the output of dropout layer 128 is given as input to dense layer. The dense layer has 5 nodes with a sigmoid as an activation function. CNN is trained with the categorical_crossentropy loss function and a Nadam optimizer.

4.3. Transfer learning approach

Transfer learning is a machine learning research subject that focuses on storing and transferring information learned while addressing one issue to an interrelated problem. We can deal with these instances using transfer learning, which uses previously labeled data from a comparable task or topic.

4.3.1. IndicBERT

IndicBERT is a multilingual ALBERT model that covers 12 main Indian languages, including Tamil, and is trained on large-scale data sets. IndicBERT contains a lot fewer parameters than other public models such as mBERT and XLM-R, yet it performs quite well on a variety of tasks. The model is pre-trained on a corpus of around 9 billion tokens out of which 549M tokens are Tamil. The IndicBERT model can be evaluated for a set of tasks such as news article headline prediction, cross-lingual sentence retrieval, and sentiment analysis on product and movie reviews. This research work utilizes IndicBERT for social media sentiment analysis for the first time in recent research.

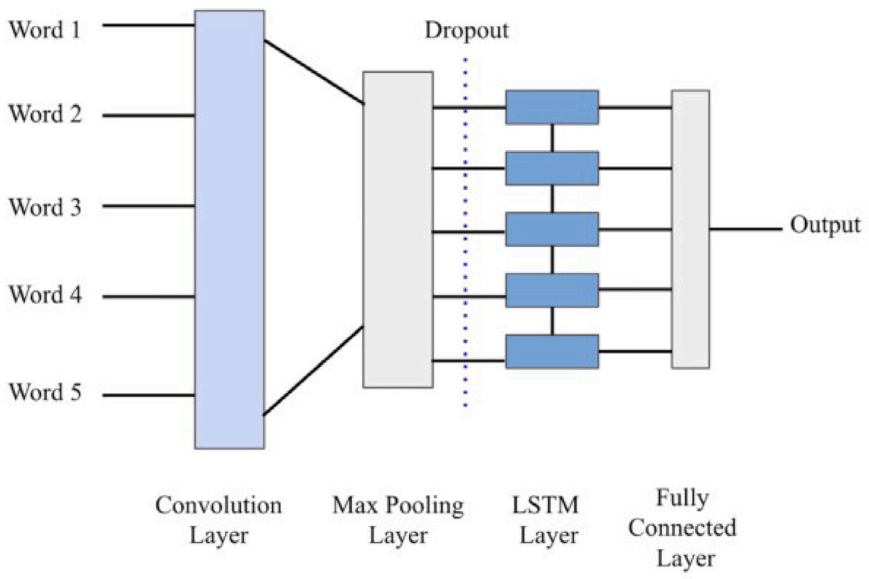

Fig. 3. CNN+LSTM model.

4.4. *Hybrid model*

4.4.1. *Convolutional Neural Network + Long–Short Term Memory*

In the processing of serial inputs, the RNN offers a wide range of applications. This is a common paradigm in deep learning frameworks. An RNN version is the long poor memory (LSTM) network. Several gates, such as an input gate, forget gate and output gate, as well as memory cells that have the same structure as the data point for storing extra data are used in LSTM.

Fig. 3 depicts the CNN+LSTM model. The input data layer consists of a fixed-dimension matrix that includes various vector word embeddings based on different degrees of sentiment. Tokens vary depending on the amount of sentiment classification. A Char-level token, for example, is a specific character. At the Ch5gram level, each token is a whole word if the phrase is five characters or less. Every word with more than five letters has a token that is broken down into five-gram characters. All tokens at the word level are based on the words in each tweet. In the input sequence, each token is kept as a repaired vector. The max-pooling layer's output vectors are fed into LSTM networks, which measure long-term relationships between showcase patterns. To obtain the final result, the LSTM outcome matrices are synthesized and an activation function is used.

The model's initial step is to encode each tweet as a sequence of vectors. Depending on the sentiment analysis level used, each vector symbolizes a token. Each stage has its own tokens, such as the Char-level, where each Twitter character is described by a specific vector with a size of 100. Each token in the phrase (each Twitter word) is contained in a 100-dimensional vector, which is the same as every token in the Ch5gram-level. Every layer is a w*v matrix, where w denotes the number of tokens included inside the tweets and v denotes the vector length. The effective length of a tweet is determined by the value of w. Any message containing more tokens than the maximum will be padded. The effective length of a tweet is determined by the value of w. Any message with more tokens than the limit will be stretched using <Pad> to make it fit the maximum tweet length.

Using Sequential_3 model, the input for maximum_features is given as 20,000, for embedding dimension as 128, for sequential length as 40 and for lstm_out as 196. Conv1d is provided with output filters as 128, pooling size as 2 and kernel size as 3, with relu as activation function. Conv1d is used for generating sequences of text. The output of the embedding layer (40,128) is given as input to the conv1d layer, the output of the conv1d layer (40,128) is given as input to the MaxPooling1d layer, the output of MaxPooling1d layer (20,128) is given as input to the last layer and the output of LSTM layer 196 is given as input to dense layer. The dense layer has 5 nodes with a sigmoid as an activation function. CNN LSTM is trained with the categorical_crossentropy loss function and a Nadam optimizer.

4.4.2. *Long-Short Term Memory + Convolutional Neural Network*

The LSTM+CNN model exceeds the CNN+LSTM model in terms of performance. In the LSTM+CNN model, an initial LSTM layer accepts word embeddings for each token in the phrases as inputs. The core idea would be that the output token will include data from both initial and previous tokens. Fig. 4 illustrates the LSTM+CNN structure.

In this method, the LSTM layer is in charge of producing new encodes for the incoming data. The LSTM layer, which is capable of extracting local features, sends its output to the CNN. Using the sequential model, the input for maximum_features are given as 20,000, for embedding dimension as 128, for sequential length as 40 and for lstm_out as 196. Conv1d is provided with output filters as 128, pooling size as 2 and kernel size as 3, with relu as activation function. Conv1d is used for generating sequences of text. The

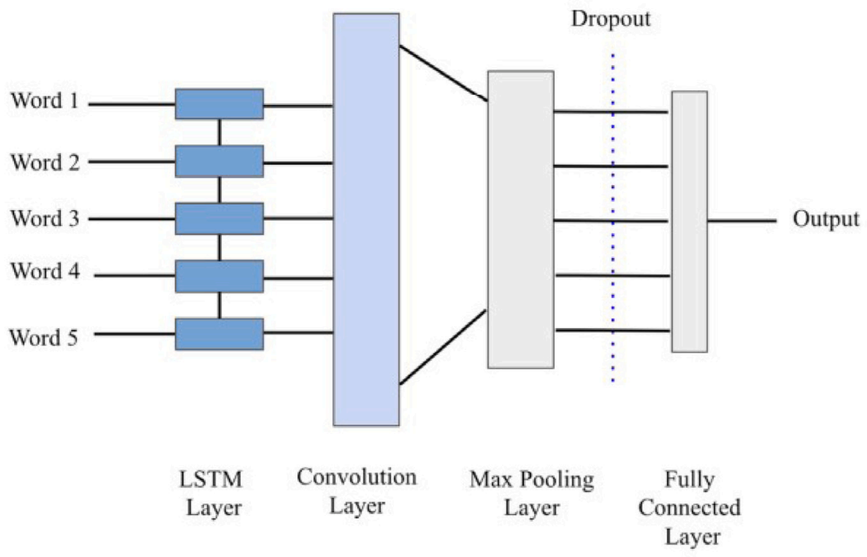

Fig. 4. LSTM+CNN model.

output of the embedding layer (40,128) is given as input to the LSTM layer, the output of the LSTM layer (40,196) is given as input to the conv1d layer, the output of the conv1d layer (40,128) is given as input to the flatten layer and the output of flattening layer 5,120 is given as input to dense layer. The dense layer has 5 nodes with a sigmoid as an activation function. LSTM+CNN is trained with the categorical_crossentropy loss function and a Nadam optimizer.

4.4.3. Convolutional Neural Network + Bidirectional Long–Short Term Memory

The CNN and BiLSTM models were integrated and tweaked, resulting in a new model that combines the benefits of both. Our objective was to address the shortcomings of both the CNN model, which is susceptible to local factors, and the RNN model, which is prone to gradient disappearance problems. We utilized this combination to evaluate how effectively CNN can adapt to BiLSTM, which is well-known for its ability to analyze opinions. The strength of this model is that it extracts the maximum amount of information from texts using CNN convolution layers. This output becomes a BiLSTM input, which allows the data in both routes to be retained in chronological order. Combining each model has its own model and strengths, combining CNN and RNN models necessitates a unique design:

1. CNN is well-known for extracting as many characteristics as possible from the text;
2. An approach based on deep learning;
3. Because LSTM/BiLSTM maintains the chronological sequence of words in a text, it may use the delete gate to discard unneeded words.

The goal of integrating the two models is to construct a new model that makes use of both CNN and BiLSTM's capabilities, capturing CNN's extracting features and using them as an LSTM input. Therefore, we have created a model that achieves this goal, where the vectors generated in the word embedding component serve as CNN input. Then, four filters with sizes of 2, 3, 4, and 5 are applied 100 times each. A layer of max-pooling is performed after each filter to update and minimize the amount of the data.

The outputs of all max-pooling layers are then concatenated to create the BiLSTM input, which employs the three gates of a BiLSTM layer to filter the data. The output of this step is the input for the completely connected layer, which connects each piece of input data with a piece of output data. Finally, in order to get the required result, we use the Softmax function as an activation function to assign classes to articles.

Thus, the model is composed of three parts:

1. **Pre-Processing:** Data cleaning and pre-processing are carried out in this step. The data is then prepared for convolution through distributed document representation using the embedded method. The generated vector is fed into the following stage as an input.
2. **Convolution:** In this step, convolution and max-pooling layers are utilized to extract high-level features. The output of this step becomes the input of the next stage.
3. **BiLSTM/Fully connected layers:** BiLSTM and fully connected layers are utilized in this stage to categorize document sentiment. This stage leads to the final classification of the document (as positive, negative, mixed sentiments, not in Tamil and uncertain state).

Using the sequential_3 model, the input for maximum_features is given as 20,000, for embedding dimension as 128, for sequential length as 40 and for lstm_out as 196. Conv1d is provided with output filters as 128, pooling size as 2 and kernel size as 3, with relu as activation function. The output of the embedding layer (40,128) is given as input to the conv1d layer, the output of the conv1d layer (40,128) is given as input to the MaxPooling1d layer, the output of MaxPooling1d layer (20,128) is given as input to the Bidirectional layer and the output of Bidirectional layer 392 is given as input to dense layer. The dense layer has 5 with sigmoid as activation function. CNN+Bi-LSTM is trained with the categorical_crossentropy loss function and a Nadam optimizer.

4.4.4. *Bidirectional Long–Short Term Memory + Convolutional Neural Network*

A BiLSTM is a CNN that combines the BiLSTM and CNN models. It learns both character-level and phrase characteristics in the original model for named entity recognition. Character-level characteristics are inferred using the CNN component. For each word, the model uses a convolution and a max-pooling layer to recover a new feature representation that includes per-character vectors such as character embeddings and (optionally) character type.

Using the sequential_5 model, the input for maximum_features is given as 20,000, for embedding dimension as 128, for sequential length as 40 and for lstm_out as 196. Conv1d is provided with output filters as 128, pooling size as 2 and kernel size as, 3 with relu as activation function. The output of the embedding layer (40,128) is given as input to the Bidirectional layer, the output of the Bidirectional layer (40,392) is given as input to the Conv1d layer, the output of the Conv1d layer (40,128) is given as input to the flatten layer and the output of flattening layer 5120 is given as input to dense layer. The dense layer has 5 nodes with a sigmoid as an activation function. Bi-LSTM+CNN is trained with the categorical_crossentropy loss function and a Nadam optimizer.

5. Results and discussion

In this section, we present the experiments conducted to compare the performance of the proposed traditional machine learning, deep learning, transfer learning and hybrid deep learning models. The results of 13 sets of experiments: four traditional machine learning models (RF, LSVC, MNB and LR), four deep learning models (LSTM, BiLSTM, BiGRU and CNN), one transfer learning technique IndicBERT and four hybrid models (CNN+LSTM, LSTM+CNN, CNN+BiLSTM and BiLSTM+CNN) are shown in this section. All of these models were tested with the Tamil code-mixed data set, as described in Section 3.1, that has been preprocessed with various text processing techniques. To work with traditional machine learning models, the TF–IDF weighting scheme is utilized. Accuracy; macro-average precision, recall and F1-score; as well as Weighted average precision, recall and F1-score were the metrics used to evaluate the performance of these models.

5.1. *Precision, recall and F1-score for all models*

Table 7 represents the precision, recall and F1-score for the four classes, namely positive, negative, mixed_feelings and unknown_state, for each of the machine learning models, RF, LSVC, MNB and LR. Among all the proposed machine learning models, highest precision is obtained by the LSVC model (0.73) and the highest F1-Score (0.79) is achieved by the LR machine learning model when classifying the positive samples. Table 8 represents the precision, recall and F1-score for the four classes, namely, positive, negative, mixed_feelings and unknown_state, for each of the deep learning models, LSTM, BiLSTM, BiGRU and CNN. Among all the proposed deep learning models, highest precision is obtained by the BiGRU deep learning model (0.7**3**) and the highest F1-Score (0.79) is achieved by all the deep learning models when classifying the positive samples. Table 9 represents the precision, recall and F1-score for the four classes, namely, positive, negative, mixed_feelings and unknown_state for each of the hybrid models, CNN+LSTM, LSTM+CNN, CNN+BiLSTM and BiLSTM+CNN. Among all the proposed hybrid models, highest precision is obtained by BiLSTM+CNN hybrid model (0.75) and the highest F1-score (0.80) is achieved by CNN+BiLSTM hybrid model when classifying the positive samples.

5.2. *Classification report for tamil code-mixed data set*

The classification report for the Tamil code-mixed data set using the various traditional machine learning, deep learning, transfer learning and hybrid deep learning models are listed in Table 10. For machine learning models, the TF_IDF weighting scheme is used to select features. Among all the machine learning models included in this study, the logistic regression model obtained the highest accuracy of 0.65. Further, comparing the different deep learning models mentioned earlier, it was found that both BiGRU and CNN models produced the same accuracy of 0.65, the IndicBERT model achieved an accuracy of 0.62, and the hybrid deep learning approach combining CNN with BiLSTM achieved a high accuracy of 0.66. From these accuracy values, it is understood that the hybrid deep learning model CNN+BiLSTM works better among all, classifying the given Tamil code-mixed data into four classes with the highest accuracy of 0.66.

Table 7
Precision, recall and F1-score for machine learning models.

Machine learning models	Class labels	Precision	Recall	F1-score
RF	Positive	0.67	0.90	0.77
	Negative	0.42	0.26	0.32
	Mixed_feelings	0.27	0.09	0.13
	Unknown_state	0.50	0.25	0.33
LSVC	Positive	**0.73**	0.84	0.78
	Negative	0.42	0.34	0.37
	Mixed_feelings	0.27	0.15	0.19
	Unknown_state	0.44	0.38	0.40
MNB	Positive	0.64	0.98	0.78
	Negative	0.48	0.12	0.20
	Mixed_feelings	0.50	0.01	0.02
	Unknown_state	0.66	0.14	0.22
LR	Positive	0.71	0.90	**0.79**
	Negative	0.47	0.29	0.35
	Mixed_feelings	0.29	0.11	0.16
	Unknown_state	0.52	0.35	0.42

Table 8
Precision, recall and F1-score for deep learning models.

Deep learning models	Class labels	Precision	Recall	F1-score
LSTM	Positive	0.69	0.92	**0.79**
	Negative	0.34	0.37	0.35
	Mixed_feelings	0.42	0.11	0.17
	Unknown_state	0.34	0.06	0.10
BiLSTM	Positive	0.67	0.95	**0.79**
	Negative	0.46	0.15	0.23
	Mixed_feelings	0.36	0.16	0.23
	Unknown_state	0.57	0.18	0.27
BiGRU	Positive	**0.73**	0.87	**0.79**
	Negative	0.40	0.33	0.36
	Mixed_feelings	0.41	0.11	0.18
	Unknown_state	0.47	0.42	0.44
CNN	Positive	0.72	0.88	**0.79**
	Negative	0.37	0.37	0.37
	Mixed_feelings	0.41	0.04	0.08
	Unknown_state	0.50	0.38	0.43

Table 9
Precision, recall and F1-score for hybrid models.

Hybrid model	Class labels	Precision	Recall	F1-score
CNN+LSTM	Positive	0.69	0.91	0.79
	Negative	0.43	0.20	0.27
	Mixed_feelings	0.41	0.14	0.21
	Unknown_state	0.56	0.34	0.42
LSTM+CNN	Positive	0.73	0.87	0.79
	Negative	0.39	0.38	0.38
	Mixed_feelings	0.38	0.10	0.16
	Unknown_state	0.47	0.38	0.38
CNN+BiLSTM	Positive	0.71	0.91	**0.80**
	Negative	0.41	0.32	0.36
	Mixed_feelings	0.43	0.14	0.21
	Unknown_state	0.55	0.30	0.39
BiLSTM+CNN	Positive	**0.75**	0.84	0.79
	Negative	0.37	0.40	0.38
	Mixed_feelings	0.35	0.15	0.21
	Unknown_state	0.46	0.41	0.44

5.3. Confusion matrix

The confusion matrix obtained after applying various machine learning, deep learning, transfer learning and hybrid deep learning models is specified in Fig. 5. It is clear that out of 2546 support sentences in the Positive class, only 2313 sentences were correctly

Table 10
Classification report for Tamil code-mixed data set.

Models	ACC(BP)	ACC(AP)	Pmac	Rmac	F1mac	Pw	Rw	F1w
RF	0.61	0.63	0.48	0.38	0.39	0.58	0.63	0.58
LSVC	0.62	0.63	0.46	0.43	0.44	0.59	0.63	0.61
MNB	0.61	0.64	**0.57**	0.31	0.31	0.61	0.64	0.54
LR	0.63	0.65	0.50	0.41	0.43	0.60	0.65	0.61
LSTM	0.62	0.63	0.44	0.37	0.35	0.56	0.63	0.56
BiLSTM	0.61	0.64	0.52	0.36	0.38	0.60	0.64	0.58
BiGRU	0.60	0.65	0.50	0.43	0.44	0.61	0.65	**0.62**
CNN	0.62	0.65	0.50	0.42	0.42	0.61	0.65	0.61
CNN+LSTM	0.61	0.65	0.52	0.40	0.42	0.61	0.55	0.60
LSTM+CNN	0.61	0.65	0.49	0.43	0.44	0.61	0.65	**0.62**
CNN+BiLSTM	0.63	**0.66**	0.53	0.42	0.44	**0.62**	**0.66**	**0.62**
BiLSTM+CNN	0.61	0.64	0.48	**0.45**	**0.45**	**0.62**	0.64	**0.62**
IndicBERT	0.60	0.62	0.49	0.35	0.36	0.57	0.62	0.55

Table 11
Comparative analysis of samples taken from test data set.

Example	Actual	Positive	Negative	Mixed_feelings	Unknown_state
Org:JJ mam we miss u **Trans:** JJ mam we miss you	Positive	Yes	No	No	No
Org:chiyan neanga fast and furious ha nadikalam **Trans:** Chiyan you can act fast and furious	Positive	No	Yes	No	No
Org: Vijay anna maari thala enna mo try pandranga pakalam **Trans:**Thala is trying something as vijay bro lets see.	Negative	No	Yes	No	No
Org: Oru shotla kooda target achieve pannala **Trans:** The target is not achieved even in single shot	Negative	No	No	Yes	No

classified as positive. The remaining 233 positive sentences were wrongly classified into Mixed_feelings (59), Negative (83) and Unknown_State (91) with the multinomial Naive Bayes approach. Among all the deep learning models in this study, BiLSTM techniques produced the best results by classifying 2407 out of 2546 sentences as positive. The remaining 139 positive sentences were wrongly classified into Mixed_feelings (51), Negative (34) and Unknown_State (54). In the hybrid models, CNN+LSTM worked better, wherein 2328 sentences were correctly classified as positive. The remaining 218 positive sentences were wrongly classified into Mixed_feelings (24), Negative (64) and Unknown_State (127).

Finally, a general summary of the results achieved in the experiments referenced earlier are discussed as follows:

1. The hybrid models, specifically CNN with BiLSTM, had an increased accuracy for sentiment analysis on Tamil code-mixed data set, compared with all other models;
2. The combination helped to take advantage of the strengths of CNN and BiLSTM, wherein CNN has the ability to extract characteristics from the Tamil code-mixed data set and BiLSTM has the ability to store past information;

5.4. Error analysis

This section looks at the quantitative and qualitative performance of our framework. We begin our examination by starting with a few output instances anticipated by our system. Using the pseudo-labeling technique, we additionally present their original class labels as well as alternative predicted labels. The first sentence is an example of a Positive class label in which the model gets predicted correctly. The same is followed for all the classes such as positive, negative, mixed_feelings and unknown_state. Sample outputs from our unified framework are illustrated in Table 11. The proposed hybrid model BiLSTM+CNN correctly classifies the 1st sentence into the positive sentiment class. Next, consider the 2nd sentence, which belongs to the positive class. It can be observed that the proposed model wrongly classifies it into the negative category. The reason for this misclassification may be because of the word “furious”, which the proposed model predicted as having a negative sentiment. If the model is trained based on not only words but also context, this misclassification can be avoided, and accuracy can be further improved. The proposed hybrid model BiLSTM+CNN correctly classifies the 3rd sentence into the negative sentiment class. Next, consider the 4th sentence, which belongs to negative category. It can be observed that the proposed model wrongly classifies it into mixed_feelings class based on the context present in the sentence.

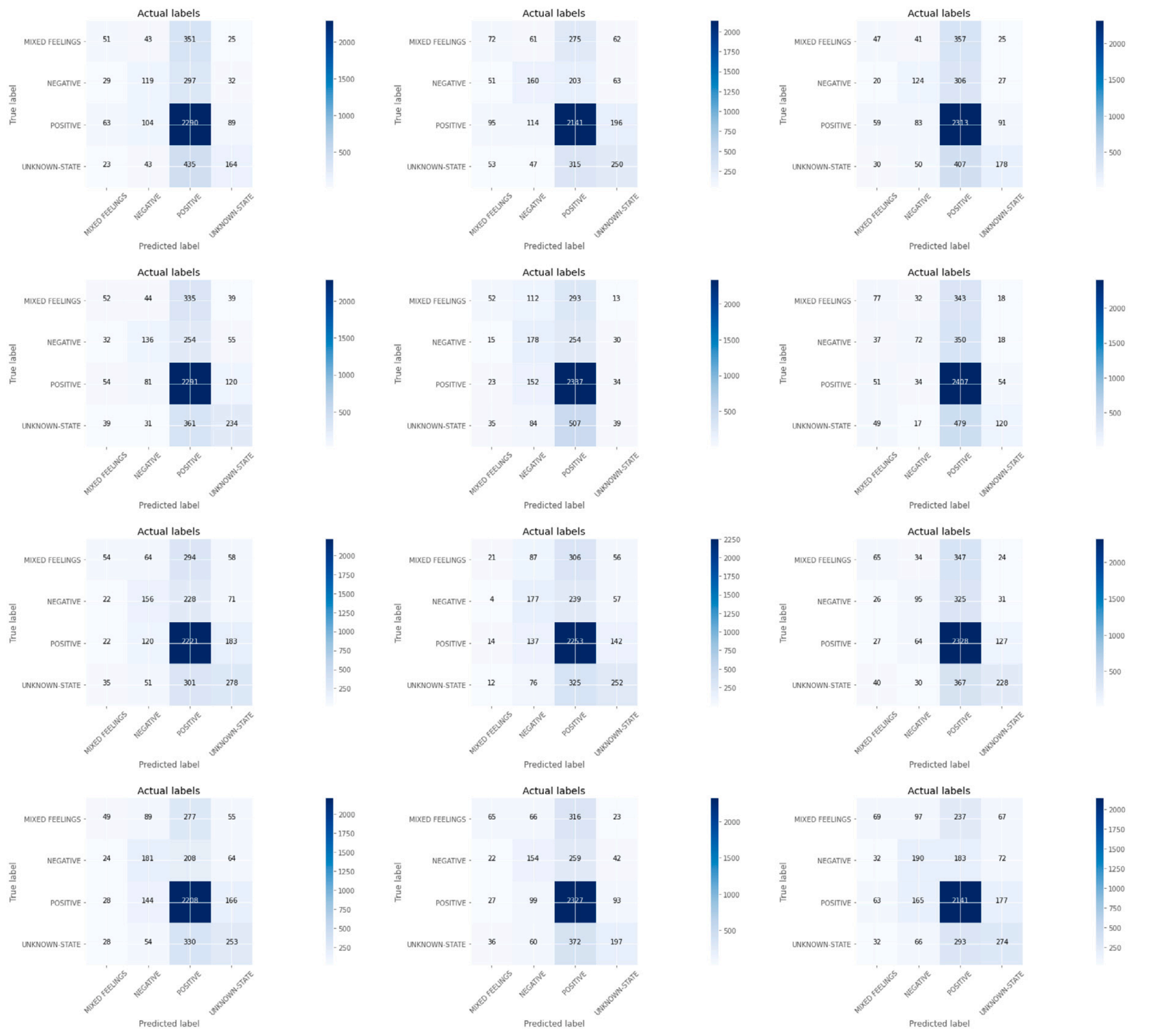

Fig. 5. Confusion Matrices for Random Forest, Linear SVC, Multinominal Naive Bayes, Logistic Regression, LSTM, BiLSTM, BiGRU, CNN, CNN+LSTM, LSTM+CNN, CNN+BiLSTM and BiLSTM+CNN.

6. Conclusion

Many real-world applications, such as review analysis and recommendation systems, benefit from sentiment analysis. When data is noisy and gathered through social media, sentiment analysis becomes more challenging, particularly when the data is code-mixed. In this research work, to check class imbalance in Tamil code-mixed data, class re-sampling is performed and the accuracy before and after applying re-sampling techniques is compared and analyzed. In addition, the importance of pre-processing techniques on Tamil code-mixed data is examined by applying three pre-processing techniques, namely emojis removal, repeated characters removal and punctuations, symbols and number removal, on the code-mixed data. Both raw as well as pre-processed data are applied to various state-of-the-art classifiers. The result shows that pre-processed data produce better accuracy. Furthermore, to extract features from the Tamil code-mixed data, the TF–IDF weighting scheme is adopted by the traditional machine learning approaches included in this study. Then, the proposed research work performs sentiment analysis on the Tamil code-mixed data to classify them into four classes, namely positive, negative, mixed_feelings and unknown_state, using various traditional machine learning, deep learning, transfer learning and hybrid deep learning models. The state-of-the-art classifiers utilized include traditional machine learning models such as random forest, multinomial Naive Bayes, logistic regression and linear SVC; deep learning models such as LSTM, BiLSTM, BiGRU and CNN; a transfer learning model called IndicBERT and hybrid deep learning models such as CNN+LSTM, LSTM+CNN, CNN+BiLSTM and BiLSTM+CNN. Among traditional machine learning models, logistic regression produced an accuracy of **0.63**; among various deep learning models, BiLSTM produced an accuracy of **0.63**; and the transfer learning model produced an accuracy of **0.62**. The novelty of this research work is the development of hybrid deep learning models by combining CNN+LSTM, LSTM+CNN, CNN+BiLSTM and BiLSTM+CNN. The findings from this research work indicate that the hybrid deep learning model CNN+BiLSTM works well to perform sentiment analysis on Tamil code-mixed data, producing an accuracy of **0.66**. In future, context-based models such as Elmo, ULMFiT language models should be utilized to obtain better accuracy in sentiment analysis on social media data. In addition, sentiment analysis on multimodal data set should also be focused.

CRediT authorship contribution statement

Kogilavani Shanmugavadivel: Conceptualization, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Sai Haritha Sampath:** Methodology, Visualization, Writing – original draft. **Pramod Nandhakumar:** Methodology, Formal analysis, Software, Writing – editing. **Prasath Mahalingam:** Methodology, Formal analysis, Software, Writing – editing. **Malliga Subramanian:** Methodology, Formal analysis, Software, Writing – original draft, Writing – editing. **Prasanna Kumar Kumaresan:** Methodology, Formal analysis, Software, Writing – original draft, Writing – editing. **Ruba Priyadharshini:** Data curation, Writing – review & editing.

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.

References

Abdelwahab, O., Bahgat, M., Lowrance, C.J., Elmaghraby, A., 2015. Effect of training set size on SVM and Naive Bayes for Twitter sentiment analysis. In: 2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT). IEEE, pp. 46–51.

Alessia, D., Ferri, F., Grifoni, P., Guzzo, T., 2015. Approaches, tools and applications for sentiment analysis implementation. Int. J. Comput. Appl. 125 (3).

Bali, K., Sharma, J., Choudhury, M., Vyas, Y., 2014. "I Am borrowing ya mixing?" An analysis of english-hindi code mixing in facebook. In: Proceedings of the First Workshop on Computational Approaches To Code Switching. pp. 116–126.

Barbosa, L., Feng, J., 2010. Robust sentiment detection on twitter from biased and noisy data. In: Coling 2010: Posters. pp. 36–44.

Barman, U., Das, A., Wagner, J., Foster, J., 2014. Code mixing: A challenge for language identification in the language of social media. In: Proceedings of the First Workshop on Computational Approaches To Code Switching. pp. 13–23.

Bharathi, B., Chakravarthi, B.R., Chinnaudayar Navaneethakrishnan, S., Sripriya, N., Pandian, A., Valli, S., 2022. Findings of the shared task on speech recognition for vulnerable individuals in Tamil. In: Proceedings of the Second Workshop on Language Technology for Equality, Diversity and Inclusion. Association for Computational Linguistics.

Bhuvan, M.S., Rao, V.D., Jain, S., Ashwin, T., Guddeti, R.M.R., 2015. Semantic sentiment analysis using context specific grammar. In: International Conference on Computing, Communication & Automation. IEEE, pp. 28–35.

Bojanowski, P., Grave, E., Joulin, A., Mikolov, T., 2017. Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. 5, 135–146.

Cambria, E., Schuller, B., Xia, Y., Havasi, C., 2013. New avenues in opinion mining and sentiment analysis. IEEE Intell. Syst. 28 (2), 15–21.

Chakravarthi, B.R., 2020. HopeEDI: A multilingual hope speech detection dataset for equality, diversity, and inclusion. In: Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media. Association for Computational Linguistics, Barcelona, Spain (Online), pp. 41–53, URL: https://aclanthology.org/2020.peoples-1.5.

Chakravarthi, B.R., Jose, N., Suryawanshi, S., Sherly, E., McCrae, J.P., 2020a. A sentiment analysis dataset for code-mixed malayalam-english. In: Proceedings of the 1st Joint Workshop on Spoken Language Technologies for under-Resourced Languages (SLTU) and Collaboration and Computing for under-Resourced Languages (CCURL). European Language Resources association, Marseille, France, pp. 177–184, URL: https://aclanthology.org/2020.sltu-1.25.

Chakravarthi, B.R., Muralidaran, V., 2021. Findings of the shared task on hope speech detection for equality, diversity, and inclusion. In: Proceedings of the First Workshop on Language Technology for Equality, Diversity and Inclusion. Association for Computational Linguistics, Kyiv, pp. 61–72, URL: https://aclanthology.org/2021.ltedi-1.8.

Chakravarthi, B.R., Muralidaran, V., Priyadharshini, R., McCrae, J.P., 2020b. Corpus creation for sentiment analysis in code-mixed Tamil-English text. In: Proceedings of the 1st Joint Workshop on Spoken Language Technologies for under-Resourced Languages (SLTU) and Collaboration and Computing for under-Resourced Languages (CCURL). European Language Resources association, Marseille, France, pp. 202–210, URL: https://aclanthology.org/2020.sltu-1.28.

Chakravarthi, B.R., Muralidaran, V., Priyadharshini, R., McCrae, J.P., 2020c. Corpus creation for sentiment analysis in code-mixed Tamil-English text. In: Proceedings of the 1st Joint Workshop on Spoken Language Technologies for under-Resourced Languages (SLTU) and Collaboration and Computing for under-Resourced Languages (CCURL). European Language Resources association, Marseille, France, pp. 202–210, URL: https://aclanthology.org/2020.sltu-1.28.

Chakravarthi, B.R., Priyadharshini, R., Durairaj, T., McCrae, J.P., Buitaleer, P., Kumaresan, P.K., Ponnusamy, R., 2022a. Findings of the shared task on homophobia transphobia detection in social media comments. In: Proceedings of the Second Workshop on Language Technology for Equality, Diversity and Inclusion. Association for Computational Linguistics.

Chakravarthi, B.R., Priyadharshini, R., Muralidaran, V., Jose, N., Suryawanshi, S., Sherly, E., McCrae, J.P., 2022b. DravidianCodeMix: sentiment analysis and offensive language identification dataset for dravidian languages in code-mixed text. Language Resour. Evalu http://dx.doi.org/10.1007/s10579-022-09583-7, URL: http://dx.doi.org/10.1007/s10579-022-09583-7.

Chakravarthi, B.R., Priyadharshini, R., Muralidaran, V., Suryawanshi, S., Jose, N., Sherly, E., McCrae, J.P., 2020d. Overview of the track on sentiment analysis for dravidian languages in code-mixed text. In: Forum for Information Retrieval Evaluation. pp. 21–24.

Chakravarthi, B.R., Priyadharshini, R., Ponnusamy, R., Kumaresan, P.K., Sampath, K., Thenmozhi, D., Thangasamy, S., Nallathambi, R., McCrae, J.P., 2021. Dataset for identification of homophobia and transophobia in multilingual YouTube comments. arXiv preprint arXiv:2109.00227.

Devlin, J., Chang, M., Lee, K., 2019. Kt google, and AI language,"BERT: Pretraining of deep bidirectional transformers for language understanding". Tech. Rep.

Farrugia, P.-J., 2004. Tts pre-processing issues for mixed language support. University of Malta. Faculty of ICT.

Gamallo, P., Garcia, M., 2014. Citius: A naivebayes strategy for sentiment analysis on english tweets. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014. Citeseer.

Ganie, A.G., Dadvandipour, S., 2021. Sentiment analysis on the effect of trending source less news: Special reference to the recent death of an Indian actor. In: International Conference on Artificial Intelligence and Sustainable Computing. Springer, pp. 3–16.

Gautam, G., Yadav, D., 2014. Sentiment analysis of twitter data using machine learning approaches and semantic analysis. In: 2014 Seventh International Conference on Contemporary Computing (IC3). IEEE, pp. 437–442.

Gupta, F., Singal, S., 2017. Sentiment analysis of the demonitization of economy 2016 India, regionwise. In: 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence. IEEE, pp. 693–696.

Hasan, A., Moin, S., Karim, A., Shamshirband, S., 2018. Machine learning-based sentiment analysis for twitter accounts. Math. Comput Appl. 23 (1), 11.

Hegde, B., NH, S., Prakash, M., 2018. Sentiment analysis of Twitter data: A machine learning approach to analyse demonetization tweets. Int. Res. J. Eng. Technol.

Jose, N., Chakravarthi, B.R., Suryawanshi, S., Sherly, E., McCrae, J.P., 2020. A survey of current datasets for code-switching research. In: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE, pp. 136–141.

Kamps, J., Marx, M., Mokken, R.J., De Rijke, M., et al., 2004. Using WordNet to measure semantic orientations of adjectives.. In: LREC. 4, Citeseer, pp. 1115–1118.

Kanakaraj, M., Guddeti, R.M.R., 2015. Nlp based sentiment analysis on Twitter data using ensemble classifiers. In: 2015 3Rd International Conference on Signal Processing, Communication and Networking (ICSCN). IEEE, pp. 1–5.

Kudo, T., 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. arXiv preprint arXiv:1804.10959.

Le, B., Nguyen, H., 2015. Twitter sentiment analysis using machine learning techniques. In: Advanced Computational Methods for Knowledge Engineering. Springer, pp. 279–289.

Mullen, T., Collier, N., 2004. Sentiment analysis using support vector machines with diverse information sources. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing. pp. 412–418.

Myers-Scotton, C., 1993. Common and uncommon ground: Social and structural factors in codeswitching. Lang Soc 22 (4), 475–503.

Onan, A., 2021. Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks. Concurr. Comput 33.

Onan, A., Toçoğlu, M.A., 2021. A term weighted neural language model and stacked bidirectional lstm based framework for sarcasm identification. IEEE Access 9.

Parveen, H., Pandey, S., 2016. Sentiment analysis on Twitter data-set using Naive Bayes algorithm. In: 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (ICATccT). IEEE, pp. 416–419.

Patel, A.P., Patel, A.V., Butani, S.G., Sawant, P.B., 2017. Literature survey on sentiment analysis of Twitter data using machine learning approaches. IJIRST-Int. J. Innovat. Reas Sci Technol 3 (10).

Pennington, J., Socher, R., Manning, C.D., 2014. Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543.

Priyadharshini, R., Chakravarthi, B.R., Chinnaudayar Navaneethakrishnan, S., Durairaj, T., Subramanian, M., Shanmugavadivel, K., U Hegde, S., Kumaresan, P.K., 2022. Findings of the shared task on abusive comment detection in Tamil. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics.

Priyadharshini, R., Chakravarthi, B.R., Vegupatti, M., McCrae, J.P., 2020. Named entity recognition for code-mixed Indian corpus using meta embedding. In: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE, pp. 68–72.

Ramadhani, A.M., Goo, H.S., 2017. Twitter sentiment analysis using deep learning methods. In: 2017 7th International Annual Engineering Seminar (InAES). IEEE, pp. 1–4.

Ravikiran, M., Chakravarthi, B.R., Madasamy, A.K., Sivanesan, S., Rajalakshmi, R., Thavareesan, S., Ponnusamy, R., Mahadevan, S., 2022. Findings of the shared task on offensive span identification in code-mixed tamil-english comments. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics.

Sahni, T., Chandak, C., Chedeti, N.R., Singh, M., 2017. Efficient Twitter sentiment classification using subjective distant supervision. In: 2017 9th International Conference on Communication Systems and Networks (COMSNETS). IEEE, pp. 548–553.

Sakuntharaj, R., Mahesan, S., 2016. A novel hybrid approach to detect and correct spelling in Tamil text. In: 2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS). pp. 1–6. http://dx.doi.org/10.1109/ICIAFS.2016.7946522.

Sakuntharaj, R., Mahesan, S., 2017. Use of a novel hash-table for speeding-up suggestions for misspelt Tamil words. In: 2017 IEEE International Conference on Industrial and Information Systems (ICIIS). pp. 1–5. http://dx.doi.org/10.1109/ICIINFS.2017.8300346.

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.

Sampath, A., Durairaj, T., Chakravarthi, B.R., Priyadharshini, R., Chinnaudayar Navaneethakrishnan, S., Shanmugavadivel, K., Thavareesan, S., Thangasamy, S., Krishnamurthy, P., Hande, A., Benhur, S., Ponnusamy, K.K., Pandiyan, S., 2022. Findings of the shared task on emotion analysis in Tamil. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics.

Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

Shobana, G., Vigneshwara, B., Maniraj Sai, A., 2018. Twitter sentimental analysis. Int. J. Recent Technol Eng. (IJRTE) 7.

Thavareesan, S., Mahesan, S., 2019. Sentiment analysis in Tamil texts: A study on machine learning techniques and feature representation. In: 2019 14th Conference on Industrial and Information Systems (ICIIS). pp. 320–325. http://dx.doi.org/10.1109/ICIIS47346.2019.9063341.

Thavareesan, S., Mahesan, S., 2020a. Sentiment lexicon expansion using Word2vec and fastText for sentiment prediction in Tamil texts. In: 2020 Moratuwa Engineering Research Conference (MERCon). pp. 272–276. http://dx.doi.org/10.1109/MERCon50084.2020.9185369.

Thavareesan, S., Mahesan, S., 2020b. Word embedding-based part of speech tagging in Tamil texts. In: 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS). pp. 478–482. http://dx.doi.org/10.1109/ICIIS51140.2020.9342640.

Thavareesan, S., Mahesan, S., 2021. Sentiment analysis in Tamil texts using k-means and k-nearest neighbour. In: 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS). pp. 48–53. http://dx.doi.org/10.1109/ICIAfS52090.2021.9605839.

Trupthi, M., Pabboju, S., Narasimha, G., 2017. Sentiment analysis on twitter using streaming API. In: 2017 IEEE 7th International Advance Computing Conference (IACC). IEEE, pp. 915–919.

Tsapatsoulis, N., Djouvas, C., 2017. Feature extraction for tweet classification: Do the humans perform better? In: 2017 12th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP). IEEE, pp. 53–58.

Zhang, M.-L., Peña, J.M., Robles, V., 2009. Feature selection for multi-label naive Bayes classification. Inform. Sci. 179 (19), 3218–3229.