INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 4, April 2026 | ISSN: 2456-4184 | IJNRD.ORG

HINGLISH SENTIMENT ANALYSIS USING MACHINE LEARNING: A FASTTEXT-BILSTM APPROACH FOR CODE-MIXED SOCIAL MEDIA TEXT

[1]Ankit Kaushik, [2]Rishabh Nehra, [3]Rishabh Tyagi

[1,2,3]Department of Computer Science & Engineering

International Research Submission, Academic Year 2025–26

***Abstract* —** Social media communication in India has evolved into a unique linguistic phenomenon known as Hinglish—a code-mixed blend of Hindi and English written primarily in Roman script. While Hinglish offers high emotional expressiveness, it creates a low-resource environment for traditional NLP tools due to non-standardized grammar, diverse phonetic spelling variations (e.g., 'karo' vs 'kro' vs 'karro'), and the pervasive use of regional slang and abbreviations. This paper proposes a robust, end-to-end deep learning architecture that combines a domain-specific Hinglish Normalization Lexicon, FastText sub-word embeddings (300-dimensional), and a stacked Bidirectional Long Short-Term Memory (BiLSTM) network for three-class sentiment classification. Our model is trained and evaluated on a meticulously curated, class-balanced dataset of 3,500 annotated samples sourced from Twitter and YouTube. The proposed system achieves a state-of-the-art accuracy of 86.4%—an 18.1% improvement over the Naive Bayes baseline—while maintaining a real-time end-to-end inference latency of 76.4 ms per sentence, confirming its viability for live social media stream analysis.

***Index Terms* —** *Hinglish, Code-Mixed NLP, Sentiment Analysis, FastText, BiLSTM, Social Media Mining, Deep Learning, Indian Languages, Phonetic Normalization*

I. INTRODUCTION

The rapid digitalization of the Indian subcontinent has produced an unprecedented communication revolution. With over 700 million active internet users, India now constitutes the world's second-largest online population. Platforms such as Twitter, YouTube, Instagram, and WhatsApp process billions of messages daily, and a significant proportion of this content is code-mixed—users fluidly transitioning between Hindi and English within a single utterance, sometimes even within a single word. This hybrid register, colloquially termed Hinglish, enables high emotional expressiveness and is particularly prevalent among the 18–35 age demographic. However, it creates a fundamentally low-resource environment for conventional Natural Language Processing (NLP) systems that are designed primarily for monolingual, grammatically standardized corpora.

Sentiment analysis—the computational task of identifying and classifying the emotional polarity of a piece of text—holds enormous practical value for businesses, policymakers, and researchers. Real-time public opinion mining on product reviews, political discourse, and crisis communication can drive data-informed decisions. However, deploying such systems in the Indian social media context requires overcoming the unique linguistic challenges that Hinglish presents, challenges that existing monolingual English or Hindi systems are fundamentally ill-equipped to handle.

1.1 The Problem of Linguistic Fluidity

Classical sentiment analysis tools rely on fixed lexicons or static word-level embeddings such as Word2Vec or GloVe, which operate on the assumption that each word maps to a unique, fixed vector. In Hinglish, this assumption breaks down catastrophically. A single concept may manifest in dozens of phonetically equivalent but orthographically distinct forms—'karo', 'kro', 'karro', and 'krro' all represent the same imperative verb. Furthermore, sentiment is frequently context-dependent and may shift polarity mid-sentence. The utterance 'Phone accha hai but battery bekar hai' embeds contradictory polarities within a single sentence. Resolving the dominant sentiment requires modelling long-range syntactic and semantic dependencies that shallow classifiers are incapable of capturing. Compounding this challenge is the widespread use of sarcasm and irony, where lexically positive tokens carry a decidedly negative communicative intent.

1.2 Motivation and Significance

Despite the scale of Hinglish content on Indian social media, the field remains significantly under-resourced relative to its linguistic complexity. Most publicly available sentiment datasets cover formal English or Devanagari Hindi, leaving the vast Romanized Hinglish space largely unaddressed. Commercial sentiment APIs (Google Natural Language, AWS Comprehend) routinely misclassify Hinglish text, as they are trained predominantly on English corpora. This research directly addresses this gap by constructing a purpose-built dataset, a domain-specific preprocessing pipeline, and a neural architecture validated empirically against multiple baselines.

1.3 Research Objectives

This study pursues four principal objectives: (1) Dataset Engineering — Construct a balanced, three-class (Positive / Negative / Neutral) Hinglish corpus from real-world social media with verified inter-annotator agreement. (2) Normalization Framework — Design a domain-specific preprocessing layer that maps thousands of phonetic variants, slang terms, and emoji to standardized forms. (3) Comparative Evaluation — Rigorously benchmark classical ML models (Naive Bayes, SVM, Random Forest) against the proposed deep BiLSTM architecture on identical data splits. (4) Failure Analysis — Identify and characterize systematic error patterns, with particular focus on sarcasm and irony detection in code-mixed text.

The remainder of this paper is organized as follows: Section 2 surveys related literature; Section 3 describes the proposed methodology in detail; Section 4 presents implementation details and experimental results; and Section 5 concludes with future directions.

II. LITERATURE REVIEW

Code-mixed NLP has attracted growing academic interest over the past decade, with a notable acceleration following the widespread adoption of social media in multilingual societies. Early efforts focused on language identification—distinguishing Hindi tokens from English tokens within a sentence—as a prerequisite for any downstream task. More recent work has attempted sentiment classification directly on code-mixed text, with varying degrees of success.

Sharma et al. (2021) demonstrated that sub-word tokenization strategies outperform word-level tokenization for Indian code-mixed text, laying the theoretical foundation for our use of FastText. Concurrent work by Patwa et al. (2020) released the SemEval-2020 Task 9 dataset for Hinglish sentiment, establishing an early benchmark but noting that the dataset was limited in size and did not capture the full phonetic diversity of organic social media text. A systematic survey of representative works from 2022 to 2024 is presented in Table 1, highlighting the specific limitations our approach is designed to overcome.

Paper / Authors	Year	Methodology	Critical Gaps Identified
Joshi et al.	2022	SVM + TF-IDF	Only 1,200 samples; no Neutral class; failed on phonetic variants ('bhaut' vs 'bohot'); no sub-word handling.
Prabhu et al.	2022	CNN (Char-level)	Captured local character n-grams but missed global semantic context; high inference latency on CPU.
Lal et al.	2023	mBERT (Transformer)	Pre-trained on Devanagari; poor performance on Romanized phonetic Hindi slang common in Hinglish.
Kumar et al.	2024	RNN / Simple LSTM	Vanishing gradient problem in long Hinglish comments; single-direction processing missed right-to-left context.

Table 1: Critical Gap Analysis of Prior Literature (2022–2024)

The overarching observation from this survey is that no existing work simultaneously addresses all three core challenges of Hinglish NLP: (1) phonetic variation at the character level, (2) long-range contextual dependencies across code-switched tokens, and (3) sufficient dataset scale with a balanced three-class label distribution. Our proposed system is specifically architected to resolve all three challenges in a unified pipeline.

III. PROPOSED METHODOLOGY

The proposed pipeline is structured into five rigorous and sequential phases, each designed to address a specific challenge inherent to Hinglish sentiment analysis. Figure 4 conceptually illustrates the end-to-end flow: raw social media text enters the Normalization Lexicon, passes through FastText embedding, is fed into the BiLSTM network, and a final Softmax layer produces the sentiment probability distribution.

3.1 Data Acquisition and Dataset Engineering

A primary dataset of 3,500 unique Hinglish samples was curated using Python-based web scraping with the Tweepy API (Twitter) and the YouTube Data API v3. Search queries targeted high-volume Indian product categories (smartphones, food delivery, online shopping) and entertainment topics (Bollywood, cricket) to ensure lexical diversity. All personally identifiable information was anonymized prior to annotation in compliance with ethical research standards.

Three sentiment classes were assigned: Positive, Negative, and Neutral. Class balance was actively managed during scraping to achieve near-uniform distribution: Positive 35% (1,225 samples), Negative 33% (1,155 samples), and Neutral 32% (1,120 samples). This balance is critical—imbalanced datasets cause classifiers to exhibit bias toward majority classes, artificially inflating overall accuracy while degrading recall for minority classes.

Annotation quality was ensured through a triple-annotation protocol: three independent native Hinglish speakers labelled each sample, and the final label was determined by majority voting. Annotators were provided a detailed labelling rubric with 50 worked examples spanning sarcasm, negation, and mixed-polarity sentences. The measured inter-annotator agreement (Cohen's Kappa, $\kappa = 0.74$) falls in the 'Substantial Agreement' band (0.61–0.80), confirming annotation reliability. The dataset was partitioned using a fixed random seed into 80% training (2,800 samples) and 20% testing (700 samples).

Sentiment Class	Sample Count	Percentage
Positive	1,225	35%
Negative	1,155	33%
Neutral	1,120	32%

Table 2: Dataset Class Distribution (Total N = 3,500)

3.2 Specialized Preprocessing: The Hinglish Normalization Lexicon

Raw social media text in Hinglish is highly noisy: spelling is non-standardized, grammar is informal, and the signal-to-noise ratio is low. Standard NLP preprocessing pipelines (stop-word removal, lowercasing, punctuation stripping) are insufficient and in some cases counterproductive—for instance, stripping 'not' as a stop word inverts sentiment polarity. The Hinglish Normalization Lexicon is a domain-specific preprocessing layer developed specifically to address these challenges.

The lexicon operates through four sequential sub-processes: Step 1 — Phonetic Mapping: A hand-crafted lookup table of 4,200 phonetic variant entries maps non-standard spellings to their canonical base forms. Variants were compiled by analysing a 50,000-sample unlabelled Hinglish corpus and grouping by phonetic Soundex codes. Step 2 — Slang and Abbreviation Expansion: Textual abbreviations and shorthand are expanded to full forms. A secondary dictionary of 800 Hindi slang terms with their sentiment polarity values is used to augment the base lexicon. Step 3 — Negation Handling: Negation markers ('nahi', 'mat', 'na', 'never', 'not') are identified and a NEGATION_PREFIX tag is appended to the subsequent three tokens. Step 4 — Emoji Sentiment Preservation: Each emoji is mapped to a descriptive sentiment token via a 512-entry emoji dictionary, preserving affective signal that is often the primary sentiment carrier in informal Hinglish text.

3.3 Feature Engineering: FastText Sub-word Embeddings

Word embedding quality is the single most consequential design decision in any NLP pipeline. We evaluated three candidate embedding strategies—Word2Vec (skip-gram), GloVe, and FastText—and selected FastText on the basis of its theoretical and empirical superiority for morphologically rich, noisy text such as Hinglish.

FastText's core innovation is representing each word not as a single atomic vector but as the sum of vectors for its constituent character n-grams (subwords of length 3 to 6 characters). The similarity score between a word w and a context word c is computed as: $s(w, c) = \Sigma_{g \in G_w} z_g^T \cdot v_c$, where G_w denotes the set of all n-grams of word w, z_g is the vector representation of n-gram g, and v_c is the embedding vector of the context word c. This means that even if a specific misspelling such as 'bhaut' was never encountered during training, the model can construct a meaningful representation by summing the vectors of its constituent n-grams—'bha', 'hau', 'aut'—which overlap significantly with the canonical form 'bahut'.

We utilized Facebook's pre-trained FastText vectors (300-dimensional) trained on the Common Crawl and Wikipedia corpora. These vectors are loaded as the initial weights of the embedding layer and frozen (trainable=False) during training to prevent vocabulary drift on our relatively small dataset of 3,500 samples.

Property	Word2Vec	GloVe	FastText (Ours)
OOV Word Handling	UNK token (info lost)	UNK token (info lost)	N-gram fallback (info preserved)
Phonetic Variant Support	None — needs exact match	None — needs exact match	Native via shared n-grams
Morphology Awareness	No	No	Yes — character-level

Table 3: Embedding Strategy Comparison — FastText vs Alternatives

3.4 Neural Architecture: Stacked BiLSTM

Long Short-Term Memory (LSTM) networks address the vanishing gradient problem of vanilla RNNs through gating mechanisms—the input gate, forget gate, and output gate—that allow the network to selectively retain or discard information across arbitrarily long sequences. A Bidirectional LSTM (BiLSTM) augments this by processing the input sequence in both the forward (left-to-right) and backward (right-to-left) directions simultaneously and concatenating the resulting hidden states. This is particularly valuable for code-mixed text, where the sentiment-determining word may appear before or after its contextual modifiers depending on code-switching patterns.

The proposed six-layer architecture is defined as follows: (1) Embedding Layer (300-dim, frozen) — Pre-trained FastText weights, trainable=False prevents vocabulary drift. (2) SpatialDropout1D (rate=0.4) — Drops entire feature maps rather than individual neurons, forcing the model to learn multiple independent semantic paths. (3) BiLSTM Layer 1 (128 units, return_sequences=True) — First recurrent layer with dropout=0.3 on input connections; recurrent_dropout=0.2 on recurrent connections. (4) BiLSTM Layer 2 (64 units) — Second recurrent layer consolidates the temporal representations into a fixed-size context vector. (5) Dense Layer (64 units, ReLU activation) — Non-linear feature extraction and dimensionality consolidation. (6) Output Layer (3 units, Softmax activation) — Produces a probability distribution over the three sentiment classes, optimized with categorical cross-entropy loss and the Adam optimizer.

3.5 Technical Environment

Component	Specification
CPU	Intel Core i7-12700H (14 Cores) / AMD Ryzen 9 5900X
GPU	NVIDIA GeForce RTX 3070 Ti — 8 GB Dedicated VRAM
RAM / Storage	32 GB DDR4 @ 3200 MHz | 1 TB NVMe SSD (>3500 MB/s R/W)
Deep Learning	Python 3.9.12 | TensorFlow 2.12.0 | Keras (integrated)
NLP Libraries	NLTK 3.7, Spacy 3.4, FastText (Facebook AI Research)

Component	Specification
Data Science	Pandas 1.5.3, NumPy 1.23.5, Scikit-learn 1.2.2
Visualization	Matplotlib 3.7.1, Seaborn 0.12.2

Table 4: Experimental Hardware & Software Configuration

IV. IMPLEMENTATION AND RESULTS

4.1 Training Configuration

The model was trained for a maximum of 50 epochs with an early stopping callback (patience=5, monitored on validation loss) to prevent overfitting. The batch size was set to 64, and the learning rate was initialized at 0.001 with an exponential decay schedule (decay_rate=0.95 per epoch). The maximum sequence length was set to 50 tokens, with shorter sequences zero-padded and longer sequences truncated from the right. The vocabulary size after preprocessing was approximately 12,400 unique tokens. Total trainable parameters: 1.24 million (embedding layer frozen; only BiLSTM + Dense layers trained).

4.2 Accuracy Comparison with Prior Literature

Figure 1 presents a direct side-by-side comparison of the proposed BiLSTM's accuracy against prior literature models (2022–2024) and the classical ML baselines evaluated in this study. The proposed model achieves the highest accuracy (86.4%) among all compared systems, outperforming the closest competitor—Random Forest at 81.1%—by 5.3 percentage points, and surpassing the best reported transformer-based result (Lal et al., 2023, mBERT: 74.8%) by 11.6 percentage points.

[Fig. 1 — Accuracy Comparison: Prior Literature (2022–2024) vs This Study]

4.3 Quantitative Performance Metrics

All four models were evaluated on the identical 20% hold-out test set comprising 700 samples (245 Positive, 231 Negative, 224 Neutral). Table 5 reports the comprehensive metric profile including accuracy, macro-averaged precision, recall, and F1-score. The BiLSTM achieves consistent performance across all metrics, indicating balanced classification across all three classes.

Classification Model	Accuracy (%)	Precision	Recall	F1-Score
Naive Bayes (Multinomial)	68.3	0.67	0.67	0.67
Support Vector Machine	78.6	0.78	0.78	0.78
Random Forest Classifier	81.1	0.81	0.80	0.81
Proposed Deep BiLSTM (Best)	**86.4**	**0.86**	**0.86**	**0.86**

Table 5: Comparative Performance Metrics (20% Hold-out Test Set, N=700)

4.4 Per-Class F1-Score Analysis

Figure 3 disaggregates F1-scores by sentiment class across all evaluated models. The Negative class is consistently the most challenging across all systems—a well-documented phenomenon in Hinglish NLP attributed to the high prevalence of sarcasm, negation, and understatement in critical reviews. Despite this inherent difficulty, the proposed BiLSTM achieves F1 = 0.85 on the Negative class, representing a gain of 22 points over Naive Bayes and 10 points over Random Forest on this specific class.

[Fig. 3 — Per-Class F1-Score Comparison Across All Models]

4.5 Training Dynamics and Convergence Analysis

Figure 2 presents the training and validation accuracy and cross-entropy loss curves across 35 epochs. Several key observations emerge from this analysis. First, the model converges smoothly near epoch 27, after which validation accuracy plateaus at 86.4%. Second, and critically for generalizability, the gap between training accuracy (89.2%) and validation accuracy (86.4%) remains below 3% throughout training. This tight gap indicates that the model generalizes well and is not overfitting—a direct result of the SpatialDropout1D regularization and the frozen embedding layer. Third, the validation loss curve mirrors the training loss without diverging, confirming that early stopping triggered appropriately and prevented over-training.

[Fig. 2 — Training vs Validation: Accuracy (left) and Cross-Entropy Loss (right) over 35 Epochs]

4.6 Inference Latency Profile

For a sentiment analysis system to be practically deployable in a real-time social media monitoring context, it must process incoming text with minimal latency. Table 6 profiles the end-to-end inference time broken down by pipeline stage. The total latency of 76.4 ms per sentence comfortably satisfies the sub-100 ms threshold for real-time applications and is well within the processing capacity required to handle live Twitter stream data.

Pipeline Stage	Avg. Latency (ms)	Hardware
Preprocessing — Regex + Lexicon Lookup	12.4 ms	CPU
FastText Sub-word Embedding Generation	45.8 ms	CPU / GPU

Pipeline Stage	Avg. Latency (ms)	Hardware
BiLSTM Neural Network Inference	18.2 ms	GPU (CUDA)
Total End-to-End Pipeline	**76.4 ms**	**Full Optimised Stack**

Table 6: End-to-End Inference Latency Profile

4.7 Qualitative Failure Analysis — The Sarcasm Problem

A rigorous analysis of the 13.6% misclassified samples (95 of 700 test instances) reveals that 68% of errors involve sarcasm or irony—sentences where the lexical sentiment polarity is the inverse of the communicative intent. We term this the 'Sentiment Flip' phenomenon. Table 7 presents representative failure cases with detailed linguistic analysis.

Input Sentence (Hinglish)	True Label	Model Output	Linguistic Failure Mode
"Wah! Kya delivery di hai, sirf 7 din lag gaye"	Negative	Positive	Exclamatory tokens 'Wah' and 'Kya' dominated attention. '7 din lag gaye' as temporal irony undetected.
"Bhai tu toh bohot bada genius hai!" (as insult)	Negative	Positive	Positive adjective 'genius' and intensifier 'bohot bada' overwhelmed contextual sarcasm signal.

Table 7: Sarcasm-Induced Misclassification Examples — Sentiment Flip Phenomenon

These failure cases confirm a fundamental limitation of sequence-only models: sarcasm resolution requires contextual pragmatic awareness that extends beyond the syntactic and semantic information encoded in token sequences. Attention mechanisms—specifically self-attention (as in Transformers) that can model long-range token interactions—represent the most promising architectural direction for addressing this gap.

4.8 Discussion

The experimental results validate the central hypothesis of this research: that a domain-specific preprocessing pipeline combined with sub-word embeddings and a bidirectional recurrent architecture provides a superior solution for Hinglish sentiment classification compared to either classical ML models or general-purpose transformer models not specifically adapted for Romanized code-mixed text. The 86.4% accuracy achieved represents the highest reported figure for three-class Hinglish sentiment classification on a dataset of comparable size and phonetic diversity, to the best of our knowledge.

The ablation between classical and deep models is particularly instructive. The 7.8-point accuracy gap between Random Forest (81.1%) and BiLSTM (86.4%) underscores the value of temporal context modelling. The further 11.6-point gap over the mBERT baseline (Lal et al., 2023: 74.8%) demonstrates that general multilingual transformers pre-trained on formal Devanagari text are significantly disadvantaged on informal Romanized Hinglish without task-specific fine-tuning.

4.9 Future Research Directions

MuRIL Fine-Tuning — Evaluate Google's Multilingual Representations for Indian Languages (MuRIL) model, specifically pre-trained on Romanized Indian language data, against the proposed BiLSTM using identical train/test splits. Dialectal Robustness — Extend the corpus to include Bambaiya Hindi (Mumbai), Delhi NCR slang, and Bihari-accented Hinglish to ensure geographical robustness and reduce urban demographic bias. Aspect-Based Sentiment — Develop an entity-level system capable of detecting sentiment toward specific product features within a single review sentence. Sarcasm Detection Module — Train a dedicated binary sarcasm classifier as a gating pre-processor, routing sarcastic inputs through an attention-augmented sub-network before final sentiment classification. Multimodal Integration — Incorporate image and video metadata from social media posts to provide visual context that could help resolve text-level sentiment ambiguity.

V. CONCLUSION

This research presented a complete, end-to-end framework for three-class sentiment classification of Hinglish social media text. The proposed system integrates three tightly coupled innovations: (1) a domain-specific Hinglish Normalization Lexicon with 4,200 phonetic mappings, negation handling, and emoji preservation; (2) pre-trained 300-dimensional FastText sub-word embeddings that natively handle out-of-vocabulary phonetic variants; and (3) a six-layer stacked Bidirectional LSTM architecture with SpatialDropout1D regularization.

Evaluated on a class-balanced dataset of 3,500 annotated samples, the proposed system achieves 86.4% accuracy with macro-averaged F1-score of 0.86—an 18.1% absolute improvement over the Naive Bayes baseline, a 5.3% gain over Random Forest, and an 11.6% gain over the mBERT transformer baseline. The model operates with a total inference latency of 76.4 ms per sentence, confirming its viability for real-time deployment. The qualitative failure analysis identifies sarcasm as the primary remaining challenge, accounting for 68% of misclassifications, and motivates the development of dedicated sarcasm detection modules as the primary avenue for future improvement.

This work establishes a rigorous, reproducible benchmark for the Indian NLP community and provides a concrete, scalable foundation for multilingual sentiment analysis systems serving the rapidly growing Indian digital economy.

REFERENCES

[1] A. Joshi et al., "Sentiment Analysis of Code-Mixed Hinglish Text: A Multi-modal Approach," Journal of Advanced NLP Research, vol. 14, no. 3, 2022.

[2] R. Prabhu et al., "Character-Level CNN for Code-Mixed Text Classification," in Proc. EMNLP Workshop on NLP for Low-Resource Languages, 2022.

[3] S. Lal et al., "Cross-Linguistic Challenges: Evaluation of mBERT on Indian Romanized Dialects," in Proc. IEEE Int. Conf. on Artificial Intelligence, pp. 112–119, 2023.

[4] R. Kumar, "Vanishing Gradients and Long Dependencies in Indian Social Media Text," International Journal of Computational Linguistics, vol. 9, no. 2, 2024.

[5] P. Patwa et al., "SemEval-2020 Task 9: Overview of Sentiment Analysis of Code-Mixed Tweets," in Proc. SemEval-2020, pp. 774–790, 2020.

[6] V. Sharma et al., "Sub-word Tokenization Strategies for Indian Code-Mixed Text," in Proc. ACL Workshop on Multilingual NLP, 2021.

[7] A. Bojanowski et al., "Enriching Word Vectors with Subword Information," Transactions of the ACL, vol. 5, pp. 135–146, 2017.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] Facebook AI Research, "FastText: Library for Efficient Text Classification and Representation Learning," 2024. [Online]. Available: https://fasttext.cc

[10] V. Sharma, "Emoji Sentiment Vectorization: A New Frontier for Indian NLP," Tech-India Review, vol. 3, no. 1, pp. 45–58, 2024.