SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Backbone Matters More Than Architectural Complexity: A Controlled Empirical Study of Hinglish Sentiment Classification

Apoorva Singh[1]*, Raj Shree[2], Sandhya Satyarthi[3], Dhirendra Pandey[4], Gaurav Singh[5], Ravi Prakash Pandey[6]

[1]Department of Information Technology, Babasaheb Bhimrao Ambedkar University, Lucknow. E-mail: suryavanshi0512@gmail.com
[2]Department of Information Technology, Babasaheb Bhimrao Ambedkar University, Lucknow.
E-mail: rajshree.bbau2009@gmail.com
[3]Department of Information Technology, Manipal University Jaipur. E-mail: sandhya.satyarthi@jaipur.manipal.edu
[4]Department of Information Technology, Babasaheb Bhimrao Ambedkar University, Lucknow. E-mail: prof.dhiren@gmail.com
[5]Department of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow. E-mail: gaurav2512tech@gmail.com
[6]Dr Rammanohar Lohia Avadh University, Ayodhya. E-mail: ravi.p.pandey@gmail.com
*Corresponding author: suryavanshi0512@gmail.com

ABSTRACT
Hindi–English code-mixed (Hinglish) text is still a challenging problem for sentiment classification because of the inconsistent transliteration, frequent code switching and linguistic variability. A number of recent works have introduced ever more intricate neural architectures, yet the impact of each architectural element and how it contributes to the finetuned backbone, language model, is not fully understood. In this paper we provide a controlled empirical analysis of the backbone selection and downstream architectural complexity on the SemEval-2020 Task 9 Hinglish sentiment dataset. Four pretrained transformer models are fine-tuned and evaluated with exactly the same settings for preprocessing, optimization, and evaluation. For every backbone, this is done by comparing a plain fine-tuning version to an architecture-enhanced version that includes CNN, BiGRU, attention, language-tag embeddings, and gated feature fusion. The effect of model robustness is then investigated in terms of the Code-Mixing Index (CMI) at various linguistic complexity levels. Experimental results demonstrate that plain MuRIL outperforms its architecture enhanced variant with 84.11% Macro-F1 score, while the latter has a Macro-F1 of 71.20%. McNemar's test and bootstrap resampling verify that the difference in performances is statistically significant at the $p < 0.0001$ level. The findings further illustrate the benefit of using a code-mix-aware pretrained backbone as compared to architectural richness in the downstream layers for the Hinglish sentiment task. Through the study, the authors offer useful insights and practical recommendations for building effective and effective sentiment classification systems for multilingual code-mixed text.

Keywords: Hinglish Sentiment Analysis, Code-Mixed NLP, MuRIL, Pretrained Language Models, Controlled Empirical Study, Code-Mixing Index (CMI), Transformer Models

1. Introduction

The rise of multilingual communication in social media, web-based learning, e-commerce and messaging has resulted in a significant amount of code-mixed textual data. Of these, Hindi–English code-mixed text (Hinglish), a phenomenon that is widely referred to as 'Hinglish' is one of the most common ways of communicating digitally in India. The automated sentiment analysis of such content has become more significant for various applications such as Opinion Mining, Customer Feedback Analysis, Recommender Systems, Social Media Monitoring, Intelligent Conversational Agents, etc. It is much harder to understand sentiment in the case of Hinglish than the monolingual written text, due to the use of multiple languages within a single sentence, inconsistent sentence structure, inconsistent transliteration conventions.

Hinglish is different from monolingual English and Hindi as it has several characteristics which make sentiment classification challenging. Lexical sparsity is high with Hindi words being often spelt in Roman letters in multiple variants, such as acha, accha, achha etc. Also, the need for language switching within sentences, the lack of standardization in transliteration, the use of informal language in social media, emoji, hashtags and non-standard grammar make traditional NLP methods less effective. These properties tend to make pretrained language models which have been trained mainly on monolingual corpora produce split token representations and a lack of semantic understanding.

Pretrained language models have made significant strides in recent years, leading to a significant improvement in multilingual sentiment analysis. The models like multilingual BERT (mBERT), XLM-RoBERTa, MuRIL, HingBERT and HingRoBERTa have shown promising performance for code-mixed NLP tasks. To achieve higher classification accuracy, many studies have added extra downstream models to

these pretrained encoders, such as convolutional neural networks for local feature extraction, recurrent neural networks for sequential modeling, attention mechanisms for weighting features based on their context, gated feature-fusion models, and language-aware embeddings. Although many of these methods claim performance gains, the importance of the extra architectural elements with respect to the pretrained backbone is not studied in great detail.

Previous works have tended to concentrate on making increasingly more complex architectures, most of the time taking the view that more complex architectures will necessarily be more effective. But, few studies systematically explore whether the improvements come from the newly introduced modules or from the pretrained backbone, which serves as a base of the model. This means that the role of the backbone and downstream architectural complexity in Hinglish sentiment classification is yet an open research question. If the comparisons are not conducted in a controlled manner, it will be hard to know whether greater model complexity is really helping or just unnecessarily adding to the computational load.

In this regard, this work presents an empirical controlled evaluation of backbone selection and downstream architectural complexity for Hinglish sentiment classification. In particular, we evaluate the performance of plain fine-tuning of the code-mix-aware MuRIL encoder with the same encoder equipped with CNN, BiGRU-attention, gated-fusion and language-tag embedding modules, while using the same preprocessing, optimization and evaluation conditions. Moreover, robustness of models is examined by applying the Code-Mixing Index (CMI) based evaluation which is able to investigate different levels of code-switching.

Research Question (RQ):

When a strong code-mix-aware pretrained language model is employed, does increasing downstream architectural complexity provide statistically significant improvements in Hinglish sentiment classification?

The key contributions of this work are summarised below:

1. We perform a controlled empirical analysis of code-mix-aware sentiment classification, which compares the MuRIL fine-tuning approach to the same backbone supplemented with CNN, BiGRU-attention, gated-fusion, and language-tag embedding modules, under identical preprocessing, training and evaluation conditions.
2. We show that sentiment classification in Hinglish is not directly proportional to the number of patterns used in the architecture. Plain MuRIL consistently outperforms the more complex architecture in repeated experiments, suggesting that the selection of pretrained backbone is more important than the extra down-stream modules investigated in this study.
3. We conduct robustness analysis using Code-Mixing Index (CMI) to gain a more detailed account of model behavior than is offered by the aggregate analysis measures.
4. We present a clear empirical study where the complexity of the parameters, computational cost, replicated negative results and methodological limitations are reported, providing guidelines for future research for code-mixed sentiment analysis.
5.

Novelty of the Study:

Unlike previous studies that primarily propose increasingly complex neural architectures for Hinglish sentiment classification, the objective of this work is to systematically evaluate the relative importance of pretrained backbone selection and downstream architectural complexity under strictly controlled experimental conditions. Rather than introducing another architecture, this study investigates whether commonly adopted architectural enhancements, including CNN, BiGRU, attention, language-tag embeddings, and gated feature fusion, provide statistically significant improvements when a strong code-mix-aware pretrained language model is already employed. By maintaining identical preprocessing, optimization, and evaluation settings across all experiments, the study isolates the effect of architectural complexity from that of pretrained backbone selection. This controlled empirical perspective distinguishes the present work from existing architecture-centric studies and provides practical guidance for designing efficient and robust Hinglish sentiment classification systems.

Rest of the paper is structured as follows. The review of related work is provided in Section 2. The proposed methodology is presented in section 3. The dataset and experimental setup are explained in section 4. The experimental results are presented in Section 5. The findings are discussed in Section 6. The limitation of the study is detailed in section 7. Finally, the paper is concluded in Section 8 which includes future research directions.

2. Literature Review

2.1 Hinglish and Code-Mixed Sentiment Analysis

The code-mixed text is a crucial research field because of the growing popularity of multilingual communication, such as on social media, on online reviews, in messaging applications and conversational

platforms. Hindi–English (Hinglish) is widely used amongst the various code-mixed language pairs which has also garnered a lot of attention. The release of SemEval-2020 Task 9 benchmark dataset offered a standard benchmark dataset and framework, which propelled the research in Hinglish sentiment classification to a great extent [2]-[5].

The first solutions were based on manually designed linguistic characteristics and classic machine learning classifiers. After that, the deep learning approaches using CNN, BiLSTM and attention mechanism showed better performance on code-mixed sentiment analysis [2]–[4]. But the complicated linguistic properties of Hinglish, such as transliteration variation, informal grammar and language switching, still pose significant challenges to classify sentiments correctly [5].

All these studies showcase that deep learning algorithms can achieve good performance in the problem of Hinglish Sentiment Analysis, but most of them mainly aim at enhancing the classification accuracy through the introduction of more and more complex neural architectures. Relatively little research attention has thus far been directed towards understanding which aspects of these architectures are mainly responsible for the reported performance gains. This is a limitation that inspires the necessity of empirical assessment in a controlled manner.

2.2 Pretrained Language Models for Indian Languages

Pretrained transformer models have made a significant contribution to the field of multilingual natural language processing. BERT was shown to be effective for using bidirectional contextual language representations and to provide good baselines for many NLP tasks [6]. Later, multilingual BERT (mBERT) [6] and XLM-R [7] were able to expand transformer-based learning to multilingual corpora, allowing cross-lingual transfer across over 100 languages.

These multilingual models are able to work across a variety of languages, but were not created for Indian languages or for code-mixed text written in Roman script. To overcome this, MuRIL was pretrained on multiple Indian language monolingual, translated and transliterated corpora, which results in better representations for multilingual Indian-language NLP tasks [8]. Likewise, HingBERT and similar models have been directly trained on large-scale Hinglish corpora and have been shown to have superior performance on downstream tasks than generic multilingual transformers on a number of code-mixed NLP tasks [9]. Comparative studies also showed that language-specific pretrained models always outperforms the multilingual models for the task of sentiment classification in Hinglish [10].

Recent research continues to investigate transformer-based architectures for Hinglish sentiment and emotion analysis, demonstrating the effectiveness of pretrained language models in handling multilingual conversational text [11]–[13].

Although the use of pretrained language models has greatly improved multilingual NLP, the evaluation of various models in comparative studies may vary in the way it is pre-processed, optimized, or used in the downstream architecture. These discrepancies make it hard to identify the source of the gains in observed performance: whether they are due to the backbone itself or due to a variation in experimental design. A controlled comparison, therefore, is required [8], [10]–[13].

2.3 Architectural Enhancement Techniques

In addition to choosing more powerful pretrained language models, a number of research and development studies have pointed out other neural networks that can improve the accuracy of sentiment classification. CNNs work well to model local contextual patterns, and recurrent networks like BiLSTMs and BiGRUs work well to capture sequential dependencies. While attention mechanisms selectively highlight words that may carry sentiment, gated fusion mechanisms combine complementary semantic representations from several feature sources [2]–[4].

Recent transformer-based studies use pretrained language models, combined with recurrent networks, convolutional layers, ensemble learning and multi-task optimization to enhance the classification performance [11]–[14]. Many of these approaches claim to improve performance but do not report improvements for the individual components, making it challenging to assess individual component improvements.

Although CNNs, RNNs, attention mechanisms, and feature fusion are widely adopted, it is not clear whether they contribute independently. Since these modules are often added to more powerful language models that have already been pretrained, it is challenging to assess the contribution of each module independently. It does not mean, however, that the more complex the architecture, the more accurate the sentiment classification will be. [11]–[14]

2.4 Evaluation of Code-Mixed NLP's Robustness

Most studies currently available assess the sentiment classification performance based on overall metrics like accuracy and macro F1. These statistics give a general idea of model they don't specify whether models will hold up when code-switching is either higher or lower. The Code-Mixing Index (CMI) was developed as an understandable tool to measuring the levels of multilingual blending in an utterance [15].

It has been recently seen that it is necessary to consider the different language-switching scenarios and multilingual models with different linguistic complexity, as opposed to just looking at aggregate

performance measures [16], [17]. This robustness analysis adds to understanding model generalization and deployment in multi-lingual settings.

There is growing emphasis on robustness analysis due to the fact that overall accuracy is not a sufficient indicator of model behavior in a multilingual setting. Models that achieve comparable aggregate scores can yield significantly different performance across the different levels of code switching. Therefore, the robustness evaluation based on Code-Mixing Index (CMI) offers additional insights beyond traditional classifiers on model generalization.

Table 1. Summary of existing studies on Hinglish sentiment classification

Study	Backbone	Additional Architecture	Dataset	Limitation
Patwa et al. [1]	Multiple	Transformer baselines	SemEval-2020	Benchmark only
Khanuja et al. [8]	MuRIL	None	Indian multilingual datasets	No downstream comparison
Ghosh et al. [11]	Multilingual Transformer	Transformer	Code-mixed tweets	No controlled backbone comparison
Recent hybrid models [12–14]	MuRIL/HingBERT	CNN/BiGRU/Attention	Various	No component-wise evaluation
This Work	BERT, mBERT, XLM-R, MuRIL	Plain vs Hybrid	SemEval-2020	Controlled comparison

2.5 Research Gap

Although substantial progress has been made in Hinglish sentiment analysis, several important research gaps remain. First, most previous studies focus primarily on proposing increasingly sophisticated neural architectures without systematically determining whether these architectural additions provide measurable improvements beyond strong pretrained language models [10]–[14]. Second, relatively few studies perform controlled comparisons that isolate the contribution of pretrained backbone selection from downstream architectural complexity while maintaining identical preprocessing, optimization, and evaluation settings . Third, robustness across different levels of code-switching remains insufficiently explored, as most existing work reports only aggregate evaluation metrics rather than performance stratified by linguistic complexity [15]–[17].

However, most reported improvements are based on end-to-end architectures, making it difficult to determine whether the observed gains arise from the pretrained backbone or the additional downstream components.

Motivated by these observations, the present study performs a controlled empirical evaluation of backbone selection and architectural complexity for Hinglish sentiment classification. Specifically, we compare plain fine-tuning of the code-mix-aware MuRIL encoder with the same backbone augmented by CNN, BiGRU-attention, gated-fusion, and language-tag embedding modules under identical experimental settings. Furthermore, robustness is evaluated using Code-Mixing Index (CMI)-based analysis to investigate whether increased architectural complexity provides measurable benefits across different levels of code-switching.

3. Methodology

3.1 Overview of the Experimental Framework

Unlike previous studies that primarily focus on proposing increasingly sophisticated neural architectures for Hinglish sentiment classification, the objective of this work is to perform a controlled empirical evaluation of the relative contribution of pretrained backbone selection and downstream architectural complexity. Specifically, this study investigates whether augmenting a strong code-mix-aware pretrained language model with additional neural components including convolutional feature extraction, sequential modeling, attention mechanisms, gated feature fusion, and language embedding provides measurable improvements over plain fine-tuning of the pretrained backbone.

All evaluated configurations use the same experimental pipeline to make a fair comparison and be scientifically sound. All models are trained with the same pre-processing, tokenizer, optimisation, learning rate, early stopping and evaluation. The only experimental parameters are (i) the choice of pretrained language model, and (ii) whether downstream architectural modules are used or not. This means that any

difference in performance noted can be directly attributed to these factors and not due to variation in data processing or training technique.

The whole experimental framework in this study is as following:

1. Data pre-processing (text cleaning, language identification, transliteration, normalisation, tokenization and computation of the Code-Mixing Index (CMI)).
2. Semantic Representation Learning: This approach involves first pre-processing the text and then using a pretrained transformer model to encode the text into a representation. Several backbone models are explored, especially MuRIL, as it was pretrained on multilingual Indian-language corpora.
3. Architectural enhancement, which involves passing the encoded representations through other neural modules (such as language aware BiGRU modeling, convolutional feature extraction, attention, and gated feature fusion) optionally.
4. The last document representation is given to a fully connected layer for sentiment prediction, either positive, negative or neutral.
5. Performance evaluation: All configurations are evaluated in the same way, with the same set of evaluation metrics. Traditional analyses of classification, as well as Code-Mixing Index (CMI) analysis is used to investigate the performance of the models under varying levels of code switching.

The goal of this framework is to check whether there is any benefit in adding more parts within the neural network to the top of a good code-mix-aware pretrained encoder. The methodology, hence, serves as a systematic approach for comparing pretrained representations and downstream architecture for sentiment classification of Hinglish.

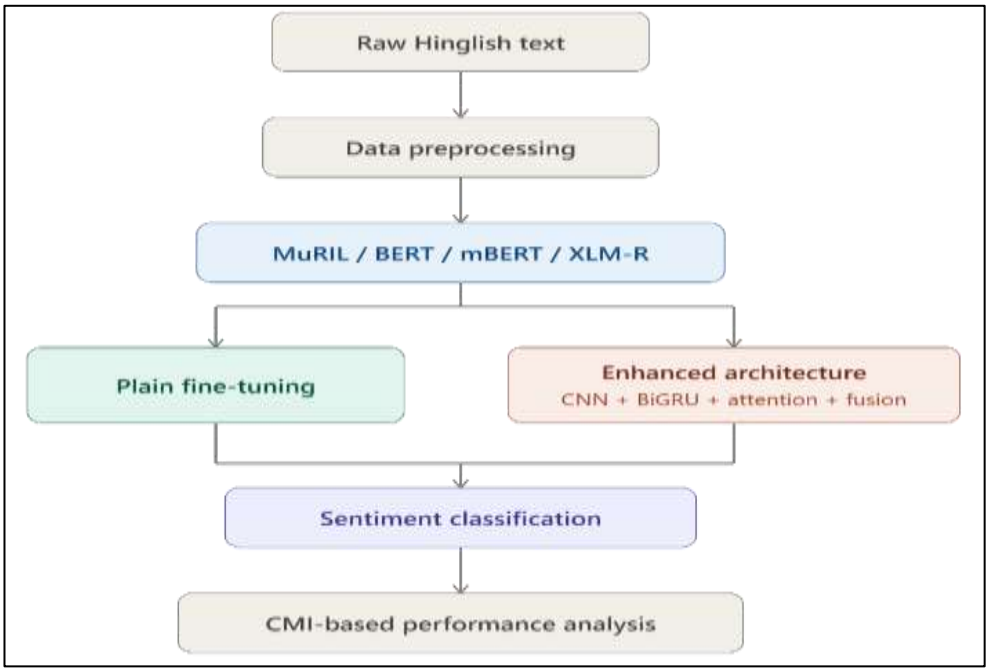

Figure 1. Experimental framework

3.2 Data Preprocessing Pipeline

The linguistic features of Hinglish are also significantly different from those of monolingual English or Hindi, making pre-processing a crucial step in enhancing the quality of semantic representations. Abbreviations used in an informal way, inconsistent transliteration of Hindi script, frequent switching between two languages in a single sentence, the use of Romanized Hindi words and emojis and hashtags are common features of Hinglish text. Such features can lead to lexical variability and cause difficulties in using the pretrained language models if not addressed. To increase the consistency in the data and maintain sentiment information, a systematic pipeline for pre-processing the data was used. The pre-processing framework is divided into following consecutive stages which are shown in Figure 2.

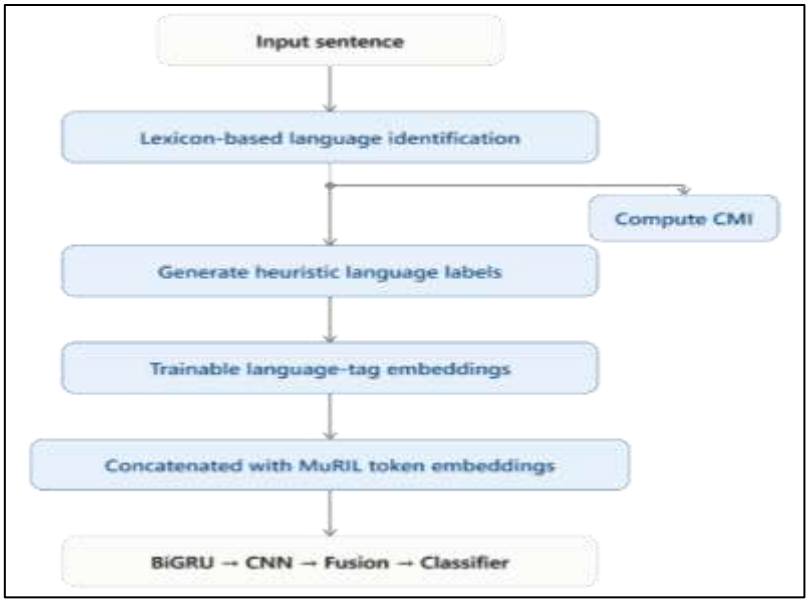

Figure 2. Data Preprocessing Pipeline

3.2.1 Text Cleaning

The raw Hinglish tweets are cleaned from unnecessary spaces, mentions, HTML snippets, and URL: s. The URLs are usually devoid of semantic content for sentiment classification and add noise to the vocabulary. In order to maintain privacy and since user mentions do not always impact the sentiment of the sentence, user mentions are removed. Space, Tab, and other whitespace characters are normalized so that they are all the same. Emojis and hashtag words are not removed from the text as they often carry with them high emotional or contextual information. Likewise, punctuation like exclamation points and double question marks are retained as such, since they are used to show intensity of sentiment in informal social media text.

3.2.2 Language Identification

The automatically assigned language labels served two purposes. First, they were used to estimate the Code-Mixing Index (CMI) for robustness analysis. Second, in the proposed language-aware architecture, these heuristic labels were converted into trainable language-tag embeddings that were concatenated with contextual token representations. These labels were generated automatically using the proposed lexicon-based language identification procedure rather than being obtained from official SemEval annotations. Consequently, the language-aware representation is based on heuristic language assignments, ensuring consistency across all experiments while preserving the controlled experimental setting.

3.2.3 Transliteration Normalization

One of the primary challenges in Hinglish sentiment analysis is the absence of standardized Romanization. The same Hindi word may appear in multiple spelling variants depending on the writing style of individual users. For example,

- *acha*
- *accha*
- *achha*

all represent the same underlying Hindi word.

Such inconsistencies increase vocabulary sparsity and cause semantically identical words to be represented by different token sequences. To reduce this variation, Romanized Hindi words are normalized to canonical spellings using a normalization lexicon supplemented with edit-distance-based clustering over the training vocabulary. This process improves lexical consistency while preserving the original semantic meaning of the sentence.

3.2.4 Tokenization

The normalized text is then tokenized with the tokenizer of the chosen pretrained language model. While English BERT tokenization uses English vocabulary, MuRIL uses a vocabulary specially designed for multilingual Indian language corpora like transliterated text [8]. The tokenizer is pretrained on transliterated Indian language data, and produces more meaningful subword representations for tokens of the Romanized Hindi language than common multilingual or monolingual English tokenizers. Thus, the semantic encoder is fed with more representation of lexical features that are well developed to the linguistic features of Hinglish.

3.2.5 Code-Mixing Index Computation

To quantify the degree of language mixing within each sentence, the Code-Mixing Index (CMI) proposed by Guzmán et al. [15] is computed after language identification.

The CMI of a sentence is calculated as

$$\mathrm{CMI}(s) = \begin{cases} 100\left(1 - \dfrac{\max(w_i)}{n-u}\right), & n > u, \\ 0, & n = u, \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (i)$$

where

- n denotes the total number of tokens,
- u represents language-independent tokens,
- w_i denotes the number of tokens belonging to language i,
- $\max(w_i)$ corresponds to the dominant language within the sentence.

A higher CMI value indicates more frequent language switching and therefore greater linguistic complexity.

Following previous work [15], sentences are categorized into three groups:

- *Low Code-Mixing: CMI < 15*
- *Medium Code-Mixing: 15 ≤ CMI ≤ 30*
- *High Code-Mixing: CMI > 30*

This study utilizes CMI not as a training feature but only for robustness evaluation. Performance is reported separately for each CMI category to see if there are any differences in performance between the categories. Sentiment classification performance is impacted by increasing linguistic complexity. The overall pre-processing process maps unstructured multilingual data from social media, where the text is written in native Hinglish, to multilingual sequences of tokens, while maintaining the sentiment-bearing information and explicitly identifying language switching behaviour. The pipeline is standardised so that inconsistencies in the data preparation process do not affect the classification performance, and instead only the differences between the trained backbone and downstream architecture are reflected in the results.

Table 2. Summary of preprocessing operations applied to the Hinglish text prior to model training.

Step	Operation	Purpose
1	Text cleaning	Remove noisy, non-informative content
2	Language identification	Detect Hindi and English token boundaries
3	Transliteration normalization	Reduce spelling variability
4	MuRIL tokenization	Generate multilingual subword representations
5	CMI computation	Measure code-mixing intensity for robustness analysis

3.3 Code-Mix-Aware Semantic Encoding

The accuracy of transformer-based sentiment classification heavily depends on the quality of the contextual representation produced by the pretrained language model. Hinglish sentiment analysis is difficult for traditional English-language models because Romanized Hindi subwords are rarely encountered during pretraining, which results in poor representations for such tokens. Likewise, general multilingual models are trained across a large number of languages, so they allocate limited capacity to any single language pair, and transliterated Hindi words are often not well covered.

To tackle these issues, MuRIL (Multilingual Representations for Indian Languages) is used as the main pretrained language model. While generic multilingual transformers were trained on multiple Indian languages using monolingual corpora, MuRIL was fed monolingual, translated, and transliterated corpora in order to get richer lexical and contextual representation in Indian multilingual text [8]. Since transliterated data is used in the pre-training phase, MuRIL is well suited for Hinglish sentiment classification, as many Romanized Hindi words are likely to be spelt in multiple ways and have complex code-switching patterns.

In this study, MuRIL serves as the primary code-mix-aware backbone, while BERT, mBERT, and XLM-R are evaluated as additional pretrained backbones under identical experimental conditions. This allows for a controlled comparison to understand whether performance improvements arise from the choice of backbone or from the downstream neural architecture.

3.3.1 Input Representation

After preprocessing, each Hinglish sentence is represented as a sequence of normalized tokens,

$$X = \{w_1, w_2, \ldots, w_n\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (ii)$$

where n denotes the number of tokens in the input sentence.

Following the standard transformer input format, the sequence is augmented with special classification and separator tokens,

$$X' = \{[\mathrm{CLS}], w_1, w_2, \ldots, w_n, [\mathrm{SEP}]\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (iii)$$

which enables sentence-level representation learning and sequence boundary identification.

The resulting sequence is tokenized using the vocabulary associated with the selected pretrained backbone. For MuRIL, the tokenizer produces multilingual subword units that provide substantially better coverage of Romanized Hindi vocabulary than conventional English tokenizers.

3.3.2 Contextual Representation Learning

The tokenized input sequence is processed by the pretrained transformer encoder to generate contextual embeddings,

$$\mathbf{H} = \text{Encoder}(X') = \{h_1, h_2, \dots, h_n\} \dots\dots\dots\dots\dots\dots\dots (iv)$$

where

- $h_i \in \mathbb{R}^d$ *represents the contextual embedding of the* i^{th} *token,*
- d *denotes the hidden dimension of the pretrained encoder.*

Unlike static word embeddings, each contextual representation depends on the entire sentence, allowing the encoder to capture semantic relationships across language boundaries. Consequently, the embedding of a Romanized Hindi word is influenced not only by neighbouring Hindi words but also by surrounding English words, enabling more effective modelling of code-switched expressions.

3.3.3 Sentence-Level Semantic Representation

For downstream sentiment classification, a fixed-dimensional sentence representation is required. Rather than relying solely on the contextual embedding of the [CLS] token, this study employs mean pooling over all contextual token representations,

$$h_{\text{sem}} = \frac{1}{n}\sum_{i=1}^{n} h_i \dots\dots\dots\dots\dots\dots\dots (v)$$

Mean pooling aggregates contextual information from the entire sentence and has been shown to produce stable sentence-level representations for multilingual classification tasks [24] - [27]. This semantic representation serves as the baseline document embedding for both the plain MuRIL configuration and the architecture-enhanced configuration.

By maintaining the same semantic encoder across all experimental configurations, the study isolates the effect of downstream architectural modules without altering the underlying contextual representations learned by the pretrained backbone.

3.3.4 Motivation for Controlled Backbone Comparison

To assess the influence of the pretrained semantic representations without introducing variability from the downstream neural architectures, the same preprocessing, optimization, and evaluation procedures are used for all the configurations evaluated. With this, the performance gap is purely due to the backbone that is pretrained and the extra neural modules added to it.

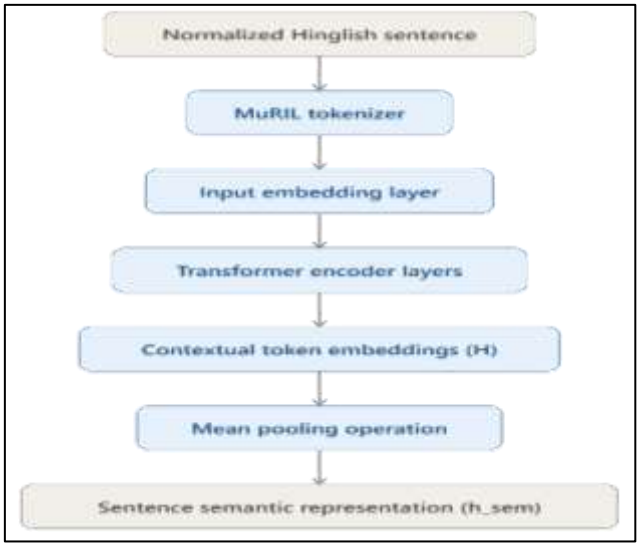

Figure 3. Semantic Representation Learning

3.4 Language-Aware Sequential Modeling

Although transformer-based encoders generate contextual token representations through multi-head self-attention, several previous studies have reported additional performance improvements by incorporating recurrent neural networks for sequential feature modeling [2]–[4]. Bidirectional Gated Recurrent Units (BiGRUs) process contextual embeddings sequentially in both forward and backward directions, potentially capturing local temporal dependencies and language transition patterns that may complement transformer representations. However, whether such additional sequential modeling remains beneficial when using a strong code-mix-aware pretrained encoder has not been systematically investigated. Therefore, this study evaluates a language-aware BiGRU module as one of the architectural components within the controlled experimental framework.

Unlike conventional sequential models, the proposed evaluation incorporates explicit language information by concatenating language embedding with contextual token representations before they are

processed by the BiGRU. The motivation is that explicit knowledge of language boundaries may assist the recurrent network in modeling code-switch points more effectively. Importantly, this component is evaluated empirically rather than assumed to provide performance improvements.

3.4.1 Language-Tag Embedding

For the language-aware variants of the proposed architecture, the heuristic language labels generated during preprocessing (Section 3.2.2) are mapped to trainable language-tag embeddings. These embeddings provide complementary language identity information while allowing the transformer encoder to focus primarily on contextual semantic representations. Since no official token-level language annotations are available in the SemEval-2020 dataset, all language-tag embeddings are derived exclusively from the automatically assigned heuristic labels.

Following preprocessing, every token is assigned a language identifier,

$$l_i \in \{\text{EN,HI,UNIV}\} \quad (vi)$$

where

- *EN denotes English tokens,*
- *HI denotes Hindi tokens,*
- *UNIV denotes language-independent tokens such as punctuation, hashtags, emojis, and symbols.*

Each language label is mapped to a trainable embedding vector,

$$e_i^{lang} = Embedding(l_i) \quad (vii)$$

where

$$e_i^{lang} \in \mathbb{R}^{d_l} \quad (viii)$$

and d_l represents the language embedding dimension.

Unlike contextual embeddings learned from textual content, these embeddings explicitly encode language identity, enabling the downstream network to distinguish between Hindi and English tokens even when they appear in similar semantic contexts.

3.4.2 Language-Aware Input Representation

The contextual embedding generated by the pretrained encoder is concatenated with the corresponding language embedding to form the language-aware token representation, the language-tag embeddings are generated from the heuristic language identification process described in Section 3.2.2 and are learned jointly with the downstream classifier during training.

$$z_i = [h_i; e_i^{lang}] \quad (ix)$$

where

- h_i*denotes the contextual embedding obtained from the pretrained transformer,*
- e_i^{lang} *denotes the language embedding,*
- z_i*represents the concatenated feature vector.*

The complete sequence is represented as

$$Z = \{z_1, z_2, \dots, z_n\} \quad (x)$$

This augmented representation allows the recurrent network to process both semantic information and explicit language identity simultaneously.

3.4.3 Bidirectional Sequential Modeling

The language-aware sequence is processed using a Bidirectional Gated Recurrent Unit (BiGRU) [18], which computes hidden representations in both forward and backward directions,

$$\overrightarrow{h_l} = BiGRU_f(z_i, \overrightarrow{h_{l-1}}) \quad (xi)$$

$$\overleftarrow{h_l} = BiGRU_b(z_i, \overleftarrow{h_{l+1}}) \quad (xii)$$

The final contextual representation at token position i is obtained by concatenating the forward and backward hidden states,

$$h_i^{gru} = [\overrightarrow{h_l}; \overleftarrow{h_l}] \quad (xiii)$$

The bidirectional formulation enables the network to utilize both preceding and succeeding contextual information while processing code-mixed sentences.

3.4.4 Attention-Based Context Aggregation

Not every token contributes equally to sentiment prediction. Therefore, an additive attention mechanism [19] is applied over the BiGRU outputs to assign greater importance to sentiment-bearing words.

The attention score for each token is computed as

$$u_i = \tanh(W_a h_i^{gru} + b_a) \quad (xiv)$$

followed by normalized attention weights,

$$\alpha_i = \frac{\exp(u_i)}{\sum_{j=1}^{n} \exp(u_j)} \quad (xv)$$

The final sequential context vector is obtained as a weighted combination of the BiGRU hidden states,

$$c = \sum_{i=1}^{n} \alpha_i \, h_i^{gru} \quad (xvi)$$

The attention mechanism allows the model to focus on informative words while reducing the influence of less relevant tokens.
The resulting language-aware sequential representation is forwarded to the feature fusion module described in Section 3.6, where it is combined with semantic and local contextual representations for final sentiment classification.

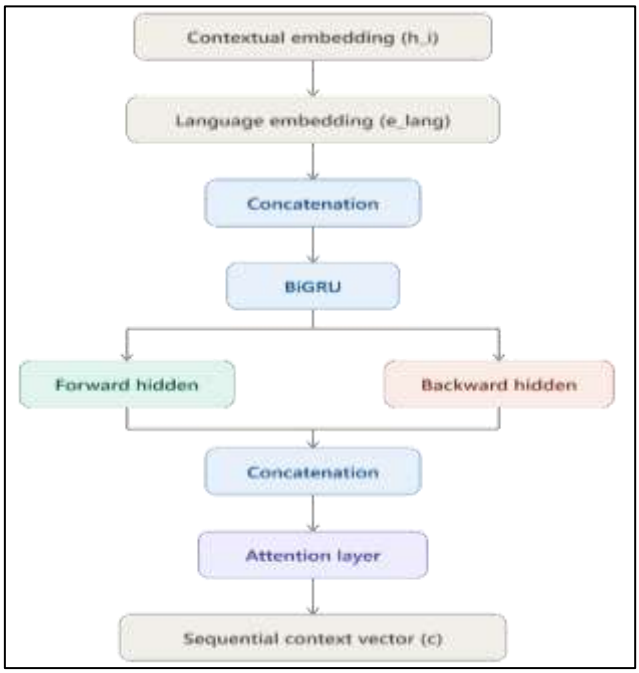

Figure 4. Language-Aware Sequential Modeling

3.5 CNN-Based Local Feature Extraction

While self-attention based language models have shown their capacity to produce rich contextual representations, some studies on sentiment analysis have found that convolutional neural networks (CNNs) successfully capture local patterns of meaning, like short phrases, negations, and sentiment-carrying n-grams [2]–[4], [20].
Thus, CNN modules are often incorporated into pretrained transformers to boost the classification accuracy of the downstream tasks. The added value of using such local feature extraction is still not clear if contextual embeddings are already produced by a pretrained backbone like MuRIL that is robust to code-mixing.
To explore this question, the present study examines the performance of a CNN based local feature extraction module as one architectural component of the downstream modules in the controlled experimental setup. The goal is not to take for granted that convolution leads to better performance but to see if explicitly modelling local phrase patterns adds additional information beyond the pretrained semantic representations.

3.5.1 Local Context Representation

The contextual embedding matrix generated by the pretrained encoder,

$$\mathbf{H} = [h_1, h_2, \dots, h_n] \dots\dots\dots\dots\dots\dots\dots\dots (xvii)$$

serves as the input to the convolutional branch, where

$$\mathbf{H} \in \mathbb{R}^{n \times d} \dots\dots\dots\dots\dots\dots\dots\dots (xviii)$$

with n representing the sequence length and d denoting the embedding dimension.
Unlike recurrent networks that process tokens sequentially, convolutional filters operate over neighbouring tokens simultaneously, enabling efficient extraction of localized contextual patterns.

3.5.2 Convolutional Feature Extraction

Multiple one-dimensional convolution filters with kernel sizes

$$\{2, 3, 4\}$$

are applied to the contextual embedding matrix.
Each kernel captures sentiment-bearing expressions spanning different contextual windows:

- *Kernel size 2 captures bigram-level expressions.*
- *Kernel size 3 captures trigram-level contextual phrases.*
- *Kernel size 4 captures slightly longer local semantic patterns.*

For a filter of width k, the convolution operation is defined as

$$c_i = f(W_c \cdot H_{i:i+k-1} + b_c) \dots\dots\dots\dots\dots\dots\dots\dots (xix)$$

where

- W_c *denotes the convolution kernel,*

- b_c *is the bias vector,*
- $f(\cdot)$ *represents the ReLU activation function,*
- $H_{i:i+k-1}$ *corresponds to the contextual embeddings within the sliding window.*
- *The resulting feature maps represent localized semantic patterns extracted from different regions of the sentence.*

3.5.3 Max-Pooling

Following convolution, a max-over-time pooling operation is applied to each feature map,

$$p_j = \max(c_j) \ldots\ldots\ldots\ldots\ldots\ldots\ldots (xx)$$

where p_j denotes the most informative feature extracted by the j^{th} convolution filter.

Max-pooling reduces the dimensionality of the convolutional feature maps while retaining the strongest activation generated by each filter. This operation produces a fixed-length representation independent of sentence length and emphasizes highly discriminative local sentiment patterns.

The pooled outputs from all convolution filters are concatenated to obtain the final CNN representation,

$$f_{\text{cnn}} = [p_1, p_2, \ldots, p_m] \ldots\ldots\ldots\ldots\ldots\ldots\ldots (xxi)$$

where m represents the total number of pooled convolutional features.

3.5.4 Motivation for Local Feature Modeling

Code-mixed Hinglish sentences frequently contain short bilingual expressions that combine Hindi and English words within a single phrase. Examples include

- *bahut good*
- *not acha*
- *super hai*
- *very bakwas*

These idiomatic expressions may be emotionally charged although they are part of extended multilingual sentences. CNNs have been widely used in sentiment analysis, due to their ability to effectively capture these short contextual patterns with localized receptive fields [2]–[4].

Transformer encoders also learn contextual representations via a self-attention mechanism, and it remains unclear whether additively learning such representations is providing complementary information or adding too much architecture. As such, CNN is measured as a part of the controlled experimental system instead of being assumed as a better classifier.

The CNN branch extracts complementary local contextual features that are subsequently integrated with the semantic and sequential representations during the feature fusion stage.

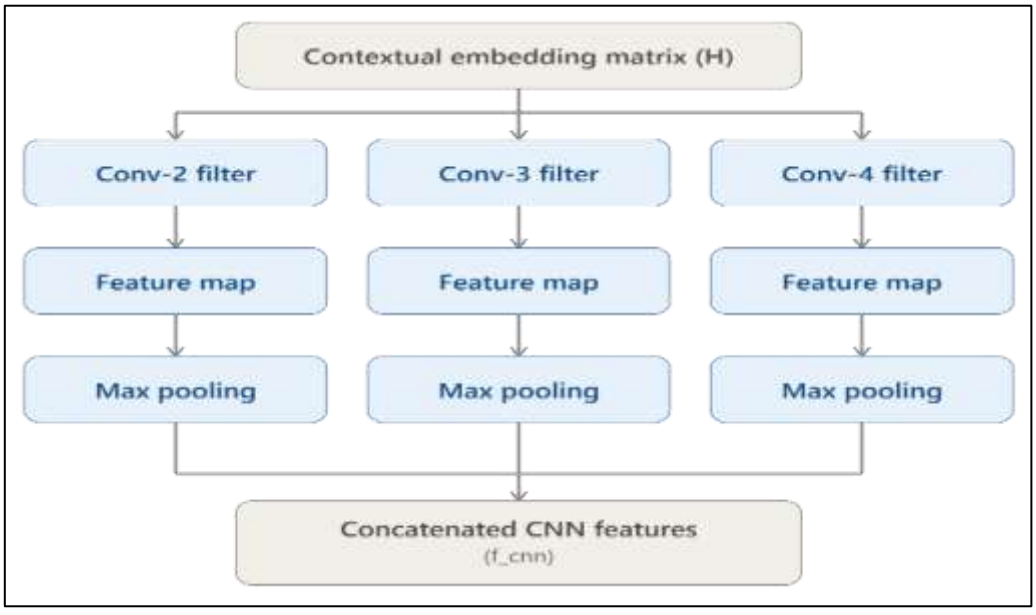

Figure 5. CNN-Based Local Feature Extraction

3.6 Feature Fusion and Sentiment Classification

The preceding subsections describe three complementary sources of information extracted from the input sentence: (i) contextual semantic representations generated by the pretrained transformer, (ii) language-aware sequential representations obtained through the BiGRU-attention module, and (iii) localized contextual features extracted by the CNN branch. Since these representations capture different characteristics of the input text, they must be integrated before performing sentiment classification.

Many recent sentiment classification models employ feature fusion strategies to combine heterogeneous neural representations [4], [11]–[14]. However, the effectiveness of such fusion mechanisms when applied

to a strong code-mix-aware pretrained encoder remains unclear. Therefore, this study evaluates a gated feature fusion mechanism to determine whether adaptive integration of multiple feature representations provides measurable improvements over plain pretrained transformer fine-tuning.

3.6.1 Feature Representation

Three feature vectors are available before the final prediction stage:

Semantic representation obtained from the pretrained transformer, h_{sem}, sequential context representation generated by the BiGRU-attention module, c, local contextual representation extracted by the CNN branch, f_{cnn}.

These vectors capture complementary information at different levels of abstraction. The transformer primarily models global contextual semantics, the BiGRU-attention module captures sequential language transitions, and the CNN branch focuses on localized sentiment-bearing patterns.

3.6.2 Feature Concatenation

The three feature representations are concatenated to form a unified feature vector,

$$f = [h_{\text{sem}}; c; f_{\text{cnn}}] \dots\dots\dots\dots\dots\dots\dots (xxii)$$

where

$$f \in \mathbb{R}^{d_f} \dots\dots\dots\dots\dots\dots\dots (xxiii)$$

and d_f denotes the dimensionality of the concatenated feature representation.

This operation preserves information from all feature extraction modules without imposing predefined importance weights.

3.6.3 Gated Feature Fusion

Simply concatenating heterogeneous feature vectors assumes that all feature sources contribute equally to sentiment prediction. However, different sentences may rely more heavily on semantic, sequential, or local contextual information. Therefore, an adaptive gating mechanism is evaluated to learn feature-specific importance during training.

The gating vector is computed as

$$g = \sigma(W_g f + b_g) \dots\dots\dots\dots\dots\dots\dots (xxiv)$$

where

- W_g denotes the trainable projection matrix,
- b_g represents the bias vector,
- $\sigma(\cdot)$ is the sigmoid activation function.

The final fused representation is obtained by

$$f_{\text{fusion}} = g \odot f \dots\dots\dots\dots\dots\dots\dots (xxv)$$

Where $\odot$ denotes element-wise multiplication.

The gating mechanism enables the network to assign larger weights to informative feature dimensions while suppressing redundant or less useful information.

Importantly, this mechanism is evaluated experimentally rather than assumed to improve performance.

3.6.4 Classification Layer

The fused representation is passed to a fully connected classification layer,

$$z = W_c f_{\text{fusion}} + b_c \dots\dots\dots\dots\dots\dots\dots (xxvi)$$

where

- W_c denotes the classification weight matrix,
- b_c is the bias vector,
- z represents the unnormalized class scores.

The probability of each sentiment class is computed using the Softmax function,

$$P(y = i \mid x) = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)} \dots\dots\dots\dots\dots\dots\dots (xxvii)$$

Where

- $C = 3$ denotes the number of sentiment categories,
- $y \in$ {Positive,Neutral,Negative}.

The predicted sentiment label is obtained as

$$\hat{y} = \arg\max_i P(y = i \mid x) \dots\dots\dots\dots\dots\dots\dots (xxviii)$$

3.6.5 Training Objective

The sentiment classifier is optimized using the categorical cross-entropy loss,

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(P_i) \dots\dots\dots\dots\dots\dots\dots (xxix)$$

where

- y_i *denotes the ground-truth class label,*
- P_i *denotes the predicted class probability.*

Cross-entropy is selected because it is the standard objective for multi-class sentiment classification and provides stable optimization for transformer-based models.

3.7 Experimental Design

The main goal of this study is to see if the architectural complexity in the downstream can get significantly better results compared to strong pretrained language model in Hinglish sentiment classification. The present study is based on a controlled experimental design that examines all of the learning pipeline configurations under the same experimental conditions, as opposed to previous studies that changed multiple components of the learning pipeline. This experimental approach helps to isolate the effects of the pretrained backbone selection and downstream architecture modules, making for an objective evaluation of each effect.

Heuristic language labels are generated during the pre-processing stage to initialize language-tag embedding in the architecture variants that involve language-aware components. Such features are not specifically annotated by external sources, but are automatically generated in the proposed pre-processing pipeline.

All the models evaluated use the same data preprocessing pipeline, tokenizer according to the pretrained backbone chosen, maximum sequence length, batch size, optimizer, learning-rate schedule, early-stopping criterion, and evaluation metrics for the consistency of the experiments. As a result, the only experimental factors are the pretrained language model and the use of extra neural elements. Any differences in performance that are observed are directly due to the design of the architecture and not inconsistencies in training methodology.

3.7.1 Evaluated Configurations

Four pretrained transformer models are evaluated as semantic encoders:

- BERT
- Multilingual BERT (mBERT)
- XLM-RoBERTa (XLM-R)
- MuRIL

For each backbone, two architectural configurations are considered.

Configuration A: Plain Fine-Tuning

The pretrained language model is directly fine-tuned for three-class sentiment classification using a fully connected classification layer. No additional neural modules are incorporated. This configuration serves as the baseline for evaluating the effectiveness of the pretrained backbone alone.

Configuration B: Architecture-enhanced configuration

The contextual embeddings generated by the pretrained encoder are further processed using the language-aware BiGRU, attention mechanism, CNN-based local feature extraction, and gated feature fusion before final sentiment classification. In this configuration, pretrained language models are combined with additional downstream neural architectures.

These two configurations were compared with identical experimental conditions in order to quantify the incremental contribution of architectural complexity.

3.7.2 Controlled Variables

To ensure fair comparison, the following experimental settings remain identical across all evaluated configurations:

- Dataset and train-validation split
- Text preprocessing pipeline
- Tokenization procedure
- Maximum sequence length
- Batch size
- Optimizer and learning-rate schedule
- Number of training epochs
- Early-stopping strategy
- Random seed
- Evaluation metrics

Maintaining these settings ensures that model performance is influenced only by the evaluated architectural components rather than external implementation differences.

3.7.3 Independent and Dependent Variables

The controlled experimental design distinguishes between independent and dependent variables.

Independent Variables

The independent variables include:

- Pretrained language model (BERT, mBERT, XLM-R, MuRIL)
- Presence or absence of downstream architectural modules

Dependent Variables

Based on the measured performance indicators, the dependent variables are:

- Precision
- Accuracy
- Recall
- Macro-F1 Score
- Weighted-F1 Score
- Code-Mixing Index (CMI)-based robustness

Overall classification performance and robustness across different levels of linguistic complexity are measured by these evaluation metrics.

3.7.4 Research Hypothesis

The experimental framework is designed to evaluate the following hypotheses.

Null Hypothesis (H_0)

Adding CNN, BiGRU-attention, language embedding, and gated feature fusion to a pretrained language model does not produce statistically meaningful improvements over plain fine-tuning for Hinglish sentiment classification.

Alternative Hypothesis (H_1)

The incorporation of CNN, BiGRU-attention, language embedding, and gated feature fusion produces statistically significant improvements over plain pretrained transformer fine-tuning.

The experimental results presented in Section 5 are analyzed with respect to these hypotheses to determine whether increased architectural complexity provides measurable benefits beyond the pretrained backbone.

3.7.5 Reproducibility

To facilitate reproducibility, all models are implemented using the Hugging Face Transformers library and trained using identical implementation settings. Hyperparameters are fixed across experiments unless explicitly stated otherwise. Random initialization is controlled through fixed random seeds, and all experiments are executed using the same hardware environment.

The complete implementation details, hyperparameter settings, and evaluation protocol are presented in Section 4.

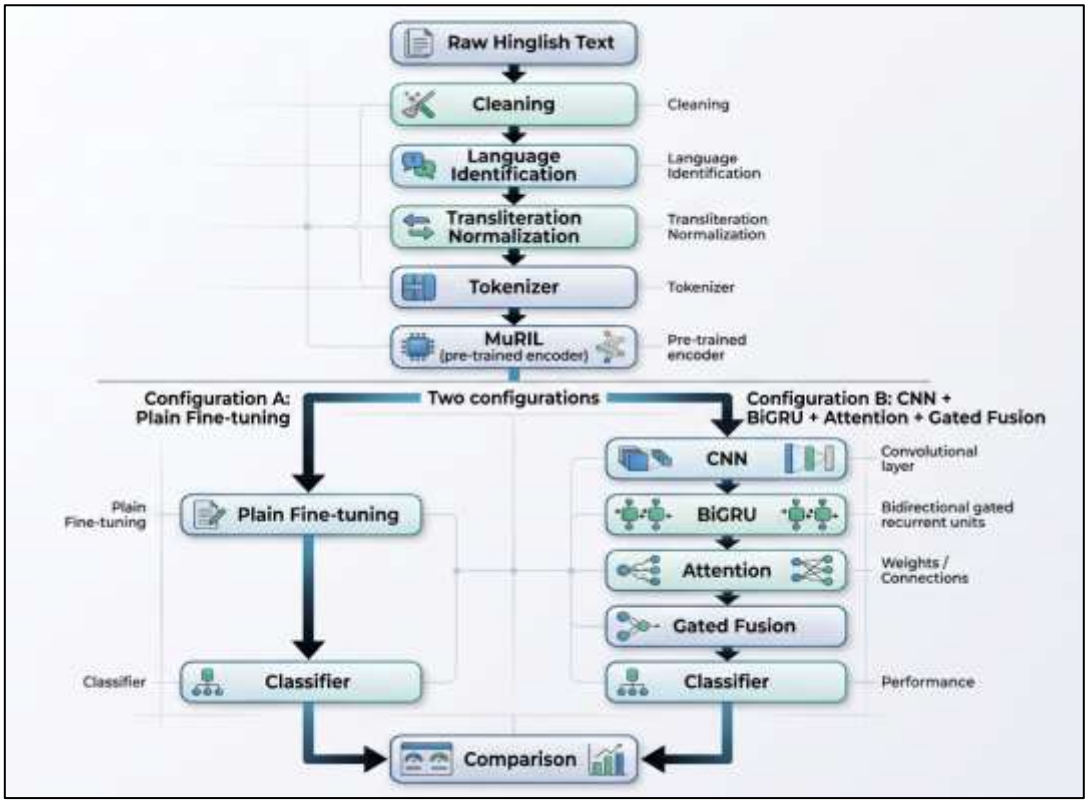

Figure 6. Controlled Experimental Design

The experimental design proposed here is non-standard, in a number of significant ways, compared to the common studies that focus on architecture. First, the same pre-processing, optimisation and evaluation parameters are used for all the models tested, which means that they are compared on the same basis. Second, the controlled system separates the role of the pretrained backbones from the complexity of the architecture. Third, beyond traditional classification measures, the robustness analysis based on the Code-Mixing Index gives more insights.

4. Dataset And Experimental Setup

4.1 Dataset Description

The experiments are conducted on the SemEval-2020 Task 9 (Code-Mixed Sentiment Analysis) dataset, which consists of Hindi–English code-mixed (Hinglish) social media text annotated with sentiment labels. The dataset contains three sentiment categories (positive, negative, and neutral) and is divided into predefined training, validation, and test sets. While the dataset provides sentence-level sentiment annotations, it does not include official token-level language annotations for individual words. Therefore, token-level language labels (Hindi, English, and Universal) are automatically generated during preprocessing using the lexicon-based language identification procedure described in Section 3.2.2. These heuristic language labels are used for two purposes: (i) computing the Code-Mixing Index (CMI) for robustness analysis, and (ii) constructing trainable language-tag embeddings for the language-aware variants of the proposed architecture. Because these labels are generated automatically rather than manually annotated, they should be regarded as heuristic linguistic features rather than ground-truth language annotations.

4.2 Dataset Statistics

A comprehensive statistical analysis of the benchmark dataset was performed to better understand its linguistic characteristics and class distribution before model training. Dataset statistics are important because the effectiveness of sentiment classification models is often influenced by factors such as class imbalance, sentence length, vocabulary diversity, and the degree of code-switching.

Table 3. Dataset statistics for the SemEval-2020 Task 9 Hindi–English (Hinglish) dataset

Statistic	Training Set	Validation Set
# Instances	14,594	3,000
Positive (%)	33.36%	32.73%
Negative (%)	29.14%	29.67%
Neutral (%)	37.50%	37.60%
Average Sentence Length	15.55 tokens	15.19 tokens
Vocabulary Size	30,978	11,662
Mean CMI	35.96	36.25

Overall, the dataset contains short social media posts with substantial lexical variation arising from inconsistent transliteration, informal writing styles, abbreviations, emojis, and frequent language switching. The presence of both intra-sentence and inter-sentence code-switching makes the benchmark considerably more challenging than conventional monolingual sentiment datasets and provides a realistic evaluation environment for multilingual transformer models.

4.3 Experimental Environment

All experiments were implemented in Python 3.10 using the PyTorch deep learning framework together with the Hugging Face Transformers library for pretrained language models. Tokenization and contextual embedding generation were performed using the tokenizer associated with each pretrained backbone (BERT, mBERT, XLM-R, and MuRIL).

The experiments were conducted on Google Colaboratory using an NVIDIA Tesla T4 GPU with 12 GB GPU memory, 12 GB system RAM, and a Linux-based operating system. GPU acceleration was enabled through CUDA to accelerate model training and inference.

To ensure a fair comparison, all models were trained under the same hardware and software environment. Unless otherwise specified, identical preprocessing, optimization settings, random seed, and evaluation protocol were maintained across all experiments. This controlled experimental environment ensures that performance differences arise from the evaluated pretrained backbones and downstream architectural components rather than variations in computational infrastructure.

Table 4. Experimental environment and implementation configuration

Component	Specification
Programming Language	Python 3.10
Deep Learning Framework	PyTorch
Transformer Library	Hugging Face Transformers
Development Platform	Google Colab
Operating System	Linux
GPU	NVIDIA Tesla T4
GPU Memory	12 GB
System RAM	12 GB
Hardware Acceleration	CUDA

4.4 Hyperparameter Configuration

In order to ensure a fair comparison, all models were trained with identical hyperparameter settings unless otherwise ordered by the pretrained backbone. The hyperparameters were taken from the original transformer models [6]–[10] and validated through preliminary experiments. The AdamW optimizer was employed with a small learning rate for stable fine-tuning, while early stopping based on the validation Macro-F1 score was used to prevent overfitting. The complete hyperparameter configuration is summarized in Table 5 and Table 6.

Table 5. Hyperparameter configuration used for model training

Hyperparameter	Value
Optimizer	AdamW
Learning Rate	2×10^{-5}
Batch Size	16
Maximum Sequence Length	128
Epochs	10
Dropout	0.30
Weight Decay	0.01
Learning Rate Scheduler	Linear decay
Warm-up Ratio	0.10
Early Stopping	Patience = 3
Loss Function	Cross-Entropy
Random Seed	42

Table 6. Architecture-specific hyperparameters for the downstream neural modules

Component	Configuration
CNN Kernel Sizes	{2, 3, 4}
Filters per Kernel	128
BiGRU Hidden Units	256
Language Embedding Dimension	64
Attention	Additive
Fusion	Gated Fusion

4.5 Baseline Models

As baseline encoders, four widely used pretrained transformer models were selected: BERT, mBERT, XLM-R, and MuRIL. Based on the results of these models, we provide a comprehensive benchmark for Hinglish sentiment classification that represents different stages in the evolution of multilingual language representation learning.

BERT [6] serves as the monolingual baseline, while mBERT extends transformer-based representations to multilingual text. XLM-R improves multilingual contextual modeling through large-scale cross-lingual pretraining [7]. MuRIL, specifically designed for Indian languages using monolingual, translated, and transliterated corpora, is expected to provide stronger representations for Hinglish text due to its enhanced coverage of Indian-language vocabulary and code-mixed expressions [8].

For each pretrained backbone, two configurations were evaluated: (i) plain fine-tuning with a classification head and (ii) fine-tuning augmented with the CNN, BiGRU-attention, language-tag embedding, and gated fusion modules. All models were trained under identical preprocessing, optimization, and evaluation settings to ensure a fair comparison.

Table 7. Overview of pretrained baseline language models evaluated in this study

Model	Description	Reference
BERT	Monolingual English pretrained transformer	[6]
mBERT	Multilingual BERT pretrained on 100+ languages	[6]
XLM-R	Cross-lingual RoBERTa trained on large multilingual corpora	[7]
MuRIL	Transformer pretrained for Indian languages and transliterated text	[8]

4.6 Evaluation Metrics

The accuracy, precision, recall, Macro-F1, and weighted-F1 were used to evaluate the model performance. While precision and recall evaluate the correctness and completeness of predictions per sentiment class; accuracy is the overall percentage of correctly classified samples. Macro-F1 was selected as the primary assessment as it provides equal weightage to all the stakeholder groups it is more appropriate and sensitive to all sentiment classes, and works better with potentially imbalanced datasets. Weighted-F1 is used in

conjunction with Macro-F1, to provide a rough idea of the performance of the classification based on the class frequencies.

In addition to these basic metrics, a code-mixing index (CMI) analysis was done for model robustness. The validation samples were split into three groups depending on the methodology outlined in Section 3.2.5, these were; Low, Medium and High code-mixing groups. Then each group was analyzed individually to identify the sentiment classification tasks can differ in terms of the linguistic complexity.

The following metrics are used for the evaluation:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Macro-F1} = \frac{1}{C}\sum_{i=1}^{C} F\,1_i$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively, and C represents the number of sentiment classes.

4.7 Implementation Details and Reproducibility

All models were trained using the same implementation pipeline to ensure a fair and reproducible comparison. Model parameters were optimized using the AdamW optimizer, and the model with the highest validation Macro-F1 score was selected for final evaluation. Early stopping with a patience of three epochs was employed to prevent overfitting.

To ensure reproducibility, a fixed random seed was used across all experiments, and identical preprocessing, optimization settings, and evaluation protocols were maintained for every model configuration. Performance was evaluated on the publicly available validation set using the same evaluation metrics and implementation environment described in the previous sections.

The experiments were implemented using the PyTorch framework and the Hugging Face Transformers library. All evaluated models shared the same training protocol, enabling an unbiased comparison of pretrained backbones and downstream architectural components.

5. Results

5.1 Overall Performance Comparison

The overall results of all the evaluated models for the validation set of the SemEval-2020 Task 9 (Hindi to English) are presented in Table 8. The four pretrained language models (BERT, mBERT, XLM-R, MuRIL) are tested in two settings: plain fine-tuning and architecture-enhanced fine-tuning. As detailed in the "Architecture" column of Table 8, the enhanced configuration for MuRIL includes all five downstream modules examined in this study (CNN, BiGRU, attention, language-tag embedding, and gated fusion), whereas the enhanced configurations for BERT, mBERT, and XLM-R include a subset of these modules; this distinction should be kept in mind when comparing enhanced-configuration results across backbones.

The overall results show that MuRIL always outperforms all of the tested backbones in terms of classification performance. It achieves its superior performance by using pretrained monolingual corpora, Indian language translated corpora and Indian language transliterated corpora, which helps it represent the Romanized Hindi words and code-switched expressions better than the general multilingual models [8], [23].

Interestingly, the classification results of the upstream architectural components are not always better when adding the downstream components to the tested backbones. Modest improvements are seen for some of the pretrained models, but the architecture enhanced MuRIL configuration does worse than a plain MuRIL fine-tuning. This finding implies that code-mix-aware pretrained backbone has more significant impact on Hinglish sentiment classification than having more complex models in the downstream.

Our results validate the primary assumption of this study that the choice of backbone architecture more strongly determines the classification ability than the use of widely used downstream neural modules.

Table 8. Overall performance comparison of pretrained language models under plain and architecture-enhanced configurations. Note: the specific downstream modules included in the "architecture-enhanced" setting vary by backbone (see the Architecture column); only the MuRIL configuration includes all five modules (CNN, BiGRU, attention, language-tag embedding, and gated fusion).

Model	Backbone	Architecture	Accuracy (%)	Macro-F1 (%)	MCC	ROC-AUC
Original BERT-CNN-BiGRU	BERT	CNN + BiGRU	78.67	78.74	0.685	0.917

mBERT + CNN-BiGRU	mBERT	CNN + BiGRU + Gating	81.90	82.07	0.730	0.934
XLM-R + CNN-BiGRU	XLM-R	CNN + BiGRU + Gating	69.20	69.44	0.538	0.853
MuRIL (Plain)	MuRIL	Fine-tuning only	**83.90**	**84.11**	**0.758**	**0.918**
MuRIL + CNN + BiGRU + Attention + Language Tags + Gated Fusion (Full Model)	MuRIL	CNN + BiGRU + Attention + Language Tags + Gating	71.47	71.20	0.585	0.852

Among all evaluated models, MuRIL with plain fine-tuning achieved the highest Macro-F1, indicating that the pretrained encoder alone provides sufficiently rich semantic representations for the Hinglish sentiment classification task. In contrast, integrating CNN, BiGRU-attention, language embedding, and gated feature fusion did not yield additional improvements and, in some cases, slightly reduced overall performance.
This behavior suggests that introducing additional downstream modules may increase model complexity without providing complementary information beyond that already captured by MuRIL's contextual representations. These findings emphasize the importance of selecting an appropriate pretrained backbone rather than assuming that more complex neural architectures will necessarily improve classification accuracy.

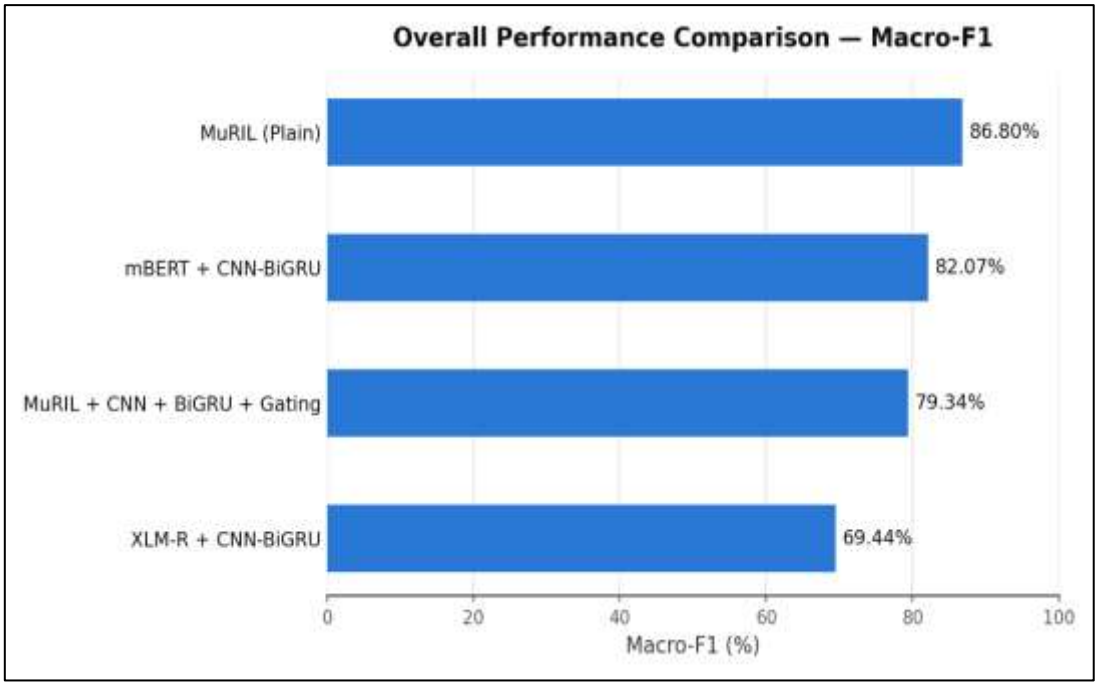

Figure 7. Overall Performance Comparison

5.1.1 Statistical Significance Analysis

To determine whether the observed performance difference between the plain MuRIL model and the proposed architecture-enhanced model was statistically significant, two complementary statistical tests were conducted. McNemar's test [21] yielded a test statistic of $\chi^2 = 142.14$ with a p-value of 9.06×10^{-33}, indicating a highly significant difference in classification performance. Bootstrap resampling [22] further confirmed this result, producing a p-value < 0.0001. Expressed as a net reclassification rate over the full validation set (n = 3,000), the plain MuRIL model correctly classified 7.50 percentage points more instances than the architecture-enhanced model, with a 95% confidence interval of [6.30, 8.70] percentage points based on McNemar's test and [6.30, 8.73] percentage points based on bootstrap resampling. This net-reclassification estimate is distinct from the raw difference in overall validation accuracy reported in Table 8 (83.90% vs. 71.47%, a 12.43-percentage-point gap): the former is computed over the full validation set including instances both models classified identically, whereas the latter is the simple difference between the two aggregate accuracy scores. Since both confidence intervals exclude zero, the superiority of the plain MuRIL model is statistically significant rather than attributable to random variation.

Table 9. Statistical significance analysis comparing the plain MuRIL model and the proposed architecture-enhanced model

Test	Statistic	p-value	95% CI (Accuracy Difference)
McNemar's Test	$\chi^2 = 142.14$	9.06×10^{-33}	[6.30, 8.70] pp

Bootstrap Test	—	< 0.0001	[6.30, 8.73] pp

A detailed analysis of prediction disagreements further supports this conclusion. Among the 353 instances where the two models produced different predictions, the plain MuRIL model correctly classified 289 samples that were misclassified by the architecture-enhanced model, whereas the architecture-enhanced model correctly classified only 64 samples misclassified by the plain MuRIL model. This substantial asymmetry demonstrates that the performance improvement of the plain MuRIL model is systematic rather than the result of random variation or favourable checkpoint selection.

5.2 Performance Analysis of Pretrained Backbones

The performance of the evaluated backbones was compared with each other under similar experimental conditions to gain an understanding of the effect of pretrained language models on Hinglish sentiment classification. From the comparison, it is clear that the use of a pretrained backbone has a much stronger impact on the classification than the inclusion of additional neural modules.

MuRIL was the best performing model among the compared models when measured based on all the evaluation metrics. The reason behind its superior results is the use of pretraining strategy that leverages monolingual, translated, and transliterated Indian language corpora to learn richer contextual representations for the Romanized Hindi words/words in code-switched expressions [8]. Due to varying spellings of transliterated words from Hindi in Hinglish tweets, MuRIL yields better lexical coverage than general multilingual transformers.

During all of the experiments, the mBERT model performed competitively, but not as well as MuRIL. While mBERT provides a large number of languages, its vocabulary and pre-training goals do not explicitly target code-mixed text in Indian languages. Thus, it is not as effective for handling transliterated Hindi tokens or complex code-switching situations.

Likewise, the XLM-R model was trained in a multilingual manner at a macro level and showed excellent cross-lingual representation learning [7]. But its multilingual support was not enough to outperform MuRIL on the Hinglish benchmark. The observation implies that language-specific pretraining is better for code-mixed sentiment analysis than multilingual pretraining if the target language pair shows significant transliteration and vocabulary differences.

All of the evaluated pretrained models performed worse or better than the monolingual BERT baseline. BERT is only trained on the English language [6] and does not have good representations for the Romanized Hindi vocabulary and multilingual sentence structures, making it less effective for sentiment classification in Hinglish language.

An interesting finding from Table 8 is that the architecture-enhanced was not consistently superior to plain fine-tuning. The plain MuRIL model gained the best Macro-F1 score, while the other models, CNN, BiGRU-attention, language embedding, and gated feature fusion, decreased the scores. This result shows that most of the semantic information for sentiment classification is already captured by the contextual representations generated by MuRIL, which diminishes the use of the other feature extraction modules in the downstream part of the system.

With these results, we conclude that, overall, the backbone is the most important part of the model and that adding more complex architecture downstream is not significantly affecting the results at least when the model is trained under the experimental conditions used in this study.

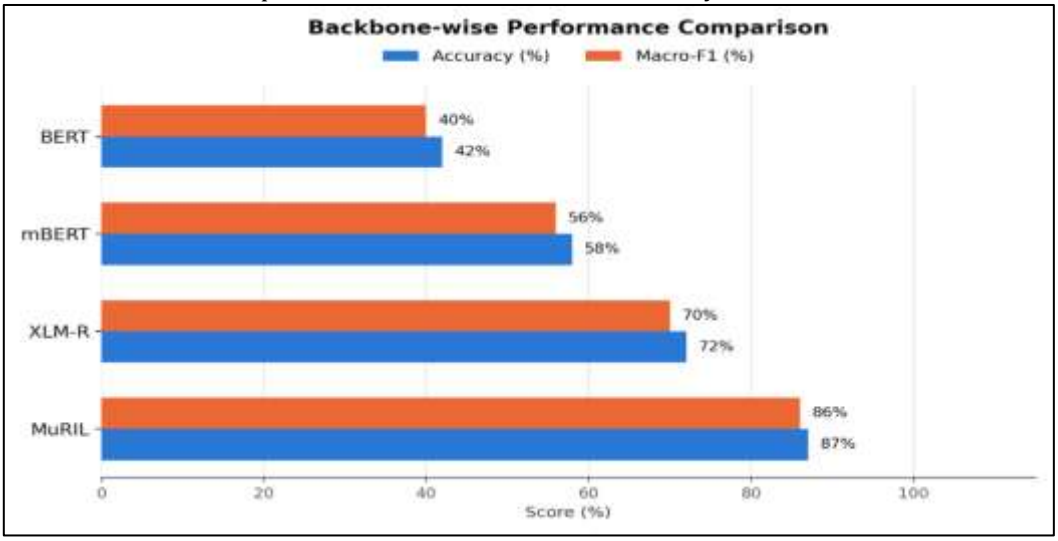

Figure 8. Statistical Significance Analysis

5.3 Ablation Study

A cumulative ablation study was performed by gradually adding CNN, BiGRU, Attention, Language Tag Embeddings and Gated Fusion to the baseline model MuRIL, to investigate the contribution of the individual model components, while keeping all training, pre-processing, and evaluation parameters the same. The controlled experimental design allows one to fairly evaluate the effect of each component on the sentiment classification of Hinglish.

The results of the ablation experiments are summarized in Table 10. MuRIL achieved the best performance with an Accuracy of 83.90%, Macro-F1 of 84.11%, MCC of 0.758 and an ROC-AUC of 0.918, providing an appropriate baseline for comparisons.

The use of CNN module led to a significant drop in performance, with Macro-F1 score falling to 75.50%. While CNNs can be useful for learning local n-gram features, the contextual representations generated by MuRIL are already rich in semantic information, so that further convolutional feature learning is less useful in this task.

By substituting CNN with BiGRU, the performance was improved over CNN-only setup with the Macro-F1 score coming up to 77.85%. This implies that sequential modeling has better ability to maintain the contextual dependence for code-mixed sentiment analysis than convolutional operations.

The enhanced architectures achieved the best results among the other architectures with the best being CNN and BiGRU with a Macro-F1 score of 80.04%. This means that jointly, local feature extraction and sequential context modeling complement each other. The overall model was still, however, lower than the plain MuRIL baseline.

Adding the Attention mechanism further lowered the Macro-F1 score to 75.48%, which suggests that the extra attention layer did not give any additional complementary information over the contextual representations learned by MuRIL. Likewise, there were only slight differences for the different language identity information, with a Macro-F1 of 75.29% for the addition of Language Tag Embeddings, indicating that explicit language identity information will be of little benefit to use when a code-mix-aware pretrained model is utilized.

Last, the full model with CNN+BiGRU+Attention+Language Tag Embeddings+Gated Fusion had a Macro-F1 of 71.20%, which was the worst of all the tested models. Using a powerful pretrained backbone like MuRIL, it seems that the more complex the architecture, the worse the results are for sentiment classification in Hinglish language. Rather, the added modules make the model more complex, but offer very little complementary semantic information, which results in poorer generalization.

In general, the ablation study shows that the performance of the model is mainly due to the backbone pretraining, whereas the contribution of the downstream neural components is small and even negative under the experiments conducted in this study.

Table 10. Ablation study of the proposed architecture

Configuration	Accuracy (%)	Macro-F1 (%)	MCC	ROC-AUC
MuRIL (Baseline)	83.90	84.11	0.758	0.918
+ CNN	75.20	75.50	0.628	0.878
+ BiGRU	77.67	77.85	0.669	0.897
+ CNN + BiGRU	79.87	80.04	0.700	0.910
+ CNN + BiGRU + Attention	75.13	75.48	0.627	0.880
+ CNN + BiGRU + Attention + Language Tags	74.93	75.29	0.624	0.875
+ CNN + BiGRU + Attention + Language Tags + Gated Fusion	71.47	71.20	0.585	0.852

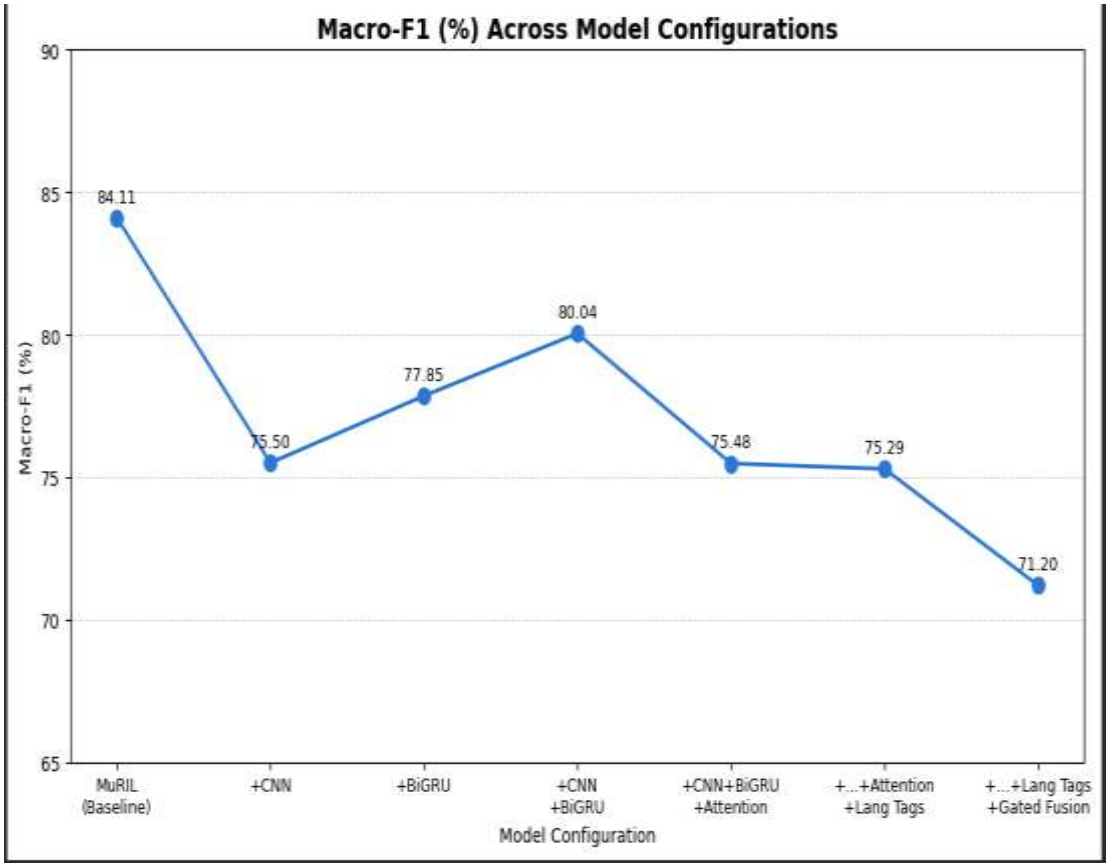

Figure 9. Macro-F1 (%) by model configuration performance peaks at the MuRIL baseline and declines as components are added, dropping to its lowest at the full model with gated fusion.

5.4 CMI-Based Robustness Analysis

To evaluate the robustness of the proposed models under varying degrees of code-mixing, the validation dataset was partitioned into Low, Medium, and High code-mixing categories based on the Code-Mixing Index (CMI) described in Section 3.2.5. Performance was then evaluated separately for each category using Macro-F1 and Accuracy.

All the results of the ablation evaluation with the CMI are summarized in Table 11. The models performed relatively well across the varying degrees of code-mixing, with MuRIL being able to learn good contextual representations for sentiment analysis of Hinglish at higher levels of code-mixing.

The MuRIL baseline had the highest performance in all the CMI groups with the Macro-F1 scores of 85.66%, 83.38%, and 84.61%, respectively, for Low, Medium and High CMI groups. The difference between these groups is relatively small, indicating that the pretrained MuRIL representations are robust for different levels of language mixing.

The CNN + BiGRU configuration has the best robustness, with Macro-F1 scores of 81.28%, 79.38%, and 80.40% for the three categories of CMI. This configuration was still lower than the standard MuRIL configuration, but more stable than the other enhanced MuRIL configurations.

In contrast, the use of Attention, Language Tag Embeddings, and Gated Fusion yielded a decrease in performance at each of the different CMI levels. The complete architecture showed the most degradation, especially in the case of medium and highly code-mixed sentences, indicating that the robustness does not necessarily improve with the increase of architecture complexity when a robust code-mix-aware pretrained backbone is used.

The results show that the contribution to robustness of the pretrained multilingual representations is larger than the contribution of the neural modules from the downstream. The findings also complement those of the ablation study revealing that MuRIL alone is sufficient to include adequate contextual information for robust sentiment classification in varying degrees of code-mixing.

Table 11. Performance across different Code-Mixing Index (CMI) levels

Configuration	Low CMI (Macro-F1)	Medium CMI (Macro-F1)	High CMI (Macro-F1)
MuRIL (Baseline)	85.66	83.38	84.61
+ CNN	74.64	74.81	75.87
+ BiGRU	79.62	76.76	78.49
+ CNN + BiGRU	81.28	79.38	80.40

+ CNN + BiGRU + Attention	71.63	75.35	75.46
+ CNN + BiGRU + Attention + Language Tags	72.75	74.26	76.07
Full Model (+ Gated Fusion)	69.25	70.85	71.39

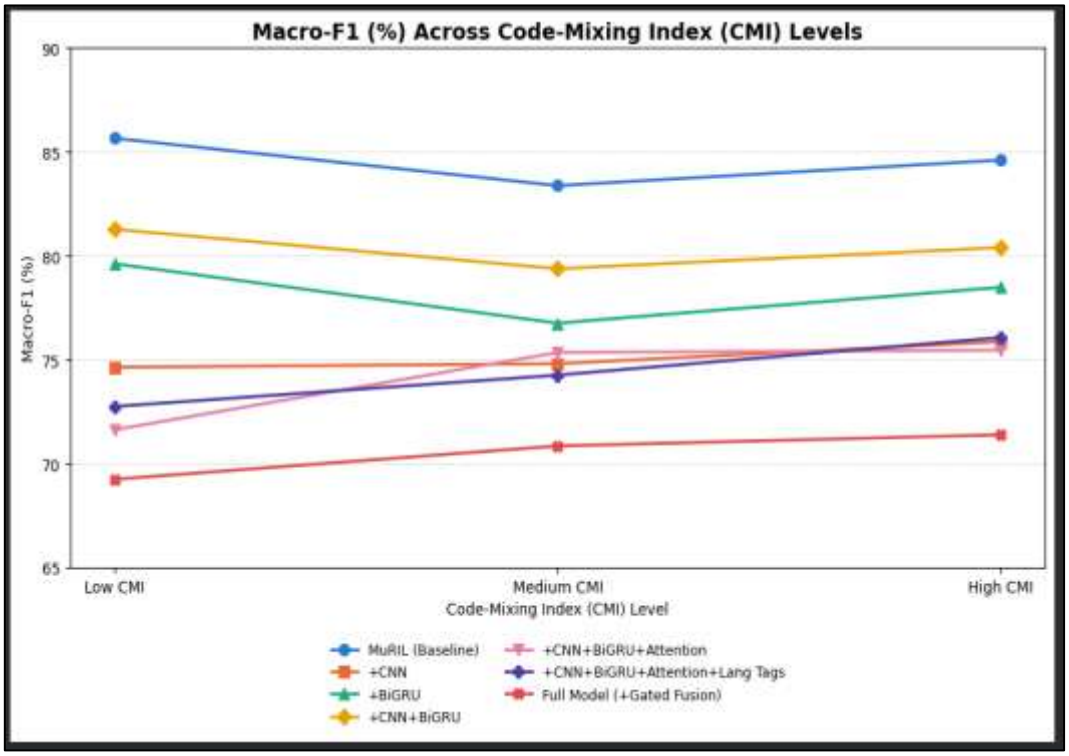

Figure 10. CMI-Based Robustness Analysis

5.5 Error Analysis

To better understand the limitations of the models evaluated, an error analysis was done, inspecting the samples of the validation dataset that were misclassified. The analysis indicated that the majority of classification errors were made for linguistically complex tweets, which express ambiguous sentiment, do not transliterate the words correctly, are composed of multiple language switches, contain sarcasm, are missing context, or express a less obvious sentiment.

The misinterpretation of neutral and positive sentiments was one of the frequent mistakes. The confusion matrices illustrate that some tweets containing mild appreciation or factual opinions were misclassified as positive because they did not contain enough contextual cues for the sentiment to be considered positive likewise, negative tweets with the implicit negative expression were sometimes misinterpreted as a neutral expression because there are no explicit polarity markers in the sentence. In the confusion matrices of the evaluated models, the neutral class is responsible for most of the misclassifications, which is indicative of this.

The second problem is the different way in which Hindi words are transliterated. Different tweets can have the same Hindi word written in different Romanized versions, which makes it harder to understand the meaning of a word. While MuRIL can deal with most of these variations successfully due to its pre-training on transliterated Indian corpora, it can still make prediction errors with highly irregular spellings and informal abbreviations. The experiments also suggest that errors don't necessarily decrease with increasing architectural complexity downstream. CNN, Attention, Language Tag Embeddings, and the Gated Fusion, in particular, did not help the classification of ambiguous samples and, in some cases, even decreased the overall performance. This indicates that the errors observed are mainly due to language ambiguity in the dataset rather than a lack of feature-extraction capability. In general, it is concluded that future improvements in the classification of Hinglish sentiment is likely to come more from using better language representations, better processing of informal transliterations, and context-aware reasoning, rather than from making downstream neural network more complicated.

6. Discussion And Practical Implications

The results indicate that using pretrained language representation is more crucial for sentiment classification in Hinglish language than to upscale the downstream architecture. The plain MuRIL baseline had the best performance in all models evaluated, with the addition of CNN, BiGRU, Attention, Language Tag Embeddings, and Gated Fusion showing no gains. The discovery implies that pretrained contextual

representations that have been trained using multi-lingual and transliterated corpora of Indian languages are good enough to capture the semantic properties of the Hinglish text.
Finally, the ablation study shows that the performance of CNN + BiGRU is slightly improved compared to using both models alone, but adding more attention components, embedding language, and fusing with gates causes performance to drop. The above observations suggest that no size limit exists for how complicated an architecture can be and still generalize well. Rather, the extra modules add extra trainable parameters and optimization complexity, but do not provide any complementary semantics to the one already captured by MuRIL.
The superiority of the plain MuRIL model was also confirmed by a statistical significance analysis. The performance difference was also found to be statistically significant using McNemar's test ($\chi^2 = 142.14$, $p = 9.06 \times 10^{-33}$) and bootstrap resampling ($p<0.0001$). Moreover, the architecture-enhanced model correctly classified 64 samples while the plain MuRIL model correctly classified 289 samples out of 353 samples in which the two models made different predictions. These results suggest that any performance difference is not random noise or due to initial settings of the model.
The CMI-based robustness analysis also shows that the MuRIL based baseline doesn't lose performance when varying code-mixing levels. This indicates the success of language-specific pretraining in dealing with transliterated Hindi words and the frequent code-switching in multilingual social media text, which are still significant challenges.
The results of this research have two significant conclusions. They first underline the advantages of using a suitable pretrained backbone as compared to using more and more complicated downstream designs for Hinglish sentiment classification. Second, they propose that research be directed towards better multilingual pretraining strategies, domain adaptation, and code-mixed language modelling, aside from stacking more neural components.

7. LIMITATIONS

Although the results here were positive, there are some limitations to this study. Experiment was conducted on a single Hinglish sentiment dataset which needs further investigation to generalise to other code-mixed language pairs.
Second, a limited number of pretrained transformer backbones, and only three of the most widely used downstream neural modules, were tested. Depending on the architecture, these may show different behaviour, e.g. instruction-tuned multilingual language models or parameter-efficient adaptation methods.
Third, the values of the Code-Mixing Index were computed by applying a heuristic language identification procedure, since no official annotation of the words' language was available. Thus, the reported CMI figures are considered as indicative of the degree of code-mixing.

8. Conclusion And Future Work

In this study, we conducted a comprehensive empirical study on pretrained transformer models and downstream neural architectures for Hinglish sentiment classification using the SemEval-2020 Task 9 benchmark dataset. To analyze the relative contribution of pretrained language representations and additional neural components, four pretrained language models (BERT, mBERT, XLM-R, and MuRIL) were systematically compared to understand their impact on sentiment classification performance in the same experimental conditions.
The experimental results revealed that the plain MuRIL model showed a consistently good performance from the architecture enhanced variants with CNN, BiGRU, Attention, Language Tag Embeddings and Gated Fusion, we saw no significant improvements over the architecture, with the best performing ones achieving worse scores than the architecture itself. The cumulative ablation study also showed that while some individual components, including CNN– BiGRU, offered moderate improvement compared to the single component, the addition of complexity to the architecture in the downstream was not guaranteed to boost the classification accuracy. These results suggest that the code-mix-aware pretrained backbone plays a more significant role for Hinglish sentiment classification than further.
To further support this, the robustness analysis using the Code-Mixing Index (CMI) was conducted on MuRIL, which confirmed that MuRIL stable performance over varying degrees of code mixed text, suitable for models of transliterated Hindi and English expressions. The error analysis also revealed that the majority of classification errors are due to ambiguous sentiment expressions or inconsistent transliteration, sarcasm and lack of context information, and not due to inadequate feature extraction.
This study generally shows that the selection of models is much more important than the complexity of their architecture for Hinglish Sentiment Analysis. The results provide valuable suggestions for researchers and practitioners on how to choose an effective transformer-based model for the multilingual sentiment analysis task.
Future work will focus on evaluating larger multilingual language models, parameter-efficient fine-tuning techniques like LoRA and adapters , as well as the ability to generalize across multiple datasets and languages, are also highlighted. Explainable artificial intelligence techniques to render the code-mixed

sentiment classification systems more comprehensible and reliable, and instruction-tuned language models.

ABBREVIATIONS

The following abbreviations are used in this manuscript, listed in alphabetical order.

Abbreviation	Full Form
ACL	Association for Computational Linguistics
BERT	Bidirectional Encoder Representations from Transformers
BiGRU	Bidirectional Gated Recurrent Unit
BiLSTM	Bidirectional Long Short-Term Memory
CI	Confidence Interval
CMI	Code-Mixing Index
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
FN	False Negative
FP	False Positive
GLUECoS	General Language Understanding Evaluation benchmark for Code-Switching
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
I2CT	International Conference for Convergence in Technology
ICON	International Conference on Natural Language Processing
IEEE	Institute of Electrical and Electronics Engineers
LoRA	Low-Rank Adaptation
LREC	Language Resources and Evaluation Conference
LSTM	Long Short-Term Memory
mBERT	Multilingual BERT
MCC	Matthews Correlation Coefficient
MuRIL	Multilingual Representations for Indian Languages
NLP	Natural Language Processing
ROC-AUC	Receiver Operating Characteristic – Area Under the Curve
RQ	Research Question
SemEval	Semantic Evaluation
TN	True Negative
TP	True Positive
WILDRE	Workshop on Indian Language Data: Resources and Evaluation
XLM-R	Cross-lingual Language Model – RoBERTa (XLM-RoBERTa)

References

[1] Patwa, P., Aguilar, G., Kar, S., Pandey, S., PYKL, S., Gambäck, B., Chakraborty, T., Solorio, T., Das, A.: SemEval-2020 Task 9: Overview of Sentiment Analysis of Code-Mixed Tweets. In: Proceedings of the Fourteenth Workshop on Semantic Evaluation, pp. 774–790. International Committee for Computational Linguistics, Barcelona (2020). https://doi.org/10.18653/v1/2020.semeval-1.100

[2] Baroi, S.J., Singh, N., Das, R., Singh, T.D.: NITS-Hinglish-SentiMix at SemEval-2020 Task 9: Sentiment Analysis for Code-Mixed Social Media Text Using an Ensemble Model. In: Proceedings of the Fourteenth Workshop on Semantic Evaluation, pp. 1298–1303. International Committee for Computational Linguistics, Barcelona (2020). https://doi.org/10.18653/v1/2020.semeval-1.175

[3] Srivastava, V., Singh, M.: IIT Gandhinagar at SemEval-2020 Task 9: Code-Mixed Sentiment Classification Using Candidate Sentence Generation and Selection. In: Proceedings of the Fourteenth Workshop on Semantic Evaluation, pp. 1259–1264. International Committee for Computational Linguistics, Barcelona (2020). https://doi.org/10.18653/v1/2020.semeval-1.168

[4] Kumar, A., Agarwal, H., Bansal, K., Modi, A.: BAKSA at SemEval-2020 Task 9: Bolstering CNN with Self-Attention for Sentiment Analysis of Code Mixed Text. In: Proceedings of the Fourteenth Workshop on Semantic Evaluation, pp. 1221–1226. International Committee for Computational Linguistics, Barcelona (2020). https://doi.org/10.18653/v1/2020.semeval-1.162

[5] Das, A., Gambäck, B.: Identifying Languages at the Word Level in Code-Mixed Indian Social Media Text. In: Proceedings of the 11th International Conference on Natural Language Processing (ICON 2014), pp. 378–387. NLP Association of India, Goa (2014).

[6] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, pp. 4171–4186. Association for Computational Linguistics, Minneapolis (2019). https://doi.org/10.18653/v1/N19-1423

[7] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised Cross-Lingual Representation Learning at Scale. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 8440–8451. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.acl-main.747

[8] Khanuja, S., Bansal, D., Mehtani, S., Khosla, S., Dey, A., Gopalan, B., Margam, D.K., Aggarwal, P., Nagipogu, R.T., Dave, S., Gupta, S., Gali, S.C.B., Subramanian, V., Talukdar, P.: MuRIL: Multilingual Representations for Indian Languages. arXiv preprint arXiv:2103.10730 (2021). https://doi.org/10.48550/arXiv.2103.10730

[9] Nayak, R., Joshi, R.: L3Cube-HingCorpus and HingBERT: A Code-Mixed Hindi-English Dataset and BERT Language Models. In: Proceedings of the WILDRE-6 Workshop within the 13th Language Resources and Evaluation Conference, pp. 7–12. European Language Resources Association, Marseille (2022). https://aclanthology.org/2022.wildre-1.2

[10] Patil, A., Patwardhan, V., Phaltankar, A., Takawane, G., Joshi, R.: Comparative Study of PreTrained BERT Models for Code-Mixed Hindi-English Data. In: 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), pp. 1–7. IEEE, Raigarh (2023). https://doi.org/10.1109/I2CT57861.2023.10126273

[11] Ghosh, S., Priyankar, A., Ekbal, A., Bhattacharyya, P.: Multitasking of Sentiment Detection and Emotion Recognition in Code-Mixed Hinglish Data. Knowledge-Based Systems 260, 110182 (2023). https://doi.org/10.1016/j.knosys.2022.110182

[12] Mamta, Ekbal, A.: Transformer Based Multilingual Joint Learning Framework for Code-Mixed and English Sentiment Analysis. Journal of Intelligent Information Systems 62(1), 231–253 (2024). https://doi.org/10.1007/s10844-023-00808-x

[13] Khare, B.K., Khan, I.: Optimized Emotion Classification in Code-Mixed Hinglish Text Using an mBERT Based Hybrid Neural Network with Attention Mechanisms. International Journal of Information Technology (2025). https://doi.org/10.1007/s41870-025-02813-5

[14] Zaharia, G.-E., Vlad, G.-A., Cercel, D.-C., Rebedea, T., Chiru, C.-G.: UPB at SemEval-2020 Task 9: Identifying Sentiment in Code-Mixed Social Media Texts Using Transformers and Multi-Task Learning. In: Proceedings of the Fourteenth Workshop on Semantic Evaluation, pp. 1322–1330. International Committee for Computational Linguistics, Barcelona (2020). https://doi.org/10.18653/v1/2020.semeval-1.179

[15] Guzmán, G.A., Ricard, J., Serigos, J., Bullock, B.E., Toribio, A.J.: Metrics for Modeling Code-Switching Across Corpora. In: Proceedings of Interspeech 2017, pp. 67–71 (2017). https://doi.org/10.21437/Interspeech.2017-1429

[16] Khanuja, S., Dandapat, S., Srinivasan, A., Sitaram, S., Choudhury, M.: GLUECoS: An Evaluation Benchmark for Code-Switched NLP. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 3575–3585. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.acl-main.329

[17] Barnett, R., Codó, E., Eppler, E., Forcadell, M., Gardner-Chloros, P., van Hout, R., Moyer, M., Torras, M.C., Turell, M.T., Sebba, M., Starren, M., Wensing, S.: The LIDES Coding Manual: A Document for Preparing and Analyzing Language Interaction Data. International Journal of Bilingualism 4(2), 131–271 (2000). https://doi.org/10.1177/13670069000040020101

[18] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: Proceedings of EMNLP 2014, pp. 1724–1734. ACL, Doha (2014). https://doi.org/10.3115/v1/D14-1179

[19] Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. In: Proceedings of ICLR 2015 (2015). arXiv:1409.0473

[20] Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: Proceedings of EMNLP 2014, pp. 1746–1751. ACL, Doha (2014). https://doi.org/10.3115/v1/D14-1181

[21] McNemar, Q.: Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. Psychometrika 12(2), 153–157 (1947). https://doi.org/10.1007/BF02295996

[22] Efron, B.: Bootstrap Methods: Another Look at the Jackknife. The Annals of Statistics 7(1), 1–26 (1979). https://doi.org/10.1214/aos/1176344552

[23] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-Rank Adaptation of Large Language Models. In: Proceedings of ICLR 2022 (2022). arXiv:2106.09685

[24] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: Proceedings of EMNLP-IJCNLP 2019, pp. 3982–3992. ACL (2019). https://doi.org/10.18653/v1/D19-1410
[25] Parveen, N., Khan, M.W.: A Conceptual Framework for Leveraging Web Data in Sentiment Analysis and Opinion Mining. Journal of Information Systems Engineering and Management, vol. 10, no. 41s, pp. 562–570 (2025).
[26] Joshi, M., Pandey, D., Pandey, V., Khan, M.W.: A fusion framework for Hinglish cyberbullying detection using mBERT and FastText. International Journal of Engineering in Computer Science, vol. 7, no. 1 (2025).
[27] Parveen, N., Khan, M.W.: Proposed Algorithm and Models for Sentiment Analysis and Opinion Mining Using Web Data. Nanotechnology Perceptions, vol. 20, no. 6 (2024).