A Comprehensive Survey of Machine Learning Approaches for Sentiment Analysis in Code-Mixed Languages

Prasad A. Joshi [1], Varsha M. pathak [2]

[1]*JET's Zulal Bhilajirao Patil College, Dhule*
[2]*Institute of Management and Research, Jalgaon*

Abstract: *Code-mixing on social media and networking sites offers a simpler platform for people to express themselves from anywhere in the world without exerting any effort. These expressions can have disrespectful language, insulting comments, and trolling, which can destabilize social harmony. Analyzing such negative sentiments is crucial if we want to stop their spread. This paper investigates the efforts of the researchers in analyzing sentiments in code-mixing. It covers different stages of sentiment analysis in code-mixing that include dataset preparation, feature extraction techniques, and classifier implementations. It presents a generalized model for preparing datasets for sentiment analysis in code-mixed languages. It discusses a variety of issues, such as class imbalance and the inter-annotator agreement approach. The best performing classifiers and their comparative analysis with other classifiers are emphasized for the understanding of the researchers in this very recent research area. The respective charts and tables are used to visualize the analysis and interpretation of this comparative study. This extensive survey encompasses relevant research articles in this domain. The outcome of this review article will undoubtedly help researchers and developers to understand the technical aspects of this field. The generic functional approach of this study will be useful for developing sentiment analysis models in code-mixed languages.*

Keywords: code-mixing, sentiment analysis, machine learning, deep learning, SVM, RF, etc.

1. Introduction:

In the twenty-first century, social media platforms like YouTube, Facebook, Twitter, WhatsApp, Instagram, Twitter, etc. are considered the major spinners for a regular increase in Internet users all over the world. Concerning India, the statistics show that 448 million people are social media users, which is roughly 32% of the total population. Last year, 31.2% of new social media users were added [1]**.** This statistic confirms that most Indian people rely on social media platforms. Indians used these platforms to express their feelings, thoughts, and views on different topics.

These views are casual in nature. In social media applications, they have different names such as comments, reviews, posts, tweets, etc. On social media, people use their native language or English language, i.e., they use monolingualism to express their views on state or national issues. However, most Indians cannot express themselves freely in English, whereas native languages have a restricted reach on national issues. At such times, mixed languages [2] or code-mixing or code-switching resolve this issue.

A mixed language is a language that has a bilingual group of two or more languages, whereas code-mixing or code-switching stands for the mixing of two or more languages in speech or writing. According to linguists, mixed language, code-mixing, or code-switching are the same terms used interchangeably by researchers and academicians. A minor difference exists between code-switching and code-mixing. In code-switching, the user intentionally uses his style to prove a point, while in code-mixing; the user does it with no intention because of a lack of vocabulary. The examples of code-mixed languages

along with different sentiments are given in Table 1. Use of mixed languages or code-mixed is easier than monolingual, and it also helps with productive and creative expression in real time [3]. Additionally, multiple languages in day-to-day conversations and heavy use of them on social media platforms lead to code-mixing. In recent times, the use of code-mixed languages has potentially increased the number of social media users all over the world. In the case of India, there are several factors for potential growth in social media users' viz.: 22 scheduled languages, freedom of speech and expression for every Indian, 65% of the population below the age of 35, and Indians' ease of using code-mixed languages.

These views or comments may be positive or negative. Negative comments may contain hate speech, absurd or offensive comments, which may damage a person or community. It also offers violent people the opportunity to publicize their acts. In India in 2020, an offensive Facebook post about the Prophet Muhammad played a critical part in causing brutal conflicts in Bengaluru, India. Tragically, this is not the only incident [4]. In Germany, attacks on refugees strongly relate to anti-refugee Facebook posts. [5], In Myanmar, military leaders and Buddhist nationalists used social media to insult the Rohingya Muslim minority [6]. In 2018, Sri Lanka witnessed anti-Muslim riots stimulated by rumours spread online [7], in 2015; the United States also saw an attack on black clergy by white supremacists, which were part of the online self-learning process [8]. These are some of the cases mentioned, but hate speech comments on various social media platforms have brought about crowd savagery, lynching, communal riots, and a lot more horrible occasions. Due to these incidences, many countries have made strategies against hate speech content on social media. Likewise, the Law Commission of India has introduced non-legal measures which have strategic involvement to monitor the dissemination of hate speech. [9].

Table 1. Examples of code-mixed language with and sentiments

Code-mixed Language	**Examples of Code-mixed language**	**Sentiment**
Tamil-English ***English Translation***	yematha arambichitanunga nambathinga *(It is unbelievable to start cheating)*	Not Offensive
Malayalam-English ***English Translation***	Areyum poori ennu vilikkum munpu iyaalum oru pooril ninnaanu vannathennu orkanam. Aa poorinte samskaaram. Cheta!!! *(You can not vote in polls in this forum. Aa poorinte samskaaram. Cheta !!!)*	Offensive
Hindi-English ***English Translation***	Aisa PM naa hua hai aur naa hee hoga. [Aditya Josh, 2016] *Neither there has been a PM like him, nor there will be*	Positive
Bengali-English ***English Translation***	Script ta khub tiring chilo amar mote, aro onek better hote parto *The script was very tiring according to me, could have been much better*	Negative

Finding such hateful content from social media text is a necessary task. Substantial work has been done to detect hate content in social media text in monolingual languages, but very few efforts have been made to detect such content in code-mixed social media text. So far, world-wide, different monolingual languages like English, Danish, Russian, German, and Indonesian are considered to have hate content. In the case of Indian languages, Hindi, Bangla, Tamil, Telugu, etc. are considered to have hate content. Code-mixed languages such as Chinese-English, Hindi-English, English-Bengali, Tamil-English, Telugu-English, and Malayalam-English text comments are considered to detect hate content or to categorize positive and negative sentiments. For researchers and academicians, finding sentiments from code-mixed data is a demanding task because of its characteristics, which increase the overall complexity.

Some of the characteristics of code-mixed are:

i. Difficulty in detecting the start and end of a sentence
ii. Word order is not followed in a sentence
iii. Mixture of grammar of mixed languages
iv. Variations in spellings, high proportion of spelling mistakes
v. Creative spellings using abbreviations [10].

Together with sentiment analysis, other research areas in code-mixed text are:

i. Part-Of-Speech tagging,
ii. Machine Translation (MT)
iii. Language Identification
iv. Mixed Script Information Retrieval (MSIR)
v. Question Answering.

This survey paper is significant for many reasons. First, it gives details about constructing the code-mixed datasets containing different sentiments, such as positive, negative, or hate speech, not hate speech. This will help the researchers to construct the new datasets in code-mixed text and will further assist with data preprocessing and labelling techniques. The second reason is that different techniques regarding extracting and selecting the features were discussed. It gives comprehensive insight to the researchers into selecting the appropriate feature extraction technique. Third, it presents experimental results with respect to different machine learning, deep learning, and transfer learning classifiers. This can help novel researchers in the arena of classifiers to have a panoramic view of the entire arena.

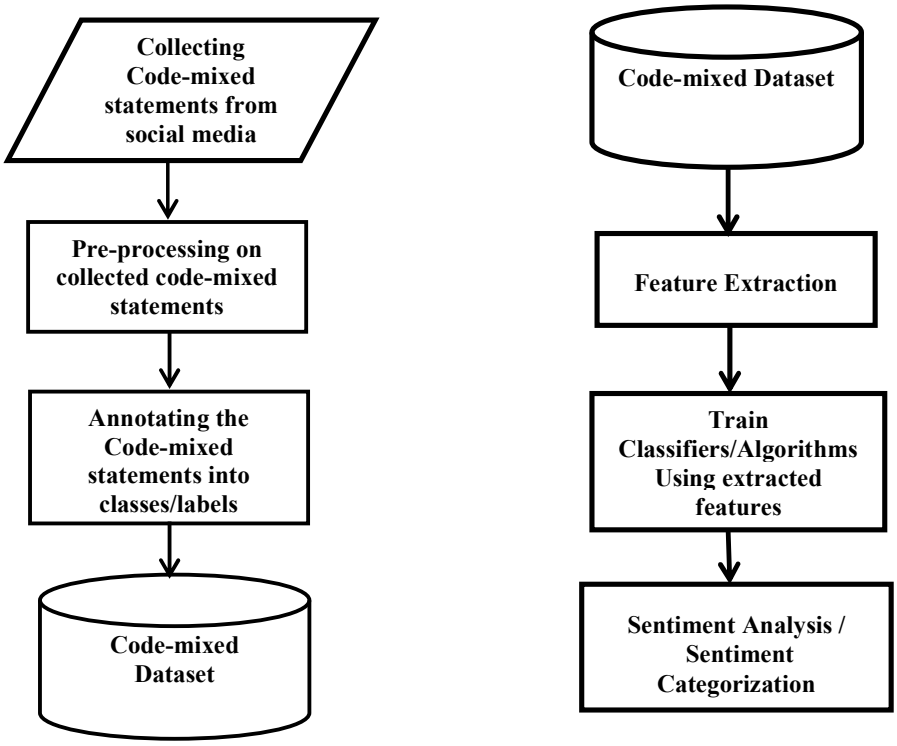
This paper is organized as follows: Section 2 comprises tools, annotation methods, and social media platforms used to create datasets in code-mixed languages. This section also has feature selection techniques and different classifiers used to categorize the datasets. Section 3 presents the results and discussions, and finally, the conclusion is discussed in Section 4.

2. Code-mixed language datasets

As mentioned previously, the task of sentiment analysis is still untouched in many code-mixed Indian languages and code-mixed foreign languages. In India, out of 22 scheduled languages, only 5 code-mixed datasets are available for sentiment analysis. This section will explore different tools and techniques used to create datasets in code-mixed languages. Along with this, it will also discover the classifiers and features used for categorizing the dataset into sentiments.

Vyas et al., 2014 [11] have developed a Hindi-English dataset using popular Facebook public pages of Amitabh Bachchan, Shahrukh Khan, Narendra Modi, and the BBC Hindi news. The collected posts were pre-processed and thus the dataset has 6,983 posts, which are then annotated. Annotation is done at the language, word, transliteration, and POS levels. On the annotated dataset, the authors applied language identification, normalization, and POS tagging.

In 2015, Sophia and Zhongqing [12] used the Chinese social media platform Weibo to collect posts in English, Chinese, both, and mixed languages. The posts were collected from different domains such as: life, finance, service, celebrities, products, and politics. From the collected dataset, noise and advertisement posts were removed. Annotation is divided into five categories: happiness, sadness, fear, anger, and surprise using a designed annotation tool. To check the annotation quality, 1000 posts were annotated by two people, and then inter-annotator agreement between the posts was calculated using Cohen's Kappa coefficient. From the collected dataset, 50% of the data is considered as training data and the rest is considered as test data. Unigrams for each post were extracted as features. FudanNLP is used for segmenting Chinese words. A Maximum Entropy (ME) algorithm is used for training and is trained using extracted unigrams. Experimentation is done with the MALLET Toolkit.



(a) Steps for Code-mixed dataset preparation (b) Steps for Sentiment Analysis

Figure 1: Generalized model / Structure to analyse sentiments from Code-mixed languages

Aditya Joshi et al., 2016 [13] have performed sentiment analysis on a Hindi-English code-mixed dataset into three classes: positive, negative, and neutral. The dataset has

comments from the Facebook pages of two popular Indian icons: Narendra Modi and Salman Khan. Their pages have millions of followers and have lots of comments. The comments in Roman script and in English sentences were removed. Comments having a length over 50 words were also eliminated. The comments with more than one sentence were also removed. Then annotation is applied by two annotators to the above classes. The final dataset has 15% negative, 50% neutral, and 35% positive comments. On the developed and annotated dataset in the code-mixed Hindi-English dataset, Aditya Joshi et al., 2016 applied sub-word level representation. In the sub-word level feature, the individual total of the words in a given sentence is counted and combined. This score is used to determine to which class the sentence belongs, i.e., positive, negative, and neutral. This character embedding is fed to the CNN-1D layer with a filter and a bias, resulting in a sub-word level feature map. The sub-word level is submitted to the globalMaxPooling layer, which in turn is submitted to the LSTM layer, so that it can predict the class of the sentence. The Subword-LSTM system gives an F-score of 0.658, which is better than Char-LSTM, which has an F-score of 0.511.

Souvick et al., 2017 [14] performed sentiment analysis by classifying the English-Bengali posts into positive, negative, and neutral classes. They used a dataset from a shared task on POS tagging of transliterated social media text, conducted by ICON-2015. The dataset has Facebook posts in English-Bengali with a few Hindi words, which are annotated by two annotators into the above mentioned classes. The inter-annotator agreement is measured using the Kappa co-efficient. The posts were preprocessed by removing punctuation, multiple character repetitions, and expanding the abbreviations. At a later stage, different features were extracted. Four sets of lists of words were found by matching with Sentiwordnet, opinion lexicon, English sentiment words, and Bengali sentiment words. Other features were also extracted, like: the number of colloquial Bengali sentiment words; bad words; all uppercase words; exclamation points; question marks; smiley matches; character repetitions; and part-of-speech tags. The experiments were performed using WEKA by dividing the dataset into training and testing sets. The features extracted were grouped into word-based, syntactic, and style-based. Experiment shows that a combination of word-based and syntactic features produces the best results.

In the year 2017, a Shared task named as SAIL (Sentiment Analysis of Indian Language) [15] conducted with the goal to identify the sentiment categorization of the code-mixed Hindi-English and Bengali-Hindi-English languages. The common Bengali and Hindi words in Roman format were searched and collected. These common words were then searched using the Twitter4j API and tweets were collected. The incomplete tweets, tweets not having Bengali or Hindi words, and spam tweets were totally removed. Hashtags and URLs are kept as it is. The labelling of the tweets into positive, negative, and neutral categories is done manually. The dataset is split into training and test sets. The authors performed a random baseline system with a macro average f-score of 0.331 and 0.339 for the Hindi-English and Bengali-Hindi-English datasets, respectively. A total of 40 participants registered, but only nine teams have submitted results. The techniques used by the teams are discussed below.

Experimental setup performed by participants: Team, "BIT Mesra" [15], participated only in predicting sentiment in the Hindi-English dataset. Before feature extraction, they preprocessed the tweets by removing words with URLs, UN language tags, hashtags, and user mentions. Along with this, an Emoji dictionary was created with sentiment tags. Features like unigram and bigram were prepared on which the machine classifiers like SVM and Naive Bayes are trained. The team got an F-score of 56.4 and placed second. The team "NLP CEN AMRITA" [15] has used different distributional and distributed representations. They used a document term matrix with N-grams varying from 1 to 5 for the representation and Support Vector Machines (SVM) as a classifier to make the final

prediction. Using the linear kernel, their system performed well for n-gram range 5 and minimum document frequency 2. The team "CFIL" [15] used simple deep learning for sentiment analysis on code-mixed data. The fastText tool is used to create word embeddings on sentiment corpora. Additionally, convolutional neural networks were used to extract sub-word features. A bi-LSTM layer is used on word embedding and sub-word features, together with max-pooling at the output, which is again sent to a softmax layer for prediction. No additional features are used, and hyper-parameters are selected after dividing the training corpus into 70% and 30%. The "Subway" [15], team submitted systems for the HI-EN dataset only. Initially, words other than HI and EN tags are removed during the cleaning process. Then, a dictionary with bi-grams and tri-grams is collected from training data, and sentiment polarity is annotated manually. The TF-IDF scores for each matched n-grams are calculated, and weights of 1.3 and 0.7 are assigned to bi-grams and tri-grams, respectively. Finally, the Naive Bayes classifier is used to get the sentiment.

Pruthwik Mishra et al. [16] named their team "IIIT-NBP", and they got the first rank with a macro average f-score of 0.569 for the HI-EN dataset and for the BN-EN dataset 0.526. They used different features for both datasets, such as TF-IDF vectors for character n-grams ranging from 2 to 6, and GloVe word embedding with 300 dimensions. On the extracted features, two models were applied: the first is an ensemble model having classifiers such as linear SVM, logistic regression, and random forest, and the second is a linear SVM. Along with this, the authors also applied MultiLayer Perceptron (MLP), Bi-LSTM trained using Glove, and TF-IDF. The team JU_KS [17] used n-gram and SentiWordNet features. For the Bengali language, they collected 1700 positive and 3750 negative words, and for the English language, 2006 positive and 4783 negative words were collected. Finally, the Naive Bayes classifier is used to classify the tweets into the mentioned categories. With an F-score of 50.4 for Bengali-Hindi and 56.2 for Hindi-English, the team achieves the 3rd rank.

Soumil Mandal et al. created a Bengali-English code-mixed dataset for sentiment analysis in March 2018 [18]. For data collection, they created a list of positive and negative Bengali words. With the help of these words, tweets were collected using Twiter4j. Around 89,000 tweets were collected, out of which spam, incomplete tweets, and tweets with conflicting sentiments were removed. Thus, after filtering, 10,000 tweets were available. Out of these, 5000 code-mixed tweets were manually selected. Annotation is done by humans, and it is divided into two stages. In the first stage, a single annotator with Bengali as their mother tongue and a computer science background is used, and in the second stage, five experts carry out the final evaluation process. These tweets are annotated using a hybrid system. The newly developed hybrid system has a rule-based and supervised method for language and sentiment tagging. In language tagging, a tweet is labelled at word level as: Bengali (BN), English (EN), or Unknown (UN). In sentiment tagging, a tweet is labelled as positive, negative, or neutral. In the supervised method, different classifiers such as Gaussian Naive Bayes (GNB), Bernoulli Naive Bayes (BNB), Multinomial Naive Bayes (MNB), Linear Regression (LRC), and Stochastic Gradient Descent (SGDC) were used. Using a manually annotated dataset, they trained these classifiers and found that SGDC got the best F1-Score of 78.70.

For sentiment analysis, Madan and Arpita, 2018 [19] used a code-mixed Hindi-English dataset developed by Aditya Joshi et al., 2016 for sentiment analysis. The dataset is preprocessed by removing punctuation, stop-words, and lowercasing the sentences. Word unigrams, word bigrams, and character trigrams were extracted as features. Machine learning classifiers, SVM and MNB, were trained using word unigram and word bigram features. Character trigrams were used as a sub-word feature for training the LSTM network. The authors developed a model by collaborating (Ensemble) machine learning

model (MNB) and a deep learning (LSTM) model. MNB is trained using word n-grams and for training LSTM, character trigrams were used. The outputs of both these models were used to categorise the sentences into positive, negative, and neutral classes. The ensemble approach's results were compared to SentiWordNet [66], Vowel-Consonant [18], and Sub-word composition [13] models. The result of the ensemble approach is the best among all and gives an F1-core of 0.661.

Aditya Bohra et al., 2018 [20] constructed the dataset containing tweets in Hindi-English code mixed. Tweets are collected by picking hashtags and keywords that are inclined towards hate speech using the Twitter Python API. After collection, the process of cleaning is applied by removing the timestamp, URL, text, user, re-tweets, replies, full name, id, and likes. Later on, tweets in pure English or pure Hindi language were removed. Due to this thorough processing; they got the dataset with 4575 code-mixed tweets. In the next step, annotations are being done at two levels: word level and hate speech or normal speech. At the word level, annotation is done as per language, i.e., three tags were applied: "eng", "hin" and "other" as per the words in the tweets. In the hate speech or normal speech annotation, 1661 tweets are annotated as hate speech and 2914 tweets are annotated as normal speech. The process of hate speech or normal speech annotation is done by two human annotators. The quality of annotation is calculated using inter-annotator agreement (IAA). For extracting the features, they used five different ways, such as: character n-grams, word n-grams, punctuation, negation words, and Lexicon. For character n-grams and word n-grams, they applied the n-gram range, varying from 1 to 3. In the case of punctuation marks and negation words, the same strategy is used. For punctuation, the number of times punctuation occurs in a sentence is counted, and for negation words, the number of times a negation word occurs in a sentence is counted, and thus both are considered as separate features. Negation words are taken from Christopher Pott's sentiment tutorial. In the Lexicon-based feature, 177 Hindi and English hate words were detected and used as a feature. On these extracted features, two supervised machine learning approaches are implemented as: SVM and RFC. The extracted features are huge in number, so chi-square feature selection was implemented, and it reduced the size to 1200 vectors. SVM with character n-gram and with five features altogether, almost produces the same accuracy of 71.6. RFC gives the highest accuracy of 69.9 with word n-grams as compared to other features.

Shalini et al., 2018 [21] created a code-mixed Kannada-English dataset using Facebook comments. They used the Graph API to crawl comments and then annotate them into positive, negative, and neutral sentiments. Alongside authors used the SAIL 2017 dataset for sentiment analysis. The dataset is divided into train (80%) and test (20%) sets and experimented with different models such as: SVM trained using Doc2Vec, Fasttext, CNN, and Bi-LSTM. All the models were tuned using hyper parameters and the results were measured in terms of accuracy. For the Hindi-English and Bengali-English datasets, the Bi-LSTM model produced the highest accuracy of 60.20% and 72.20%, respectively, whereas for the Kannada-English dataset, the CNN model got the highest accuracy score of 71.50%.

Kamble and Joshi, 2018 [22] use the dataset created by Aditya Bohra et al. They downloaded 3849 tweets using the Twitter API, of which 1436 were hateful. The same feature extraction techniques and machine learning algorithms were implemented and the results are compared. In another way, the team created domain-specific word embeddings. Domain-specific word embeddings represent the semantics of hate speech. For creating the domain-specific word embeddings, the Twitter API is used to search for tweets having Hindi profane words. It creates a dataset with 255,309 tweets; using this tweet dataset, word-embeddings are trained. For training, the word-embedding gensim library is used. Also, the average quantity of Hindi words in a tweet is counted using the Google

Translate API. Three different deep learning models were used and hyperparameters were tuned on these domain-specific word-embeddings: CNN-1D, LSTM, and BiLSTM. For the CNN-1D model, filter size 3 is used, and convolve over the embedding creates feature maps. Following this, a globalMaxPooling layer with a droupot of 0.5 and sigmoid as an activation function is applied, which results in a single feature vector. The second model LSTM feeds with word-embeddings, resulting in the returned sequences that are further inputted to the globalMaxPooling layer, which is passed to the output layer with a sigmoid activation function. As like LSTM, BiLSTM is also feed with word-embeddigns, but the sequences are processed in both directions and the results of both the directions are combined together which are further passed to globalMaxPooling layer, and finally passed to output layer with sigmoid activation function. After comparing the results, it is found that to capture the sentiments, domain-specific word-embeddings help and improve the results sustainably.

Mathur et al., 2018 [23] designed a Multi-Input Multi-Channel Transfer Learning-based Model (MIMCT) to detect offensive tweets in the Hinglish language. The authors used an English offensive tweet dataset collected from CrowdFlower, which is used to pre-train the MIMCT model. They also designed Hinglish offensive tweets collected using the Twitter Streaming API. The collected tweets are selected from the Indian subcontinent and manually annotated. In the preprocessing of URLs, punctuation, user mentions, and numbers were removed. Hash tags and emoticons were converted into their text. Words which provide less information were removed using NLTK's stopwords corpus. The text is lowercased followed by transliteration and translation into English words using the Hinglish dictionary. Thus, the tweets from both the datasets were distributed into English Offensive (EOT) and Hinglish Offensive (HOT) and tweets were labelled as non-offensive, abusive, and hate-inducing. For experimentation purposes, an 80:20 train-test split was applied. For baseline, SVM and RF machine learning models were implemented by tuning their hyperparameters. For training machine learning classifiers, features were extracted using character n-grams, bag of words, and TF-IDF countvector. CNN and LSTM transfer learning models were trained using 10-fold cross validation by identifying the best hyper-parameters. The MIMCT model is a combination of CNN and LSTM, which is trained using different word embeddings such as Glove, word2vec, FastText and their concatenation. In the baseline, SVM and RFC were implemented with character n-grams, TF-IDF and a bag of words. The results of the MIMCT model were compared with baseline models and found MIMCT achieved good results.

Santosh and Aravind, 2019 [24] experimented sentiment analysis on Hindi-English using dataset created by Aditya Bohra et al., 2018 They downloaded the dataset containing 3800 tweets, of which 2300 are labelled as hate and 1500 are labelled as non-hate tweets. In the preprocessing stage, they remove the '#' and separate the token. Further URLs, user-mentions, stop words, emoticons, and punctuation were also removed. The same baseline is followed and implemented, which is used by Aditya Bohra and his team. In addition to the baseline, phonemic sub-words were used as a feature to implement the sub-word level LSTM model and the hierarchical LSTM model. Experiments are performed using 10-Fold Cross Validation and accuracy is measured using, recall and F1-scores. The extracted features using baseline are huge in quantity, hence the chi-square feature selection algorithm was used, which resulted in a feature size of 1200. For machine learning classifiers, the Scikit-learn library and for deep learning, Keras, are used. For training the deep learning models, the Adam optimizer with a batch size of 32 is used. In the sub-word level LSTM model, to get the feature map, the sub-word representations are submitted to the CNN-1D layer with a filter and a bias. Then sub-word representations are submitted to the LSTM layer, so that the model can

distinguish between hate and non-hate tweets. To build a hierarchical attention-based LSTM model with phonemic sub-words, the words are segmented into phonemic sub-words. The segmentation is done using consonant-vowel sequences. The hierarchical attention-based model has an embedding layer, syllable encoder, word encoder, word attention layer, and output layer. In baseline, character n-grams, word n-grams, negation words, and punctuation mark features were trained using SVM and RFC. In the results, the SVM overshadows the RFC performance. Though the hierarchical LSTM model with attention based on phonemic sub-words produced less accuracy, it gave good recall and an F1-score by a huge margin.

K Sreelakshmi et al., 2019 [25], performed hate speech detection on code-mixed Hindi-English language by using three different datasets. The datasets used are Bohra et al., 2018, Mathur et al., 2018 and the dataset from shared task HASOC. Mathur et al., dataset has three labels, but for the task, hate-inducing classes are labelled as hate, and non-offensive as non-hate. Thus, the final dataset has hate and non-hate categories, each containing 5000 records. The dataset is preprocessed by removing URL's, usernames, hashtags, special characters, etc. and further, the dataset is used for training the pre-trained models- fastText and domain-specific word embedding. The team performed experiments in three different ways. In the first experiment, machine learning classifiers SVM and Random Forest were trained using features extracted using the CBOW of Doc2vec. In a second way, domain-specific embedding using word2vec was used for feature extraction and trained using the same set of machine learning classifiers. From both the experiments, word2vec outshines Doc2vec, but both the feature extraction techniques are not able to tackle words that are out of vocabulary. So they used the FastText pre-trained model and trained the same set of machine learning classifiers in the third experiment. The results of the third experiment are better as compared to the first two experiments.

For data collection, Anita Saroj and Sukomal Pal, 2020 [26] used the parliamentary election (PEI) event conducted in 2019. They collected social media posts from Facebook and Twitter in Hindi and English languages and some of them were code-mixed. For Twitter, they used different hashtags and for Facebook they used the Facepager tool and collected more than 10,000 posts. Out of these posts, only 20% were related to hate speech and offensive content. To detect whether the post is offensive or not, they created three tasks: Task A, Task B, and Task C. In Task A, posts are classified into Hate and Offensive (HOF) and Non-Hate, or offensive (NOT). Task B classifies the Hate (HOF) posts of TASK A into 3 ways, i.e., Hate speech content (HATE), Offensive (OFFN), and non-hate or non-offensive (NONE). Lastly, Task C checks the type of hate and offensive (HOF) from Task A and classifies them into Targeted Insult (TIN), Untargeted (UNT), and Non-Hate or Non-offensive (NONE). Annotations of posts into the given categories are done by undergraduate students. For Tasks A, B, and C, the average score is calculated using Cohen's Kappa. They preprocessed tweets using the tweet preprocessing library by removing the Retweets(RT), hashtags, URLs, Twitter Mentions, emojis, and Smileys. Tweets were tokenized, stemmed, and stop-words were removed. For extracting the features, language independent TF-IDF is applied for both languages. Machine classifiers such as Multinomial Naive-Bayes (MNB), Stochastic Gradient Descent (SGD), Linear Support Vector Machine (Linear SVM), and Linear Regression (LR) were applied to the extracted features. While conducting the experiments, they used other datasets developed by Davidson [68] and another dataset by the FIRE 2019 HASOC track [67]. They applied the methodology to these two datasets and compared the results. The results are measured in terms of precision, recall, F1-Score, and accuracy. In the results, they found that their dataset performs better because it concerns a specific domain.

In the year 2020, the Forum for Information Retrieval Evaluation (FIRE) runs several shared tasks, among which two tasks are related to code-mixing, named as: HASOC-Offensive Language Identification-DravidianCodeMix [27] and Sentiment Analysis of Dravidian Languages in Code-Mixed Text [28]. The details about the tasks and techniques used by the participants are elaborated below.

HASOC-Offensive Language Identification-DravidianCodeMix [27]: The goal of the task is to recognize offensive language from a code-mixed dataset of Malayalam-English and Tamil-English. The task is further divided into Task 1 and Task 2. Task 1 has code-mixed Malayalam comments collected from YouTube, and participants have to classify the comments as offensive or not-offensive. The YouTube comment scrapper is used to collect the comments and was collected from movie trailers in 2019. The dataset contains all types of code-mixing. The dataset has 3200 training data containing 2633 not-offensive and 567 offensive comments, and 400 development data containing 328 not-offensive and 72 offensive comments, thus representing class imbalance. Task 2 has Tanglish (Tamil-English) and Manglish (Malayalam-English) comments, and participants have to classify them as offensive or not-offensive. The comments are in Latin characters only and are annotated as offensive (OFF) or not-offensive (NOT). For collecting Tanglish comments, YouTube and the Helo app are used, whereas for Manglish comments, only YouTube is used. Training data is provided with 4000 comments for both languages. For the Manglish language, 2047 not-offensive and 1953 offensive comments were provided, and for Tanglish, 2020 not-offensive and 1980 offensive comments were provided. The baseline system has an SVM classifier trained using TF-IDF features.

Experimental setup performed by Participants: For both the tasks, Siva Sai et al. [29] performed selective translation and transliteration methods to convert the romanized dataset into its native language. On the transliterated dataset, transformer networks like XLM-RoBERTa and Multilingual BERT were applied. CUSATNLP [30] used a one-hot encoding vector and a paragraph vector for representing the Malayalam dataset. On these n-dimensional vectors, the LSTM network is applied. Varsha Pathak et al. [3] participated in task 2 and used different machine learning models such as MNB, SVC, LR, RFC, and ensemble model. These models were trained using character n-grams, word n-grams, and combining both character and word n-grams of different ranges for both the languages. Gaurav Arora [31] generated a code-mixed dataset as a Markov process using Markov chains and then implemented a pre-trained ULMFiT on it. He applied the same approach to both the tasks. For both the tasks, SSNCSE [32] used character n-gram, count vectorizer, and BERT models for feature extraction. On the basis of these extracted features, machine learning classifiers like RF and MLP were implemented. CENMates [33] implemented four different classifiers for both the tasks, of which LR and XGBOOST classifiers were trained using TF-IDF character n-gram, and the other two classifiers are long short-term memory networks and attention networks. NITP-AI-NLP [34] implemented a fine-tuned pre-trained BERT for Task 1 and two different models for Task 2. The first model has deep learning models like CNN and Bi-LSTM, while the second model has machine learning classifiers like SVM, LR, NB, RF, and DNN using TF-IDF character and word n-gram features. YUN [35] participated in both the tasks and designed a self-attention based on the BiLSTM and the sub-word representation learning. Zyy1510 [36] implemented an ensemble approach consisting of basic CNN, BiLSTM, and an LSTM layer + convolution layer for both the tasks. Ajees [37] used three classifiers: MLP, CNN-BiLSTM, and BiLSTM for both the tasks. They used CountVectorizer and BERT word embedding techniques to convert the text into features. CFILT IIT Bombay [38] designed an ensemble of multilingual BERT models for both tasks and developed a novel training strategy comprising data augmentation using random transliteration. WLV-RIT [39] used two models: the first model has traditional machine

learning classifiers like MNB, SVM, and RF, and the second model uses transformer models like XLMRoberta and Multilingual-BERT. IIITG-ADBU [40] used the XLM-RoBERTa model and SVM classifier for both the tasks. A SVM was trained using character n-grams of range (1, 6) and word n-grams of range (1, 3) and a combination of both.

Sentiment Analysis of Dravidian Languages in Code-Mixed Text [28]: Organizers created code-mixed dataset for Tamil and Malayalam languages. Dataset is created from social media platform: YouTube. YouTube comments were collected using YouTube Comment Scrapper. 184753 Tamil sentences were collected regarding 2019 Tamil movie trailer and 116711 Malayalam sentences were collected regarding keyword: Malayalam movie 2019. Among these statements many of the non-code-mixed statements were filtered out using language identification library: langdetect. Pre-processing is also done for both languages which apply a sentence length-filter and removing emoticons. For length-filtering sentence less than 5 words and more than 15 words were removed. Thus 15,744 code-mixed Tamil-English (organizers refereed as Tanglish) and 6739 code-mixed Malayalam-English (organizers refereed as Manglish) sentences were collected. Each sentence is annotated by atleast 3 annotators into five categories: Positive, Negative, Neutral, Mixed feeling and other language. Finally for Tamil language the corpus of 15,744 sentences were randomly shuffled and categorized into training, validation and test set containing 11335, 1260, 3149 sentences respectively. For Malayalam language, the corpus of 6739 sentences were selected and categorized into training, validation and test set containing 4717, 674, 1348 sentences respectively. The task for participants is to develop system which classifies the comments into the five mentioned categories. After creating both the datasets, for benchmark system organisers used traditional TF-IDF for feature extraction and machine learning approaches like : Logistic regression (LR), Support vector machine (SVM), Decision tree (DT), Random Forest (RF), Multinomial Naive Bayes (MNB), K-nearest neighbours (KNN) on code-mixed Malayalam-English dataset. Along with this pre-trained word embeddings like: Word2Vec and FastText were used with different deep learning models viz. Dynamic Meta-Embeddings (DME), Contextualized DME (CDME), 1D Dimensional Convolution (1DConv), Bidirectional Encoder Representations for Transformers (BERT) were also implemented. On the Tamil-English code-mixed dataset the same experimentation, but the only change is for deep learning instead of BERT they applied MultilingualBERT (mBERT). Total 32 participants for Tamil and 28 for Malayalam participated. The evaluation is done weighted average F1 score.

Experimental setup performed by Participants: JUNLP [41] designed LSTMs and bi-directional LSTMs with and without language tagging, but finally the results of bi-directional LSTMs with language tagging were considered as the final model to be categorised as Tamil comments. MUCS [42] employed three distinct features, including n-grams, word vectors (word2vec), and sub-word vectors**.** For training, a voting classifier is an ensemble classification model that works based on majority voting, including MLP, MNB, and BiLSTM. They were trained using n-grams, word vectors (word2vec), and sub-word vectors, respectively. bits2020 [43] used a sub-word level representation of the dataset and implemented an LSTM network on both datasets. CMSAOne [44] used a combination of fastText, ELMO, and TF-IDF for meta-embeddings. On these meta-embeddings transformer and GRU model is implemented. HIT_SUN [45] used the BERT model implemented in two parts: pretraining and fine-tuning. JudithJeyafreeda [46] used different machine learning models, viz. SVM, LR, NB, and RFC, trained using TF-IDF features. SSN_NLP_MLRG [47] designed language model using AWD-LSTM model with ULMFiT framework using the FastAi library for Malayalam-English and Tamil-English comments. IRLab [48] created three different word embeddings using BERT,

DistilBERT, and fastText, which were trained using multinomial logistic regression. PITS [49] implemented TF-IDF character and word n-gram range and were trained using logistic regression. SRJ [50] used the XLM-Roberta model for sentiment analysis of both datasets. SNCSE_NLP [51] used different feature extraction techniques such as TF, TFIDF, BERT, and fastText. These extracted features were trained using different machine learning classifiers: MLP, NB, and LR. Along with these, Bi-LSTM is also trained using one-hot encoded vectors. YUN [52] designed a model based on the multi-language model XLM-RoBERTa and used the K-folding method for both the languages. CIA_NITT [53] worked on Manglish comments by implementing Sentence BERT, Sentence BERT with Manglish features, and Sentence BERT with Class Balanced Loss; out of these, SBERT with CBL shows better results. LucasHub [54] implemented **a** model combining fine-tuned m-BERT and fine-tuned XLMRoBERTa for categorising Malayalam-English and Tamil-English comments. NITP-AI-NLP [55] used one-hot vectors to get character and word embedding. The first model was built using two CNN networks and was trained using character embedding and word embedding. The second model has CNN trained using character embedding and Bi-LSTM trained using word embedding. SA-SVG [56] tokenized the Tamil dataset and used a Bi-LSTM model to train and to categorize the sentiments. After preprocessing the dataset, IRLab [57] used a pretrained BERT-based model for classification. Five models are used by UMSNH-INFOTEC [58]**.** Those models are language-independent (μTC), Byte-Pair embeddings, language-specific Byte-pair embeddings, character embeddings, and linear combinations TADS [59] used three machine learning models: SVM, LR, and a Perceptron trained using vectors for classification purposes. Parameswari_faith_nagaraju [60] extracted TF-IDF features trained using MNB for both the datasets. YUN111 [61] implemented an mBERT-based model for sentiment categorizing. Theedhum Nandrum [62] trained SGD and LSTM classifiers on various features such as emoji sentiment, language tags, word vectors, and document length. NUIG-Shubhanker [63] implemented an auto-regressive XLNet model for sentiment analysis**.**

Priya et al., 2020 [64] studied hate speech detection on code-mixed Hindi-English using three different datasets. The first dataset was designed by Bohra et al., 2018 and collected from the Github repository. The second dataset is available from a Shared Task named HASOC, organised at FIRE 2019 (Mandl et al., 2019). The third dataset designed by Kumar et al., 2018 is also considered for the training and testing system. The first two datasets were already annotated; hence, the third dataset was annotated into hate and non-hate categories. Annotation is made in two phases, and the validity of annotation is checked using interannotator agreement, which was calculated using Kripendorff's alpha. The experiments were conducted using machine learning and deep learning classifiers. Different machine learning classifiers like support-vector machine (SVM), K-Nearest Neighbours (KNN), multinomial naive Bayes (MNB) and decision tree (DT) were trained using Term Frequency (TF) features. For deep learning, a character-based convolution neural network (CNN) is used. Among the applied classifiers, character-based CNN performed best. The results were compared with Bohra et al.2018, and it was found that character-based CNN and Bohra et al.2018 SVM produced the same accuracy.

Varma et al., 2021 [65] introduced a new dataset in Code-Mixed Telugu-English Text (CMTET). To collect data, they used the Twitter API and YouTube Comments API in the sports and movie domains. From collected dataset URLs, markup text and comments of less than five words were removed. The cleaned dataset is annotated into positive, negative, and neutral sentiments, and word-level annotation is also done using language tags. After word-level annotation, comments in only English or only Telugu were also removed. Annotation is carried out by five Telugu native speakers using the Telegram Bot API. An Inter Annotator Agreement score is calculated using Cohen's Kappa score. The

authors found challenges in the dataset, such as informal transliterations, informal language, and spelling and typing errors, so they applied an unsupervised data normalization technique. In that elongation normalization, normalization of English and Telugu words was done. After normalization different classifiers like Logistic Regression (LR), Naive Bayes (NB), Support Vector Machine (SVM), Random Forest (RF), and Multi-Layer Perceptron (MLP) were trained using N-Grams and TF-IDF features. All classifiers were trained using normalized data and without normalized data and authors found that the performance of every classifier was improved using normalized data. Among all MLPs, one performs better with 80.22% accuracy.

3. Results & Discussion:

Table 2 shows the dataset created in code-mixed languages so far. The table contains, social media platform used for collecting data and preprocessing techniques applied to the collected data. It also includes information about the dataset's size, labels or classes into which it is classified, the annotation method used for labelling or classification, and the nature of the classes, such as balanced or imbalanced.

In Table 2, the datasets that were created by researchers are shown in a dark blue background, while the datasets that were created using existing datasets are shown in a white background.

From Table 2, we can see that researchers frequently used English as the second language with their primary language, and for collecting raw data in code-mixed languages, different social media applications and online sites were used. Before annotating, almost all researchers had done exhaustive data preprocessing. They removed noisy information such as usernames, URLs, hashtags, special characters, English stopwords, and incomplete sentences other than code-mixed sentences. They also select sentences of a specific length so as to maintain consistency in data. For annotating data, mostly manual annotators were used, and some of them used a combination of mechanical and manual annotation. The manual annotators selected had knowledge of the primary language for proper classification, and inter-annotator agreement with the annotated dataset was also calculated by most of the researchers. To handle the imbalanced classes, researchers used deep learning classifiers because of their capability to handle imbalanced classes, and to train machine learning classifiers, they used different over-sampling and under-sampling methods. The dataset is classified into different sentiments, such as: positive, negative, neutral, hate, non-hate, offensive, not-offensive, abusive, happiness, sadness, fear, anger, surprise, etc.

Table 2: Datasets in code-mixed languages

Year	Language Dataset	Social media platform for Dataset / Dataset details	Pre-processing on raw data	Dataset Size	class / labels	Annotators & Method for Inter-annotator agreement	Class balance / Class Imbalance
2014 [11]	Hindi-English	Facebook	Username were removed, but the names in the comments were kept.	6,983	Languages: Hindi, English	02 Linguist Annotated	Not Applicable

2015 [12]	English, Chinese, Both, and Mixed languages	Weibo	Advertisement and noise posts are deleted	1000	Happiness, Sadness, Fear, Anger, Surprise	02 persons Annotated Cohen's Kappa coefficient	---
2016 [13]	Hindi-English	Facebook	Roman script comments, English sentences, Comments having more than one sentence, comments over 50 words, eliminated.	3879	Positive, Negative, Neutral	02 persons Annotated. Cohen's Kappa coefficient	Classes are highly Imbalanced
2017 [14]	English-Bengali	Facebook	Punctuations and multiple character repetitions were removed, abbreviations expansion.	882 posts	Positive, Negative, Neutral	02 persons Annotated Kappa co-efficient	---
2017 [15]	Hindi-English and Bengali-English	Twitter4j API	Incomplete tweet, tweet not having Bengali or Hindi words and spam tweets were removed. Hashtags and URLs are kept.	Hindi-English Training :12936 Test: 5525 Training :2500 Test: 3038	Positive, Negative, Neutral	Manually annotated	---
2018 [18]	Bengali-English	Twitter4j API	Tweet having minimum length as 8 and minimum 5 Bengali words were kept, rest are omitted. spam tweets, incomplete tweets, tweets with conflict sentiments were removed	5000	Language tagging: BE/EN/EU Sentiment tagging : positive or negative or neutral	Manually annotated	---
2018 [20]	Hindi-English	Twitter	Removing URLs, Punctuations and replacing User Names and Emoticons	4575 tweets	Hate speech , Normal speech	02 persons Annotated Cohen's Kappa coefficient	Classes are Imbalanced but nothing is mentioned.
2018 [21]	Kannada-English	Facebook	Comments in native script and mixed script were removed	7005 comments	Positive, Negative, Neutral	Manually annotated	Classes are Imbalanced
2018 [23]	English & Hindi-English	English: Online site (CrowdFlower),	URLs, punctuations, user mentions and numbers were removed. Hash tags and emoticons	14509 3189	Non-offensive, Abusive, Hate-inducing	Manually Annotated Cohen's	Classes are not severely imbalanced

		Hindi-English : Twitter	were converted into its text. Text is lowercased and transliteration and translation is done.			Kappa coefficient	
2019 [25]	Hindi-English	Bohra et al.,2018, [20] Mathur et al.,2018, [23] shared task HASOC Dataset	Removing URL's, Usernames, Hashtags, emoticons, punctuation marks, unwanted characters, and extra white spaces. Text is lowercased.	10000	Hate, non-hate	Manually Annotated	No Class Imbalanced
2020 [26]	Hindi, English and Hindi-English	Facebook and Twitter	Removing the RT, #, URLs, Twitter Mentions, Emoji's, stopwords and Smileys. Tweets were tokenized, Stemmed.	2000 tweets	Task A : HOF or NOT Task B : HOF into : HATE or OFFN or NOT Task C : HOF into TIN or UNT	03 students Annotated Cohen's Kappa coefficient	No Class Imbalance
2020 [27]	HASOC-Offensive Language Identification-Dravidian CodeMix Malayalam-English Tamil-English	YouTube & Helo App	*No details were mentioned.*	Task1: 3600 code-mixed Malyalam comments Task2: 4000 Manglish & 4000 Tanglish	Task1: offensive or not-offensive Task2: offensive or not-offensive	No information Available Krippendorff's alpha	Task1 has imbalance dataset of Malayalam-English
[28]	Sentiment Analysis of Dravidian Languages in Code-Mixed Text Malayala	YouTube	Comments other than code-mixed were totally removed Comments with less than 5 and greater than 15 words were removed. Emoticons, emoji's were removed	6739 comment 15744 comment	Positive, Negative, Neutral, Mixed feeling and other language	Manually Annotated Krippendorff's alpha	Classes were highly imbalanced.

	m-English Tamil-English						
2021 [65]	Telugu-English	Twitter & Facebook	URLs, markup text and comments less than five words were removed	19,857 sentence	Positive, Negative, Neutral	Manually annotated using Telegram Bot API Cohen's Kappa coefficient	Positive and Negative classes have near about equal distribution but neutral sentences are less in comparison

Table 3 shows the other factors of the research. It includes further processing techniques applied to the dataset in column 2. The columns 3 and 4 mention feature extraction techniques and classifiers applied, respectively. Finally, the results obtained were stated. While declaring the results, the best-performing classifiers were given along with their accuracy score, or F1-score, in the round parenthesis. In the case of shared tasks, teams' results were mentioned along with their rank. Likewise, in Table 2, the same colour scheme is implemented in Table 3 for distinguishing between the datasets.

Table 3: Showing Factors - Feature Extraction, Classifiers and Results

Year / Team name	**Further processing on Dataset**	**Feature Extraction techniques**	**Classifiers / Models**	**Results**
2015 [12]	----	Unigrams for all words, Unigrams for Chinese, Unigrams for English, Combine results of Chinese and English	Maximum Entropy	Combine results of Chinese and English classifiers gives better performance
2016 [13]	----	Character-level features	Char-LSTM, Subword-LSTM, MNB, SVM	Subword-LSTM performs better than Char-LSTM, MNB and SVM
2017 [14]	----	Word based features, Syntactic features, Style based features	Machine learning algorithms, Artificial neural network	Artificial neural network model performed best. Word based and syntactic features yield the best results.
2017 [15]	----	*No details were mentioned*	Random baseline system is implemented.	More instances in Hindi-English than Bengali-English dataset, which directly influence the result.
2018 [18]	----	Word N-Grams, Negation words, Tagged words, Tagged phrase, Tagged acronyms, SentiWordNet, SOCAL	GNB, BNB, MNB, LRC and SGDC	SGDC got the best F1-Score of 78.70

		lexicon and NRC Emotion Lexicon.		
2018 [19]	Used dataset created by Aditya Joshi et al., 2016. Punctuations and stop words were removed. Sentences were lowercased.	Word unigram, word bigram, character trigrams.	SVM, MNB, LSTM, Ensemble approach.	In Ensemble approach, MNB model assist to overcome shortcomings of LSTM model.
2018 [20]	----	character n-grams, word n-grams, Punctuation, negation words and Lexicon	SVM and RFC	SVM with character n-gram and with other features performed well.
2018 [21]	----	Doc2Vec	SVM trained using Doc2Vec, Fasttext, CNN and Bi-LSTM	For Hindi-English and Bengali-English dataset Bi-LSTM and for Kannada-English CNN performed well
2018 [22]	Used dataset created by Aditya Bohra et al.,2018 Downloaded 255,309 tweets having Hindi cuss words	Downloaded tweets were used to train word embeddings using gensim library.	CNN-1D, LSTM and BiLSTM	Domain-specific word-embeddings helps in capturing the sentiments and improve the results.
2018 [23]	----	Glove, word2vec, FastText and their combination.	CNN, LSTM and Ensemble of CNN and LSTM	Ensemble along with combine embeddings gives better results than baseline.
2019 [24]	Used dataset created by Aditya Bohra et al.,2018 Remove the '#', URLs, user-mentions, stop words, emoticons and punctuations. Tokens were Separated.	phonemic sub-words	Sub-word level LSTM model Hierarchical LSTM model implemented on phonemic sub-words.	In all the models SVM performed better.
2019 [25]	----	FastText , word2vec, Doc2vec	SVM, RF	Machine learning classifiers trained sing FastText performs better.
[26]	----	TF-IDF	MNB, SGD, Linear SVM and LR. Also applied on datasets developed by Davidson et al. [68] and FIRE 2019 HASOC track dataset [67]	Dataset (PEI-2019) performed better because it concerns with specific domain. SGD performed well,
2020 [27]	----	TF-IDF	SVM	

siva sai et al.,[29]	Removing URLs, emojis from, special characters, numbers, user mentions and punctuation. Task2: Lowercasing Manglish & Tanglish comments	XLM-RoBERTa	XLM-RoBERTa, mBERT, Ensembling of XLM-RoBERTa, XLM-RoBERTa base + XLM-RoBERTa large and XLM-RoBERTa base + mBERT	For task1 XLMR-B and XLMR-B + mBERT, In task 2 for Tanglish XLMR-B + mBERT and for Manglish XLMR-B, XLMR-B + mBERT, XLMR-B + XLMR-L performed well. For Task1: got 1st rank (0.95). For Task 2: Manglish got 2nd (0.77) and for Tanglish got 1st (0.90) rank.
CUSATNLP [30]	URLs, hash in hashtags, repeated characters, unwanted numbers, usernames were removed. Lowercasing and tokenization is done.	One-hot vector and paragraph vectors for text representation	LSTM	Model got F-Score of 0.54 for Manglish Subtask and got 10th rank.
Varsha pathak et al., [3]	Unnecessary, stop words, white spaces, digits, special characters, extra spaces, @USER, @RT and TAG etc. are eliminated.	TF-IDF, Custom WordEmbedding	SVC, MNB, LR, AdaBoost, DTC and RF. Neural Network	For Manglish MNB and for Tanglish SVC performed well with TF-IDF and got 2nd (0.77) and 3rd (0.87) place respectively
Gaurav arora [31]	Removing @username mentions etc. and lower-casing comments.	Generated Markov process using Markov chains	ULMFiT	ULMFiT model for Task 1 got 3rd rank (0.91), in Task 2, for Manglish 5th (0.74) and for Tanglish got 2nd rank (0.88).
SSNCSE [32]	No pre-processing details were mentioned.	char n-gram, TFIDF and fine-tuned BERT	MLP, RF and NB	For both tasks RF trained using TF-IDF performed better than others. For Task 1 got 2rd rank (0.94). In Task 2, for Manglish 4th (0.75) and for Tanglish got 2nd rank (0.88).
CENMates [33]	Punctuation, emojis and special characters were removed. Tokenzing and lowercasing comments.	TFIDF	LR, XGBoost, LSTM and Attention with LSTM	TF-IDF with character level n-gram performed well with machine learning classifiers and for Task 1 got 2rd rank (0.93), in Task 2, for Manglish 1st (0.78) and for Tanglish got 4th rank (0.86).
NITP-AI-NLP [34]	Removing letter word and punctuation. Translating &, @ and numbers (1-10) into and, at and English numbers resp. Lowercasing comments.	TF-IDF	CNN and Bi-LSTM SVM, LR, NB, RF and DNN	LR for Manglish and Dense Neural Network for Tanglish performed well with character n-gram. For Task 1 got 3rd rank (0.93). In Task 2, for Manglish 7th (0.69) and for Tanglish got 6th rank (0.84).

YUN[35]	Removing emotional symbols, @username and so on. Lowercasing the comments	sub-word representation	BiLSTM	Model for Task 1 got 3rd rank (0.93), for Task 2, for Manglish 9th (0.67) and for Tanglish got 5th rank (0.85).
Zyy1510 [36]	Noise like usernames, emoticon, hashtags were removed. Transliteration is performed.	CNN layer for local features and maximum pool layer is used to extract the essential features	Ensemble of basic CNN, BiLSTM and an LSTM layer + Convolution layer	Ensemble model for Task 1 got 3rd rank (0.93), for Task 2, for Manglish 3rd (0.87) and for Tanglish got 9th rank (0.67).
Ajees [37]	No pre-processing details were mentioned.	CountVectorizer & BERT	MLP, BiLSTM with BERT, CNN-BiLSTM	For Task 1 got 7th rank (0.44). In Task 2, for Manglish 8th (0.68) and for Tanglish got 7th rank (0.83).
CFILT IIT Bombay [38]	Removing @mention from, RT, URLs, digits and extra spaces.	---	Ensemble of multilingual BERT	Ensemble model for Task 1 got 2nd rank (0.94), in Task 2, for Manglish 6th (0.72) and for Tanglish got 4th rank (0.86).
WLV-RIT [39]	Removing punctuations, emojis and lemmatising the English words	Bag-of-words	MNB, SVM, RF, XLMRoberta and Multilingual-BERT	XLM-R with transfer learning in Task 1 got 5th rank (0.89).
IIITG-ADBU [40]	No pre-processing details were mentioned.	Character n-gram and word n-gram	XLM-RoBERTa model and SVM	SVM trained using TF-IDF character and word n-grams performed well with Manglish and for Task 1 got 1st (0.95) and for Task 2 got 3rd (0.76) rank. for Tanglish in Task 2 got 3rd rank (0.87) using XLMRoBERTa.
FIRE 2020 Malayalam -English [28]	----	TF-IDF and pre-trained word embeddings: Word2Vec and fastText	LR, SVM, DT, RF, MNB, KNN on TF-IIDF and on pre-trained word embeddings : DME, CDME, 1DConv and BERT	Machine learning classifiers successfully categorized the comments into all classes. FastText in combine with word2vec for DME and CDME offers local and global context.
FIRE 2020 Tamil-English [28]	----	TF-IDF and pre-trained word embeddings: Word2Vec and fastText	LR, SVM, DT, RF, MNB, KNN on TF-IIDF and on pre-trained word embeddings : DME, CDME, 1DConv and mBERT	Machine leaning and Deep learning classifiers not produced satisfying results because of nature of dataset
JUNLP [41]	Training and validation dataset were added.	Language tags using NLTK	Bidirectional LSTM with and without language tag, LSTM with and without language tag	Bi-Directional LSTM model with language tag features produced 0.58 f1-score

MUCS [42]	No pre-processing details were mentioned.	n-grams, word vectors using Word2Vec, and sub-words	Ensemble model of MLP, MNB and BiLSTM	Ensemble model got 4th rank (0.62) and 6th rank (0.68) in Tamil-English and Malayalam-English respectively
bits2020 [43]	Replace all emojis with their meanings in Malayalam & Tamil	sub-word level representation	LSTM	LSTM model got rank of 5th (0.61) and 12th rank (0.60) in Tamil-English and Malayalam-English respectively
CMSAOne [44]	No pre-processing details were mentioned.	fastText, ELMO and TF-IDF	Transformer and GRU model	Model got F1-score of 0.58 for Tamil-English and 0.66 for and Malayalam-English
HIT_SUN [45]	Train and development datasets were merged	---	pretrained BERT and fine-tuned BERT	BERT model got 2nd rank in the Malayalam-English and 4th in the Tamil-English
JudithJeyafreeda [46]	No pre-processing details were mentioned.	TF-IDF	SVM, LR, NB and RFC	NB classifier got highest F1-score of 0.54 for Tamil-English and 0.58 for and Malayalam-English among other classifiers.
SSN_NLP_MLRG [47]	Numerals, punctuation, were removed and replace the noisy strings	---	AWD-LSTM model with ULMFiT framework	F1-score of 0.60 for both languages using the AWD-LSTM model.
IRLab [48]	Removing continuous repeating characters in word, exclamation, punctuations, non-ASCII, emoticons, symbols, numbers, special characters	---	BERT, DistilBERT and fasttext	Fasttext outperforms BERT & DistilBERT. Got 8th (0.58) rank in Tamil-English & 11th (0.63) in Malayalam-English
PITS [49]	emojis & smiles removed using tweetpreprocessor	TF-IDF	LR, DT, SVM, XGBoost, Catboost	LR showed best F1-score of 0.62 in Tamil-English & 0.71 in Malayalam-English amongst all.
SRJ [50]	No cleaning of text	---	XLM-Roberta	XLM-Roberta got F1-score of 0.65 in Tamil-English & 0.74 in Malayalam-English
SSNCSE_NLP [51]	No pre-processing details were mentioned.	TF, TFIDF, BERT, fastText, one-hot embedding /vector	LR, MLP, NB, RFC, BiLSTM	MLP with char-count(2,3) vectorizer got F1-score of 0.61 in Tamil-English & LR with TF-IDF char n-gram(1,5) got 0.71 in Malayalam-English
YUN [52]	No cleaning of text	---	XLM-RoBERTa,	XLM-RoBERTa, got rank of 3rd (0.63) and 1st rank (0.74) in Tamil-English and Malayalam-English respectively
CIA_NITT [53]	Removing special characters & repeating characters, lowercasing text, replacing emojis with meanings	Manglish sentiment words collected from YouTube	Sentence BERT	Sentence BERT achieved Rank of 4th (0.71) in Malayalam-English.

LucasHub [54]	Symbols, numbers and emoticons were translated into its meaning	---	XLM-RoBERTa and m-BERT	3rd (0.63) and 2nd rank (0.73) in Tamil-English and Malayalam-English respectively
NITP-AI-NLP [55]	Removed multiple spaces, Symbols, numbers translated into its English word	one-hot embedding /vector	CNN, Bi-LSTM and their hybrid approaches	Hybrid CNN-CN xnetwork got F1-score of 0.61 in Tamil-English & 0.69 in Malayalam-English
SA-SVG [56]	Removed special symbols	Tokenizing dataset	Bi-LSTM	Bi-LSTM model got rank of 14th (0.10) in Tamil-English.
IRLab [57]	Removed multiple spaces & punctuation symbols. Symbols, numbers and emoticons were translated into its meaning. Lowercasing and stemming words.	BERT	BERT-based pretrained model	BERT-based model got F1-score of 0.59 and 0.60 in Tamil-English and Malayalam-English respectively
UMSNH-INFOTEC [58]	---	---	language-independent (μTC), Byte-Pair embeddings, language-specific Byte-Pair embeddings, Character Embeddings and linear combination	For Malayalam-English linear combination and for Tamil-English language-independent (μTC) got 3rd and 6th rank respectively.
TADS [59]	No cleaning of text	CountVectorizer	SVM, LR, Perceptron	LR performed best among all classifiers.
Parameswari_faith_nagaraju [60]	Removed stop words, punctuations, numbers and non-unicode characters	TF-IDF	MNB	MNB got F1-score of 0.55 and 0.48 in Tamil-English and Malayalam-English respectively
YUN111 [61]	removed unwanted characters, emoticons	---	mBERT	mBERT got 2nd rank with F1-score of 0.64 & 0.73 in Tamil-English and Malayalam-English respectively
Theedhum Nandrum [62]	Spellings were normalized using Indic Library	Sentiments of emojis, soundex to harmonise speeling variants of same word, language tagging, document length rage	SGD, LSTM	SGD got 4th (0.62) and 9th rank (0.65) in Tamil-English and Malayalam-English respectively
NUIG-Shubhanker [63]	----	----	auto-regressive XLNet	Model achieves 0.49 & 0.35 accuracies and 0.52 & 0.32 F-scores on both the datasets.
2020 [64]	----	TF-IDF	SVM, KNN, MNB, DT and CNN	CNN performed best.

2021 [65]	----	N-Grams, TF-IDF	SVM, RF, LR, NB, MLP	Results of all classifiers were improved using normalized data.

We used the charts to quantify the use of feature extraction techniques and classifiers used by the researchers. It will help the researchers the classifiers and techniques frequently used. In total three charts were used. In Chart 1, the feature extraction techniques were split into pre-trained word embeddings and other techniques, and those were represented on the y-axis while the x-axis shows the number of times they were used. The ocean colour represents the pre-trained word embeddings BERT, FastText, ELMO, word2vec, Glove, and Doc2Vec, while the rest of the techniques are black in colour.

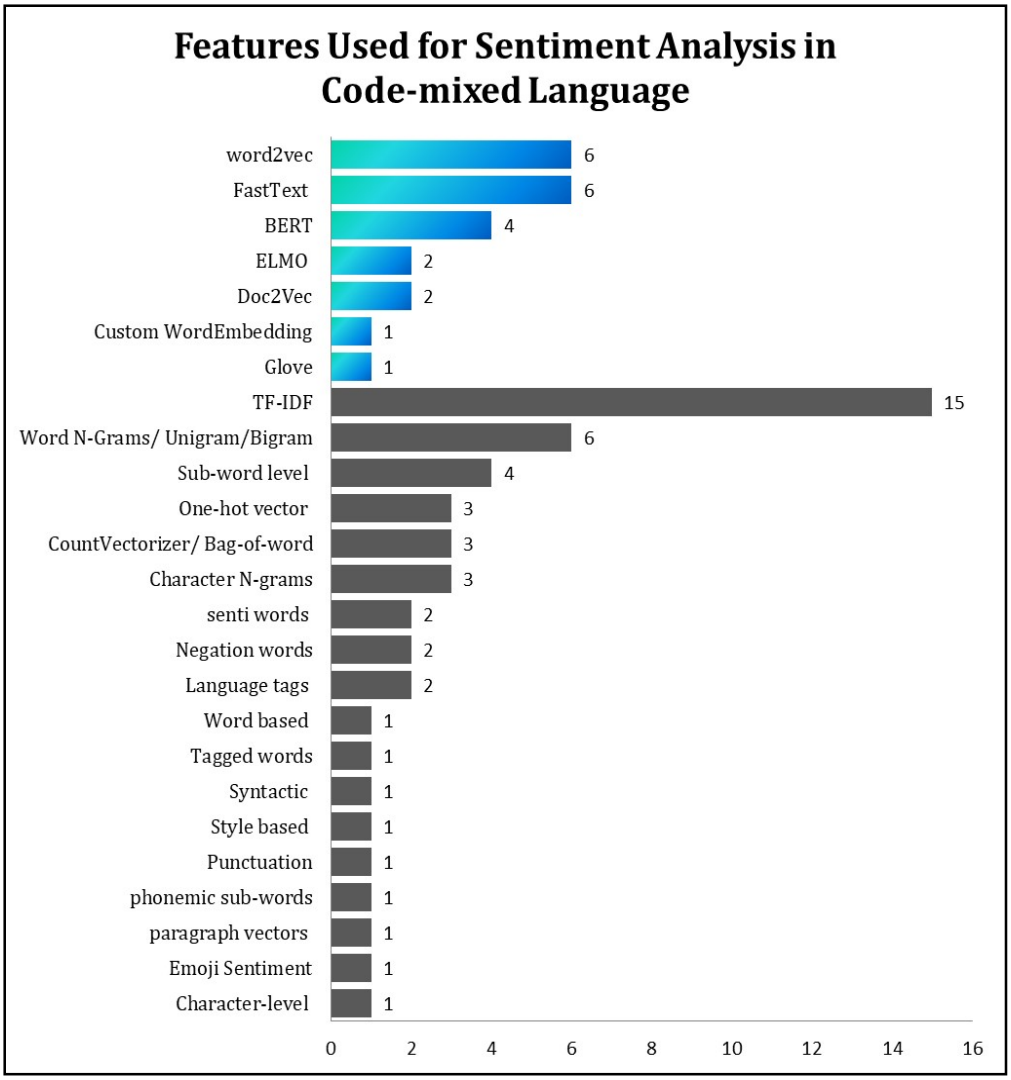


Chart 1: Usage of Various Features

The charts 2 and 3 represent the machine learning algorithms and the deep learning algorithms, respectively. The names of the algorithms appear on the x-axis of both charts, and the count in front of them indicates the number of times the algorithms have been

used so far. The transfer learning classifiers like BERT, DistilBERT, and XLMRoBERTa were considered under the deep learning classifiers.

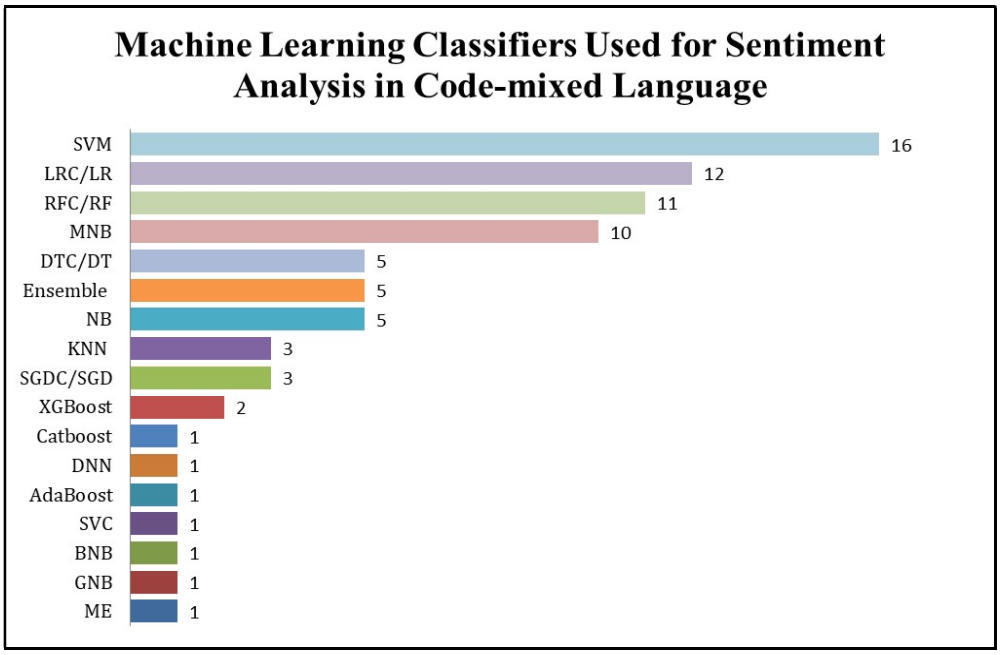


Chart 2: Usage of Machine Leaning classifiers

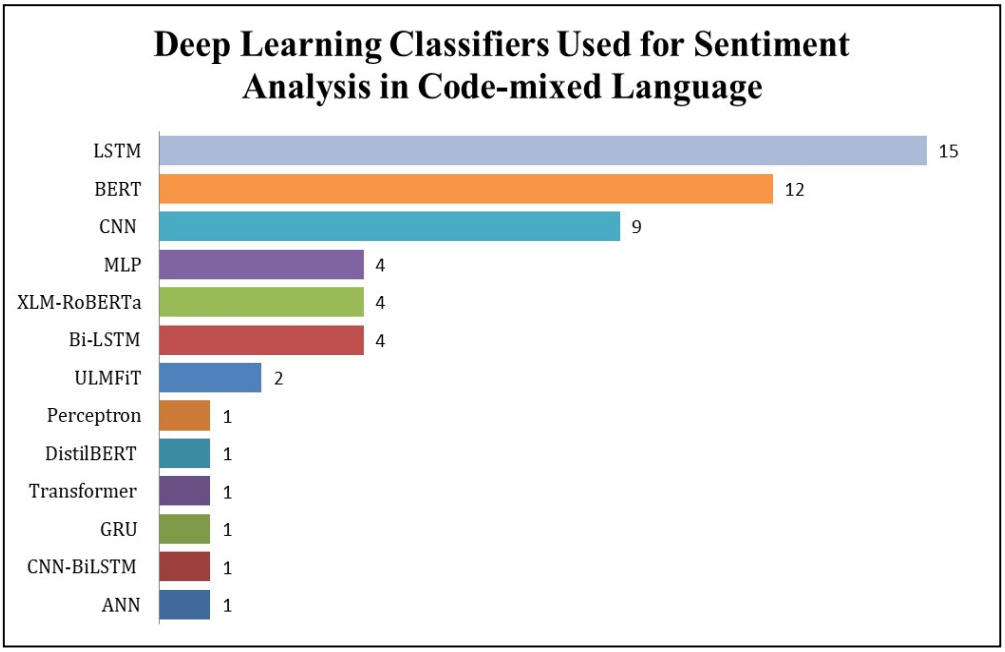


Chart 3: Usage of Machine Deep Leaning Classifiers

4. Conclusion:

The language independent feature extraction techniques like bag of words, TF-IDF, word n-grams, character n-grams, one-hot vectors, CountVectorizer, and Sub-word level were found useful for extracting the features, and other language dependent feature extraction approaches like negation words, senti words, language tags, and phonemic sub-words were supportive for handling sentiments in code-mixed Indian languages. Alongside the pre-trained word-emneddings like Word2Vec, Gensim, FastText, BERT were also useful. Among these, BERT, DistilBERT, XLMRoBERTa, etc., were trained on more than 100 languages, including many Indian languages. To develop the framework, traditional machine learning classifiers are still helpful, together with deep learning classifiers found helpful for classifying the sentiments into desired labels. In the case of machine learning classifiers, SVM, LR, RF, and MNB were mostly used, and in the case of deep learning classifiers, LSTM, BERT, and CNN were mostly used. The newly invented transfer learning classifiers like BERT, DistilBERT, XLM-RoBERTa, IndicBERT, etc. are proving their importance in sentiment detection.

A number of approaches were offered by different researchers for identifying hate speech from social media content. Most of the work is done and still going for detecting such hateful content in monolingual languages, but a limited amount of work is done in Indian code-mixed languages for detecting such hateful content considering the importance of the topic. Hence, more focus is needed in this area.

REFERENCES

[1] The global statistics. https://www.theglobalstatistics.com/india-social-media-statistics/

[2] Bharathi Raja Chakravarthi, Vigneshwaran Muralidaran, Ruba Priyad- harshini, and John P McCrae. Corpus creation for sentiment analysis in code-mixed tamil-english text. *arXiv preprint arXiv:2006.00206*, (2020).

[3] Varsha Pathak, Manish Joshi, Prasad Joshi, Monica Mundada, and Tanmay Joshi. Kbcnmujal@ hasoc-dravidian-codemix-fire2020: Using machine learning for detection of hate speech and offensive code-mixed social mediatext. *arXiv preprint arXiv:2102.09866*, (2021).

[4] VICE World News. https://www.vice.com/en/article/v7gq7j/controversial-facebook-post-on-prophet-muhammad-sparks-violent-riots

[5] Karsten Müller and Carlo Schwarz. Fanning the flames of hate: Social media and hate crime. *Journal of the European Economic Association*, 19(4):2131–2167, (2021).

[6] Fortify Rights. https://www.fortifyrights.org/mly-inv-rep-2018-07-19/.

[7] WIKIPEDIA, (2018) anti-Muslim riots in Sri Lanka. https://en.wikipedia.org/wiki/2018_anti-Muslim_riots_in_Sri_Lanka.

[8] House Hearing, 116 Congress, U.S. Government Publishing Office https://www.govinfo.gov/content/pkg/CHRG-116hhrg36563/html/CHRG-116hhrg36563.htm.

[9] Law Commission of India chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.latestlaws.com/wp-content/uploads/2017/10/Law-Commission-Report-No.-267-Hate-Speech.pdf

[10] Ramakrishnan Srinivasan and C. N. Subalalitha. Sentimental analysis from imbalanced code-mixed data using machine learning approaches. *Dis- tributed and Parallel Databases*, pages 1 – 16, 2021.

[11] Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 conference on empirical methods in natural lan- guage processing (EMNLP)*, pages 974–979, (2014).

[12] Sophia Lee and Zhongqing Wang. Emotion in code-switching texts: Corpus construction and analysis. In *Proceedings of the Eighth SIGHAN workshop on chinese language processing*, pages 91–99, (2015).

[13] Ameya Prabhu, Aditya Joshi, Manish Shrivastava, and Vasudeva Varma. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. *arXiv preprint arXiv:1611.00472*, (2016).

[14] Souvick Ghosh, Satanu Ghosh, and Dipankar Das. Sentiment identification in code-mixed social media text. *arXiv preprint arXiv:1707.01184*, (2017).

[15] Braja Gopal Patra, Dipankar Das, and Amitava Das. Sentiment analysis of code-mixed indian languages: An overview of sail code-mixed shared task@icon-2017. *arXiv preprint arXiv:1803.06745*, (2018).

[16] Pruthwik Mishra, Prathyusha Danda, and Pranav Dhakras. Code-mixed sentiment analysis using machine learning and neural network approaches. *arXiv preprint arXiv:1808.03299*, (2018).
[17] Kamal Sarkar. Ju ks@ sail codemixed-2017: Sentiment analysis for indian code mixed social media texts. *arXiv preprint arXiv:1802.05737*, (2018).
[18] Soumil Mandal, Sainik Kumar Mahata, and Dipankar Das. Preparing bengali-english code-mixed corpus for sentiment analysis of indian lan-guages. *arXiv preprint arXiv:1803.04000*, (2018).
[19] Madan Gopal Jhanwar and Arpita Das. An ensemble model for sentiment analysis of hindi-english code-mixed data. *arXiv preprint arXiv:1806.04450*, (2018).
[20] Aditya Bohra, Deepanshu Vijay, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. A dataset of hindi-english code-mixed social media text for hate speech detection. In *Proceedings of the second workshop on computational modeling of people's opinions, personality, and emotions in social media*, pages 36–41, (2018).
[21] K Shalini, HB Barathi Ganesh, M Anand Kumar, and KP Soman. Sen- timent analysis for code-mixed indian social media text with distributed representation. In *2018 International conference on advances in comput- ing, communications and informatics (ICACCI)*, pages 1126–1131. IEEE, (2018).
[22] Satyajit Kamble and Aditya Joshi. Hate speech detection from code- mixed hindi-english tweets using deep learning models. *arXiv preprint arXiv:1811.05145*, (2018).
[23] Puneet Mathur, Ramit Sawhney, Meghna Ayyar, and Rajiv Shah. Did you offend me? classification of offensive tweets in hinglish language. In *Proceedings of the 2nd workshop on abusive language online (ALW2)*, pages138–148, (2018).
[24] TYSS Santosh and KVS Aravind. Hate speech detection in hindi-english code-mixed social media text. In *Proceedings of the ACM India joint inter- national conference on data science and management of data*, pages 310– 313, (2019).
[25] K Sreelakshmi, B Premjith, and KP Soman. Detection of hate speech text in hindi-english code-mixed data. *Procedia Computer Science*, 171:737–744,(2020).
[26] Anita Saroj and Sukomal Pal. An indian language social media collection for hate and offensive speech. In *Proceedings of the Workshop on Resources and Techniques for User and Author Profiling in Abusive Language*, pages 2–8, (2020).
[27] Bharathi Raja Chakravarthi, Anand Kumar M, John P McCrae, B Pre- mjith, KP Soman, and Thomas Mandl. Overview of the track on hasoc- offensive language identification-dravidiancodemix. In *FIRE (Working Notes)*, pages 112–120, (2020).
[28] Bharathi Raja Chakravarthi, Ruba Priyadharshini, Vigneshwaran Mural- idaran, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John P McCrae. Dravidiancodemix: Sentiment analysis and offensive language identification dataset for dravidian languages in code-mixed text. *Language Resources and Evaluation*, pages 1–42, (2022).
[29] Siva Sai and Yashvardhan Sharma. Siva@ hasoc-dravidian-codemix-fire- 2020: Multilingual offensive speech detection in code-mixed and romanized text. In *FIRE (Working Notes)*, pages 336–343, (2020).
[30] Sara Renjit and Sumam Mary Idicula. Cusatnlp@ hasoc-dravidian- codemix-fire2020: Identifying offensive language from manglishtweets. *arXiv preprint arXiv:2010.08756*, (2020).
[31] Gaurav Arora. Gauravarora@ hasoc-dravidian-codemix-fire2020: pre- training ulmfit on synthetically generated code-mixed data for hate speech detection. *arXiv preprint arXiv:2010.02094*, (2020).
[32] Nitin Nikamanth Appiah Balaji and B Bharathi. Ssncse nlp@ hasoc- dravidian-codemix-fire2020: Offensive language identification on multilin- gual code mixing text. In *FIRE (Working Notes)*, pages 370–376, (2020).
[33] PV Veena, Praveena Ramanan, and Remmiya Devi G. Cenmates@ hasoc- dravidian-codemix-fire2020: Offensive language identification on code- mixed social media comments. In *FIRE (Working Notes)*, pages 377–383,(2020).
[34] Abhinav Kumar, Sunil Saumya, and Jyoti Prakash Singh. Nitp-ai-nlp@ hasoc-dravidian-codemix-fire2020: A machine learning approach to identify offensive languages from dravidian code-mixed text. In *FIRE (Working Notes)*, pages 384–390, (2020).
[35] Kunjie Dong and Yao Wang. Yun@ hasoc-dravidian-codemix-fire2020: A multi-component sentiment analysis model for offensive language identifi- cation. In *FIRE (Working Notes)*, pages 391–396, (2020).
[36] Yueying Zhu and Xiaobing Zhou. Zyy1510@ hasoc-dravidian-codemix- fire2020: An ensemble model for offensive language identification. In *FIRE (Working Notes)*, pages 397–403, (2020).
[37] AP Ajees. Ajees@ hasoc-dravidian-codemix-fire2020. In *FIRE (Working Notes)*, pages 404–410, (2020).
[38] Pankaj Singh and Pushpak Bhattacharyya. Cfilt iit bombay@ hasoc- dravidian-codemix fire 2020: Assisting ensemble of transformers with ran- dom transliteration. In *FIRE (Working Notes)*, pages 411–416, (2020).

[39] Tharindu Ranasinghe, Sarthak Gupte, Marcos Zampieri, and Ifeoma Nwogu. Wlv-rit at hasoc-dravidian-codemix-fire2020: Offensive lan- guage identification in code-switched youtube comments. *arXiv preprint arXiv:2011.00559*, (2020).
[40] Arup Baruah, Kaushik Amar Das, Ferdous Ahmed Barbhuiya, and Kun- tal Dey. Iiitg-adbu@ hasoc-dravidian-codemix-fire2020: Offensive content detection in code-mixed dravidian text. *arXiv preprint arXiv:2107.14336*, (2021).
[41] Sainik Kumar Mahata, Dipankar Das, and Sivaji Bandyopadhyay. Junlp@ dravidian-codemix-fire2020: Sentiment classification of code- mixed tweets using bi-directional rnn and language tags. *arXiv preprint arXiv:2010.10111*, (2020).
[42] Fazlourrahman Balouchzahi and HL Shashirekha. Mucs@ dravidian- codemix-fire2020: Saco-sentimentsanalysis for codemix text. In *FIRE (Working Notes)*, pages 495–502, (2020).
[43] Yashvardhan Sharma and Asrita Venkata Mandalam. Bits2020@ dravidian- codemix-fire2020: Sub-word level sentiment analysis of dravidian code mixed data. In *FIRE (Working Notes)*, pages 503–509, (2020).
[44] Suman Dowlagar and Radhika Mamidi. Cmsaone@ dravidian-codemix- fire2020: A meta embedding and transformer model for code-mixed sen- timent analysis on social media text. *arXiv preprint arXiv:2101.09004*, (2021).
[45] Huilin Sun, Jiaming Gao, and Fang Sun. Hit sun@ dravidian-codemix- fire2020: Sentiment analysis on multilingual code-mixing text base on bert. In *FIRE (Working Notes)*, pages 517–521, (2020).
[46] Judith Jeyafreeda Andrew. Judithjeyafreedaandrew@ dravidianlangtech- eacl2021: offensive language detection for dravidian code-mixed youtube comments. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 169–174, (2021).
[47] A Kalaivani and D Thenmozhi. Ssn nlp mlrg@ dravidian-codemix-fire2020: Sentiment code-mixed text classification in tamil and malayalam using ulm- fit. In *FIRE (Working Notes)*, pages 528–534, (2020).
[48] Supriya Chanda and Sukomal Pal. Irlab@ iitbhu@ dravidian-codemix- fire2020: Sentiment analysis for dravidian languages in code-mixed text.In *FIRE (Working Notes)*, pages 535–540, (2020).
[49] Nikita Kanwar, Megha Agarwal, and Rajesh Kumar Mundotiya. Pits@ dravidian-codemix-fire2020: Traditional approach to noisy code-mixed sen- timent analysis. In *FIRE (Working Notes)*, pages 541–547, (2020).
[50] Ruijie Sun and Xiaobing Zhou. Srj@ dravidian-codemix-fire2020: Auto- matic classification and identification sentiment in code-mixed text. In *FIRE (Working Notes)*, pages 548–553, (2020).
[51] Nitin Nikamanth Appiah Balaji, B Bharathi, and J Bhuvana. Ssncse nlp@ dravidian-codemix-fire2020: Sentiment analysis for dravidian languages in code-mixed text. In *FIRE (Working Notes)*, pages 554–559, (2020).
[52] Xiaozhi Ou and Hongling Li. Ynu@ dravidian-codemix-fire2020: Xlm- roberta for multi-language sentiment analysis. In *FIRE (Working Notes)*, pages 560–565, (2020).
[53] Yandrapati Prakash Babu, Rajagopal Eswari, and K Nimmi. Cia nitt@ dravidian-codemix-fire2020: Malayalam-english code mixed sentiment analysis using sentence bert and sentiment-features. In *FIRE (Working Notes)*, pages 566–573, (2020).
[54] Bo Huang and Yang Bai. Lucashub@ dravidian-codemix-fire2020: Sen- timent analysis on multilingual code mixing text with m-bert and xlm- roberta. In *FIRE (Working Notes)*, pages 574–581, (2020).
[55] Abhinav Kumar, Sunil Saumya, and Jyoti Prakash Singh. Nitp-ai-nlp@ dravidian-codemix-fire2020: A hybrid cnn and bi-lstm network for senti- ment analysis of dravidian code-mixed social media posts. In *FIRE (Work-ing Notes)*, pages 582–590, (2020).
[56] S Anbukkarasi and S Varadhaganapathy. Sa svg@ dravidian-codemix- fire2020: Deep learning based sentiment analysis in code-mixed tamil- english text. In *FIRE (Working Notes)*, pages 591–596, (2020).
[57] Anita Saroj and Sukomal Pal. Irlab@ iitv@ dravidian-codemix-fire2020: Sentiment analysis on multilingual code mixing text using bert-base. In *FIRE (Working Notes)*, pages 597–606, (2020).
[58] Jose Ortiz-Bejar, Jesus Ortiz-Bejar, Jaime Cerda-Jacabo, Mario Graff, and Eric Sadit Tellez. Umsnh-infotec@ dravidian-codemix-fire2020: An ensem- ble approach based on a multiple text representation. In *FIRE (WorkingNotes)*, pages 607–614, (2020).
[59] Deepesh Sharma. Tads@ dravidian-codemix-fire2020: Sentiment analysis on codemix dravidian language. In *FIRE (Working Notes)*, pages 615–619,(2020).
[60] Parameswari Krishnamurthy, Faith Varghese, and Nagaraju Vuppala. Parameswari faith nagaraju@ dravidian-codemix-fire: A machine-learning approach using n-grams in sentiment analysis for code-mixed texts: A case study in tamil and malayalam. In *FIRE (Working Notes)*, pages 620–627, (2020).
[61] Yueying Zhu and Kunjie Dong. Yun111@ dravidian-codemix-fire2020: Sen- timent analysis of dravidian code mixed text. In *FIRE (Working Notes)*, pages 628–634, (2020).

[62] BalaSundaraRaman Lakshmanan and Sanjeeth Kumar Ravindranath. Theedhum nandrum@ dravidian-codemix-fire2020: A sentiment polarity classifier for youtube comments with code-switching between tamil, malay-alam and english. *arXiv preprint arXiv:2010.03189*, (2020).

[63] Shubhanker Banerjee, Arun Jayapal, and Sajeetha Thavareesan. Nuig- shubhanker@ dravidian-codemix-fire2020: sentiment analysis of code- mixed dravidian text using xlnet. *arXiv preprint arXiv:2010.07773*, (2020).

[64] Priya Rani, Shardul Suryawanshi, Koustava Goswami, Bharathi Raja Chakravarthi, Theodorus Fransen, and John Philip McCrae. A compar- ative study of different state-of-the-art hate speech detection methods in hindi-english code-mixed data. In *Proceedings of the second workshop on trolling, aggression and cyberbullying*, pages 42–48, (2020).

[65] Siva Subrahamanyam Varma Kusampudi, Preetham Sathineni, and Rad- hika Mamidi. Sentiment analysis in code-mixed telugu-english text with un- supervised data normalization. In *Proceedings of the International Confer- ence on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 753–760, (2021).

[66] Shashank Sharma, PYKL Srinivas, and Rakesh Chandra Balabantaray. Text normalization of code mix and sentiment analysis. In 2015 international conference on advances in computing, communications and informatics (ICACCI), pages 1468–1473. IEEE, (2015).

[67] Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In Proceedings of the 11th forum for information retrieval evaluation, pages 14–17, (2019).

[68] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In Proceedings of the international AAAI conference on web and social media, volume 11, pages 512–515, (2017).