Romanized to Native Malayalam Script Transliteration Using an Encoder-Decoder Framework

Bajiyo Baiju, Kavya Manohar, Leena G Pillai, Elizabeth Sherly
Digital University Kerala
Thiruvananthapuram
Kerala, India

Abstract

In this work, we present the development of a reverse transliteration model to convert romanized Malayalam to native script using an encoder-decoder framework built with attention-based bidirectional Long Short Term Memory (Bi-LSTM) architecture. To train the model, we have used curated and combined collection of 4.3 million transliteration pairs derived from publicly available Indic language transliteration datasets, *Dakshina* and *Aksharantar*. We evaluated the model on two different test dataset provided by *IndoNLP-2025-Shared-Task* that contain, (1) General typing patterns and (2) Adhoc typing patterns, respectively. On the Test Set-1, we obtained a character error rate (CER) of 7.4%. However upon Test Set-2, with adhoc typing patterns, where most vowel indicators are missing, our model gave a CER of 22.7%.

1 Introduction

Typing in native script has always remained a challenge for speakers of many Indian languages including Malayalam, across diverse digital platforms. In the pre-smartphone era, where native language typing was virtually non-existent due to the unavailability of accessible and user-friendly keyboards, typing Malayalam in the Roman script was the norm. Even with advancements in technology, typing in the Roman script has become the natural and preferred mode of input across devices for an average user (Madhani et al., 2023). While romanized communication seems convenient, it is not preferred in formal contexts.

Transliteration from romanized input to native scripts is inherently complex due to variations in typing styles, the absence of standardized romanization schemes, and the context-dependent nature of character mappings. Hence there is a need for real-time transliteration tools that can seamlessly convert romanized Malayalam into its native script.

In this work, we address this need by developing a robust reverse transliteration model for Malayalam, where romanised Malayalam is automatically converted into native script.

The proposed model leverages an attention-based bidirectional Long Short Term Memory (Bi-LSTM) encoder-decoder framework, trained on large-scale transliteration datasets, namely *Dakshina* (Roark et al., 2020) and *Aksharantar* (Madhani et al., 2023). The code for training the model is published under MIT License [1]. This paper outlines the related works, datasets, model architecture and results, highlighting the model's performance on datasets that reflect both general and adhoc typing patterns.

2 Related Works

Rule-based and data-driven approaches are the two main strategies for transliteration (Manohar et al., 2022). Prior to the advent of deep learning approaches of learning from huge data, rule based approaches were the norm. In the context of well defined romanization standards (Transliteration, 2001), rule based approaches are the best in terms of speed and accuracy. However there are non-standard romanised Malayalam used in informal communication contexts, that calls for deep learning solutions.

A rule based system available for transliteration among Indian languages based on soundex algorithms is introduced in *Libindic* (Thottingal, 2018). *Aksharamukha* script converter is is another rule-based systems that transliterates among 121 scripts and 21 standard romanization methods (Rajan, 2018). The *Brahmi-Net* tool covers 306 language pairs across 13 Indo-Aryan, 4 Dravidian languages, and English, utilizing an unsupervised method to mine parallel transliteration corpora for

---

[1] https://github.com/VRCLC-DUK/ml-en-transliteration

statistical training. This hybrid system leverages Unicode ranges and an extended ITRANS encoding to enable script conversions between Brahmi-derived scripts (Kunchukuttan et al., 2015).

Deep learning approaches rely on carefully crafted transliteration corpora for training the models. *Dakshina* is an open licensed and curated transliteration corpora (Roark et al., 2020) consisting of native script text, a romanization lexicon and some romanized full sentences in 12 south Asian languages. *Aksharantar* is the largest publicly available transliteration dataset with 26 million transliteration pairs for Indian languages created by mining from monolingual and parallel corpora, as well as collecting data from human annotators (Kunchukuttan et al., 2021; Madhani et al., 2023). It has also been reported that mined name pair datasets (Thottingal, 2023) could be used for training general purpose transliteration models (Baiju et al., 2024).

A multitask learning based training for multilingual neural transliteration leveraging orthographic similarity between languages was described in (Kunchukuttan et al., 2018). Non-neural method like pair *n-gram* and nueral methods like sequence-sequence LSTM and transformer architectures were compared in Roark et al. for single words transliteration task. Transliteration implemented using neural machine translation system (NMT) was proposed by Kunchukuttan et al., where *Marian* (Junczys-Dowmunt et al., 2018) was used for training the model. IndicXlit is a multilingual neural transliteration model trained on the *Aksharantar* (Madhani et al., 2023) dataset using an encoder-decoder transformer architecture. Grapheme to Phoneme Conversion systems for mapping of Malayalam script to precise romanisation schemes have been explored in rule based (Baby et al., 2016; Parlikar et al., 2016; Manghat et al., 2020) and data driven (Priyamvada et al., 2022) fashions.

Transliterating sentences are considered as a different task than transliterating single words in (Roark et al., 2020). Identifying word contexts can improve sentence level transliteration. Kirov et al. describes methods to incorporate language models to improve transliteration of full sentences as opposed to single words.

3 Methodology

The methodology involved in this study encompasses the curation and preprocessing of training datasets, design of the model architecture, training, and evaluation on the test data set.

In the current work, we train word-level transliteration model. During testing, we preprocess sentences by extracting individual words, performing word-level transliteration, and then reconstructing the full sentence in the post-processing stage. Non-alphabetic characters like punctuation and numbers are excluded from model input, preserved in their original positions, and reinserted after generating the transliterated token sequence.

3.1 Datsets

The reverse transliteration model for romanized Malayalam is trained on two publicly available curated collection of Indic language transliteration datasets: *Dakshina*[2] and the *Aksharantar*[3]. The *Dakshina* dataset comprises of 244 thousand single word transliteration pairs, while the *Aksharantar* dataset adds a significantly larger volume of 4.100 million pairs. Together, these datasets comprise a total of 4.344 million word-level transliteration pairs. The combined dataset ensures a rich and diverse training set that includes both common and rare transliteration patterns, capturing variations in typing styles and phonetic representations.

Each entry in the dataset is structured as a pair of columns: 'ml' and 'en'. The 'ml' column represents the native Malayalam script, while the 'en' column contains the corresponding romanized representation. This consistent and simple structure facilitates efficient preprocessing and model training, enabling the encoder-decoder framework to learn the mapping between the romanized input and the native script output effectively.

3.2 Model Architecture and Training

The proposed reverse transliteration model for converting romanized Malayalam to native script is based on an attention-enabled encoder-decoder framework utilizing Bi-LSTM layers. We define separate source and target tokenizers. The source tokens are lower case Latin characters and the target tokens are Malayalam characters comprising

---

[2]https://github.com/google-research-datasets/dakshina
[3]https://huggingface.co/datasets/ai4bharat/Aksharantar

Table 1: An illustration of 3 character errors distributed across 3 words severely deteriorating WER and BLEU scores. The errors in transliteration are indicated in red color.

Ground Truth	Predictions	CER(%)	WER(%)	BLEU(%)
കനകദ്ദർഗയ്യം വിളയോടി ശിവൻകട്ടിയ്യും വിവാഹിതരായി	കനകട്ടർഗയ്യം വിലയോടി ശിവൻകട്ടിയ്യുംഓ വിവാഹിതരായി	6.8	75.0	8.03

of vowels, vowel signs, consonants and the special characters *anuswaram*, *visargam*, *virama* and *chillu* (Manohar et al., 2022).

The architecture begins with the encoder input layer, which accepts input sequences of up to 57 characters, which is identified as a maximum input sequence length from the training data. These sequences are integer-encoded representations of characters, serving as the foundation for subsequent layers. The next step involves the embedding layer, which transforms each character in the sequence into a 64-dimensional dense vector which allows the model to capture semantic relationships among characters in a continuous space.

Following this, a bidirectional LSTM layer processes the embedded input sequences to capture information from both past and future characters in the sequence. The bidirectional output, consisting of hidden states from both directions, is concatenated to form a 256-dimensional representation for each timestep. To reduce dimensionality and adjust the feature representation, a dense layer is applied, resulting in a 128-dimensional vector for each timestep. The context vector extracts from this processed sequence, and it serves as the initial input to the decoder.

The decoder begins with the repeat vector layer, which duplicates the context vector for each timestep of the target sequence. This ensures that all decoder timesteps have access to the same initial context. The repeated context vector is processed by an LSTM layer in the decoder, which generates a sequence of hidden states by modeling temporal dependencies in the target sequence. These states form the basis for the generation of the transliterated output. The model incorporates an attention mechanism (attention layer) to enhance its ability to focus on relevant parts of the input sequence during decoding.

The output of the LSTM decoder and the attention layer is concatenated to form a unified representation, combining temporal dependencies with context-aware features. This enriched representation is passed through a time-distributed dense layer, which applies a dense transformation to each timestep. The result is a sequence of probability distributions over the 76 output characters, from which the final transliterated word is constructed. A single Nvidia DGX A100 GPU with 80 GB RAM was used for training the model.

4 Results

Table 2: Evaluation metrics averaged over respective test datsets

Dataset	CER (%)	WER (%)	BLEU (%)
Test Set-1	7.4	34.5	32.7
Test Set-2	22.7	66.9	7.5

We evaluate our model's performance using the IndoNLP Shared Task dataset[4] for Malayalam. The test set is divided into two categories: Test Set-1, which includes general transliteration patterns, and Test Set-2, which features adhoc transliteration patterns where the romanized text omits several vowels. These datasets consist of sentence-level samples. Samples of ground truth and predicted samples in test sets are linked in the repository[5] and an example is given in Table 1. As recommended by the task organizers, we report CER, WER, and BLEU scores separately for each test set (Table 2). The distribution of these evaluation metrics over the entire test set is illustrated in Figure. 1 and Figure. 2.

5 Discussion

In Test Set-1 with standard typing patterns, the model achieved a 7.4% CER, demonstrating strong performance aligned with the model's training data. For Test Set-2 involving adhoc typing patterns with frequent vowel omissions, the model's performance significantly declined as indicated by the performance metrics.

---

[4] https://github.com/IndoNLP-Workshop/IndoNLP-2025-Shared-Task
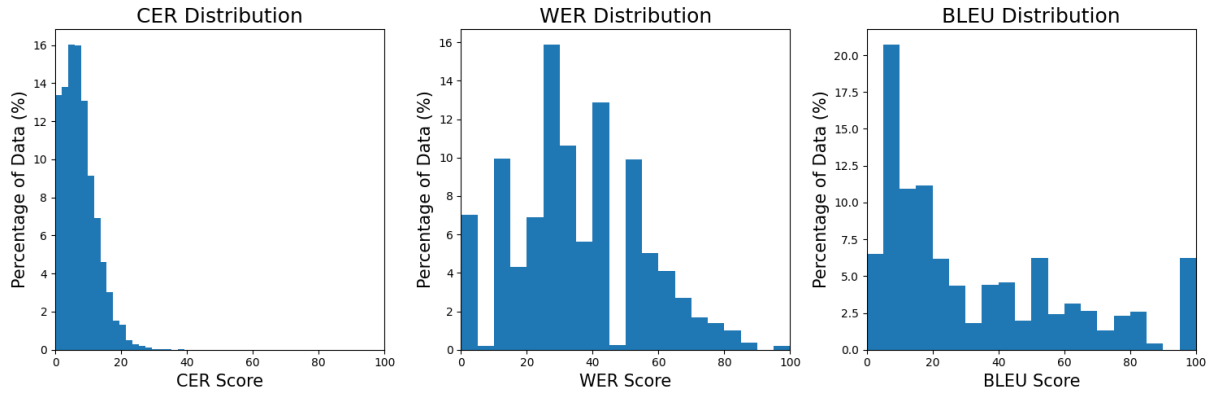[5] https://github.com/VRCLC-DUK/ml-en-transliteration

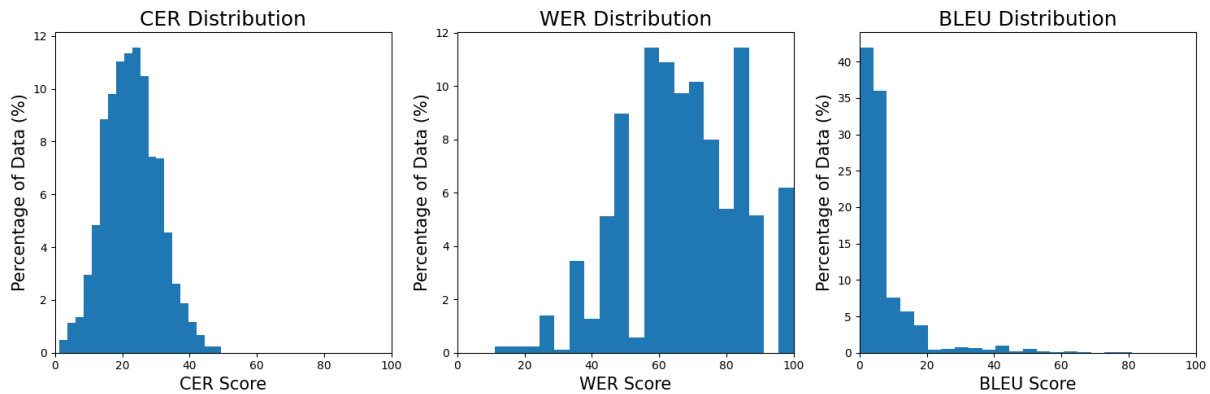Figure 1: The distribution of WER, CER and BLEU over the Test Set-1.

Figure 2: The distribution of WER, CER and BLEU over the Test Set-2.

While most test sentences exhibited low character-level error rates, the accompanying WER and BLEU scores appear comparatively poor. This does not indicate model inadequacy, but rather reflect the inherent limitations of WER and BLEU scores in evaluating sentence transliterations. As they penalize even minor character variations as complete word errors misrepresenting the transliteration quality (James et al., 2024) (Table 1). An error analysis exposed the model's difficulty in distinguishing phonetically similar Malayalam characters represented using same romanised form.

6   Conclusion

Our reverse transliteration model for Malayalam demonstrates promising capabilities in converting romanized text to native script, particularly for standard typing patterns. However, the research reveals significant challenges in handling adhoc typing styles, especially those with frequent vowel omissions. Future efforts should focus on fine-tuning the model using a diverse dataset that includes a significant proportion of adhoc typing patterns to enhance its robustness and adaptability.

Limitations

The training data primarily covers standard typing patterns and missing the nuanced variations found in irregular typing scenarios. This restricted training set significantly constrains the model's ability to generalize and accurately handle diverse input styles and patterns. Additionally, the model's design lacks a language model that could capture word dependencies and improve overall sentence-level transliteration.

References

Arun Baby, Anju Leela Thomas, NL Nishanthi, TTS Consortium, et al. 2016. Resources for Indian languages. In *Proceedings of Text, Speech and Dialogue*. CBBLR Workshop.

Bajiyo Baiju, Kavya Manohar, Leena G Pillai, and Elizabeth Sherly. 2024. Malayalam to English Named Entity Transliteration using Attention based BiLSTM. In *2024 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, pages 1–6.

Jesin James, Deepa P Gopinath, et al. 2024. Advocating character error rate for multilingual asr evaluation. *arXiv preprint arXiv:2410.07400*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Christo Kirov, Cibu Johny, Anna Katanova, Alexander Gutkin, and Brian Roark. 2024. Context-aware Transliteration of Romanized South Asian Languages. *Computational Linguistics*, pages 1–60.

Anoop Kunchukuttan, Siddharth Jain, and Rahul Kejriwal. 2021. A large-scale evaluation of neural machine transliteration for Indic languages. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3469–3475, Online. Association for Computational Linguistics.

Anoop Kunchukuttan, Mitesh Khapra, Gurneet Singh, and Pushpak Bhattacharyya. 2018. Leveraging orthographic similarity for multilingual neural transliteration. *Transactions of the Association for Computational Linguistics*, 6:303–316.

Anoop Kunchukuttan, Ratish Puduppully, and Pushpak Bhattacharyya. 2015. Brahmi-net: A transliteration and script conversion system for languages of the Indian subcontinent. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 81–85, Denver, Colorado. Association for Computational Linguistics.

Yash Madhani, Sushane Parthan, Priyanka Bedekar, Gokul Nc, Ruchi Khapra, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh Khapra. 2023. Aksharantar: Open Indic-language transliteration datasets and models for the next billion users. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 40–57, Singapore. Association for Computational Linguistics.

Sreeja Manghat, Sreeram Manghat, and Tanja Schultz. 2020. Malayalam-English Code-Switched: Grapheme to Phoneme System. In *Proc. Interspeech 2020*, pages 4133–4137.

Kavya Manohar, A. R. Jayan, and Rajeev Rajan. 2022. Mlphon: A multifunctional grapheme-phoneme conversion tool using finite state transducers. *IEEE Access*, 10:97555–97575.

Alok Parlikar, Sunayana Sitaram, Andrew Wilkinson, and Alan W Black. 2016. The Festvox Indic frontend for grapheme to phoneme conversion. In *WILDRE: Workshop on Indian Language Data-Resources and Evaluation*.

R. Priyamvada, D. Govind, Vijay Krishna Menon, B. Premjith, and K. P. Soman. 2022. Grapheme to phoneme conversion for malayalam speech using encoder-decoder architecture. In *Intelligent Data Engineering and Analytics*, pages 41–49, Singapore. Springer Nature Singapore.

Vinodh Rajan. 2018. Aksharamukha script converter web application.

Brian Roark, Lawrence Wolf-Sonkin, Christo Kirov, Sabrina J. Mielke, Cibu Johny, Işin Demirşahin, and Keith Hall. 2020. Processing South Asian languages written in the Latin script: the Dakshina dataset. In *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*, pages 2413–2423.

Santhosh Thottingal. 2018. Libindic soundex and transliteration module.

Santhosh Thottingal. 2023. Malayalam-English Name Pair Dataset.

ISO 15919:2001 Transliteration. 2001. Transliteration of Devanagari and related Indic scripts into Latin characters.