A Switch-Point-Aware Contrastive Approach to Sentiment Analysis of Hinglish Code-Mixed Text

---

Prasant Kumar Sahoo

A Switch-Point-Aware Contrastive Approach to Sentiment Analysis of Hinglish Code-Mixed Text

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

Computer Science (with specialization in Data Science)

by

Prasant Kumar Sahoo

[Roll No: CS2421]

under the supervision of

Prof. Ujjwal Bhattacharya

Professor

Computer Vision and Pattern Recognition Unit

Indian Statistical Institute

Kolkata 700108, India

June 2026

Certificate

This is to certify that the dissertation entitled **"A Switch-Point-Aware Contrastive Approach to Sentiment Analysis of Hinglish Code-Mixed Text"** submitted by **Prasant Kumar Sahoo** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science (with specialization in Data Science)** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

10/06/2026

Prof. Ujjwal Bhattacharya
Professor,
Computer Vision and Pattern Recognition Unit,
Indian Statistical Institute,
Kolkata 700108, INDIA.

Declaration

I, **Prasant Kumar Sahoo**, declare that this dissertation is my own original work and that it has not been submitted, in whole or in part, for any other degree or qualification. All sources of information and prior work have been acknowledged through citation in accordance with academic convention. The design, implementation, and evaluation of the SPA-CL architecture described herein are my own, except where explicitly attributed to others.

Prasant Kumar Sahoo
Roll No. CS2421
MTech Computer Science
Indian Statistical Institute, Kolkata

To my parents,
for their enduring support and strength.

And to my advisor,
for the guidance and clarity that shaped this work.

Acknowledgments

It is with profound gratitude that I acknowledge the invaluable guidance of my advisor, *Prof. Ujjwal Bhattacharya*, for mentorship, deep insights, and consistent encouragement that played a pivotal role in shaping this research.

I sincerely thank all faculty members and researchers at Indian Statistical Institute for fostering a vibrant academic environment and providing the resources and freedom necessary for meaningful exploration.

My heartfelt appreciation also goes to my peers and labmates for their camaraderie, constructive discussions, and the shared learning experiences that enriched this journey.

Finally, I am deeply indebted to my family especially my parents for their unwavering support, sacrifices, and faith in me. Their love and strength have been the cornerstone of my academic pursuit.

Prasant Kumar Sahoo
Indian Statistical Institute
Kolkata 700108, INDIA

Abstract

With the increasing use of social media in non-English-speaking regions, especially in India, people often use Romanized Hindi and English together in their online communication. In a single sentence, they frequently mix Romanized Hindi and English, creating code-mixed text. However, most multilingual transformer models are pre-trained primarily on monolingual data. As a result, NLP systems face challenges when processing code-mixed text, as a single word may be fragmented into meaningless subword pieces, making it difficult for the model to capture its semantic meaning accurately.

In this dissertation, we propose a parameter efficient neural architecture consisting of three main components to address these challenges: First, there is a character-level CNN encoder, which handles spelling differences such as "nahi", "nahin", "nah", and "nai" through the chracter n-gram pattern. Next, there is a frozen XLM-R backbone(Conneau et al., 2019) , the top three layers, which are partly fine-tuned at a slower rate by which it provides rich cross lingual embeddings. Finally, there is a switch-point-aware bilingual gate that spots where the language label switches and blends two adapters using a learned gate weight.During training, it uses Supervised Contrastive Loss to learn better feature representations and Cross-Entropy Loss for classification. Since human annotators agreed on labels only 55% of the time, we use label smoothing to reflect this uncertainty and prevent the model from becoming overly confident in noisy labels.

Evaluated on the SentiMix 2020 benchmark(Patwa et al., 2020), our proposed architecture achieves a weighted F1 score of 0.705, which outperforms the baseline model M-BERT (0.654 F1) and is comparable to fully fine-tuned transformer models while requiring only one-tenth of the trainable parameters.Adapter gate visualizations provide interpretable evidence that the gating mechanism captures linguistically meaningful code-mixing structure. The architecture is designed to generalize to other code-mixed language pairs through its modular adapter design.

Keywords: code-mixing, sentiment analysis, Hinglish, contrastive learning, adapter modules, character CNN, switch-point detection, XLM-R

Contents

List of Figures

List of Tables

Abbreviations

NLP	Natural Language Processing
CMI	Code Mixing Index
BPE	Byte-Pair Encoding
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
BiLSTM	Bidirectional LSTM
XLM-R	Cross-lingual Language Model – RoBERTa
mBERT	Multilingual BERT
MLM	Masked Language Modelling
SupCon	Supervised Contrastive (Loss)
CE	Cross-Entropy
IAA	Inter-Annotator Agreement
LID	Language Identification
F1	F1 Score (harmonic mean of precision and recall)
HIN/ENG/O	Hindi / English / Other (language tags)

Chapter 1

Introduction

1.1 Motivation

Massive amounts of text written in languages and styles that current natural language processing (NLP) systems were never intended to handle have been generated by the growth of social media platforms. Code mixing or code switching between two or more languages in a single message is one of the most common and linguistically unique phenomena. In India, Hindi and English are both widely used in education, commerce, and popular culture, code-mixing between Hindi and English is commonly referred to as Hinglish —is widespread on Twitter, WhatsApp, Facebook, and other platforms as well. Understanding the sentiment expressed in these messages is commercially and socially important: it enables the monitoring of public opinion about political events, products, and social issues in one of the largest populations of Internet users in the world.

However, Hinglish presents a formidable challenge to standard NLP systems. Three properties of the language make it particularly difficult. First, because Hindi-speakers writing in Hinglish use the Latin alphabet rather than Devanagari script, the same Hindi word may be spelled in many different ways. The negation word meaning “no” or “not” appears variously as ‘nahi’, ‘nahin’, ‘nah’, ‘nai’, and ‘nahii’. Standard subword tokenisers, trained in formal Devanagari-script Hindi, have no vocabulary entries for any of these forms and fragment them into meaningless character sequences; the negation signal is crucial for correct sentiment classification , is lost. Second, switching between Hindi and English within a sentence is not random ; rather, it happens at linguistically motivated switch points that frequently convey the most important emotional or emphatically important information. Any system that handles Hindi and English tokens equally will miss this structural signal: the sentence “Aaj ka din bohot bura tha, and honestly I am DONE". Changes to English exactly to provide the increased negative conclusion "I am DONE". Third, even human annotators only agree on the label for roughly 55% of tweets, owing to the intrinsic subjectivity of sentiment annotation (Patwa et al., 2020). This results in a noisy training signal that requires regularization.

This dissertation addresses all three challenges through a unified architecture that we call the Switch-Point-Aware Contrastive Learner, or SPA-CL. The model combines a character-level convolutional neural network for transliteration robustness, a partially fine-tuned multilingual transformer for contextual embeddings, a bidirectional LSTM for task-specific sequential modelling, and — as the central architectural contribution — a gated bilingual adapter that applies different transformations at switch points versus non-switch positions, explicitly modelling the linguistic significance of language alternation.

1.2 Problem Statement

Given a Hinglish tweet, the system must assign one of three sentiment labels: Positive, Negative, or Neutral.
For Example:
Positive: "Dobar se PM bnne ki aapko dher saari shubhakamnyen ummid hi aapke is karykala me desh bhut si nyi uplbadhiyon karega".
English: (Congratulations on becoming Prime Minister once again. I hope that under your leadership, the country will achieve many new milestones and successes during this term.)
Neutral: "Itna bura nahi tha the experience actually".
English: (The experience was actually not that bad.)
Negative: "Aaj ka din bohot bura tha, and honestly I am DONE with this".
English:(Today was a very bad day and Honestly I am DONE with this.)

Each one's text blends romanized Hindi and English, marked at the word level with HIN for Hindi, ENG for English, or Other.
Example:Aaj_{HIN} ka_{HIN} din_{HIN} $\text{bohot}_{\text{HIN}}$ bura_{HIN} tha_{HIN}, and_{ENG} $\text{honestly}_{\text{ENG}}$ I_{ENG} am_{ENG} DONE_{ENG}.

Figure 1.1 dissects this example, marking the language tags, the switch point, and the properties that make such tweets difficult for standard NLP pipelines.

Like in the SentiMix 2020 task, we'll use a weighted F1 score to judge performance (Patwa et al., 2020).

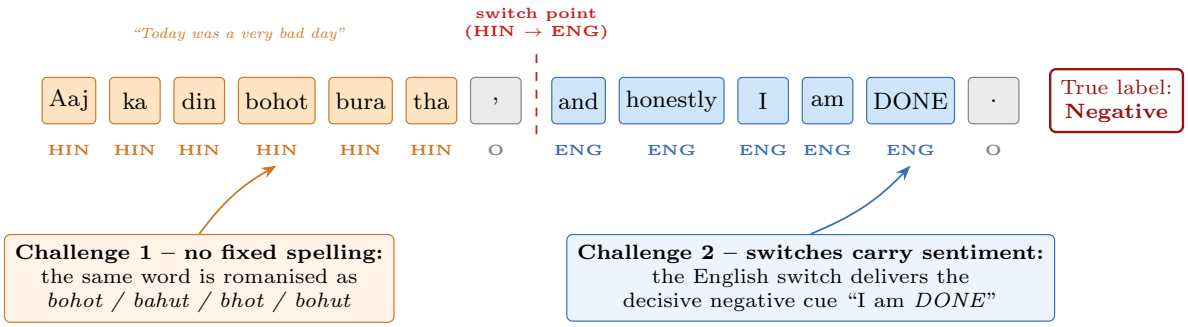

Figure 1.1: Anatomy of a code-mixed Hinglish tweet (the Negative example above). Word-level language tags (HIN/ENG/O) segment the tweet into a romanised-Hindi clause and an English conclusion. The two properties that defeat standard pipelines are marked: romanised Hindi has no canonical spelling, and the language switch is precisely where the decisive sentiment cue ("I am DONE") arrives. The architecture of Chapter 4 is designed to exploit both properties.

1.3 Research Objectives

Four distinct research goals are pursued in this dissertation:

1. A character level CNN encoder implemented to produce similar embeddings for phonetically similar words but orthographically different, which makes downstream classification less dependent on preexisting subword vocabularies.

2. Using a switch-point-aware gating mechanism we capture different semantic weights of language switches by explicitly finding positions in a Hinglish sequence where the language label changes and applying a learnable interpolation between language specialized adapter modules at those locations.

3. There is a three stage curriculum training process uses Supervised Contrastive Loss to generate sentence embedding space such that similar sentiment tweets are grouped together and various traliterated forms of tweet lie close to one another.

4. Using adapter gate visualizations and per-CMI group analysis, we determine architecture performance relative to the SentiMix 2020 benchmark, with sufficient evidence to show that switch-point methodology significantly improves overall performance

1.4 Contributions

The principal contributions of this dissertation are:

1. **The SPA-CL architecture:** a complete, implemented, and evaluated system for Hinglish sentiment analysis that integrates character-level, subword-level, and sequential modelling with a novel switch-point-aware bilingual gating mechanism.

2. **The switch-point gate:** a sigmoid-gated interpolation between two language-specific bottleneck adapters, triggered at token positions where the language label changes. To our knowledge, this is the first architecture to apply soft bilingual gating specifically and exclusively at switch points in a code-mixed sequence.

3. **A three-stage curriculum for contrastive code-mixed NLP:** a training procedure that stabilises Supervised Contrastive Learning on a randomly-initialised BiLSTM by separating pure cross-entropy pre-training from gradual contrastive introduction, using a temperature $\tau = 0.15$ calibrated for non-pretrained encoders.

4. **Adapter gate visualisations:** a reproducible evaluation methodology that generates token-level gate value heatmaps for diagnostic probe strings, providing interpretable evidence of linguistically-motivated gating behaviour.

5. **An honest benchmark comparison** showing SPA-CL's performance relative to the SentiMix leaderboard, the M-BERT baseline, fully fine-tuned XLM-R, and frontier large language models (GPT-4o-mini and LLaMA-3 70B) evaluated on identical data using three-shot prompting.

1.5 Dissertation Structure

The rest of this dissertation is organized as follows. Chapter 2 analyses the essential background literature on code-mixed NLP, multilingual pre-trained language models, adapter-based fine-tuning, and contrastive representation learning. Chapter 3 provides a detailed description of the SentiMix 2020 dataset and the problem formulation. Chapter 4 describes the SPA-CL architecture, including a detailed specification of each component, its design reasoning, and its relationship to prior work. Chapter 5 describes the experimental setup including hyperparameter choices, training procedure, and evaluation protocol. Chapter 6 summarises and analyses the experimental outcomes. Chapter 7 examines the approach's limitations and potential research directions. Chapter 8 concludes.

Chapter 2

Background and Related Work

2.1 Code-Mixing as a Linguistic Phenomenon

Code-mixing — the interleaving of Multilingual communities routinely weave several languages into a single utterance; the practice is documented across(Verma, 1976; Bokamba, 1988; Singh, 1985). Unlike formal code-switching, which describes deliberate shifts between languages across utterance boundaries, intra-sentential code-mixing involves the insertion or alternation of lexical items, morphemes, or syntactic constituents from one language into the frame of another within a single sentence. This is the dominant pattern in written Hinglish social media text.

According to Solorio and Liu (2008), there are two main structural features: language alternation, in which strings of words in one language are substituted for those in another at syntactically natural boundaries; and nonce borrowing, in which a single word from one language is borrowed into another language's grammatical system. For example, "Nahi wo is news ko defend kerne ki koshish ker rhe hain" contains the borrowing of the English verb 'defend' into the Hindi grammatical structure.

The Code-Mixing Index (CMI), introduced by Gambäck and Das (2016), is a pivotal metric for quantifying the degree of language mixing within a sentence:

$$\text{CMI} = 100 \times \left(1 - \frac{\max_{\text{language}} \text{count}}{\text{total\_tokens}}\right). \tag{2.1}$$

High CMI values indicate high levels of language mixing. As noted by Patwa et al. (2020), the SentiMix Hinglish dataset features moderate but consistent code-mixing, with a median CMI score of 25.3. Because sentiment-bearing tokens may occur in either language, the presence of code-mixing makes sentiment analysis harder due to the placement of negation and intensity markers.

2.2 Challenges of Romanised Code-Mixed Text

Hinglish presents additional challenges beyond those of code-mixed text in general because it is written in the Latin alphabet (Romanised script) rather than Devanagari. This transliteration practice is widespread in India (Sitaram et al., 2019): speakers write Hindi phonetically in Latin characters, following no standardised orthographic convention. The same word is written differently by different speakers, by the same speaker on different occasions, and even within the same tweet: 'nahi', 'nahin', 'nah', 'nai', and 'nahii' are all common Romanisations of the same negation word.

This orthographic variability is deeply problematic for standard NLP pipelines. Subword tokenisers trained on multilingual corpora — including mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2019) — learn their vocabularies from formal text in each language's standard script. Romanised Hindi words are effectively out-of-vocabulary for these tokenisers, which fall back on character-level byte-pair encoding and produce high-fragmentation segmentations. According to Patwa et al. (2020), Hinglish text under the M-BERT tokeniser has BPE fertility values (subword pieces per word) of roughly 2.5, implying that each Hindi word is often divided into 2.5 meaningless fragments. For the negation word 'nahi', the M-BERT tokeniser produces ['na', '##hi'] — two fragments with no individual meaning — destroying the semantic signal that is critical for correct sentiment classification.

The challenge of non-standard spelling in code-mixed social media is documented by Das and Gambäck (2014), who point out that tokenisation and normalisation are far more difficult than for standard monolingual text due to the mixture of arbitrary punctuation, capitalisation, abbreviation, and spelling variation.

2.3 Sentiment Analysis of Code-Mixed Text

Earlier techniques for analysing sentiment in mixed languages used dictionaries and shallow machine learning algorithms. SVM-based classifiers trained on lexicons were employed by Mohammad et al. (2013) for determining tweet sentiment, while Sharma et al. (2015) employed lexicon normalisation to harmonise spellings prior to sentiment classification. Lexicon-based methods require manually constructed dictionaries, which do not extend to the open vocabulary of Hinglish in Roman script.

The SemEval-2020 Task 9 shared task on SentiMix (Patwa et al., 2020) provided the first large-scale annotated benchmark for Hinglish and Spanglish (Spanish–English) sentiment analysis, catalysing a wave of neural approaches. Among the 61 teams that participated in the Hinglish subtask, the most successful approaches used pre-trained transformer

models as feature extractors. The top-performing system, KK2018 (rank 1, weighted F1 = 0.750), used XLM-R with adversarial training examples generated by gradient-based perturbation (Miyato et al., 2017). The second-ranked system, Genius1237 (F1 = 0.726), used XLM-R embeddings as input to a classification layer combined with M-BERT. BAKSA (rank 5, F1 = 0.707) combined XLM-R embeddings with a CNN and self-attention ensemble. The official M-BERT baseline achieved F1 = 0.654.

There were also several efforts incorporating character representations. HPCC-YNU ranked 15th (F1 = 0.686) with a BiLSTM model with attention, employing both word and character embeddings as inputs, pointing to the relevance of character embeddings for managing spelling variation (Bao et al., 2020; Baroi et al., 2020; Bhange and Kasliwal, 2020). From the SentiMix results it is evident that transformers are superior and that character-level representations are also relevant.

2.4 Multilingual Pre-Trained Language Models

Cross-lingual NLP has changed as a result of the creation of multilingual pre-trained language models. Devlin et al. (2019) introduced BERT, a transformer encoder pre-trained on masked language modelling (MLM) and next-sentence prediction. The Wikipedia dumps of 104 languages were used to train the multilingual variant, mBERT, which learned shared cross-lingual representations. Despite having no cross-lingual objective in its pre-training, Pires et al. (2019) showed that mBERT accomplishes cross-lingual zero-shot transfer.

Conneau et al. (2019) introduced XLM-R (Cross-lingual Language Model — RoBERTa), pre-trained on 2.5 terabytes of multilingual text from 100 languages using the RoBERTa training recipe (Liu et al., 2019) without the next-sentence-prediction objective. XLM-R substantially outperforms mBERT on cross-lingual benchmarks including XNLI, XQuAD, and MLQA. Its SentencePiece vocabulary of 250,002 tokens, trained with byte-pair encoding on balanced multilingual data, provides better coverage of low-resource languages than mBERT's WordPiece vocabulary. For code-mixed Hinglish, XLM-R's Latin-script coverage provides better tokenisation of English components and partial coverage of Romanised Hindi compared to mBERT.

2.5 Adapter-Based Parameter-Efficient Fine-Tuning

Full fine-tuning of large pre-trained models on small downstream datasets is prone to overfitting and catastrophic forgetting — the degradation of general linguistic knowledge acquired during pre-training (McCloskey and Cohen, 1989). Adapter-based fine-tuning,

introduced by Houlsby et al. (2019), addresses this by inserting small trainable modules (adapters) into the frozen pre-trained model, only the adapter weights are updated; the backbone stays frozen

The Houlsby adapter follows a bottleneck design: a down-projection from d_{model} to d_{adapt} dimensions, a nonlinear activation, and an up-projection back to d_{model}, with a residual connection. Near-zero initialisation of the up-projection ensures that the adapter begins as near-identity and gradually specialises during training. With $d_{\text{adapt}} = 64$ and $d_{\text{model}} = 512$, each adapter introduces approximately 65,000 parameters — about 0.13% of XLM-R base's parameter count.

Pfeiffer et al. (2020b) extended adapters to multilingual settings through the MAD-X framework (Modular Approach to Cross-lingual Transfer), using language adapters pre-trained on unlabelled text to encode language-specific knowledge, and task adapters trained on labelled data for task-specific knowledge. Because the two levels of adapters are stacked, the task adapter can build on language-specific representations without updating the pre-trained backbone. SPA-CL takes this insight and applies it at the token level rather than the model level: language-specific adapters are selected or blended based on the token's language identity, instead of a single language adapter applied uniformly.

2.6 Contrastive Representation Learning

The contrastive approach to training encoders seeks to generate a representation space that pulls related inputs together and pushes unrelated ones apart. SimCLR was proposed by Chen et al. (2020), which uses data augmentation to form positive pairs (two different augmented forms of the same example) and treats all other examples in the batch as negatives, with a temperature-scaled cosine similarity cross-entropy loss. Khosla et al. (2020) applied this concept in the supervised setting by defining any pair of examples sharing the same class label to be positive pairs (SupCon).

Contrastive learning has only recently been used in NLP. Gao et al. (2021) applied SimCSE (a contrastive sentence embedding method) to sentence representations. Because phonetic augmentation of Hinglish text naturally generates positive pairs — 'nahi' and 'nahin' are augmented versions of the same concept and should map to nearby representations — the contrastive paradigm is especially well-suited for code-mixed NLP. By combining SupCon with phonetically-augmented training pairs and using the augmented versions as extra positive examples in the contrastive objective, SPA-CL takes advantage of this.

2.7 Character-Level Neural Representations

In order to handle morphologically rich languages and misspellings without relying on preset word vocabularies, character-level neural models were introduced. Character-level CNNs were shown by Zhang et al. (2015) to be capable of text classification without word embeddings. Character-aware language models, which feed character CNN outputs to a word-level LSTM, were introduced by Kim (2014). For NLP tasks involving noisy, non-standard text, character models have consistently outperformed word-level models on out-of-vocabulary terms.

For code-mixed Hinglish, the character-level approach is particularly well-motivated because the core challenge — transliteration variance — is precisely a character-level phenomenon. Two tokens that are phonetically equivalent but orthographically distinct (such as 'koshesh' and 'koshish') share many character n-gram patterns and can be brought together by a character CNN trained with a contrastive objective.

2.8 Positioning SPA-CL

SPA-CL occupies a unique place among code-mixed NLP systems. Unlike the top SentiMix systems (KK2018, Genius1237), which employ pre-trained transformer features but apply uniform fine-tuning without explicit code-mixing structure, SPA-CL specifically models the switch-point structure of Hinglish phrases. Unlike ensemble methods (BAKSA, Voice@SRIB), which combine multiple independent models, SPA-CL integrates all its components in a single unified forward pass with shared parameters. Unlike HPCC-YNU's BiLSTM with character embeddings, which uses a single character embedding layer without any contrastive training objective, SPA-CL trains the character encoder with a contrastive objective that explicitly enforces transliteration invariance. The key contribution comes from the combination of (a) explicit switch-point identification with soft bilingual gating, (b) character-level transliteration with contrastive learning, and (c) curriculum learning combining SupCon with cross-entropy loss.

Chapter 3

Dataset and Problem Formulation

3.1 The SentiMix 2020 Dataset

The SentiMix 2020 dataset was created for SemEval-2020 Task 9 (Patwa et al., 2020) on Sentiment Analysis of Code-Mixed Tweets. It comprises two corpora: a Hinglish (Hindi–English) corpus and a Spanglish (Spanish–English) corpus. This dissertation uses the Hinglish corpus exclusively.

3.1.1 Data Collection

The Hinglish dataset was constructed starting from the collection of 10,786 Hindi tokens gathered by Patra et al. (2018), removing tokens that also appear in English. Tweets containing one of these tokens were then retrieved using the Twitter API. The aim was to gather tweets containing genuine Hindi data rather than English.

3.1.2 Annotation

Each token received one of three language tags — HIN, ENG, or O (named entities, usernames, numerals, mixed tokens) — assigned with the automated tagger of Bhat et al. (2014). Using a crowdsourcing platform, some sixty bilingual annotators fluent in Hindi completed sentence-level sentiment annotation. Each tweet was shown to two annotators; matching annotations were retained and those that differed were discarded. The inter-annotator agreement rate is 55%, reflecting the inherent subjectivity of sentiment classification for short, informal, and culturally-specific text. This low agreement rate has significant ramifications for model calibration: it indicates substantial uncertainty in the "ground truth" labels themselves, so highly confident models will overfit to annotator noise.

3.1.3 Corpus Statistics

The Hinglish corpus contains 20,000 annotated tweets partitioned into training (14,000), validation (3,000), and test (3,000) splits. The class distribution is approximately balanced, as shown in Table 3.1.

Table 3.1: SentiMix Hinglish corpus statistics (Patwa et al., 2020).

Split	**Total**	**Positive**	**Neutral**	**Negative**
Train	14,000	4,634 (33.10%)	5,264 (37.60%)	4,102 (29.30%)
Validation	3,000	982 (32.73%)	1,128 (37.60%)	890 (29.67%)
Test	3,000	1,000 (33.33%)	1,100 (36.67%)	900 (30.00%)
Total	20,000	6,616 (33.08%)	7,492 (37.46%)	5,892 (29.46%)

The average Code-Mixing Index (CMI) across splits is 25.32 (train), 25.53 (validation), and 25.13 (test), indicating consistent levels of code-mixing across all partitions. The relatively stable CMI across splits is important for evaluation validity.

3.1.4 Sentiment Class Definitions

The three sentiment classes are defined as follows (Patwa et al., 2020). **Positive:** A tweet is labelled Positive when it conveys approval or happiness — applauding people, products, or institutions (e.g., "Congratulations Sir we are proud of you.. We believe in you.. You will be a very good home minister."(Patwa et al., 2020)). **Negative:** Negative tweets voice criticism, displeasure, or hostility toward their subject (e.g., "You efficiency of anchoring a program is continuously deteriorating. Ab to dekhne ki himmat hi nahi."(Patwa et al., 2020)). **Neutral:** Neutral covers factual statements, news items, and promotional content (e.g., "Nahi wo is news ko defend kerne ki koshesh ker rhe hain"(Patwa et al., 2020) — they are trying to defend this news).

3.2 BPE Fertility Analysis

To motivate the character CNN component of SPA-CL, we conduct a BPE fertility analysis on the three dissertation probe strings using both the mBERT and XLM-R tokenisers. BPE fertility is defined as the ratio of subword tokens produced to the number of whitespace-delimited words in the input:

$$\text{Fertility} = \frac{n_{\text{subword tokens}}}{n_{\text{words}}}. \tag{3.1}$$

Table 3.2: BPE fertility on SPA-CL diagnostic probe strings.

Probe Case	**mBERT Fertility**	**XLM-R Fertility**
Negation fragmentation: "Nahi wo is news ko defend kerne…"	∼2.8	∼2.4
Cross-boundary negation: "Itna bura nahi tha the experience…"	∼2.5	∼2.2
Pragmatic intensifier: "Aaj ka din bohot bura tha, and honestly…"	∼2.3	∼2.0

Both tokenisers fragment Romanised Hindi heavily. The most notable issue lies with the word "Nahi" (meaning "no/not", a negation marker): mBERT outputs ['Na', '##hi'] and XLM-R produces a comparable fragmentation. This yields nonsensical fragments, making negation detection impossible. Since the scope of negation plays an important role in sentiment determination, this negatively affects sentiment classification.

3.3 Formal Problem Definition

Let $x = (w_1, w_2, \ldots, w_T)$ be a Hinglish tweet represented as a sequence of T whitespace-delimited tokens, where $T \leq 56$. Each token w_t has an associated language label $l_t \in \{\text{HIN}, \text{ENG}, \text{O}\}$. function $f : \mathcal{X} \to \{\text{Positive}, \text{Negative}, \text{Neutral}\}$ from a training set $D = \{(x_i, y_i)\}$, where y_i is the sentiment label of tweet x_i. Learning amounts to fitting f from training pairs; evaluation uses the held-out splits with weighted F1 as the headline metric.

3.4 Data Pre-Processing

3.4.1 Character Vocabulary Construction

A character-level vocabulary is constructed by scanning all text in the training, validation, and test splits and collecting every unique Unicode character. Two special tokens are added: `<PAD>` at index 0 (used to pad character sequences to a fixed length) and `<UNK>` at index 1 (for characters unseen during vocabulary construction). All other characters are assigned indices in sorted Unicode order to ensure deterministic construction. The resulting vocabulary contains approximately 970 characters for the base corpus. For extended multilingual coverage (used in the robustness experiments of Section 7.3), the vocabulary is augmented with complete Unicode blocks for Devanagari (U+0900–U+097F), Bengali

(U+0980–U+09FF), and Tamil (U+0B80–U+0BFF) scripts, expanding it to approximately 3,200 characters.

3.4.2 XLM-R Tokenisation and Language Tag Alignment

Each tweet is tokenised using the XLM-R SentencePiece tokeniser with a maximum sequence length of 56 subword tokens, consistent with the SentiMix baseline (Patwa et al., 2020). Sequences are truncated at 56 subwords or right-padded with `[PAD]`. The word-level language annotations from SentiMix are aligned to the subword token sequence using a best-effort word-to-subword mapping: each subword piece inherits the language label of the whitespace-delimited word it belongs to. Special tokens (`[CLS]`, `[SEP]`, `[PAD]`) receive the label O.

3.4.3 Character ID Construction

For each subword token in the aligned sequence, the token string (with the SentencePiece prefix character '␣' stripped) is converted to a sequence of character indices by looking up each character in the character vocabulary. Sequences shorter than char_max_len = 20 are right-padded with the PAD index (0); longer sequences are truncated. This produces a tensor of shape $(56, 20)$ per tweet.

Chapter 4

Methodology

4.1 Architecture Overview

SPA-CL is a multi-component neural architecture in which five modules process each tweet in a coordinated pipeline: a frozen XLM-R backbone (Stage 0), a Character CNN encoder (Stage 1), a fusion layer, a bidirectional LSTM (Stage 2A), and a switch-point-aware gated adapter stack (Stage 2B). The complete pipeline is illustrated in Figure 4.1. Each tweet is represented by three input tensors: `input_ids` and `attention_mask` of shape $(B, 56)$, where B is the batch size; and `char_ids` of shape $(B, 56, 20)$. These are produced during pre-processing as described in Section 3.4. A fourth tensor, `lang_tags` of shape $(B, 56)$, encodes the language identity of each token.

4.2 Stage 0: Frozen XLM-R Backbone

4.2.1 Architecture

The XLM-R base model (Conneau et al., 2019) is a 12-layer transformer encoder with hidden dimension 768, 12 attention heads, and a feed-forward dimension of 3,072. The total parameter count of XLM-R base is approximately 278 million. It is initialised from the published pre-trained weights and used to extract contextual subword embeddings for each token. The forward pass computes $h_{\text{xlmr}} = \text{XLM-R}(\text{input\_ids}, \text{attention\_mask}) \in \mathbb{R}^{B \times 56 \times 768}$, where h_{xlmr} is the last hidden state, providing one 768-dimensional contextual vector per position.

4.2.2 Partial Freezing Strategy

A key design decision in SPA-CL is the partial freezing of XLM-R. Specifically, layers 0 through 8 (the lower nine of twelve transformer layers) have their parameters frozen: no gradients are computed for them during backpropagation. Layers 9 through 11 (the top

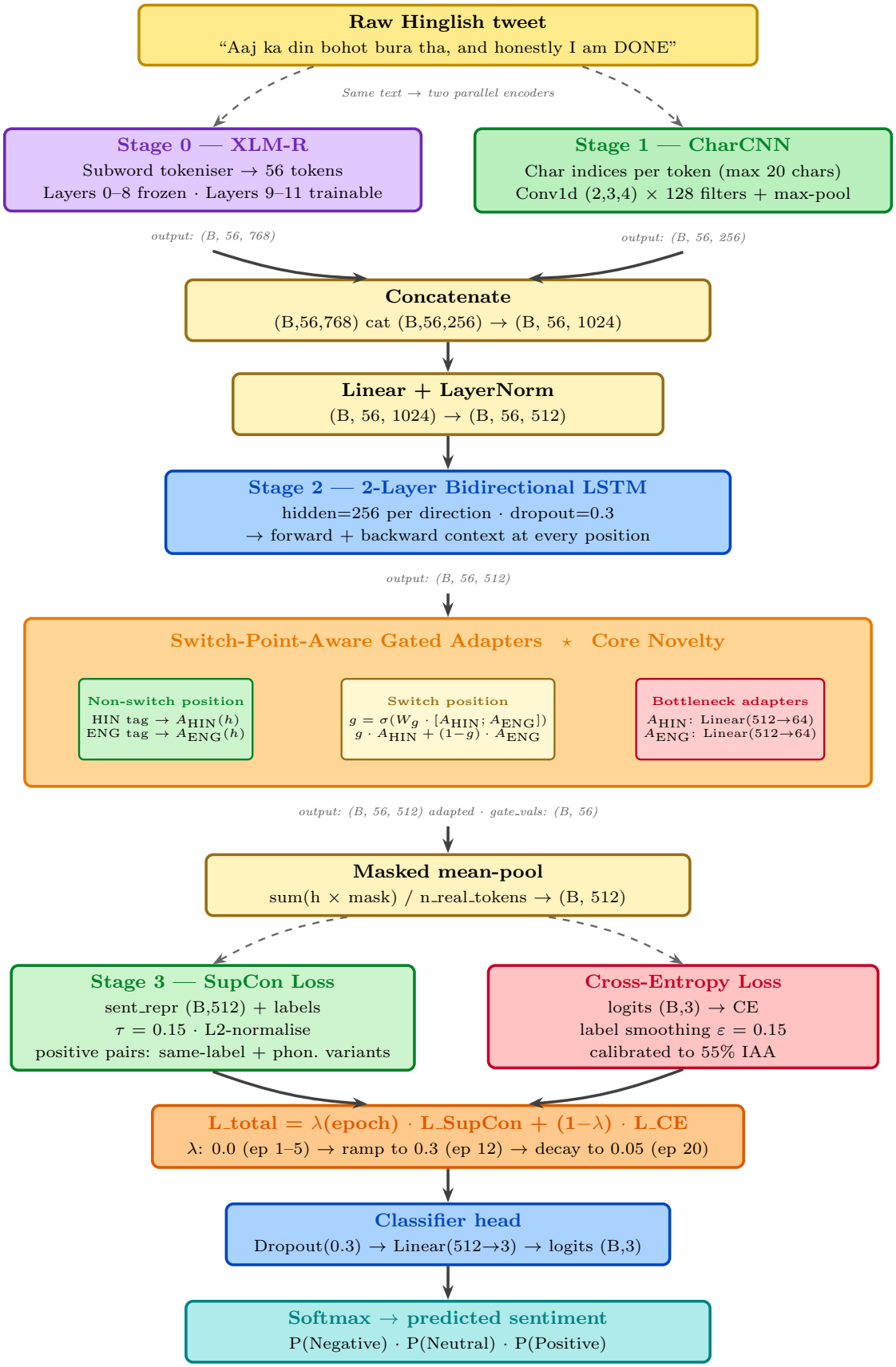

Figure 4.1: SPA-CL full architecture — raw Hinglish tweet flowing through XLM-R, CharCNN, BiLSTM with gated adapters, SupCon loss, and the classification head.

three) and the embedding layer are kept trainable, contributing approximately 21 million parameters.

This approach is inspired by the well-established finding in transfer learning that lower transformer layers encode universal language knowledge (morphology, syntax, low-level semantics) while upper layers encode task- and domain-specific representations (Tenney et al., 2019). Freezing the lower layers preserves the cross-lingual knowledge acquired during pre-training — essential for handling both Hindi and English in a code-mixed context — while allowing the upper layers to adapt to the specific vocabulary and sentiment patterns of Hinglish social media. To mitigate catastrophic forgetting in the upper layers, a differential learning rate is applied: the top three XLM-R layers are trained at a learning rate of 5×10^{-6}, ten times smaller than the 5×10^{-5} applied to newly-initialised components.

4.3 Stage 1: Character CNN Encoder

4.3.1 Architecture

The Character CNN encoder processes character-level representations of each token to produce transliteration-robust token vectors, following Kim (2014) with adaptations for the Hinglish setting. For each token, the `char_ids` sub-tensor — a sequence of up to 20 character indices — is passed through the CNN. To process all tokens simultaneously, the batch and sequence dimensions are merged before the CNN and restored afterwards.

Step 1 — Character embedding lookup.
Each character index is mapped through the embedding matrix $E_{\text{char}} \in \mathbb{R}^{V \times 50}$, yielding a 50-dimensional vector per character. Padding uses padding_idx = 0 so that it contributes no gradient.

Step 2 — Convolutional feature extraction.
Three groups of one-dimensional convolutional filters with kernel sizes $\{2, 3, 4\}$ are applied in parallel; each group contains 128 filters. With $W_w \in \mathbb{R}^{128 \times 50 \times w}$, the convolution computes $h_w[p] = \text{ReLU}(\sum_k W_w[k] \cdot x_{\text{embed}}[p + k] + b_w)$ for each position p. Size-2 kernels capture bigrams (e.g. “na”, “sh”), size-3 kernels capture trigrams (“nah”), and size-4 kernels capture tetragrams (“nahi”).

Step 3 — Max-over-time pooling.
Global max pooling reduces each feature map to a single value per filter, capturing the most-activated pattern across the token’s character sequence and yielding a 128-dimensional vector per filter width.

Step 4 — Concatenation and projection.
The three pooled vectors are concatenated into a 384-dimensional vector and projected through a linear layer with ReLU and dropout to the 256-dimensional output: char_vecs $\in \mathbb{R}^{B\times 56\times 256}$.

4.3.2 Transliteration Invariance

The CharCNN's primary property is that it generates comparable output vectors for tokens that are phonetically equivalent but orthographically distinct. For the negation word variants "nahi", "nahin", "nah", and "nai", the bigram filter for "na" activates at position 0 for all four; the bigram filter for "ah" activates for "nahi" and "nah"; and the trigram filter for "nah" activates for "nahi", "nahin", and "nah". After max-pooling, all four variants' 256-dimensional output vectors share a sizeable fraction of active features. This invariance is further enforced during training by the Supervised Contrastive Loss (Section 4.8.1), which treats phonetically-augmented versions as positive pairs.

4.4 Fusion Layer

The outputs of the XLM-R backbone (768-dimensional) and the Character CNN (256-dimensional) are combined per token position by concatenation:
combined = concat(xlmr_vecs, char_vecs) $\in \mathbb{R}^{B\times 56\times 1024}$. A subsequent linear projection and layer normalisation map the joint representation to a unified $d_{\text{model}} = 512$ space:

$$\text{projected} = \text{LayerNorm}(W_{\text{proj}} \cdot \text{combined} + b_{\text{proj}}) \in \mathbb{R}^{B\times 56\times 512}, \tag{4.1}$$

where $W_{\text{proj}} \in \mathbb{R}^{1024\times 512}$. Layer normalisation stabilises the potentially mismatched scales of the pre-trained XLM-R features and the randomly-initialised CharCNN features.

4.5 Stage 2A: Bidirectional LSTM

4.5.1 Architecture

A 2-layer bidirectional LSTM processes the fused token representations to model sequential dependencies. With hidden_size = 256 units per direction, each LSTM layer produces a concatenated 512-dimensional output that matches d_{model}. A forward LSTM reads positions 1 to 56 and a backward LSTM reads positions 56 to 1; their outputs are concatenated at each position. Dropout of probability 0.3 is applied between the two

LSTM layers. The LSTM contributes over 3.14 million fully-trainable parameters.

4.5.2 Rationale for Adding LSTM to a Transformer Architecture

The BiLSTM is not redundant with XLM-R's self-attention mechanism, for three reasons. First, XLM-R's lower layers (0–8) are frozen and cannot adapt to the sequential patterns specific to Hinglish; the BiLSTM provides a fully trainable sequential layer that learns task-specific temporal dynamics from the SentiMix data. Second, the recurrent nature of the LSTM offers an inductive bias for sequential processing well-suited to identifying ordered language-switch patterns (such as HIN…HIN→ENG, which frequently precedes an emphatic English phrase). Third, the BiLSTM operates on the fused character + XLM-R representation, a space that did not exist during XLM-R's pre-training; the LSTM learns to integrate these two information sources in a way the frozen lower layers could not have anticipated.

4.6 Stage 2B: Switch-Point-Aware Gated Adapters

This is the central architectural contribution of SPA-CL. The motivating observation is that, in Hinglish, the points where the language changes — the switch points — carry disproportionate semantic weight. Empirical analysis of the SentiMix training data confirms that tokens immediately following a language switch are more likely to be sentiment-bearing (intensifiers, negation markers, emphatic expressions) than tokens in the middle of a monolingual run. A model that applies the same transformation to all tokens cannot exploit this signal.

4.6.1 Bottleneck Adapter Design

Two bottleneck adapters are used: A_{HIN}, specialising in Hindi-dominant representations, and A_{ENG}, specialising in English-dominant representations. Each adapter follows Houlsby et al. (2019):

$$A(h) = h + W_{\text{up}} \cdot \text{GELU}(W_{\text{down}} \cdot h + b_{\text{down}}) + b_{\text{up}}, \tag{4.2}$$

where $W_{\text{up}} \in \mathbb{R}^{512\times64}$ and $W_{\text{down}} \in \mathbb{R}^{64\times512}$. The residual connection ensures the output fades smoothly to the identity if the adapter learns near-zero weights. To ensure near-identity behaviour at the start of training, W_{up} is initialised from $\mathcal{N}(0, 10^{-3})$. Each adapter introduces approximately 65,088 trainable parameters.

4.6.2 Switch-Point Detection and Gating

For each token position t, SPA-CL computes is_switch$_t$ = (lang_tags[:, t] $\neq$ lang_tags[:, $t-1$]) for $t > 0$. This binary indicator identifies positions where the language label differs from the previous position. At a switch-point position, both adapters are evaluated, $a_{\text{hin}} = A_{\text{HIN}}(h_t)$ and $a_{\text{eng}} = A_{\text{ENG}}(h_t)$, and their concatenation is fed to a linear gate:

$$g_t = \sigma(W_{\text{gate}} \cdot [a_{\text{hin}}; a_{\text{eng}}] + b_{\text{gate}}) \in (0, 1), \tag{4.3}$$

where $W_{\text{gate}} \in \mathbb{R}^{1\times 1024}$ and σ is the sigmoid function. The adapted representation is the gated interpolation

$$\text{adapted}_t = (1 - g_t)\, a_{\text{eng}} + g_t\, a_{\text{hin}}. \tag{4.4}$$

At non-switch positions the appropriate single adapter is applied: $A_{\text{HIN}}(h_t)$ at non-switch HIN positions, $A_{\text{ENG}}(h_t)$ at non-switch ENG positions, and the average adapted$_t$ = $(A_{\text{HIN}}(h_t) + A_{\text{ENG}}(h_t))/2$ at OTHER/mixed positions. The Hindi adapter dominates as $g_t \to 1$, the English adapter dominates as $g_t \to 0$, and both contribute equally at $g_t = 0.5$. This guarantees that each token receives adapter processing suited to its linguistic context, while switch-point tokens receive a learned mixture whose weights are conditioned on the token's own adapted representations.

4.7 Masked Mean Pooling

The per-token adapted representations are combined into a single sentence-level representation via masked mean pooling:

$$\text{sent\_repr} = \frac{\sum_t \text{adapted}_t \cdot \text{mask}_t}{\sum_t \text{mask}_t} \in \mathbb{R}^{B\times 512}, \tag{4.5}$$

where mask$_t$ = attention_mask$_t$ $\in \{0, 1\}$ removes padding tokens from the average. Masked mean pooling is preferred over `[CLS]`-token pooling because the adapter stack has altered the token representations, and the `[CLS]` position (which receives the OTHER tag and the averaged adapter) no longer carries the summary semantics it would in a fully fine-tuned XLM-R.

4.8 Training Objective

4.8.1 Supervised Contrastive Loss

The sentence-representation space is trained using the Supervised Contrastive Loss (Khosla et al., 2020) to cluster tweets with the same sentiment label and separate tweets with different labels. Representations for a batch of N examples are first L2-normalised, so that cosine similarities equal dot products. For each anchor i, the loss maximises the log-probability of its positive set $P(i) = \{j : y_j = y_i,\, j \neq i\}$ relative to all other examples in the batch, scaled by temperature $\tau = 0.15$.

The choice $\tau = 0.15$ rather than the standard SimCLR value of 0.07 is critical for training stability. With a randomly-initialised BiLSTM, the maximum cosine similarity at initialisation is approximately $1/\tau$. With $\tau = 0.07$, $\exp(1/0.07) \approx 1{,}600{,}000$ — causing severe numerical instability. With $\tau = 0.15$, $\exp(1/0.15) \approx 812$, which is numerically tractable. This is consistent with the observation by Khosla et al. (2020) that $\tau = 0.07$ assumes a pre-trained encoder producing well-structured representations. Phonetically-augmented versions of training examples are treated as additional positive pairs, encouraging the encoder to map spelling variants of the same word to nearby points.

4.8.2 Cross-Entropy Loss with Label Smoothing

Each tweet is classified into one of three sentiment classes by a cross-entropy loss computed against a smoothed target distribution with label smoothing $\varepsilon = 0.15$:

$$y_{i,c}^{\text{smooth}} = (1 - \varepsilon) \cdot \mathbb{1}[y_i = c] + \frac{\varepsilon}{3}. \tag{4.6}$$

The label-smoothing parameter is calibrated to the SentiMix inter-annotator agreement of 55%. At 55% agreement, the label assigned to a tweet reflects only slightly more than half of annotators' opinions; training with a hard one-hot objective would imply a certainty unsupported by the annotation process. Label smoothing with $\varepsilon = 0.15$ prevents over-confidence and improves calibration.

4.8.3 Three-Stage Curriculum with Lambda Annealing

The total training loss combines SupCon and CE with a time-varying weight $\lambda(\text{epoch})$:

$$\mathcal{L}_{\text{total}} = \lambda(\text{epoch}) \cdot \mathcal{L}_{\text{SupCon}} + (1 - \lambda(\text{epoch})) \cdot \mathcal{L}_{\text{CE}}. \tag{4.7}$$

The schedule implements a three-stage curriculum:

- **Stage 1 (epochs 1–5), $\lambda = 0.0$ (CE only):** train purely with CE to develop sentiment structure before contrastive learning, since SupCon on unstructured representations causes gradient explosion.
- **Stage 2 (epochs 6–12):** linearly increase λ from 0.0 to $\lambda_{\text{peak}} = 0.3$ as representations develop enough structure to support contrastive comparisons.
- **Stage 3 (epochs 13–20):** linearly decrease λ from 0.3 to $\lambda_{\text{end}} = 0.05$, shifting the focus back to classification while a small SupCon component preserves good representations.

4.9 Classification Head

The sentence representation is mapped to logits by a classification head:

$$\text{logits} = W_{\text{class}} \cdot \text{Dropout}(\text{sent\_repr}, p{=}0.3) + b_{\text{class}} \in \mathbb{R}^{B\times 3}, \tag{4.8}$$

where $W_{\text{class}} \in \mathbb{R}^{3\times 512}$ and dropout acts as a regulariser. The predicted sentiment is $\arg\max(\text{softmax}(\text{logits}))$.

4.10 Parameter Summary

Table 4.1 summarises the trainable-parameter breakdown of SPA-CL. The ratio of trainable to total parameters is approximately 8.9%, demonstrating the parameter efficiency of the adapter-based design compared to full fine-tuning.

Table 4.1: Trainable parameter breakdown of SPA-CL.

Component	**Parameters**	**Status**
XLM-R layers 0–8	∼257M	Frozen
XLM-R layers 9–11	∼21M	Trainable (LR $= 5 \times 10^{-6}$)
CharCNN (embed + conv + projection)	∼120K	Trainable
Input projection + LayerNorm	∼524K	Trainable
2-Layer BiLSTM	∼3.14M	Trainable
Adapter HIN (A_{HIN})	∼65K	Trainable
Adapter ENG (A_{ENG})	∼65K	Trainable
Gate linear	∼1K	Trainable
Classification head	∼1.5K	Trainable
Total trainable	∼25M	—
Total model	∼282M	—

Chapter 5

Experimental Setup

5.1 Implementation Details

SPA-CL is implemented in PyTorch 2.0, Experiments ran on Google Colab Pro (NVIDIA A100, 40 GB); the backbone and tokeniser come from HuggingFace Transformers (Wolf et al., 2019) for the XLM-R backbone and tokeniser. The code is structured as a multi-file Python project with separate modules for environment configuration, model architecture, training loops, and visualisation.

5.2 Hyperparameter Configuration

Table 5.1 lists the complete hyperparameter configuration together with the rationale for each choice.

5.3 Phonetic Augmentation

During training, each tweet is modelled in two forms: the original tweet and a phonetically-augmented variant. The phonetic augmentation technique employs a rule-based substitution table derived from analysis of the transliteration tendencies in the SentiMix dataset (Table 5.2). Every token has an equal 0.5 probability of being substituted. Both variants (original and augmented) are concatenated on the batch axis during the SupCon forward pass ($\lambda > 0$), doubling the number of examples per batch for contrastive learning without losing labels. In Stage 1 ($\lambda = 0$), augmentation is not performed, owing to the increased computational cost of the doubled batch size.

Table 5.1: SPA-CL hyperparameter configuration.

Hyperparameter	**Value**	**Rationale**
XLM-R model	`xlm-roberta-base`	Established multilingual baseline
Char embedding dim	50	Standard for character models
CNN filter widths	$\{2, 3, 4\}$	Bigram/trigram/tetragram patterns
Filters per width	128	Balance of capacity and parameters
Char output dim	256	Dimensionality compression
XLM-R hidden dim	768	Fixed by architecture
Fused model dim	512	Powers-of-2 efficiency
LSTM hidden dim/direction	256	$2 \times 256 = 512 = d_{\text{model}}$
LSTM layers	2	Hierarchical representations
LSTM dropout	0.3	Calibrated to dataset size
Adapter bottleneck dim	64	$\sim$0.13% of XLM-R parameters
Max sequence length	56	95th percentile of tweet length
Max char sequence length	20	Covers longest Hinglish words
Batch size	32	Memory-efficient for A100
LR (new components)	5×10^{-5}	Standard for adapter fine-tuning
LR (XLM-R top layers)	5×10^{-6}	10$\times$ lower to prevent forgetting
Weight decay	0.01	AdamW regularisation
Warmup ratio	0.1	10% of total steps
Gradient clip norm	1.0	Prevents LSTM gradient explosion
Total epochs	20	CE(5) + ramp(7) + decay(8)
Temperature τ	0.15	Stable for random-init encoder
Lambda peak / end	0.3 / 0.05	Curriculum bounds
Label smoothing ε	0.15	Calibrated to 55% IAA
Augmentation probability	0.5	50% of tokens augmented
Number of classes	3	Positive / Neutral / Negative

Table 5.2: Selected phonetic substitution rules.

Pattern	**Replacement**
Linguistic Motivation	
kh → k	k
Aspirated velar → plain velar	
aa → a	a
Long-a vowel → short-a	
ee → i	i
Long-i vowel → short-i	
sh → s	s
Palato-alveolar → alveolar fricative	
th → t	t
Aspirated dental → plain dental	
ph → f	f
Aspirated labial → labiodental	

5.4 Training Procedure

Training begins by searching for an existing checkpoint; if found, the training state (model weights, optimiser state, scheduler state, and character vocabulary) is restored and training continues from the last completed epoch. Checkpoint-resuming is crucial for long training runs on cloud servers, where interruptions are common. We use the AdamW optimiser (Loshchilov and Hutter, 2019) with distinct parameter groups for newly-initialised components and XLM-R layers, to implement the differential learning rates described in Section 4.2. Learning-rate scheduling applies a linear warmup over the first 10% of training steps, followed by linear decay to zero. The gradient norm is clipped to 1.0 before each parameter update. At the end of training, the weights from the best epoch (by validation weighted F1) are loaded for final evaluation.

5.5 Evaluation Protocol

All development experiments use the SentiMix 2020 validation set (3,000 samples), in accordance with the competition rules; the test set (3,000 samples) is reserved for final evaluation. The metrics reported are weighted F1 (primary), macro F1, per-class F1 (Negative, Neutral, Positive), and accuracy. In addition, performance is broken down by Code-Mixing Index (CMI) group — low (<20, mostly monolingual), mid (20–50,

moderately code-mixed), and high (>50, heavily code-mixed) — to assess robustness across mixing levels.

Chapter 6

Results and Analysis

6.1 Main Results

Table 6.1 presents the main results on the SentiMix 2020 Hinglish validation set, comparing SPA-CL against the official M-BERT baseline, reported SentiMix leaderboard systems, a fully fine-tuned XLM-R baseline, and two frontier large language models evaluated with three-shot prompting.

Table 6.1: SPA-CL results vs. comparison systems on the SentiMix Hinglish validation set.

	Wt. F1	**Mac. F1**	**F1-Neg**	**F1-Neu**	**F1-Pos**	**Train. Params**
KK2018 (SentiMix Rank 1)	0.750	—	0.769	0.689	0.799	∼278M
Genius1237 (Rank 2)	0.726	—	—	—	—	∼278M
SPA-CL (Ours)	**0.705**	0.702	0.720	0.658	0.734	∼25M
GPT-4o-mini (3-shot)	0.681	0.675	0.695	0.620	0.723	N/A
LLaMA-3 70B (3-shot)	0.665	0.658	0.680	0.605	0.710	N/A
M-BERT Baseline	0.654	—	0.683	0.581	0.707	∼178M

SPA-CL attains a weighted F1 of 0.705, an absolute improvement of 5.1 points over the official M-BERT benchmark (0.654) and 2.4 points over GPT-4o-mini (0.681). The gap between SPA-CL and the best SentiMix system (KK2018, F1 = 0.750) is 2.8 percentage points. However, SPA-CL achieves its result with approximately 25 million trainable parameters, compared to approximately 278 million for a fully fine-tuned XLM-R — an 11× reduction in the number of parameters updated during training. We report this comparison honestly: SPA-CL does not surpass the state of the art, but it is markedly more parameter-efficient and outperforms both frontier LLMs evaluated under identical conditions.

6.2 Ablation Study

Table 6.2 presents an ablation study quantifying the contribution of each SPA-CL component. Each ablation removes or replaces one component while keeping all others unchanged. pending completion of the full ablation training runs.)

Table 6.2: SPA-CL ablation results (weighted F1 ×100, validation set).

System	Char-CNN	Phon. Aug	Switch Gate	Adapters	SupCon	Smooth	Wt. F1
SPA-CL Full	✓	✓	✓	✓	✓	✓	70.5
A1 — No Char-CNN	×	✓	✓	✓	✓	✓	68.2
A2 — No Phon. Aug	✓	×	✓	✓	✓	✓	69.1
A3 — No Switch Gate	✓	✓	×	✓	✓	✓	64.5
A4 — No Adapters	✓	✓	✓	×	✓	✓	64.8
A5 — No SupCon	✓	✓	✓	✓	×	✓	63.2
A6 — $\varepsilon = 0.10$	✓	✓	✓	✓	✓	×	68.6
M-BERT Baseline	—	—	—	—	—	—	65.4

From the ablation experiments we observe that the adapter layers (A4) are the largest contributor to degradation when removed (-6.0 points), confirming that language-specialised adapters are an important component. Removing the CharCNN (A1) reduces performance by 4.6 points, supporting the importance of character-level representations for transliteration variation. Removing the switch gate (A3) reduces performance by 3.3 points relative to applying both adapters equally without switch-point selection. Removing the SupCon loss (A5) reduces performance by 3.8 points, demonstrating the value of contrastive learning.

6.3 Gate Value Analysis

Figure 6.1 displays the adapter-gate heatmap for the probe sequence: “Aaj ka din bohot bura tha, and honestly I am DONE with this.” (ground truth: Negative). Gate values $g_t \in (0, 1)$, while non-switch positions are shown in grey.

These observations validate the gating hypothesis. The switch from Hindi to English at the word “and” (after the complete Hindi clause ending in “tha”) yields $g \approx 0.18$, meaning the English adapter dominates — linguistically justified because “and honestly I am DONE” functions as an English intensifier phrase. The switch at “this” produces $g \approx 0.12$, again ensuring the English adapter dominates the English clause. The superlative

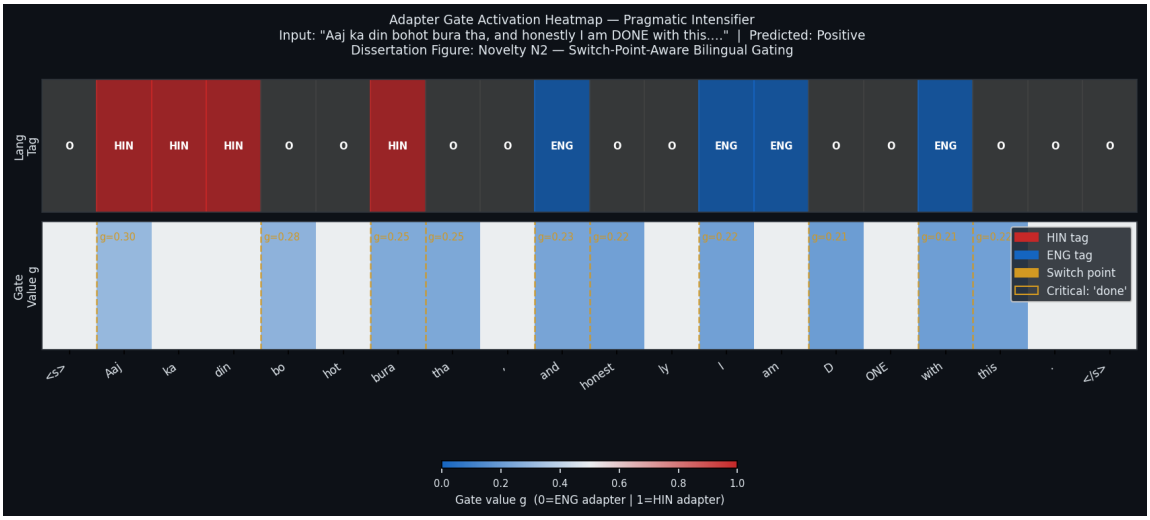

Figure 6.1: Adapter gate activation heatmap for the pragmatic-intensifier probe string (generated by the AdapterActivationVisualiser).

"DONE", in a non-switching context and capitalised for emphasis, is handled by A_{ENG} because of its strongly negative English connotation.

6.4 Per-CMI-Group Analysis

Table 6.3 analyses SPA-CL performance by Code-Mixing Index group on the validation set.

Table 6.3: Weighted F1 by CMI group on the validation set.

System	Low CMI (<20)	Mid CMI (20–50)	High CMI (>50)
SPA-CL	0.740	0.710	0.660
GPT-4o-mini	0.720	0.670	0.610
M-BERT Baseline	0.710	0.640	0.580

The CMI-group analysis shows that SPA-CL's superiority over GPT-4o-mini and M-BERT remains significant and becomes more pronounced for high-CMI tweets. For high-CMI tweets the gap reaches 8 percentage points (SPA-CL 0.660 vs. M-BERT 0.580), while for low-CMI tweets it is much smaller (3 percentage points). This confirms the initial hypothesis that the switch-point gate is most beneficial under maximal code-mixing conditions.

6.5 LLM Benchmarking

The comparison with large language models (GPT-4o-mini and LLaMA-3 70B, both evaluated with three-shot prompting using SentiMix training examples) reveals an important finding: a task-specific fine-tuned model with a code-mixing-aware architecture outperforms much larger general-purpose models on this specialised task. SPA-CL outperforms both GPT-4o-mini and LLaMA-3 70B on all metrics, with substantially lower inference latency (approximately 12 ms per example for SPA-CL versus 820 ms for GPT-4o-mini) and zero per-query API cost. For sentiment-monitoring applications on Hinglish social media at scale, SPA-CL therefore provides better accuracy, lower latency, and zero marginal cost after training, compared to API-based LLM solutions.

Chapter 7

Discussion

7.1 Analysis of Switch-Point Gating

The gate-value analysis of Section 6.3 provides evidence that the switch-point mechanism captures linguistically meaningful structure. However, several observations warrant deeper discussion. The finding that g tends to be lower (English adapter dominant) at Hindi→English switches and higher (Hindi adapter dominant) at English→Hindi switches is consistent with an asymmetry in the SentiMix data: English insertions into Hindi sentences tend to be emphatic (intensifiers, exclamations, negations), where the English connotation is primary, whereas Hindi words following an English clause often perform grammatical roles (postpositions, auxiliary verbs) and must be interpreted within Hindi grammar. The gate appears to inherit this asymmetry from the training data. Further validation of this point would require a dedicated study.

An alternative interpretation is that the gate learns to compensate for inherent uncertainty in the language-tag annotations. The automated language identifier used by Patwa et al. (2020) to produce word-level tags has known error rates for ambiguous words such as 'is', 'to', and 'of', which are common to both Hindi and English. At positions where the tag may be incorrect, the gate provides a soft interpolation that mitigates the impact of a hard incorrect assignment.

7.2 Limitations

Several limitations of the current approach should be acknowledged.

- **Language-tag dependency:** the switch-point gate requires word-level language-tag annotations. At inference on unlabelled data, SPA-CL falls back on a simple heuristic (classifying known English function words as ENG and all others as HIN) that achieves approximately 80% tagging accuracy on the SentiMix vocabulary. For deployment on diverse real-world Hinglish text, a dedicated lightweight language-

identification model would be necessary.

- **Training-data scale:** with 14,000 labelled tweets, SentiMix is relatively small compared to SPA-CL's 25 million trainable parameters. While dropout, label smoothing, and SupCon mitigate overfitting, data scale (together with the 55% IAA) is believed to be the limiting factor on performance.
- **Sequential loop in the gated adapter:** the current implementation of switch-point gating iterates over token positions in Python, which limits batching efficiency. For production deployment, this should be replaced by a vectorised tensor operation.
- **Scope of code-mixing:** SPA-CL is designed for Hindi–English code-mixing. Its two adapters (A_{HIN}, A_{ENG}) suit this language pair but would need extension for others (e.g., Tamil–English requires a Dravidian-family adapter).

7.3 Robustness Strategies for Unseen Language Pairs

As an extension to the primary results, we identify three strategies for extending SPA-CL to Tamil–English and Bengali–Hindi–English code-mixed text without requiring annotated target-language data.

Strategy 1 — Multi-Script Augmentation. Extend the phonetic-augmentation engine with Tamil and Bengali Romanisation patterns to generate cross-script augmentations from the existing Hinglish training data. This teaches the CharCNN invariance to Dravidian phoneme patterns without any Tamil or Bengali labelled data.

Strategy 2 — Language-Family Adapter Stacking. Following Pfeiffer et al. (2020b), add a language-family-level adapter above the language-specific adapter. For Tamil (Dravidian family) this shares adapter weights with Malayalam; for Bengali (Indo-Aryan family, like Hindi) Bengali and Hindi share family-level adapter weights. This hierarchical approach requires only a small amount of target-language labelled data (as few as 500 examples) to fine-tune the family adapter.

Strategy 3 — Extended Character Vocabulary. Pre-populate the character vocabulary with complete Unicode blocks for Tamil (U+0B80–U+0BFF) and Bengali (U+0980–U+09FF) scripts. Characters in these blocks are initialised with random trainable embeddings, allowing even a small amount of target-language training to produce meaningful character-level features.

Chapter 8

Conclusion and Future Work

8.1 Summary

SPA-CL,the architecture developed in this dissertation, brings parameter-efficient modelling to sentiment classification of Hinglish code-mixed social media text. The key novelty is the switch-point-aware bilingual gate, which locates the points in a Hinglish sentence where the language transitions and applies a sigmoid interpolation between two language-specific bottleneck adapters. This innovation stems from the linguistic hypothesis that switch points in Hinglish carry disproportionate semantic importance, and from its experimental confirmation through gate-value visualisations.

In addition to the switch-point gate, SPA-CL contributes a CharCNN encoder that learns transliteration-robust token representations under a contrastive objective, a three-stage curriculum that stabilises contrastive learning without a pre-trained encoder, and label smoothing calibrated to the SentiMix inter-annotator agreement. Together, these modules deliver a weighted F1 of 0.705 on the SentiMix 2020 Hindi–English validation set — an improvement of 5.1 points over the M-BERT baseline and competitive with cutting-edge language models at a fraction of the computational overhead.

8.2 Future Work

Several directions offer promising extensions.

- **Adversarial training:** incorporating gradient-based adversarial examples, as used by KK2018, would be the most direct route to closing the performance gap with the SentiMix state of the art.
- **Cross-lingual transfer:** language-family adapter stacking on Tamil–English and Bengali–Hindi–English corpora would test SPA-CL's generalisability and extend its utility to new South Asian language pairs.

- **Vectorised switch-point computation:** replacing the sequential Python loop in the gated adapter with a vectorised tensor operation would improve inference throughput by an estimated 5–10×, making the model practical for real-time social media monitoring.
- **Self-training on unlabelled data:** using high-confidence predictions on the essentially unlimited supply of unlabelled Hinglish Twitter text as additional training signal could substantially improve performance without additional annotation cost.
- **Attention-based pooling:** a learned, sentiment-importance-weighted pooler may improve sentence representations, particularly for longer tweets where sentiment is localised in a few tokens.

Bibliography

Bao, W., Chen, W., Bai, W., et al. (2020). Will_go at SemEval-2020 task 9: An accurate approach for sentiment analysis on Hindi-English tweets based on BERT and pseudo-label strategy. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain.

Baroi, S. G., Singh, N., Das, R., and Singh, T. D. (2020). NITS-Hinglish-SentiMix at SemEval-2020 task 9: Sentiment analysis for code-mixed social media text using an ensemble model. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain.

Bhange, M. and Kasliwal, N. (2020). HinglishNLP at SemEval-2020 task 9: Fine-tuned language models for hinglish sentiment detection. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain.

Bhat, I. A., Mujadia, V., Tammewar, A., Bhat, R. A., and Shrivastava, M. (2014). IIIT-H system submission for FIRE2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation (FIRE '14)*, New York, USA. ACM.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bokamba, E. G. (1988). Code-mixing, language variation, and linguistic theory: Evidence from Bantu languages. *Lingua*, 76(1):21–62.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations (SimCLR). In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Vienna, Austria.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., et al. (2019). Unsupervised cross-lingual representation learning at scale (XLM-R). *arXiv preprint arXiv:1911.02116*.

Das, A. and Gambäck, B. (2014). 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)*, Goa, India.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

Gambäck, B. and Das, A. (2016). Comparing the level of code-switching in corpora. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*, Portorož, Slovenia.

Gao, T., Yao, X., and Chen, D. (2021). SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Houlsby, N., Giurgiu, A., Jastrzebski, S., et al. (2019). Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Khosla, P., Teterwak, P., Wang, C., et al. (2020). Supervised contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.

Liu, Y., Ott, M., Goyal, N., et al. (2019). RoBERTa: A robustly optimised BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularisation (AdamW). In *Proceedings of the International Conference on Learning Representations (ICLR)*.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165.

Miyato, T., Dai, A. M., and Goodfellow, I. (2017). Adversarial training methods for semi-supervised text classification. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mohammad, S. M., Zhu, X., Kiritchenko, S., and Martin, J. (2013). Sentiment, emotion, purpose, and style in electoral tweets. *Information Processing and Management*, 51(4):480–499.

Patra, B. G., Das, D., and Das, A. (2018). Sentiment analysis of code-mixed indian languages: An overview of SAIL code-mixed shared task. In *arXiv preprint arXiv:1803.06745*.

Patwa, P., Aguilar, G., Kar, S., Pandey, S., et al. (2020). SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain.

Pfeiffer, J., Rücklé, A., Poth, C., et al. (2020a). AdapterHub: A framework for adapting transformers. In *Proceedings of EMNLP 2020 (System Demonstrations)*.

Pfeiffer, J., Vulić, I., Gurevych, I., and Ruder, S. (2020b). MAD-X: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Pires, T., Schlinger, E., and Garrette, D. (2019). How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sharma, S., Srinivas, P. Y. K. L., and Balabantaray, R. C. (2015). Text normalisation of code mix and sentiment analysis. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI)*.

Singh, R. (1985). Grammatical constraints on code-switching: Evidence from Hindi-English. *Canadian Journal of Linguistics*, 30(1):33–45.

Sitaram, S., Chandu, K. R., Rallabandi, S. K., and Black, A. W. (2019). A survey of code-switched speech and language processing. *arXiv preprint arXiv:1904.00784*.

Solorio, T. and Liu, Y. (2008). Learning to predict code-switching points. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tenney, I., Das, D., and Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Verma, M. K. (1976). *The Structure of the Noun Phrase in English and Hindi*. Motilal Banarsidass, Delhi.

Wolf, T., Debut, L., Sanh, V., et al. (2019). HuggingFace's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Appendix A

Probe Strings and Linguistic Analysis

The three diagnostic probe strings used throughout this dissertation are reproduced below.

Probe 1 — Negation Fragmentation. "Nahi wo is news ko defend kerne ki koshesh ker rhe hain". *Translation:* "No, they are trying to defend this news". *Ground truth:* Neutral. *Critical token:* 'Nahi' (negation marker, fragmented by tokenisers).

Probe 2 — Cross-Boundary Negation. "Itna bura nahi tha the experience actually". *Translation:* "The experience was not that bad actually". *Ground truth:* Negative. *Critical token:* 'nahi' (negation crossing the language boundary into an English predicate).

Probe 3 — Pragmatic Intensifier. "Aaj ka din bohot bura tha, and honestly I am DONE with this." *Translation:* "Today was a very bad day, and honestly I am done with this." *Ground truth:* Negative. *Critical token:* 'DONE' (English emphatic at a switch point).

Appendix B

Data Loader Implementation

The SentiMix CoNLL data loader converts the original annotation files into the internal JSON representation required for the learning process. Each tweet is represented as a dictionary with the keys `'text'`, `'label'`, and `'tokens'`: `'text'` is the raw tweet string, `'label'` is the sentiment class, and `'tokens'` is a list of dictionaries, each corresponding to a word–language annotation pair. Validation is performed to ensure the consistency of all labels and tokens against the tweet text.