Original Research Article

Improving Sentiment Analysis of Tamil-English Code-Mixed Sentences

Mohnish Sivakumar

Irvington High School, 41800 Blacow Rd, Fremont, California, 94538, United States

ABSTRACT

This paper investigates sentiment analysis for Tamil-English code-mixed text, a common feature of social media communication in multilingual regions. Code-mixing in Romanized Tamil introduces challenges such as inconsistent spelling, transliteration, and noisy syntax that traditional models are not designed to handle. Using the FIRE-DravidianCodeMix 2020 (hereafter FIRE2020) dataset, lexicon-based methods, classical machine learning models, deep learning (LSTM), the multilingual transformer RemBERT, and hybrid approaches combining lexicon-based features with machine learning models were evaluated on sentiment classification. Results showed that classical models such as Logistic Regression, Naive Bayes, and SVM achieved the most stable performance, reaching around 69% accuracy with weighted F1-scores near 0.60. Deep learning and transformer models offered no clear advantage, with both LSTM and RemBERT performing slightly lower than the classical models, plateauing near 67% accuracy and weighted F1-scores around 0.54. These results emphasize that lightweight statistical models remain the most reliable in noisy and resource-constrained code-mixed environments, while deep learning and transformer architectures require greater adaptation to succeed.

Keywords: Artificial Intelligence (AI); Natural Language Processing (NLP); Sentiment Analysis; Code-Mixed Text; Tamil-English; Machine Learning (ML)

INTRODUCTION

Code-mixing, the blending of two or more languages within a single sentence or phrase, has become increasingly common in digital communication, especially on platforms such as YouTube, Twitter, and WhatsApp. This phenomenon is particularly prevalent in multilingual societies, including India and Sri Lanka, where speakers routinely combine English with regional languages such as Tamil (1). Among Tamil-English bilinguals, a distinct form of code-mixed writing has emerged. Often, Tamil is written in Roman script for ease of input, especially on mobile devices that default to English keyboards (2, 3). This style of communication departs from conventional monolinguistic structures and often includes creative spellings and phonetic approximations.

This informal and hybrid language style poses unique challenges for natural language processing (NLP) systems. Tools designed for monolingual or even multilingual text typically assume consistent grammatical structure and standardized spelling (4, 5). In contrast, code-mixed text exhibits unpredictable shifts in language, orthographic variation, and informal transliteration practices. These complications are

Corresponding author: Mohnish Sivakumar, E-mail: mohnu10@gmail.com.

Accepted November 13, 2025
https://doi.org/10.70251/HYJR2348.36574580

further amplified for low-resource languages like Tamil, which lack the extensive annotated corpora and language tools available for widely studied languages such as English or Mandarin (6).

Existing sentiment analysis models are largely built on monolingual corpora and assume consistent linguistic norms. When applied to code-mixed text, particularly Tamil-English data written in Roman script, these models often fail to capture meaning accurately (2). The root of the issue lies in both the linguistic nature of code-mixing and the scarcity of tailored resources. Phonetic spelling varies from user to user, and transliteration is not standardized. In addition, mixed-language texts often contain elements like emojis, abbreviations, and internet slang, all of which contribute to lexical noise (2).

While multilingual models have shown promise in recent years, they are not optimized for handling inconsistent or informal code-mixing. Their vocabulary coverage may include both English and Tamil, but they lack mechanisms to reconcile non-standard forms or understand mixed syntax (4). As a result, the sentiment classification of such text remains unreliable, particularly in informal and noisy domains such as social media.

This paper aims to compare modern machine learning models to traditional methods to determine which is most adept at handling code-mixed (Tamil-English) inputs. Classical machine learning models, deep learning architectures, and transformer-based approaches are all evaluated on a curated dataset. In addition, the potential for lexicon-based sentiment scores to meaningfully complement statistical classifiers is examined. The broader objective is to identify practical methods for improving sentiment analysis in low-resource, code-mixed language settings.

LITERATURE REVIEW

Tamil Sentiment Analysis

Sentiment analysis for Tamil has historically been limited by the lack of large-scale annotated resources. Lexicon-based approaches initially dominated, often relying on sentiment mapping from English to Tamil via bilingual dictionaries or translation services. Kannan *et al*. (7) laid early groundwork by constructing Tamil SentiWordNet, leveraging English sentiment labels through translated glosses and human verification. This lexicon continues to support lexically driven pipelines, although it remains sensitive to dialect, context, and domain shifts.

More sophisticated pipelines have emerged in recent years. Ramanathan *et al*. (8) proposed a layered sentiment analysis system combining baseline Term Frequency–Inverse Document Frequency (TF-IDF), domain-specific ontologies, Contextual Semantic Sentiment Analysis (CSSA), and an enhanced Tamil SentiWordNet. This hybrid pipeline achieved 77.9% accuracy on Tamil tweets for the film *Petta*, far outperforming the TF-IDF-only baseline of 34.6%. The work underscored the benefits of integrating context-aware semantics and informal language expansion into lexicon-driven models.

Deep learning methods have also shown potential in Tamil sentiment tasks. While such models demand large corpora, even relatively shallow neural networks (e.g., single-layer Long Short-Term Memory (LSTM) models) have proven effective on short-form text when paired with domain-aligned embeddings and careful preprocessing (9). However, these models still struggle with transliterated or noisy inputs.

Code-mixed Sentiment Analysis

Code-mixing presents an added challenge to traditional sentiment models due to lexical inconsistencies, lack of standardized transliteration, and unpredictable grammatical switches. To address this, Chakravarthi *et al*. (2) released a benchmark dataset of Tamil-English code-mixed YouTube comments as part of the FIRE-DravidianCodeMix shared task. Their dataset included over 4,000 annotated sentences, and their experiments showed that Multilingual Bidirectional Encoder Representations from Transformers (mBERT) achieved the highest macro F1-score (58.52%) on the Tamil-English subset. This outperformed classical methods like Support Vector Machines (SVM) (51.98%) and Bidirectional LSTM (BiLSTM) (53.44%), demonstrating that transformer-based models are better suited to capturing context in phonetically spelled, mixed-language text.

Building on the challenge of code-mixed sentiment detection, Raveendirarasa and Amalraj (10) proposed a hybrid approach using dictionary-based preprocessing and a subword-level LSTM trained on a small curated corpus of 1,500 comments. Their system reached 74.6% accuracy by combining manually developed lexicons and Part-of-Speech (POS) tagging with machine learning. The authors emphasized the role of structured preprocessing for handling inconsistent Romanized Tamil tokens—particularly in settings where training data is limited.

Ramanathan *et al*. (8) approached the problem with a more lexicon-heavy perspective, incorporating a semantic ontology and enhanced sentiment lexicon. Although their work focused on monolingual Tamil tweets, the strategies they employed—such as sentiment inversion based on syntactic cues—are also relevant in mixed-language contexts where negation and informal grammar often distort surface polarity.

Multilingual Transformers and Code-Mixed Pretraining

While early multilingual models like mBERT showed improvements over classical baselines, more recent transformer architectures have significantly advanced code-mixed sentiment detection. Krasitskii *et al*. (11) conducted a comprehensive comparison of models including RemBERT, IndicBERT, mT5, and XLM-RoBERTa. Their results highlighted RemBERT as the most effective model for Tamil-English code-mixed data, achieving 87.5% accuracy and an F1-score of 86.4%. The evaluation was conducted over several datasets, including CMD-Tamil and DravidianCodeMix, using both BLEU scores and standard sentiment metrics. Their findings emphasize that multilingual pretraining is not sufficient alone; robustness to transliteration and subword variation is critical.

These results build on findings from shared tasks such as DravidianCodeMix, which showed that code-mixed pretraining or fine-tuning on noisy, mixed-language corpora consistently improves sentiment model performance (12). General-purpose multilingual models underperform when not adapted to the linguistic quirks of code-mixed data, especially Romanized input with inconsistent spelling and syntax.

Hybrid and Preprocessing Approaches

Across studies, preprocessing emerges as a pivotal stage for success. Common strategies include transliteration to native script, normalization of phonetic variants, and filtering of non-textual noise. In several cases, sentiment performance improved significantly when models were fed cleaned and language-aligned input (2), (10). Ramanathan *et al*. (8) and Krasitskii *et al*. (11) both integrated domain-specific sentiment lexicons into their models, showing that even transformer-based architectures benefit from explicit lexical priors in code-mixed settings.

Hybrid systems, such as lexicon-enhanced XGBoost pipelines, demonstrate how linguistic knowledge and statistical modeling can complement each other. For example, incorporating VADER scores or Tamil SentiWordNet features into TF-IDF-based models allows machine learning systems to make better use of prior sentiment cues. While this technique has yet to outperform the latest transformers, it provides more interpretable and lightweight alternatives, particularly in resource-constrained or low-latency applications.

METHODS AND MATERIALS

Dataset

The primary dataset used in this paper is the FIRE2020 Tamil-English Sentiment Analysis Dataset (2), which contains code-mixed YouTube comments collected from South Indian channels. The Tamil-English portion of the dataset includes over 4,000 examples, each manually labeled into one of five sentiment categories: positive, negative, neutral, mixed sentiment, and other-language. For the purposes of this paper, only the positive, negative, and neutral labels were retained, resulting in a simplified three-class classification problem. This format was supported by the datasets used and matches with previous experiments, allowing for smoother testing and consistency.

All comments in the dataset are written in Roman script or were transliterated from Tamil script to Roman script. This reflects how users typically input Tamil text using English keyboards, often resulting in informal spellings, inconsistent transliteration, and ungrammatical structures. Many comments switch between Tamil and English words unpredictably and exhibit features typical of social media text, including emojis, abbreviations, repeated characters, and slang.

To support the lexicon-based component of the analysis, a modified version of the VADER sentiment lexicon was constructed. Google Translate was used to add Tamil words to the dataset, allowing for code-mixed processing.

Preprocessing Pipeline

To ensure consistency and usability across modeling stages, a structured preprocessing pipeline was applied to the dataset. The raw data, comprising Tamil-English code-mixed comments annotated with sentiment labels, was initially sourced from tab-separated value (TSV) files. Preliminary cleaning involved the removal of metadata fields not required for training and the normalization of label strings to eliminate leading or trailing whitespace. The training and test

partitions were merged to apply uniform preprocessing procedures. Sentiment labels were converted from categorical to numerical format using ordinal label encoding techniques, facilitating compatibility with supervised learning algorithms. Following this, the data was stratified and split into training, validation, and test sets using a 70-10-20 ratio with stratification to preserve the original class distribution across splits. To support reproducibility and simplify downstream experimentation, the resulting subsets were saved as TSV files in a shared directory. These preprocessed files were subsequently reloaded for use in both classical and deep learning model pipelines. This preprocessing strategy ensured that the input data was clean, standardized, and readily reusable across all modeling experiments.

Models (Table 1)

Table 1. Classification of Sentiment Analysis Models Across Lexicon-Based, Machine Learning, Deep Learning, Transformer-Based, and Hybrid Categories

Type	Model
Lexicon-Based	VADER (lexicon-based)
Classical Machine Learning	Naive Bayes
Classical Machine Learning	Logistic Regression
Classical Machine Learning	Support Vector Machine
Machine Learning	XGBoost
Deep Learning Model	LSTM
Transformer-Based Model	RemBERT + MLP
Hybrid Model	VADER + XGBoost
Hybrid Model	VADER + Bayes

Lexicon-Based Model

The baseline lexicon model was built using VADER, a rule-based sentiment analyzer designed for social media text. VADER calculates sentiment polarity based on word intensity scores and syntactic cues such as negation and punctuation. The original English lexicon was expanded with transliterated Tamil sentiment words commonly found in the training data. The model computed a compound sentiment score for each comment based on the scores given to each word, and a rule-based mapping converted the score into a discrete sentiment class.

Although VADER does not model context beyond token-level cues, it provides a transparent baseline for evaluating the impact of lexicon augmentation and preprocessing.

Classical Machine Learning Models

Three classical machine learning models were tested using TF-IDF representations of the preprocessed text: Multinomial Naive Bayes, Logistic Regression, and Support Vector Machines. These models were trained using scikit-learn and evaluated using stratified three-fold cross-validation. Feature extraction included unigrams, bigrams, and trigrams, with token frequency thresholds set to ignore rare words. Different models used different ngrams based on which were more optimal.

The purpose of these models was to establish baseline performance for shallow learning techniques. They require minimal resources and are relatively robust to small datasets, but lack the ability to model long-range dependencies or subword variation.

Deep Learning Models

A Long Short-Term Memory (LSTM) network was implemented to capture sequential patterns in the comments. This model uses text tokenization and padding, an embedding layer, two LSTM layers, and dropout layers, compiled with an Adam optimizer and trained with early stopping.

Training was conducted using categorical cross-entropy loss with early stopping based on validation accuracy. The LSTM model was selected for its effectiveness in handling variable-length input and informal text, characteristics that define code-mixed data.

Transformer-Based Models

To evaluate the potential of multilingual pretrained transformers, RemBERT with MLP was chosen. This model was chosen based on previous findings indicating its strong performance on South Asian code-mixed text (11).

Tokenization was handled using each model's native tokenizer, which allowed subword segmentation of phonetic variations and hybrid word forms. Each model was trained using the HuggingFace Transformers library with a learning rate schedule and early stopping. Inputs were limited to 128 tokens to match the dataset's average length and reduce memory load.

This model uses a pre-trained tokenizer and model

with a fixed sequence length, and was tuned on learning rate, number of epochs, and batch size.

Hybrid Model

Finally, two hybrid approaches were tested by combining the lexicon-based VADER-derived sentiment scores with TF-IDF features, fed into an XGBoost classifier. This model utilizes TF-IDF features with a variable number of max features, and an XGBoost classifier tuned on the number of estimators, learning rate, and maximum depth.

Additionally, the TF-IDF features and the sentiment scores were also fed into a Naive Bayes model. This model uses a variable number of max features and ngram ranges, and a Multinomial Naive Bayes classifier tuned on the smoothing parameter. The goal was to integrate rule-based sentiment priors into a statistical model capable of learning discriminative patterns. This combination allowed the model to benefit from both symbolic sentiment cues and contextual token-level representations.

The inclusion of lexicon features as an additional dimension was intended to improve robustness on sparse or noisy examples, particularly those dominated by informal Tamil sentiment terms not well represented in pretrained embeddings.

RESULTS AND DISCUSSION

As shown in Table 2, evaluation showed that classical machine learning models provided the most stable performance on the FIRE2020 Tamil-English dataset. Logistic Regression, Naive Bayes, and SVM all achieved test accuracies close to 69%, with weighted F1-scores around 0.60–0.61. XGBoost performed slightly below these baselines, reaching 68.2% accuracy and a weighted F1-score of 0.57. Hybrid approaches that integrated VADER lexicon features into statistical classifiers have similar results. VADER + XGBoost reached 68.9% accuracy and a weighted F1-score of 0.60, comparable to the best-performing classical models, while VADER + Naive Bayes performed slightly lower at 68.4% accuracy and 0.58 weighted F1. Deep learning approaches were less consistent: the LSTM attained a comparable accuracy of 67.1% but almost exclusively predicted the majority "positive" class, resulting in very low macro-level performance. Similarly, RemBERT, despite being identified in prior work as one of the strongest models for Tamil-English code-mixed text, only reached 67.1% accuracy and a weighted F1-score of 0.54. The lexicon-based VADER baseline, expanded with transliterated Tamil entries, produced lower overall performance compared to the statistical models, with an accuracy of 64.24% and a weighted F1 of 0.343.

While there was no substantial variation between most of the models, all of them outperformed the lexicon-based baseline. These findings highlight that while transformer-based and deep learning models have shown superior performance on larger or better-curated code-mixed corpora (e.g., CMD-Tamil, DravidianCodeMix), their advantage diminishes on smaller, noisier datasets such as FIRE2020, perhaps due to their complex nature and immense parameters

Table 2. Summarizes the performance metrics of all tested models, including accuracy, precision, recall, and weighted F1-score

Model	Accuracy %	Weighted F1	Precision	Recall
VADER (lexicon-based)	64.24	0.343	0.526	0.280
Naive Bayes	69.16	0.602	0.634	0.692
Logistic Regression	**69.36**	**0.611**	**0.640**	**0.694**
Support Vector Machine	69.32	0.600	0.634	0.693
XGBoost	68.24	0.577	0.617	0.682
LSTM	67.07	0.540	0.450	0.671
RemBERT + MLP	67.07	0.538	0.450	0.672
VADER + XGBoost	68.88	0.603	0.614	0.689
VADER + Bayes	68.40	0.575	0.632	0.684

to train and generalize. In contrast, lightweight classical models—and to a slightly lesser extent, hybrid approaches—displayed higher precision, recall, and f1 scores, remaining robust and competitive in resource-constrained conditions.

CONCLUSION

This paper shows that sentiment analysis for Tamil-English code-mixed text depends as much on dataset characteristics as on model architecture. Classical machine learning models such as Logistic Regression, Naive Bayes, and SVM proved most effective on the FIRE 2020 dataset, achieving around 69% accuracy with stable weighted F1-scores. Their strength lies in handling smaller, noisier, and more imbalanced data without requiring extensive preprocessing. In particular, probabilistic and linear models may remain advantageous because they make relatively few assumptions about linguistic regularity, allowing them to capture broad statistical trends even when the signal is sparse or inconsistent. The hybrid approaches tested show that lexicons can provide some support to these traditional methods, but despite the prominence of deep learning, classical approaches continue to offer practical benefits in low-resource and noisy environments.

In contrast, larger transformer-based models like RemBERT, which have demonstrated strong results on curated corpora, where Krasitskii *et al*. (11) reported F1-scores near 87%, seemed to struggle to generalize under FIRE2020's noisier conditions, achieving only ~67% accuracy and F1 ~0.54. This contrast underscores that while transformers excel when trained on large, balanced, and carefully managed datasets, classical models seem more reliable for resource-constrained and noisy settings. Our results align more closely with the incremental improvements observed by Chakravarthi *et al*. (2), rather than the dramatic gains claimed by Krasitskii *et al*., reinforcing the importance of evaluating models across diverse datasets. At the same time, excessive normalization and cleaning, such as aggressive spelling correction or transliteration alignment, risk producing artificially simplified data that does not reflect real-world language use, as well as significant inflation of model performance on data.

Limitations

Several limitations shaped the outcomes of this paper. First, reliance on Google Translate for producing romanized Tamil introduced inconsistencies, as automated transliteration often fails to capture phonetic nuances and colloquial variations present in code-mixed social media text. Similarly, spelling variations and informal script-switching were not always consistently handled, which likely contributed to misclassifications. In addition, while hyperparameter tuning improved model performance, results may still reflect local optima rather than the true best possible configurations, meaning additional search could yield further gains. These issues highlight the challenges of working with inherently noisy, under-resourced code-mixed datasets, where preprocessing choices can significantly shape model performance.

Future Work

Building on these findings, future research should aim to balance preprocessing with model robustness, avoiding excessive normalization while still addressing key noise sources. Promising directions include domain-adaptive pretraining of transformers on social media-specific Tamil-English corpora, which could help bridge the gap between curated benchmarks and real-world conditions. Hybrid approaches that combine lexical and contextual modeling, such as integrating TF-IDF features with transformer embeddings, may also yield improvements by leveraging the complementary strengths of probabilistic and neural methods. Beyond technical modeling, developing standardized transliteration schemes and expanding annotated datasets for code-mixed Tamil-English would reduce inconsistencies and enable more reliable benchmarking. Additionally, more data would allow for the development of more complex deep learning models and neural networks. Together, these efforts would advance the development of sentiment analysis systems that generalize effectively across diverse and noisy code-mixed contexts.

PREPRINT DECLARATION

Mohnish Sivakumar, the sole author of this paper, acknowledge and approve of the submission of this manuscript to the Research Archive of Rising Scholars preprint server.

CONFLICT OF INTEREST

The author declares no conflicts of interest related to this work.

REFERENCES

1. Pratapa A, Bhat RA, Choudhury M, Bali K. Language modeling for code-mixing: the role of linguistic theory-based synthetic data. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*; 2018; p.1543–53. doi:10.18653/v1/P18-1143.
2. Chakravarthi B, Muralidaran V, Priyadharshini R, McCrae JP. Corpus creation for sentiment analysis in code-mixed Tamil–English text. *arXiv preprint* arXiv:2006.00206; 2020. doi:10.48550/arXiv.2006.00206.
3. Sridhar SN. Code-mixing in Indian languages: typological and sociolinguistic aspects. In: Karduna H, Malchukov A, Subbarao P, editors. *Handbook of the South Asian languages*. Cham: Springer; 2020; p.359–87. doi:10.1007/978-3-030-46010-3_13.
4. Solorio T, Blair E, Maharjan L, Bethard S, *et al*. Overview for the first shared task on language identification in code-switched data. In: *Proceedings of the 1st Workshop on Computational Approaches to Code Switching*; 2014; p.62–72. https://doi.org/10.3115/v1/W14-3907
5. Aguilar G, Kar S, Solorio T, González FA. LinCE: a centralized benchmark for linguistic code-switching evaluation. In: *Proceedings of the 12th Language Resources and Evaluation Conference (LREC)*; 2020. p.1803–13.
6. Bhat RA, Choudhury M, Malu A, Bali K. Universal dependency parsing for Hindi–English code-switching. In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*; 2018; p.987–98. Available from: https://aclanthology.org/N18-1090/ (accessed 2025-10-12). https://doi.org/10.18653/v1/N18-1090
7. Kannan A, Mohanty F, Mamidi R. Towards building a SentiWordNet for Tamil. In: *Proceedings of the 13th International Conference on Natural Language Processing (ICON)*; 2016. NLP Association of India. Available from: https://aclanthology.org/W16-6305/ (accessed 2025-10-12).
8. Ramanathan V, Thirunavukkarasu M, Thamarai S. Sentiment analysis: an approach for analysing Tamil movie reviews using Tamil tweets. In: *Research advances in modern science*. Book Publisher International; 2021; 3: 44–55. doi:10.9734/bpi/ramrcs/v3/4845F.
9. Padmamala R, Prema VM. Sentiment analysis of online Tamil contents using recursive neural network models approach for Tamil language. In: *Proceedings of the 2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*; 2017 Aug; p.28–31. doi:10.1109/ICSTM.2017.8089122.
10. Raveendirarasa V, Amalraj CRJ. Sentiment analysis of Tamil–English code-switched text on social media using subword-level LSTM. In: *Proceedings of the 5th International Conference on Information Technology Research (ICITR)*; 2020; p.1–5. doi:10.1109/ICITR51448.2020.9310817.
11. Krasitskii M, Kolesnikova O, Chanona Hernandez L, Sidorov G, Gelbukh A. Advancing sentiment analysis in Tamil–English code-mixed texts: challenges and transformer-based solutions. In: *Proceedings of the 5th International Conference on Natural Language Processing for Digital Humanities (NLP4DH)*; 2025; p.305–12. https://doi.org/10.18653/v1/2025.nlp4dh-1.27
12. Chakravarthi BR, Priyadharshini R, Thavareesan S, Chinnappa D, Thenmozhi D, Sherly E, *et al*. Findings of the sentiment analysis of Dravidian languages in code-mixed text. *FIRE / Dravidian-CodeMix Shared Task Report*; 2021. Available from: https://arxiv.org/abs/2111.09811 (accessed 2025-10-12).