Word Level Language Identification for Romanized code-mixed Indian Languages

B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Devika Singh
(210101036)

under the guidance of

Dr. Sanasam Ranbir Singh

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Word Level Language Identification for Romanized code-mixed Indian Languages**" is a bonafide work of **Devika Singh (Roll No. 210101036)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Sanasam Ranbir Singh**

Professor,

Department of Computer Science & Engineering,

Indian Institute of Technology Guwahati, Assam.

Nov, 2024

Guwahati.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, **Professor Dr. Sanasam Ranbir Singh**, and my mentor, **Saurabh Kumar**, for their invaluable support and continuous guidance throughout this project. Their expertise and encouragement have contributed significantly to my learning and the successful completion of my BTP work. I am truly thankful for their valuable time and the constant efforts they invested in assisting me during this journey.

Abstract

This research addresses the unique challenges of language identification in Romanized code-mixed Indic languages, a growing phenomenon in digital communication, especially on social media. Traditional language identification models struggle with code-mixed text due to script mixing, borrowed words, and homographs. This project introduces a new word-level language-tagged dataset for 22 Indic languages, all represented in Roman script. We explore two models—a fastText-based classifier and a combined fastText and Unicode-range classifier - for sentence-level language identification for native script. Building on these, we propose a third model to achieve precise word-level tagging within multi-script code-mixed sentences. This ongoing BTP provides a foundation for advancing language identification techniques in multilingual code-mixed and low-resource settings.

Contents

List of Figures **vi**

List of Tables **vii**

1 Introduction **1**

1.1 Challenges 2

1.1.1 Code-mixed dataset 2

1.1.2 Borrowed words 3

1.1.3 Homographs 3

1.2 Indic Language Dataset 3

1.3 Our Contributions 4

1.4 Organization of The Report 5

2 Background **6**

2.1 Data 6

2.1.1 Code - Mixed data 6

2.2 Language identification 7

2.3 Language Tagging 7

2.4 Conclusion 8

3 Related Works **9**

3.1 Resources 9

3.2 Language identification 10

3.2.1 fastText 10

3.3 AI4Bharat 10

3.3.1 IndicCorp 11

3.3.2 Bhasha-Abhijnaanam 11

3.4 Conclusion 12

4 Proposed Work **13**

4.1 Dataset 13

4.1.1 Current Progress in dataset 13

4.2 Models 16

4.2.1 fastText based 17

4.2.2 fastText and Unicode-Range based sentence level 17

4.2.3 fastText and Unicode-Range based word level 18

4.3 Conclusion 19

5 Experimental Setup **20**

5.1 Setup 20

5.2 Dataset 20

5.3 Model parameters and Hyperparameters 20

5.4 Evaluation Metrics 21

6 Results and Analysis **23**

6.1 Model Selection 23

6.1.1 Analysis of Model Performance 23

6.1.2 Model Performance 24

7 Conclusion and Future Work **25**

Bibliography **26**

List of Figures

1.1 Example of homographs in code-mixed text, showing the as an English article and the in Hindi meaning “were”. 3

1.2 Languages included in the Romanized code-mixed dataset. 4

1.3 Example of word-level language identification in a Romanized Hindi-English code-mixed sentence. (a) An example code-mixed sentence; (b) Tagged output with language and script label. 4

2.1 An example of single-script code-mixing in Romanized Hindi-English (Hinglish), where both Hindi and English words use the Roman(Latin) alphabet. . . . 6

2.2 An example of mixed-script code-mixing with Hindi in Devanagari and English in Roman(Latin) script. 7

2.3 Examples of language identification at sentence-level and word-level. 7

3.1 Hybrid IndicLID Classifier Workflow (referenced from [MKK23]) 12

4.1 Snapshot of the Native Script dataset. 15

4.2 Snapshot of the Romanized dataset. 16

4.3 Examples of Romanized Sentences from the Dataset with Translations and Corresponding Language Tags 16

4.4 fastText + Unicode-Range based sentence based language identification for native languages in a single script dataset. 18

List of Tables

4.1 Language Tags with Corresponding Languages, Scripts, and number of single and multi-script sentences. 14
4.2 Language Item Counts in the Romanized Dataset 16

5.1 fastText Parameters and Hyperparameters 21

6.1 Performance comparison of fastText and fastText + Unicode-Range based models 24

Chapter 1

Introduction

The growth of social media has resulted in vast amounts of code-mixed data, especially within multilingual communities where code-mixed language use is prevalent. Users from such communities, for example, in regions like India, frequently blend multiple languages within sentences, often using a single script (e.g., Roman). This phenomenon presents valuable opportunities to gain insights into linguistic behaviour, cross-cultural interactions, and the unique dynamics of multilingual communication. However, practical analysis of such data requires precise word-level language identification to distinguish between languages within code-mixed text, essential for enabling downstream NLP tasks like sentiment analysis and entity recognition.

Despite the potential of code-mixed data, existing language tagging systems face significant limitations. State-of-the-art language identification models fail to process code-mixed inputs accurately, and high-quality datasets designed specifically for code-mixed Indic languages are scarce. This scarcity is particularly challenging for low-resource languages, where data limitations make training effective models even more difficult. Developing robust models capable of handling code-mixed content in multilingual settings is crucial to improving performance across diverse language environments and ensuring inclusivity in NLP applications. With over 735 million internet users in India [KPM17], effective romanized language identification is crucial, particularly in the chat and social media domains.

We now outline the distinct challenges associated with code-mixed data, focusing on the complexities of Indic languages and the development of an effective word-level tagging system, which impact model performance and accuracy in multilingual environments.

1.1 Challenges

1.1.1 Code-mixed dataset

- Existing NLP tools often struggle to handle the complexities of code-mixing in downstream applications (Srivastava and Singh, 2021) [SS21].
- We find a noticeable gap between the widespread use of code-mixed language and the amount of data available to study it further. This scarcity mainly arises because code-mixed language rarely appears in formal texts commonly found on the web.
- A major challenge to code-mixed dataset generation is that many online platforms have privacy restrictions that prevent data use or scraping.
- Studies indicate that training on code-mixed sentences often yields better results than using separate monolingual corpora. However, most code-mixed datasets are synthetically generated, which may limit their ability to capture the full complexity of natural code-mixing.
- In mixed-script code-mixing, spelling variations arise from transliterations that are based on phonetic sounds of words, leading to inconsistencies that pose challenges for precise language processing.

- The effectiveness of pre-trained multilingual models is considerable for high-resource languages. These represent low-resource languages poorly, often due to the limited availability of pre-training data for these languages in large multilingual corpora. Even when data is available, its volume is significantly smaller than that for English and other high-resource languages. Existing evaluation benchmarks also provide limited representation for low-resource languages.

1.1.2 Borrowed words

In mixed-script code-mixed text, borrowed words present a unique challenge.Borrowed words are terms adopted from one language into another, often due to cultural or contextual relevance.

For instance, कंप्यूटर is a borrowed word from English into Hindi language. Borrowed words make it difficult for language identification systems to determine if they belong to the "borrowing" language or the "native" language of the word.

1.1.3 Homographs

In code-mixed text, certain words may have identical spellings across languages but carry different meanings. This creates ambiguity for word-level language identification systems, as they must determine the correct language based on context.

The car is parked outside.

Ham ja rahe the. (*Translation: "We were going."*)

Figure 1.1: Example of homographs in code-mixed text, showing the as an English article and the in Hindi meaning "were".

1.2 Indic Language Dataset

We focus on building a Romanized code-mixed dataset for the 22 languages listed in the Indian constitution,annotated with word-level language tags. All elements in the dataset will be represented in the Latin script.

Assamese	Bangla	Bodo	Dogri
Gujarati	Hindi	Kannada	Kashmiri
Konkani	Maithili	Malayalam	Manipuri
Marathi	Nepali	Oriya	Punjabi
Sanskrit	Santali	Sindhi	Tamil
Telugu	Urdu		

Figure 1.2: Languages included in the Romanized code-mixed dataset.

1.3 Our Contributions

This research addresses the problem of word-level language identification for Romanized code-mixed Indian languages, specifically focusing on data from social media platforms where such code-mixing is prevalent. In code-mixed data, users often blend multiple languages within a single sentence, for instance, using the Roman script to express different native languages.

Our goals are to create a word-level, Romanized code-mixed language-labelled dataset for 22 Indic languages, which will support further research and model development for code-mixed language identification tasks, and to use this dataset to develop a model that can identify the language of each word in these code-mixed sentences, enabling accurate tagging where each word is associated with its respective language label, similar to Named Entity Recognition (NER).

Mujhe coffee pasand hai

(a)

$$\begin{bmatrix} (\text{Mujhe}, \text{Latin\_Hindi}), & (\text{coffee}, \text{Latin\_English}), \\ (\text{pasand}, \text{Latin\_Hindi}), & (\text{hai}, \text{Latin\_Hindi}) \end{bmatrix}$$

(b)

Figure 1.3: Example of word-level language identification in a Romanized Hindi-English code-mixed sentence. (a) An example code-mixed sentence; (b) Tagged output with language and script label.

1.4 Organization of The Report

This report is organized as follows. Chapter 1 introduces the motivation and objectives of our work, along with the key challenges we aim to address. Chapter 2 provides background information on the fundamental topics relevant to this report, including code-mixed language usage and language tagging. Chapter 3 reviews related works in the field of language identification and handling of code-mixed data. Chapter 4 presents our approach, including the initial models and current progress toward building a word-level language identifier for code-mixed data. Chapter 5 describes the dataset and experimental setup used to evaluate our approach, and Chapter 6 details the results obtained. Finally, in Chapter 7, we discuss potential directions for future work.

Chapter 2

Background

2.1 Data

2.1.1 Code - Mixed data

Code-switching or Code-mixing is the blending of linguistic elements from multiple languages within a single conversation, and often even within a single sentence. While initially observed in casual, spoken interactions, the rise of social media has extended this phenomenon to digital spaces, particularly on forums and messaging platforms. This increased visibility has led to a growing body of research in code-mixed natural language processing (NLP), as the prevalence of mixed-language data on social media and chat applications continues to grow.

2.1.1.1 Single-Script Code-Mixing

At sentence level, this approach to code-mixing involves using a single script, often with elements from different languages.

Mujhe coffee bohot pasand hai.

Translation: "I really like coffee."

Figure 2.1: An example of single-script code-mixing in Romanized Hindi-English (Hinglish), where both Hindi and English words use the Roman(Latin) alphabet.

2.1.1.2 Mixed-Script Code-Mixing

In mixed-script code-mixing, languages with different native scripts are combined in a sentence. This phenomenon, known as Intra-sentential script-mixing, reveals patterns in how scripts and languages interact in multilingual communities.

मुझे coffee बहुत पसंद है।

Translation: "I really like coffee."

Figure 2.2: An example of mixed-script code-mixing with Hindi in Devanagari and English in Roman(Latin) script.

This type of mixing is common in code-mixed text where speakers blend languages fluidly within a single thought or phrase. All permutations of scripts and languages can be observed in mixed-script code-mixing sentences.

2.2 Language identification

Language identification is the task of identifying the language of a given input at the sentence or word levels. At the sentence level, the sentence is classified into a single language, whereas, at the word level, each word in a sentence is tagged with its respective language.

Sentence-level:

"I love coffee." [*English*]

Word-level:

Mujhe[*Hindi*] coffee[*English*] pasand[*Hindi*] hai[*Hindi*]

Figure 2.3: Examples of language identification at sentence-level and word-level.

2.3 Language Tagging

Language tagging in NLP is the task of detecting and labelling the language for each word or section in a text. Language tagging shares similarities with Named Entity Recognition (NER), a well-researched area. In NER, named entities in the text are identified and classified into predefined categories. Similarly, language tagging assigns each word a language label from a set of supported languages. Both tasks require contextual understanding: just as NER must consider the surrounding words to label an entity correctly, language tagging relies on context to accurately assign language labels, handling nuances such as borrowed words and homographs.

2.4 Conclusion

This chapter provides background information on the core topics covered in this report. We begin with an overview of code-mixed language use in multilingual communities, highlighting the various types of code-mixing. We then describe language identification and the language tagging process, discussing their application to code-mixed data and explaining the complexities involved in performing word-level tagging across multiple languages.

Chapter 3

Related Works

With the growth of social media, large amounts of code-mixed data have become available, creating new research opportunities in NLP. Social media platforms show great language diversity, as users often mix different languages in a single sentence or phrase. This unique setting has led recent NLP research to focus on processing code-mixed data. Recent studies in this area can be classified into three main categories.

3.1 Resources

Twitter is a popular source for code-mixed data in Indic languages, as many bilingual and multilingual users code-mix on the platform [CWT13, SBM$^+$14, JTJ17, RSC$^+$17]. Indic Wikis are also used as a pre-training resource due to their quality, though they are sparse. Other sources, such as CC100 [CKG$^+$20] and mC4 [XCR$^+$21], provide larger datasets but are often noisy and may contain inappropriate content [KCW$^+$22]. IndicCorp v1 [KKG$^+$20] is the first dataset specifically created for Indic languages.

3.2 Language identification

A reliable language identifier is essential for building corpora in low-resource languages, where challenges like noisy web data, small datasets, and similarity to high-resource languages make language identification difficult [C+20]. Existing tools like CLD3 [AI16], LangID [LB11], fastText [JGBM16], and NLLB [N+22] lack full support for all 22 official Indian languages and struggle to detect romanized text, except for CLD3's[AI16] limited support for Latin Hindi. The performance of general-purpose multilingual models like mBERT [DCLT19], mT5 [XCR+21], and XLM [CL19] trained on major Indic languages is limited by the need to share capacity with high-resource languages [CKG+20, K+22]. Indic-specific models, such as MuRIL [K+21] and IndicBERT v1 [KKG+20], achieve better results on Indian language tasks.

3.2.1 fastText

FastText is an efficient text classification tool designed to handle large-scale datasets effectively. It uses character n-gram features to capture subword information, which is especially useful for processing rare words. This approach also helps distinguish between languages with similar spellings, enhancing its performance in multilingual tasks. FastText incorporates n-gram features, dimensionality reduction, and an optimized version of the softmax classifier, making it fast and lightweight. Our current work employs fastText for language identification, leveraging its ability to accurately classify text at the sentence level across multiple Indic languages. These qualities make fastText well-suited for applications requiring scalable and resource-efficient text classification.

3.3 AI4Bharat

Ai4Bharat, a research lab dedicated to advancing AI resources for Indian languages, has developed and released an extensive collection of datasets, tools, and state-of-the-art models.

Among these, two resources central to this report are IndicCorp and Bhasha-Abhijnaanam[MKK23].

3.3.1 IndicCorp

IndicCorp provides a large sentence-level monolingual corpus covering 11 languages from the Indo-Aryan and Dravidian language families, along with Indian English. This corpus is, on average, nine times larger than the OSCAR dataset, offering significant data resources for these languages.

3.3.2 Bhasha-Abhijnaanam

The Bhasha-Abhijnaanam[MKK23] dataset and models focus on language identification (LID) for native-script and romanized text in 22 Indic languages. The dataset includes labelled data in both native and romanized scripts and contains clean sentences (grammatically correct, single script, etc.), which enhances its application for various NLP tasks in Indic languages. Bhasha-Abhijnaanam provides three LID models:

- Linear fastText-based model: This efficient model is split into two versions—one trained on native script data and the other on romanized script data. It utilizes character n-gram features, which help distinguish languages with similar spelling patterns.

- BERT-based model: This model, based on pre-trained BERT architecture, offers higher accuracy for LID tasks, albeit with a slower processing speed due to its complexity.

- Hybrid model: This model combines the strengths of both fastText and BERT-based approaches. In this model, the fastText classifier initially processes input for quick classification. If the fastText model's confidence is low, the BERT-based model is used, allowing a balance between computational efficiency and accuracy.

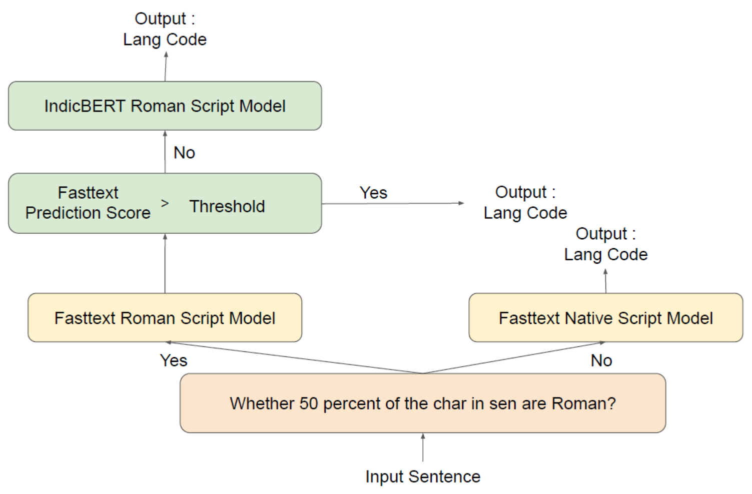

Output :
Lang Code
IndicBERT Roman Script Model
No
Fasttext Prediction Score > Threshold
Yes
Output :
Lang Code
Output :
Lang Code
Fasttext Roman Script Model
Fasttext Native Script Model
Yes
No
Whether 50 percent of the char in sen are Roman?
Input Sentence

Figure 3.1: Hybrid IndicLID Classifier Workflow (referenced from [MKK23])

This structured approach allows Bhasha-Abhijnaanam to support reliable and accurate language tagging across Indic languages for single script scenarios, covering both native and romanized scripts effectively.

3.4 Conclusion

This chapter reviews existing resources, tools, and models relevant to code-mixed data processing and language identification for Indic languages. A variety of distributed resources for NLP have been developed, notably by Ai4Bharat and other research initiatives, to support language tagging, classification, and resource building for low-resource Indian languages.

Chapter 4

Proposed Work

The aim of this BTP is to create a comprehensive dataset for Romanized code-mixed text across 22 Indic languages, annotated with word-level language tags. Additionally, this project seeks to develop a model trained on this dataset to accurately perform word-level language tagging for Romanized code-mixed sentences in these languages.

4.1 Dataset

There is a notable scarcity of publicly available datasets that provide phrase-level annotations for code-mixed Indic languages. Recognizing this gap, we actively engaged in developing such a dataset to facilitate advanced natural language processing applications in multilingual contexts.

4.1.1 Current Progress in dataset

4.1.1.1 Native Script Dataset

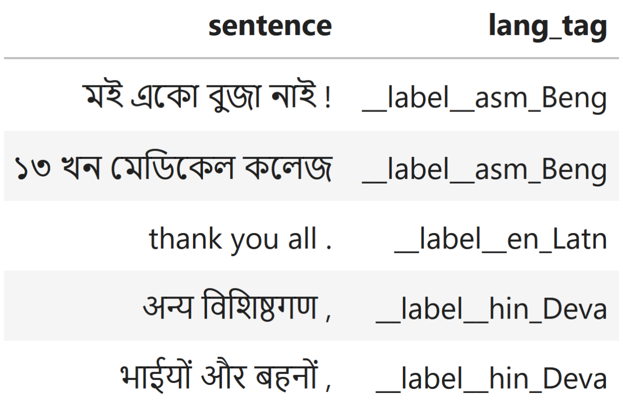
The dataset contains sentences in multiple Indic languages, each labelled with one of 31 distinct language and script tags at the sentence level language identification.

Label	Language	Script	Single Script	Multi Script
__label__awa_Deva	Awadhi	Devanagari	962	35
__label__bho_Deva	Bhojpuri	Devanagari	864	133
__label__brx_Deva	Bodo	Devanagari	984	13
__label__hne_Deva	Chhattisgarhi	Devanagari	906	91
__label__dgo_Deva	Dogri	Devanagari	963	34
__label__hin_Deva	Hindi	Devanagari	54837	2991
__label__kas_Deva	Kashmiri	Devanagari	718	279
__label__gom_Deva	Konkani	Devanagari	976	21
__label__mag_Deva	Magahi	Devanagari	985	12
__label__mai_Deva	Maithili	Devanagari	924	73
__label__mar_Deva	Marathi	Devanagari	36780	348
__label__npi_Deva	Nepali	Devanagari	904	93
__label__san_Deva	Sanskrit	Devanagari	902	95
__label__snd_Deva	Sindhi	Devanagari	989	8
__label__kas_Arab	Kashmiri	Arabic	929	68
__label__snd_Arab	Sindhi	Arabic	915	82
__label__urd_Arab	Urdu	Arabic	12057	107
__label__en_Latn	English	Latin	56788	43
__label__lus_Latn	Mizo	Latin	997	0
__label__asm_Beng	Assamese	Bengali	10540	189
__label__ben_Beng	Bengali	Bengali	30103	478
__label__guj_Gujr	Gujarati	Gujarati	49717	1124
__label__kan_Knda	Kannada	Kannada	35303	926
__label__mal_Mlym	Malayalam	Malayalam	34521	145
__label__mni_Beng	Manipuri	Bengali	8224	192
__label__mni_Mtei	Manipuri	Meetei Mayek	993	4
__label__ory_Orya	Odia	Oriya	38076	1270
__label__pan_Guru	Punjabi	Gurmukhi	32501	2891
__label__sat_Olck	Santali	Ol Chiki	885	112
__label__tam_Taml	Tamil	Tamil	39312	1211
__label__tel_Telu	Telugu	Telugu	40364	897

Table 4.1: Language Tags with Corresponding Languages, Scripts, and number of single and multi-script sentences.

4.1.1.2 Romanized code-mixed dataset

The data for the Romanized dataset is being collected from various social media platforms, including Twitter, Facebook, and YouTube. Raw data is retrieved from the web using location-based filtering, particularly from Twitter. This data undergoes preprocessing to



sentence	lang_tag
মই একো বুজা নাই !	__label__asm_Beng
১৩ খন মেডিকেল কলেজ	__label__asm_Beng
thank you all .	__label__en_Latn
अन्य विशिष्ठगण ,	__label__hin_Deva
भाईयों और बहनों ,	__label__hin_Deva

Figure 4.1: Snapshot of the Native Script dataset.

ensure that all sentences are in Latin script. The approach assumes that a Roman script sentence posted by a user located in a specific language spekaing region(for instance, Tamil Nadu) has a higher likelihood of being in the same language(Tamil). However, this method presents certain limitations:

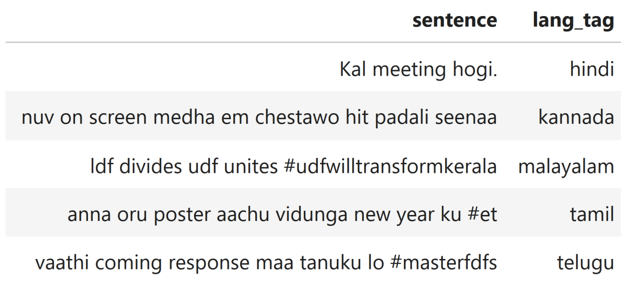
- The sentence being in pure English without any code-mixing.
- The sentence belonging to another Romanized code-mixed language, such as Hinglish, even if it was posted from Tamil speaking region.

To overcome these challenges, Llama 3 [D^+24] is employed to determine the language of each sentence. So far, the dataset has been processed using Llama 3 once, but it has yet to undergo human verification. At this stage, we are focusing on sentence-level language identification, with plans to extend the dataset for word-level tagging in the next phase of the BTP.

Our current progress involves collecting data from Hindi, Tamil, Telugu, Malayalam, and Kannada speaking regions. 570 out of 73,517 sentences (0.78%) in the dataset were classified as “unknown” by LLaMA 3 and subsequently discarded.

Category	Count
Total	1,25,389
Hindi	51,872
Tamil	24,864
Telugu	39,579
Kannada	3,921
Malayalam	4,582
Unknown	570

Table 4.2: Language Item Counts in the Romanized Dataset



sentence	lang_tag
Kal meeting hogi.	hindi
nuv on screen medha em chestawo hit padali seenaa	kannada
ldf divides udf unites #udfwilltransformkerala	malayalam
anna oru poster aachu vidunga new year ku #et	tamil
vaathi coming response maa tanuku lo #masterfdfs	telugu

Figure 4.2: Snapshot of the Romanized dataset.

Example 1: Romanized Sentence with Language Tag and Translation
Sentence: bro first itha reply panunga kannadigas enna pananga kollywood vs tollywood thaana poitu irunchu neenga enna puthusa start panitenga
Translation: Bro, first reply here. Kannadigas, what are Kannadigas doing? It was just Kollywood vs Tollywood, and now you've started something new with Kannada?
Language Tag: Kannada

Example 2: Romanized Sentence with Language Tag and Translation
Sentence: ennaanda ellaam needhaane #beast #master
Translation: What is this? You are everything! #beast #master
Language Tag: Tamil

Figure 4.3: Examples of Romanized Sentences from the Dataset with Translations and Corresponding Language Tags

4.2 Models

Our work on this BTP began with an in-depth study of language identification for Indic languages in native and Romanized scripts. Initially, we practised using fastText for text classification and word representation, gaining familiarity with its capabilities. The first phase then involved replicating the results from the Bhasha-Abhijnaanam[MKK23] paper, focusing on its fastText-based linear classifier for sentence-level language tagging in native scripts. Following this, we explored fastText and Unicode-Range based approaches to optimize training efficiency and model performance further. Finally, we extended our approach to word-level language tagging, utilizing fastText and Unicode-Range based methods to address intra-sentential script-mixing.

4.2.1 fastText based

In this approach, we employ supervised training with fastText to identify the native language at the sentence level. By training the model with labelled sentences, it learns to recognize and classify the language of each sentence, improving accuracy in identifying the language of the native-script texts.

4.2.2 fastText and Unicode-Range based sentence level

The data is first pre-processed by removing sentences that contain more than one script. This refined dataset is then passed through a sentence-level Unicode-Range based script classifier to determine the script of each sentence. This step is notably efficient, as 9 of the 31 tags in the dataset correspond to unique language-script pairs, allowing these cases to be identified with just script classification. For sentences identified in Devanagari, Bengali, Arabic, or Latin scripts, the data is further classified using fastText. Four separate fastText classifiers are trained on script-specific subsets of the data for these four scripts. This combined approach of script and language classification ensures accurate and efficient sentence-level language and script tagging for each sentence.

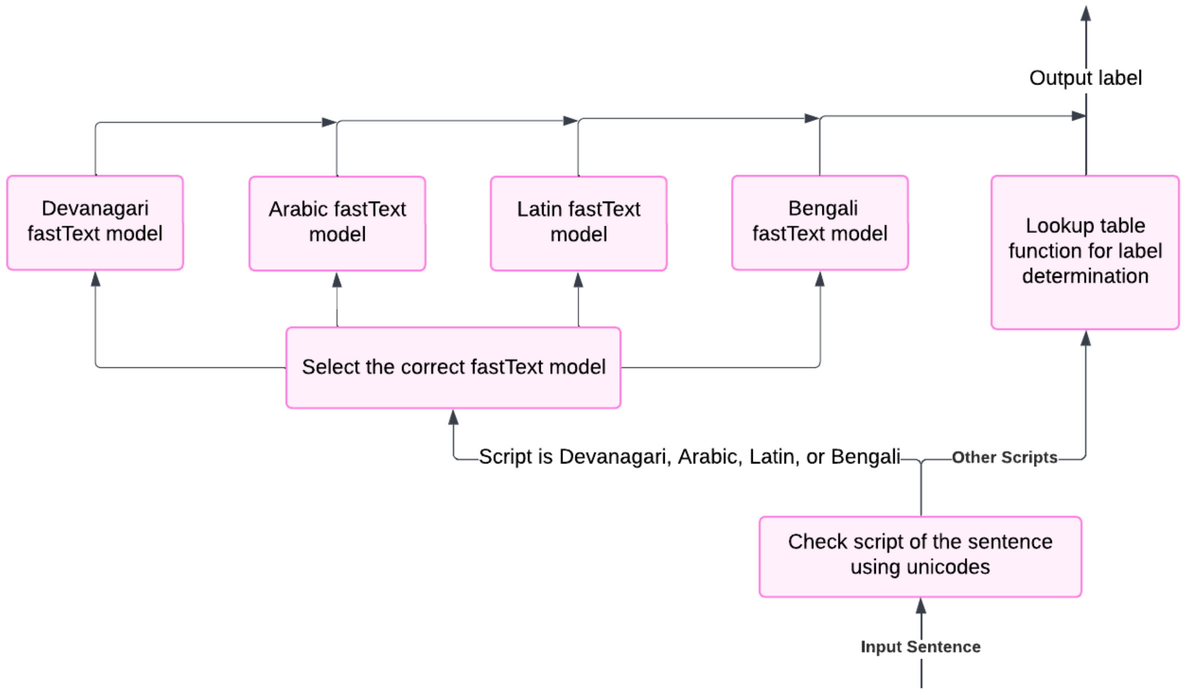

Output label
Devanagari fastText model
Arabic fastText model
Latin fastText model
Bengali fastText model
Lookup table function for label determination
Select the correct fastText model
Script is Devanagari, Arabic, Latin, or Bengali
Other Scripts
Check script of the sentence using unicodes
Input Sentence

Figure 4.4: fastText + Unicode-Range based sentence based language identification for native languages in a single script dataset.

4.2.3 fastText and Unicode-Range based word level

For intra-sentential code-mixing, we propose a word-level tagging system that builds upon sentence-level language identification system. First, a word-level labeled dataset is used to train the system. The sentence is then split into individual words, and each word's script is determined using a Unicode-Range based script classifier. For words that do not fall into one of the four major scripts—Devanagari, Bengali, Arabic, or Latin—the script itself can be used to tag the word.

For words in one of these four scripts, the data is further classified using fastText models trained on word-level subsets. Separate fastText classifiers are trained for each script on subsets of word-level data, enabling accurate language identification for each word. Using this language information along with the word's script and position within the sentence, a system then generates language labels for each word. This approach combines script identification and language classification, ensuring accurate tagging of each word within code-mixed sentences.

4.3 Conclusion

In this chapter, we proposed a structured approach to tackle the challenges of word-level language tagging for Romanized code-mixed Indic languages. We developed a native script dataset covering 31 Indic languages, annotated at the sentence level, and utilized fast-Text and Unicode-Range based techniques for sentence-level language identification. This combined approach accelerates model training across various scripts and language pairs.

Chapter 5

Experimental Setup

5.1 Setup

The proposed approach is tested on a workstation featuring an Intel Xeon W-1370P processor with 16 cores and two threads per core, coupled with 125GB of RAM and equipped with an NVIDIA T400 GPU with 1867 MiB of VRAM, utilizing NVIDIA driver version 470.239.06. The operating system used was Ubuntu 20.04.6 LTS.

5.2 Dataset

For current work in the two models, we only use the single script sentences in the Native script dataset.

5.3 Model parameters and Hyperparameters

In both fastText and a combination of fastText and Unicode-range based models, the model parameters are inspired by the findings in the Bhasha-Abhijnaanam[MKK23] paper, where these settings were identified as optimal for language identification tasks in Indic languages. The only hyperparameter tuned is the dimensionality (dim), while other settings, including loss function, verbosity level, and autotuning configuration, remain as fixed model parameters.

According to the Bhasha-Abhijnaanam[MKK23] study, extensive experimentation with the model's dimensionality showed that increasing dimensions beyond 8 did not significantly improve performance, though it did increase model size. Thus, a dimensionality of 8 was chosen to balance model performance and size. The same dimensionality setting has been applied here, reflecting its suitability for lightweight and efficient language identification.

Both models have used these parameters and hyperparameter values in our current work for supervised training of fastText models.

Parameter	**Value**
Dimension (Hyperparameter)	8
Loss Function	Hierarchical Softmax (hs)
Verbose Level	1
Autotune Validation File	validation_data.txt
Autotune Duration	43200 seconds

Table 5.1: fastText Parameters and Hyperparameters

5.4 Evaluation Metrics

We evaluate our system across multiple metrics, each capturing different aspects of performance. The evaluation metrics are as follows:

- **Precision**: Precision is the ratio of true positive predictions to the sum of true positive and false positive predictions. It measures the accuracy of positive predictions and is formulated as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall**: Recall is the ratio of true positive predictions to the sum of true positives and false negatives. It measures the system's ability to identify all relevant instances and is calculated as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **F1 Score**: The F1 score is the harmonic mean of precision and recall, providing a balanced metric that considers both false positives and false negatives. It is calculated as:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Accuracy**: Accuracy is the ratio of correct predictions (both true positives and true negatives) to the total number of predictions. It provides an overall measure of how often the model makes correct predictions and is given by:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}}$$

- **Throughput**: Throughput refers to the number of records processed by the system per second. This metric evaluates the efficiency and speed of the system, particularly in real-time or high-volume processing environments. It is calculated as:

$$\text{Throughput} = \frac{\text{Total Records Processed}}{\text{Total Time (seconds)}}$$

Each metric is plotted against varying dimensions to observe its behavior and impact on system performance. These plots help identify optimal configurations for achieving high accuracy, balanced precision and recall, and maximum throughput.

Chapter 6

Results and Analysis

6.1 Model Selection

In this study, we evaluate and compare the two models. Each model was assessed based on its precision, recall. F1 score, accuracy and throughput to determine its suitability for real-world applications.

6.1.1 Analysis of Model Performance

1. **Model 1: High Accuracy in Sentence-Level Language identification**

   The first model demonstrates high accuracy in identifying the language at the sentence level. This model effectively captures language distinctions within code-mixed text, ensuring reliable performance in scenarios where accuracy is prioritized. The precise identification of languages in sentence-level text makes this model suitable for applications that require rigorous language tagging.

2. **Model 2: Improved Throughput with Moderate Accuracy**

   The second model is optimized to achieve higher throughput, processing a larger number of records per second, albeit with slightly lower accuracy than Model 1. This trade-off between accuracy and processing speed makes Model 2 advantageous for real-time or large-scale language identification tasks where quick processing is essential. The streamlined architecture of this model ensures lower computational demands, thus enhancing throughput.

6.1.2 Model Performance

The performance of the fastText model in replicating the results from [MKK23] on our native dataset is summarized in the following table, highlighting accuracy, precision, recall, F1 score, and throughput across various configurations.

Model	**Precision**	**Recall**	**F1 Score**	**Accuracy**	**Throughput**
fastText	0.9919	0.9918	0.9918	0.9918	101777.0

Table 6.1: Performance comparison of fastText and fastText + Unicode-Range based models

Chapter 7

Conclusion and Future Work

In phase 1 of this BTP, we have explored sentence-level language identification using two different models: a lightweight linear classifier based on fastText and a custom-enhanced version of the fastText model, improved with methods tailored specifically for this dataset to accelerate training and testing. These models provided insights that helped us propose a design for a word-level language tagging model for Romanized code-mixed text. Our work on the dataset includes the development of a sentence-level labelled dataset and the initial creation of a word-level labelled dataset specifically for Romanized code-mixed sentences across 22 Indic languages.

We plan to expand the word-level tagged dataset for future work significantly, ensuring comprehensive coverage across the 22 Indic languages in Romanized script. Our next steps include implementing the proposed model for intra-sentential code-mixed tagging at the word level. Building on the insights gained from this, we aim to develop a tagging system for single-script code-mixed word-level tagging in Romanized script. We will conduct a detailed evaluation of the word-level tagging model to assess its performance and practicality, and based on these insights, we will explore improvements in model design to enhance accuracy and efficiency. This work aims to contribute to broader advancements in Indic language processing in Romanized script within multilingual NLP contexts.

Bibliography

[AI16] Google AI. Compact language detector 3 (cld3). 2016. https://github.com/google/cld3.

[C$^+$20] Isaac A. Caswell et al. Language identification for low resource languages: A case study of tigrinya. *arXiv preprint arXiv:2009.10717*, 2020.

[CKG$^+$20] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzman, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[CL19] Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, 2019.

[CWT13] Simon Carter, Wouter Weerkamp, and Manos Tsagkias. Microblog language identification: Overcoming the limitations of short, unedited, and idiomatic text. *Language Resources and Evaluation*, 2013.

[D$^+$24] Abhimanyu Dubey et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[JGBM16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[JTJ17] David Jurgens, Yulia Tsvetkov, and Dan Jurafsky. Incorporating dialectal variability for socially equitable language identification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2017.

[K$^+$21] Simran Khanuja et al. Muril: Multilingual representations for indian languages. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

[K$^+$22] Simran Khanuja et al. Supervised models for multilingual token classification on low-resource indian languages. *arXiv preprint arXiv:2205.03983*, 2022.

[KCW$^+$22] J. Kreutzer, I. Caswell, Y. Wang, A. Wahab, and S. Wu. Quality at a glance: An audit of web-crawled multilingual datasets. *arXiv preprint arXiv:2201.08239*, 2022.

[KKG$^+$20] D. Kakwani, A. Kunchukuttan, D. Golla, A. Bhattacharjee, M. Khapra, and P. Kumar. Indicnlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

[KPM17] KPMG and Google. Indian languages - defining india's internet, 2017. https://www.kpmg.com.

[LB11] Marco Lui and Timothy Baldwin. langid.py: An off-the-shelf language identification tool. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.

[MKK23] Yash Madhani, Mitesh M. Khapra, and Anoop Kunchukuttan. Bhasha-abhijnaanam: Native-script and romanized language identification for 22 indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 816–826. Association for Computational Linguistics, 2023.

[N$^+$22] NLLB Team et al. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*, 2022.

[RSC$^+$17] Shyamanthe Rijhwani, Russell Sequiera, Monojit Choudhury, Kalika Bali, and C. Sandeep Maddila. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

[SBM$^+$14] Thamar Solorio, Emily Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Angel Chang, et al. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, 2014.

[SS21] Vivek Srivastava and Mayank Singh. Challenges and considerations with code-mixed nlp for multilingual societies. 2021.

[XCR$^+$21] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2021.