Language Identification for Indian Languages: A Native and Transliterated Script Approach

Devika Singh
Indian Institute of Technology Guwahati / Guwahati, Assam, India
devikas.iitg@gmail.com

Abstract

Language detection for transliterated text and native script is essential for processing multilingual social media content, enabling better text classification, sentiment analysis, and machine translation. This paper presents a dataset containing both native script and transliterated Latin script text, annotated with language and script labels for Indian languages. We introduce two models: one for sentence-level language detection in native scripts and another for sentence-level detection in transliterated Latin scripts. Our approach focuses on language identification in social media text at the sentence level, highlighting the practical applications of our dataset and models in this domain.

1 Introduction

Language identification for both native script and transliterated text is crucial for multilingual natural language processing, particularly in a linguistically diverse country like India. With over 22 scheduled languages and a significant presence of transliterated content in digital communication, accurately detecting languages in both native and Romanized scripts is essential for downstream tasks such as translation, sentiment analysis, and information retrieval. However, existing datasets for Indian languages are often incomplete, outdated, or lack sufficient representation in both native and transliterated forms.

This paper introduces a large-scale dataset[1] for sentence-level language identification in both native script and transliterated Latin script across Indian languages. The native script dataset was constructed by collecting text from social media and consolidating publicly available datasets. To refine this dataset, we trained a language identification model and applied it to additional data, filtering out low-confidence predictions and retaining only sentences with a model-assigned confidence score above 90%. The transliterated dataset was developed using a combination of web crawling, Wikipedia data, transliteration websites, and existing datasets. For resource-scarce languages, we leveraged IndicXlit (Bhat et al., 2021) to generate transliterated text from native scripts, ensuring a minimum dataset size of 10,000 sentences for most languages.

To improve reliability, all sentences in our dataset contain at least three words, as shorter sentences were found to negatively impact model performance. We employ partial human validation through random sampling and native speaker verification, comprehensive manual validation remains part of future work.

Our dataset and models provide a valuable resource for Indian language identification and contribute to ongoing research in multilingual NLP. By addressing key gaps in existing datasets, we enable better handling of social media text, transliterated content, and underrepresented Indian languages.

2 Related works

Existing transliteration corpora for Indic languages include Aksharantar and Dakshina. Aksharantar is the largest publicly available parallel corpus, containing 26 million word pairs across 20 Indic languages (AI4Bharat, 2022). However, it does not cover all 22 scheduled Indian languages, and its data volume is relatively limited in both native and Roman script.

The Dakshina dataset (Roark et al., 2020) provides text in both Latin and native scripts for 12 South Asian languages. It includes a large corpus of native script Wikipedia text, a romanization lexicon consisting of words in the native script with attested romanizations, and a subset of full-sentence parallel data in both the native script and the basic Latin alphabet.

[1] Model and dataset will be made publicly available after acceptance of paper

While these datasets serve as important resources, our dataset expands upon them by ensuring a minimum of 10,000 sentences in most languages, except for Latin script data in Dogri, Konkani, and Santali, where available resources are scarce. Additionally, Dakshina covers only 12 languages, whereas our dataset provides broader language coverage. Given the multilingual nature of Indian social media text, our dataset enhances existing resources and serves as a valuable addition for processing of Indian languages.

3 Dataset

Scheduled languages of the Indian Constitution are Assamese, Bengali, Bodo, Dogri, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Maithili, Malayalam, Manipuri, Marathi, Nepali, Odia, Punjabi, Sanskrit, Santali, Sindhi, Tamil, Telugu and Urdu. Our dataset consists of two components: a native script dataset and a transliterated (Latin script) dataset. All sentences in the dataset contain more than three words, as it was observed that shorter sequences negatively impacted model performance and reduced reliability. Filtering out sentences below this threshold ensured more robust language identification and improved overall dataset quality.

3.1 Native Script Dataset

The native script dataset was created using two approaches: collecting data from social media sources through web crawling and consolidating publicly available datasets. This initial dataset served as the foundation for training a native script language identification model. The trained model was then applied to additional unlabeled data, filtering was applied to retain only sentences labeled by the native script model with a confidence score exceeding 90%, contributing to the final version of the native script dataset. Languages Kashmiri, Manipuri have data in multiple scripts.

3.2 Latin Script Dataset

The Latin script dataset was constructed using data collected from social media, web crawling Wikipedia and language-specific transliteration websites, and consolidating publicly available datasets. In addition to these sources, for languages with limited available resources, we utilized IndicXlit (Bhat et al., 2021) to transliterate native script data into Roman script. This approach was applied to Assamese, Kashmiri (from Arabic script), Maithili, Nepali, Malayalam, Manipuri, Oriya, and Sanskrit to ensure that the dataset size exceeded 10,000 samples for these languages. However, transliterated data for Santali and Dogri was not found, and Bodo and Konkani had significantly lower data availability.

4 Native Model

This model performs sentence-level language identification for native script text. Among Indian languages, 15 have unique scripts, allowing us to use a Unicode-based preprocessing method to determine sentences in these languages. The script with the highest characters is identified as the main script if it exceeds a threshold of 90% of the sentence length. If this main script corresponds to a language with a unique script-language pair, we directly assign the language and script label. Otherwise, we apply a FastText (Bojanowski et al., 2017) model, selecting from four separate models, for scripts Devanagri, Arabic, Bengali and Latin, trained on data of script groups. For sentences without a dominant script, we use a general model trained on all native scripts, providing a result only if confidence is high; otherwise, the sentence is marked as unidentifiable.

5 Roman Model

In this model, we use the FastText (Bojanowski et al., 2017) model to train a sentence-level language identification model for Indian languages written in Latin script. The dataset includes sentences across all 22 scheduled Indian languages, ensuring robust coverage. Since transliterated text lacks script-based cues, the model learns language-specific patterns directly from textual data. By leveraging FastText's subword embeddings, we can improve identification accuracy. Sentences are classified using a trained FastText model, and predictions are retained only if the confidence exceeds a predefined threshold. Otherwise, the sentence is labeled as unidentified.

.

6 Experimental Analysis

Our experiments were conducted using GPU acceleration for IndicXlit-based transliteration and CPU-based processing for language identification models. The script-wise models for Arabic, Devanagri and Bengali were trained and evaluated on their respective datasets, splitting them into training and testing sets. For evaluation of the combined native

Algorithm 1 Language Detection on Native Script

```
function LANGUAGEDETECTIONNATIVESCRIPT(models, sentence)
    script ← GETMAJORSCRIPT(sentence)
    if script.percentage > thresholdScript then
        if script.value in [Arabic, Bengali, Devanagri] then
            return PREDICTLANGUAGE(models[script.value], sentence)
        else
            return lookupTable[script.value]
        end if
    else
        return undetectable
    end if
end function
```

Language	Script	Native	Latin
Kashmiri	Arabic	196291	30437
Sindhi	Arabic	27048	12288
Urdu	Arabic	12164	12054
Assamese	Bengali	105509	24556
Bengali	Bengali	631860	13366
Manipuri	Bengali	68571	38164
Awadhi	Devanagri	1071	-
Bhojpuri	Devanagri	351657	-
Bodo	Devanagri	997	1429
Dogri	Devanagri	1015	-
Hindi	Devanagri	178295	13367
Kashmiri	Devanagri	1017	30437
Konkani	Devanagri	1009	1441
Magadhi	Devanagri	1004	-
Maithili	Devanagri	3520	10438
Marathi	Devanagri	372747	12340
Nepali	Devanagri	128623	10465
Sanskrit	Devanagri	182885	17518
Sindhi	Devanagri	1012	12288
English	English	55887	55887
Mizo	English	195184	237168
Gujarati	Gujarati	842348	13090
Punjabi	Gurmukhi	35052	12440
Kannada	Kannada	35784	13131
Malayalam	Malayalam	34221	11799
Manipuri	Meetei Mayek	997	38164
Santali	Ol Chiki	9094	-
Odia	Odia	38936	10423
Tamil	Tamil	40098	12463
Telugu	Telugu	41261	11847

Table 1: Languages, their Scripts, and Samples in Native and Latin Scripts

script model, we used a separate test set consisting of unseen sentences from the datasets od training and teting of the script-wise model, along with dedicated data for languages with unique script-language pairs. The model's performance was assessed using standard classification metrics, and we analyzed the covariance matrix of predictions (Figure X) to understand cross-language misclassification patterns.

6.1 Evaluation of the Native Script Model

The testing sets of the nine Indian languages with unique scripts helped validate the effectiveness of our Unicode-based preprocessing. The model achieved high accuracy in classifying languages with distinct scripts. However, misclassifications were observed among languages that share scripts. Overall, the throughput was improved, as the Unicode-based preprocessing reduced processing time for parts of the test dataset from these nine Indian Languages in their native scripts.

6.2 Evaluation of the Roman Script Model

For the transliterated Latin script model, we trained and evaluated it using data from the 22 Indian languages. The dataset was split into training and testing subsets, ensuring a balanced distribution across languages. The model's performance was assessed using precision, recall, F1-score, and accuracy. Unlike the native script model, which benefits from script-based cues, the Roman script model relies solely on linguistic patterns, making it more susceptible to ambiguity between languages with similar phonetic structures. The results show that high-resource languages achieve strong classification performance, while low-resource languages, such as Konkani, struggle due to limited training

Model	Precision	Recall	F1-score	Accuracy	Throughput
Devanagari (Native)	0.9866	0.9865	0.9865	0.9865	38258.96
Bengali (Native)	0.9917	0.9917	0.9917	0.9917	45816.03
Arabic (Native)	0.9998	0.9998	0.9998	0.9998	36314.46
Native Model	0.9987	0.9988	0.9993	0.9988	33572.07
Latin Model	0.9878	0.9874	0.9875	0.9874	32452.77

Table 2: Performance metrics for different models.

Model	Precision	Recall	F1-score	Accuracy	Throughput
Unicode-based	0.9987	0.9988	0.9993	0.9988	33572.07
Native (Single Model)	0.9917	0.9917	0.9917	0.9917	45816.03
Confidence-based filtering	0.9987	0.9988	0.9993	0.9988	33572.07
No filtering	0.	0.	0.	0.	.

Table 3: Performance metrics for different models.

data. According to the confusion matrix, Konkani is frequently misclassified as Marathi, highlighting the challenge of differentiating closely related languages in the transliterated script due to less training data.

7 Ablation Studies

In this section, we present various ablation studies that informed the design choices for both the native script and transliterated Latin script models. The results of these experiments are summarized in Table 3.

7.1 Effect of Unicode-Based Preprocessing on Native Script Classification

To evaluate the impact of Unicode-based preprocessing, we compared two approaches: (i) using Unicode-based script detection to first determine the script before language classification with script based models, and (ii) training a single model for direct language classification across all native scripts without preprocessing. The second approach involved pooling data from all native scripts into a single model without leveraging script-specific cues. The results indicate that the Unicode-based preprocessing method significantly improved classification accuracy, particularly for languages with unique scripts, as it helped eliminate ambiguity and reduced throughput.

7.2 Impact of Confidence -Based Filtering on Prediction Reliability

we analyzed the effect of applying a 90% confidence threshold to the native script model's predictions. Instead of accepting all predictions, only those with a confidence score above the threshold were retained, reducing the inclusion of uncertain classifications. The results indicate that this filtering approach enhanced precision and overall classification accuracy by reducing misclassified instances. However, it also introduced a trade-off in recall, particularly for low-resource languages, where fewer sentences met the confidence threshold. It can limit model generalization for languages with inherently lower representation in the dataset.

8 Conclusion

This paper presents a comprehensive dataset and models for sentence-level language identification in both native script and transliterated Latin script for Indian languages. Our dataset expands upon existing resources by ensuring broader language coverage and incorporating high-confidence automated labeling to improve data quality.

Experimental results demonstrate that the native script model performs well for languages with unique scripts, benefiting from script-based distinctions. However, languages sharing scripts exhibit some misclassification errors. The Roman script model, trained on transliterated text, achieves high accuracy for resource-rich languages but faces challenges in differentiating closely related languages due to phonetic similarity, particularly in low-resource cases like Konkani.

Our work enhances existing multilingual NLP applications by improving language processing for Indian languages in social media, transliteration tasks, and multilingual communication systems.

Limitations

A key limitation of this model is its inability to accurately identify multiple languages in code-mixed sentences where no single language is dominant. In such cases, the model assigns only one language label instead of detecting both. Given that India is a multilingual society, social media text often contains a mix of languages rather than a single language per sentence. Addressing code-mixed language detection is crucial for improving real-world applicability and ensuring better handling of multilingual digital communication.

Our dataset relies on partial human validation. We employ random sampling, where selected sentences are evaluated by native speakers to assess accuracy and quality. While this method provides valuable insights, it does not guarantee comprehensive validation across the entire dataset. Random sampling is a recognized approach in dataset validation, offering a balance between resource constraints and quality assurance. However, it may not fully capture all errors or biases present in the dataset.

References

AI4Bharat. 2022. Aksharantar: Open-source transliteration dataset for indic languages.

Irshad A Bhat, Mitesh M Khapra Jain, and Raghavan Kumar. 2021. Indicxlit: A multilingual transliteration benchmark for indic languages. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3311–3324.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Brian Roark, Raghavan Kumar, Sarah King, Ryan Cotterell, and Lakshmi Krishnan. 2020. The dakshina dataset: Supporting multilingual modeling of south asian languages. *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3787–3792.