ISSN 2007-9737

Sentiment Analysis of Code-Mixed Indian Languages Using Context-Engineered Prompts

Sauvik Bal[1,2,*], Shillpi Mishrra[1], Lopa Mandal[3]

[1] Techno India University, Department of Computer Science and Engineering, India

[2] Maulana Abul Kalam Azad University of Technology, India

[3] Alliance University, Department of Computer Science and Engineering, India

{sauvikbal, shllpimishrra91, drmandal.lopa}@gmail.com

Abstract. This study investigates the application of Large Language Models (LLMs) on sentiment analysis tasks on code-mixed Indian languages, wherein English and Indian languages are combined. This study also points out challenges faced by this technical community, such as uncertainty associated with transliteration and cultural nuances, and holds that classical models require substantial fine-tuning to perform effectively. This study proposes the application of context-engineered prompts, wherein linguistic cues and appropriate sentiment judgments from a cultural standpoint, could provide improvements on sentiment type categorization, as opposed to proposals on developing another model, and demonstrates more consistent behavior than simpler prompting strategies when handling mixed polarity and culturally implicit sentiment. This requires no extra training data and thus pertains to efficiency as well as valuable observations on improving LLM usability on diverse linguistic settings.

Keywords. Code-mixed languages, context engineering, large language models, low-resource languages, prompt engineering, sentiment analysis.

1 Introduction

The fast changes happening with LLMs have greatly influenced NLP activities, especially sentiment analysis and machine translation. Low-resource and non-standard language varieties, such as code-mixed languages coming from multilingual scenarios like India, still remain challenging for LLMs. Code-mixing, or the switching between English and one's vernacular, complicates sentiment analysis due to the irregular transliteration and mixed structures present.

Since traditional approaches depend upon regular rules of the language, they cannot infer emotions from such writings. Distributed emotion cues and culturally unique expressions that run counter to standard evaluations are major obstacles.

While there are emerging efforts at using LLMs for low-resource languages, methods more advanced than simple prompting are often required for any successful sentiment analysis in code-mixed settings. This result brings to the fore the relevance of context engineering as a preprocess that alters model performance rather than parameters by injecting linguistic, cultural, and relevant task-based meaning to improve prompts.

This implies that prompts define a space of design that not only requires linguistic meaning, like sentiment explanations, to be infused. This is very relevant to code-mixed Indian languages [4, 10]. This research makes use of context-engineered prompts to be used along with LLMs to carry out sentiment analysis tasks on code-mixed Hindi-English as well as Bengali-English datasets. This is done by

comparing various prompt designs to machine learning as well as transformers.

The topic of this paper is related to the ability of LLMs to handle non-standard or low-resource linguistic events and, specifically, sentiment analysis tasks for code-mixed Indian languages [2, 18].

Through quantitative analysis and qualitative studies, the paper has pointed out a number of failure cases, including mixed polarity with sentiment analysis tasks and the misuse of the notion of neutrality. The three major contributions of this paper will be summarized as follows: i) an empirical study related to sentiment analysis of code-mixed languages by the application of LLMs, ii) the discussion and analysis of context-designed tasks tailored to the specific characteristics of the text, and iii) the analysis of the improvement of performances and the challenges, with a strong focus on the role of reproducibility and applicability. The necessity of more accessible and applicable sentiment analysis for multilingual and low-resource environments through the power of context engineering instead of retraining the models is supported by the paper.

2 Related Work

Sentiment analysis, the task of automatically understanding views and feelings expressed in text, has remained an extremely important area of natural language processing for over 20 years [19]. Initially, approaches to this task used supervised machine learning techniques, which assumed perfectly well-formed, monolingual inputs [20]. However, in realistic multilingual interaction, especially on social networking sites, these assumptions are often violated. Reasons for this are noted to be the weaknesses of these traditional approaches, especially when the use of informal language tends to rise, especially in multilingual environments where languages are code-mixed.

This challenge tends to be exacerbated, especially for low-resource, multilingually mixed languages lacking an overarching standardization to their alphabet, as well as culturally distinctive expressions of sentiment.

2.1 Sentiment Analysis for Indian and Low-Resource Languages

The initial sentiment analysis for Indian languages, which used supervised machine learning algorithms like SVMs and Naïve Bayes, primarily focused on monolingual text data in Indian scripts [7]. Because they depended on small annotated corpora and underutilized language-specific resources, many algorithms proved ineffectual even on real-world data, even though they worked well on annotated data [19].

Manual feature design is normally ineffective for everyday text data and cannot handle context and irony. While the use of deep learning and transformer models, like multilingual BERT has upgraded the quality of sentiment analysis because of improvements in understanding context, the discrepancy between the pretraining tasks and the noisy nature of user-generated data, as well as the need for annotated data for fine-tuning, remain as challenges [8].

2.2 Code-Mixed Sentiment Analysis

A new area of research called “code-mixed sentiment analysis” has emerged that deals with the complexities of code-mixed texts that involve several languages [23]. This is more significant in India, where English is often code-mixed with native Indian languages like Bengali and Hindi, which write in the Roman alphabet [18]. Some of the significant problems that have been identified in this area of research include the irregularities in transliteration, borrowing of words, grammatical errors, and shift in the polarity of the sentiment due to discourse markers such as “but,” “lekin,” or “kintu.” The presence of overall sentiment in code-mixed sentences that involve several languages is also a factor that makes the determination of overall sentiment through simple analyses more challenging [14].

The most common approaches in the area of code-mixed sentiment analysis have been supervised learning-based approaches, which lack scalability because high-quality annotations that can be useful for accurate supervised learning for sentiment determination involve significant effort.

This is also further aggravated because previous analyses in this area have focused on the quantity of performance rather than delving into error analyses that lack insights into common error patterns, such as improperly classifying neutral sentiments or expressing sentiments about significant culture behaviors [5].

2.3 Large Language Models and Prompt-Based Sentiment Analysis

Natural language processing and sentiment analysis have received a boost due to the emergence of LLMs [6]. The capability of the model has been proved for zero-shot as well as a few-shot classification of the sentiment without specific training [5]. This is quite helpful for a low-resource language. However, most of the work has been done using the English language. This creates a concern regarding the effectiveness of the model in a multilingual as well as code-mixed setting.

Although naive prodding experiments have shown many accomplishments in the classification of the sentiment of the model within a high-resource language. However, often the accuracy of the model goes lower when testing Indian languages because the model might not grasp the specific attitudes of the culture and the use of the transliterated form of the phrases. Moreover, this study would consider the fixed form of the prompts without significant research regarding the specific impact of different parts of the context [12].

2.4 Context Engineering in Prompt Design

In contrast to fine-tuning, context engineering helps to improve the performance of models in the context of the prompt by including relevant information in the prompt during testing [21]. In the case of sentiment analysis in low resource language applications such as the code-mixed Indian languages, the procedure achieves the desired results [10]. Context plays an important role in the case of the Indian languages because phrases such as ”thik-thak,” ”chal jata hai,” and ”time pass ke liye” can indicate a sentiment of being in a state of no emotions or faintly negative emotions, which are often missed by the models in the concerned research area and hence are considered in the proposed research to address the issue of sentiment analysis in the case of code-mixed Indian languages [11].

2.5 Comparative Analysis of Existing Approaches

Table 1 summarizes key differences between representative strands of prior work and the proposed study.

2.6 Positioning of the Present Work

Unlike prior sentiment analysis studies that focus on model architecture or dataset creation, this work adopts an empirical and analysis-driven perspective [3]. The proposed approach does not require fine-tuning or additional annotation effort. Instead, it systematically examines how different forms of contextual information embedded within prompts influence LLM performance on code-mixed Indian language sentiment analysis [9]. This research work employs both quantitative analysis and qualitative error analysis to explore the link between inference-driven promotion and linguistic realism. In low-resource and code-mixed settings, it also explores context engineering as a possible alternative to data-intensive approaches in sentiment analysis [15].

3 Methodology

A methodological approach for conducting sentiment analysis on code-mixed languages in India using artificial prompts is discussed in this section. It strongly focuses on an empirical and replicable approach that uses inference by artificial prompts and publicly available data, rendering the need for fine-tuning unnecessary. Sentiment classification, rapid construction, prediction selection, and assessment are all formally defined in mathematics by this approach.

Table 1. Comparison of sentiment analysis approaches for code-mixed Indian languages

Approach Category	Methodological Characteristics	Limitations in Code-Mixed Indian Languages
Few-shot LLM prompting [5]	Provides limited labeled examples within prompts	Performance depends on example selection, increased token cost, unstable behavior under prompt variation
Traditional deep learning [6]	CNN or RNN architectures trained on labeled sentiment datasets	Requires large annotated corpora, struggles with long-range context, limited interpretability for sentiment cues
Rule-based systems [7]	Manually crafted linguistic rules and heuristics for sentiment detection	Poor scalability, brittle to informal expressions, ineffective for noisy and evolving social media language
Zero-shot LLM prompting [8]	Uses generic natural language instructions without task-specific context	Highly prompt-sensitive, inconsistent outputs, misclassification of neutral and mixed-polarity expressions
Prompt-based sentiment analysis (generic) [10]	Applies prompting strategies without linguistic or cultural grounding	Lacks awareness of code-mixing patterns, discourse markers, and culturally implicit sentiment
Context-aware NLP approaches [12]	Incorporates limited contextual cues such as language tags or sentiment hints	Often ad hoc, lacks systematic evaluation, limited focus on Indian code-mixed settings
Sequence models for sentiment [13]	LSTM or GRU-based architectures capturing sequential dependencies	Sensitive to spelling inconsistencies, limited effectiveness under transliterated and mixed-language input
Classical machine learning [19]	Bag-of-words or TF–IDF features combined with SVM or Logistic Regression classifiers	Heavy dependence on feature engineering, weak handling of code-switch boundaries, limited robustness across domains
Transformer-based multilingual models [20]	Fine-tuned pretrained models such as mBERT or IndicBERT	High computational cost, data-intensive fine-tuning, insufficient modeling of cultural sentiment expressions
Lexicon-based sentiment analysis [21]	Uses predefined sentiment lexicons and polarity scoring rules, often designed for a single language	Fails under transliteration variability, cannot resolve mixed sentiment polarity, ignores discourse structure and cultural pragmatics
Domain-adapted transformers [22]	Transformers fine-tuned on domain-specific sentiment datasets	Reduced generalization to unseen domains, limited portability across languages

3.1 Problem Definition

Let $X = \{x_1, x_2, \ldots, x_N\}$ denote a dataset of N code-mixed text instances, where each instance x_i is composed of tokens from an Indian language and English, written in Roman script. Each instance is annotated with a sentiment label y_i drawn from the label set:

$$\mathcal{Y} = \{\text{Positive}, \text{Negative}, \text{Neutral}\}. \tag{1}$$

The sentiment analysis task is defined as learning a mapping:

$$f : X \to \mathcal{Y}, \tag{2}$$

such that, for each input sentence x_i, the function $f(\cdot)$ assigns a sentiment label y_i.

3.2 Preprocessing

To preserve the linguistic characteristics of code-mixed text, only minimal preprocessing is applied. Each input instance x_i is transformed using a preprocessing function $g(\cdot)$:

$$\tilde{x}_i = g(x_i), \tag{3}$$

To preserve sentiment-bearing cues typically present in real-code mix communication, $g(\cdot)$ efficiently removes URLs, user mentions, and redundant whitespace, while preserving transliteration words, emoticons, and informal phrases.

3.3 Baseline Models

In order to put the effectiveness of Prompt-Based Inference in context, a variety of baseline techniques of sentiment analysis have been employed, particularly those techniques based on traditional machine learning using the traditional TF-IDF feature set:

$$\mathbf{v}_i = \text{TF–IDF}(\tilde{x}_i), \tag{4}$$

followed by Logistic Regression and Support Vector Machine classifiers. In addition, transformer-based multilingual models are fine-tuned on the same training splits and evaluated under identical conditions. These baselines provide reference points for comparative evaluation and are not the primary focus of the study.

3.4 Prompt-Based Sentiment Inference

Sentiment prediction using large language models is performed through prompt-based inference. A base prompt template $P(\cdot)$ is defined to specify the sentiment classification task. Each preprocessed input $\tilde{x}_i$ is embedded into the prompt as:

$$p_i = P(\tilde{x}_i), \tag{5}$$

where p_i represents the complete textual prompt provided to the language model. Given the prompt p_i, the model produces an output sequence o_i, from which a sentiment label is extracted:

$$\hat{y}_i = \arg\max_{y \in \mathcal{Y}} \Pr(y \mid p_i). \tag{6}$$

No model parameters are updated during inference, and each instance is processed independently using a single prompt invocation.

3.5 Context Engineering

Context engineering is incorporated by augmenting the base prompt with additional task-relevant information. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$ denote a set of predefined context types, such as linguistic, discourse-level, or culturally grounded context. For a given context C_k, the context-engineered prompt is defined as:

$$p_i^{(k)} = C_k \oplus P(\tilde{x}_i), \tag{7}$$

where $\oplus$ denotes concatenation. For each input instance x_i, the language model produces a set of context-specific predictions $\{\hat{y}_i^{(1)}, \hat{y}_i^{(2)}, \ldots, \hat{y}_i^{(K)}\}$. A final sentiment label $\hat{y}_i$ is selected using a fixed, non-adaptive selection rule:

$$\hat{y}_i = \mathcal{S}\left(\{\hat{y}_i^{(k)}\}_{k=1}^{K}\right), \tag{8}$$

where $\mathcal{S}(\cdot)$ denotes a predefined aggregation strategy (e.g., majority or priority-based selection). No learning or optimization is performed during this selection step.

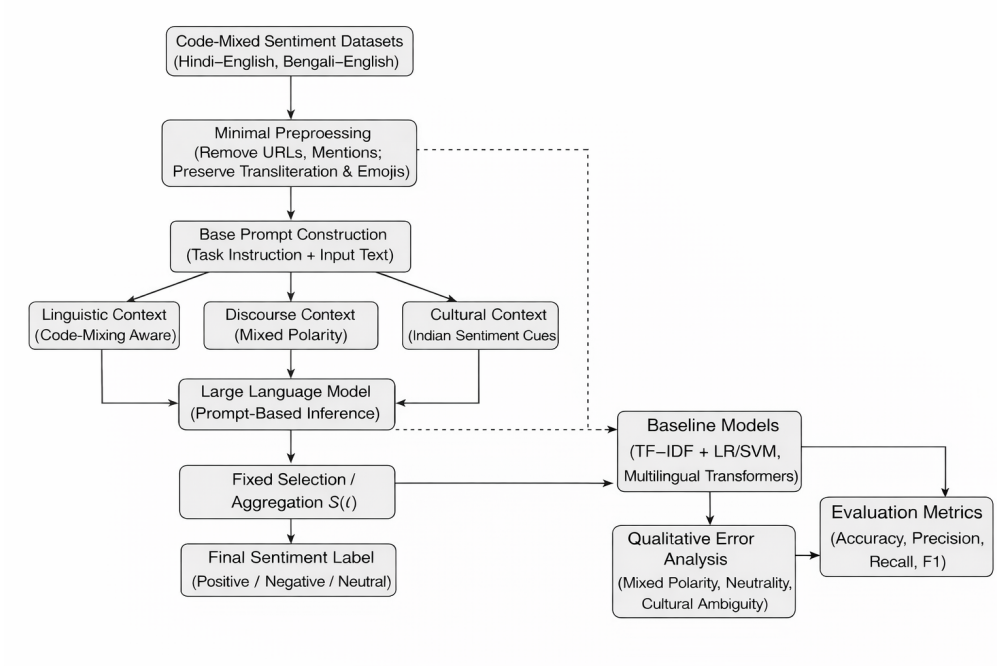

Fig. 1. System architecture for sentiment analysis of code-mixed Indian languages using context-engineered prompts

3.6 Evaluation Metrics

Model performance is evaluated using standard classification metrics. Accuracy is defined as:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} I(\hat{y}_i = y_i), \tag{9}$$

where $I(\cdot)$ is the indicator function. The measures of precision, recall, and F1-score are calculated for each sentiment category and then averaged. Apart from the quantitative evaluation, error analysis is also performed through manual inspection of the incorrectly classified samples to detect the linguistic and cultural patterns of errors.

3.7 Experimental Validity

Experiments are performed under fixed inference conditions, which include temperature and the limit of the tokens that the language model is allowed to use. All prompt templates, context conditions, and dataset partitions remain constant throughout the experiments. The study does not attempt to generalize outside the datasets used for evaluation and confines itself only to empirical observations of prompts.

Figure 1 illustrates the overall procedure of the proposed research approach. It covers data pre-processing, prompt engineering for the given context, inference using large language models based on prompts, and aggregation of contextual results. The figure also illustrates the process of baseline model comparison and evaluation.

4 Experimental Setup and Evaluation

This section describes the experimental setup in relation to the proposed prompt-based sentiment analysis system which is focused on controlled analysis rather than long time model training.

4.1 Datasets

The three publicly available code mixed sentiment analysis datasets that were employed for Bengali-English and Hindi-English are those that have been selected as being appropriate to

Algorithm 1 In-Depth Workflow for Context-Engineered Prompt-Based SA

Require: Code-mixed sentiment dataset $X = \{x_1, x_2, \ldots, x_N\}$
Require: Ground-truth sentiment labels $Y = \{y_1, y_2, \ldots, y_N\}$
Require: Base prompt template $P(\cdot)$
Require: Context set $\mathcal{C} = \{C_{\text{ling}}, C_{\text{disc}}, C_{\text{cult}}\}$
Require: Large Language Model $\mathcal{M}$ (prompt-based inference)
Ensure: Predicted sentiment labels $\hat{Y}$ and evaluation metrics

Initialize prediction set $\hat{Y} \leftarrow \emptyset$
for each input instance $x_i \in X$ **do**
  Apply minimal preprocessing:

$$\tilde{x}_i \leftarrow g(x_i)$$

  {Remove URLs and mentions; preserve transliteration and emojis}
  Initialize context-specific predictions $\mathcal{P}_i \leftarrow \emptyset$
  for each context $C_k \in \mathcal{C}$ **do**
    Construct context-engineered prompt:

$$p_i^{(k)} \leftarrow C_k \oplus P(\tilde{x}_i)$$

    Query the language model:

$$o_i^{(k)} \leftarrow \mathcal{M}(p_i^{(k)})$$

    Extract sentiment label $\hat{y}_i^{(k)}$ from $o_i^{(k)}$
    Add $\hat{y}_i^{(k)}$ to $\mathcal{P}_i$
  end for
  Select final sentiment prediction $\hat{y}_i$ from $\mathcal{P}_i$ {Fixed selection rule (e.g., majority or priority-based)}
  Append $\hat{y}_i$ to $\hat{Y}$
end for
Train baseline sentiment classifiers on X using identical data splits
Obtain baseline predictions $\hat{Y}_{\text{base}}$
Compute evaluation metrics:

$$\text{Accuracy}, \text{Precision}, \text{Recall}, \text{F1-score}$$

for both $\hat{Y}$ and $\hat{Y}_{\text{base}}$
Perform qualitative error analysis on misclassified instances {Analyze mixed polarity, neutral ambiguity, cultural sentiment cues}
return $\hat{Y}$ and evaluation results

the required criteria of being low resource, relevant and of high quality of annotation. In addition, these have been chosen to allow consistent comparison using the same datasets.

SentMix-3L is used as the primary multilingual benchmark dataset. Sentiments can be classified positively, negatively or neutrally based on code-mixed content found within various social networks using Bangla, Hindi, and English language combinations [17]. These classifications also represent the extraction of actual linguistic features, such as the use of non-standard transliteration practices, non-standard syntactic forms, and code-switching (the act of switching back and forth between multiple languages within the same conversation or sentence) as they apply in a multilingual, low-resourced setting.

SemEval 2020 Task 9 (Hinglish) It is the industry standard when it comes to sentiment analysis of Hindi-English Code Mixed User-Generated Twitter Data [16].

BnSentMix applied a specialized evaluation approach to Bengali-English Code Mixed Sentiment Analysis using an annotated data sample of informal user-generated content sourced from various platforms such as Facebook and YouTube. The nature of pragmatics, colloquialisms, and Mixed Language use in Bangla-English conversations is reflected in BnSentMix [1].

Each dataset uses a train-split, validation-split, and test-split to annotate sentiment for a three-way classification. No synthetic data has been used. All choices are designed to assess the effect of context-specific engineered prompts on code-mixed Indian language sentiment analysis in low resource contexts and therefore contain ample linguistic variations to allow comparisons against established benchmarks.

4.2 Experimental Environment and Parameters

TF-IDF classifiers, along with all other traditional methods of Natural Language Processing, were tested using Python 3.10, the NLTK and Scikit-Learn libraries, and on low-resource Windows 11 (Intel Core i5, 8 GB RAM) systems frequently found in academia, using no GPU acceleration. The use of the GPT-4o-mini model

Table 2. Performance comparison on code-mixed sentiment datasets

Method	Acc.	F1
TF-IDF + LR	61.3	58.7
TF-IDF + SVM	63.9	60.8
mBERT (fine-tuned)	67.2	65.4
Static Prompt (LLM)	64.5	62.1
+ Linguistic Ctx	65.3	63.0
+ Discourse Ctx	65.0	62.7
+ Cultural Ctx	65.6	63.4
Combined Contexts (Proposed)	**66.1**	**64.0**

with the OpenAI API enables the researcher to easily carry out prompt-based sentiment inference without any contextual bias. For code-mixed Indian languages, inference criteria are based on consistency, plausibility and stability of results through repetition.

4.3 Evaluation Metrics

To assess the effectiveness of the proposed framework, standard evaluation metrics commonly used in sentiment analysis are employed. These metrics provide a balanced view of classification performance across sentiment categories:

— **Accuracy (Acc):** Measures the proportion of correctly classified sentiment instances.

— **Precision:** Indicates the proportion of correct predictions among all instances predicted for a given sentiment class.

— **Recall:** Measures the proportion of correctly identified instances for each sentiment class.

— **F1-Score:** The harmonic mean of precision and recall, reported as a macro-average to account for potential class imbalance.

In order to discover error trends in the context of mixed polarity expressions or sentiment ambiguity in neutrals and culturally implied sentiment information, qualitative error analysis of the results in conjunction with the above-mentioned numerical data even requires manual inspection of the erroneous classifications.

4.4 Baseline Models

The effectiveness of available sentiment analysis techniques, from conventional machine learning approaches like traditional support vector machines and logistic regression, and recent neural approaches, is explored to assess the efficacy of the proposed approach. In conventional techniques, the focus is on fixed low-level attributes like TF-IDF and traditional models like support vector machines and logistic regression, whereas learning the context relies on pretrained multi-lingual models and their optimization on the same dataset, which is effective when labeled data is available.

Static prompting, which only utilizes the prompting command while ignoring other forms of context, is employed within the task to leverage the prompted task as a means to examine the feasibility of the baseline task of sentiment classification. Moreover, as can be noted, the entire task emphasizes the usage of identical partitions to ensure the evaluation of the performance of the entire baseline task. Consequently, as can also be noted, the task deliberately omitted the employment of sophisticated optimization techniques to ensure the research was only meant to examine quick behavior instead of performance optimization through computationally costly methods.

5 Results and Discussion

Without any claims regarding optimality or state-of-the-art performance, this section will outline the experimental outcome of the context-engineered prompt-based sentiment analysis framework, focusing on performance trends associated with empirical behavior, interpretability, and robustness in low-resource scenarios.

5.1 Quantitative Results

The performance of the proposed method is summarized in Table 2 in comparison with classical machine learning baselines, a fine-tuned multilingual transformer model, and the use

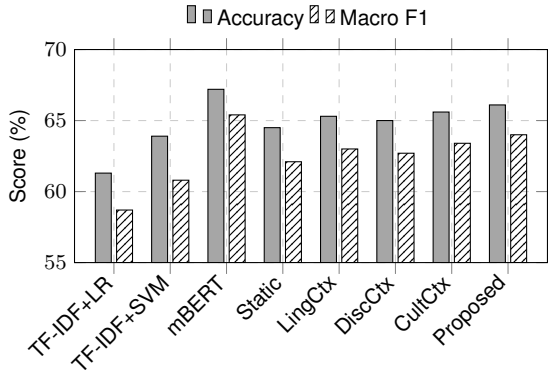

Fig. 2. Grayscale comparison of accuracy and macro F1-score across baseline models and context-engineered prompt configurations

Table 3. Effect of different context types on prompt-based sentiment inference

Prompt Configuration	Accuracy (%)	Macro F1 (%)
Base Prompt Only	64.5	62.1
+ Linguistic Context	65.3	63.0
+ Discourse Context	65.0	62.7
+ Cultural Context	65.6	63.4
Combined Contexts	**66.1**	**64.0**

of static prompts for inference. In general, context-specific prompting outperforms static prompting in all considered datasets.

In case of sentimental sentences that are obvious, the classical machine learning approach using TF-IDF features is efficient; nonetheless, it fails when it comes to dealing with informal expressions and contrastive language in code-mixing text. Despite their reliance on annotated data and computational capacity, multilingual transformer models with more advanced capabilities are comparable. Yet, the proposed technique behaves similarly under resource constraints.

The comparative performance of the various baselines and prompts is presented graphically in Figure 2. This chart reflects the gradual influence of contextual information in prompt-based sentiment analysis. There are no disproportionate gains in performance implied here.

5.2 Impact of Context Engineering

To analyze the contribution of different context types, experiments are conducted using individual context variants as well as their combinations. Table 3 reports the effect of linguistic, discourse-level, and cultural context on prompt-based sentiment inference.

Linguistic context improves sentiment recognition in sentences containing explicit code-switching and contrastive markers, while discourse-level context assists in resolving sentiment shifts introduced by connectors such as *but*. Cultural context contributes to better interpretation of pragmatically weak or neutral expressions commonly used in Indian languages. Although no single context type consistently dominates across all instances, combining multiple context cues through a fixed aggregation strategy yields more consistent predictions.

5.3 Illustrative System Outputs

To further demonstrate the behavior of the proposed framework, illustrative system outputs obtained via API-based prompt inference will be included. These examples will present the complete interaction flow, including the input sentence, the context-engineered prompt, and the corresponding sentiment prediction produced by the system. The screenshots are intended to provide visual evidence of system behavior and are included for illustrative purposes only.

Figure 3, 4, and 5 present representative system outputs obtained through API-based inference using context-engineered prompts for Bengali–English, Hindi–English, and English inputs, respectively.

Input Sentence: এটা ঠিকঠাক লাগলো, টাইম পাসের জন্য (*It felt okay, just for time pass*)
Prompt: Analyze the sentiment of the following Bengali–English code-mixed sentence. Consider cultural expressions such as "ঠিকঠাক" and informal usage. Classify the sentiment as Positive, Neutral, or Negative.
System Output: Sentiment: **Neutral**

Fig. 3. Sample system output for Bengali-English (Banglish) input using context-engineered prompts

Input Sentence: यह ठीक-ठाक लगा, टाइम पास के लिए (*It felt okay, just for time pass*)
Prompt: Analyze the sentiment of the following Hindi–English code-mixed sentence. Consider colloquial Hindi expressions such as "ठीक-ठाक" and everyday usage. Classify the sentiment as Positive, Neutral, or Negative.
System Output: Sentiment: **Neutral**

Fig. 4. Sample system output for Hindi-English (Hinglish) input using context-engineered prompts

5.4 Language-Specific Qualitative Analysis

To complement the quantitative evaluation, a language-specific qualitative analysis is conducted to examine how different context-engineered prompts influence sentiment predictions for code-mixed Indian language inputs.

The analysis focuses on behavioral differences across contextual configurations rather than iterative prompt refinement.

Bengali–English Examples. Table 4 presents representative examples in Bengali, shown in native script with English explanations for clarity. These examples illustrate common challenges such as mixed polarity, informal expressions, and culturally implicit sentiment.

Input Sentence: *The movie was good, but the ending was disappointing.*
Prompt: Analyze the sentiment of the following sentence. Consider the contrastive discourse marker "but" when determining overall sentiment. Classify the sentiment as Positive, Neutral, or Negative.
System Output: Sentiment: **Negative**

Fig. 5. Sample system output for English input using context-engineered prompts

Table 4. Qualitative analysis of sentiment predictions using different context-engineered prompts (Bengali)

Code-Mixed Input	Base	Ling.	Cult.
(Movie was good but the ending was disappointing)	Pos.	Neu.	Neg.
, *(It felt okay, just for time pass)*	Pos.	Neu.	Neu.
, *(The service was very bad, but the staff were polite)*	Neg.	Neu.	Neg.
(The phone is expensive but the performance is quite good)	Neu.	Neu.	Pos.

Static prompts are prone to polarity bias, as they tend to overemphasize surface-level attitude indicators, as has been analyzed in the cases of Bengali. Although they may express weak or neutral emotions in common conversation, "" (*okay*) or " " (*time pass*) are very often perceived as positive in generic requests. The inclusion of cultural context prompts helps in better comprehension and results in more accurate interpretations of sentiment.

Hindi-English Examples. Table 5 presents analogous examples for Hindi–English code-mixed inputs.

Table 5. Qualitative analysis of sentiment predictions for Hindi–English code-mixed inputs

Code-Mixed Input	Base	Ling.	Cult.
(The movie was good but the ending was disappointing)	Pos.	Neu.	Neg.
- , *(It felt okay, just for time pass)*	Pos.	Neu.	Neu.
, *(The service was very bad, but the staff were polite)*	Neg.	Neu.	Neg.
(The phone is expensive but the performance is quite good)	Neu.	Neu.	Pos.

The Hindi examples exhibit trends similar to those observed for Bengali inputs. Generic prompts tend to overemphasize explicit positive or negative adjectives, leading to polarity bias in sentences containing contrastive discourse markers.

Linguistic context improves handling of Hindi–English code-switching, while cultural context aids in interpreting pragmatically weak expressions such as ”-” (*okay*), which are frequently misclassified under static prompting.

English Example. For English sentences containing contrastive discourse, such as *“The movie was good, but the ending was disappointing”*, static prompting often exhibits polarity bias toward the dominant positive clause. Context-engineered prompts incorporating discourse-level cues more consistently capture the overall negative sentiment by accounting for the contrastive structure.

5.5 Qualitative Error Analysis

Sentiment Analysis Error Patterns (e.g. Sarcasm, Implicit Sentiment) were found through analyzing misclassified samples qualitatively. Neutral Sentiment Statements are more difficult to classify than others due to the context-dependent nature of the statement.

It was demonstrated that Context-Engineered Prompting will increase the interpretability of sentiment classification from code-mixed Indian languages, however there are limitations to Prompt-Based Inference Methods when utilizing this approach.

6 Discussion

The study aims to explore the impact of context engineering techniques on the sentiment analysis of code-mixed Indian languages with a particular emphasis on the development of prompt-based techniques. It has been shown that with the incorporation of context techniques, the generation of consistent sentiment analysis predictions is possible. There is an increase in the accuracy and macro F1-measure. Interesting findings: the context can reduce the polarity bias of the data since the method encourages the data models to pay attention to the interpretation of the data rather than focusing on positive or negative terminologies.

The emphasis is seen in the consideration of the Bengali and Hindi combinations with English. Such a method is not useful for the interpretation of contexts like metaphors or the use of sarcasm. So, in order to add robustness while avoiding alterations in its core capabilities, prompt engineering ought to be accompanied by model adaptation, as well as supervised learning techniques. With publicly available datasets, as well as API-based testing, this method provides a simple approach for sentiment analysis in low-resource setups; therefore, this technique might find applications in a wide range of applications.

The technique presents empirical studies for understanding the effectiveness of context-engineered prompts in sentiment analysis of code-mixed Indian languages, rather than asserting its optimal testing, thereby creating pathways for further studies involving prompt engineering and fine-tuning techniques.

7 Future Scope

It studies the low-resource context-engineered prompt-based sentiment analysis for code-mixed Indian languages. It provides a simple, understandable framework employing pre-defined prompts for controlled analysis

and recommendations of future study directions. The development of fast formulations, further expansion of language coverage beyond Bengali-English and Hindi-English, and adaptable aggregation techniques is a priority.

This work highlights the difficulty of dealing with implicit sentiment such as irony and sarcasm, and it is proposed that future improvements can be achieved by extending this approach to other related tasks while maintaining viability and interpretability, which will improve the understanding of context engineering in low-resource multilingual settings.

References

1. **Alam, S., Ishmam, M. F., Alvee, N. H., Siddique, M. S., Hossain, M. A., Kamal, A. R. M. (2025).** Bnsentmix: A diverse bengali-english code-mixed dataset for sentiment analysis. Proceedings of the First Workshop on Language Models for Low-Resource Languages, pp. 68–77.
2. **Bal, S., Mandal, L. (2026).** A prompt-based interactive dialogue framework for low-resource Indian languages. Procedia Computer Science, Vol. 282, pp. 220–229.
3. **Bal, S., Mishra, S., Mandal, L. (2025).** A review of quantum computing approaches to semantic search and text classification in natural language processing. The 1st Workshop on QuantumNLP, pp. 33.
4. **Chen, B., Zhang, Z., Langrené, N., Zhu, S. (2025).** Unleashing the potential of prompt engineering for large language models. Patterns.
5. **Geng, M., Wang, S., Dong, D., Wang, H., Li, G., Jin, Z., Mao, X., Liao, X. (2024).** Large language models are few-shot summarizers: Multi-intent comment generation via in-context learning. Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, pp. 1–13.
6. **Hollmann, N., Müller, S., Hutter, F. (2023).** Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. Advances in Neural Information Processing Systems, Vol. 36, pp. 44753–44775.
7. **Kapil, P., Kumari, G., Ekbal, A., Pal, S., Chatterjee, A., Vinutha, B. (2023).** HHSD: Hindi hate speech detection leveraging multi-task learning. IEEE Access, Vol. 11, pp. 101460–101473.
8. **Koto, F., Beck, T., Talat, Z., Gurevych, I., Baldwin, T. (2024).** Zero-shot sentiment analysis in low-resource languages using a multilingual sentiment lexicon. Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 298–320.
9. **Kowtal, N., Deshpande, T., Joshi, R. (2024).** Chain-of-translation prompting (CoTR): A novel prompting technique for low resource languages. Proceedings of the 38th Pacific Asia Conference on Language, Information and Computation, pp. 645–655.
10. **Marvin, G., Hellen, N., Jjingo, D., Nakatumba-Nabende, J. (2023).** Prompt engineering in large language models. International conference on data intelligence and cognitive informatics, Springer, pp. 387–402.
11. **Mishrra, S., Barman, A. K., Senapati, A. (2025).** Hate content detection through prompt engineering: A pioneering attempt. Journal of Information and Optimization Sciences, Vol. 46, No. 1, pp. 107–119.
12. **Mishrra, S., Kumar Barman, A., Senapati, A. (2025).** PromptMaster: Engineering essentials and basic NLP tools. Computación y Sistemas, Vol. 29, No. 3.
13. **Nazir, M. K., Faisal, C. N., Habib, M. A., Ahmad, H. (2025).** Leveraging multilingual transformer for multiclass sentiment analysis in code-mixed data of low-resource languages. IEEE Access.

14. **Pakray, P., Krishna, R., Pal, S., Vetagiri, A., Dash, S., Maji, A. K., Lyngdoh, S. A., Laitonjam, L., Jamatia, A., Sambyo, K., et al. (2025).** Findings of WMT 2025 shared task on low-resource Indic languages translation. Proceedings of the Tenth Conference on Machine Translation, pp. 532–553.

15. **Pal, S., Pakray, P., Jain, P., Ekbal, A., Bandyopadhyay, S. (2025).** Editorial for proceedings of the quantumNLP{:} integrating quantum computing with natural language processing. Proceedings of the QuantumNLP{:} Integrating Quantum Computing with Natural Language Processing.

16. **Patwa, P., Aguilar, G., Kar, S., Pandey, S., PYKL, S., Gambäck, B., Chakraborty, T., Solorio, T., Das, A. (2020).** SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. Proceedings of the Fourteenth Workshop on Semantic Evaluation (SemEval@ACL), Association for Computational Linguistics, Barcelona (online), pp. 774–790. DOI: 10.18653/v1/2020.semeval-1.100.

17. **Raihan, M. N., Goswami, D., Mahmud, A., Anastasopoulos, A., Zampieri, M. (2023).** SentMix-3L: A Bangla-English-Hindi code-mixed dataset for sentiment analysis. arXiv preprint arXiv:2310.18023. Dataset available at GitHub; Last accessed: January, 2026.

18. **Rani, P., Fransen, T., McCrae, J. P., Negi, G. (2024).** MaCmS: Magahi code-mixed dataset for sentiment analysis. Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pp. 10880–10890.

19. **Rayala, U. R., Seshadri, K., Sristy, N. B. (2023).** Sentiment analysis of code-mixed Telugu-English data leveraging syllable and word embeddings. ACM Transactions on Asian and Low-Resource Language Information Processing, Vol. 22, No. 10, pp. 1–30.

20. **Sampath, K. K., Supriya, M. (2024).** Transformer based sentiment analysis on code mixed data. Procedia Computer Science, Vol. 233, pp. 682–691.

21. **Strobelt, H., Webson, A., Sanh, V., Hoover, B., Beyer, J., Pfister, H., Rush, A. M. (2022).** Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. IEEE transactions on visualization and computer graphics, Vol. 29, No. 1, pp. 1146–1156.

22. **Wang, S. (2026).** Domain adaptation using transformer models for automated detection of exterior cladding materials in street view images. Scientific Reports, Vol. 16.

23. **Zeng, L. (2024).** Leveraging large language models for code-mixed data augmentation in sentiment analysis. Proceedings of the Second Workshop on Social Influence in Conversations (SICon 2024), pp. 85–101.

Article received on 29/03/2026; accepted on 20/06/2026.
**Corresponding author is Sauvik Bal.*