

# IndoNLP 2025: Shared Task on Real-Time Reverse Transliteration for Romanized Indo-Aryan languages

Deshan Sumanathilaka<sup>1\*</sup>, Isuri Anuradha<sup>2</sup>, Ruvan Weerasinghe<sup>3</sup>, Nicholas Micallef<sup>1</sup>, Julian Hough<sup>1</sup>

<sup>1</sup>Swansea University, Wales, UK

<sup>2</sup>Lancaster University, UK

<sup>3</sup>Informatics Institute of Technology, Sri Lanka

\*Corresponding email :deshankoshala@gmail.com

## Abstract

The paper overviews the shared task on Real-Time Reverse Transliteration for Romanized Indo-Aryan languages. It focuses on the reverse transliteration of low-resourced languages in the Indo-Aryan family to their native scripts. Typing Romanized Indo-Aryan languages using ad-hoc transliterations and achieving accurate native scripts are complex and often inaccurate processes with the current keyboard systems. This task aims to introduce and evaluate a real-time reverse transliterator that converts Romanized Indo-Aryan languages to their native scripts, improving the typing experience for users. Out of 11 registered teams, four teams participated in the final evaluation phase with transliteration models for Sinhala, Hindi and Malayalam. These proposed solutions not only solve the issue of ad-hoc transliteration but also empower low-resource language usability in the digital arena.

## 1 Introduction

Languages transcend mere systems of communication. They are profound reflections of the cultures they represent. Embedded within each language are the accumulated wisdom, traditions, beliefs, and historical narratives cherished by its speakers. South Asia, a region renowned for its cultural richness and linguistic diversity, is home to a vast array of languages that serve not only as tools for communication but also as vibrant expressions of its multifaceted heritage.

Among these, the Indo-Aryan languages hold a significant place, forming part of the larger Indo-Iranian branch of the Indo-European language family. As of the early 21st century, the Indo-Aryan languages (sometimes called Indic languages) have more than 800 million speakers, primarily concentrated east of the Indus River in Bangladesh, North India, Eastern Pakistan, Sri Lanka, Maldives, and Nepal (KJ et al., 2024). Additionally, large immigrant and expatriate Indo-Aryan-speaking commu-

nities can be found in Northwestern Europe, Western Asia, North America, the Caribbean, Southeast Africa, Polynesia, and Australia. There are over 200 known Indo-Aryan languages, further emphasizing the immense diversity within this linguistic group (Talukdar and Sarma, 2023). Historically, the Indo-Aryan languages have evolved from Old Indo-Aryan (Sanskrit) through Middle Indo-Aryan (Prakrits) to their modern forms, such as Hindi, Bengali, Gujarati, Punjabi, and others. This historical evolution underscores their linguistic richness and role as carriers of cultural and historical narratives (Masica, 1993).

In contrast, the Dravidian languages constitute a distinct language family, primarily spoken in South India and parts of Eastern and Central India, as well as in northeastern Sri Lanka, Pakistan, Nepal, Bangladesh, and among diaspora communities worldwide. The Indo-Aryan and Dravidian languages illustrate South Asia's profound linguistic and cultural complexity, weaving a vibrant tapestry of communication and heritage that resonates far beyond the region (De Silva, 2019). With the advancement of digital technologies, communication on social media has become increasingly prominent. Social media users often employ both native scripts and Romanized versions of their native languages for accessible communication and compatibility with digital platforms. However, the transliteration of Indo-Aryan languages into native scripts remains under-explored, mainly due to a scarcity of language resources needed for their development. While numerous rule-based systems have been implemented to transliterate Indo-Aryan languages, significant challenges persist in addressing ad-hoc transliteration effectively. Social media users frequently rely on abbreviated and informal typing styles to communicate in their native languages. This variability in linguistic representation complicates the verification of hate speech and misinformation, as the diverse formats pose substantial

challenges for detection and analysis ([Sumanathilaka et al., 2023](#)).

The key contributions of this shared task are :

- Introducing novel models for Romanized scripts to Native Indo-Aryan Languages Transliteration.
- Introducing a standard evaluation dataset for five languages: Sinhala, Hindi, Bengali, Gujarati and Malayalam.

Moving forward, the paper will discuss related works, shared task overview, results and discussion along with the conclusions.

## 2 Related Works

Early back-transliteration methods for Indo-Aryan languages heavily relied on rule-based approaches, employing predefined character mappings and linguistic rules. For example, [Vidanaralage et al. \(2018\)](#) introduced a rule-based method for Sinhala transliteration using transliteration and phoneme rule bases. Similarly, [Tennage et al. \(2018\)](#) utilized character mapping tables for converting Sinhala into English. These methods demonstrated effectiveness in controlled environments but often faltered with informal Romanized text, such as Singlish, which includes significant variability and ambiguity. Researchers have combined rule-based methods with statistical models to overcome the limitations of purely rule-based systems. [Li-wera and Ranathunga \(2020\)](#) applied a trigram model trained on Romanized Sinhala YouTube comments, integrated with rule-based techniques, to address transliteration challenges. Hybrid models have shown promise but struggle with contextual dependencies and ambiguities in informal text. Classical machine learning models were explored to transliterate Hindi and Marathi to English efficiently ([Rathod et al., 2013](#)).

Recent advancements in deep learning have revolutionized transliteration tasks. Neural network-based sequence-to-sequence models have gained traction, particularly those utilizing recurrent architectures. [Kunchukuttan et al. \(2015\)](#) introduced Brahmi-Net, a transliteration system leveraging statistical approaches for 18 Indo-Aryan languages. [Nanayakkara et al. \(2022\)](#) employed a Bidirectional LSTM and LSTM for Sinhala transliteration, incorporating character-level context. Transformer architectures have emerged as state-of-the-art for transliteration due to their ability to handle

long-range dependencies with self-attention mechanisms. [Zohrabi et al. \(2023\)](#) employed Transformer-based models for Azerbaijani transliteration while [Madhani et al. \(2023\)](#) introduced IndicXlit, a multilingual system leveraging Transformer architecture. Pretrained multilingual models like mT5 and M2M100 have also been fine-tuned for transliteration tasks, treating transliteration as a translation problem. A deep learning approach proposed by ([Deselaers et al., 2009](#)) used a deep belief network for Arabic-English transliteration, and a few other Arabizi to Arabic transliteration are explored ([Masmoudi et al., 2019; Al-Badrashiny et al., 2014](#)).

Hybrid approaches combining rule-based, statistical, and deep learning methods have demonstrated increased accuracy and robustness. [Athukorala and Sumanathilaka \(2022\)](#) proposed a hybrid approach for Singlish transliteration, integrating rule-based methods with fuzzy logic and achieving a word-level accuracy of 0.64. This work has been further extended to Ngram with Rule based approach by [Sumanathilaka et al. \(2023\)](#) and later [Dharmasiri and Sumanathilaka \(2024\)](#) by using a GRU for efficient transliteration. TAMZHI by [Mudiyanselage and Sumanathilaka \(2024\)](#) developed a back-transliteration system for Romanized Tamil, achieving 93% character-level accuracy and 70% word-level accuracy. Pix2Pix Generative Adversarial Network for transliterating ancient Indian scripts, demonstrating the adaptability of GANs for image-to-text transliteration were further explored ([Sharma et al., 2023](#)).

The availability of datasets like Dakshina ([Roark et al., 2020](#)) and Aksharantar([Madhani et al., 2023](#)) has significantly enhanced transliteration model training. [Sumanathilaka et al. \(2024\)](#) has built a large corpus of data which consists of Sinhala and its Romanized Sinhala patterns along with a dataset to handle Word sense disambiguation in Romanized Sinhala scripts. These large-scale resources provide robust transliteration pairs, enabling models to learn from diverse and complex Romanized text patterns. However, the lack of standardized rules for transliteration leads to ad-hoc spelling variations, including the omission or alteration of vowels, inconsistent phonetic mappings, and user-specific adaptations. These inconsistencies create significant challenges in accurately converting Romanized text into native scripts. Existing keyboard systems and transliteration tools frequently fall short of addressing these complexities, often pro-

ducing inaccurate or unintelligible outputs ([Mammadzada, 2023](#)). This results in frustration for users, reduces the accessibility of native scripts, and hinders seamless communication in their preferred languages. Therefore, the demand for a robust and reliable solution to this problem is practical and urgent.

### 3 Shared Task Overview

#### 3.1 Shared Task Definition

The widespread use of Romanized Indo-Aryan languages and native languages expressed using the English alphabet has become a prevalent mode of communication in digital spaces. This practice is especially common in informal settings such as social media and messaging platforms ([Yadav et al., 2023](#)). However, the absence of standardized transliteration rules results in inconsistent spellings, phonetic mismatches, and user-specific adaptations, creating significant challenges for accurately converting Romanized text to native scripts and highlighting the need for reliable solutions.

To address these challenges, this shared task introduces a real-time reverse transliterator capable of converting Romanized Indo-Aryan language input into its accurate native script equivalent. The solution aims to improve the transliteration process significantly, providing a smoother and more user-friendly typing experience for speakers of Indo-Aryan languages who rely on Romanized input. By enabling accurate and efficient native script outputs, the project seeks to bridge the gap between linguistic preferences and digital accessibility.

Participants in this shared task are required to develop models capable of:

- Accepting Romanized text input for one or more Indo-Aryan languages.
- Producing accurate native script output in real-time or near-real-time settings.
- Handling ad-hoc transliterations, including inputs with inconsistent or missing vowels, abbreviations, and phonetic variations.
- Operating efficiently under constraints such as limited computational resources.

#### 3.2 Objectives of shared Task

The primary objective of this project is to develop a system that accurately converts Romanized Indo-Aryan language text into the native script in real-

Romanized Indo-Aryan Language	Native script
ayubown, Sapa sanepa khmda	ଆଯୁବୋନ, ସାପା ଶନେପା ଖମ୍ଦା
Nimaste, aap kaise hai	ନମସ୍କେ, ଆପ କୈସେ ହେ
Vnakkam eppti irukkiray	வணக்கம் எப்படி இருக்கிறாய்

Figure 1: Example cases of Input and Output.

time, ensuring high precision and usability. The system is designed to handle transliterations with or without vowels, effectively resolving ambiguities arising from such variations. Handling such ambiguities has been a major drawback in the current transliteration system used in commercial platforms ([AbeySiriwardana and Sumanathilaka, 2024](#)). By addressing the limitations and inaccuracies inherent in existing keyboard systems, the project aims to provide users with a seamless and efficient typing experience, enabling reliable communication in native scripts while accommodating the diverse and ad-hoc transliteration styles commonly used in digital interactions. Detailed input and output are shown in the Figure 1.

#### 3.3 Key Challenges to Address

Transliterating Romanized Indo-Aryan languages into native scripts presents several challenges that stem from linguistic and technological complexities. Ad-hoc transliterations are common, with users employing inconsistent spellings, including omitting or substituting vowels, which complicates accurate mapping to native scripts. The lack of standardization in Romanized spellings, which are often user-specific and devoid of formal rules, further contributes to significant variability. Phonetic ambiguity adds another layer of difficulty, as multiple Romanized representations can correspond to a single native script word and vice versa. Additionally, resource scarcity poses a major hurdle, with Indo-Aryan languages lacking comprehensive datasets for training and evaluating effective transliteration systems. To ensure usability, addressing these challenges while meeting real-time constraints and delivering low-latency, high-accuracy transliteration for seamless user interaction is essential.

#### 3.4 Datasets

##### 3.4.1 Training datasets

The proposed study leverages multiple comprehensive datasets for transliteration tasks. The first

dataset, the **Dakshina Dataset**<sup>1</sup>, provides a rich collection of text in native scripts and their Romanized counterparts for various Indo-Aryan languages. This dataset is particularly useful for training and evaluating models on diverse transliteration patterns. This dataset was mostly used by the participants to train and test their models. The second dataset, the **Aksharantar Dataset**<sup>2</sup>, offers an extensive repository of Romanized and native script pairs, focusing on Indian languages.

Additionally, for Sinhala, the **Swa-Bhasha Dataset**<sup>3</sup> is proposed, which contains a wide range of transliteration scripts specifically tailored for Romanized Sinhala to Sinhala. This dataset comprises four distinct types: unique words, Sinhala - Romanized Sinhala ad-hoc transliterations, Romanized Sinhala - Sinhala social media datasets, and the WSD Romanized Sinhala - Sinhala Transliteration Dataset. These datasets provide a robust foundation for developing and benchmarking the transliteration system, addressing challenges such as ad-hoc spellings, phonetic ambiguities, and resource scarcity. This dataset has been used by Team Vectora for their training process.

### 3.4.2 Test datasets

This test dataset has been created and augmented specifically for the **IndoNLP Shared Task**<sup>4</sup>. Please note that some data records are a combination of existing datasets that are publicly available for the respective languages. The augmentation process involved generating new data samples based on these existing resources while ensuring data diversity and relevance to the task. The results are presented using Word Error Rate (WER), Character Error Rate (CER), and BiLingual Evaluation Understudy (BLEU), which are standard metrics to evaluate the quality of transliteration systems, measuring errors and assessing linguistic fidelity. The data distribution is presented in the Table 1.

## 3.5 Participant's Systems

Table 2 presents the registered team information, their affiliations, and the primary languages they worked on. The discussion below provides an analysis of the users' proposed solutions.

<sup>1</sup><https://github.com/google-research-datasets/dakshina>

<sup>2</sup><https://github.com/AI4Bharat/IndicXlit>

<sup>3</sup><https://github.com/Sumanathilaka/Swa-Bhasha-Sinhala-Singlish-Dataset>

<sup>4</sup><https://github.com/IndoNLP-Workshop/IndoNLP-2025-Shared-Task/>

The **IndiDataMiner** team (Kumar et al., 2025) at the Indian Institute of Technology Guwahati proposed a sentence-level back-transliteration approach using the LLaMa 3.1 model for Hindi. Their approach addresses the challenges posed by the increasing use of Romanized typing for Indo-Aryan languages on social media, which often lacks standardization and results in loss of linguistic richness. To resolve the ambiguities in Romanized Hindi text, they leveraged fine-tuning with the Dakshina dataset. The team's approach includes both zero-shot learning and fine-tuning techniques to enhance the model's transliteration capabilities. The model was trained using the Dakshina dataset, which provides a parallel corpus of 12 Indian languages, including Hindi. The dataset was formatted to fit the Alpaca prompt structure, with the instruction to transliterate Romanized Hindi back to Devanagari script, ensuring consistency across training and testing. They used the LLaMA 3.1 8B model, a large-scale transformer-based architecture, which was optimized for causal language modeling and enhanced with Low-Rank Adaptation (LoRA) and 4-bit quantization. The training process utilized the SFTTrainer class from the trl library and the Unslot framework for 4-bit quantization, which improved memory efficiency. The fine-tuned model demonstrated significant improvements in transliteration accuracy, achieving high BLEU scores on the Hindi test dataset.

**Team Vectora**'s (Perera et al., 2025) approach to Sinhala back-transliteration is a context-aware system that combines multiple techniques to handle the complexities of "Singlish". Their method employs several sophisticated components that work together to achieve accurate transliteration. At the core of their system is a dictionary-based mapping approach that uses an ad-hoc transliteration dictionary to map common Singlish words to their Sinhala equivalents. This dictionary is particularly effective at handling frequent words and their different possible Sinhala representations. To accommodate the complexity of the language, the dictionary incorporates multiple mappings for ambiguous words. For handling words not found in the dictionary (out-of-vocabulary words), the system implements rule-based transliteration based on Sinhala phonetic patterns. This component proves essential for processing words that may not be common or are absent from their dictionary. To resolve lexical ambiguities where one Romanized word

Language	Test Set 1 : General Typing Patterns	Test Set 2 : Ad-hoc Typing Patterns
Sinhala	10,000	5,000
Bengali	10,000	5,000
Gujarati	5,000	5,000
Hindi	5,000	5,000
Malayalam	10,000	5,000

Table 1: Test sets for general and ad-hoc typing patterns across languages.

Team Name	Affiliation	Language
IndiDataMiner	Indian Institute of Technology Guwahati, India	Hindi
Vectora	Informatics Institute of Technology, Sri Lanka	Sinhala
MoraCSE	University of Moratuwa, Sri Lanka	Sinhala
NexText	Digital University Kerala, India	Malayalam

Table 2: Team Information and Language Distribution

can map to multiple Sinhala words, the system employs a BERT-based language model to analyze the sentence-level context and select the most appropriate Sinhala word. The contextual disambiguation process works through a sophisticated sequence of operations. The system first generates all possible sentences by filling the masked ambiguous words with candidate Sinhala words. These generated sentences are then evaluated using a BERT model configured for Masked Language Modeling (MLM). The final transliterated output is determined by selecting the sentence that achieves the highest probability score. To address processing time challenges, the system incorporates several optimization strategies. These include reducing candidate words for ambiguous words through a filtering mechanism and implementing sentence chunking based on the number of BERT calls. The effectiveness of the entire system is rigorously evaluated using multiple metrics.

The **Team MoreCSE** (De Mel et al., 2024) explored two distinct approaches for Sinhala transliteration: a rule-based system and a deep learning-based system. The rule-based approach implements a systematic method using predefined linguistic rules to map Latin script (Singlish) to Sinhala script. This system operates through a character-by-character matching strategy, processing each input word by matching the longest possible substring (up to three characters) with rules in a transliteration table. The system follows a straightforward logic: when a match is found, the corresponding Sinhala character is appended to the result; if no match is found, the character is added

unchanged. This method builds upon and enhances a previous rule-based system by incorporating additional rules for two and three-character mappings. Their deep learning approach takes a fundamentally different path by modelling transliteration as a translation task, leveraging a Transformer-based encoder-decoder architecture. The team fine-tuned the M2M100 model, a pre-trained multilingual sequence-to-sequence model, by treating Romanized Sinhala as the English input and the Sinhala script as the target language. This implementation utilizes the M2M100’s tokenizer and learns the intricate relationships between the two language pairs. The deep learning method offers several key advantages: it enables context-based generation, eliminates the need for manual rule definition, and can effectively handle code-mixed and code-switched cases through expanded training data. The model’s training was conducted on a substantial dataset comprising 10k parallel data points. Their analysis revealed that while the deep learning approach demonstrated superior robustness in handling language variability, it proved less computationally efficient than its rule-based counterpart.

The approach of **Team NexText** (Baiju et al., 2024) for Malayalam transliteration utilizes Bi-LSTM layers. They trained their model on a substantial combined dataset of 4.3 million transliteration pairs from the Dakshina and Aksharantar datasets. Their data preprocessing strategy operates at the word level, with the model trained on transliteration pairs. During testing, sentences undergo preprocessing, where individual words are

<b>Team</b>	<b>Model</b>	<b>Test</b>	<b>WER</b>	<b>CER</b>	<b>BLEU</b>
Team Vectora	Sinhala BERT	Test 1	0.0886	0.0200	0.9115
		Test 2	0.0914	0.0212	0.9088
	Finetuned BERT	Test 1	0.0850	0.0194	0.9151
		Test 2	0.0895	0.0210	0.9107
Team MoraCSE	Rule-based	Test 1	0.6689	0.2119	0.0177
		Test 2	0.6809	0.2202	0.0163
	DL-based	Test 1	0.1983	0.0579	0.5268
		Test 2	0.2413	0.0789	0.4384
Team IndiDataMiner	LLaMa 3.1	Test 1	0.2154	0.0881	0.5996
		Test 2	0.2851	0.1339	0.4879
	Proposed	Test 1	0.1892	0.0684	0.6288
		Test 2	0.2640	0.1183	0.5105
Team NexText	Bi-LSTM	Test 1	0.3450	0.0740	0.3270
		Test 2	0.6690	0.2270	0.0750

Table 3: Comparison of Model Performance Metrics Across Teams

extracted, transliterated independently, and then reconstructed into complete sentences. The system carefully preserves non-alphabetic characters, reinserting them after the complete transliteration process. The model architecture is comprehensive and multi-layered. It begins with an encoder input layer capable of processing up to 57 characters in length character sequences. These characters are transformed into 64-dimensional vectors through an embedding layer. A bidirectional LSTM layer then processes this information, capturing sequence patterns from both directions and creating a 256-dimensional representation. This representation is refined through a dense layer, which reduces the dimensionality to a 128-dimensional vector, creating a context vector that feeds into the decoder. The decoder architecture employs a repeat vector layer to duplicate the context vector for each timestep, followed by an LSTM layer for generating hidden states. An attention mechanism enhances the model’s ability to focus on relevant parts of the input sequence during the decoding process. The final stage combines the LSTM decoder output with the attention layer through concatenation, feeding into a time-distributed dense layer that produces a probability distribution over possible output characters. The training was conducted on a high-performance computing setup, specifically a single Nvidia DGX A100 GPU with 80 GB RAM. The evaluation was conducted on two distinct test sets: one focusing on general transliteration patterns and another featuring ad-hoc patterns with frequent vowel omissions. While the

model demonstrated strong performance on standard typing patterns, it showed some performance degradation when handling ad-hoc typing patterns.

#### 4 Shared Task Results and Discussion

The results from the Table 3 reveal insightful patterns in the performance of various transliteration approaches across Indo-Aryan languages. Team Vectora’s Sinhala transliteration models, utilizing Sinhala BERT and Finetuned BERT, exhibited exceptional performance, achieving BLEU scores around 0.91 and remarkably low WER (below 0.09) and CER (around 0.02) for both general and ad-hoc typing patterns. This consistency across diverse test scenarios indicates the robustness of these models in handling both standard transliteration tasks and more challenging cases with omitted vowels. These results position Team Vectora’s BERT-based approaches as a strong benchmark for Sinhala transliteration. In stark contrast, Team MoraCSE’s rule-based approach faced significant difficulties, particularly for Sinhala. With a high WER of around 0.67 and very low BLEU scores (below 0.02), it is evident that the rule-based system struggled to address the complexities inherent in Sinhala transliteration. On the other hand, their deep learning (DL)-based approach showed improvement, yielding BLEU scores between 0.43 and 0.52, and WER ranging from 0.19 to 0.24, though these results still fell short of Team Vectora’s performance. For Hindi, Team IndiDataMiner’s models demonstrated moderate success. The proposed model outperformed their LLaMa 3.1 implementation in Test 1 (gen-

eral typing patterns), achieving a BLEU score of 0.6288 compared to 0.5996. However, both models showed noticeable degradation in performance in Test 2 (ad-hoc typing patterns), reflecting challenges in dealing with irregular vowel combinations. Team NeXText’s Bi-LSTM model for Malayalam exhibited interesting characteristics. While it achieved a relatively low WER of 0.074 for general typing patterns, there was a marked decline in performance for ad-hoc typing patterns (WER increased to 0.227). The high CER values (0.345 and 0.669) and low BLEU scores (0.327 dropping to 0.075) indicated that the model struggled with character-level accuracy, particularly when dealing with irregular typing patterns.

The key observations are listed below.

- BERT-based models (Team Vectora) demonstrated superior performance and consistency across different typing patterns, setting a strong benchmark for other transliteration tasks in Indo-Aryan languages.
- Deep learning models consistently outperformed rule-based approaches across all languages, highlighting the advantages of neural methods in handling complex language structures.
- Ad-hoc typing patterns (Test 2) posed challenges for all models, with performance degradation observed across the board, although the extent varied.
- Language-specific characteristics were crucial in model performance, with Sinhala models, particularly Team Vectora’s, proving to be the most robust in handling different typing patterns.

These results emphasize that while deep learning models outperform traditional rule-based methods, the specific architecture and training approach, particularly the use of BERT-based models, significantly impacts performance. Team Vectora’s success in Sinhala transliteration provides a promising direction for future work in Indo-Aryan language transliteration tasks.

## 5 Conclusion

In this paper, we present the results of the IndoNLP 2025 shared task, which addresses the challenges of

transliteration on Indo-Aryan languages. The participating teams’ findings highlight these tasks’ ongoing challenges and research gaps. In conclusion, the results of this study underscore the superiority of deep learning approaches, particularly BERT-based models, in handling the complexities of Indo-Aryan language transliteration. Team Vectora’s Sinhala models set a new benchmark with their exceptional performance, achieving high BLEU scores and low error rates across both standard and ad-hoc typing patterns. The comparative performance of other teams, such as Team MoraCSE’s deep learning approach and Team IndiDataMiner’s models for Hindi, further confirms the effectiveness of neural-based methods over traditional rule-based systems. While challenges persist in handling irregular typing patterns, the findings highlight the potential of deep learning models to significantly improve transliteration accuracy, particularly when tailored to the specific linguistic characteristics of each language. These results pave the way for more refined and robust transliteration systems, with Team Vectora’s approach offering valuable insights for future work in this domain.

## Ethics Statement

This paper has benefited from the use of generative AI tools, such as ChatGPT, Notebook LLM and Claude AI, to enhance its clarity and readability. These tools were employed solely for refining language and improving textual coherence without compromising the originality of the research content. The authors take full responsibility for the integrity and accuracy of the content presented in this work.

## References

- Miuru Abeysiriwardana and Deshan Sumanathilaka. 2024. A survey on lexical ambiguity detection and word sense disambiguation. *arXiv preprint arXiv:2403.16129*.
- Mohamed Al-Badrashiny, Ramy Eskander, Nizar Habash, and Owen Rambow. 2014. Automatic transliteration of romanized dialectal arabic. In *Proceedings of the eighteenth conference on computational natural language learning*, pages 30–38.
- Maneesha Athukorala and Deshan Sumanathilaka. 2022. Swa Bhasha: Message-Based Singlish to Sinhala Transliteration.
- Bajiyo Baiju, Kavya Manohar, Leena G Pillai, and Elizabeth Sherly. 2024. Romanized to native malayalam

- script transliteration using an encoder-decoder framework. *arXiv preprint arXiv:2412.09957*.
- Yomal De Mel, Kasun Wickramasinghe, Nisansa De Silva, and Surangika Ranathunga. 2024. Sinhala transliteration: A comparative analysis between rule-based and seq2seq approaches. *arXiv preprint arXiv:2501.00529*.
- Nisansa De Silva. 2019. Survey on publicly available sinhala natural language processing tools and research. *arXiv preprint arXiv:1906.02358*.
- Thomas Deselaers, Saša Hasan, Oliver Bender, and Hermann Ney. 2009. A deep learning approach to machine transliteration. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 233–241.
- Sachithya Dharmasiri and T.G.D.K. Sumanathilaka. 2024. *Swā Bhāṣa 2.0: Addressing Ambiguities in Romanized Sinhala to Native Sinhala Transliteration Using Neural Machine Translation*. In *2024 4th International Conference on Advanced Research in Computing (ICARC)*, pages 241–246, Belihuloya, Sri Lanka. IEEE.
- Sankalp KJ, Vinija Jain, Sreyoshi Bhaduri, Tamoghna Roy, and Aman Chadha. 2024. Decoding the diversity: A review of the indic ai research landscape. *arXiv preprint arXiv:2406.09559*.
- Saurabh Kumar, Dhruvkumar Babubhai Kakadiya, and Sanasam Ranbir Singh. 2025. Team indidataminer at indonlp 2025: Hindi back transliteration-roman to devanagari using llama. In *Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages*, pages 129–134.
- 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.
- W.M.P. Liwera and L. Ranathunga. 2020. Combination of Trigram and Rule-based Model for Singlish to Sinhala Transliteration by Focusing Social Media Text. In *2020 From Innovation to Impact (FITI)*, pages 1–5, Colombo, Sri Lanka. IEEE.
- Yash Madhani, Sushane Parthan, Priyanka Bedekar, Gokul Nc, Ruchi Khapra, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M 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.
- Sabina Mammadzada. 2023. A review of existing transliteration approaches and methods. *International Journal of Multilingualism*, 20(3):1052–1066.
- Colin P Masica. 1993. *The indo-aryan languages*. Cambridge University Press.
- Abir Masmoudi, Mariem Ellouze Khmekhem, Mourad Khrouf, and Lamia Hadrich Belguith. 2019. Transliteration of arabizi into arabic script for tunisian dialect. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 19(2):1–21.
- Anuja Dilrukshi Herath Herath Mudiyanselage and TG Deshan K Sumanathilaka. 2024. Tamzi: Short-hand romanized tamil to tamil reverse transliteration using novel hybrid approach. *The International Journal on Advances in ICT for Emerging Regions*, 17(1).
- Rushan Nanayakkara, Thilini Nadungodage, and Randil Pushpananda. 2022. *Context Aware Back-Transliteration from English to Sinhala*. In *2022 22nd International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 051–056, Colombo, Sri Lanka. IEEE.
- Sandun Sameera Perera, Lahiru Prabhath Jayakodi, Deshan Koshala Sumanathilaka, and Isuri Anuradha. 2025. Indonlp 2025 shared task: Romanized sinhala to sinhala reverse transliteration using bert. In *Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages*, pages 135–140.
- Pravin H Rathod, Manikrao L Dhore, and RM Dhore. 2013. Hindi and marathi to english machine transliteration using svm. *International Journal on Natural Language Computing*, 2(4):55–71.
- Brian Roark, Lawrence Wolf-Sonkin, Christo Kirov, Sabrina J Mielke, Cibu Johny, Isin Demirsahin, and Keith Hall. 2020. Processing south asian languages written in the latin script: the dakshina dataset. *arXiv preprint arXiv:2007.01176*.
- Anshuman Sharma, Ayushi Verma, Chetan Shahra, and S. Indu. 2023. *Ancient Indian Script Transliteration Using GANs*. In *2023 10th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 621–626, Noida, India. IEEE.
- Deshan Sumanathilaka, Nicholas Micallef, and Ruwan Weerasinghe. 2024. *Swā-Bhāṣa Dataset: Romanized Sinhala to Sinhala Adhoc Transliteration Corpus*. In *2024 4th International Conference on Advanced Research in Computing (ICARC)*, pages 189–194, Belihuloya, Sri Lanka. IEEE.
- T.G.D.K. Sumanathilaka, Ruwan Weerasinghe, and Y.H.P.P. Priyadarshana. 2023. *Swā-Bhāṣa: Romanized Sinhala to Sinhala Reverse Transliteration using a Hybrid Approach*. In *2023 3rd International Conference on Advanced Research in Computing (ICARC)*, pages 136–141, Belihuloya, Sri Lanka. IEEE.
- Kuwali Talukdar and Shikhar Kumar Sarma. 2023. Parts of speech taggers for indo aryan languages: A critical review of approaches and performances. In *2023 4th International Conference on Computing and Communication Systems (I3CS)*, pages 1–6. IEEE.

Pasindu Tennage, Achini Herath, Malith Thilakarathne, Prabath Sandaruwan, and Surangika Ranathunga. 2018. [Transliteration and Byte Pair Encoding to Improve Tamil to Sinhala Neural Machine Translation](#). In *2018 Moratuwa Engineering Research Conference (MERCon)*, pages 390–395, Moratuwa. IEEE.

A.J. Vidanaralage, A.U. Illangakoon, S.Y. Sumanaweera, C. Pavithra, and S. Thelijjagoda. 2018. [Sinhala Language Decoder](#). In *2018 National Information Technology Conference (NITC)*, pages 1–5, Colombo. IEEE.

Mahima Yadav, Ishan Kumar, and Ayush Kumar. 2023. Different models of transliteration-a comprehensive review. In *2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA)*, pages 356–363. IEEE.

Reihaneh Zohrabi, Mostafa Masumi, Omid Ghahroodi, Parham AbedAzad, Hamid Beigy, Mohammad Hossein Rohban, and Ehsaneddin Asgari. 2023. Borderless azerbaijani processing: Linguistic resources and a transformer-based approach for azerbaijani transliteration. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 175–183.