Sentiment Analysis of Code-Mixed Gujarati-English Text Using Machine Learning Techniques with Language Augmentation

Chirag D. Shah
Research Scholar,
Department Of Computer Science
VNSGU,Surat, India
E-mail : chirags268@ gmail.com
Mo:- 9825902140

Dr. Shailesh A. Chaudhari
Assistant Professor,
Department of IT and ICT,
JP Dawer Institute of Information
Science and Technology, VNSGU, Surat, India
E-mail : sachaudhari@vnsgu.ac.in
Mo:- 9825295669

Abstract:

Code-mixed text, which combines words from multiple languages in a single sentence, is common on social media. This poses challenges for traditional Natural Language Processing (NLP) methods. In this study, we address five-class sentiment classification of Gujarati–English code-mixed text across Roman and Gujarati scripts. We propose a language-augmented approach that incorporates word-level language identification tags generated by fine-tuning of DistilBERT model to enhance machine learning–based sentiment classification. These language tags integrated with traditional text representations that includes Bag of Words (BoW), Term Frequency–Inverse Document Frequency (TF-IDF), Word2Vec, and FastText. We evaluate several classifiers Logistic Regression (LR), Multinomial Naive Bayes (MNB), Support Vector Machines (SVM), K-Nearest Neighbours (KNN), Decision Trees (DT), Random Forest (RF), and Extreme Gradient Boost (XGB). We handle class imbalance using the Synthetic Minority Oversampling Technique (SMOTE) and optimized hyper-parameters with GridSearchCV. Results show that augmenting language tags consistently improves sentiment classification performance across all feature representations. On the original dataset, BoW with Logistic Regression achieves the best performance with 0.79 accuracy and 0.70 macro F1-score; with SMOTE, TF-IDF with SVM achieves 0.76 accuracy and 0.70 macro F1-score.While dense embeddings like Word2Vec and FastText capture semantic meaning but sparse representations yield stronger gains when augmented with language tags. Our work contributes toward more inclusive and reliable sentiment analysis through inclusion of explicit language awareness into machine learning models. By applying word-level language cues alongside commonly used text representations, the proposed approach improves the ability of sentiment classifiers to interpret mixed-language expressions found in everyday online communication. This can benefit applications such as public opinion analysis, content moderation, customer feedback mining, and language translation, especially while studying linguistically diverse content.

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. India's linguistic diversity (constitution recognizing 22 official languages), and hundreds of regional dialects and linguistic variations coexisting across the country has a major impact on these online interactions. This rich multilingual landscape creates unique challenges and opportunities for NLP research, mainly in the analysis of informal and user-generated text.

Gujarati, 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. In urban and semi-urban areas, the growing use of English in digital communication often produces code-mixed text, where Gujarati and English elements coexist within a single sentence or utterance. This phenomenon 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.

The COVID-19 pandemic accelerated Internet adoption across India, reducing the digital divide between urban and rural regions. As a result, native language speakers, particularly in rural and semi-urban Gujarat, increasingly participate in online discourse. Many of these users express themselves in code-mixed Gujarati-English, using either Romanized or native Gujarati scripts, often without formal training in Indic keyboard input systems.

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 fluid 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, of which 44,672 were manually annotated for sentiment polarity. 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 five categories: Extremely Negative, Negative, Neutral, Positive, and Extremely Positive. The dataset captures a wide range of sentiment expressions that enables us to capture subtle variations in opinion intensity. We manually performed annotations to ensure high quality and consistency across diverse domains and writing styles. The resulting 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 language identification tags generated by a fine-tuned DistilBERT model. Each word is tagged as Gujarati (GJ), English (EN), or Other (OT), and these tags are concatenated with the text features to provide explicit linguistic context to the classifiers. We evaluate seven classifiers and address class imbalance using SMOTE and hyperparameters optimized via GridSearchCV. The main contributions of our work are as follows:

1. A large-scale, manually annotated Gujarati–English code-mixed sentiment dataset covering Romanized scripts.
2. A word-level language identification module using fine-tuned DistilBERT, integrated as a feature augmentation strategy for sentiment classification.
3. A comprehensive comparative evaluation of traditional and hybrid feature representations across multiple machine learning classifiers, with and without class balancing.

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. (2024) conducted cross-lingual comparisons involving Malayalam–English, Tamil–English, Hindi–English, and Bengali–English data. They used Weighted Word Unigram (WWU) representation paired with a Multinomial Naïve Bayes (MNB) classifier. The approach 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. (2018) examined character n-gram features and TF-IDF vectorisations in ensemble configurations for Hindi–English and Bengali–English corpora. They demonstrated that sub-word representations can partially compensate for the inconsistencies inherent in romanized code-mixed text.

Jhanwar and Das (2018) 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. (2018) provided 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, surpassing several deep learning submissions that were constrained by limited training data.

Addressing the issue of class imbalance in code-mixed corpora, Srinivasan and Subalalitha (2023) 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 resulted in improvements in macro F1 relative to the baselines. Within the context of the SemEval-2020 Task 9 (SentiMix) competition, Advani et al. (2020) 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. (2020) 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 have improved the ability of sentiment analysis models to handle the irregular syntax and morphological complexity of code-mixed text. Prabhu et al. (2016) developed a morpheme-aware subword LSTM for Hindi–English code-mixed data. Their model outperformed character-level LSTM and unigram MNB baselines, achieving an accuracy of 69.7% and a macro F1 score of 0.658. Jamatia et al. (2020) compared traditional machine learning classifiers with different deep learning architectures on English–Hindi and English–Bengali datasets. Their results showed that deep learning models could provide considerable performance improvements, particularly when sufficient training data were available.

Khan and Mamidi (2022) developed a CNN–LSTM ensemble using Word2Vec, GloVe, and FastText embeddings. Kogilavani et al. (2022) combined CNN and BiLSTM layers with class-resampling preprocessing for Tamil–English sentiment analysis and achieved an accuracy of 0.66. These studies show that hybrid deep learning architectures can combine different types of features and improve sentiment classification for code-mixed text.

2.3 Language-Aware Representations and Transformer Models

Recent studies have shown that language information can be useful for sentiment analysis of code-mixed text. Chanda et al. (2024) added word-level language identification (LID) tags

to multilingual BERT (mBERT) for Tamil–English, Kannada–English, and Malayalam–English data. Their results showed improvements over models without language tags. Takawane et al. (2023) also investigated language identification features with BERT-based models for Hindi–English datasets and reported strong performance using language-aware representations.

Mahata et al. (2021) incorporated word-level language identification tags into a Bi-RNN model for English–Tamil sentiment classification and reported a weighted F1 score of 0.58. Hossain et al. (2021) investigated multilingual code-mixed hope-speech detection using several transformer and neural architectures. Their results further demonstrated the usefulness of multilingual pretrained models for code-mixed language tasks.

Shakeel et al. (2020) investigated Russian–English sentiment classification using attention-based LSTM, mBERT, and XLM-R. Yadav and Chakraborty (2021) proposed a zero-shot approach for English–Spanish code-mixed sentiment analysis using cross-lingual embeddings. Nayak and Joshi (2022) introduced the L3Cube-HingCorpus and HingBERT models for Hindi–English code-mixed language processing, providing useful pretrained resources for Hinglish sentiment analysis.

Translation and transliteration have also been used to address the limited availability of annotated code-mixed data. Koyyalagunta and Supriya (2024) combined transliteration, machine translation, and transformer fine-tuning for Hindi–English, Malayalam–English, Tamil–English, and Telugu–English language pairs. Their results showed improvements of up to 6% in accuracy and 9% in F1 compared with non-translation baselines. These findings suggest that translation and transliteration can be useful when annotated resources for code-mixed languages are limited.

2.4 Shared Tasks, Benchmarks, and Evaluation Frameworks

Shared tasks and benchmark datasets have played an important role in evaluating different approaches to code-mixed sentiment analysis. SemEval-2020 Task 9, known as SentiMix, provided benchmark datasets for Hinglish and Spanglish sentiment classification. Several participating systems used feature engineering, ensemble methods, and neural or transformer-based models (Advani et al., 2020; Javdan et al., 2020).

The DravidianCodeMix benchmark introduced by Chakravarthi et al. (2022) supported comparative evaluation of sentiment analysis and offensive language identification for Tamil–English, Malayalam–English, and Kannada–English. Asrarul et al. (2024) further evaluated machine learning, deep learning, and transformer-based models for Tamil–English and Tulu–English sentiment analysis in the DravidianLangTech shared task.

Choudhary et al. (2018) proposed the Siamese Attention-based Code-Mixed Text (SACMT) framework for low-resource code-mixed sentiment classification. Their approach used a Siamese architecture and contrastive learning to transfer sentiment information from resource-rich language pairs. The results showed improvements over the baseline models for Hindi–English and Telugu–English data.

2.5 Sentiment Analysis in Gujarati and Gujarati–English

Research on Gujarati sentiment analysis has increased gradually, although it remains less developed than research on Hindi–English and several Dravidian language pairs. Gohil and Patel (2019) developed Gujarati SentiWordNet (G-SWN), providing a lexical resource for Gujarati sentiment analysis. Joshi (2022) applied Naïve Bayes classifiers using count-based and TF-IDF features to Gujarati film reviews. Patel et al. (2022) used TF-IDF and n-gram features with an SVM classifier for Gujarati newspaper headline classification and reported accuracy close to 92%.

Shah et al. (2023) compared lexicon-based and machine-learning approaches for Gujarati film-review sentiment analysis. Their GMLSA framework included Logistic Regression, Random Forest, k-Nearest Neighbours, SVM, and Naïve Bayes classifiers with TF-IDF and count-vectorisation features. The machine-learning-based approach generally performed better than the lexicon-based approach, with SVM providing the strongest individual performance.

Gokani and Mamidi (2023) introduced the Gujarati Sentiment Analysis Corpus (GSAC), a manually annotated collection of Gujarati Twitter posts. They also reported baseline results using TF-IDF with SVM and Logistic Regression. The results improved further after applying SMOTE-based oversampling.

Patel and Parikh (2020) addressed preprocessing challenges in English–Gujarati code-mixed social media data. Their work included language identification, text normalisation, and transliteration-to-native-script translation. However, the main focus was on linguistic preprocessing rather than five-class sentiment classification.

Overall, existing Gujarati studies have mainly focused on monolingual Gujarati sentiment analysis or preprocessing of Gujarati–English code-mixed text. Comparatively less research has investigated five-class sentiment classification of Gujarati–English code-mixed data using language-aware features. In addition, publicly available annotated resources specifically designed for Gujarati–English code-mixed sentiment classification remain limited. These limitations provide a strong motivation for the present study.

2.6 Feature Engineering, Auxiliary Techniques, and Summary

Apart from model architecture, feature representation plays an important role in sentiment classification. Semary et al. (2024) compared Bag-of-Words, TF-IDF, Word2Vec, and GloVe representations with SVM classifiers and found that TF-IDF provided a favourable balance between classification performance and computational cost. Ayyub et al. (2022) combined linguistic features, such as part-of-speech information and sentiment-polarity indicators, with Word2Vec and GloVe embeddings. Their findings showed that combining different types of features can improve sentiment classification performance.

Overall, previous research has mainly concentrated on Hindi–English and Dravidian code-mixed language pairs, while Gujarati–English has received comparatively less attention. Existing Gujarati research has largely focused on monolingual Gujarati sentiment analysis, Gujarati sentiment resources, or preprocessing techniques for Gujarati–English text. Therefore, there is a need for further research on fine-grained Gujarati–English code-mixed sentiment classification, particularly using five sentiment classes and language-aware features. The present study addresses this gap by developing a computationally efficient machine learning framework that integrates word-level language identification with different feature-representation techniques.

Research Gap

Although sentiment analysis of code-mixed text has received considerable attention in recent years, most existing studies have focused on relatively well-resourced language pairs, particularly Hindi–English and other Indian or Dravidian language combinations (Jhanwar & Das, 2018; Joshi et al., 2024; Prabhu et al., 2016; Takawane et al., 2023). Research has also examined Spanish–English code-mixed sentiment classification, including feature-engineering and zero-shot approaches (Advani et al., 2020; Yadav & Chakraborty, 2021). However, much of the existing work concentrates on binary or three-class sentiment classification, with comparatively less attention given to fine-grained sentiment categories that capture different levels of sentiment intensity.

In comparison, research on Gujarati sentiment analysis remains relatively limited. Existing studies have primarily focused on monolingual Gujarati text, including Gujarati Twitter data, newspaper headlines, film reviews, and lexicon-based sentiment analysis (Gohil & Patel, 2019; Gokani & Mamidi, 2023; Joshi, 2022; Patel et al., 2022; Shah et al., 2023). Patel and Parikh (2020) addressed English–Gujarati code-mixed data through language identification, text normalisation, and transliteration, but their work primarily focused on linguistic preprocessing rather than sentiment classification. Consequently, Gujarati–English code-mixed sentiment

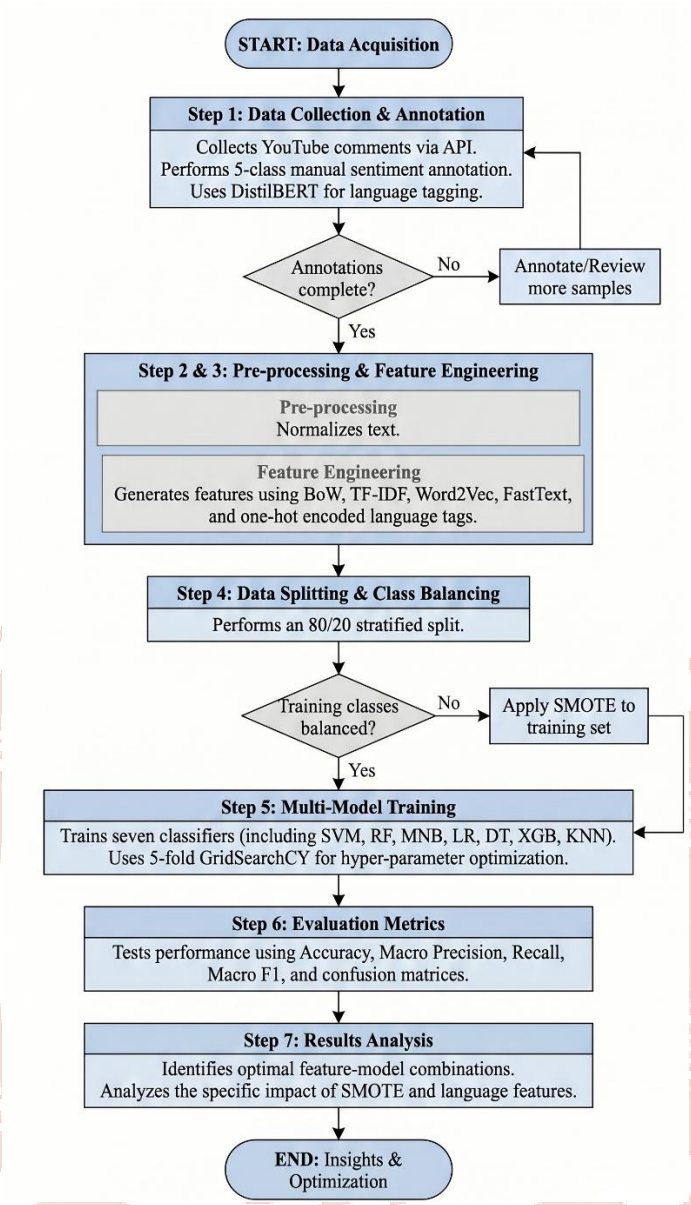
analysis, particularly using large-scale annotated datasets and five-class sentiment classification, remains comparatively underexplored.

Furthermore, language-aware modelling has demonstrated promising results for Hindi–English and Dravidian code-mixed language pairs. Studies incorporating word-level language identification tags into neural and transformer-based architectures have reported improvements over models without explicit language information (Chanda et al., 2024; Mahata et al., 2021; Takawane et al., 2023). However, the effectiveness of language-aware features for sentiment classification in low-resource Gujarati–English code-mixed text has received limited investigation.

To address these research gaps, the present study develops a comprehensive framework for five-class sentiment classification of Gujarati–English code-mixed text. The proposed approach integrates word-level language identification with different feature-representation techniques to investigate whether explicit language information can improve sentiment classification performance in this low-resource setting. The study therefore contributes to the relatively underexplored area of Gujarati–English code-mixed sentiment analysis and provides an empirical evaluation of language-aware feature integration for fine-grained sentiment classification.

2.7 Research Objectives and Hypothesis

Based on the identified research gaps, the main goal of this study is to develop and evaluate effective ML based approaches for sentiment analysis of Gujarati–English code-mixed text in social media environments. Our study explores how different feature representation techniques and language augmentation impacts sentiment classification performance in multilingual and low-resource settings.



The specific research objectives of this work are as follows:

1. To curate an annotated dataset of Gujarati–English code-mixed social media comments for five-class sentiment analysis.
2. To develop a word-level language identification mechanism capable of tagging Gujarati, English, and Other words.
3. To evaluate the effectiveness of different feature representation techniques including BOW, TF-IDF, Word2Vec, and FastText for sentiment classification in code-mixed text.
4. To assess the impact of word level language-tag augmentation into sentiment classification models.
5. To conduct a comparative study of machine learning classifiers using optimized hyperparameters for five-class sentiment prediction.

We assume that incorporating word-level language identification with feature-based representations will improve sentiment classification performance for Gujarati–English code-mixed text compared to models that rely solely on conventional feature extraction techniques.

2.8 Significance, and Novelty of the Study

The widespread use of code-mixed language on social media reflects natural communication patterns in multilingual societies. However, most existing sentiment analysis systems are not designed to effectively process such text, particularly for under-resourced languages like Gujarati. The challenges are further amplified by informal writing styles, script variations, character elongation, and frequent mixing with English, which significantly reduce the performance of conventional natural language processing approaches.

The novelty in this work lies in focusing on fine-grained sentiment analysis of Gujarati–English code-mixed text. Prior works primarily focuses on binary or coarse-grained classification. We apply a five-class sentiment scheme (extreme negative to extreme positive) to capture nuanced variations in sentiment intensity, which are crucial for understanding user opinions in social media contexts. Also in this work we integrate word-level language identification with diverse feature representation techniques for sentiment classification in a low-resource code-mixed setting. This is a combination a combination that, to the best of our knowledge, has not been explored for Gujarati–English code-mixed sentiment analysis. Additionally, we present a large-scale manually annotated dataset for Gujarati–English code-mixed text, which, to the best of our knowledge, is among the first to support fine-grained sentiment analysis in this domain. Furthermore, we conduct a systematic comparative evaluation of traditional statistical features and embedding-based representations, providing new insights into effective feature engineering strategies for multilingual and low-resource environments.

Overall, this work contributes to bridging the gap in sentiment analysis for code-mixed, low-resource languages by offering both a novel methodological framework and valuable linguistic resources.

2.9 Contributions of This Work

1. Dataset Curation: We created a high-quality dataset of 44,672 Gujarati–English code-mixed YouTube comments. The dataset was manually annotated for sentence level sentiment labels by the authors. The data was collected from diverse genres of real-world YouTube channels.
2. Word Level Language Identification: We implemented a lightweight, fine-tuned DistilBERT model to perform accurate word-level language tagging with labels GJ,

EN and OT of the code-mixed text.

3. Comparative Study: We conducted a comprehensive comparison of four vectorization techniques—BoW, TF-IDF, Word2Vec, and FastText— to evaluate their effectiveness with multiple machine learning classifiers.
4. Hyper-parameter Optimization: We applied GridSearchCV to fine-tune each machine learning classifier to ensure optimal performance for different feature combinations.

The proposed framework is designed to be extensible to other low-resource, code-mixed language pairs and provides a replicable methodology for integrating linguistic features into machine learning pipelines for multilingual NLP tasks.

3. 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 detection, 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 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.Raw data was retrieved using 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. The 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. We performed annotation manually given the 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. Therefore, all sentences were manually annotated to ensure high consistency and uniformity across the dataset.

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:** Strong disappointment, anger, or extreme criticism, often signalled 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:** General dissatisfaction or mild negative opinion without extreme emphasis.
  Examples:
  - "bakwas che"
  - "not so good"
  - "thodu boring lagyu"

- "didn't like it much"

- **Neutral:** Factual or descriptive comments lacking strong sentiment, including 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:** Mild positive sentiment, appreciation, or light praise without strong intensifiers.
  Examples:
  - "saras che bhai"
  - "nice acting"
  - "good try"
  - "mast lage che"
- **Extremely Positive:** Strong appreciation or excitement with intensified expressions such as "ekdum", "khub", "bov", "ghanu", "super", repeated characters, or emoticons.
  Examples:
  - "ekdum saras movie che"
  - "goooooood job"
  - "mastttttt 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.



Fig 2. Top 50 Sentiment words

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:

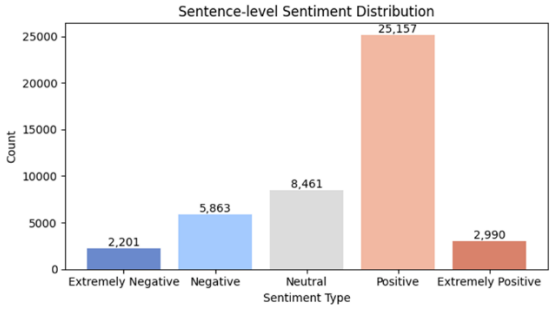


Fig. 3. Sentiment class distribution

The dataset shows a skew toward the Positive class, which could bias models to favour majority classes during training. To address this, we applied SMOTE to generate synthetic samples for minority classes, balancing the dataset and improving model robustness.

During the split into training and testing subsets, we employed stratified sampling to preserve the original class proportions, ensuring consistent evaluation across all sentiment categories.

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 language identification model and 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 Word-level Language Identification using DistilBERT

Accurately identifying the language of each word is crucial for sentiment analysis of code-mixed text. The annotation involved assigning one of three tags to each word: GJ (Gujarati), EN (English), or OT (Other). We used all 77,761 sentences scraped totalling to 732,917 words. We developed a Telegram based Python bot to streamline annotation process. Two volunteers performed the annotation, with annotators reviewing one comment at a time. For each word of a comment, they would have to choose one of the three mentioned options. Annotators were also given the option to skip any word if they were unable to determine its meaning and were allowed to stop annotating whenever they felt tired. After they finished annotation, the inter

annotator agreement was evaluated using Cohen's Kappa. The resulting Cohen's Kappa score of 0.95 indicates near-perfect inter-annotator agreement. At last, we reviewed all words which were marked as Skipped and then assigned the appropriate tags to these words. The annotated dataset was unbalanced with 39,017 Gujarati (GJ), 19,989 English (EN) and 17,839 Other (OT) uniquely identified tags. We fine-tuned a lightweight DistilBERT model specifically for word-level language identification. DistilBERT offers performance comparable to BERT—achieving over 95% accuracy—while being computationally efficient. The fine-tuned DistilBERT achieved an accuracy of 97.09%, F1 score of 97.01% and precision of 97.02%.

For every sentence, the model outputs language tags for each word. These word-level tags are then converted into one-hot encoded features and integrated into the sentiment analysis pipeline. This language-level annotation provides two key advantages:

1. It helps capture the structural characteristics of code-mixed text.
2. It allows the sentiment model to incorporate language-specific information, thereby improving its ability to interpret emotionally nuanced words tied to their language of origin.

Our dataset has multiple dialects and regional variations. So, this word-level tagging helps in disambiguating mixed expressions and improve the overall accuracy of sentiment classification.

3.4 Feature Engineering and Text Representation

To effectively capture the nuances of sentiment in Romanized Gujarati–English code-mixed text, we designed a comprehensive set of feature representations. We had two objectives in doing this:

1. To compare traditional sparse text representations with dense, embedding-based representations for sentiment classification.
2. To evaluate the impact of incorporating word-level language information predicted by our DistilBERT-based language tagger.

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 structures, it serves as a robust baseline for sentiment analysis. Importantly, we enhanced BoW vectors by integrating word-level language tags, enabling the model to differentiate words based on their language context—an essential factor in code-mixed scenarios. We generated BoW features using the CountVectorizer implementation from scikit-learn.

3.4.2 TF-IDF

Building upon BoW, we implemented TF-IDF to weigh each word according to its importance within the corpus. TF-IDF down weights frequently occurring, less informative words while emphasizing rarer, sentiment-bearing tokens. Similar to BoW, TF-IDF produces sparse vector representations and was augmented with language tags to incorporate language-specific context. We generated TF-IDF features using the TfidfVectorizer implementation from scikit-learn.

3.4.3 Word2Vec

To capture semantic and syntactic relationships beyond mere word occurrence, we trained Word2Vec embeddings. Word2Vec encodes words as dense numerical vectors by analyzing their co-occurrence with neighboring words in sentences. We used the 100 dimension vector with Skip-Gram variant, which models the relationship between a central word and its surrounding context, resulting in robust word associations.

3.4.4 FastText

FastText extends Word2Vec by representing words as collections of character n-grams. This subword modeling allows FastText to generate reliable embeddings for rare, misspelled, or phonetically transliterated words—common phenomena in informal code-mixed social media text. By incorporating sub word information, FastText better handles out-of-vocabulary tokens, making it well-suited for our dataset. We used Fasttext vector of size 100 with window of size 5 and minimum count of 2.

3.4.5 Integration with Language Tags

A key aspect of our feature engineering was the integration of language identification information. For both sparse (BoW and TF-IDF) and dense (Word2Vec and FastText) representations, we concatenated each word's vector with a one-hot encoded vector representing its predicted language tag (Gujarati, English, or Other). This augmented feature vector enables the model to distinguish between terms with similar spellings but different linguistic meanings—for example, the English word "bad" versus the Gujarati phonetic spelling "bad".

For sparse representations, sentence vectors were generated by aggregating word-level vectors followed by appending the language tag encodings. For dense embeddings, sentence-level vectors were obtained by averaging word embeddings and concatenating the corresponding language tag vectors.

3.4.6 Word-to-Sentence Aggregation

To align token-level features with sentence-level classification tasks, we transformed all word-level features into fixed-size numerical formats. This process involved padding

sequences to a uniform length (95th percentile) and applying one-hot encoding for word level language tags. The aggregated features were organized into sentence-level feature matrices compatible with machine learning classifiers.

3.4.7 Integration Strategies

To systematically evaluate the contribution of different feature sets and the influence of language tags, we experimented with multiple input configurations:

- BoW only
- TF-IDF only
- Word2Vec only
- FastText only
- BoW combined with Word-Level Language Tags
- TF-IDF combined with Word-Level Language Tags
- Word2Vec combined with Word-Level Language Tags
- FastText combined with Word-Level Language Tags

This thorough comparative analysis allowed us to assess the relative strengths of sparse versus dense features and quantify the performance gains from incorporating explicit language information into the sentiment classification model.

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. Each classifier was trained and evaluated on multiple feature representations and fine-tuned through exhaustive hyper-parameter optimization using grid search. This approach enabled the models to effectively handle the high dimensionality, sparsity, and linguistic complexity characteristic of code-mixed social media data.

3.5.1 LR is a linear classifier that models class probabilities by applying a logistic (sigmoid) function to a weighted combination of input features. We adapted LR for multi-class classification using a one-vs-rest scheme. Despite its conceptual simplicity, LR performs robustly on high-dimensional data like text, especially when combined with regularization techniques such as the L2 penalty to prevent overfitting. We applied LR across all feature sets—including sparse representations (BoW, TF-IDF), dense embeddings, and language-tag augmented features—owing to its interpretability and computational efficiency, making it a strong baseline for sentiment classification.

3.5.2 **MNB** is a probabilistic classifier that assumes conditional independence of features given the class label. It models the distribution of discrete word counts within each class, which makes it particularly effective for text represented as frequency vectors. Due to its simplicity and fast training times, MNB serves as a robust baseline in multi-class text classification. We primarily applied MNB to discrete feature spaces such as BoW and TF-IDF. For continuous embeddings (Word2Vec, FastText), we applied Min-Max scaling to adapt the inputs to the model's expected range. Despite its strong independence assumptions, MNB often performs competitively in high-dimensional, sparse data scenarios.

3.5.3 **SVM** is a powerful discriminative classifier that seeks the hyperplane maximizing the margin between classes in the feature space. We employed a linear kernel SVM for its scalability and effectiveness in handling sparse, high-dimensional textual features. Although SVMs do not natively provide probability estimates, they generally exhibit strong generalization and resist overfitting when appropriately regularized. The model was trained using a one-vs-one multi-class strategy and evaluated across all feature variants, including dense embeddings combined with language-level features.

3.5.4 **KNN** is a non-parametric, instance-based algorithm that assigns class labels based on the majority class among the closest neighbours in feature space. While simple and intuitive, KNN's performance can degrade in high-dimensional settings due to the "curse of dimensionality." To alleviate this, we applied dimensionality reduction prior to KNN training, improving both computational efficiency and accuracy. We explored Euclidean distance and cosine similarity as proximity metrics, common in text similarity tasks. KNN was evaluated on reduced representations of all feature types to study its behaviour on complex data.

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, Hyper-parameter Optimization and Performance Evaluation

To ensure a fair and reliable comparison across all classifiers, we adopted a consistent experimental framework. We split the dataset into training and test subsets, allocating 80% for training and 20% for testing. Stratified sampling preserved the original class distribution, ensuring unbiased representation of all sentiment categories.

To achieve optimal performance for each classifier, we systematically performed hyperparameter tuning using GridSearchCV on each classifier. This exhaustive search evaluated multiple combinations of parameter values, allowing the selection of the configuration that maximized predictive accuracy and generalization for each model.

Table 1 summarizes the optimal hyper-parameters identified for all classifiers in our experiments, providing a standardized setup for fair comparison across different feature representations.

Table 1 Hyperparameters

Algorithm	Hyperparameters Ranges Tuned	Parameters Applied
Logistic Regression (LR)	C: [0.01, 0.1, 1, 10, 100], penalty: ['l1', 'l2'], max_iter: [100, 200, 500, 1000], solver: ['lbfgs', 'saga']	C=10, penalty='l2', max_iter=1000, solver='saga'
Multinomial Naive Bayes (MNB)	alpha: [0.1, 0.5, 1.0, 2.0]	alpha=0.1
Support Vector Machine (SVM)	C:[0.1,1,10], kernel:['linear','rbf']	C=1, kernel='linear'
K-Nearest Neighbors (KNN)	n_neighbors: [3, 5, 7, 9], weights: ['uniform', 'distance']	n_neighbors=3, weights='distance'
Decision Tree (DT)	max_depth: [5, 10, 20, None], criterion: ['gini', 'entropy','log_loss']	max_depth=20, criterion='gini'
Random Forest (RF)	n_estimators:[100, 200,300,400], max_depth: [10, 20, None]	n_estimators=400, max_depth=10
XGBoost (XGB)	max_depth: [3, 6, 10], learning_rate: [0.01, 0.1, 0.3], n_estimators:[100, 200, 300,500]	max_depth=10, learning_rate=0.1, n_estimators=300

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.
    - **Precision:** Assesses the correctness of positive predictions by comparing the number of correctly predicted instances to the total predicted instances for a class.
    - **Recall:** Evaluates coverage by measuring the proportion of correctly identified instances relative to the total number of true instances in a class.
    - **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.

4. KEY FINDINGS & DISCUSSION

This section presents and discusses the experimental results obtained from evaluating seven machine learning classifiers across multiple feature representations, with and without word-level language tag augmentation and SMOTE-based oversampling.

We evaluated all classifiers on first without SMOTE and after that SMOTE-balanced dataset to mitigate the effects of class imbalance. We did this for both original dataset and then with language word level feature augmentation. Figure 4 and Tables 2, 3 summarize the performance in terms of accuracy, precision, recall and macro-averaged F1 score, respectively, across different feature representations.

This comprehensive evaluation allows us to compare sparse versus dense representations, assess the impact of language tag integration, and identify the models that best capture sentiment patterns in code-mixed text.

- Across all models, TF-IDF and BOW representations consistently perform better than embedding based methods while TF-IDF being best across all the models. This trend is particularly noticeable in LR, SVM, and XGB, where TF-IDF achieves the highest accuracy and F1-scores.
- Adding language features consistently improved performance across almost all models. Models with language features achieved higher F1-scores compared to those without them.

  This indicates that language-related information provides useful context that helps the model better understand the text.
- As shown in Table 4 that without applying SMOTE, the best performance was achieved by LR using TF-IDF or BoW with language features, with a macro F1-score of 0.70. This shows that LR works very well on the original imbalanced dataset, especially when strong text features like TF-IDF are used.
- As shown in Table 5 that while applying SMOTE the best model was SVM with TF-IDF and language features, achieving a macro F1-score of about 0.70. SMOTE helped improve the balance between classes, which is why SVM performed slightly better in this setting, especially in terms of recall.

Table 2: Performance of Dataset without Oversampling

		Without SMOTE							
		Without Language				With Language			
Model	**Feature Type**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
LR	**BOW**	0.79	0.72	0.67	0.69	0.79	0.72	0.68	0.7
	TF-IDF	0.78	0.73	0.64	0.67	0.79	0.75	0.66	0.69
	Word2Vec	0.74	0.66	0.54	0.55	0.74	0.65	0.57	0.58
	FastText	0.73	0.76	0.53	0.54	0.74	0.64	0.55	0.56
MNB	**BOW**	0.74	0.64	0.66	0.65	0.75	0.65	0.67	0.66
	TF-IDF	0.77	0.71	0.6	0.62	0.72	0.62	0.6	0.6
	Word2Vec	0.56	0.18	0.2	0.15	0.53	0.38	0.41	0.39
	FastText	0.62	0.22	0.28	0.25	0.54	0.39	0.43	0.4
SVM	**BOW**	0.78	0.71	0.67	0.68	0.78	0.7	0.67	0.68
	TF-IDF	0.79	0.79	0.62	0.64	0.8	0.76	0.64	0.66
	Word2Vec	0.74	0.58	0.54	0.55	0.74	0.68	0.56	0.57
	FastText	0.73	0.56	0.53	0.54	0.74	0.67	0.54	0.55
DT	**BOW**	0.66	0.61	0.46	0.49	0.65	0.57	0.46	0.49
	TF-IDF	0.66	0.61	0.46	0.49	0.65	0.57	0.46	0.49
	Word2Vec	0.62	0.49	0.5	0.49	0.63	0.5	0.51	0.5
	FastText	0.61	0.48	0.49	0.49	0.63	0.5	0.51	0.5
RF	**BOW**	0.6	0.57	0.66	0.58	0.61	0.53	0.61	0.55
	TF-IDF	0.71	0.62	0.59	0.59	0.72	0.65	0.58	0.59
	Word2Vec	0.74	0.71	0.55	0.56	0.74	0.71	0.55	0.56
	FastText	0.72	0.62	0.57	0.58	0.73	0.64	0.56	0.57
KNN	**BOW**	0.74	0.68	0.57	0.59	0.71	0.62	0.55	0.57
	TF-IDF	0.59	0.86	0.31	0.3	0.63	0.52	0.42	0.43
	Word2Vec	0.72	0.58	0.55	0.55	0.7	0.58	0.53	0.54
	FastText	0.71	0.58	0.54	0.55	0.71	0.62	0.54	0.55
XGB	**BOW**	0.77	0.74	0.61	0.64	0.78	0.75	0.62	0.66
	TF-IDF	0.78	0.74	0.63	0.66	0.79	0.75	0.63	0.66
	Word2Vec	0.74	0.66	0.56	0.58	0.75	0.7	0.57	0.59
	FastText	0.74	0.69	0.55	0.57	0.74	0.7	0.57	0.59

Table 3: Performance of Dataset with Oversampling (SMOTE)

		With SMOTE							
		Without Language				With Language			
Model	**Feature Type**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
LR	**BOW**	0.67	0.59	0.62	0.6	0.75	0.68	0.7	0.69
	TF-IDF	0.74	0.67	0.67	0.67	0.76	0.69	0.71	0.7
	Word2Vec	0.59	0.55	0.63	0.56	0.61	0.55	0.66	0.57
	FastText	0.56	0.53	0.61	0.53	0.6	0.54	0.64	0.55
MNB	**BOW**	0.64	0.64	0.65	0.62	0.72	0.65	0.67	0.65
	TF-IDF	0.67	0.63	0.66	0.63	0.69	0.61	0.63	0.62
	Word2Vec	0.46	0.4	0.46	0.39	0.48	0.39	0.46	0.38
	FastText	0.5	0.4	0.47	0.41	0.49	0.39	0.46	0.38
SVM	**BOW**	0.66	0.57	0.61	0.58	0.74	0.66	0.69	0.67
	TF-IDF	0.74	0.67	0.7	0.68	0.76	0.69	0.72	0.7
	Word2Vec	0.56	0.55	0.63	0.54	0.61	0.55	0.66	0.57
	FastText	0.54	0.54	0.61	0.53	0.59	0.54	0.65	0.55
DT	**BOW**	0.42	0.52	0.49	0.44	0.61	0.53	0.48	0.5
	TF-IDF	0.42	0.52	0.49	0.44	0.61	0.53	0.48	0.5
	Word2Vec	0.59	0.48	0.53	0.5	0.61	0.5	0.55	0.52
	FastText	0.58	0.47	0.53	0.49	0.6	0.49	0.54	0.51
RF	**BOW**	0.63	0.51	0.58	0.53	0.66	0.57	0.61	0.59
	TF-IDF	0.7	0.61	0.61	0.61	0.7	0.62	0.58	0.6
	Word2Vec	0.71	0.61	0.61	0.61	0.71	0.62	0.63	0.62
	FastText	0.67	0.59	0.61	0.59	0.69	0.6	0.63	0.61
KNN	**BOW**	0.45	0.48	0.57	0.46	0.55	0.52	0.63	0.52
	TF-IDF	0.6	0.7	0.32	0.32	0.63	0.52	0.42	0.43
	Word2Vec	0.56	0.48	0.56	0.5	0.57	0.48	0.56	0.5
	FastText	0.54	0.46	0.54	0.47	0.56	0.48	0.56	0.49
XGB	**BOW**	0.76	0.72	0.6	0.63	0.77	0.73	0.61	0.65
	TF-IDF	0.75	0.69	0.64	0.66	0.76	0.71	0.63	0.66
	Word2Vec	0.71	0.61	0.61	0.61	0.75	0.67	0.61	0.63
	FastText	0.7	0.6	0.6	0.6	0.74	0.65	0.6	0.61

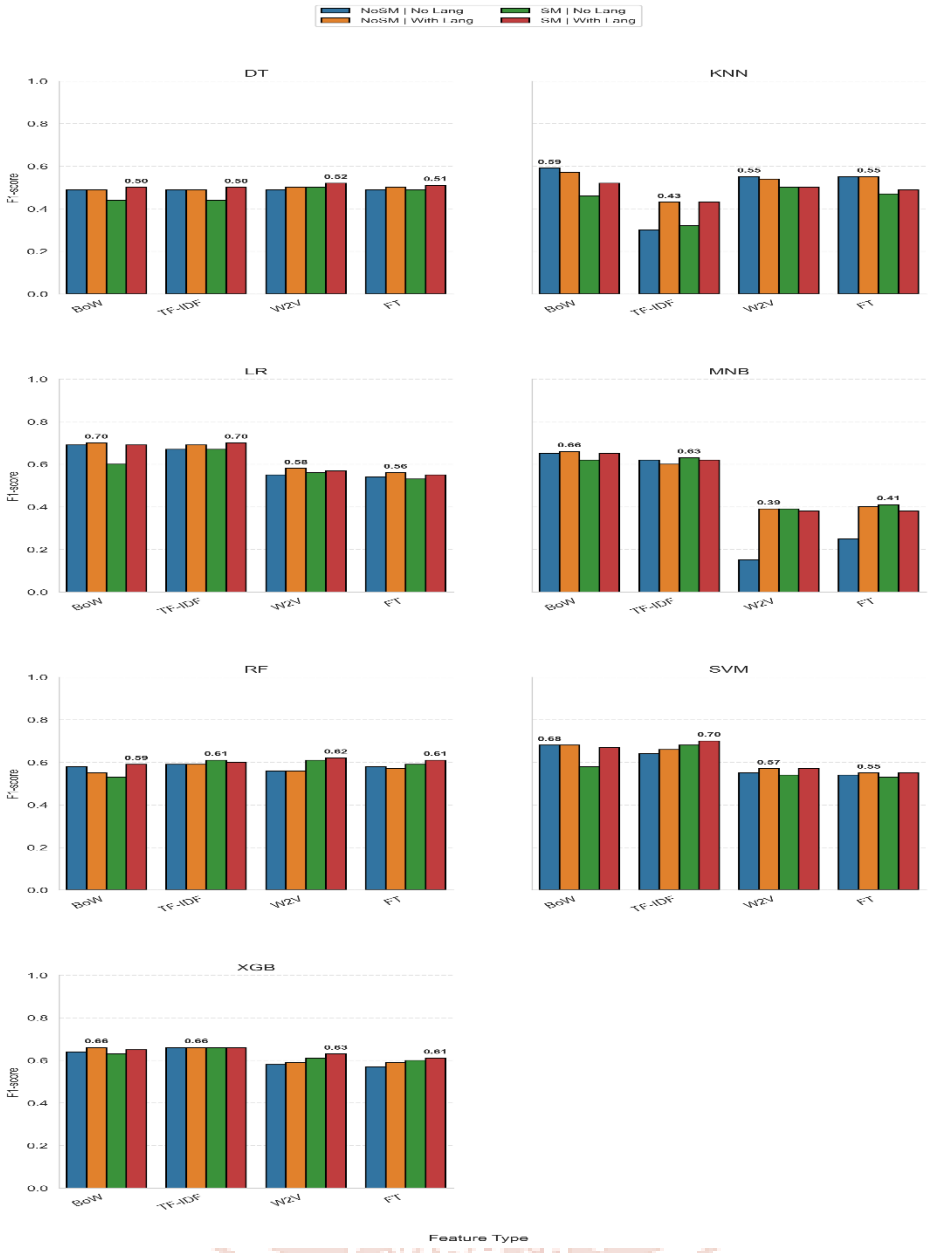


Fig. 4 Performance of various ML algorithms with different feature types

LR emerges as the strongest classifier on the original imbalanced dataset. With BoW and language features, LR achieves an accuracy of 0.79 and a macro F1-score of 0.70, the highest observed without SMOTE. TF-IDF also performs competitively with LR, yielding a slightly lower macro F1 (0.69) but higher precision (0.75). These results confirm that LR is highly effective for high-dimensional sparse text features, even under class imbalance.

SVM delivers the best overall results when SMOTE is applied. With TF-IDF and language features, SVM achieves an accuracy of 0.76 and a macro F1-score of 0.70, benefiting from the improved class balance introduced by oversampling. SVM's ability to maximize the decision

margin in high-dimensional feature spaces makes it particularly well-suited to sparse text representations.

XGB exhibits strong and consistent performance across all feature types, closely trailing LR and SVM. It performs best with TF-IDF and BoW representations and benefits moderately from language tag augmentation, making it a reliable alternative when ensemble-based approaches are preferred.

MNB performs competitively with sparse features (BoW and TF-IDF) but degrades substantially with dense embeddings. This is expected, as MNB assumes discrete feature distributions and is not designed for continuous vector representations. Language tag augmentation provides modest improvements for MNB in the SMOTE setting.

RF provides robust performance across feature types but consistently yields macro F1-scores below those of LR and SVM. While its ensemble nature reduces overfitting, the model does not fully leverage the discriminative power of sparse features in this setting.

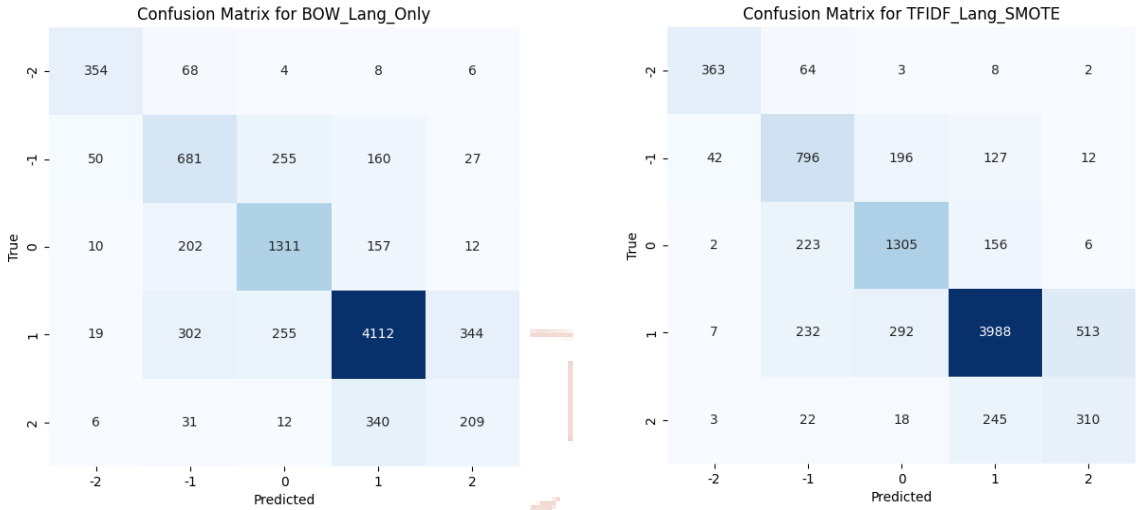
KNN produces moderate and inconsistent results. It performs relatively better on BoW features without SMOTE but struggles considerably with TF-IDF, likely due to the curse of dimensionality in high-dimensional sparse spaces. Language tag augmentation provides only marginal gains for KNN.

DT is the weakest classifier in this study. Even with language features and SMOTE, improvements remain minimal, suggesting that single-tree models lack the capacity to handle the high dimensionality and complexity of code-mixed text features.

The consistent dominance of TF-IDF can be attributed to the characteristics of Gujarati–English code-mixed text, where sentiment is often conveyed through a small set of high-signal lexical tokens such as "jordar", "mast", "bakwas", or "super". TF-IDF assigns higher weights to such discriminative terms while suppressing the influence of frequently occurring but less informative words — an advantage that simple frequency-based BoW counting does not provide. Furthermore, TF-IDF partially mitigates the effects of lexical variation and informal spelling, common in social media comments. Users frequently write Gujarati words in Roman script with inconsistent transliterations (e.g., "majama", "saras"), creating sparse lexical distributions; TF-IDF's weighting mechanism highlights these distinctive tokens more effectively than raw frequency counts.

Dense embeddings such as Word2Vec and FastText capture semantic relationships between words but underperform relative to sparse methods in this dataset. This is likely because sentiment cues in code-mixed comments tend to rely on explicit lexical signals rather

than distributional semantics. Additionally, the relatively small corpus size and high lexical variability may limit the quality of trained embeddings.



LR Without Applying SMOTE & With Language (BoW)

SVM With SMOTE & With Language (TF-IDF)

Fig. 5. Confusion matrix of LR & SVM

To gain deeper insight into model behaviour, we performed error analysis using the confusion matrices of the two best-performing configurations: LR with BoW and language features on the original dataset, and SVM with TF-IDF and language features on the SMOTE-balanced dataset, as shown in Figure 5.

Adjacent Class Confusion. Both models exhibit a consistent pattern of misclassification between neighbouring sentiment classes. Negative (−1) instances are frequently misclassified as Neutral (0) or Positive (+1), and Neutral instances are sometimes predicted as Positive. This is likely because many comments in the dataset express mild appreciation or general reactions without strong sentiment indicators. Short expressions such as "nice video", "good", or mixed Gujarati–English phrases may carry subtle positivity that the model interprets as Positive sentiment even when the intended label is Neutral. Similarly, some Negative comments are classified as Neutral, possibly due to the indirect manner in which dissatisfaction is expressed in social media text — through sarcasm, understatement, or contextual references that do not contain explicit negative terms. Since BoW relies solely on word frequency rather than contextual understanding, such nuanced cases are difficult to distinguish. TF-IDF partially alleviates this by down-weighting high-frequency uninformative tokens, which may explain why SVM with TF-IDF achieves better recall on minority classes after SMOTE.

Extreme Class Confusion. The extreme sentiment categories — extremely negative (−2) and extremely positive (+2) — are the most challenging for both classifiers. These classes

contain fewer training samples and exhibit lower correct prediction rates. A large proportion of Extremely Positive comments are misclassified as Positive (+1), and in a smaller number of cases, Extremely Negative instances are misclassified as Negative (−1). This pattern suggests that the boundary between adjacent intensity levels is not always captured effectively, particularly when intensifiers such as "ekdum" or "bov" are present in both extreme and non-extreme classes. Although SMOTE generates synthetic samples for minority classes and partially improves recall for the SVM model, the confusion between extreme and adjacent classes persists, indicating that class imbalance alone does not fully account for this misclassification — feature overlap between these classes is a contributing factor.

Transliteration-Induced Sparsity. An additional source of error arises from the non-standard transliteration of Gujarati words written in Roman script. The same word can appear in multiple orthographic forms — for example, "mast", "maast", and "masta" — which are treated as distinct tokens during feature extraction. This orthographic variation increases sparsity in the feature space and reduces the model's ability to identify consistent sentiment patterns across lexically equivalent but differently spelled tokens. This limitation affects both sparse and dense feature representations, though it is more pronounced in BoW and TF-IDF, where each unique spelling occupies a separate dimension in the feature vector.

5. CONCLUSION

We presented a comprehensive machine learning framework for five-class sentiment classification of Gujarati–English code-mixed text, addressing a significantly under-explored problem in multilingual NLP. We curated a large-scale, manually annotated dataset of 44,672 YouTube comments spanning diverse content domains, covering both Romanized and native Gujarati scripts. To the best of our knowledge, this is among the first datasets to support fine-grained sentiment analysis for this language pair at this scale.

A central contribution of this work is the integration of word-level language identification — performed using a fine-tuned DistilBERT model achieving 97.09% accuracy — as an augmentation strategy for traditional text feature representations. We systematically evaluated four feature extraction techniques (BoW, TF-IDF, Word2Vec, and FastText) across seven machine learning classifiers (LR, MNB, SVM, KNN, DT, RF, and XGBoost), with and without language tag augmentation and SMOTE-based class balancing.

The experimental results demonstrate that sparse representations, particularly TF-IDF, consistently outperform dense embeddings in this setting. This is attributable to the lexically driven nature of sentiment expression in code-mixed social media text, where high-signal tokens such as "jordar", "mast", and "bakwas" serve as primary sentiment indicators. Dense

embeddings, while capable of capturing semantic relationships, are limited by the high orthographic variability and small corpus size characteristic of low-resource code-mixed data.

Word-level language tag augmentation consistently improved macro F1-scores across nearly all model–feature combinations, confirming that explicit linguistic context is a valuable signal for sentiment classification in multilingual settings. Without SMOTE, LR with BoW and language features achieved the best performance, with an accuracy of 0.79 and a macro F1-score of 0.70. With SMOTE applied, SVM with TF-IDF and language features achieved the same macro F1-score of 0.70 with improved recall across minority sentiment classes.

Error analysis revealed consistent misclassification patterns between adjacent sentiment classes, particularly at the boundaries between Neutral and Positive, and between extreme and non-extreme categories. These errors are partly attributable to the indirect expression of sentiment in informal text, orthographic variability in Romanized Gujarati, and residual class imbalance in the extreme categories even after oversampling.

Overall, this work contributes both a novel methodological framework and a reusable linguistic resource for sentiment analysis of code-mixed, low-resource language pairs. The proposed approach demonstrates that incorporating explicit language awareness into machine learning pipelines meaningfully improves classification performance and offers a scalable strategy for building more inclusive NLP systems for linguistically diverse communities.

6. FUTURE WORK

Building on the promising outcomes of the current machine learning framework, we would like to extend and enhance the system, particularly for real-time applications and advanced modeling:

1. **Deep Learning Integration**: The next step involves transitioning from traditional classifiers to deep learning architectures. We would initially experiment with Single-Layer and Multi-Layer Perceptron (SLP, MLP) models, then by sequential models such as RNN, LSTM, and Bi-LSTM. They would be evaluated using both standalone contextual embeddings and combining with handcrafted linguistic features. Our goal is to assess their ability to capture extended contextual dependencies and the complex structural patterns characteristic of code-mixed text.
2. **Transformer-Based Fine-Tuning**: Our future research will investigate transformer-based models, particularly fine-tuning pre-trained language models such as BERT and multilingual BERT (mBERT). These models are well-suited to handle the rich contextual and linguistic diversity of Gujarati–English code-mixed data, potentially improving

sentiment classification by leveraging deep contextual understanding and prior exposure to Indian languages.

3. **Dataset Expansion and Cross-Domain Generalization.** The current dataset is sourced exclusively from YouTube comments. Future work will explore expanding the corpus to include data from platforms such as Twitter/X, WhatsApp, and Facebook, which exhibit different linguistic registers and mixing patterns. Cross-domain evaluation will be conducted to assess the generalizability of the proposed framework across varied social media contexts and writing styles.
4. **Deployment on Edge Devices**: Another direction is the deployment of the sentiment classification framework on resource-constrained platforms (e.g., Raspberry Pi). This will enable real-time sentiment analysis for practical applications, including local content moderation, mobile-based feedback systems, vernacular social media monitoring, and intelligent conversational agents. Edge deployment will also enhance the scalability and accessibility of systems processing informal, multilingual text in real-world environments.

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.