# NEPTUN: Normalization for Romanized Nepali Sentiment Analysis

Chandra Prakash Chaudahry[1], Basanta Joshi[2], Aman Shakya[3], Santosh Giri[4,*]
[1,2,3,4]Institute of Engineering, Tribhuvan University (TU), Lalitpur, Nepal
*Corresponding author: santoshgiri@pcampus.edu.np

**Abstract**

The growth of e-commerce has led to rise in user-generated reviews, many of which in Nepal are written in Romanized Nepali a non-standard form with inconsistent spelling, grammar and code-switching with English. These irregularities challenge traditional sentiment analysis methods. This study presents NEPTUN(NEpali Phonetic Translation-Based Unified Normalization), a novel module for normalizing Romanized Nepali, NEPTUN uses phonetic transliteration to map Romanized words to Devnagari, verifies them via a Nepali dictionary, and then back-transliterates them into standardized Romanized forms. It also applies frequency-based filtering to retain common variants, improving consistency. While similar techniques exist for Romanized Hindi and Urdu, NEPTUN is the first tailored to Romanized Nepali. Its effectiveness was tested using various sentiment classifiers- Logistic Regression, Naive Bayes, K-Nearest Neighbors, and BERT. NEPTUN-enhanced preprocessing improved model accuracy, with BERT achieving the highest at 87.56%. These results emphasize the need for domain-specific preprocessing in low-resource language like Nepali.

**Keywords:** Romanize Nepali, Phonetic Normalization, Sentiment Analysis, NEPTUN, Text preprocessing

## 1. Introduction

With the rapid rise of e-commerce platforms, user-generated product reviews have become a crucial source of customer feedback. These reviews often reflect consumer sentiment regarding product quality, delivery efficiency, and overall satisfaction, offering valuable insights for both buyers and sellers. Sentiment analysis, a widely used Natural Language Processing (NLP) technique, enables the automated interpretation of such textual data to classify user opinions as positive, negative, or neutral. While sentiment analysis has seen significant success in high-resource languages, its effectiveness diminishes when applied to low-resource and linguistically diverse contexts. One such underexplored area is Romanized Nepali [18]language text written using the Roman script. This form of text, common in social media and e-commerce platforms, poses unique challenges due to inconsistent spellings (e.g.ramro, ramroo, raamro), grammatical irregularities, and frequent code-switching with English. These linguistic inconsistencies hinder the performance of traditional sentiment analysis models.

Although similar issues have been addressed in Romanized Hindi and Urdu [17], [19] using phonetic normalization techniques such as TERUN, these methods rely on language specific rules and lexicons that do not generalize well to Nepali. For example, TERUN's reliance on Hindi dictionaries leads to misinterpretation of valid Nepali words like ramro, which are absent in Hindi corpora. Moreover, widely used phonetic algorithms such as Soundex and Metaphone often over normalize semantically distinct words that are phonetically similar, such as choto, chuddyo, and chodyo. To address these challenges, we propose NEPTUN (NEpali Phonetic Transliteration-Based Unified Normalization) a tailored phonetic normalization module designed specifically for Romanized Nepali. NEPTUN transliterates Romanized tokens into Devanagari using phonetic rules, validates them against a Nepali dictionary, and then back-transliterates them into standardized Romanized forms. This reduces lexical variation, enhances input consistency, and improves the performance of downstream sentiment classifiers.

The originality of this work lies in the design of NEPTUN, a normalization approach specifically tailored for Romanized Nepali - a language variant that has received little to no dedicated computaional treatment. Unlike prior phonetic normalization system such as TERUN, which are adapted from Hindi and rely heavily on language specific lexicons, NEPTUN introduces rule-based phonetic mapping and dictionary validation optimized for Nepali phonology and orthography.The main goal of this study is to enhance sentiment analysis on Romanized Nepali user reviews by addressing language specific inconsistencies through NEPTUN. Specifically, the proposed approach develops a dedicated phonetic normalization pipeline, reduces lexical variation and improves text consistency, and evaluates the impact of normalization on multiple sentiment classification models, including Logistic Regression, Naive Bayes, K-Nearest Neighbors(KNN) and BERT. This integration of language-aware normalization and sentiment modeling represents a novel contribution to low resource language processing research.

## 2. Literature Review

Sentiment Analysis (SA) remains a core task in Natural Language Processing (NLP), widely applied across domains such as product reviews, political discourse, and financial forecasting [9], [15]. The granularity of SA tasks varies from document-level and sentence-level classification to aspect-based sentiment extraction offering both high-level and nu-

Article History:
Submission: 2025-06-11, Revision: 2025-10-28, Acceptance: 2025-11-17

anced interpretations of opinionated text [1], [6]. While early SA models were predominantly lexicon-based, recent advances leverage machine learning and deep learning techniques. Nevertheless, cross-domain and multilingual performance remains inconsistent due to factors such as domain shift, data sparsity, and language-specific characteristics [2], [5].

In multilingual and low-resource contexts, text normalization emerges as a critical preprocessing step especially for Romanized scripts where nonstandard spelling, informal language, and code switching are prevalent. In Roman Urdu, [10] demonstrated improved performance using a Self-Attention BiLSTM architecture, while [8] introduced an unsupervised phonetic clustering method to address lexical variability. [16] synthesized various normalization techniques, underscoring their importance in mitigating input noise and enhancing model robustness.

Phonetic normalization techniques are particularly impactful in Romanized and code-mixed settings, where inconsistent spelling variations often degrade the performance of natural language processing (NLP) systems. [17] proposed a novel technique called transliteration based Encoding for Roman Hindi/Urdu text Normalization (TERUN), designed to address lexical variability in Roman scripts. TERUN employs a three-stage architecture: (i) a transliteration based encoder that generates hash codes for variant spellings, (ii) a filter module that discards irrelevant or noisy encodings, and (iii) a hash code ranker that selects the most contextually appropriate variant. This framework facilitates robust normalization by effectively ranking plausible spellings, thereby improving downstream tasks such as sentiment classification and intent detection.

Similarly, [19] address the linguistic challenges inherent in code-mixed and low resource scenarios, where phonetic inconsistencies often disrupt semantic interpretation. Their proposed normalization pipeline emphasizes phonetic similarity to map diverse orthographic forms to a unified representation. The approach includes a computationally efficient normalization module that standardizes spelling variations while preserving semantic integrity. Additionally, the authors introduce a scalable data pipeline capable of generalizing normalization across multiple languages in code-mixed data. Empirical results demonstrate notable gains in downstream applications like intent classification, slot-filling, and response generation, underscoring the critical role of normalization in enhancing NLP model performance in noisy, multilingual environments.

Complementary research on noisy social media text has explored hybrid normalization pipelines. [12] segmented tokens into phonetic blocks and applied machine translation for normalization in short text like tweets and SMS. [12] proposed a cognitively inspired model combining letter transformation, phonetic similarity, and visual priming. Meanwhile, [11] tackled named entity recognition in tweets using a semi-supervised approach combining KNN pre-labeling with CRF for sequential tagging.

Recent studies in low-resource NLP have increasingly focused on improving normalization, adaptation, and data efficiency for underrepresented languages. [4] conducted a survey of NLP progress in Sino-Tibetan low-resource languages, identifying normalization and data-centric strategies as essential for model improvement. [?] evaluated sequence-to-sequence models for spoken language normalization of Slovenian, showing that neural normalization notably improves processing of dialectal and spontaneous speech. [22] investigated sentiment analysis for Slovene, Croatian, and Slovak, demonstrating that data augmentation combined with normalization enhances sentiment classification accuracy in low-resource settings. [21] explored large language model adaptation for Ukrainian, illustrating how prompt-based fine-tuning can significantly advance named entity recognition in morphologically rich, low-resource languages. Collectively, these works highlight a shift toward integrating normalization, augmentation, and efficient multilingual adaptation to strengthen NLP performance in low-resource settings—yet few explicitly target Romanized or informal scripts such as Romanized Nepali, where normalization remains a core unmet challenge. Despite advancements in Romanized text normalization across languages, work specific to Romanized Nepali remains sparse. Singh et al., 2024 constructed an aspect-based sentiment dataset for social media, while [18] leveraged BERT for Romanized Nepali ecommerce reviews. The ONRMD dataset Anonymous, 2024 targets offensive language detection using multilingual models. However, these studies primarily focus on model performance rather than upstream preprocessing.

Critically, few efforts have explored phonetic or domain-aware normalization techniques tailored to Romanized Nepali particularly for informal ecommerce texts rife with spelling inconsistencies, dialectal mixing, and domain-specific jargon. This presents a clear research gap: the need for a systematic normalization framework that addresses these linguistic challenges and integrates seamlessly into sentiment classification pipelines for low-resource and morphologically rich languages like Nepali.

## 3. Methodology

This chapter outlines the methodology adopted to enhance sentiment analysis for Romanized Nepali reviews in the e-commerce context. The approach addresses key challenges such as inconsistent spellings, mixed-language usage, and noisy data through structured data collection and preprocessing.

### 3.1 Data Collection

Multiple datasets were used to support the normalization and sentiment analysis tasks:

#### 3.1.1 Sentiment Analysis Dataset

The primary dataset includes over 30,000 user generated product reviews from Daraz, covering diverse categories like electronics, fashion, and home appliances. Initial data was obtained from prior research [18] and expanded. Irrelevant or incomplete reviews were filtered out.

#### 3.1.2 Filtered Dataset (Nepali Dictionary)

A frequency-based word list was derived from 39,000 cleaned Nepali Wikipedia articles (sourced from Kaggle), resulting in a vocabulary of 240,726 words. This list supports text normalization by highlighting common transliterations and linguistic variations.

#### 3.1.3 Manually Annotated Romanized to Devanagari Mapping Dataset

To evaluate normalization accuracy, a gold standard dataset of 1,000 Romanized Nepali words was manually mapped to Devanagari. The words were extracted from user reviews and annotated by three native speakers, with disagreements resolved via adjudication.

### 3.2 Data Preprocessing

Effective preprocessing is crucial to clean and standardize Romanized Nepali text for analysis. The steps include:

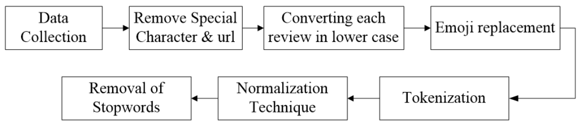


**Figure 1.** Data Preprocessing Pipeline

**Text Cleaning:**

Irrelevant Content Removal: Regular expressions were used to remove URLs, special characters, excessive punctuation, typographic symbols, and unicode art that do not contribute to sentiment.

Emoji Replacement: Sentiment-bearing emojis were replaced with explicit tags:

- Positive emojis→ positive emoji
- Negative emojis→negative emoji

Case Normalization: All text was converted to lowercase to reduce redundancy due to inconsistent capitalization.

**Tokenization**

NLTK's word tokenizer was used to segment reviews into tokens, handling Romanized Nepali and mixed English-Nepali content uniformly by relying on whitespace and punctuation-based boundaries.

**Stop word Removal**

Both Nepali and English stop words were removed to reduce noise, except for negation terms (e.g., not, don't) which are crucial for detecting sentiment polarity. Examples:

- Nepali: ma, timi, tapai, hami, yo, tyo
- English: a, about, above, after, again, against, all

The preprocessed text is then passed through the NEPTUN normalization module before being used in sentiment classification.

### 3.3 NEPTUN (NEpali Phonetic Transliteration Based Unified Normalization)

#### 3.3.1 Capture Code Mixing

The first step of the NEPTUN normalization module focuses on identifying code-mixed tokens, specifically distinguishing Romanized Nepali words from English words. To achieve this, a WordNet based English dictionary is used to verify whether a token belongs to the English language. If the token is found in the dictionary, it is classified as English and retained without modification. Tokens not recognized as English are considered potential Romanized Nepali words and are passed to the next stage of the NEPTUN module for further normalization processing. Since the dataset used in this research is code-mixed comprising both Romanized Nepali and English tokens extracted from user reviews this step ensures that English words are preserved while only the Romanized Nepali tokens are normalized. This reflects the natural linguistic blending common in online Nepali communication.

#### 3.3.2 Phonetic Transliteration to Devanagari

In NEPTUN's second stage, Romanized Nepali tokens detected during code-mixing are transliterated into Devanagari using a phonetic approach. The Indic Transliteration Library [14] is used for base transliteration, enhanced by NEPTUN's custom variation generator to handle inconsistent Romanized spellings.

The variation generator applies the following steps:

**Normalization** Reduces repeated characters for consistency while preserving phonetics.

Example: aaaa → aa

**Phonetic Substitution** Uses a variation map to account for common vowel/consonant alternatives.

Examples: aa →a, aa sh → s, shh

**Consonant Mapping** Handles phonetic overlaps in Nepali pronunciation.

Examples: chh → x, ch pha →fa

**Expansion and Tokenization** Splits tokens into subcomponents, applies substitutions, and generates valid Romanized variants.

**Devanagari Conversion** All variants are transliterated to Devanagari using the ITRANS scheme.

**Table 1.** Example of Phonetic Normalization between Romanized and Devanagari Variants of "Bigreko"

| **Word** | **bigreko** |
|---|---|
| Romanized variations | ['biigreeko', 'bigreeko', 'bigreko', 'biigreko'] |
| Devanagari variations | ['बीग्रीको', 'बिग्रीको', 'बिग्रेको ', बीग्रेको] |

#### 3.3.3 Candidate Selection (Devanagari Phoneme)

After generating multiple phonetic transliterations, the NEPTUN module proceeds to select the most appropriate Devanagari variant for each Romanized token. This candidate selection stage ensures that the final normalized output aligns with commonly used and linguistically accurate Nepali words.

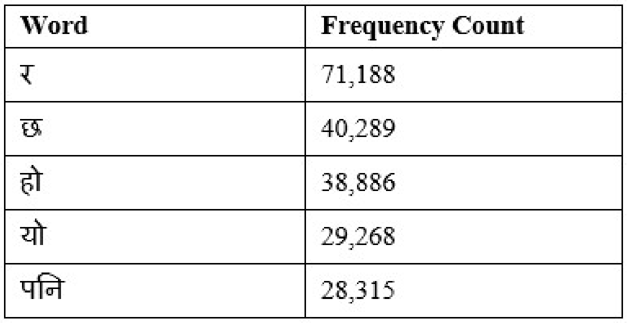
**Reference Dataset Construction and Preprocessing**

Due to the lack of a large-scale Nepali dictionary corpus, NEPTUN builds its own reference dataset from Nepali Wikipedia articles. These articles are cleaned by removing Latin characters, numbers, special symbols, and Devanagari-specific punctuation, retaining only Devanagari script and spaces. Words are tokenized using space-based segmentation to preserve integrity.

**Word Frequency Dictionary Generation**

Text from multiple Wikipedia files is processed using Count Vectorizer to compute word frequencies. The resulting dictionary is sorted by frequency, prioritizing common words.

**Table 2.** Example of Devanagari Word Frequency Dictionary Generated from Nepali Wikipedia Corpus



| Word | Frequency Count |
|---|---|
| र | 71,188 |
| छ | 40,289 |
| हो | 38,886 |
| यो | 29,268 |
| पनि | 28,315 |

**Dataset Significance**

The reference dataset improves transliteration accuracy by:

- Reflecting real-world language use.
- Supporting statistical candidate selection.
- Filtering out rare or incorrect words.

**Feature Extraction and Similarity Computation**

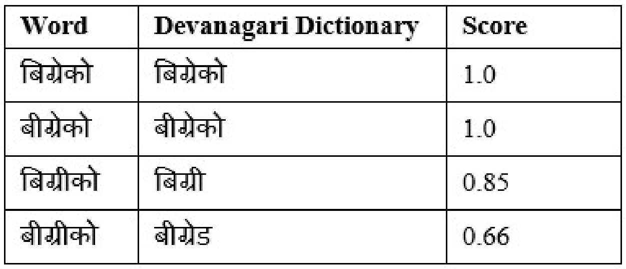
Candidate and dictionary words are vectorized using character-level TF-IDF (bigrams to four grams). Cosine similarity scores are computed to quantify phonetic closeness.

**Best Match Selection**

The candidate with the highest similarity score is selected as the normalized output.

Example: bigreko

**Table 3.** Example of Candidate Selection Based on TF-IDF Similarity Scores



| Word | Devanagari Dictionary | Score |
|---|---|---|
| बिग्रेको | बिग्रेको | 1.0 |
| बीग्रेको | बीग्रेको | 1.0 |
| बिग्रीको | बिग्री | 0.85 |
| बीग्रीको | बीग्रेड | 0.66 |

### 3.3.4 Back-Transliteration to Standardized Romanized Nepali

After converting tokens to Devanagari, NEPTUN performs back-transliteration to obtain consistent Romanized Nepali forms. This step addresses the variability in Romanized spellings and ensures uniform representation for downstream tasks. For example, the word ”ramro” may appear as “raamro,” “ramrooo,” or “ramro.” NEPTUN maps each Devanagari token to its Romanized variants and selects the most frequent form based on real world usage from a sentiment dataset.

**Process Overview**

- Token Mapping: Each Devanagari word is associated with all observed Romanized variants.
- Frequency Counting: The usage frequency of each variant is calculated.
- Form Selection: The variant with the highest frequency is chosen as the standardized form.

Example

- Devanagari: ”ramro”
- Variants: ’raamro’, ’ramrooo’, ’ramroooo’, ’ramro’
- Frequencies: ’raamro’: 16, ’ramrooo’: 3, ’ramroooo’: 2, ’ramro’: 1159
- Standardized: ramro

This approach ensures all Romanized forms converge to a single, frequent, and normalized version, enabling consistent input for downstream tasks. To facilitate reproducibility, the source code for the NEPTUN normalization pipeline and sentiment analysis experiments is available at NEPTUN GitHub Repository.

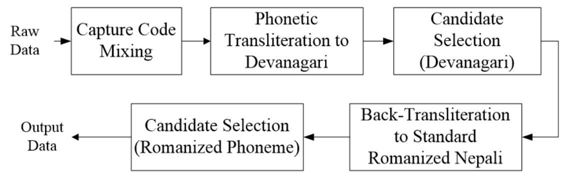

Raw Data
Capture Code Mixing
Phonetic Transliteration to Devanagari
Candidate Selection (Devanagari)
Back-Transliteration to Standard Romanized Nepali
Candidate Selection (Romanized Phoneme)
Output Data

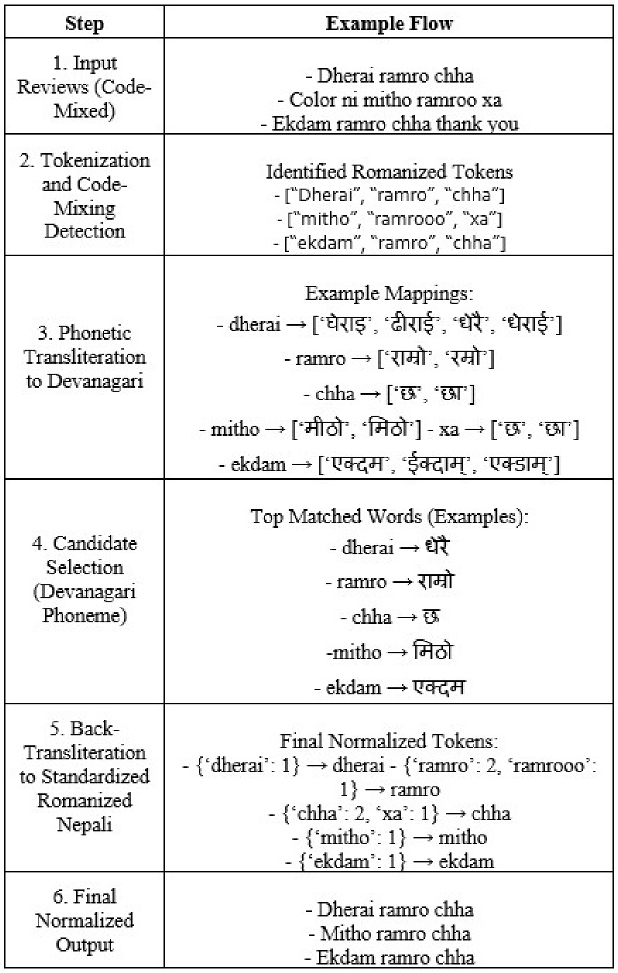
**Figure 2.** NEPTUN Normalization

### 3.4 Integration of NEPTUN with sentiment Analysis)

After normalization, the standardized Romanized Nepali text is used as input for sentiment classification. The integration of NEPTUN into Sentiment analysis pipeline ensures consistent token representation and improved lexical matching with pre-trained embbeddings. This process minimizes variations caused by inconsistent spellings and mixed-language content, allowing sentiment models such as Logistic Regression, Naïve Bayes, KNN, and BERT to perform more effectively.

To demonstrate how NEPTUN normalization integrates into the sentiment analysis workflow, three sample reviews from the Daraz dataset are processed step-by-step. The examples illustrate how NEPTUN handles code-mixed text, applies phonetic transliteration to Devanagari, performs candidate selection, and finally back-transliterates tokens to standardized Romanized Nepali before sentiment classification.

**Table 4.** : Illustration of NEPTUN Normalization and Sentiment Analysis Pipeline



| Step | Example Flow |
|---|---|
| 1. Input Reviews (Code-Mixed) | - Dherai ramro chha<br>- Color ni mitho ramroo xa<br>- Ekdam ramro chha thank you |
| 2. Tokenization and Code-Mixing Detection | Identified Romanized Tokens<br>- ["Dherai", "ramro", "chha"]<br>- ["mitho", "ramrooo", "xa"]<br>- ["ekdam", "ramro", "chha"] |
| 3. Phonetic Transliteration to Devanagari | Example Mappings:<br>- dherai → ['घेराइ', 'ढीराई', 'धैरै', 'धेराई']<br>- ramro → ['राम्रो', 'रम्रो']<br>- chha → ['छ', 'छा']<br>- mitho → ['मीठो', 'मिठो'] - xa → ['छ', 'छा']<br>- ekdam → ['एक्दम', 'ईक्दाम्', 'एक्डाम्'] |
| 4. Candidate Selection (Devanagari Phoneme) | Top Matched Words (Examples):<br>- dherai → धेरै<br>- ramro → राम्रो<br>- chha → छ<br>-mitho → मिठो<br>- ekdam → एक्दम |
| 5. Back-Transliteration to Standardized Romanized Nepali | Final Normalized Tokens:<br>- {'dherai': 1} → dherai - {'ramro': 2, 'ramrooo': 1} → ramro<br>- {'chha': 2, 'xa': 1} → chha<br>- {'mitho': 1} → mitho<br>- {'ekdam': 1} → ekdam |
| 6. Final Normalized Output | - Dherai ramro chha<br>- Mitho ramro chha<br>- Ekdam ramro chha |

After normalization, the standardized Romanized Nepali text is vectorized using pre-trained embeddings and passed into sentiment classifiers. The normalization ensures uniform token representation, reducing noise from spelling variations and code-mixing. As a result, models like Logistic Regression, Naïve Bayes, KNN, and BERT achieve improved accuracy and stability compared to unnormalized input.

### 3.5 Evaluation of Normalization Impact Sentiment Analysis)

To evaluate the impact of NEPTUN normalization, we compare sentiment classification performance under two preprocessing settings: **Standard Preprocessing (SP)** and **NEPTUN Preprocessing (NP)**. Four models are used for evaluation: Logistic Regression, Naive Bayes, K-Nearest Neighbors (KNN), and BERT, selected for their diversity in handling text classification.

**Model Descriptions**

- **Logistic Regression:** A robust baseline for binary classification.
- **Naive Bayes:** Effective on noisy, small datasets due to its probabilistic nature.
- **KNN:** A non-parametric method that captures local patterns.
- **BERT:** A pre-trained transformer offering deep contextual understanding.

**Preprocessing Settings**

**Standard Preprocessing (SP)** Applies generic English NLP techniques such as text cleaning, lemmatization, and stopword removal.

**NEPTUN Preprocessing (NP)** Extends SP by incorporating NEPTUN normalization and a Romanized Nepali stopword list to handle phonetic variation, code-switching, and non-standard spellings.

**Comparative Evaluation** Models are evaluated using accuracy, precision, recall, and F1-score. This comparison reveals the effectiveness of NEPTUN in improving sentiment analysis for Romanized Nepali text.

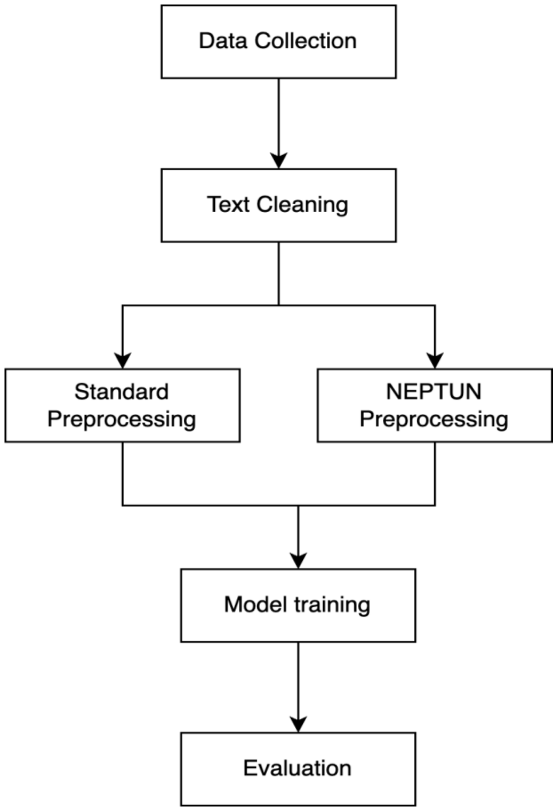


**Figure 3.** System Flow Diagram

## 4. Results and Discussion

This section presents a focused analysis of how existing phonetic algorithms handle Romanized Nepali text, highlighting their limitations in sentiment analysis tasks. A manually annotated dataset was developed to map frequently used Romanized words to standard Devanagari forms, ensuring semantic consistency during evaluation

### 4.1 Data Annotation for Phonetic Evaluation)

A dataset was created by three linguistically trained native speakers to align Romanized Nepali words with their correct

Devanagari counterparts. Annotation guidelines emphasized semantic equivalence over surface-level similarity. For instance, raamrai, ramra, and ramro were grouped together under the root form ramro (good), whereas phonetically similar but semantically different words were kept distinct.

**Table 5.** Example of Manually Annotated Romanized–Devanagari Word Pairs for Phonetic Evaluation

| Words | Devanagari | Normalized Devanagari |
|---|---|---|
| raamrai | राम्रै | राम्रा |
| raamro | राम्रो | राम्रा |
| bigriyeko | बिग्रिएको | बिग्रेका |
| fohor | फोहर | फोहर |
| xodyo | छोड्यो | छोड्यो |

Manual cross-verification ensured quality, and disagreements were resolved by a third annotator using predefined guidelines.

### 4.1.1 Limitations of Existing Phonetic Algorithms on Romanized Nepali

#### Soundex

Soundex, designed for English, performs overgeneralized matching on Romanized Nepali. While it effectively normalizes spelling variants (e.g., ramro, ramra, raamrai$\rightarrow$R56), it fails semantically:

- Incorrect Merging: manparyo (liked) and manparyena (did not like) are both encoded as M516, despite conveying opposite sentiments.
- False Separation: Synonyms like foto and photo are assigned different codes (F3 vs. P3).

#### Double Metaphone

Although more flexible than Soundex, Double Metaphone still struggles with Romanized Nepali:

- **Incorrect Grouping:** damiii (awesome) and daam (price) share the same code (TM) despite semantic differences.
- **Inconsistent Normalization:** chhaina, xaina, and xhaina (all meaning ”is not”) are inconsistently mapped, missing semantically identical groupings.
- **Accurate Distinction:** It does, however, correctly distinguish some sentiment pairs, like manparyo vs. Manparyena.

Both Soundex and Double Metaphone suffer from semantic blind spots in Romanized Nepali due to their English-centric design. These issues motivate the need for a context-aware, dictionary based approach like our proposed method, NEPTUN, which ensures semantic fidelity through Devanagari mapping.

**4.1.2 Advantages of NEPTUN over Existing Phonetic Algorithms NEPTUN** is specifically designed for Romanized Nepali, offering advantages over general purpose phonetic algorithms like Soundex and Double Metaphone. It incorporates

**Table 6.** Illustration of Soundex Collisions and Misclassifications in Romanized Nepali Words

| Soundex | Romanize Words | Devanagari Words |
|---|---|---|
| D5 | [‘damiii’,’dhanna’, ‘daam’, ‘diyena’] | [‘दामी’,’ धन्न’, ‘दाम’, ‘दिएन’] |
| M516 | [‘manparyena’, ‘manparyo’] | [‘मनपरेन’, ‘मनपर्यो’] |
| H535 | [‘hudaina’, ’handim’] | [‘हुदैन’, ‘हान्दिम’] |
| L3 | [‘lutya’, ‘lida’, ‘lyauda’] | [‘लुटे’, ‘लिदा’, ‘ल्याउदा’] |
| K2 | [‘khojeko’, ‘khassai’, ‘kesh’, ‘khusi’] | [‘खोजेको’, ‘खास्सै’, ‘केश’, ‘खुसी’] |

**Table 7.** Illustration of Double Metaphone Groupings and Misclassifications on Romanized Nepali Words

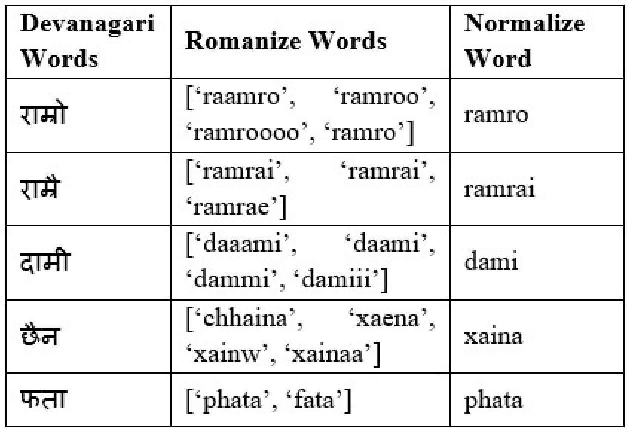
| Double Metaphone | Romanize Words | Devanagari Words |
|---|---|---|
| TM | [‘damiii’, ‘daam’] | [‘दामी’, ‘दाम’] |
| LT | [‘lutya’, ‘lida’, ‘lyauda’] | [‘लुटे’, ‘लिदा’, ‘ल्याउदा’] |
| KS | [‘ghas’, ‘kasso’, ‘khassai’, ‘khusi’] | [‘घास’, ‘कस्सो’, ‘खास्सै’, ‘खुसी’] |
| ST | [‘sayad’, ‘sutae’, ‘xodyo’, ‘sadhai’, ‘sodda’] | [‘सायद’, ‘सुते’, ‘छोड्यो’, ‘सधै’, ‘सोद्दा’] |
| XT | [‘chodyo’, 'choto', ‘chutyo’, ‘chitae’] | [‘छोड्यो’, ‘छोटो’, ‘चुट्यो’, ‘छिटै’] |

linguistic rules, phonetic variation mapping, and dictionary validation to robustly normalize informal and inconsistent spellings.

Unlike traditional methods, NEPTUN handles noisy Romanized inputs, filters out English words in code-mixed sentences using WordNet, and validates candidates against a Nepali lexicon to avoid semantically incorrect merges. This approach significantly reduces phonetic collisions and improves normalization accuracy.

**NEPTUN in Action:** As shown in Table 8, NEPTUN effectively generates phonetic variants, selects correct standardized forms, and distinguishes between semantically distinct words (e.g., “ramro” vs. “ramrai”). It accurately normalizes informal variants without losing meaning, ensuring linguistic integrity across diverse input forms. Table 8: Illustration of NEPTUN’s Normalization Process for Romanized Nepali Word Variants

**Table 8.** Illustration of NEPTUN’s Normalization Process for Romanized Nepali Word Variants



| Devanagari Words | Romanize Words | Normalize Word |
|---|---|---|
| राम्रो | [‘raamro’, ‘ramroo’, ‘ramroooo’, ‘ramro’] | ramro |
| राम्रै | [‘ramrai’, ‘ramrai’, ‘ramrae’] | ramrai |
| दामी | [‘daaami’, ‘daami’, ‘dammi’, ‘damiii’] | dami |
| छैन | [‘chhaina’, ‘xaena’, ‘xainw’, ‘xainaa’] | xaina |
| फता | [‘phata’, ‘fata’] | phata |

NEPTUN’s precision in normalization enables future enhancements in morphological and lemmatization tasks.

#### 4.1.3 Error Analysis and Limitations of NEPTUN

While **NEPTUN** improves normalization for Romanized Nepali, a few limitations remain:

- **Limited Lexicon**: Due to the absence of a comprehensive Nepali dictionary, some valid words (e.g. bikash) may be wrongly discarded during validation.
- **Variant Explosion in Long Tokens:** Long or compound words like “hajuuraaharuulee” produce many phonetic variants, increasing computational load and reducing selection accuracy.
- **Phonetic Ambiguity:** Words with similar sounds (e.g. ”xoina” vs. ”xaina”) can cause confusion if phonetic mappings are applied too broadly, leading to incorrect normalizations.

Phonetic Ambiguity: Words with similar sounds (e.g., ”xoina” vs. “xaina”) can cause confusion if phonetic mappings are applied too broadly, leading to incorrect normalizations.

#### 4.1.4 Comparative Phonetic Accuracy (Summary)

To assess normalization performance, we used a manually annotated Romanized-to-Devanagari dataset and computed phonetic accuracy as:

Phonetic Accuracy= (Number of Correctly Normalized Pairs)/(Total Number of Evaluated Pairs) × 100

Only exact matches with the gold-standard Devanagari representations without semantic drift were considered correct. Mappings causing semantic collisions were excluded from correct counts.

NEPTUN outperformed baseline phonetic algorithms by incorporating Nepali specific phoneme mapping, lexicon validation, and real world frequency based variant handling, making it more reliable for downstream NLP applications.

**Table 9.** Phonetic Accuracy Comparison

| Method | Phonetic Accuracy |
|---|---|
| Soundex | 52.58% |
| Double Metaphone | 65.00% |
| **NEPTUN** | **73.00%** |

### 4.2 Sentiment Categorization and Dataset Statistics

The sentiment dataset described in section 3.3.1 (Daraz reviews; [18]) was used for model development and evaluation. An initial corpus of 34,438 Romanized Nepali product reviews was collected, covering categories such as electronics, fashion, and home appliances. After removing empty, placeholder, and Devanagari-scripted entries, the dataset was reduced to 16,439 valid reviews.

**Labeling Approach:**

Sentiment labels were assigned using a hybrid rating–text heuristic. Reviews were first categorized based on star ratings, and then cross-validated with sentiment-indicative keywords (e.g., “ramro xa”, “not working”, “thikai xa”). In cases where the star rating conflicted with the textual sentiment, the reviews were manually reviewed and corrected to ensure label accuracy. Preprocessing steps included tokenization, lower casing, and stop word removal. For traditional classifiers, unigram and bigram TF–IDF features were extracted, while BERT used contextual embeddings.

**Final Sentiment Distribution:**

After data cleaning and labeling, a total of 16,076 valid reviews were retained for sentiment analysis. Among them, 12,141 reviews were positive, 2,887 were negative, and 1,048 were neutral.

The distribution of sentiment categories is summarized in Table 10, while Figure 4 presents the corresponding visual representation in the form of a pie chart. The table provides the exact counts and proportions, ensuring transparency in how the dataset was used for performance evaluation.

**Table 10.** Sentiment Distribution in the Final Dataset.

| Sentiment | Count | Percentage |
|---|---|---|
| Positive | 12,141 | 75.5% |
| Negative | 2,887 | 17.96% |
| Neutral | 1,048 | 6.52% |

### 4.3 Experimental Evaluation Setup

To evaluate the effectiveness of the proposed sentiment classification methods, the processed dataset of 16,076 labeled reviews was divided into training (80%) and testing (20%) subsets. The following classifiers were trained and evaluated: Logistic Regression, Naïve Bayes, K-Nearest Neighbors (KNN), and BERT. Feature extraction was based on unigram and bigram TF–IDF representations for traditional classifiers, while BERT utilized contextual embeddings. Model performance was assessed using Accuracy, Precision, Recall, and

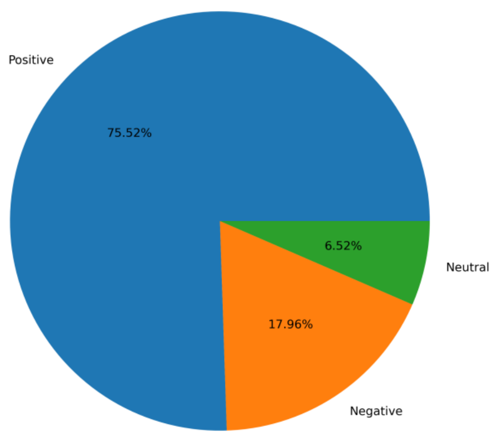


**Figure 4.** Pie Chart Illustrating the Sentiment Distribution Across the Final Dataset.

F1-score. Additionally, a confusion matrix was generated for each classifier to visualize correctly and incorrectly predicted sentiment categories, computed by comparing model outputs against the manually verified ground-truth labels.

## 4.4 Impact of NEPTUN Normalization on Sentiment Analysis Performance

In this phase of our experiment, we evaluated the impact of normalization techniques on sentiment classification for Romanized Nepali reviews by fine-tuning and training different models, including Logistic Regression, Naive Bayes, , K-Nearest Neighbors (KNN) and BERT. Two training configurations were considered: (i) **NEPTUN Preprocessing (NP)**, which includes phonetic normalization, stopword removal, and enhanced text normalization tailored for Romanized Nepali; and (ii) **Standard Preprocessing (SP)**, which applies conventional English preprocessing functions without phonetic enhancements. The models were evaluated based on Accuracy, Precision, Recall, and F1 score. The results are summarized in Table 11.

**Table 11.** Performance Comparison of Different Models under NEPTUN and Standard Preprocessing

| Model | Scenario | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| Logistic Regression | NP | 0.8341 | 0.8184 | 0.8341 | 0.7977 |
| | SP | 0.8323 | 0.8158 | 0.8323 | 0.7954 |
| BernouliNB | NP | 0.8126 | 0.7889 | 0.8126 | 0.7918 |
| | SP | 0.8130 | 0.7917 | 0.8130 | 0.7929 |
| KNeighbors Classifier | NP | 0.8141 | 0.7915 | 0.8141 | 0.7806 |
| | SP | 0.8093 | 0.7835 | 0.8093 | 0.7753 |
| BERT | NP | **0.8756** | **0.8688** | **0.8756** | **0.8703** |
| | SP | 0.8731 | 0.8615 | 0.8731 | 0.8641 |

### 4.4.1 Quantitative and Statistical Analysis

The impact of NEPTUN normalization was evident across most models, as seen from the improvements in classification performance. Logistic Regression demonstrated a slight increase in all four evaluation metrics under the NEPTUN pipeline, suggesting that enhanced preprocessing contributes to more effective feature representation. While Bernoulli Naive Bayes performed marginally better with standard preprocessing in terms of accuracy, the differences across all metrics were negligible, indicating that normalization had limited influence on models heavily reliant on binary word presence. For K-Nearest Neighbors, the benefits of NEPTUN were more pronounced. Improved accuracy, precision, and recall suggest that phonetic normalization reduces lexical variation, enhancing token consistency and thus benefiting distance-based models. BERT, as expected, achieved the highest overall performance in both preprocessing settings. The slight but consistent improvement with NEPTUN highlights the value of structured input in enhancing deep learning models' semantic understanding.

**4.4.2 Visual Comparison of Model Performance** To visually contrast the impact of the two preprocessing approaches across different models, we present a histogram that compares Accuracy, Precision, Recall, and F1-Score for each configuration. This visualization helps identify performance trends and reveals which models benefit the most from normalization.

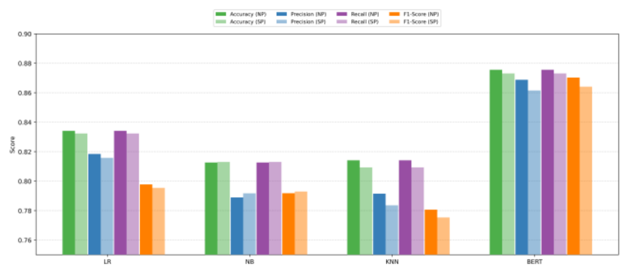


**Figure 5.** Histogram Comparing Classification Performance of Different Models under NEPTUN (NP) and Standard (SP) Preprocessing.

**4.4.3 Confusion Matrix-Based Error Analysis** To further investigate the classification behavior of the best-performing model, we present confusion matrices for BERT under both NEPTUN (NP) and Standard Preprocessing (SP) configurations. These matrices illustrate how the model performs across the sentiment classes—positive, neutral, and negative—highlighting patterns in misclassification.

Under Standard Preprocessing, the model exhibits noticeable confusion between sentiment classes, particularly between negative and positive, as well as difficulty in accurately identifying neutral sentiments. This is a common challenge in Romanized Nepali due to inconsistent spellings and informal expressions that hinder precise tokenization and semantic interpretation.

With NEPTUN preprocessing, these ambiguities are significantly reduced. The model demonstrates clearer separation between sentiment categories and fewer misclassifications, especially in the neutral class. This suggests that NEPTUN's normalization strategy improves the consistency and clarity of input text, enhancing the model's ability to capture sentiment cues more effectively.

Overall, the confusion matrices reveal that NEPTUN preprocessing contributes to more accurate and balanced sentiment classification by addressing the inherent noise and variability in Romanized Nepali text.

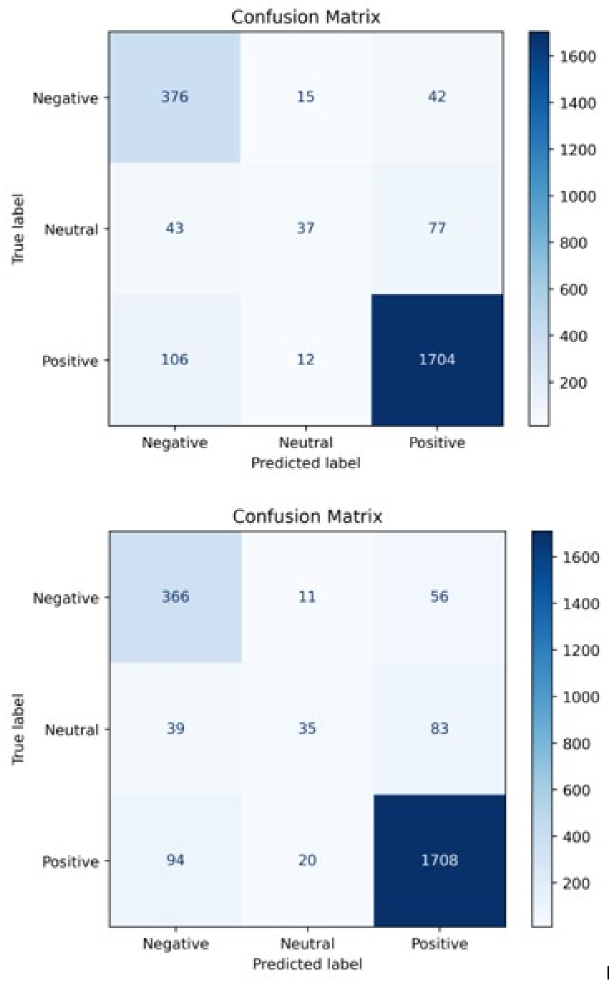


**Figure 6.** Confusion Matrices Showing Classification Performance of BERT under NEPTUN (top) and Standard (bottom) Preprocessing Configurations.

**4.4.3.1 Analysis of Misclassified Examples** To better understand sources of misclassification, we analyzed false positive predictions from the BERT model under both preprocessing configurations. Table 12 shows representative examples and brief explanations of the observed contradictions.

**Table 12.** Performance Comparison of Different Models under NEPTUN and Standard Preprocessing

| Text | True | Pred | Reason |
|---|---|---|---|
| "Product ramrai lagyo look like cheap plastic material paisa aanusar kassai ramro lagena" | Negative | Positive | Mixed sentiment—positive start followed by negative clause. |
| "thau red light problem xa first day dekhinai ramro xa taar haru alikati touch garyo vane light bigrinxa taar chahi not good last ramrai ca use care" | Neutral | Positive | Conflicting opinions: model fails to balance positive and negative cues. |
| "ramro xa bt small raix vaneko vanda" | Neutral | Positive | Contrastive "bt" softens praise; model ignores dissatisfaction cue. |
| "good product jacket chain bad material replaces chain single use" | Neutral | Positive | Praise followed by defect mention; negative part dominates interpretation. |

False positives mainly stem from mixed sentiments, contrastive connectives (e.g., but, tara), and ambiguous contextual modifiers. Although NEPTUN normalization enhances lexical consistency, deeper discourse cues and implicit contrasts remain difficult for models to interpret accurately.

**4.4.4 Qualitative Analysis** Normalization through the NEPTUN pipeline played a critical role in reducing ambiguity in Romanized Nepali text, particularly where multiple variations of the same word are common (e.g., ramro, ramrooo). By mapping such variants to a canonical form, the NEPTUN approach improved token consistency, which significantly helped models like KNN and Logistic Regression. Furthermore, phonetic normalization enabled better handling of variant spellings such as chhain and xain, thus reducing feature sparsity and enhancing classification consistency. Deep models like BERT particularly benefited from the semantic clarity provided by normalized input, allowing them to make more accurate sentiment predictions—for instance, correctly interpreting expressions like thikai cha as neutral sentiment. In contrast, BernoulliNB did not exhibit significant gains, likely due to its simplistic feature independence assumptions and limited sensitivity to nuanced input transformations. Overall, NEPTUN preprocessing proved most beneficial for models sensitive to word forms, as it reduces lexical inconsistencies and improves input quality. The proposed NEPTUN normalization framework significantly improved sentiment classification performance across all models. For example, BERT achieved an accuracy of 87.56% and an F1-score of 0.87, demonstrating that language-specific preprocessing enhances both precision and interpretability for Romanized Nepali sentiment analysis.

## 5. Conclusion

The experimental results reveal that NEPTUN Preprocessing (NP) offers consistent improvements in sentiment classification for Romanized Nepali text across multiple models. While the performance gains over Standard Preprocessing (SP) are sometimes marginal in terms of accuracy, NP notably enhances recall and F1-score, indicating its strength in handling linguistic variability common in informal user-generated content. A key implementation uniqueness of NP lies in its integration of a phonetic translation pipeline that is specifically tuned for the linguistic patterns of Romanized Nepali. Unlike general phonetic algorithms such as Soundex or Double Metaphone, which are often overly aggressive and can introduce semantic ambiguity by conflating phonetically similar but semantically distinct terms, NP combines phonetic normalization with dictionary-based validation. This approach helps retain semantic integrity while reducing lexical noise—a critical advantage for low-resource languages lacking standardized orthography. Furthermore, NP demonstrates its utility across both classical and deep learning models. Particularly in models like BERT and KNN, the preprocessing improves token consistency and reduces sparsity by resolving phonetic and orthographic variations. Looking ahead, NEPTUN can be further enhanced through the integration of morphological analysis and custom stemming or lemmatization tailored for Romanized Nepali. These additions would allow for deeper syntactic and semantic normalization, offering even more robust input representations for downstream NLP tasks. Optimized tokenization strategies and broader coverage of spelling variants also represent promising directions for future research. In summary, NEPTUN provides a linguistically informed, domain-specific preprocessing framework that significantly enhances sentiment analysis performance in noisy, phonetic rich, Romanized text settings marking a notable contribution toward NLP tools for under-resourced languages.

Quantitatively, the NEPTUN preprocessing pipeline achieved measurable improvements across all evaluated models. For instance, BERT attained an accuracy of 87.56% and an F1-score of 0.87 under NEPTUN preprocessing compared to 87.31% and 0.86 with standard preprocessing. Similarly, traditional models such as Logistic Regression and KNN showed gains of 0.3–0.5% in F1-score, confirming the statistical effectiveness of NEPTUN for Romanized Nepali sentiment analysis.

In summary, NEPTUN provides a linguistically informed, domain-specific preprocessing framework that significantly enhances sentiment analysis performance in noisy, phonetic-rich, Romanized text settings marking a notable contribution toward NLP tools for under-resourced languages.

## References

[1] B. Narayanaswamy and G. Reddy, "Exploiting BERT and RoBERTa to improve performance for aspect based sentiment analysis," 2021.

[2] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Engineering Journal*, pp. 1093–1113, 2014, Elsevier.

[3] Anonymous, "Benchmarking Language Models for Offensive Sentences Classification in Offensive Nepali Roman Multi-Label Dataset," submitted to ACL Rolling Review, 2024.

[4] S. Liu and M. Best, "A survey of NLP progress in Sino-Tibetan low-resource languages," in *Proc. NAACL-HLT*, 2025.

[5] B. Chandio *et al.*, "Sentiment analysis of Roman Urdu on e-commerce reviews using machine learning," 2022.

[6] Z. Toh and J. Su, "NLANGP at SemEval-2016 Task 5: Improving aspect-based sentiment analysis using neural network features," in *Proc. SemEval-2016*, San Diego, CA, 2016.

[7] O. M. Singh, B. K. Bal, S. Timilsina, and A. Joshi, "Aspect based abusive sentiment detection in Nepali social media texts," 2024.

[8] A. Rafae *et al.*, "An unsupervised method for discovering lexical variations in Roman Urdu informal text," Qatar Computing Research Institute, HBKU; LUMS; ITU, 2021.

[9] S. Mboutayeb, A. Majda, and N. S. Nikolov, "Multilingual sentiment analysis: A deep learning approach," in *Proc. BML'21*, 2021.

[10] M. A. Manzoor *et al.*, "Lexical variation and sentiment analysis of Roman Urdu sentences with deep neural networks," 2021.

[11] X. Liu, S. Zhang, F. Wei, and M. Zhou, "Recognizing named entities in tweets," in *Proc. ACL-HLT*, Portland, OR, 2011.

[12] F. Liu, F. Weng, and X. Jiang, "A broad-coverage normalization system for social media language," in *Proc. ACL*, vol. 1, pp. 1035–1044, 2012.

[13] C. Li and Y. Liu, "Improving text normalization using character-blocks based models and system combination," in *Proc. COLING*, Richardson, TX, 2012.

[14] I. A. Bhat *et al.*, "IIIT-H system submission for FIRE2014 shared task on transliterated search," in *Proc. FIRE'14*, New York, NY, USA, 2015.

[15] R. K. Behera, M. Jena, S. K. Rath, and S. Misra, "Co-LSTM: Convolutional LSTM model for sentiment analysis in social big data," *Information Processing & Management*, 2021.

[16] A. A. Aliero, A. G. Tafida, B. S. Adebayo, and B. U. Kangiwa, "Systematic review on text normalization techniques and its approach to non-standard words," Kebbi State, 2022.

[17] K. Mehmood, D. Essama, K. Shafi, and M. K. Malik, "An unsupervised lexical normalization for Roman Hindi and Urdu sentiment analysis," Australia–Pakistan, 2020.

[18] A. Pradhananga and A. K. Sah, "Transformer-based deep learning models for sentiment analysis in romanized Nepali: A comparative investigation of BERT and RoBERTa," in *Proc. 14th IOE Graduate Conference*, 2023.

[19] K. Yadav, M. S. Akhtar, and T. Chakraborty, "Normalization of spelling variations in code-mixed data," in *Proc. Conference*, Delhi, India, 2020.

[20] M. S. Maučec, D. Verdonik, and G. Donaj, "Sequence-to-sequence models and their evaluation for spoken language normalization of Slovenian," *Applied Sciences*, 2024.

[21] V. Radchenko and N. Drushchak, "Improving named entity recognition for low-resource languages using large language models: A Ukrainian case study," in *Proc. UNLP 2025*, 2025.

[22] G. Thakkar, N. M. Preradović, and M. Tadić, "Examining sentiment analysis for low-resource languages with data augmentation techniques," 2024.