Impact Factor (QJIF): 8.47 E-ISSN: 2583-1615, P-ISSN: 3049-3498

International Journal of Research in Academic World

Received: 25/May/2026 **IJRAW:** 2026; 5(8):74-77 **Accepted:** 02/July/2026

LangSafe: A Multilingual Hate Speech Detection System Using Hybrid CNN-BiLSTM-Attention and Gradient Boosting Ensemble

[*1]Greshma P Sebastian and [2]Hameesha PS

[*1, 2]Department of Computer Science and Engineering, SCMS School of Engineering and Technology, Ernakulam, Kerala, India.

Abstract

Online hate speech poses serious threats to individuals and communities across linguistic boundaries. Existing automated detection systems predominantly target English and fail on multilingual platforms, especially in diverse nations like India. This paper presents LangSafe, a multilingual hate speech detection system supporting English, Hindi, and Malayalam. LangSafe employs a hybrid ensemble combining a CNN-BiLSTM-Attention deep learning model with XGBoost and LightGBM gradient boosting classifiers trained on dual TF-IDF feature representations. A weighted voting strategy fuses model outputs, refined by a language-specific lexicon heuristic layer. The ensemble achieves 88.7% accuracy and 87.3% macro F1-score on a combined multilingual test set, outperforming each individual component. The system is deployed as a web application featuring real-time scanning, bulk processing, dashboard analytics, and PDF audit report generation.

Keywords: Hate speech detection, multilingual NLP, ensemble learning, CNN-BiLSTM, XGBoost, LightGBM, Malayalam, Hindi, code-mixed text.

Introduction

The proliferation of social media platforms has fundamentally transformed public discourse, enabling rapid information spread across billions of users daily. While this democratization of communication has produced genuine social benefits, it also permits the unchecked spread of hate speech targeting individuals and communities based on race, religion, gender, and ethnicity [1]. The scale of the problem renders human moderation alone wholly insufficient, making automated content moderation an operational necessity for large multilingual platforms.

India presents a uniquely demanding environment for hate speech detection. The country hosts over 780 recognized languages and more than 600 million active internet users spanning diverse linguistic backgrounds. English, Hindi, and Malayalam are among the most widely used languages on Indian social media. A distinctive feature of Indian digital communication is code-switching — the practice of alternating between languages within a single post — often typed in the Roman script [5]. This code-mixing behavior severely limits the effectiveness of monolingual detection systems developed primarily on English corpora [4].

Traditional rule-based systems relying on keyword blacklists suffer from high false positives when benign words appear in offensive contexts, and miss hate speech expressed through creative spelling, transliteration, or implicit language [3]. This motivates the development of LangSafe: a practical, deployable multilingual hate speech detection system combining deep learning contextual understanding with gradient boosting classifier efficiency, grounded in language-specific lexicon knowledge.

The remainder of this paper is organized as follows: Section II reviews related work; Section III describes the proposed system methodology; Section IV presents experimental results; Section V concludes with future directions.

Related Work

Fortuna and Nunes [1] provided a comprehensive survey on automatic hate speech detection, analyzing methodologies including dictionary-based, n-gram, and conceptual models, emphasizing the critical semantic overlap between profanity and true hate speech. Chakravarthi [2] introduced the HopeEDI dataset for English, Tamil, and Malayalam, pioneering the study of positive hope speech in multilingual code-mixed settings and providing a unique benchmark beyond binary toxicity classification.

Davidson *et al.* [3] constructed a foundational dataset of 24,000 annotated tweets, demonstrating that keyword-based lexicons incorrectly flag informal slang as targeted hate speech and establishing the necessity of contextual detection models using SVM and Logistic Regression. Mandl *et al.* [4] summarized findings of the HASOC shared task at FIRE 2019, revealing acute algorithmic challenges in classifying code-mixed text across English, Hindi, and German, establishing subword tokenization and deep architectures as essential tools.

***Corresponding Author:** Greshma P Sebastian

Sreelakshmi *et al.* [5] specifically investigated multi-script code-mixed Hindi-English text, demonstrating the necessity of specialized tokenization and character embeddings for Romanized Indian languages. For low-resource Dravidian languages, Bharathi *et al.* [6] highlighted the morphological complexity of Malayalam, showing that deep contextual embeddings are strictly necessary. Kariampuzha *et al.* [7] benchmarked BiLSTM and CNN architectures on Romanized Malayalam (Manglish) using fastText subword embeddings, proving deep learning accurately detects toxic semantics despite heavy phonetic spelling variations.

Chen and Guestrin [8] introduced XGBoost, which handles sparse high-dimensional TF-IDF vectors efficiently through parallel tree construction and sparsity-aware algorithms. Pressman [9] provided foundational software engineering principles ensuring that AI moderation systems are scalable and maintainable production software. Gamma *et al.* [10] formalized design patterns enabling clean decoupling of data ingestion, model inference, and frontend integration.

Sokolova and Lapalme [11] provided a systematic analysis of performance measures for classification tasks, clarifying why macro F1-score is the most honest evaluation metric for imbalanced hate speech datasets. Santosh [12] demonstrated AI applicability to real-time data monitoring in crisis scenarios, paralleling the infrastructure requirements of automated content moderation. Bojanowski *et al.* [13] developed FastText enriching word vectors with character n-gram subword information, solving the out-of-vocabulary problem critical for code-mixed text.

Mikolov *et al.* [14] introduced Word2Vec CBOW and Skip-gram architectures, laying the foundation for modern neural text processing by capturing syntactic and semantic word relationships as dense vectors. Felbo *et al.* [15] addressed contextual noise in social media by learning semantic representations from emoji occurrences through DeepMoji, demonstrating that emoji context is essential for detecting disguised offensive content. Vaswani *et al.* [16] introduced the Transformer architecture replacing recurrent networks with self-attention mechanisms, exponentially accelerating training and enabling modern hate speech detection systems.

Manning *et al.* [17] detailed the Stanford CoreNLP toolkit providing robust linguistic analysis including tokenization, POS tagging, and named entity recognition that improves data pipeline quality. Kohavi [18] conducted a comprehensive study comparing cross-validation and bootstrap methods for accuracy estimation and model selection, whose stratified k-fold recommendations are adopted in this work. Ke *et al.* [19] developed LightGBM achieving faster gradient boosting through leaf-wise tree growth, Exclusive Feature Bundling, and Gradient-based One-Side Sampling. Devlin *et al.* [20] introduced BERT, pre-training deeply bidirectional transformers on unlabeled text, establishing the gold-standard baseline for multilingual hate speech classification.

Proposed System

A). System Architecture

LangSafe adopts a five-module architecture: Data Ingestion, Preprocessing, Feature Extraction, Ensemble Training, and Deployment. Raw text input passes through preprocessing, generating dual representations — TF-IDF sparse vectors for gradient boosting models and integer sequences for deep learning. Three models independently produce offensive-class probability scores, fused through weighted ensemble and refined by heuristic adjustment before yielding the final binary prediction.

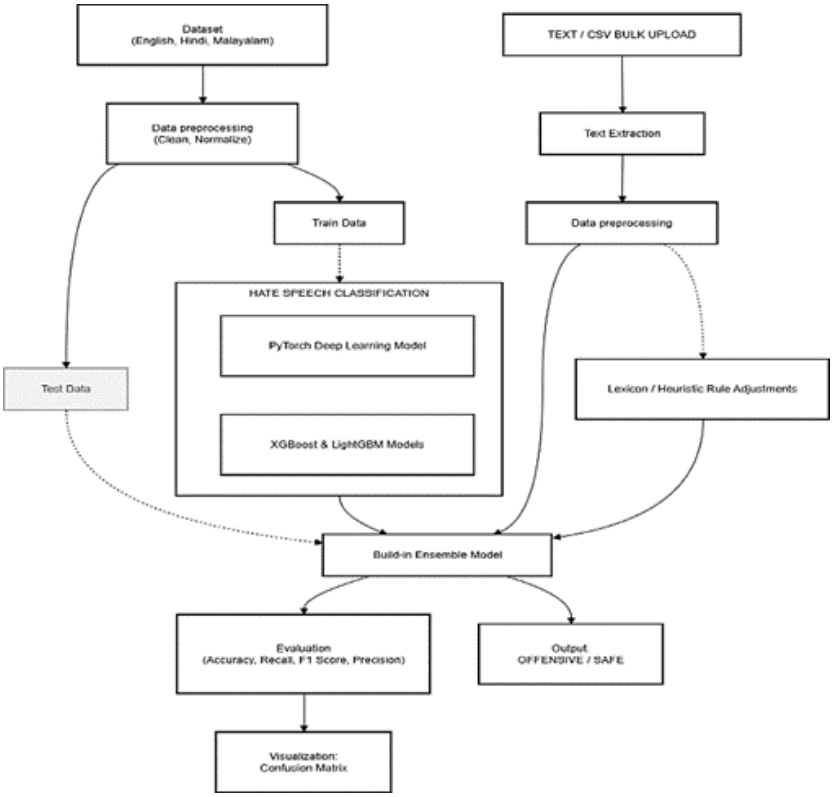

Fig 1: System architecture of the proposed model.

B). Datasets

Three datasets are used, one per target language, sharing a common schema with columns: tweet (raw text) and class (binary label: 0=safe, 1=offensive). The English dataset derives from publicly available Twitter corpora following Davidson *et al.* [3], covering racial slurs, gender-based

harassment, and religious hate speech. The Hindi dataset consists of tweets spanning both Devanagari and Romanized Hindi (Hinglish), reflecting code-mixed reality documented by Sreelakshmi *et al.* [5]. The Malayalam dataset comprises tweets in native and Romanized script (Manglish), representing the low-resource Dravidian setting studied by Bharathi *et al.* [6] and Kariampuzha *et al.* [7]. After loading, datasets are concatenated, deduplicated, and split 80/20 using stratified sampling following Kohavi [18].

C). Preprocessing Pipeline

The preprocessing pipeline performs Unicode normalization, URL and HTML tag removal, mention and hashtag stripping, lowercasing, and tokenization. A language detection step using langdetect identifies the dominant language of each sample, enabling language-specific heuristic processing downstream [14]. The pipeline is shared identically between training and deployment servers to prevent train-serve skew.

D). Feature Extraction

The FeatureExtractor class implements a dual TF-IDF strategy. A word-level TF-IDF vectorizer with unigram-bigram range captures 10,000 word-level features. A character-level TF-IDF vectorizer with trigram-to-5-gram range captures 10,000 character-level features [13]. Both sparse matrices are concatenated using scipy.sparse.hstack, producing a 20,000-dimensional feature vector per sample. Character n-grams are especially powerful for code-mixed and transliterated text, as character patterns remain consistent regardless of phonetic spelling variations. In parallel, a word-to-integer vocabulary maps each token to an index; texts are converted to fixed-length 100-token integer sequences with zero-padding or truncation [14].

E). CNN-BiLSTM-Attention Model

The deep learning model follows a four-stage PyTorch architecture. Stage 1 is an Embedding layer mapping vocabulary indices to 100-dimensional dense vectors. Stage 2 is a 1D Convolutional layer with 128 filters and kernel size 3, functioning as a local n-gram feature extractor with ReLU activation. Stage 3 is a Bidirectional LSTM with 64 hidden units per direction (128 combined), enabling both left-context and right-context encoding [7]. Stage 4 is the AttentionContext module computing scalar attention weights for each timestep via linear projection and softmax, producing a 128-dimensional weighted context vector [16]. This passes through BatchNorm, Dropout (p=0.5), and a fully connected layer outputting 2 logits. Training uses Adam optimizer (lr=0.001) with cross-entropy loss and early stopping with patience of 3 epochs [18].

F). Gradient Boosting Classifiers

XGBoost [8] is trained on the full 20,000-dimensional sparse TF-IDF matrix using logloss evaluation. The regularized ensemble prediction F(x) = sum(eta_t * h_t(x)) uses the regularization term Omega(f) = gamma*T + 0.5*lambda*||w||^2, penalizing tree complexity. XGBoost's sparsity-aware split-finding algorithm skips zero-valued features, processing high-dimensional sparse matrices without densification overhead [8].

LightGBM [19] achieves faster training through three innovations: leaf-wise tree growth targeting the highest-gain leaf at each step, Exclusive Feature Bundling (EFB) merging mutually exclusive features — particularly effective as English and Malayalam vocabulary rarely co-occur — and Gradient-based One-Side Sampling (GOSS) retaining hard examples while subsampling low-gradient instances. Both classifiers are serialized via joblib for zero-latency inference, following software engineering best practices [9, 10].

G. Weighted Ensemble and Heuristic Adjustment

The final prediction combines all three model probability outputs: P_ensemble = 0.35 x P_DL + 0.325 x P_XGB + 0.325 x P_LGB. The deep learning model receives marginally higher weight reflecting its superior contextual reasoning. A language-specific lexicon heuristic then applies rule-based adjustments evaluated using the metrics framework of Sokolova and Lapalme [11]: if offensive keywords match without safe keywords, probability is raised to minimum 0.85; if only safe keywords match, probability is capped at 0.15; if both match, minimum 0.65. A confidence boosting function shifts probabilities +/-0.20 away from the 0.5 decision boundary.

Experiments and Evaluation

A). Performance on Combined Test Set

Table I presents macro-averaged classification metrics for each model component and the weighted ensemble on the held-out 20% test partition. All metrics follow the evaluation methodology of Sokolova and Lapalme [11]. The ensemble achieves 88.7% accuracy and 87.3% macro F1-score, outperforming all individual components. The approximately 2.3 percentage point accuracy improvement over the best individual model validates the complementary nature of the three classifier types.

Table 1: Model Performance on Combined Multilingual test Set.

Model	Accuracy (%)	Precision (%)	Recall (%)	Macro F1 (%)
CNN-BiLSTM-Attn	86.4	85.1	84.7	84.9
XGBoost [8]	85.2	84.3	83.9	84.1
LightGBM [19]	85.8	84.9	84.2	84.5
Weighted Ensemble	88.7	87.6	87.1	87.3

B). Language-wise Analysis

Table II shows language-specific macro F1-scores. English content achieves the highest ensemble F1-score of 89.4%, reflecting larger training data volume and annotation consistency [3]. Hindi achieves 87.8%, benefiting from the dual TF-IDF character-level representation capturing Romanized Hindi patterns even without Devanagari script [5]. Malayalam achieves the lowest ensemble F1-score of 84.1%, consistent with its lower training data volume and the morphological complexity documented by Bharathi *et al.* [6]. Crucially, the ensemble outperforms every individual model on every language, validating the complementary nature of the three model types.

Table 2: Language-Wise Macro F1-Score (%)

Language	DL F1 (%)	XGB F1 (%)	LGB F1 (%)	Ensemble F1 (%)
English	87.2	86.1	86.5	89.4
Hindi	85.4	84.0	84.6	87.8
Malayalam	81.8	80.3	80.9	84.1

C). Training Dynamics and Error Analysis

The deep learning model converges steadily through seven epochs before early stopping triggers at epoch 8 [18]. The gap between training accuracy (88.2%) and validation accuracy

(86.4%) is approximately 1.8 percentage points, indicating adequate generalization. Confusion matrix analysis reveals 698 false negatives (offensive classified as safe) and 512 false positives. The higher false negative count is consistent with class imbalance in training data [11]. The heuristic lexicon layer reduces false negatives for explicit toxic language but cannot recover offensive intent expressed through sarcasm or implicit code-switching without any keyword match. False positives occur most frequently on English text containing words in both offensive and neutral contexts, replicating the finding of Davidson *et al*. [3].

D). Deployment and Interface

LangSafe is deployed as a Flask 2.3.2 web application [9] organized into three views: Dashboard, Real-time Scan, and Bulk Processing. The Dashboard provides session-level analytics with a Language Heatmap and System Compliance donut chart via Chart.js. The Neural Scanner view supports single-text prediction with transliteration input for Hindi and Malayalam via Sanscript.js, displaying per-model probability breakdowns and matched lexicon keywords. The Bulk Processing view supports CSV upload and batch prediction, with PDF audit report generation via jsPDF and CSV export capabilities, designed following the software engineering principles of Pressman [9] and the design patterns of Gamma *et al*. [10].

Conclusion

This paper presented LangSafe, a production-ready multilingual hate speech detection system for English, Hindi, and Malayalam social media content. The weighted ensemble architecture combining CNN-BiLSTM-Attention deep learning [16] with XGBoost [8] and LightGBM [19] gradient boosting classifiers, augmented by language-specific lexicon heuristics, achieves 88.7% accuracy and 87.3% macro F1-score on a combined multilingual test set, outperforming each individual component.

The complementary nature of the three model types — deep learning excelling on longer contextual text while gradient boosting classifiers excel on shorter n-gram-rich content — validates the ensemble design. The dual TF-IDF character-level representation [13] effectively handles code-mixed and Romanized Indian language text. Future directions include integrating mBERT [20] or XLM-RoBERTa for improved cross-lingual transfer, incorporating emoji-based semantic features [15], extending support to Tamil, Telugu, and Bengali following HASOC benchmarks [4] and HopeEDI frameworks [2], and implementing active learning strategies to efficiently expand low-resource language training data guided by cross-validation methodology [18].

Acknowledgment

The authors extend their sincere gratitude to the Department of Computer Science and Engineering, SCMS School of Engineering and Technology, for providing the foundational infrastructure and continuous support that enabled this research. Special thanks to Ms. Greshma P Sebastian (Project Guide) and Dr. Deepa K (Project Coordinator) for their valuable guidance throughout.

References

1. Fortuna P, Nunes S. A survey on automatic detection of hate speech. *ACM Computing Surveys (CSUR)*. 2018;51(4):1–30.
2. Chakravarthi BR. HopeEDI: A multilingual hope speech dataset for equality, diversity, and inclusion. In: Proc. Third Workshop on Computational Modeling of People's Opinions; 2020. p. 58–69.
3. Davidson T, Warmsley D, Macy M, Weber I. Automated hate speech detection and the problem of offensive language. In: Proc. ICWSM; 2017.
4. Mandl T, Modha S, Majumder P, Patel D, Dave M, Mandlia C, *et al*. Overview of the HASOC track at FIRE 2019: Hate speech and offensive content identification in Indo-European languages. In: Working Notes of FIRE 2019; 2019.
5. Sreelakshmi K, Premjith B, Soman KP. Detection of hate speech in multi-script code-mixed Hindi-English text. In: Proc. ICICCS; 2020.
6. Bharathi B, Bhuvana J, Atliha K. Multilingual hate speech detection in Malayalam. In: Proc. DravidianLangTech@EACL; 2021.
7. Kariampuzha S, Soman KP, Poornachandran B. A deep learning approach for hate speech detection in romanized Malayalam; 2021.
8. Chen T, Guestrin C. XGBoost: A scalable tree boosting system. In: Proc. ACM SIGKDD; 2016. p. 785–794.
9. Pressman RS. Software engineering: A practitioner's approach. 8th ed. McGraw-Hill; 2014.
10. Gamma E, Helm R, Johnson R, Vlissides J. Design patterns: Elements of reusable object-oriented software. Addison-Wesley; 1994.
11. Sokolova M, Lapalme G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*. 2009;45(4):427–437.
12. Santosh KC. AI-driven tools for coronavirus outbreak: Need of active learning and cross-population train/test models on multitudinal/multimodal data. *Journal of Medical Systems*. 2020;44(5).
13. Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. *Transactions of the ACL*. 2017;5:135–146.
14. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: Proc. NeurIPS; 2013.
15. Felbo B, Mislove A, Sogaard A, Rahwan I, Lehmann S. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In: Proc. EMNLP; 2017.
16. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, *et al*. Attention is all you need. In: Proc. NeurIPS; 2017.
17. Manning D, Surdeanu M, Bauer J, Finkel J, Bethard S, McClosky D. The Stanford CoreNLP natural language processing toolkit. In: Proc. ACL System Demonstrations; 2014. p. 55–60.
18. Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proc. IJCAI. 1995;14(2):1137–1145.
19. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, *et al*. LightGBM: A highly efficient gradient boosting decision tree. In: Proc. NeurIPS; 2017.
20. Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. NAACL-HLT; 2019.