International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(17s) 2026 pp. 775-799
https://doi.org/10.70917/ijcisim-2026-4797
Article

Linguistically Informed Machine Learning for Gujarati–English Code-Mixed Sentiment Classification: A Comparative Study of Feature Fusion Strategies

Chirag D. Shah[1], Shailesh A. Chaudhari[2]

[1]Department Of Computer Science VNSGU, Surat, India
chirags268@gmail.com
[2] Department of IT and ICT, JP Dawer Institute Of Information Science and Technology, VNSGU, Surat, India.
sachaudhari@vnsgu.ac.in

Abstract: Code-mixed text, in which words from multiple languages are used within the same sentence, is common on social media platforms and poses significant challenges for conventional natural language processing techniques. This study focuses on five-class sentiment classification of of Gujarati–English code-mixed text. The dataset consists of 5 sentiment classes ranging from extremely negative to extremely positive distributed over 44,672 sentences. A linguistically informed framework is proposed that incorporates word-level annotations, including language identity, sentiment polarity, and intensifier presence, generated using a multi-task fine-tuned DistilBERT tagger. These annotations are aggregated into sentence-level handcrafted features and combined with conventional text representations, namely Bag of Words (BoW), Term Frequency–Inverse Document Frequency (TF-IDF), Word2Vec, and FastText. The proposed framework is evaluated using a late fusion approach based on out-of-fold (OOF) stacking and is compared with early fusion, where features are directly concatenated, and with embedding-only baselines. The statistical significance of performance differences is assessed using McNemar's test. Seven machine learning classifiers—Logistic Regression (LR), Multinomial Naïve Bayes (MNB), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), and Extreme Gradient Boost(XGB)—are evaluated. The hyperparameters of all classifiers are optimized using GridSearchCV and then kept fixed throughout the experiments to ensure a fair comparison. Class imbalance is addressed using cost-sensitive learning through class-weight adjustment. The experimental results show that early fusion of linguistic and text features consistently outperforms the embedding-only and late fusion approaches for most classifier–representation combinations. The best-performing model is XGB with FastText under the early fusion framework, achieving an accuracy of 0.78 and a macro F1-score of 0.74 on the unseen test set. Overall, this work demonstrates that incorporating explicit linguistic information, including language identity, sentiment polarity, and intensifier information, improves sentiment classification of Gujarati–English code-mixed text. In addition, it provides a comprehensive and statistically validated comparison of feature integration strategies for sentiment analysis in low-resource, code-mixed language settings.
Keywords: BoW, DL, DT, KNN, LR, ML, MNB, NLP, RF, SA, SVM, TF-IDF, XGB, OOF

1. Introduction

In the current age of digital communication, people increasingly use social media platforms to express opinions, provide feedback, and engage in informal interactions while maintaining anonymity. In India linguistic diversity with hundreds of regional dialects and variations exists. This has a major impact on the online interactions. This rich multilingual landscape creates unique challenges and opportunities for Natural Language Processing (NLP) research, mainly in the analysis of informal and user-generated text.

Gujarati is spoken by over 55 million people exemplifies India's linguistic diversity. The language has different dialectal variation influenced by geography (e.g., Surti, Kathiyawadi, Charotari, Kachhi), community (e.g., Parsi, Muslim, Sindhi), and a widespread global diaspora. With growing use of Internet in rural and semi-urban areas past Covid-19, digital divide has narrowed. There is a growth in the use of English due to this digital literacy and easy availability of English and regional keypads in mobile devices. In digital expressions, Gujarati and English elements coexist within a single sentence or utterance. Many of users express themselves in either Romanized or native Gujarati scripts, often without formal training in Indic keyboard input systems.

This situation is especially common on informal digital platforms such as YouTube comments, WhatsApp chats, Facebook posts, and Twitter(X) threads. Here, linguistic norms are relaxed and expressions tend to be spontaneous and creative.

Such inter-mixing of Gujarati and English in user-generated text presents unique challenges for existing NLP systems. Unlike monolingual data, which typically follows standard grammar and syntax, code-mixed content is often noisy and unstructured. It may contain irregular grammar, creative spellings, morphological complexity, regional slang, and inconsistent syntax. These factors make standard NLP tasks, such as sentiment analysis, particularly difficult. The conventional models are not designed to handle the hybrid and non-standard nature of such inputs.

In this study, we focus on sentiment analysis of Gujarati–English code-mixed text, emphasizing user comments written in both Romanized and native Gujarati scripts. To support this study, we curated a large-scale dataset of over 77,761 sentences. Out of these scrapped data, we found 44,672 sentences appropriate for our study. Author1 of this paper labelled the sentences on 5 different categories. During the analysis of the dataset, we found that users tend to express their opinions in the spectrum of extreme disappointment to extreme joy. As a result we decided to classify each sentence into one of 5 categories: Extremely Negative, Negative, Neutral, Positive, and Extremely Positive. Our filtered dataset has a wide range of sentiment expressions that enables us to capture subtle variations in opinion intensity. This dataset reflects real-world language use, having informal structures, with variations in dialects, and spontaneous expressions typical of code-mixed online content. This provides a robust foundation for developing and evaluating sentiment classification models.

To address these challenges, we propose a machine learning framework that augments traditional text representations — BoW, TF-IDF, Word2Vec, and FastText — with word-level linguistic annotations. Each word is annotated with its language identity (Gujarati, English, or Other), a three-class sentiment polarity score, and an intensifier indicator. Using these as base features, we derive sentence-level handcrafted statistical features capturing code-switching patterns, sentiment distribution, and intensification within each sentence. First an early fusion where features are concatenated directly is performed. After that, these handcrafted features are integrated with embedding-based representations through a late fusion strategy based on OOF stacking, in which base classifiers trained separately on embeddings and handcrafted features are combined via a logistic regression meta-learner. We evaluate seven classifiers — LR, MNB, SVM, KNN, DT, RF, and XGB — with hyperparameters for each classifier–representation combination optimized via GridSearchCV. Class imbalance across the five sentiment categories is addressed through cost-sensitive class-weighting rather than synthetic oversampling.

2. Literature Review

Sentiment analysis of code-mixed text has gained significant attention in recent years due to the widespread use of multilingual and informal language on social media platforms. Early research in this area primarily focused on traditional machine learning approaches combined with feature engineering techniques, particularly for Indian code-mixed languages.

2.1 Traditional Machine Learning Approaches

Initial efforts into sentiment analysis of code-mixed social media content were mainly based on machine learning pipelines. These approaches combined hand-crafted features with probabilistic or margin-based classifiers. Joshi et al. [1] did cross-lingual comparisons, consisting of Malayalam–English, Tamil–English, Hindi–English, and

Bengali–English data. They used Weighted Word Unigram (WWU) representation and paired with a MNB classifier. It consistently surpassed conventional TF-IDF baselines, with accuracy scores between 0.75 and 0.88 across the four language pairs. Extending this line of work to character-level granularity, Mishra et al. [2] examined character n-gram features and TF-IDF vectorisations in ensemble configurations for Hindi–English and Bengali–English corpora. They proved that sub-word representations can partially compensate for the inconsistencies natural to romanized code-mixed text.Jhanwar and Das [3] explored the application of n-gram ensemble methods. They combined character-trigram-based Naïve Bayes models with a character-level LSTM within a voting ensemble for Hindi–English data, achieving an accuracy of 70.8% and a macro F1 of 0.66. Patra et al. [25], showed an overview of the SAIL Code-Mixed Shared Task at ICON-2017. They found that linear SVM classifiers with TF-IDF features and SentiWordNet augmentation achieved macro F1 scores of 0.569 and 0.526 for Hindi–English and Bengali–English respectively, which surpassed several deep learning submissions which were constrained by limited training data.Having faced the issue of class imbalance in code-mixed corpora, Srinivasan and Subalalitha [26] proposed integrating Levenshtein distance-based text normalisation with SMOTE and ADASYN oversampling within an SVM framework for Tamil–English data. They found that removing pure instances gave improvements in macro F1 relative to baselines. Within the context of the SemEval-2020 Task 9 (SentiMix) competition, Advani et al. [27] proposed a feature-engineering-centric approach in which lexical, sentiment-polarity, emoji-frequency, and metadata features were weighted by TF-IDF scores and supplied to a Logistic Regression classifier, attaining weighted F1 scores of 0.65 for Hinglish and 0.63 for Spanglish. Javdan et al. [28] demonstrated that NBSVM coupled with character n-gram TF-IDF embeddings constitutes a language-independent model achieving sixth rank among sixty-two Hinglish submissions.

2.2 Deep Learning and Hybrid Architectures

Neural architectures substantially improves the capacity of sentiment models to accommodate the syntactic irregularities and morphological complexity of code-mixed text. Prabhu et al. [4] demonstrated a morpheme-aware subword LSTM, trained on Hindi–English data. It significantly outperformed character-level LSTMs and unigram MNB baselines, with an accuracy of 69.7% and a macro F1 of 0.658. Jamatia et al. [5] conducted a systematic comparison between conventional machine learning classifiers and a range of deep learning architectures on English–Hindi and English–Bengali corpora. The results demonstrated performance improvements of 20–60% in favour of deep models, particularly in contexts with rich training data. Khan and Mamidi [6], developed a CNN–LSTM ensemble leveraging Word2Vec, GloVe, and FastText embeddings, and Kogilavani et al. [7], combined CNN–BiLSTM layers with class-resampling pre-processing for Tamil–English data, achieving an accuracy of 0.66. Pradhan et al. [26] developed ensemble deep learning approach for Hindi–English code-mixed sentiment analysis. The work of Shanmugavadivel et al. [8] explored multilingual code-mixed offensive language identification using deep learning model of CNN-RNN combination across Dravidian and other Indian language pairs.

2.3 Language-Aware Representations and Transformer Models

A key perception to emerge from recent research is that incorporating word-level language identification (LID) tags into model inputs can substantially improve classification performance on code-mixed text. Chanda et al. [8] augmented multilingual BERT (mBERT) with token-level language annotations for Tamil–English, Kannada–English, and Malayalam–English corpora, achieving F1 improvements of 4.3–10 percentage points over unaugmented mBERT baselines, with a Malayalam–English weighted F1 of 0.69. Takawane et al. [9] extended this paradigm to a wider suite of Hindi–English benchmark datasets using BERT, mBERT, and domain-adapted variants including HingBERT and HingRoBERTa, with the latter achieving a macro F1 of 0.733 on the GLUECoS benchmark. Mahata et al. [10] integrated word-level LID tags into a Bi-RNN architecture for English–Tamil sentiment classification, reporting a weighted F1 of 0.58. Hossain et al. [11] further substantiated these findings in a cross-lingual hope-speech detection task, comparing mBERT, Indic-BERT, XLNet, and CNN-BiLSTM configurations against XLM-RoBERTa, with the latter yielding weighted F1 scores of 0.93, 0.60, and 0.85 for English, Tamil–English, and Malayalam–English data respectively. Shakeel et al. [18] investigated sentiment classification of Russian–English election commentary using attention-based LSTM, mBERT, and XLM-R, with XLM-R achieving an F1 of approximately 0.71. Yadav and

Chakraborty [19] proposed a zero-shot framework for English–Spanish code-mixed sentiment using cross-lingual embeddings, obtaining F1 scores of 0.58 without parallel corpora and 0.62 with parallel data. Nayak and Joshi [17] contributed the L3Cube-HingCorpus and associated HingBERT family of models, establishing strong Hinglish-specific pretrained representations that have informed subsequent efforts on other Indian language pairs.Translation and transliteration pipelines have also been leveraged to circumvent the scarcity of code-mixed annotated data. Koyyalagunta and Supriya [22] combined transliteration, machine translation, and transformer fine-tuning across Hindi–English, Malayalam–English, Tamil–English, and Telugu–English pairs, reporting improvements of up to 6% in accuracy and 9% in F1 over non-translation baselines. Shah and Bakrola [25] further demonstrated the utility of attention-based neural machine translation for Indic language processing, providing a base for cross-lingual and code-mixed sentiment pipelines.

2.4 Shared Tasks, Benchmarks, and Evaluation Frameworks

The SemEval-2020 Task 9 (SentiMix) [28] attracted eighty-nine submissions for Hinglish and Spanglish tracks; the best-performing systems achieved F1 scores of 75.0% and 80.6% respectively, with dominance of transformer-based and ensemble methods. The Dravidian-CodeMix benchmark introduced by Chakravarthi et al. [13] enabled broad comparative evaluation of sentiment analysis and offensive language identification across Tamil–English, Malayalam–English, and Kannada–English, with top systems achieving weighted F1 scores of 0.711, 0.804, and 0.630. Asrarul et al. [12], participated in the DravidianLangTech track at EACL 2024, benchmarked classical, deep, and transformer models on Tamil–English and Tulu–English data, reporting macro F1 values of 0.124 and 0.471 respectively. Choudhary et al. [24] proposed the Siamese Attention-based Code-Mixed Text (SACMT) framework, they applied contrastive learning within a BiLSTM architecture to leverage sentiment knowledge from resource-rich language pairs for low-resource code-mixed sentiment classification. It gained improvements of 7.6% in accuracy and 10.1% in F1 over baselines for Hindi–English and Telugu–English data.

2.5 Sentiment Analysis in Gujarati and Gujarati–English

Research specifically addressing Gujarati sentiment analysis has grown incrementally, though the field remains sparse compared to Hindi-centric studies. Early lexicon-based efforts, including the construction of the Gujarati SentiWordNet (G-SWN) by Gohil and Patel [23], established foundational resources for polarity scoring in Gujarati text. Joshi and Vekariya [16] subsequently applied Naïve Bayes classifiers with both count-based and TF-IDF feature representations to Gujarati film review data, while Patel et al. [15] extended similar TF-IDF and n-gram feature sets with an SVM classifier to Gujarati newspaper headline classification, reporting accuracies close to 92%. Addressing the dual challenge of feature representation and model selection, Shah et al. [30] proposed parallel lexicon-based and machine-learning-based sentiment classification frameworks for Gujarati film reviews. Their GMLSA pipeline used LR, RF, KNN, SVM, and Naïve Bayes classifiers with TF-IDF and count-vectorisation features validated over five distinct datasets, consistently outperformed the GujSentiWordNet-based GLSA approach, with SVM emerging as the strongest individual classifier. Gokani and Mamidi [14] further formalised Gujarati sentiment analysis benchmarking by releasing the Gujarati Sentiment Analysis Corpus (GSAC), a manually annotated collection of Twitter posts, and reporting baseline results using TF-IDF-augmented SVM and Logistic Regression, with weighted F1 scores of 0.77 and 0.78 respectively, rising to 0.80 following SMOTE-based oversampling. The preprocessing challenges specific to code-mixed Gujarati content were directly addressed by Patel and Parikh [29], who developed a linguistic resource pipeline encompassing language identification, text normalisation, and transliteration-to-native-script translation for English–Gujarati code-mixed social media data. Dudhia and Rana further contributed a dictionary-driven approach to Gujarati sentiment scoring, demonstrating the practical viability of lexicon-based methods as interpretable baselines. Despite these contributions, comparative evaluation of machine learning methods specifically on Gujarati–English code-mixed text—rather than monolingual Gujarati—remains largely absent from the literature, and no publicly available annotated dataset exists for this particular code-mixed pair. This constitutes a clear gap that the present study aims to address.

2.6 Feature Engineering, Auxiliary Techniques, and Summary

Beyond architecture selection, several studies have systematically examined how feature extraction strategies affect sentiment classification outcomes. Semary et al. [20] conducted a structured comparison of BoW, TF-IDF, Word2Vec, and GloVe representations paired with SVM classifiers, finding that TF-IDF consistently provided a favourable trade-off between representation quality and computational cost. Ayyub et al. [21] extended this evaluation to a multi-benchmark setting, combining engineered linguistic features. They included part-of-speech tags and sentiment-polarity indicators with distributed word embeddings from Word2Vec and GloVe in a deep learning framework, demonstrating systematic gains over purely feature-based or purely embedding-based classifiers.

2.7 Research Gap

Even though sentiment analysis of code-mixed text has been an active research area in recent years, most previous research focuses on high-resource language pairs such as Hindi–English [1, 3, 4, 9] or Spanish–English [19, 27], having large annotated corpora. They largely solve two- or three-class sentiment classification problems, and so fail to capture the full intensity spectrum of expressions. Research on Gujarati–English code-mixed sentiment analysis remains limited, particularly for five-class sentiment classification supported by large-scale annotated datasets. Existing studies focusing on Gujarati text are either monolingual [14, 15, 16, 23, 30] or address language identification rather than sentiment classification [29]. Language-aware modelling has demonstrated efficacy for Hindi–English and Dravidian language pairs [8, 9, 10]. But, the effect of linguistic awareness beyond language identity that includes word-level polarity and intensification cues in low-resource Gujarati–English text remains largely unexplored. Furthermore, existing language-augmented approaches mainly integrate linguistic signals through simple feature concatenation but stacked late fusion has not been examined for this language pair. To address these limitations, our study presents a comprehensive framework for five-class sentiment classification of Gujarati–English code-mixed text combining word-level linguistic annotations with multiple feature representation techniques through both early and late fusion strategies.

2.8 Research Objectives and Hypothesis

Based on the above identified research gaps, the main goal of our study is to develop and evaluate effective ML-based approaches for sentiment analysis of Gujarati–English code-mixed text. Our study explores how different feature representation techniques, linguistic feature augmentation, and fusion strategies impact sentiment classification performance in multilingual, low-resource settings. The specific research objectives of this work are as follows:

1. To curate and label a dataset of Gujarati–English code-mixed social media comments for five-class sentiment analysis.

2. To evaluate the effectiveness of different feature representation techniques, including BoW, TF-IDF, Word2Vec, and FastText, for sentiment classification in code-mixed text.

3. To assess the impact of word-level linguistic annotation — language identity, sentiment polarity, and intensifier presence — on sentiment classification performance, and to compare late fusion against early fusion as integration strategies.

4. To conduct a comparative study of machine learning classifiers using optimized hyperparameters for five-class sentiment prediction.

We hypothesize that combining word-level linguistic annotations with traditional feature-based representations will improve sentiment classification performance for Gujarati–English code-mixed text compared with using conventional feature extraction techniques alone. We further hypothesize that an early fusion approach that directly concatenates features will achieve better performance than a late fusion approach based on stacking.

2.9 Contributions of This Work

1. Dataset Curation: We created a high-quality dataset of 44,672 Gujarati–English code-mixed YouTube comments. Word-level language tags were annotated via a custom Telegram bot, while sentence-level sentiment labels were manually annotated by Author 1 of this paper. The data was collected from diverse genres of real-world YouTube channels.

2. Comparative Study: We conducted a comprehensive comparison of four vectorization techniques with seven ML, in combination with word-level linguistic features.

3. Linguistic Feature Integration via Late Fusion: We propose a late fusion framework based on OOF stacking to integrate handcrafted linguistic features language identity, sentiment polarity, intensifier presence with embedding-based representations. We further conduct an ablation study, using the embedding-only condition as the ablated (no linguistic features) baseline, to isolate the contribution of the handcrafted linguistic features under both fusion strategies.

4. Hyperparameter Optimization: We applied GridSearchCV to optimize each machine learning classifier for every feature and fusion combination, with tuned configurations frozen for reproducibility across experiments.

The proposed framework is designed to be adaptable to other low-resource code-mixed language pairs, offering a practical and reproducible approach for integrating fine-grained linguistic features into machine learning-based sentiment analysis systems.

1. Materials & Methods

This section presents the framework of our sentiment analysis system, specifically designed for Gujarati–English code-mixed text. We propose a hybrid, multi-step pipeline that integrates word-level language information, feature extraction, and machine learning-based classification. This approach is tailored to address the challenges of informal, unstructured text, such as YouTube comments written in Romanized and native Gujarati scripts. The overall sentiment analysis pipeline consists of several key stages, as illustrated in above Figure 1. Each step in our pipeline is designed to capture linguistic, semantic, and contextual patterns in the dataset. This ensures accurate sentiment classification in complex code-mixed data.

3.1. Dataset

3.1.1 Data Sources

We scraped user-generated comments from YouTube, focusing specifically on Gujarati–English code-mixed text. We fetched raw data with use of YouTube Data API v3 from channels spanning five distinct content categories: cooking and recipe tutorials; film and entertainment reviews; regional news broadcasts and political commentary; spiritual and motivational content; and online education and competitive examination guidance. We selected these diverse genres to bring variation in vocabulary and sentiment-bearing expression. Our intent here is to develop a classification model that is generalised and not domain-specific. From the initial corpus of 77,761 comments, we performed an analysis and filtered out the comments containing only URLs, purely numeric content and monolingual text. As a result, a final set of 44,672 comments was retained for annotation.

3.1.2 Sentiment Labelling Criteria

Our dataset focuses on intra-sentential code-mixing, where language switches occur within a single sentence. User comments often express a wide emotional spectrum, from extreme disappointment to ultimate joy. To capture this nuance, we labelled sentiments across five fine-grained classes:

- Extremely Negative (-2)
- Negative (-1)
- Neutral (0)
- Positive (1)

- Extremely Positive (2)

This detailed categorization enables the model to understand mixed emotions frequently found in casual, multilingual user content. Author1 of the paper performed annotation manually. We performed this on own because of challenges posed by Romanized Gujarati-English text, including:

- Irregular grammar and spelling (e.g., “goooooood”, “mastttttt”, “bakwaaas”)
- Non-standard transliterations (e.g., “saras”, “sarash”, “sarrass” all meaning “good”)
- Mixed syntactic structures (English verbs with Gujarati subjects, and vice versa)
- Local expressions, dialectal modifiers, and emphasis indicators

Due to these complexities, we did not automate labelling using keyword matching or polarity lexicons. Instead, sentence-level sentiment labels were manually annotated by Author 1 of this paper, while word-level language tags were separately annotated using a custom Telegram bot, to ensure high consistency and uniformity across the dataset.

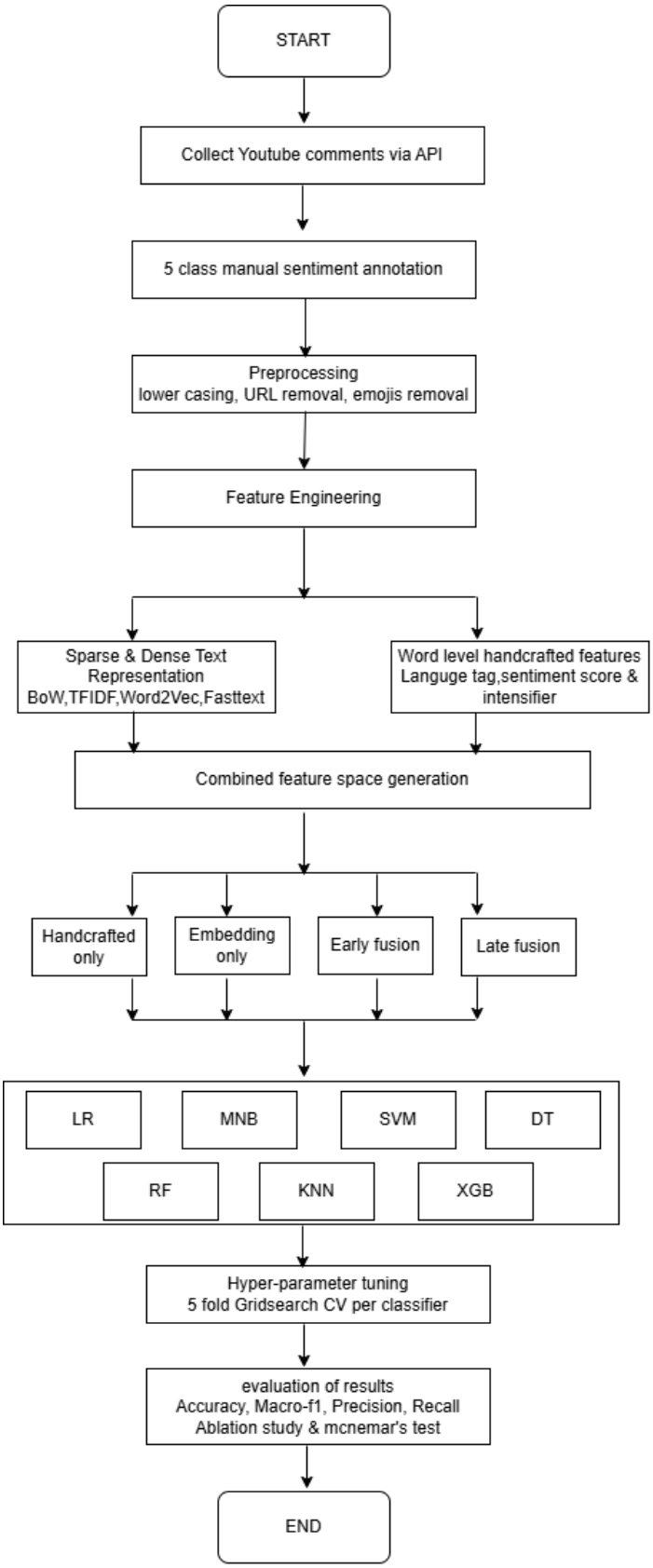

Figure 1: ML based sentiment analysis

Labelling Examples

We developed rules and guidelines based on common linguistic patterns and intensifiers to assist ourselves in assigning labels accurately. Note that certain intensifiers (e.g., 'ekdum', 'bov') appear across multiple sentiment classes; the assigned label depends on the accompanying sentiment-bearing word rather than the intensifier alone.

 Extremely negative: It represents strong disappointment, anger, or extreme criticism. It is represented by intensifiers like “bahu”, “ekdum”, “bov”, “ghanu”, “too much”, “horrible”, “terrible”, “nonsense”, “disaster”, and repeated characters.
Examples: "ekdum bakwas movie che" , "totally worstttt experience", "bekar no 1", "full time waste!"

 Negative: It is used to represent general dissatisfaction or mild negative opinion without extreme emphasis.

Examples: "bakwas che", "not so good", "thodu boring lagyu", "didn’t like it much"

 Neutral: It represents facts or descriptive comments lacking strong sentiment. It includes queries or balanced views.
Examples: "movie release date kai che?", "bhai tame kya thi cho?","sound quality is okay", "this is the recipe of dhokla"

 Positive: It shows positive view, appreciation, or light praise without strong intensifiers. Examples:"saras che bhai", "nice acting","good try","mast lage che"

 Extremely positive: Strong appreciation or excitement are represented with intensifiers. Some examples are “ekdum”, “khub”, “bov”, “ghanu”, “super”, repeated characters, or emoticons. Examples:"ekdum saras movie che", "goooooood job","mastttttttt performance bhai", "superbbb and inspiring video!"

Figure 2: illustrates the top 50 sentiment-bearing words for each class, highlighting the distinct lexical cues that guide annotation and model training.

3.1.3 Distribution of Sentiment Classes

After manual annotation, the dataset exhibited a moderate class imbalance, which is common in real-world sentiment datasets. Figure 3 shows the distribution of sentiment labels:

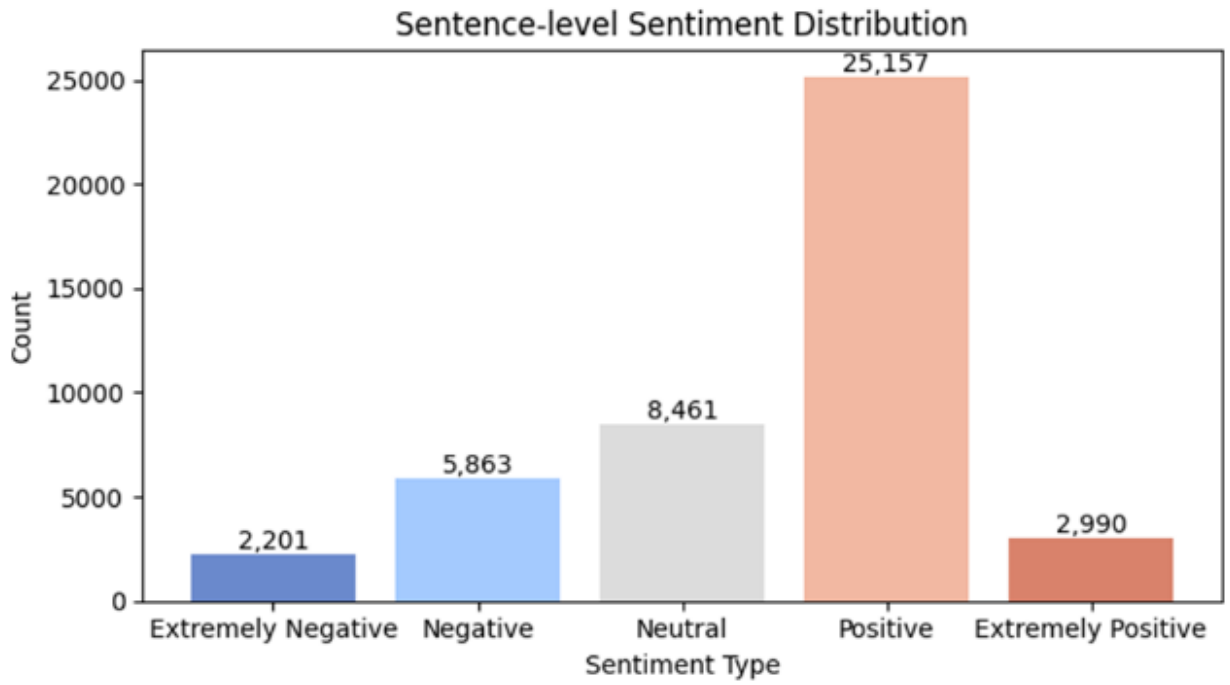

Figure 3: Fig. 3. Sentiment class distribution

The dataset shows a skew toward the Positive class, which could bias models to favour majority classes during training. To address this, during the split into training and testing subsets, we employed stratified sampling to preserve the original class proportions. This ensures consistent evaluation across all sentiment categories. The imbalance was further addressed at the modelling stage through cost-sensitive class-weighting.

3.2. Pre-processing and Tokenization

Our dataset consists of Youtube comments, which often contain orthographic variations, irregular spellings, mixed languages, and expressive cues such as emoticons or excessive punctuation. To prepare the data for modeling, we applied several pre-processing steps:

- Romanization of native Gujarati script
- Lowercasing
- Normalization of whitespace and punctuation
- Removal of emojis and emoticons
- Filtering of special symbols
- Tokenization into individual words

These cleaning steps ensure that the input to both the feature extraction components remains consistent and clean, while preserving sentiment-related cues. This facilitates more accurate and interpretable analysis of code-mixed text.

3.3 DistilBERT tagger

Accurately identifying the language of each word is crucial for sentiment analysis of code-mixed text, and this language information, together with fine-grained word-level sentiment cues, underlies our sentiment classification framework. In our previous work, we developed a manually annotated corpus comprising 77,761 code-mixed social

media sentences totalling 732,917 words, in which each token was assigned one of three language labels: GJ, EN, or OT. We evaluated annotation quality using Cohen's Kappa, obtaining a score of 0.95, indicating near-perfect agreement between annotators. We used this annotated corpus to develop a fine-tuned DistilBERT model for word-level language identification, achieving 0.98 accuracy and a 0.91 macro F1-score. In the present work, we extend this into a multi-task DistilBERT tagger that jointly predicts, for each word: (i) a three-class sentiment polarity score, and (ii) a binary intensifier indicator. The multi-task tagger achieves an accuracy of 0.98 and macro-f1 score 0.94.

To avoid label leakage, gold-standard word-level annotations are used as inputs during model training and validation. During test time, word-level tags are generated by the fine-tuned tagger to make sure that system would operate on genuinely unseen sentences. For every sentence, the tagger produces per-word predictions, which are subsequently aggregated into sentence-level linguistic features.

3.4 Feature Engineering and Text Representation

We designed a broad set of feature representations to integrate sentence-level linguistic information — language identity, sentiment polarity, and intensifier presence — with conventional text representations. We had two objectives:

1. To compare traditional sparse text representations with dense, embedding-based representations for sentiment classification.

2. To evaluate the impact of incorporating handcrafted linguistic features, derived from our multi-task DistilBERT tagger, through fusion strategy.

3.4.1 BoW: As a foundational approach, we employed the BoW model to represent each sentence as a sparse vector indicating word occurrence. While BoW does not capture sequential or syntactic structure, it serves as a robust baseline. BoW features were generated using the CountVectorizer implementation from scikit-learn.

3.4.2 TF-IDF: Building upon BoW, we implemented TF-IDF to weight each word according to its importance within the corpus, down-weighting frequent, less informative words while emphasizing rarer, sentiment-bearing tokens. TF-IDF features were generated using the TfidfVectorizer implementation from scikit-learn.

3.4.3 Word2Vec: To capture semantic and syntactic relationships beyond word occurrence, we trained 300-dimensional Word2Vec embeddings using the Skip-gram variant with a context window of 5, over 10 epochs, trained exclusively on the training split to prevent information leakage from the test set.

3.4.4 FastText: FastText extends Word2Vec by representing words as collections of character n-grams, allowing it to generate reliable embeddings for rare, misspelled, or phonetically transliterated words — common in informal code-mixed text. We trained 300-dimensional FastText embeddings using the Skip-gram variant, with a minimum token count of 1, also trained exclusively on the training split.

3.4.5 Handcrafted Linguistic Feature Aggregation: Word-level annotations produced by the multi-task DistilBERT tagger — language identity, sentiment polarity, and intensifier presence — are aggregated into sentence-level statistical features. The extended features includes per-language word proportions, a code-switch rate (the frequency of language changes between consecutive words), summary statistics of sentiment polarity (mean, standard deviation, minimum, maximum, sum, and positive/negative/neutral counts), and intensifier statistics (count, ratio, and presence). These aggregated features form a fixed-length handcrafted feature vector per sentence, independent of sentence length.

3.4.6 Fusion Strategy: Rather than directly concatenating handcrafted linguistic features with text representations, we employ a late fusion approach based OOF stacking. Separate base classifiers are trained using (i) each embedding-based representation and (ii) the handcrafted linguistic feature set. Their out-of-fold class probability predictions on the training data, together with the predicted class probabilities on the test data, are then combined and used as input to a logistic regression meta-learner, which generates the final sentiment prediction.

For comparison, we also evaluate an early fusion approach, where the embedding and handcrafted feature vectors are concatenated before being fed into a single classifier. The performance differences between the late fusion and early fusion approaches, as well as between late fusion and the embedding-only baseline, are statistically evaluated using McNemar's test.

.For each embedding type and classifier combination, we therefore evaluate three configurations:

- Embedding-only (baseline)
- Late fusion (OOF stacking) of embedding and handcrafted features
- Early fusion (concatenation) of embedding and handcrafted features, for comparison

MNB requires non negative input. So, embedding only evaluation of Word2Vec and FastText are not considered for MNB.

3.5 Classifier Models

For sentiment classification of Romanized Gujarati–English code-mixed text, we experimented with seven supervised machine learning classifiers. We selected these models for their diverse algorithmic foundations and proven success in NLP, particularly text classification tasks. We performed exhaustive hyper-parameter optimization using grid search on each of classifiers. We trained and evaluated different classifiers on multiple feature representations using these hyper-tuned values.

3.5.1 LR is a linear classifier that estimates class probabilities by applying the logistic (sigmoid) function to a weighted combination of input features. For multi-class sentiment classification, we employ a one-vs-rest strategy. Although conceptually simple, LR is known to perform well on high-dimensional text data, particularly when combined with L2 regularization to reduce the risk of overfitting. In this study, LR is applied to all feature representations, including sparse text features, dense embeddings, and handcrafted linguistic features. Its interpretability, computational efficiency, and strong performance make it a reliable baseline for sentiment classification.

3.5.2 MNB is a probabilistic classifier that assumes the features are conditionally independent given the class label. It models the distribution of discrete, non-negative feature values within each class, making it particularly well suited to text represented using frequency-based features such as BoW and TF-IDF. Because MNB requires non-negative input features, it is not evaluated with embedding-only representations based on Word2Vec and FastText, as these dense embeddings contain both positive and negative values, violating the model's assumptions. For the handcrafted linguistic feature set, which includes a combination of proportions, counts, and signed statistical features, Min–Max scaling is applied before training to transform all feature values into a non-negative range compatible with MNB.

3.5.3 SVM is a discriminative classifier that identifies the optimal decision boundary by finding the hyperplane that maximizes the margin between classes in the feature space. During hyperparameter tuning, we evaluated both linear and radial basis function (RBF) kernels, selecting the most suitable kernel for each feature representation. Linear kernels were found to be more effective for sparse text representations such as BoW and TF-IDF, whereas RBF kernels performed better with dense embeddings and the handcrafted linguistic feature set. For multi-class classification, we used scikit-learn's default one-vs-one strategy. Since SVMs do not inherently produce probability estimates, probability calibration was enabled to generate the class probabilities required for the late fusion stacking framework.

3.5.4 KNN is a non-parametric, instance-based classification algorithm that predicts the class of a sample based on the majority class among its nearest neighbours in the feature space. To measure similarity between samples, we evaluated both cosine similarity and Euclidean distance, which are widely used in text classification tasks. The most suitable distance metric and the number of neighbours were selected for each feature representation through

hyperparameter tuning. KNN was applied directly to all feature representations without dimensionality reduction, allowing us to evaluate its performance on the original structure of each representation.

3.5.5 DT classifies recursively partition the feature space by selecting features that maximize information gain or minimize impurity, resulting in a tree-like structure of decision rules. This provides interpretability through transparent, rule-based explanations of predictions. DT can also model complex, non-linear interactions between features and labels, beneficial for language data. We trained DT classifiers on all feature sets including sparse, language-tagged, and dense embeddings to assess their adaptability to varied input formats.

3.5.6 RF is an ensemble learning technique that builds multiple decision trees using random subsets of data and features, aggregating their predictions via majority voting. This randomness reduces overfitting and enhances generalization compared to single trees. The ensemble nature makes RF highly effective for handling noisy, high-dimensional, and diverse data. We trained RF classifiers on all feature configurations to evaluate scalability and robustness in multi-class sentiment classification of code-mixed text.

3.5.7 XGB is a gradient boosting framework designed for high computational efficiency and strong predictive performance. It sequentially builds an ensemble of weak decision trees in which each new model focusing on correcting errors made by previous models. With built-in regularization and efficient handling of missing values, XGB minimizes overfitting and captures complex feature interactions. We applied XGB to all feature representations to assess its ability to accurately classify sentiments in our challenging dataset.

3.5.8 Training, Hyperparameter Optimization and Performance Evaluation

To ensure a fair and consistent comparison across all classifiers, we followed a common experimental framework throughout the study. The dataset was divided into training and test sets using an 80:20 stratified split to preserve the original class distribution. The resulting train–test indices were then frozen and stored, ensuring that every classifier and feature representation was evaluated on exactly the same data split.

Instead of reserving a separate validation set, we performed 5-fold stratified cross-validation on the training data. The resulting out-of-fold predictions served two purposes: they provided an internal validation mechanism during model development and generated the base-model probability estimates required for the late fusion stacking approach described in Section 3.4.6.

Hyperparameter tuning for each classifier–feature representation pair was carried out separately using GridSearchCV before the main experiments. Once the optimal hyperparameter settings were identified, they were fixed and reused across all subsequent experiments. This ensured reproducibility while allowing the evaluation to focus on the impact of the feature representations and fusion strategies rather than differences in hyperparameter selection. Table 1 presents the optimal hyperparameter settings for each classifier across the different feature representations.

Table 1: Hyper parameter tuning using GridsearchCV

Algorithm	**Ranges Tuned**	**BoW**	**TF-IDF**	**FastText**	**Word2Vec**	**Handcrafted**
LR	C: [0.01, 0.1, 1, 10, 100], penalty: ['l1','l2'], solver: ['lbfgs','saga'], class_weight: [None,'balanced']	C=1, penalty=l2, solver=saga, class_weight= balanced	C=10, penalty=l2, solver=saga, class_weight= balanced	C=10, penalty=l2, solver=saga, class_weight= balanced	C=1, penalty=l2, solver=saga, class_weight= balanced	C=0.01, penalty=l2, solver=saga, class_weight= balanced

MNB	alpha: [0.01, 0.1, 0.5, 1.0, 2.0]	alpha=0.1	alpha=0.01	N/A	N/A	alpha=0.01
SVM	C: [0.1, 1, 10], kernel: ['linear','rbf'], class_weight: [None,'balanced']	C=0.1, kernel=linear, class_weight= balanced	C=0.1, kernel=linear, class_weight= balanced	C=0.1, kernel=rbf, class_weight= balanced	C=0.1, kernel=rbf, class_weight= balanced	C=0.1, kernel=rbf, class_weight= balanced
KNN	n_neighbors: [5,7,9,11], weights: ['uniform','distance'], metric: ['cosine','euclidean']	n_neighbors=5 , weights=distance, metric=cosine	n_neighbors=5 , weights=uniform, metric=cosine	n_neighbors=11, weights=distance, metric=euclidean	n_neighbors=11, weights=distance, metric=euclidean	n_neighbors=11, weights=uniform, metric=cosine
DT	max_depth: [10,20,None] , min_samples_split: [2,5,10], class_weight: [None,'balanced']	max_depth=None, min_samples_split=2, class_weight= balanced	max_depth=None, min_samples_split=2, class_weight= balanced	max_depth=20 , min_samples_split=2, class_weight= balanced	max_depth=20 , min_samples_split=2, class_weight= balanced	max_depth=10 , min_samples_split=2, class_weight= balanced
RF	n_estimators: [100,200,300,400], max_depth: [10,20,None] , min_samples_split: [2,5], class_weight: [None,'balanced']	n_estimators= 300, max_depth=None, min_samples_split=5, class_weight= balanced	n_estimators= 300, max_depth=None, min_samples_split=5, class_weight= balanced	n_estimators= 300, max_depth=20 , min_samples_split=5, class_weight= balanced	n_estimators= 300, max_depth=20 , min_samples_split=5, class_weight= balanced	n_estimators= 100, max_depth=10 , min_samples_split=5, class_weight= balanced
XGB	max_depth: [3,6,10], learning_rate : [0.01,0.05,0.1,0.3], n_estimators: [100,200,300,500]	max_depth=10 , lr=0.1, n_estimators= 300	max_depth=6, lr=0.1, n_estimators= 300	max_depth=6, lr=0.1, n_estimators= 300	max_depth=6, lr=0.1, n_estimators= 500	max_depth=6, lr=0.05, n_estimators= 500

Considering the multi-class nature of the sentiment classification task and the inherent class imbalance, we selected the macro-averaged F1 score as the primary evaluation metric. This metric calculates the F1 score independently for each class and averages them, thereby assigning equal importance to all classes regardless of their prevalence. Using this balanced metric allowed us to assess model effectiveness comprehensively across both majority and minority sentiment categories.

Additionally, we applied accuracy as a secondary metric to provide an overall measure of correct predictions. Together, these metrics enable comprehensive evaluation of classifier effectiveness in handling complex, code-mixed text.

 Accuracy: Measures the proportion of correctly classified instances out of the total predictions. While widely used, accuracy can be misleading in imbalanced datasets because it may favour majority classes.

 Precision, Recall, and F1-Score (Macro-Averaged): These metrics are calculated independently for each sentiment class and then averaged to provide a balanced evaluation across classes.

o Precision: Assesses the correctness of positive predictions by comparing the number of correctly predicted instances to the total predicted instances for a class.

o Recall: Evaluates coverage by measuring the proportion of correctly identified instances relative to the total number of true instances in a class.

o F1-Score: Computes the harmonic mean of precision and recall, balancing the trade-off between false positives and false negatives.

 Confusion Matrix: This visual tool provides insights into misclassification patterns by showing how often instances of each class are predicted as other classes. It highlights the sentiment categories most frequently confused and helps identify areas for model improvement.

We report macro-averaged F1 scores to ensure all sentiment classes—regardless of their frequency—contribute equally to the final evaluation. This is particularly important for capturing model performance on minority classes.

Key Findings & Discussion

This section presents and discusses the experimental results obtained by evaluating seven machine learning classifiers using four different feature representations under three experimental settings: (i) embedding-only (baseline), (ii) late fusion through out-of-fold (OOF) stacking with handcrafted linguistic features, and (iii) early fusion through direct feature concatenation. Table 2 summarizes the classification accuracy and macro F1-score for all representation–condition combinations. These experiments provide a comprehensive comparison between sparse and dense text representations, evaluate the contribution of handcrafted linguistic features, and examine how different feature integration strategies influence sentiment classification performance on Gujarati-English code-mixed text.

The results indicate that sparse representations (BoW and TF-IDF) and the dense FastText embedding are similarly competitive across the evaluated classifiers. BoW produces the highest macro F1-score for three of the seven classifiers (Logistic Regression, Multinomial Naïve Bayes, and Support Vector Machine), while FastText achieves the best performance for three others (Random Forest, K-Nearest Neighbors, and XGBoost). TF-IDF performs best only with Decision Tree, whereas Word2Vec does not produce the highest macro F1-score for any of the evaluated classifiers.

The effectiveness of incorporating handcrafted linguistic features through late fusion varies across classifiers. As shown in the ablation study presented in Table 3, late fusion improves the macro F1-score over the embedding-only baseline in 19 of the 26 valid classifier–representation combinations. This suggests that the proposed linguistic features generally provide complementary information beyond the original text representations, although the magnitude of improvement depends on the learning algorithm.

A comparison of the two fusion strategies further shows that early fusion is more effective overall. Early fusion achieves higher macro F1-scores than late fusion in 17 of the 26 valid combinations, indicating that directly combining embedding features with handcrafted linguistic features before model training enables better exploitation of the complementary information. The largest performance differences are observed for XGBoost, where late fusion performs substantially worse than both the embedding-only baseline and the corresponding early fusion models for the BoW, TF-IDF, and FastText representations.

Among all evaluated configurations, the highest overall performance is obtained by XGBoost with FastText under the early fusion setting. This model achieves a macro F1-score of 0.7361 and an accuracy of 0.7793, demonstrating that combining contextual dense word representations with handcrafted linguistic features through early feature integration provides the most effective representation for sentiment classification in Gujarati-English code-mixed text.

Table 2: Performance Across Representations & Fusion Conditions

Model		**Embedding-only**	**Late Fusion**	**Early Fusion**
Model	Feature	Acc / F1	Acc / F1	Acc / F1
LR	BoW	0.7576/0.6984	0.7236/0.6872	0.7742/0.7219
LR	TF-IDF	0.7214/0.6721	0.7055/0.6729	0.7697/0.7152
LR	Word2Vec	0.6256/0.6067	0.6949/0.6654	0.7012/0.6728
LR	FastText		0.6877/0.6594	0.6920/0.6534
MNB	BoW	0.7621/0.6549	0.6838/0.6586	0.7742/0.6717
MNB	TF-IDF	0.7525/0.6128	0.6620/0.6492	0.7762/0.6420
MNB	Word2Vec	---	---	---
MNB	FastText	---	---	---
SVM	BoW	0.7990/0.6853	0.7399/0.6979	0.7511/0.7084
SVM	TF-IDF	0.7726/0.6367	0.7108/0.6780	0.6808/0.6545
SVM	Word2Vec	0.7757/0.6413	0.7123/0.6786	0.7007/0.6709
SVM	FastText	0.7884/0.6770	0.7332/0.6988	0.7204/0.6909
DT	BoW	0.6811/0.5794	0.6468/0.6070	0.7155/0.6375
DT	TF-IDF	0.6495/0.5488	0.6428/0.6034	0.7160/0.6323
DT	Word2Vec	0.6116/0.5329	0.6345/0.5941	0.6844/0.6091
DT	FastText	0.5758/0.5154	0.6349/0.5987	0.6958/0.6171
RF	BoW	0.7643/0.6100	0.7122/0.6861	0.7961/0.6705
RF	TF-IDF	0.7641/0.6100	0.7081/0.6820	0.7959/0.6695
RF	Word2Vec	0.7636/0.6519	0.7054/0.6782	0.7965/0.6993
RF	FastText	0.7618/0.6274	0.7126/0.6844	0.8032/0.7068
KNN	BoW	0.7366/0.5813	0.7088/0.6661	0.7473/0.5835
KNN	TF-IDF	0.7278/0.5465	0.6858/0.6551	0.7773/0.6420
KNN	Word2Vec	0.7495/0.6302	0.6976/0.6621	0.7833/0.6636
KNN	FastText	0.7506/0.6417	0.7130/0.6753	0.7918/0.6788
XGB	BoW	0.7764/0.6569	0.7404/0.6920	0.8113/0.7151
XGB	TF-IDF	0.7551/0.6241	0.7346/0.6885	0.8046/0.7050
XGB	Word2Vec	0.7837/0.6548	0.7370/0.6925	0.8124/0.7040
XGB	FastText	0.7412/0.6950	0.4979/0.5543	0.7793/0.7360

LR: Among the four feature representations, BoW is the most suitable for LR. The best performance is achieved with early fusion, which attains an accuracy of 0.7757 and a macro F1-score of 0.7112, representing an improvement of 0.0128 over the embedding-only baseline. This configuration also performs noticeably better than the corresponding late-fusion model, which achieves a macro F1-score of 0.6859. These results suggest that LR effectively benefits from the direct integration of handcrafted linguistic features with sparse lexical representations.

MNB: As expected, MNB performs best with sparse, non-negative feature representations. Its highest performance is obtained using BoW with early fusion, achieving an accuracy of 0.7690 and a macro F1-score of 0.6642, corresponding to an improvement of 0.0093 over the embedding-only model. TF-IDF exhibits a similar pattern, producing an accuracy of 0.7560 and a macro F1-score of 0.6206. Word2Vec and FastText are not evaluated with MNB because the algorithm requires non-negative feature values, whereas these dense embeddings contain both positive and negative values. Applying Min-Max scaling only to the handcrafted feature block does not satisfy this requirement for the complete feature representation.

SVM: The SVM achieves its best performance with BoW under the early fusion setting, reaching an accuracy of 0.7504 and a macro F1-score of 0.7013, which is 0.0160 higher than the embedding-only baseline. FastText with late fusion produces the second-best macro F1-score of 0.6989. An interesting observation is that the embedding-only BoW model records the highest accuracy among all embedding-only configurations (0.7990), despite a comparatively lower macro F1-score of 0.6853. This difference highlights the effect of class imbalance, where high overall accuracy does not necessarily translate into balanced performance across all sentiment classes. Consequently, macro F1-score is considered the primary evaluation metric throughout this study.

DT: DT consistently produces the lowest performance among the evaluated classifiers. Nevertheless, incorporating handcrafted linguistic features through early fusion results in noticeable improvements. The best overall performance is obtained with TF-IDF under early fusion, yielding an accuracy of 0.7164 and a macro F1-score of 0.6309, an increase of 0.0821 compared with the embedding-only baseline. The largest improvement, however, is observed with FastText, where the macro F1-score increases from 0.5155 to 0.6172, corresponding to a gain of 0.1017. This represents the largest early-fusion improvement achieved by the Decision Tree model across all feature representations.

RF: RF benefits consistently from early fusion regardless of the feature representation. Compared with the embedding-only baseline, macro F1-score improves by 0.0596 for BoW, 0.0594 for TF-IDF, 0.0474 for Word2Vec, and 0.0794 for FastText. The highest overall performance is obtained using FastText with early fusion, achieving an accuracy of 0.8033 and a macro F1-score of 0.7069. This makes it the third-best performing configuration in the study, following XGBoost with FastText under early fusion and Logistic Regression with BoW under early fusion.

KNN: The behaviour of KNN varies depending on the feature representation. For sparse representations, late fusion provides larger improvements, increasing macro F1-score by 0.0677 for BoW and 0.0896 for TF-IDF compared with the embedding-only baseline. In contrast, early fusion is more beneficial for dense embeddings, producing gains of 0.0334 for Word2Vec and 0.0372 for FastText. Overall, the best KNN performance is achieved with FastText under early fusion, yielding an accuracy of 0.7919 and a macro F1-score of 0.6789.

XGB: XGB combined with FastText under early fusion delivers the strongest performance among all evaluated configurations, achieving an accuracy of 0.7793 and a macro F1-score of 0.7361. Early fusion consistently improves macro F1-score across all feature representations, with gains of 0.0506 for BoW, 0.0504 for TF-IDF, 0.0492 for Word2Vec, and 0.0410 for FastText compared with the embedding-only models. In contrast, late fusion substantially reduces performance for BoW, TF-IDF, and FastText, with macro F1-scores dropping to the range of 0.48–0.55, while Word2Vec remains relatively stable at 0.6925. This behaviour appears to be specific to the out-of-fold stacked probability features used by XGBoost and warrants further investigation to better understand the interaction between the stacking strategy and the classifier.

To gain further insight into the strengths and limitations of the proposed approach, an error analysis was performed using confusion matrices. Since XGBoost with FastText under early fusion achieved the highest overall performance, it was selected as the representative model for detailed analysis. The corresponding confusion matrix is presented in Figure 4.

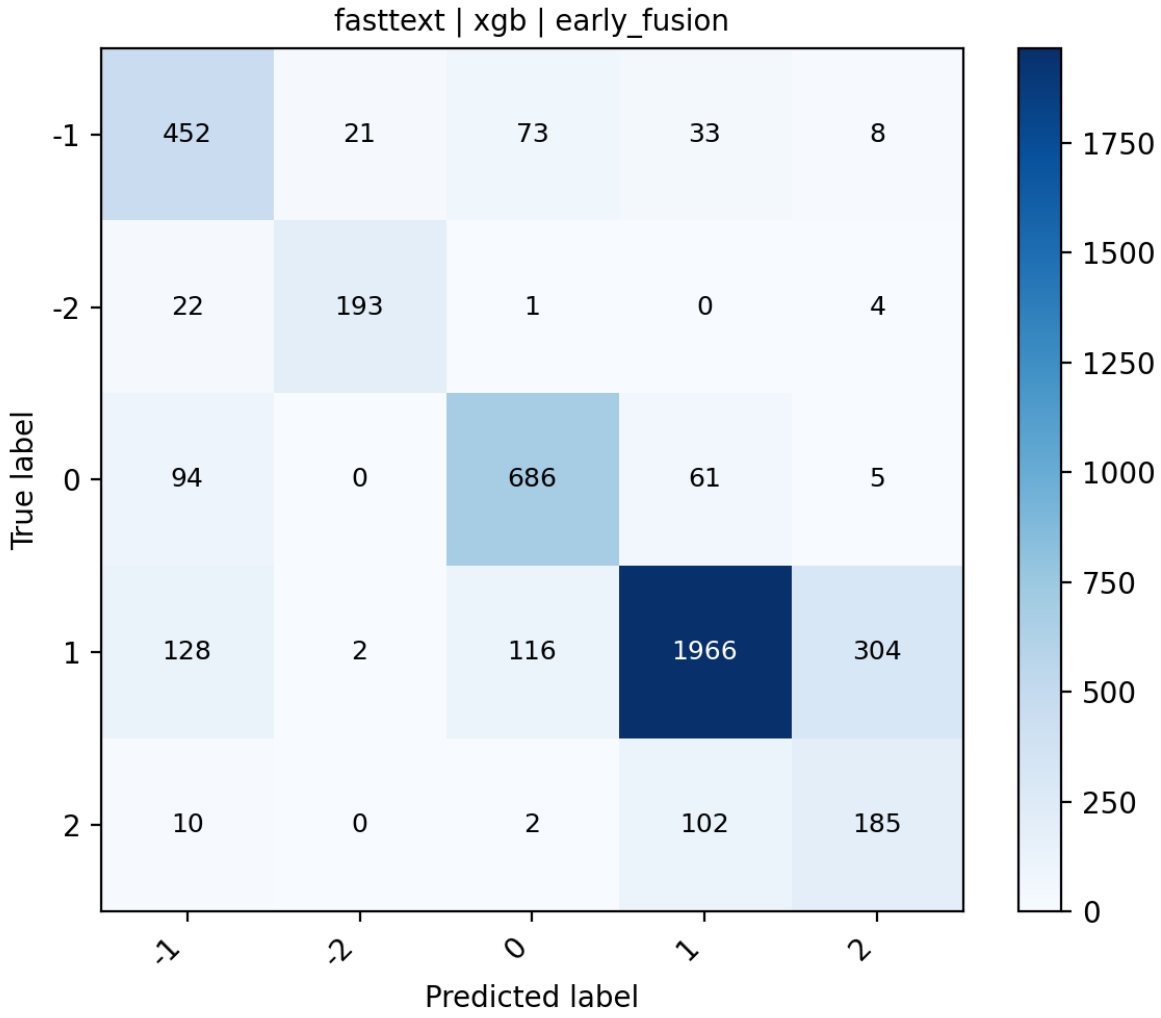

Figure 4: Fig. 4 confusion matrix

4. Fusion Strategy Comparison

To evaluate the contribution of the proposed linguistic features and examine the impact of different feature integration strategies, three experimental settings were compared for every classifier–representation pair: embedding-only (baseline), late fusion using out-of-fold (OOF) stacking, and early fusion through direct feature concatenation. The dataset, frozen train–test split, classifiers, and evaluation protocol were kept identical across all experiments to ensure a fair comparison. In addition, McNemar's test was performed to determine whether the observed performance differences between the models were statistically significant.

Ablation Study

Table 3 presents the results of the ablation study, comparing embedding-only, late fusion, and early fusion across the 26 valid classifier–representation combinations. For each combination, the table reports the macro F1-score

achieved under each setting, the improvement or decline relative to the embedding-only baseline, and the fusion strategy that produces the better overall performance.

Table 3: Ablation Study

Classifier	Representation	Embedding-only F1	Late Fusion F1	Δ Late	Early Fusion F1	Δ Early	Fusion Winner
LR	BoW	0.6984	0.6859	−0.0129	0.7112	0.0128	Early
LR	TF-IDF	0.6721	0.6685	−0.0036	0.7043	0.0322	Early
LR	Word2Vec	0.6067	0.6538	0.047	0.6423	0.0356	Late
LR	FastText	0.6201	0.6594	0.0393	0.6534	0.0333	Late
MNB	BoW	0.6549	0.6307	−0.0242	0.6642	0.0092	Early
MNB	TF-IDF	0.6128	0.6113	−0.0016	0.6206	0.0077	Early
SVM	BoW	0.6853	0.6916	0.0063	0.7013	0.0161	Early
SVM	TF-IDF	0.6367	0.6694	0.0327	0.6297	−0.0070	Late
SVM	Word2Vec	0.6414	0.6787	0.0373	0.671	0.0296	Late
SVM	FastText	0.677	0.6989	0.0218	0.6909	0.0139	Late
DT	BoW	0.5794	0.6069	0.0275	0.6249	0.0455	Early
DT	TF-IDF	0.5488	0.6058	0.057	0.6309	0.0822	Early
DT	Word2Vec	0.533	0.5942	0.0612	0.6091	0.0762	Early
DT	FastText	0.5155	0.5988	0.0833	0.6172	0.1017	Early
RF	BoW	0.61	0.6856	0.0755	0.6696	0.0596	Late
RF	TF-IDF	0.61	0.6849	0.0749	0.6694	0.0594	Late
RF	Word2Vec	0.652	0.6783	0.0263	0.6994	0.0474	Early
RF	FastText	0.6275	0.6844	0.057	0.7069	0.0794	Early
KNN	BoW	0.5798	0.6475	0.0677	0.5475	−0.0324	Late
KNN	TF-IDF	0.5465	0.6361	0.0896	0.5419	−0.0047	Late
KNN	Word2Vec	0.6302	0.6622	0.032	0.6636	0.0334	Early
KNN	FastText	0.6417	0.6753	0.0336	0.6789	0.0372	Early
XGB	BoW	0.6397	0.4824	−0.1573	0.6903	0.0506	Early
XGB	TF-IDF	0.6338	0.4834	−0.1503	0.6841	0.0504	Early
XGB	Word2Vec	0.6548	0.6925	0.0377	0.7041	0.0493	Early
XGB	FastText	0.6951	0.5543	−0.1408	0.7361	0.041	Early

The results demonstrate that incorporating handcrafted linguistic features generally improves sentiment classification performance. Compared with the embedding-only baseline, late fusion increases the macro F1-score in 19 of the 26 valid classifier–representation combinations, indicating that the handcrafted features provide complementary information that is useful for sentiment prediction. However, when the two fusion strategies are compared directly, early fusion performs better in 17 of the 26 combinations, suggesting that combining embedding

and handcrafted features before model training is, overall, the more effective integration strategy for this task. A closer examination reveals that the preferred fusion strategy varies across classifier families.

Tree-based and boosting algorithms, including DT, RF, and XGB, consistently benefit from early fusion. For example, DT achieves its largest improvement with FastText, where the macro F1-score increases by 0.1017 over the embedding-only baseline, while RF records an improvement of 0.0794 using the same representation. These findings indicate that tree-based models are better able to exploit the complementary information when embedding and handcrafted features are presented together during training, rather than being combined through stacked prediction probabilities.

SVM and KNN exhibit representation-dependent behaviour. KNN performs better with late fusion for sparse representations (BoW and TF-IDF), whereas early fusion actually reduces performance for these representations, with the largest decline being 0.0324 for BoW. In contrast, early fusion becomes more effective for dense embeddings (Word2Vec and FastText). SVM follows a different pattern, generally favouring late fusion for most feature representations, while BoW remains the only case where early fusion produces the highest macro F1-score.

Among all classifiers, XGB displays the largest difference between the two fusion strategies. When late fusion is applied, the macro F1-score decreases substantially compared with the embedding-only baseline, dropping by 0.1573 for BoW, 0.1503 for TF-IDF, and 0.1408 for FastText. Only Word2Vec maintains a modest improvement under late fusion. In contrast, early fusion consistently improves performance across all four feature representations, with macro F1-score gains ranging from 0.0410 to 0.0506. This behaviour suggests that XGB does not benefit from out-of-fold stacked probability features, even though it effectively utilizes the handcrafted linguistic features when they are directly integrated with the embedding features.

Overall, the ablation study confirms that early fusion is the more reliable and effective feature integration strategy for Gujarati-English code-mixed sentiment classification. While late fusion provides selective advantages for certain classifier–representation combinations, particularly for LR, SVM, and KNN under specific feature representations, early fusion consistently delivers stronger and more stable performance across most models. The best-performing configuration in the entire study is XGB with FastText under early fusion, achieving a macro F1-score of 0.7361. This result is consistent with the findings reported in Table 2 and further demonstrates the effectiveness of combining dense semantic representations with handcrafted linguistic features through direct feature integration.

5. Statistical Significance Testing (McNemar's Test)

To determine whether the performance improvements observed for the best-performing model are statistically significant, McNemar's test was conducted on paired test-set predictions. The reference model was the overall best-performing configuration, XGB with FastText under early fusion, and it was compared against every other classifier–condition combination within the FastText representation.

Table 4: McNemar's Test — Best Configuration (FastText + XGBoost + Early Fusion) vs. All Other FastText Configurations

Configuration Compared (FastText)	b (ref. correct, other wrong)	c (ref. wrong, other correct)	McNemar Statistic	p-value	Significant (p < 0.05)?
XGB – Late Fusion	1438	181	974.3891	6.63E-214	Yes
DT – Embedding-only	1245	336	521.4826	2.01E-115	Yes

LR – Handcrafted-only	911	176	495.6357	8.46E-110	Yes
DT – Handcrafted-only	871	179	454.7438	6.69E-101	Yes
DT – Late Fusion	828	183	410.2235	3.28E-91	Yes
XGB – Handcrafted-only	780	176	380.3441	1.05E-84	Yes
LR – Embedding-only	890	268	333.0233	2.11E-74	Yes
RF – Handcrafted-only	722	186	315.2258	1.59E-70	Yes
LR – Late Fusion	585	176	218.7438	1.70E-49	Yes
LR – Early Fusion	560	170	207.2890	5.36E-47	Yes
DT – Early Fusion	675	302	141.6418	1.16E-32	Yes
RF – Late Fusion	463	165	140.4602	2.11E-32	Yes
SVM – Early Fusion	424	161	117.3402	2.42E-27	Yes
KNN – Late Fusion	546	250	109.3279	1.38E-25	Yes
SVM – Late Fusion	372	166	78.1134	9.73E-19	Yes
XGB – Embedding-only	412	242	43.6713	3.88E-11	Yes
KNN – Handcrafted-only	583	430	22.8075	1.79E-06	Yes
RF – Early Fusion	278	385	16.9472	3.84E-05	Yes
KNN – Embedding-only	595	467	15.1874	9.74E-05	Yes
SVM – Handcrafted-only	527	426	10.4932	0.0012	Yes

RF – Embedding-only	572	494	5.5619	0.0184	Yes
KNN – Early Fusion	424	480	3.3462	0.0674	No
SVM – Embedding-only	423	464	1.8038	0.1792	No

In addition to the three experimental settings presented in Table 2 (embedding-only, late fusion, and early fusion), the comparison also included a handcrafted-only baseline, in which each classifier was trained exclusively on the aggregated linguistic features without any embedding input. Since Multinomial Naïve Bayes was not evaluated with FastText, the analysis consisted of six classifiers evaluated under four conditions, resulting in 24 FastText configurations. Using XGB with FastText under early fusion as the reference model, a total of 23 pairwise comparisons were performed.

For each comparison, b represents the number of test instances correctly classified by the reference model but misclassified by the competing model, while c represents the number of instances correctly classified by the competing model but misclassified by the reference model. The continuity-corrected McNemar statistic and the corresponding p-value were calculated from these discordant pairs, with $p < 0.05$ considered statistically significant.

The results show that the proposed reference model is statistically significantly better than 21 of the 23 FastText classifier–condition combinations evaluated. The strongest evidence of superiority is observed when the reference model is compared with the weakest-performing configurations, particularly Decision Tree and Logistic Regression under the embedding-only and handcrafted-only settings, where the p-values are as small as 6.63×10^{-214}. As expected, the level of statistical significance decreases as the performance of the competing model approaches that of the reference model.

Only two comparisons do not reach statistical significance. These correspond to K-Nearest Neighbors under early fusion ($p = 0.067$) and Support Vector Machine under the embedding-only setting ($p = 0.179$). These two configurations also achieve the closest test accuracies to the reference model (0.7919 and 0.7885, respectively, compared with 0.7793 for the reference model). This outcome is consistent with the behaviour of McNemar's test, which becomes less sensitive when two classifiers make similar predictions and exhibit highly overlapping error patterns.

The comparison between the reference model and the alternative XGBoost configurations further demonstrates the effectiveness of the proposed fusion strategy. XGBoost with FastText under early fusion performs significantly better than both its late-fusion and handcrafted-only counterparts, providing statistical evidence that directly integrating embedding and handcrafted linguistic features offers a more effective representation than combining predictions after training or relying solely on handcrafted features.

6. Conclusion

This study presents a comprehensive machine learning framework for five-class sentiment classification of Gujarati–English code-mixed text, addressing an important yet relatively underexplored problem in multilingual natural language processing. A large-scale, manually annotated dataset containing 44,672 YouTube comments was developed from a wide range of content domains. The dataset includes both Romanized and native Gujarati scripts and, to the best of our knowledge, is among the first resources of this scale designed specifically for fine-grained sentiment analysis of this language pair.

A major contribution of this work is the incorporation of word-level linguistic information, including language identity, sentiment polarity, and intensifier presence, obtained using a multi-task DistilBERT-based tagger. These linguistic features were combined with conventional text representations through a late fusion framework based on

out-of-fold (OOF) stacking and were systematically compared with embedding-only and early fusion approaches. Four feature representation techniques and seven machine learning classifiers were evaluated under all experimental settings.

The experimental results show that both sparse representations (BoW and TF-IDF) and the FastText dense embedding provide competitive performance across different classifiers, whereas Word2Vec does not emerge as the best-performing representation for any classifier in this study. The ablation analysis further demonstrates that incorporating linguistic features is generally beneficial. Compared with the embedding-only baseline, late fusion improves macro F1-score in 19 of the 26 valid classifier–representation combinations, indicating that explicit linguistic information provides complementary knowledge beyond the text embeddings alone. However, early fusion produces better performance in 17 of the 26 combinations, suggesting that direct integration of embedding and linguistic features is a more effective strategy for most classifiers. The best overall performance is achieved by XGB with FastText under early fusion, obtaining an accuracy of 0.7793 and a macro F1-score of 0.7361.

The error analysis reveals that the most common classification errors occur between closely related sentiment categories, particularly Neutral and Positive, as well as between extreme and non-extreme sentiment classes. These misclassifications are largely influenced by implicit sentiment expressions in user-generated content, spelling inconsistencies, transliteration variations in Romanized Gujarati, and the inherent difficulty of identifying minority and extreme sentiment categories, even when cost-sensitive learning techniques are employed.

Overall, this research presents a linguistically informed and reproducible framework for sentiment classification of Gujarati–English code-mixed text. By combining handcrafted linguistic features with multiple feature integration strategies and extensive comparative evaluation, the study demonstrates the value of incorporating linguistic knowledge into traditional machine learning models. In addition to proposing an effective sentiment classification framework, this work also contributes a large, manually annotated benchmark dataset that can support future research on sentiment analysis for low-resource, multilingual, and code-mixed languages.

7. Future Work

Building upon the findings of this study, future work will focus on extending the proposed framework towards more advanced modelling approaches and practical real-world applications.

7.1 Deep Learning Integration: An important next step is to investigate deep learning-based approaches beyond traditional machine learning classifiers. We plan to begin with Single-Layer and Multi-Layer Perceptron (SLP and MLP) models, followed by sequential architectures such as RNN, LSTM, Bi-LSTM, GRU, and Bi-GRU. These models will be evaluated using both standalone contextual representations and in combination with the handcrafted linguistic features introduced in this work. The objective is to examine their ability to capture long-range contextual dependencies and the complex linguistic patterns present in Gujarati–English code-mixed text.

7.2 Transformer-Based Fine-Tuning: Future research will also explore transformer-based language models through fine-tuning approaches involving models such as BERT, mBERT, MuRIL, and XLM-R. Given their ability to learn rich contextual representations and their prior training on multilingual and Indian language resources, these models are expected to provide stronger representations for code-mixed sentiment analysis. Additionally, we aim to investigate hybrid architectures that combine transformer-based contextual embeddings with explicit linguistic information through mechanisms such as feature gating or attention-based integration, extending the linguistic feature fusion strategy proposed in this work.

7.3 Dataset Expansion and Cross-Domain Evaluation: The current dataset is collected exclusively from YouTube comments, providing a focused evaluation setting. Future work will expand the dataset to include additional social media platforms such as Twitter/X, WhatsApp, and Facebook, where code-mixing behaviours and communication styles may differ. Cross-domain experiments will then be conducted to evaluate the robustness and generalizability of the proposed approaches across diverse online environments.

7.4 Deployment on Edge Devices: Another potential direction is the deployment of the sentiment analysis framework on resource-constrained edge devices, such as Raspberry Pi-based systems. This would enable real-time sentiment analysis for applications including regional-language content moderation, mobile feedback platforms, vernacular social media monitoring, and intelligent conversational systems. Optimizing the models for efficient inference while maintaining classification performance will be an important consideration for such applications.

REFERENCES

1. P. A. Joshi, V. M. Pathak, and M. R. Joshi, "Sentiment analysis from social media data in code-mixed Indian languages using machine learning classifiers with TF-IDF and weighted word features," in Data-Intensive Research. Singapore: Springer Nature, 2024, pp. 203–222, doi: 10.1007/9789819991792_16.
2. P. Mishra, P. Danda, and P. Dhakras, "Code-mixed sentiment analysis using machine learning and neural network approaches," arXiv preprint, arXiv:1808.03299, 2018, doi: 10.48550/arXiv.1808.03299.
3. M. G. Jhanwar and A. Das, "An ensemble model for sentiment analysis of Hindi–English code-mixed data," arXiv preprint, arXiv:1806.04450, 2018, doi: 10.48550/arXiv.1806.04450.
4. A. Prabhu, A. Joshi, M. Shrivastava, and V. Varma, "Towards sub-word level compositions for sentiment analysis of Hindi–English code-mixed text," in Proc. 26th Int. Conf. on Computational Linguistics (COLING), 2016, pp. 2482–2491.
5. R. Jamatia, S. Choudhury, and S. Bandyopadhyay, "Deep learning vs. traditional machine learning for sentiment analysis of code-mixed text," in Proc. 12th Int. Conf. on Natural Language Processing (ICON), 2020, pp. 1–10.
6. S. Khan and R. Mamidi, "Ensemble CNN–LSTM for code-mixed sentiment analysis using word embeddings," Int. Arab J. Inf. Technol., vol. 19, no. 3, pp. 271–280, 2022, doi: 10.34028/iajit/19/3/6.
7. K. Kogilavani et al., "Hybrid CNN–BiLSTM model for sentiment analysis of Tamil–English code-mixed text," Comput. Speech Lang., vol. 76, p. 101407, 2022.
8. S. Chanda, A. Mishra, and S. Pal, "Sentiment analysis of code-mixed Dravidian languages leveraging pretrained models and word-level language tags," Nat. Lang. Process., vol. 31, no. SI-2, pp. 477–499, 2024.
9. G. Takawane et al., "Leveraging language identification to enhance code-mixed text classification," arXiv preprint, arXiv:2306.04964, 2023.
10. S. Mahata, D. Das, and S. Bandyopadhyay, "Sentiment classification of code-mixed tweets using bi-directional RNN and language tags," in Proc. 1st Workshop on Speech and Language Technologies for Dravidian Languages, 2021, pp. 28–35.
11. E. Hossain, O. Sharif, and M. M. Hoque, "Multilingual code-mixed hope speech detection using cross-lingual representation learning," in Proc. LT-EDI, 2021, pp. 1–10.
12. E. Asrarul et al., "Sentiment analysis of code-mixed Tamil and Tulu text using transformer-based models," in Proc. DravidianLangTech–EACL, 2024, pp. 205–211.
13. B. R. Chakravarthi et al., "DravidianCodeMix: A benchmark dataset for sentiment analysis and offensive language detection," Lang. Resour. Eval., vol. 56, no. 3, pp. 765–806, 2022.
14. M. Gokani and R. Mamidi, "GSAC: A Gujarati sentiment analysis corpus from Twitter," in Proc. WASSA, 2023, pp. 129–137.
15. A. S. Patel, J. Modi, and D. R. Patel, "Text classification of Gujarati newspaper headlines," Int. J. Comput. Trends Technol., vol. 70, no. 3, pp. 164–170, 2022.
16. S. Joshi, "Sentiment analysis of Gujarati film reviews using machine learning," Int. J. Comput. Appl., 2022.
17. R. Nayak and R. Joshi, "L3Cube-HingCorpus and HingBERT: A code-mixed Hindi–English dataset and language models," arXiv preprint, arXiv:2204.08398, 2022.
18. M. Shakeel, M. Younas, and M. S. Khan, "Sentiment analysis of Russian–English election tweets," in Proc. IEEE DSAA, 2020, pp. 1–10.
19. S. Yadav and T. Chakraborty, "Zero-shot sentiment analysis for code-mixed data," in Proc. AAAI, vol. 35, no. 18, pp. 15941–15942, 2021.
20. N. A. Semary et al., "Enhancing machine learning-based sentiment analysis through feature extraction," PLOS ONE, vol. 19, no. 2, p. e0294968, 2024.
21. K. Ayyub et al., "A feature-based approach for sentiment quantification using machine learning," Electronics, vol. 11, no. 6, p. 846, 2022.
22. K. Koyyalagunta and M. Supriya, "Transformer-based sentiment analysis on code-mixed data," Procedia Comput. Sci., vol. 233, pp. 682–691, 2024.
23. L. Gohil and D. Patel, "Sentiment analysis of Gujarati text using Gujarati SentiWordNet," Int. J. Innov. Technol. Explor. Eng., vol. 8, no. 9, pp. 2290–2293, 2019.

24. N. Choudhary et al., "Sentiment analysis of code-mixed languages leveraging resource-rich languages," in Proc. CICLing, 2018.
25. B. G. Patra, D. Das, and A. Das, "Sentiment analysis of code-mixed Indian languages: An overview of SAIL_Code-Mixed Shared Task @ICON-2017," arXiv preprint, arXiv:1803.06745, 2018.
26. R. Srinivasan and C. N. Subalalitha, "Sentimental analysis from imbalanced code-mixed data using machine learning approaches," Distrib. Parallel Databases, vol. 41, no. 1, pp. 37–52, 2023, doi: 10.1007/s10619-021-07331-4.
27. L. Advani, C. Lu, and S. Maharjan, "C1 at SemEval-2020 Task 9: SentiMix: Sentiment analysis for code-mixed social media text using feature engineering," in Proc. 14th Int. Workshop on Semantic Evaluation (SemEval-2020), Barcelona, Spain, Dec. 2020.
28. S. Javdan, T. S. Atai, and B. Minaei-Bidgoli, "IUST at SemEval-2020 Task 9: Sentiment analysis for code-mixed social media text," in Proc. 14th Int. Workshop on Semantic Evaluation (SemEval-2020), Barcelona, Spain, Dec. 2020, arXiv:2007.12733.
29. D. Patel and R. Parikh, "Language identification and translation of English and Gujarati code-mixed data," in Proc. 2020 Int. Conf. on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, Feb. 2020, pp. 1–4, doi: 10.1109/ic-ETITE47903.2020.410.
30. P. Shah, P. Swaminarayan, and M. Patel, "Sentiment classification for film reviews in Gujarati text using machine learning and sentiment lexicons," J. ICT Res. Appl., vol. 17, no. 1, pp. 1–16, 2023, doi: 10.5614/itbj.ict.res.appl.2023.17.1.1.