IndoNLP 2025 Shared Task: Romanized Sinhala to Sinhala Reverse Transliteration Using BERT

Sameera Perera[1], Lahiru Prabhath[2], T.G.D.K. Sumanathilaka[3], Isuri Anuradha[4]

[1]Informatics Institute of Technology, Colombo 006, Sri Lanka
[2]Techlabs Global (PVT) LTD, 9-B Horton Place, Colombo 007, Sri Lanka
[3]Swansea University, Wales, UK
[4]Lancaster University, UK

Correspondence: sameeraperera827@gmail.com

Abstract

The Romanized text has become popular with the growth of digital communication platforms, largely due to the familiarity with English keyboards. In Sri Lanka, Romanized Sinhala, commonly referred to as "Singlish" is widely used in digital communications. This paper introduces a novel context-aware back-transliteration system designed to address the ad-hoc typing patterns and lexical ambiguity inherent in Singlish. The proposed system combines dictionary-based mapping for Singlish words, a rule-based transliteration for out-of-vocabulary words and a BERT-based language model for addressing lexical ambiguities. Evaluation results demonstrate the robustness of the proposed approach, achieving high BLEU scores along with low Word Error Rate (WER) and Character Error Rate (CER) across test datasets. This study provides an effective solution for Romanized Sinhala back-transliteration and establishes the foundation for improving NLP tools for similar low-resourced languages.

1 Introduction

The rapid growth of digital communication platforms such as social media and messaging platforms has revolutionized communication with the use of informal, Romanized representations of native scripts. Sinhala is a morphologically rich language where approximately 17 million Sri Lankans (around 87% of the total population) use it as their main language for communication(De Silva, 2019). Many Sinhala speakers use Romanized Sinhala, often referred to as "Singlish", instead of the native script on digital communication platforms due to the convenience of using English keyboards. However, Singlish is non-standardized, leading to variations in spelling and structure, which pose challenges for back-transliteration. The process of back-transliteration into native script has become crucial for NLP applications such as machine translation, information retrieval and sentiment analysis. However, the following challenges make this task complex:

- Ad-hoc Nature: Singlish text often follows informal typing patterns such as vowel omissions, further complicating back-transliteration. For an instance the word "තාත්තා" can be represented as "*Thaaththaa, Thaththa, Thattha, Thatta, Tatta*".
- Lexical Ambiguity: A single Romanized form may correspond to multiple words in the native Sinhala script, depending on the context. The word *"Adaraya"* can be back transliterated to *"ආදරය, ආධාරය"*.

A system capable of handling the typing variations, ambiguity, and contextual dependencies inherent in Singlish is required to address these challenges. Back-transliteration is a greater challenge than forward-transliteration because it requires context awareness (Nanayakkara et al., 2022). This paper introduces a novel context-aware back-transliteration system for Romanized Sinhala leveraging a hybrid approach that combines:

1. Dictionary-Based Mapping: To handle common and ambiguous words using an ad-hoc transliteration dictionary.
2. Rule-Based Techniques: For out-of-vocabulary words based on Sinhala phonetic patterns.
3. Contextual Disambiguation: Using a BERT model to resolve ambiguities by analyzing sentence-level context.

Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages (IndoNLP2025), pages 161–167
January 20, 2025.

The proposed approach enables the system to handle various typing patterns in Romanized Sinhala. Experimental results demonstrate the system's effectiveness in achieving high BLEU scores, low Word Error Rates (WER) and low Character Error Rates (WER) on benchmark datasets. This work significantly contributes to the field of backward transliteration in NLP by addressing the existing challenges in back transliteration.

The following sections provide a comprehensive overview of the related works and the system's methodology, evaluate its performance on real-world datasets, and discuss its limitations.

2 Related Works

Back-transliteration of Romanized Sinhala has been the focus of several studies exploring various approaches including rule-based, statistical, and neural approaches. Below are some recent studies on Singlish backward transliteration. In 2018, the Sinhala Language Decoder by Vidanaralage et al. (2018) introduced a rule-based transliteration method as part of their work where Romanized input text is processed using transliteration and phoneme rule bases. However, the system struggles with handling lexical ambiguity and some English proper nouns because of the static nature of its rule base. These limitations have restricted its ability to handle the informal typing patterns of Romanized Sinhala. In 2019, Priyadarshani et al. (2019) proposed a statistical machine translation (SMT) approach to transliterate personal names across Sinhala, Tamil, and English. Since the personal name transliteration depends on the ethnicity of the name, they employed ethnicity-specific models, achieving BLEU scores of more than 89% for all language pairs. This was implemented with a classification followed by the Naive Bayes algorithm. The reason for selecting the SMT approach instead of a neural approach is that NMT lacks robustness in translating rare words, and it requires a large amount of parallel data to train the model to achieve better results than SMT.

In 2020, a combination of Trigram and Rule-based Models was proposed by Liwera and Ranathunga (2020). This hybrid approach integrated trigram models with rule-based methods to transliterate Romanized Sinhala. The trigram model was trained on Singlish YouTube comments and their corresponding Sinhala transliteration. A rule-based approach was used to handle situations where the tri-gram model could not predict the Sinhala transliteration of Singlish words. However, the system occasionally fails to deliver the correct transliteration of a word due to ambiguities. Silva and Ahangama (2021) proposed another rule-based approach for Romanized Sinhala backward transliteration in 2021. The accuracy of the rule-based approach was further improved by using an error correction module which compares a news corpus from popular news sites. In 2022, a context-aware back-transliteration for Romanized Sinhala presented a neural machine translation approach (an encoder-decoder model) based on Bidirectional LSTM and LSTM architectures (Nanayakkara et al., 2022). The study presented a transliteration unit approach considering the context of characters in a word. This system also failed to handle sentence-level word disambiguation as it focuses on the context of the characters present in a word.

A back transliteration system which can handle informal shorthand Romanized Sinhala was proposed by Sumanathilaka et al. (2023). A statistical trigram model combined with a rule-based approach for back transliteration and a knowledge base with Trie data structure for word suggestions was used in the work. The proposed system achieved 0.84 word-level accuracy. This proposed architecture has been further extended for Tamil by (Mudiyanselage and Sumanathilaka, 2024), showing the generalizability of the proposed model. However, lexical ambiguity correction (word sense disambiguation) and code-mixed Romanized Sinhala remain a persistent issue in these approaches. Athukorala and Sumanathilaka (2024) proposed a novel approach which combines rule-based methods and fuzzy logic to transliterate Romanized Sinhala to native script even when vowels are omitted. It introduced a new numeric coding system to use with the Singlish letters by matching the identified typing patterns. For the mapping process, they have developed a fuzzy logic-based implementation. However, the system performs at the word level and does not handle lexical ambiguities. In 2024, Dhar-

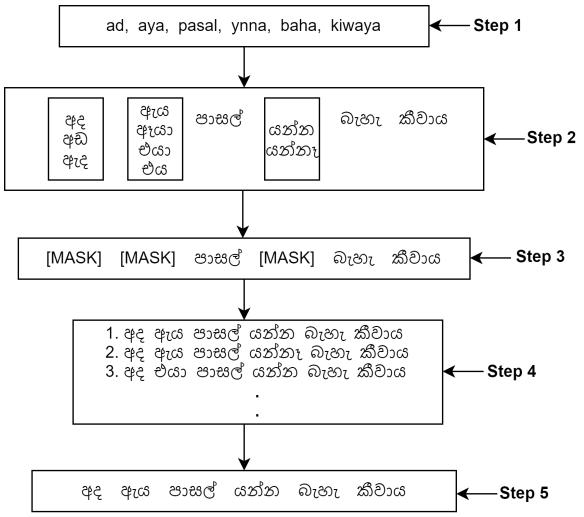

Figure 1: Transliteration Flow

masiri and Sumanathilaka (2024) proposed a GRU-based NMT model for Singlish backward transliteration. This system combined rule-based techniques with neural machine translation to address the complexities of Romanized Sinhala. A suggestion algorithm has eliminated word selection ambiguity by choosing word suggestions from a pool of predicted words. BLEU scores reaching 0.8 indicate the high word-level transliteration accuracy of the proposed model. Though many Romonized Sinhala to Sinhala transliterators have been introduced, there still exists a gap in the availability of an effective reverse transliterator, which needs context awareness to handle ambiguity.

3 Methodology

The proposed context-aware transliteration system is developed through a series of systematic steps to transliterate Romanized Sinhala text into native Sinhala script, ensuring accurate and contextually appropriate output even while dealing with lexical ambiguity and ad hoc typing patterns. The methodology consists of five key steps, as described below.

3.1 Word Separation

The first step involves breaking down the input Singlish sentence into individual words, enabling a word-level transliteration. This step facilitates word-level mapping and processing in subsequent steps.

3.2 Word-Level Mapping with Ad-hoc Transliteration Dictionary

After the input text is broken down into words, each Singlish word is mapped to its corresponding Sinhala words using an ad-hoc transliteration dictionary[1]. This dictionary includes ad-hoc Singlish words along with their corresponding Sinhala words. Because of the informal nature of Romanized Sinhala, a single Singlish word can often represent multiple Sinhala words (Sumanathilaka et al., 2024). Therefore, the dictionary provides multiple mappings for ambiguous words, retaining all possibilities to handle lexical ambiguity in the next step. If a Singlish word is not found in the transliteration dictionary, the system uses a rule-based approach to convert it into Sinhala script. This rule-based transliteration leverages predefined mappings between Romanized inputs and corresponding Sinhala characters, considering Sinhala phonetic patterns, consonant-vowel combinations, and special cases for modifiers.

3.3 Initial Sentence Assembly with Masked Tokens

After the word level translation using the dictionary and rule-based approach, the corresponding Sinhala sentence is formed by combining those transliterated Sinhala words. If any Singlish word is ambiguous (meaning it maps to multiple Sinhala words), it is replaced by a "[MASK]" token in the sentence. "[MASK]" token denotes that the correct Sinhala word is yet to be selected based on context. For each masked position, a list of candidate Sinhala words is stored, maintaining all possible interpretations of the ambiguous Romanized word. This intermediate step allows for context-aware word selection in the next step.

3.4 Context-Aware Lexical Disambiguation Using BERT

This step resolves lexical ambiguity by replacing the "[MASK]" tokens from the previous step with the most contextually appropriate words. This process involves two main substeps: candidate sentence generation and sentence scoring using BERT.In the first phase of this step, all possible sentences are generated

[1] https://www.kaggle.com/datasets/tgdeshank/wsd-romanized-sinhala-dataset?select=WSD+Romanized-Sinhala+-+Sinhala+.txt

by filling each "[MASK]" with different combinations of candidate words stored from the previous step. Then ,each generated sentence is scored using a BERT model configured for Masked Language Modeling (MLM). The goal of this scoring is to determine the most contextually appropriate sentence. Given the context, the score is calculated based on the probability of each candidate word appearing in the masked positions. To illustrate this process, let's walk through the score calculation for an example sentence in Figure 1.

sentence: "අද ඇය පාසල් යන්න බැහැ කීවාය"

Score(sentence) = P("අද"| context) × P ("ඇය"| context) × P ("යන්න"| context)

Each probability P (w | context) represents the likelihood of a candidate word appearing in its respective masked position, given the context provided by the rest of the sentence. The example of calculation for P ("අද" | context) is done as below:

- Mask the Target Word: Replace "අද" in the sentence with a [MASK] token to create a partially masked sentence: "[MASK] ඇය පාසල් යන්න බැහැ කීවාය"
- Pass the Sentence to BERT: Feed the masked sentence into the BERT model and get the generated logits for mask position. These logits represent the model's unnormalized confidence levels for each vocabulary word in the masked slot based on the sentence context.
- Apply Softmax Activation: Convert the logits into probabilities by applying the softmax activation function. Softmax normalizes the logits to create a probability distribution over all possible words for the [MASK] position.
- Retrieve the Probability for "අද": From the probability distribution, get the probability assigned to the word "අද" in the context of the sentence.
- Repeat for Remaining Masked Words: follow a similar process for "ඇය" and "යන්න" by masking each respective word in the sentence and calculating its probability in context.

3.5 Output Generation

Finally, the sentence representing the highest score from step 4 is returned as the transliterated Sinhala text. Following the example discussed above for the romanized Sinhala sentence *"ad aya pasal ynna baha kiwaya"* is transliterated to "අද ඇය පාසල් යන්න බැහැ කීවාය" as the output following the above approach.

4 Challenges and Solutions

The primary challenge of the proposed transliteration approach was the time consumption for processing long sentences containing highly ambiguous words. In the proposed transliteration approach, the major factor contributing to time consumption is the number of model inferences required for disambiguation. Two key aspects that influence the number of model inferences:

- High ambiguity words: Singlish words with high lexical ambiguity may represent multiple Sinhala words. This increases the number of candidate words for each ambiguous Singlish word. Consequently, the number of possible sentences generated in step 4 also increases, leading to an increase in the required model inferences.
- Number of ambiguous words: An increase in the number of ambiguous words in the input text also influences the number of model inferences as it directly increases the number of possible sentences generated in Step 4.

Two strategies were developed to reduce the processing time while maintaining accuracy, as described in section 4.1 and 4.2.

4.1 Reducing the Number of Candidate Words for Ambiguous Words Using a Filtering Mechanism

As the initial step of the reverse transliteration process, the candidate word generation occurs as illustrated in step 2 of Figure 1. This step used the Swa-bhasha dictionary, which contains the possible interpretation of the Sinhala word in Ad hoc Romanized Sinhala format. For highly ambiguous Singlish words, the dictionary often provides many Sinhala candidates.

To reduce the candidate list size, the vocabulary associated with the model tokenizer is considered so that any candidate words extracted from the dictionary that are not present in the tokenizer's vocabulary are removed from the candidate list.

4.2 Chunking Sentences Based on the Number of BERT Calls

A chunking mechanism is applied to sentences which contain at least three ambiguous words (masks) to reduce the number of model inferences (BERT calls). Chunking is performed while ensuring that each chunk contains at least three mask tokens. The process involves the following steps:

- Starting from the beginning of the sentence, it calculates the required number of model inferences for the first three ambiguous words (or "masks").
- If the BERT call count for the first three masks is under 20 (as our analysis showed that 20 BERT calls take approximately 1 second), the next ambiguous word is added to the chunk (adding a fourth mask) and recalculate the BERT call count for the first four masks.
- This process continues, adding one mask at a time and recalculating until the BERT call count exceeds 20.
- When the number of BERT calls exceeds 20, the words processed so far and the words up to the next mask are taken as a chunk.
- The next chunk starts with a two-mask overlap, including the last two ambiguous words (masks) from the previous chunk, and also includes the words after the third mask from the end of the previous chunk. This ensures the retention of unambiguous words in the new chunk to maintain the context.

Figure 2 illustrates the chunking process with an example: Assume the number of BERT calls required for processing the first three ambiguous words (MASK1, MASK2 and MASK3) is 15, which is below 20 (as 20 BERT calls take approximately 1 second). Therefore, the system

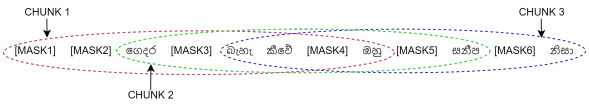

Figure 2: Transliteration Flow

includes the next ambiguous word, "MASK4", and recalculates the number of BERT calls for the first four masks (MASK1, MASK2, MASK3 and MASK4). Suppose the number of BERT calls for the first four masks is 30, which is higher than 20. As a result, the system creates the first chunk, which includes all words up to "MASK5" but excludes "MASK5" itself. The second chunk begins from the word "ගෙදර" which follows the third mask ("MASK2") from the end of the previous chunk. Then, the number of BERT calls for the first three masks (MASK3, MASK4, MASK5) of this new chunk is calculated. Assume the number of BERT calls for the first three masks of this chunk is 25, which is higher than 20. As a result, this second chunk spans from "ගෙදර" to "සනීප". Then, the third chunk starts from the word "බැහැ" which follows the third mask ("MASK3") from the end of the second chunk.

5 Result Evaluation and Discussion

For the baseline evaluation, a BERT model trained on Sinhala data sources for mask language modelling from Hugging Face (model 1[2]) was used to develop the proposed back transliteration system. Then, it was further fine-tuned using native Sinhala script data in the Dakshina dataset (Roark et al., 2020). The training hyperparameters were used during fine tuning (model 2[3]): learning-rate: 5e-05, train-batch-size: 64, eval-batch-size=16, num-epochs: 12.

The evaluation was based on the validation test sets[4] provided by the INDONLP 2025 shared task organizers [5]. The test sets 1 and 2 contained 10000 and 5000 data records, respectively. Test set 2 mainly consists of Romanized Sinhala samples in ad hoc format where vowels were omitted in its Romanized presen-

[2] https://huggingface.co/Ransaka/sinhala-bert-medium-v2

[3] https://huggingface.co/Sameera827/Sinhala-BERT-MLM

[4] https://github.com/IndoNLP-Workshop/IndoNLP-2025-Shared-Task

[5] https://indonlp-workshop.github.io/IndoNLP-Workshop/sharedTask/

Metric	Test Set 1	Test Set 2
Model 1: Sinhala BERT		
WER	0.0886	0.0914
CER	0.0200	0.0212
BLEU-1	0.9115	0.9088
BLEU-2	0.8718	0.8686
BLEU-3	0.8488	0.8452
BLEU-4	0.7963	0.7917
Model 2: Fine-tuned BERT		
WER	0.0850	0.0895
CER	0.0194	0.0210
BLEU-1	0.9151	0.9107
BLEU-2	0.8760	0.8699
BLEU-3	0.8526	0.8459
BLEU-4	0.8001	0.7916

Table 1: Evaluation Results

tation. The proposed system was evaluated using Word Error Rate (WER), Character Error Rate (CER) and BLEU scores. WER and CER measure the percentage of word-level errors and character-level errors, respectively. BLEU scores assess the similarity between the output of the system and the reference text, considering both precision and fluency across n-grams. Higher BLEU scores and Lower WER and CER values indicate better performance. The obtained results were compared between the two BERT models (Model 1 and Model 2) as shown in Table 1. According to the results, the fine-tuned model showed better results overall, but Model 1 was only 0.0001 higher in the BLEU-4 score. Overall, the results demonstrate that the model performs well in handling both ad-hoc transliteration scenarios (without vowels) and normal scenarios (with vowels) for the back-transliteration of Romanized Sinhala.

6 Conclusion

The proposed context-aware back-transliteration approach effectively converts Romanized Sinhala text into native Sinhala script, addressing the challenges of ad-hoc typing patterns and lexical ambiguity inherent in Romanized Sinhala back-transliteration. Evaluation results demonstrate the robustness of the proposed approach, achieving high BLEU scores along with low Word Error Rate (WER) and Character Error Rate (CER) across test datasets. The codebase can be accessed through the link below for further research in this area. GitHub link: https://github.com/Sameera2001Perera/Singlish-Transliterator

Limitations

While the proposed back-transliteration approach demonstrates significant accuracy, it has several limitations. As described earlier, the system can take time to transliterate long sentences containing highly ambiguous words. Although candidate word reduction and chunking mechanisms somewhat mitigate this issue, real-time applications may still face challenges in maintaining efficiency. The word-level transliteration relies on an ad-hoc Romanized Sinhala–Sinhala dictionary. If a Singlish word is not found in the transliteration dictionary, those words are handled using a rule-based approach. However, this rule-based method is not designed to handle ad-hoc typing patterns.

References

Maneesha U. Athukorala and Deshan K. Sumanathilaka. 2024. Swa Bhasha: Message-Based Singlish to Sinhala Transliteration. *arXiv preprint*. Version Number: 1.

Nisansa De Silva. 2019. Survey on publicly available sinhala natural language processing tools and research. *arXiv preprint arXiv:1906.02358*.

Sachithya Dharmasiri and T.G.D.K. Sumanathilaka. 2024. Swa Bhasha 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.

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.

Anuja Dilrukshi Herath Herath Mudiyanselage and TG Deshan K Sumanathilaka. 2024. Tam : Shorthand 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.

H.S. Priyadarshani, M.D.W. Rajapaksha, M.M.S.P. Ranasinghe, K. Sarveswaran, and G.V. Dias. 2019. Statistical Machine Learning for Transliteration: Transliterating names between Sinhala, Tamil and English. In *2019 International Conference on Asian Language Processing (IALP)*, pages 244–249.

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*. Version Number: 1.

Lahiru de Silva and Supunmali Ahangama. 2021. Singlish to Sinhala Transliteration using Rule-based Approach. In *2021 IEEE 16th International Conference on Industrial and Information Systems (ICIIS)*, pages 162–167, Kandy, Sri Lanka. IEEE.

Deshan Sumanathilaka, Nicholas Micallef, and Ruvan Weerasinghe. 2024. Swa-Bhasha 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, Ruvan Weerasinghe, and Y.H.P.P. Priyadarshana. 2023. Swa-Bhasha: 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.

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.