Comparative Analysis of BiLSTM and KNN Algorithms for Sentiment Analysis in Tamil-English Code-Mixed Text: A Comprehensive Evaluation Framework

Aakash Ashok Kumar, Poundoss Chellamuthu, Denceli Dennis and Aathithya Sharan

Abstract--- *Sentiment analysis in Tamil-Code Mixed data is challenging mainly due to its linguistic challenges. This paper analyzes several methodologies like deep learning (DL), machine learning (ML) and transformer-driven approaches and at the same time proposing a BiLSTM-driven approach for evaluating the sentiment of a Code-Mixed Tamil sentence. This research compares two approaches to analyze sentiment polarity in Code-Mixed Tamil sentence, BiLSTM and KNN, out of which the BiLSTM algorithm came out as the better algorithm for sentiment classification for Code-Mixed Tamil Text. The methodology achieves an F1 score of 0.435563, an accuracy of 0.746383, a precision of 0.332918, a recall of 0.629717, and a loss of 0.01 for the BiLSTM algorithm, and for the KNN algorithm, the methodology achieves an F1 score of 0.235714, a recall of 0.155660, a precision of 0.485294, and an accuracy of 0.828731. This study highlights several methods such from TF-IDF-based models to transformer architectures, highlighting the challenges faced during sentiment analysis in Code-Mixed Tamil sentences.*

Keywords--- *Tamil-English Code-Mixed, NLP, BiLSTM, KNN, Deep Learning, Text Classification, Sentiment Analysis, Data Imbalance, Code-Mixed Text.*

I. INTRODUCTION

RESEARCH in the domain of sentiment analysis is vast in high-resource languages like English, and this is mainly because languages like English are high-resource languages and extracting data for these languages is easy [1]. The need to build a sentiment analysis model that is language-specific will require language-specific data and will always outperform multilingual sentiment analysis models [2]. As a result, there is a growing need to develop sentiment analysis systems that can handle low-resource languages effectively. Tamil-English code-mixed text requires tools that are specialized to analyze user-generated content. Code-mixing, which is alternating between any language and English at the word, phrase, or sentence level, will complicate the NLP pipelines due to informal grammar, transliteration variations, and morphological richness [3]. Several methodologies existed before, and their works are summarized in this paragraph. Sajeetha, T. conducted an evaluation of five sentiment analysis approaches, including lexicon-based, supervised machine learning, hybrid, and clustering methods on various corpora, and concluded the best approach for sentiment analysis of Tamil Text [4]. Abhay Vishwakarma and Abhinav Kumar et al. compared a range of traditional ML models and DL architectures, like ensemble voting classifiers, LSTM, and deBERTa, to address the challenge of multiclass political sentiment analysis in Tamil social media, which eventually led them to conclude the ensemble voting classifier as the best performer among those tested [5]. Six additional recursive neural network models were suggested by Suba Sri Ramesh Babu and coworkers to perform Tamil sentiment analysis based on the performance of existing algorithms. They indicated that recursive neural networks outperform frequency-based or vector-space-based models in analyzing and determining the sentiment of complicated phrases, as well as improving the accuracy of inter-sentence sentiment prediction through the use of inter-sentential prediction methods [6]. Studies that have been done in Tamil Sentiment Analysis allow researchers to either evaluate existing methodologies, compare different methodologies, or suggest new methodologies to enhance sentiment analysis in the Tamil language by enhancing the accuracy of these methodologies or by addressing the unique challenges presented by Tamil text. Machine learning is a very important domain in AI that develops algorithms that learn data patterns to predict/decide outcomes for which it was built in the first place. Unsupervised learning, reinforcement learning, and supervised learning techniques are employed for performing operations including classification, clustering of data sets into groups, and sequential decision-making processes. Some of these include support vector machines, decision trees, and neural networks, and their

Aakash Ashok Kumar, Chennai Institute of Technology, Sarathy Nagar, Kundrathur, Chennai, Tamil Nadu, India. E-mail: writetoakashkumar@gmail.com, Orcid: https://orcid.org/0009-0005-8153-5983

Poundoss Chellamuthu, Chennai Institute of Technology, Sarathy Nagar, Kundrathur, Chennai, Tamil Nadu, India. E-mail: mrpdoss@gmail.com

Denceli Dennis, Chennai Institute of Technology, Sarathy Nagar, Kundrathur, Chennai, Tamil Nadu, India. E-mail: dencelid@citchennai.net, Orcid: https://orcid.org/0009-0008-4681-5615

Aathithya Sharan, Chennai Institute of Technology, Sarathy Nagar, Kundrathur, Chennai, Tamil Nadu, India. E-mail: aadhithyasharana.act2023@citchennai.net

DOI: 10.9756/BIJDM/V16I1/BIJ26007
Received: March 10, 2026; Revised: April 13, 2026; Accepted: May 07, 2026; Published: June 16, 2026

implementation depends upon the research requirements [7]. ML techniques can be employed to solve problems in NLP, such as recognizing emotions in language [8].

This method employs two algorithms, namely, BiLSTM and KNN. A BiLSTM classifier algorithm is used to identify the sentiment of mixed Tamil-English text. During testing of the BiLSTM classifier algorithm, an F1 score of 0.435563, an accuracy score of 0.746383, a precision score of 0.332918, a recall score of 0.629717, and a loss of 0.01 were recorded. On testing of the KNN classifier algorithm, scores of 0.235714 for the F1 score, 0.155660 for the recall score, 0.485294 for the precision score, and 0.828731 for the accuracy score were recorded.

II. Methodology

This section explains several steps that had to be undertaken to achieve the results presented in this study. Beginning with the Code-Mixed Tamil dataset, the data undergoes processing and vectorization before being divided into testing and training sets. Both BiLSTM and K-Nearest Neighbor models are then trained on the training data. Then, these trained methods are tested on the test data, and their performance is assessed through model evaluation. In the end, the evaluated results lead to sentiment prediction. The accompanying flowchart, figure 1, visually summarizes this entire process.

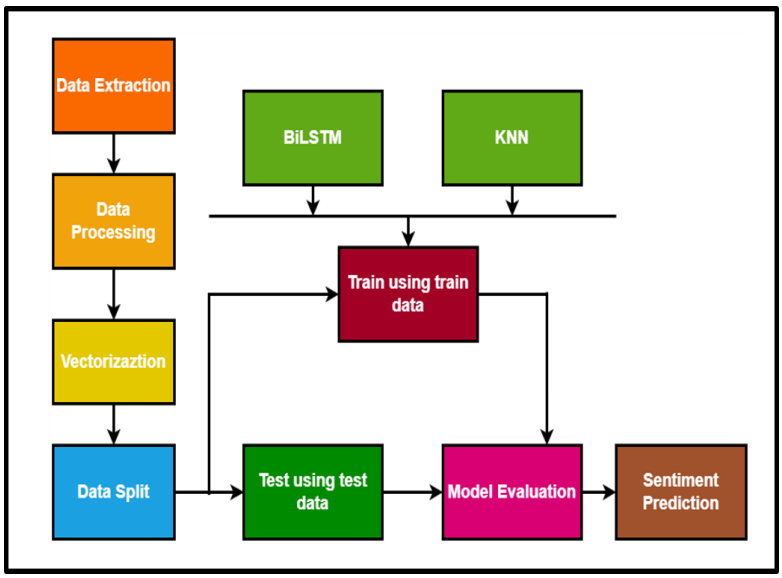

Figure 1: Workflow of the Proposed Methodology

The flowchart presented here provides an overview of the main steps involved in the process, but it does not elaborate on the specific procedures, algorithms, or parameter choices applied at each stage of the process.

2.1. Data Preparation and Class Balancing

Creating a stable and strong model for use in sentiment classification requires several critical steps; among them are the preparation of the data and class balancing of the original dataset before it can be split into test and training datasets. The dataset selected for training and testing is required to meet the designer's specifications for preparation, while also optimizing performance. This section discusses the data preparation steps for building these two proposed models, the BiLSTM and the KNN algorithms, to achieve the desired results. It is equally important to create a robust model for sentiment analysis by balancing the classes to correspond with the goals of the study. Removing unneeded labels from the prepared dataset will decrease the overall noise in the prepared dataset, thus allowing the SVM to work with the prepared dataset optimally. Detailed explanations of these processes will be discussed in the following sections, supported by a flowchart.

Preparation of the data collection and preprocessing procedures is the first step taken. Datasets used in the study were obtained from publicly available data sources, and they contain 15 744 Tamil-English code-mixed sentences, each of which has an associated sentiment label [9]. The Tamil text datasets will be loaded with pandas' data manipulation libraries and will be initially controlled for quality to maintain the integrity of the classification process. The binary classification process will be accomplished by filtering out the labels and only retaining those with either a negative (0) or a positive (1) sentiment label. All other multivariate label values will be deleted, as introducing them would add confusion in classifying future sample sentences into one of two classes, thus creating ambiguity in the classification of future sentences. The majority class will have its number of samples reduced using sklearn.utils.resample function to a balanced number of samples of 2,000 samples per class. If this were to occur, it would allow the model's training process to occur without any bias toward the dominant class, thus creating a balanced model and a strong model to classify the sentiment of samples in a general sense. Creating a balanced dataset will be followed by the process of tokenization and normalization, which will be discussed further in the next few sections. The flowchart below (Figure 2) illustrates the processes that occur in the preparation of the dataset and class balancing.

Figure 2: Flowchart for Data Processing and Class Balancing

2.2. Text Tokenization and Normalization

To develop the tokenizer for this dataset (i.e., Code-Mixed Tamil Dataset), it was decided that the tokenization process would be based on the torchtext.data.utils.get_tokenizer method with 'basic_english' as a basis, but needed to be modified to handle the Code-Mixed Tamil dataset. The initial sentence dataset has gone through a process called tokenization in which the syntactic structure of the sentence remains unchanged, but individual tokens have been created so they are manageable input sequences for the neural network architecture [10]. In addition, a standard text cleaning procedure is applied to the dataset through the preprocessing pipeline and consists of removing all special characters and punctuation marks, as well as any other non-text-related content. Noise will be introduced into the data during training because of the existence of this content. Additionally, stop-word removal and stemming techniques will both be implemented to improve any negative effect that these methods may have on the Code-Mixed Tamil text. Because of the complexity of the morphological structure of this dataset compared to a language such as English, the use of these methods must receive careful consideration before implementing them.

2.3. Vocabulary Building and Special Token Integration

Vocabulary building represents one component of the text representation pipeline. The methodology for constructing the vocabulary is through the use of torchtext.vocab.build_vocab_from_iterator to create a vocabulary to define all Tamil elements using the training data. Additionally, a list of special tokens will be used to allow for effective handling of variable-length sequences and prevent the need for out-of-vocabulary tokens. The list of special tokens includes (vector representation of an unknown word), (vector representation of the entire sequence), <sos> (vector representation of the beginning of the sequence), and <eos> (vector representation of the end of the sequence). The vocabulary will be built so as to provide computational efficiency through the use of a relatively small number of vocabulary items.

2.4. Text Vectorization and Sequence Encoding

Numeric representations of sequences are achieved through vocabulary creation, and a mapping from each word to its respective token id. The vectorizing process allows the LSTM to read the sequence as a number while maintaining the order in which the words occur and the contextual relationships among the words [11]. The padding of sequences enables the same dimensionality for all inputs across batches. Torch.nn.utils.rnn.pad_sequence performs this action with varying-length sentences.

2.5 Model Architecture

The model architecture provides a better understanding of the system operation. The system can be built through the use of various modules, leading to the successful classification of sentiment.

2.5.1 Configuration of the LSTM Network

The sentiment classification model uses a bidirectional LSTM model. This allows for the capture of sequential language dependencies and the contextual relationships among words. The Text classifier architecture presented here is made up of a number of interconnected components through which text sequence is processed hierarchically from word-level embeddings to sentence-level predictions. Tokens are transformed from token IDs to a series of densified representations through the embedding layer, thereby creating an almost continuous semantic space, creating word-to-word relationships and meanings. The result is that dense representations are fed into bidirectionally connected LSTM layers, which can read the sequence in both the forward and backward directions simultaneously. It facilitates the creation of a more complete context which is required because of the complex structure of the tokens in the dataset.

2.5.2 Components of the Network and Regularization

The LSTM architectural model using a dropout regularization approach will help avoid overfitting to the dataset [12], especially when a deep learning approach is used on under-resourced low-resource language datasets. The fully connected output layer acts as a bridge from hidden LSTM states to L-class sentiment classification scores, whereby each sentiment classification score is calculated through the sigmoid function. The model utilizes the configuration of a bidirectional LSTM to identify contextual dependencies through the maximum and minimum of each series of words in each binary classification scenario in a sentence. In addition to understanding the distribution of the sentiment indicators within the maximum and minimum within the dataset, the LSTM model will also provide a better understanding of the lack of structural complexity in the language used to express the data for which sentiment information is being derived through analytic processes through the LSTM model [13]. In addition to the use of the BiLSTM model, the KNN algorithm was implemented as an alternative approach to sentiment classification on the Code-Mixed Tamil dataset. Prior to applying any model to the dataset, all sentences were tokenized and converted to numeric representations using TF-IDF. This method captures a word's robustness across the entire dataset while preserving the code-mixed dataset's semantic sparsity and is useful for a KNN algorithm [14].

When classification is applied to a test instance using the TF-IDF features as input variables, the KNN algorithm measures the degree of geographical similarity of each, instance to each other by using the Euclidean distance between each instance and each of the training instances. The k instances that are identified as closest (k set to five for use in this research) will be assigned the label corresponding to the conceptual topic of the majority class represented by the k instances. The positions of the instances in relation to each other geographically provide a solid geometric category reference for measuring the closeness of various locations from an empirical perspective, demonstrating the KNN algorithm as a robust baseline when not controlling parameters or models. The characteristics of the KNN algorithm provide a relatively simple and straightforward approach to comparative analysis with deep learning architectures [15].

2.6. Training Methodology

Training methodology is a process that utilizes various methodologies. Metrics that are used to evaluate and monitor training include: F1 score, recall, precision, and accuracy.

2.6.1. Loss function and optimization strategy

The selection of the appropriate loss function, neural networks (such as Bilstm), and their associated optimization strategies, along with the overall model, is crucial for achieving maximum accuracy in sentiment determination. Within the analysis of sentiment, there are numerous advanced loss functions being utilized with neural networks (example: Binary cross-entropy with logits), providing numerical stability and excellent computation of gradients during the training process for tasks with binary sentiment classifications.

The Adam optimizer is also being used here to optimize the tuning of the learning rate and adaptively tune the learning rate for use in training on sequential data (example: NLP).

The K-Nearest Neighbor (KNN) Algorithm, although it uses instantaneous instance classifying methods, provides an entirely different approach than that of the previously mentioned algorithms above. The KNN algorithm is non-parametric (does not require prediction parameterization); there is no learning associated with the KNN method or optimization methods pertaining to gradient descent processes in the same manner as the BiLSTM. The KNN algorithm only utilizes the information provided in the labeled training/sample set when predicting classification, based on proximity to classified training samples in feature space, as a basis for prediction of classifications based on newly provided samples.

A comparison of the algorithms [BiLSTM & KNN] will be presented to assess the accuracy of both the bi-LSTM and KNN approaches to Tamil/English code-mixed sentiment analysis [16]. An evaluation of the training process using Binary Cross-Entropy with Logits (nn.BCEWithLogitsLoss) as a loss function provides for the most numerical stability and efficient gradient calculations in binary classification tasks and improves learning of the Tamil Sentiment. The Adam Optimizer will also provide efficient updates to the gradient based on Andrew Adaptive Learning Rates (Use for LSTM) through the process of training and therefore used; considered; for Training on NLP tasks.

KNN is a non-parametric framework that does not utilize a standard loss function and/or optimization strategy in comparison to the aforementioned neural networks [17]. Therefore, there are no model parameters for K-Nearest Neighbors to learn, and therefore, there are no gradient descent minimization processes or objective function minimization processes associated with K-Nearest Neighbors decisions. K-Nearest Neighbors provides for predictions based solely on labeled Data Set samples in the Feature space and does not involve back propagation through its evaluation process nor iterative modifications.

An evaluation of both models through the use of standard classification metrics will determine both models' performance. The following Flow Charts (Figure 3A & 3B) provide the methodology used in this evaluation process.

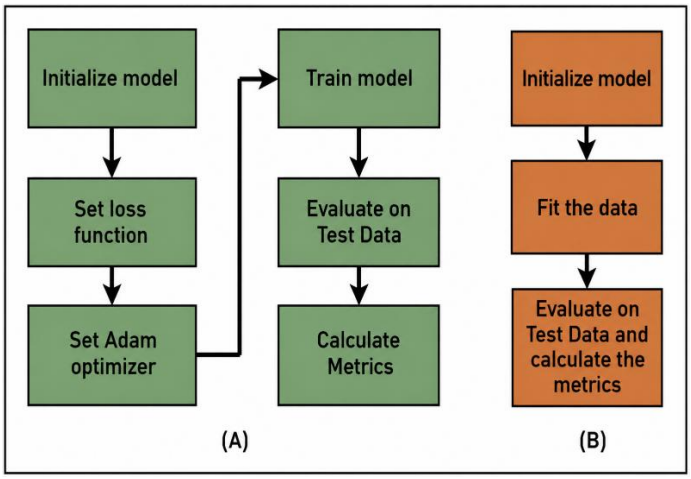

Figure 3 (A & B): Flowchart for Loss Function and Optimization Strategy in Bi-LSTM

2.6.2. Training Process and Convergence Monitoring

In the BiLSTM algorithm, the training loop processes data in batches through torch.utils.data.DataLoader. This processing of data in batches helps in increasing computational efficiency. Each epoch here includes a forward propagation through the LSTM architecture, loss computation against ground truth labels, and a backward propagation for gradient calculation, and the parameters here get updated through the optimizer.

For the KNN approach, the training as well as the test sentences are first transformed into numerical feature representations using TF-IDF vectorization, where the number of features is limited to 5000 to balance the expressiveness of the sentence and computational efficiency. The resulting TF-IDF matrices encode the importance of words across the dataset. The KNN classifier itself is trained by storing these TF-IDF feature vectors and their associated sentiment labels. During inference, the model calculates the Euclidean distance of each and every test sample and all stored training samples, identifies the five nearest neighbors, and allocates the sentiment label to the common sentiment label by majority voting. From this it is understandable that the KNN algorithm is simple but a robust model for sentiment classification

Training progress is monitored for the BiLSTM algorithm with the help of accuracy metrics calculated using custom binary accuracy functions. Validation procedures track model generalization capability and guide hyperparameter adjustments to optimize performance on Tamil sentiment analysis tasks.

2.7. Training Parameters

Both models were trained with carefully selected hyperparameters that enhance their ability to predict text sentiment. Tests were run on both models using various parameter values, and the parameters that brought the best out of the model were used here:

Table 1: Bi-LSTM Test for Variables

Epochs	Dropout	Batch size	Accuracy	Precision	recall	F1
3	0.2	2	0.685131	0.264196	0.790094	0.395981
5	0.3	4	0.712761	0.272152	0.811321	0.407583
7	0.4	8	0.745650	0.349481	0.476415	0.403194
10	0.5	16	0.746383	0.332918	0.629717	0.435563
15	0.6	32	0.741905	0.324476	0.547170	0.407375

From table 1, the method is trained for 10 epochs, which is the most suitable epoch number for achieving the balance between overfitting and underfitting as per the given dataset. Batch size of 16 was used to attain a stable training process. To enhance the generalizability of the model, a dropout layer with a dropout rate of 0.5 was employed in order to prevent overfitting. Adam optimizer was designed owing to its adaptiveness according to the data. A learning rate of 1e-3 was used. The sigmoid activation function was used because it is appropriate for binary sentiment classification of the code-mixed dataset. The results are shown in table 2 below:

Table 2: Parameters for BiLSTM

Parameter	Value
Batch Size	16
Epochs	10
Learning Rate	0.001
Dropout	0.5
Activation Func.	Sigmoid

And for the KNN model, a similar approach is taken in order to choose the right n neighbors to get the best out of the KNN model. Table 3 presents the results from the testing process.

Table 3: KNN Test Variables

N neighbors	accuracy	precision	recall	F1 score
1	0.788315	0.272727	0.148585	0.192366
3	0.824330	0.380952	0.056604	0.098563
5	0.830732	0.523810	0.025943	0.049438
7	0.829132	0.482353	0.096698	0.161100
9	0.828731	0.485294	0.155660	0.235714

From table 3, the KNN model here was trained using the binary cross-entropy loss function. In KNN-based sentiment classification the text data was first vectorized using the TF-IDF technique and was converted into numerical features. The number of neighbors, which is denoted as 'k', was set to 5, and Euclidean distance was used as the distance metric and its purpose is to obtain the closest neighbors. The labels were set to binary which means that representation was done in two ways which is both positive and negative sentiments. For the KNN algorithm the training parameters are tabulated in table 4.

Table 4: Parameters for K-Nearest Neighbour

Parameter	Value
Vectorization method	TF-IDF
Number of neighbours	5
Distance metric	Eucledian Distance
Feature shape	Based on TF-IDF output
Label Type	Binary

2.8. Evaluation Framework

There are four measures that are employed in assessing the models' performance. The model's predictive performance and its ability to identify significant negative and positive instances can be measured.

Equation (1) provides a formula for calculating the accuracy of a prediction model [18].

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

In equation (2), Precision quantifies the quality of positive predictions. This metric becomes especially significant in conditions where false positives are costly, as the precision focuses on minimizing them.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (2)$$

The value in equation (3) assesses the sensitivity of the model to correctly classify all positives; it may also be referred to as recall, TPR, or sensitivity. Recall is very important to minimize the occurrence of any FP.

$$\text{Recall} = \frac{TP}{FN+TP} \quad (3)$$

Equation 4 gives the F1 score, which provides a more realistic performance estimate than accuracy, as it penalizes extreme differences between recall and precision.

$$\text{F1} = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4)$$

All of these metrics come together to provide information about how well a model performs in classifying sentiments in a dataset.

III. RESULTS

3.1 Performance Comparison

Performance comparison is done here by evaluating models using metrics. The metrics, namely the F1 score, precision,

recall, and accuracy, are widely used when it comes to evaluating how the machine learning model performs. The metrics obtained for both algorithms are analyzed and tabulated in table 5.

Table 5: Comparison Table for Proposed Methodologies

Method	Accuracy	Precision	recall	F1	Loss
BiLSTM	0.746383	0.332918	0.629717	0.435563	**0.01**
KNN	0.828731	0.485294	0.155660	0.235714	N/A

Based on the value that these metrics hold one can evaluate that how a machine learning model performs a task for which it was trained/built in the first place. These metrics are often evaluated by relating to each other to come to a solid conclusion about the model. BiLSTM is chosen as it achieves a higher F1 score with better recall, making it more reliable than KNN despite lower accuracy. The metrics obtained for the Bi-LSTM are used for comparative analysis with past methodologies. A comparative analysis of scores obtained from the proposed methodologies and past methodologies is tabulated in table 6.

Table 6: Comparison of Proposed Methodology with Existing Sentiment Analysis Approaches

Study	Language	Method	F1 Score	Precision	Recall	Accuracy
Sripriya & Divya (2021): TF-IDF + Random Forest [19]	Tamil (Code-mixed)	TF-IDF + Random Forest	0.35 (weighted avg)	0.37 (weighted avg)	0.35 (weighted avg)	0.35
Ehsan et al. (2023): ELMo + BiLSTM [20]	Tamil (Code-mixed)	ELMo + BiLSTM	0.2877 (macro, test)	0.3245 (macro, test)	0.3138 (macro, test)	0.322 (test)
Roy et al. (2025): BERT-based Transformers [21]	Tamil-English	L3Cube-Tamil-BERT	0.19 (macro, val)	N/A	N/A	0.45 (val)
Proposed Methodology	**Tamil-English**	**BiLSTM**	**0.435563**	**0.485294**	**0.629717**	**0.746383**

A comparative bar graph shows the relative performance of each approach; the BiLSTM model outperforms all prior research. The previously highlighted performance improvement is due to the use of bi-directional sequence modeling and having a balanced training data set. The choice of F1 Score and Accuracy as principal metrics by which the performance of the model was evaluated was because both provide an equitable view of model performance and because they were used extensively across the studies, enabling an equitable comparison. Figure 4 shows the Comparative Graph.

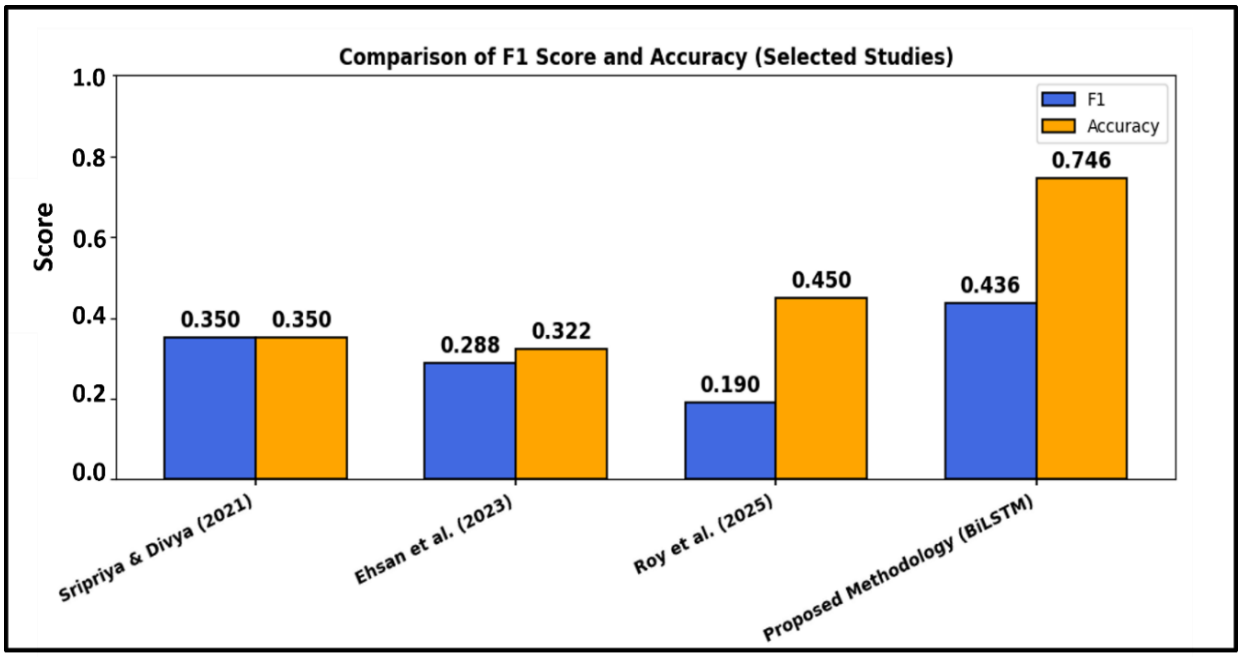

Figure 4: Comparison Table of F1 Score and Accuracy

Table 6 and figure 4 show that both the F1 score and accuracy are higher than in other methodologies, demonstrating improved precision and recall. The improved performance of BiLSTM comes from its architecture and how it was trained to create an effective model for that dataset.

3.2 Metric Analysis

The use of BiLSTM and KNN is compared to show the different trade-offs involved with obtaining accuracy, balanced classification, and generalization. Although KNN has the highest accuracy (0.82) and better precision (0.48), it has low

recall (0.15) and F1 score (0.23). That is, although KNN has identified a significant number of samples correctly overall, it has difficulty in identifying most true positives, and as such, it will not generalize well to applications needing balanced detection across classes.

Conversely, BiLSTM has a slightly lower accuracy (0.74), but has much better recall (0.62) and F1 score (0.43). Thus, BiLSTM is capable of correctly identifying positive instances and maintaining a good balance. Therefore, BiLSTM can be considered the more reliable approach, especially in scenarios where identifying sentiments accurately across all categories is more important than overall accuracy alone.

3.3. Machine Learning

This section gives a detailed description of how each model works, using flowcharts, confusion matrices, and various error metrics derived from the confusion matrix. The flowcharts used here summarize how each of the models works in a pictorial representation. Both models are very efficient in classifying tasks, and for this purpose, both models are chosen to classify the sentiment of a code-mixed dataset. The models help with finding the accurate sentiment of any given code-mixed text and are explained in detail in this section with the help of pictorial representations of the process that each model has to undergo.

The sentiment analysis pipeline of the BiLSTM model starts with a detailed preparation of the data and its tokenization, where the raw Tamil-English mixed text is cleaned of unwanted characters, special symbols, and made consistent by changing all characters to lowercase, removing excess whitespace and punctuation. The text that has been cleaned is then broken down to tokens that include both words and units of subwords, because the nature of code-mixed languages on social media is diverse and informal in its tokenization. Once tokenized, the representational form of the text has changed into an embedding representation of each token, either using pre-trained embeddings (such as GloVe) that utilise the linguistic knowledge from large amounts of prior written text, or randomly generated embeddings that are learned fine-tuned during training on the basis of the nuances of sentiment and code-mixed structures provide good representation of both the semantic (meaning) and syntactic (they order in a sentence) associations of the tokens. These embeddings will be utilized as input for the core of the architecture, which are the BiLSTM layers. By processing each token/token sequence forward, as well as backward, the BiLSTM will capture the contextual dependencies of the entire input, which is very important for understanding the complex structure, code-switching patterns, and sentiment cues that exist in Tamil-English code-mixed texts. The outputs from the LSTM layers are then regularized using dropout, a method that casually deactivates a portion of the neurons during each training iteration, minimizing the risk of overfitting and promoting the learning of more robust and generalizable features. Following regularization, the handled features are then accepted through a dense layer that further abstracts and combines the learned representations, preparing them for the final classification stage. The process is represented in the flowchart, a pictorial representation that helps with understanding the process in simple terms, without much explanation, is given in figure 5.

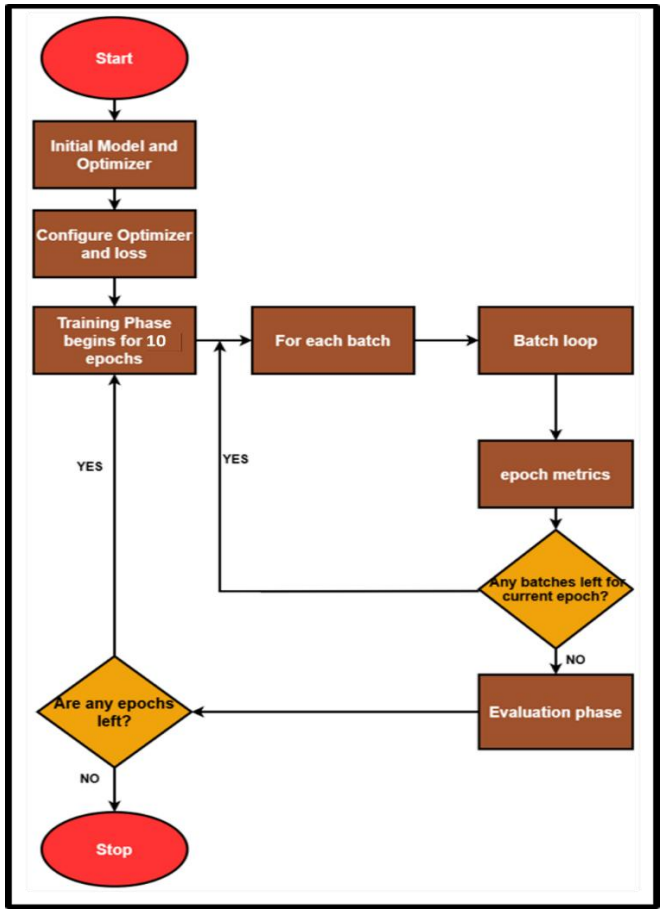

Figure 5: Flowchart for BiLSTM Algorithm

In the KNN algorithm, after preprocessing, all sentences are vectorized using TF-IDF, which changes the text into high-dimensional numerical feature vectors that quantitatively capture the importance and distribution of words across the entire dataset. These TF-IDF vectors are limited to 5,000 features for computational efficiency and are then stored alongside their corresponding sentiment labels, where there is no need for iterative training or parameter optimization. When inference takes place, the KNN algorithm computes the Euclidean distance between each test instance's feature vector and all training samples by mapping the test sample within the geometric structure of the feature space. After calculating the distance between each input and the known instances of data, it identifies the five closest instances (k = 5) and then applies a majority voting mechanism to determine what the expected sentiment is going to be. This proves that the KNN method is an easy-to-trace and clear-cut decision-making algorithm, and also continually updates as data changes, making it the most reliable of all the baseline methods used for this type of analysis. Figure 6 shows a complete visual description of all operational processes that are used by KNN in this study to classify the sentiment of mixed code using the KNN Algorithm. A visual description of all processes that occur during an analysis leads to a better understanding of how to classify sentiment. The process starts by post-processing the raw textual data (cleaning, normalization, and reduction of noise) to create a consistent and uniform dataset. Each sentence is transformed into a high-dimensional representation through the creation of

TF-IDF features that reflect the relative value of each word in the entire dataset, allowing the KNN algorithm to identify patterns related to sentiment. Feature vectors created will remain in the KNN algorithm's memory along with their associated sentiments. Any new sentence created will be subjected to the same preprocessing (cleaning, normalization, and reduction of noise) and TF-IDF processing. The KNN calculates the distance of the feature vector corresponding to the new sentence and all existing feature vectors stored in memory, identifies the KNN, and, based on the sentiments assigned by these neighbors, predicts the sentiment of the new sentence using a majority vote. Figure 6 visually highlights each of these 4 areas within KNN from data preprocessing through prediction of sentiment; the KNN process can clearly be defined and traced. Figure 6, given below, is a simple and clear pictorial representation of the KNN algorithm and how it approaches sentiment classification tasks.

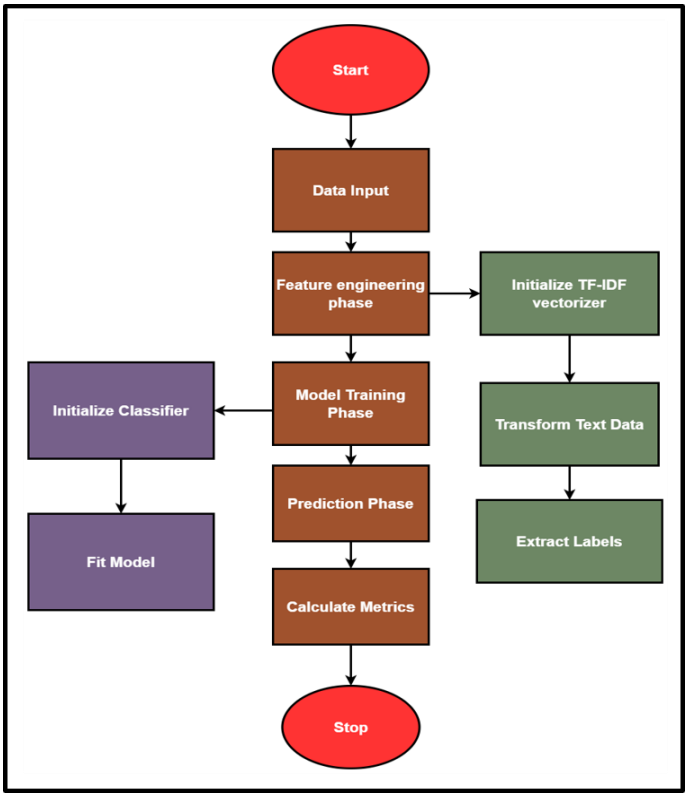

Figure 6: Flowchart of the KNN Algorithm

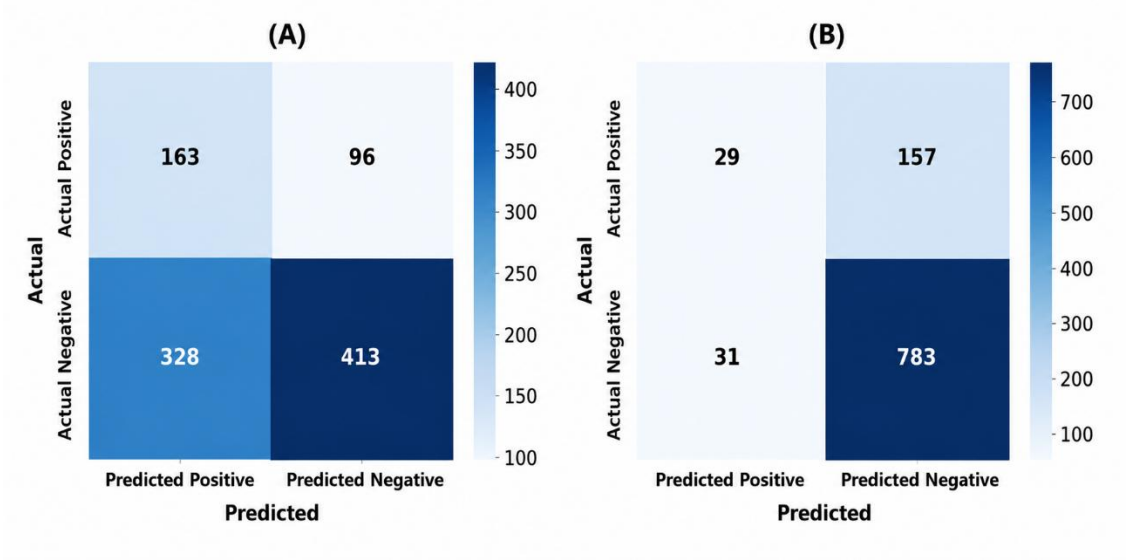

Figure 7: (A) Confusion Matrix for Bi-LSTM, (B) Confusion Matrix for KNN

By comparing the predicted output values against the expected ones, a confusion matrix can be used to measure the performance of a classifier. The conventional 2x2 confusion matrix used in binary classification comprises four columns, namely, TP, TN, FP, and FN. A true positive denotes that the classifier correctly classified the positive class. On the other hand, a true negative implies that the classifier correctly predicted the negative class.

From figure 7, the confusion matrices, figure 7(A) and figure 7(B), it is clear that the BiLSTM and KNN models demonstrate contrasting performance characteristics. As observed in figure 7(A), BiLSTM detected 163 TP and 413 TN; however, the BiLSTM framework detected 96 FP and 328 FN incorrectly. As per the results obtained, the BiLSTM model is capable of detecting quite a number of true positives; however, it does a terrible job of not being able to detect any false alarms, where it tends to classify many negatives into positives. The measurement for detecting true positives proved to be very poor, with all of the measurements taken, having only been able to measure about 29 true positives and 157 false negatives via KNE (Figure 7 (B)). There were 783 true negatives and 31 FP, as per the previous measurement of TN; hence, this shows that KNE favored TN over predicted positives. In summary,

BiLSTM can be said to have a fairly balanced detection value, where it detects many positives but false positives, while KNE has very high prevalence for TN. Therefore, BiLSTM is better suited for tasks where missing positive cases would be costly, whereas KNN is more appropriate in scenarios where avoiding false positives is critical. In order for one to measure and hence improve on the model or determine its effectiveness at performing any specific task, it is important to evaluate its performance in classification using various measures. Some important methods that are employed include measures of error.

In (5), the error rate measures the proportion of incorrect predictions out of the total samples and is calculated as: [22]

$$\text{Error Rate} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (5)$$

In (6), where FP is false positives, FN is false negatives, TP is true positives, and TN is true negatives. The FPR quantifies the proportion of negative samples incorrectly classified as positive and is given.

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (6)$$

Conversely, in (7), the FNR indicates the fraction of positive samples misclassified as negative, computed.

$$\text{False Negative Rate} = \frac{\text{FN}}{\text{FN} + \text{TP}} \quad (7)$$

Specificity, as given in (8), or the FNR, measures the model's ability to correctly identify negative instances.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad (8)$$

Finally, in (9), the balanced accuracy combines sensitivity (TPR) and specificity to provide a more reliable metric in imbalanced datasets.

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right) \quad (9)$$

These metrics help in understanding both overall and class-wise performance beyond simple accuracy, especially in scenarios with class imbalance. These metrics were obtained for both models, and based on these values, a better understanding of the model is obtained. These error values are tabulated and compared in table 7.

Table 7: Error Comparison of Proposed Methodologies

Model	Error Rate	FPR	FNR	Specificity	Balanced accuracy
BiLSTM	0.424	0.443	0.371	0.557	0.594
KNN	0.188	0.038	0.844	0.962	0.559

The performance of the BiLSTM network and the KNN algorithm varies with regard to their respective performances on various metrics, including error rate, FPR, FNR, specificity, and balanced accuracy (see figure 7). BiLSTM's overall error rate of 0.424 is higher than KNN's error rate of 0.188, which implies that BiLSTM makes more mistakes than KNN. However, after analyzing the errors made by each of these two networks, it is apparent that BiLSTM's false negative rate (0.371) is far better (lower) than KNN (0.844), indicating that BiLSTM is better at positively identifying positive cases and that BiLSTM misses fewer positive cases than KNN. Conversely, the FPR for KNN (0.038) is significantly lower than for BiLSTM, and as a result, KNN has a high specificity of 0.962, which demonstrates that KNN is very effective at correctly identifying negative cases and is unlikely to classify negative cases as positive. In contrast, BiLSTM's specificity of 0.557 demonstrates that BiLSTM fails to identify negative cases correctly compared to KNN. In terms of balanced accuracy, BiLSTM scored slightly higher than KNN (0.594 vs. 0.559), indicating that the classification performance of BiLSTM is more balanced than KNN, whereas KNN has high precision for negative cases but low recall for positive cases. Therefore, it is essential to correctly identify positive cases when using BiLSTM, such as predicting whether a sentiment, disease, or risk is present, because BiLSTM reduces the probability of missing true positive cases. However, KNN would be the preferred choice in circumstances when false alarms need to be minimized and when the correct classification of negative cases is critical, because KNN has a much lower probability of incorrectly flagging positive cases than BiLSTM. In general, although KNN has a lower error rate, BiLSTM provides a more even distribution of sensitivity and specificity scores, indicating that BiLSTM is also better suited for tasks where there is a need for comprehensive detection of entities in both groups.

Based on the flowcharts and confusion matrices BiLSTM and KNN models perform, with BiLSTM excelling at identifying complex linguistic patterns due to its deep learning architecture, while KNN provides a simple alternative to achieve similar results. This analysis of both methods demonstrates the benefit of using a deep learning neural model while still employing a classic form of machine learning to provide explanations in conducting sentiment analysis [23].

IV. Discussion

The comparison performs sentiment analysis of Tamil-English code-mixing over time using different methodologies. Sripriya & Divya (2021) [19] classified sentiment in Tamil using the 'TF-IDF + Random Forest Approach' and achieved moderate results with an F1 score of 0.35 and accuracy of 0.35; however, they found that their classic machine learning technique did not provide a deeper context of the sentiments represented in the code-mixed classifier. The methodology included contextual embedding of the ELMo model with their BiLSTM model, and they received an F1 score of 0.2877 and an accuracy of 0.322; however, even though they utilized the contextual embeddings, their results were not as good due to the complexity of the dataset. Similarly, Roy et al. (2025) tested transformer-based architectures using the L3Cube-Tamil-BERT and found an F1 score of 0.19 and an accuracy of 0.45. The lower F1 score produced by transformers indicates some difficulty in achieving a balance of recall and precision within this low-resource and code-mixed dataset. By comparison, the proposed BiLSTM method produced a balanced metric and recorded an F1 score of 0.4356 (precision of 0.4852, recall of

0.6297, accuracy of 0.7463). The performance of this model significantly improved on previous models; especially in the area of recall where the new model was able to identify sentiment bearing elements most effectively. The model did have an error rate of 0.424 and a relatively high false positive rate of 0.443 (indicating that there was evidence of misclassifications); however, the model's balanced accuracy of 0.594 suggests that the model will result in a more reliable identification of both positive and negative classes when compared to models that preceded it. When compared to KNN, which produced high specificity (0.962) and low recall (0.156) as a result of favoring negative predictions, the proposed BiLSTM produced both stronger results and a more holistic and applicable solution to the problem.

V. Conclusion

The results of this study showed that the proposed BiLSTM-based model provides significant improvement over previous methodologies for Tamil-English code-mixed sentiment analysis. Traditional machine learning-based models and transformer-based methods have had difficulties in classifying the nuanced sentiment of low-resource datasets and code-mixed text; however, the BiLSTM model proposed has shown much higher performance in recall and the overall balanced accuracy than traditional baseline methods. The higher performance demonstrated by the BiLSTM model proposed indicates that this method can be applied in practical conditions to find instances of positive sentiment. The BiLSTM model proposed has limitations related to specificity and false positives; however, it represents a strong platform upon which improvements can be made in the sentiment classification of low-resource languages, bridging the gap between conventional and advanced deep learning methods.

VI. Acknowledgment

I would like to express my sincere gratitude to Dr. Poundoss Chellamuthu for sharing his knowledge, insight, and continuous support throughout the research process. I would also like to extend my gratitude to Dr. Denceli Dennis for offering me constructive feedback, thoughtful criticism, and academic support that have helped to significantly improve the quality of the project. And I would like to thank Mr. Aathithya Sharan for his engagement, technical support, and insightful discussions, which were instrumental in the successful completion of this research.

Their collective efforts and support were instrumental in bringing this research to fruition.

References

[1] Velankar, A., Patil, H., & Joshi, R. (2022, November). Mono vs multilingual bert for hate speech detection and text classification: A case study in marathi. In *IAPR workshop on artificial neural networks in pattern recognition* (pp. 121-128). Cham: Springer International Publishing.

[2] Srivastava, V., & Singh, M. (2021). Challenges and considerations with code-mixed nlp for multilingual societies.

[3] Thavareesan, S., & Mahesan, S. (2019, December). 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). IEEE. https://doi.org/10.1109/ICIIS47346.2019.9063341.

[4] Vishwakarma, A., & Kumar, A. (2025, May). MNLP@ DravidianLangTech 2025: Transformers vs. Traditional Machine Learning: Analyzing Sentiment in Tamil Social Media Posts. In *Proceedings of the Fifth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages* (pp. 404-408).

[5] Padmamala, R., & Prema, V. (2017, August). Sentiment analysis of online Tamil contents using recursive neural network models approach for Tamil language. In *2017 IEEE International conference on smart technologies and management for computing, communication, controls, energy and materials (ICSTM)* (pp. 28-31). IEEE. https://doi.org/10.1109/ICSTM.2017.8089122.

[6] Afifi, H., Pochaba, S., Boltres, A., Laniewski, D., Haberer, J., Paeleke, L., ... & Seufert, M. (2024). Machine learning with computer networks: Techniques, datasets, and models. *IEEE access*, *12*, 54673-54720. https://doi.org/10.1109/ACCESS.2024.3384460.

[7] Jemai, F., Hayouni, M., & Baccar, S. (2021, June). Sentiment analysis using machine learning algorithms. In *2021 International Wireless Communications and Mobile Computing (IWCMC)* (pp. 775-779). IEEE. https://doi.org/10.1109/IWCMC51323.2021.9498965.

[8] Chakravarthi, B. R., Muralidaran, V., Priyadharshini, R., & McCrae, J. P. (2020, May). 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)* (pp. 202-210).

[9] Sennrich, R., Haddow, B., & Birch, A. (2016, August). Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)* (pp. 1715-1725).

[10] Vattikundala, J., & Prasad, M. S. G. (2025). A Novel Transfer Deep Learning Framework with Cross-Lingual Embeddings for High-Resource and Low Resource Languages for Sentiment Analysis. *Archives for Technical Sciences, 3*(34), 632–647. https://doi.org/10.70102/afts.2025.1834.632

[11] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

[12] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[13] Wani, M. N., Gaikwad, M. P. S., Solse, M. N., Kumar, M. A., & Mangire, M. G. (2022). Legal document classification using TF-IDF and KNN. *Int J Adv Res Sci, Commun Technol. Naksh Solutions*, 590-595.

[14] Chhatwal, R., Gronvall, P., Huber-Fliflet, N., Keeling, R., Zhang, J., & Zhao, H. (2018, December). Explainable text classification in legal document review a case study of explainable predictive coding. In *2018 IEEE international conference on big data (Big Data)* (pp. 1905-1911). IEEE.

[15] D'Souza, M., & Kale, R. (2021). Sentiment Analysis Using NLP Techniques in Social Media Monitoring. *International Academic Journal of Innovative Research, 8*(3), 15–19. https://doi.org/10.71086/IAJIR/V8I3/IAJIR0819

[16] Suyal, M., & Goyal, P. (2022). A review on analysis of k-nearest neighbor classification machine learning algorithms based on supervised learning. *International Journal of Engineering Trends and Technology*, *70*(7), 43-48.

[17] Emmert-Streib, F., Moutari, S., & Dehmer, M. (2019). A comprehensive survey of error measures for evaluating binary decision making in data science. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(5), e1303.

[18] Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, *21*(1), 6. https://doi.org/10.1186/s12864-019-6413-7

[19] Sripriya, N., & Divya, S. (2021). Sentiment Analysis Model for Code-Mixed Tamil Language. In *FIRE (Working Notes)* (pp. 1005-1010).

[20] Roy, B., Bhattacharyya, S., Gupta, P., & Kumar, N. (2025, May). LexiLogic@ DravidianLangTech 2025: Political Multiclass Sentiment Analysis of Tamil X (Twitter) Comments and Sentiment Analysis in Tamil and Tulu. In *Proceedings of the Fifth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages* (pp. 557-561).

[21] Emmert-Streib, F., Moutari, S., & Dehmer, M. (2019). A comprehensive survey of error measures for evaluating binary decision making in data science. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(5), e1303.

[22] Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: an overview.

[23] Sethuraman, P., Hassan, M. M., Veettil, R. P., Jayanthi, A., Kalyana Sundaram, N., & Patnaik, D. (2025). Evaluation of Latent Semantic Analysis in Multilingual Information Retrieval. *Indian Journal of Information Sources and Services, 15*(3), 160–169. https://doi.org/10.51983/ijiss-2025.IJISS.15.3.18