Sinhala Transliteration: A Comparative Analysis Between Rule-based and Seq2Seq Approaches

Yomal De Mel[*], Kasun Wickramasinghe*, Nisansa de Silva
Department of Computer Science & Engineering
University of Moratuwa, Katubedda 10400, Sri Lanka
{mario.23,kasunw.22,NisansaDdS}@cse.mrt.ac.lk
Surangika Ranathunga
School of Mathematical and Computational Sciences,
Massey University, Auckland, New Zealand
s.ranathunga@massey.ac.nz

Abstract

Due to reasons of convenience and lack of tech literacy, transliteration (i.e., Romanizing native scripts instead of using localization tools) is eminently prevalent in the context of low-resource languages such as Sinhala, which have their own writing script. In this study, our focus is on Romanized Sinhala transliteration. We propose two methods to address this problem: Our baseline is a rule-based method, which is then compared against our second method where we approach the transliteration problem as a sequence-to-sequence task akin to the established Neural Machine Translation (NMT) task. For the latter, we propose a Transformer-based Encode-Decoder solution. We witnessed that the Transformer-based method could grab many ad-hoc patterns within the Romanized scripts compared to the rule-based method. The code base associated with this paper is available on GitHub - https://github.com/kasunw22/Sinhala-Transliterator/

1 Introduction

Sinhala Language, spoken by over 16 million people in Sri Lanka, presents unique challenges for computational processing due to its distinct script and structure (De Silva, 2019). In modern-day digital communication, it is common to use *Singlish*[1], where Sinhala (Sinhalese) words are written with Latin (English) script (Liwera and Ranathunga, 2020). While the widespread use of Singlish in informal communication calls for efficient transliteration systems capable of accurately converting it into the Sinhala script, this task is made difficult by code-mixed and code-switched usage of Singlish scripts (Rathnayake et al., 2022; Udawatta et al., 2024). Further, ad-hoc approximations are used by users when they approximate the *Abugida* Sinhala script (Liyanage et al., 2012) using the Latin script which is an *Alphabet* (Pulgram, 1951). Yet, we do not find sufficient transliteration research done for Singlish.

As for many NLP tasks, the early solutions for transliteration were based on rule-based techniques that relied on predefined character mappings (Santaholma, 2007). However, they often struggled when confronted with the variability in the format in which Sinhala words were written using English script (Liwera and Ranathunga, 2020). In contrast, deep learning models, especially Transformer-based architectures (Vaswani, 2017), have proved to perform well for the transliteration task (Moran and Lignos, 2020). However, such deep learning methods have not been used to implement Transliteration systems related to Sinhala.

This paper introduces two distinct methods, a rule-based approach and a deep learning-based approach to solve the Singlish to Sinhala transliteration problem. The deep learning based transliteration system is implemented on a pre-trained sequence-to-sequence multilingual language model, akin to a Machine Translation task. Subsequently, we evaluate their effectiveness and limitations. According to our results, we observed that the deep learning approach is more robust to language variability compared to the rule-based approach. The rest of the sections will discuss the related work, our methodology, the results we obtained, and the Conclusions.

[*] Equal contribution

[1] Not to be confused with English-based creole used in Singapore with the same name.

Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages (IndoNLP2025), pages 168–176
January 20, 2025.

2 Related work

Machine transliteration focuses on converting text from one script to another using phonetic or spelling equivalents, ideally mapping words or letters systematically between writing systems (Kaur and Singh, 2014).

2.1 Rule-based Transliteration

Rule-based machine transliteration relies on predefined grammar rules, a lexicon, and processing software. It uses morphological, syntactic, and semantic information from source and target languages, with human experts designing rules to guide transliteration. These rules ensure the input structure and meaning are accurately mapped to the target language, preserving integrity and context in the transliterated output (Kaur and Singh, 2014; Athukorala and Sumanathilaka, 2024). It includes methods such as Direct Machine Translations (MT), Transfer-based MT, and Interlingual MT (Sumanathilaka, 2023). Although effective, rule-based machine transliteration is known for being time-consuming and complex because it requires creating detailed linguistic rules to transliterate sentences from the source language to the target language (Sumanathilaka, 2023).

Tennage et al. (2018) introduced the first transliteration system for Sinhala to English. This transliteration tool utilized character mapping tables to convert words from the native scripts of both languages into a common phonetic representation in English. The authors report that the transliteration approach allows for better preservation of word ordering and more accurate transliteration of phrases. Their system shows a good accuracy for handling of loanwords—where both languages share similar transliterated forms—and also enhances the overall translation quality by allowing for better mapping of linguistic structures, thus addressing the challenges posed by the morphological richness of both languages.

Hybrid transliteration systems that combine rule-based methods with a trigram model have shown to improve the accuracy of converting Singlish to Sinhala (Liwera and Ranathunga, 2020). The rule-based component applies predefined rules for vowels and consonants, while the trigram model uses statistical patterns from social media comments to address the variability and ambiguity of Singlish input.

2.2 Transformers for multilingual Sequence-to-Sequence Generation Tasks

For sequence-to-sequence (Seq2Seq) generation tasks such as Machine Translation (MT), the proven architecture is the Encoder-Decoder architecture. When it comes to multilingual Transformer-based pre-trained Encoder-Decoder architectures, mT5 (Xue et al., 2021) which is based on T5 (Raffel et al., 2020), mBART (Liu, 2020) which is based on BART (Lewis, 2019), M2M100 (Fan et al., 2020), MarianNMT (Tambouratzis, 2021) have been popular choices. The advantage of the Transformer-based Encoder-Decoder architecture is that due to its self-attention and cross-attention mechanisms, the relationships with and among the source and the target sequence are properly captured (Vaswani, 2017). Seq2Seq, has since been utilized in domains other than MT (de Almeida et al., 2020).

2.3 Translation Models with Sinhala Language Support

There are several free and open-source multilingual translation models that include Sinhala. Among them mT5, mBART, M2M100, MarianNMT, and NLLB[2](Costa-jussà et al., 2022) are prominent. Both M2M100 and NLLB use the same model architecture but two different training datasets. M2M100 uses CCMatrix (Schwenk et al., 2021) and CCAlighned (El-Kishky et al., 2019) datasets while NLLB uses the NLLB (Costa-jussà et al., 2022) dataset. On the other hand, MarianNMT model uses a different encoder-decoder architecture, and the dataset they use is OPUS-100 (Zhang et al., 2020). Both mBART and mT5 have been used for various Sinhala text generation tasks, including Machine Translation (Niyarepola et al., 2022; Ranathunga et al., 2024b; Thillainathan et al., 2021; Lee et al., 2022). However, according to a recent study by Ranathunga et al. (2024a), NLLB has proven to be the best among them for translation tasks that involve Sinhala.

[2] https://github.com/facebookresearch/fairseq/tree/nllb?tab=readme-ov-file

2.4 Deep Learning based Transliteration

Deselaers et al. (2009) proposed a deep belief system-based transliteration solution using Deep Belief Networks (DBN). DBN architecture is almost similar to the encoder-decoder architecture. Deselaers et al. (2009) mentioned that transliteration can be considered a translation task at the character level. Subsequent neural network-based (NN) solutions for the transliteration task mainly relied on recurrent models such as simple RNN, LSTM, and GRU (Shao and Nivre, 2016; Mahdi Mahsuli and Safabakhsh, 2017; Rosca and Breuel, 2016; Kundu et al., 2018). Zohrabi et al. (2023) have used a Transformer-based approach for the transliteration of Azerbaijani. A comparative evaluation of LSTM, biLSTM, GRU, and Transformer architectures for named entity transliteration has been carried out by Moran and Lignos (2020). According to their evaluation, Transformer-based encoder-decoder architectures outperform other architectures.

3 Methodology

3.1 Rule-Based Transliteration System

Our rule-based approach uses predefined linguistic rules to map Latin script (Singlish) to Sinhala script. These rules cover vowels, consonants, diacritics, and special characters. It extends the rule-based transliteration system of Tennage et al. (2018) with a few additions to the mapping rules when considering two and three-character mapping. Some of the rules defined are shown in Table 1, where newly added rules are highlighted. The process involves two primary stages: rule definition and application.

The transliteration function processes each input word and converts it to Sinhala using a character-by-character matching strategy, as detailed below. The pseudocode is shown in Algorithm 1.

- **Input Processing:** The system reads the input word in Latin script and ensures it contains only Latin characters.
- **Longest Match Strategy:** For each character sequence, the system matches the longest possible substring (up to three characters). This ensures that multi-character sequences such as “th” or “aa” are mapped correctly before shorter, single-character matches.
- **Rule Application:** If a match is found in the transliteration table, the corresponding Sinhala character is appended to the result. If no match is found, the character is added as is.
- **Output Generation:** The transliterated word is returned and added to the output dataset.

Latin Sequence	Sinhala Character	Latin Sequence	Sinhala Character
a	අ	aa	ආ
A	ඇ	Aa	ඈ
i	ඉ	ie	ඊ
u	උ	uu	ඌ
e	එ	ea	ඒ
I	ඓ	o	ඔ
ka	ක	ga	ග
ma	ම	ya	ය
ra	ර	ba	බ
ca	ච	ja	ජ
ta	ට	la	ල
Da	ඩ	wa	ව
tha	ත	sa	ස
da	ද	ha	හ
na	න	pa	ප
Na	ණ	La	ළ
mi	මි	thi	ති
Ka	ඛ	Ga	ඝ
cha	ඡ	Tha	ඨ
Dha	ඪ	dha	ධ
Pa	ඵ	bha	භ
fa	ෆ	Ba	ඹ
GNa	ඥ	KNa	ඤ
jha	ඣ	Lu	ඏ
Luu	ඐ	Sa	ශ
sha	ෂ	GNa	ඥ
ki	කි	ku	කු
ke	කෙ	ko	කො
kaa	කා	kAa	කෑ
kie	කී	kei	කේ
gi	ගි	gu	ගු
ge	ගෙ	go	ගො
gaa	ගා	gAa	ගෑ
gie	ගී	gei	ගේ
goe	ගෝ	guu	ගූ
gau	ගෞ	\n	◌ං

Table 1: Transliteration rules. The highlighted rules were added by us.

Algorithm 1 Transliteration Algorithm

```
Require: Latin script word word
Ensure: Sinhala script word
result ← “”                        ▷ Initialize an empty string
i ← 0                              ▷ Initialize index
while i < length(word) do
    matched ← False
    for length in {3, 2, 1} do     ▷ Check substrings of decreasing length
        substring ← word[i:i + length]
        if substring in transliteration_table then
            result ← result + transliteration_table[substring]
            i ← i + length
            matched ← True
            break
        end if
    end for
    if not matched then
        result ← result + word[i]
        i ← i + 1
    end if
end while
return result
```

3.2 Deep Learning-Based Transliteration System

In this approach, we model transliteration as a translation task, as suggested by Deselaers et al. (2009). Even though decoder-only Large Language Models (LLMs) are the state-of-the-art choice for most of the NLP tasks including Machine Translation nowadays, for many *low-resource language* translation tasks, still sequence-to-sequence modes are commonly used (Ranathunga et al., 2023). Considering these factors, a Transformer-based encoder-decoder model is our second approach to solving the reverse transliteration problem.

Apart from the context-based generation, another advantage of this approach is that unlike in rule-based approaches, we do not need to manually define the rules and we only need to find or create a rich dataset that covers the possible scenarios that could occur during the inference time. Moreover, the code-mixed and code-switched cases can also be easily addressed in this approach simply by extending the training dataset accordingly.

To have better accuracy, rather than training the model from scratch, we used an existing multilingual pre-trained sequence-to-sequence model that is trained for the translation task, which has coverage for Sinhala as well. To be specific, we have selected the 418M version of the M2M100 model[3] (Fan et al., 2020) as our base model and fine-tuned it for Romanized-Sinhala and Sinhala as a translation pair. We used the existing English language code (i.e. *en*) for Romanized Sinhala and the Sinhala language code (i.e. *si*) for Sinhala. The reason for selecting M2M100 is that the MarianMT translation quality for the Sinhala-English pair is a bit worse than M2M100 and NLLB models (see Table 2). Both NLLB and M2M100 use the same model architectures and the translation qualities are almost similar (Table 2). We choose M2M100 over NLLB since NLLB model weights are bound with some additional restricted terms and conditions[4] while M2M100 weights are not[5].

We fine-tuned M2M100 model in a way that the Romanized script is considered as the English translation of the corresponding Sinhala script. We used the M2M100 model's tokenizer[3] for the tokenization process. Since the model already knows the basic linguistics from the translation task, it only needs to learn the relationship between the two new language pairs. Also in Romanized typing, it is more common to use code-mixed usage within the content. Furthermore, since we are using a Transformer-based model, the context is also taken into account when the transliteration is done.

4 Implementation

4.1 Dataset Preparation

The task is a sequence-to-sequence text generation task, specifically developing a reverse transliterator that converts Romanized Indo-Aryan languages to their native scripts. Therefore what we need is a parallel dataset that contains Romanized text and the corresponding native script.

[3] https://huggingface.co/facebook/m2m100_418M

[4] https://github.com/facebookresearch/fairseq/blob/nllb/LICENSE.model.md

[5] https://choosealicense.com/licenses/mit/

English Input	Marian-MT Translation	M2M100 Translation	NLLB Translation
How do you know that this is correct?	ඔයා කොහොමද දන්නෙ මේක හරි කියලා?	මේ දේ නිවැරදි බව ඔබ කොහොමද දන්නේ?	ඔයා කොහොමද දන්නේ මේක හරි කියලා?
It is the way he played that matters not the amount of time he spent.	ඔහු කාලය ගත කළේ කාලය අවශ්‍ය නැහැ.	ඔහු ක්‍රිඩා කරන ආකාරය ඔහු ගත කරන කාලය කොතරම් වැදගත් නොවේ.	ඔහු සෙල්ලම් කරන විදිහ තමයි වැදගත් වෙන්නේ. ඔහු ගතකරපු කාලය නෙවෙයි.
It’s a great pleasure to meet you	ඔයාව හම්බ වෙන්න පුළුවන් වෙලා තියෙන්නේ	ඔබව හමුවීම සතුටක්	ඔයාව මුණගැහෙන්න ලැබීම සතුටක්
Nothing is impossible until you give up it	ඔයා ඒක අතහරින්න මුකුත් බැරි වෙලාවක් නෑ	ඔබ එය අතහැරීමට පෙර කිසිවක් අසාථර්ක නොවේ	ඔයා ඒක අතහරිනකම් කරන්න බැරි දෙයක් නෑ.
It is neither beautiful nor strong	ඒක ලස්සනයි නමුත් ශක්තිමත් නෙමෙයි	එය ලස්සන හෝ ශක්තිමත් නොවේ.	ඒක ලස්සනවත් ශක්තිමත්වත් නෑ.

Table 2: Qualitative evaluation of translation models. Records shaded in light gray indicate the translations are slightly incorrect and the dark gray shaded ones are really bad translations. Non-shaded ones are correct translations.

In order to create the training dataset, we used the Dakshina (Roark et al., 2020) and Swa-Bhasha (Sumanathilaka et al., 2023, 2024) datasets. We further augmented the datasets by adding some ad-hoc nature to the Romanized scripts by removing vowels and applying different common typing patterns. See Table 3 for examples. We created a dataset consisting of 10k parallel data points using these data sources. We split that into a training set of 9k data points and a validation set of 1k data points for the model training and validation.

We have evaluated our two approaches on the test sets[6] provided by the shared task on ”Reverse Transliteration on Romanized Indo-Aryan languages using ad-hoc transliterals”, organized by the IndoNLP workshop with COLING 2025. Test set 1 consists of 10,000 parallel entries containing general Romanized typing patterns and, test set 2 consists of 5000 parallel entries with ad-hoc Romanized typing patterns that come across in practical scenarios making it very challenging to solve the reverse transliteration task. The original datasets were not well structured. Therefore we converted these datasets into CSV format, containing Romanized Sinhala script (Singlish) sentences in one column and the corresponding expected Sinhala script in another.

Sinhala Script	Original Romanized Script	Augmented Alternative Romanized Scripts
ඔයා රැට කැවද ?	Oya rata kawada ?	Oya reta kewada ? Oya rata kawd ? Oya reta kewd ? Oy rat kawd ? Oy ret kewd ?

Table 3: Data augmentation example

4.2 Computational Resources

We used an NVIDIA Tesla T4 16GB GPU for the training process. The important training hyper-parameters have been listed in Table 4.

Hyperparameter	Value
learning rate	2e-5
epochs	3
train batch size	8
gradient accumulation steps	1
effective training batch size	8
training precision	fp16
weight decay	0.01
optimizer	Adam
learning rate scheduler	linear
training dataset	9000
evaluation dataset	1000

Table 4: Training hyper-parameters of the deep learning model

4.3 Evaluation Metrics

To assess the accuracy of the transliteration, we use three key metrics:

- **Word Error Rate (WER):** Measures the difference between the predicted and reference sentences at the word level. The lower the WER the better.
- **Character Error Rate (CER):** Evaluates character-level accuracy by calculating the number of edits needed to convert the predicted output to the reference. The lower the CER the better.
- **BLEU Score:** Assesses the overlap between predicted and reference outputs. The higher the BLEU score the better.

We used the metric implementations of Python evaluate[7] library for our evaluation.

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

Approach	Evaluation Matrix	Average Result for Test Set 01	Average Result for Test Set 02
Rule-based	WER	0.6689	0.6809
	CER	0.2119	0.2202
	BLEU	0.0177	0.0163
DL-based	WER	**0.1983**	**0.2413**
	CER	**0.0579**	**0.0789**
	BLEU	**0.5268**	**0.4384**

Table 5: Results for rule-based and deep learning based techniques

5 Results and Discussion

Table 5 shows the evaluation metrics for rule-based and deep learning-based approaches evaluated on the provided two test sets. As can be seen in Table 6, the deep learning approach is more robust to the ad-hoc variations of Romanized typing compared to the rule-based approach.

Romanized Script	**Rule-based Result**	**DL-based Result**
kmk nehe modyi wge	ක්මක් නෙහෙ මොදයි වගෙ	කමක් නැහැ මෝඩයි වගේ
mta ehema denila ne eth eya uda thttuwe innkota klin ehema denila ne	මට එහෙම දෙනිල නෙ එත එය උඩ තටටුවේ ඉන්න්කොට ක්ලින් එහෙම දෙනිල නෙ	මට එහෙම දැනිලා නෑ ඒත් එයා උඩ තට්ටුවේ ඉන්නකොට කලින් එහෙම දැනිලා නෑ
eka nrkyi oya dnnwa mma adahas krna de	එක න්රක්යි ඔය දන්න්ව මම අඩහස ක්රන දෙ	ඒක නරකයි ඔයා දන්නවා මම අදහස් කරන දේ

Table 6: Robustness comparison of two approaches

Nevertheless, the efficiency concerned, the rule-based approach is much faster than the deep learning approach. In the CPU, the deep learning approach becomes extremely slow making it hard to use for real-time applications. In contrast on a GPU, we can achieve real-time performance for the deep learning approach as well. Check Table 7 for the results related to computing efficiency. We used output *tokens per second* (TPS) as the performance measure. According to Table 7, we can expect better performance values for the deep learning approach with lower precision setups (i.e. fp16, INT8, INT4, etc.) possibly with a slight accuracy compromisation.

Rule-based	deep learning		
	CPU (fp32)	GPU (fp32)	GPU (fp16)
>200,000	~3	~35	~65

Table 7: Speed (in TPS) comparison of the two approaches.

6 Conclusion

We have experimented with two approaches for the transliteration task for Romanized Sinhala and English. The first approach is a rule-based statistical approach. The second approach addresses the transliteration task as a translation task using a pre-trained multilingual encoder-decoder language model. Both approaches have their own pros and cons. When it comes to accuracy, the deep learning approach outperformed the rule-based method while in terms of efficiency, it is the other way around.

Limitations

The deep learning-based approach does come with a compromise of efficiency to the accuracy. The quality of the output of the deep learning approach heavily depends on the quality of the training data.

The rule-based transliteration system for converting Latin script to Sinhala faces several key challenges. A primary limitation is ambiguity handling: certain Latin character sequences can map to multiple Sinhala characters depending on context. Without contextual awareness, the system processes each character sequence independently, leading to inaccuracies, especially with complex or compound words where pronunciation depends on neighboring syllables. Additionally, users often spell the same word differently based on their typing preferences or ease. For instance, the Romanized term "mama" could correspond to different Sinhala words such as මම \mə'mɜ\ (Nominative I), මාම \mɑː'mɜ\ (Accusative specifically me), or මාමා \mɑː'mɑː\ (Nominative uncle). This inconsistency introduces ambiguity, making it difficult to define rigid transliteration rules. In contrast, deep learning models can better handle such variations by learning context and patterns from large datasets, offering more flexibility and accuracy.

Additionally, the predefined rules may not cover all linguistic nuances, resulting in errors when encountering words that deviate from standard structures. Morphological complexities, such as inflections or compound words, further challenge the system, as it does not account for grammatical context.

We have used a training set of 9k parallel en-

[7] https://huggingface.co/docs/evaluate/v0.4.0/en/index

tries for the deep-learning model fine-tuning. Having an extended training set covering more practical cases could lead to better results.

As future work, we plan to address these limitations and also experiment with LLMs for the transliteration task.

References

Maneesha U Athukorala and Deshan K Sumanathilaka. 2024. Swa bhasha: Message-based singlish to sinhala transliteration. arXiv preprint arXiv:2404.13350.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. arXiv preprint arXiv:2207.04672.

Melonie de Almeida, Chamodi Samarawickrama, Nisansa de Silva, Gathika Ratnayaka, and Amal Shehan Perera. 2020. Legal Party Extraction from Legal Opinion Text with Sequence to Sequence Learning. In 2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer), pages 143--148. IEEE.

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, Athens, Greece. Association for Computational Linguistics.

Ahmed El-Kishky, Vishrav Chaudhary, Francisco Guzman, and Philipp Koehn. 2019. A massive collection of cross-lingual web-document pairs. arXiv preprint arXiv:1911.06154.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation. arXiv preprint.

Kamaljeet Kaur and Parminder Singh. 2014. Review of machine transliteration techniques. International Journal of Computer Applications, 107(20).

Soumyadeep Kundu, Sayantan Paul, and Santanu Pal. 2018. A deep learning based approach to transliteration. In Proceedings of the Seventh Named Entities Workshop, pages 79--83, Melbourne, Australia. Association for Computational Linguistics.

En-Shiun Lee, Sarubi Thillainathan, Shravan Nayak, Surangika Ranathunga, David Adelani, Ruisi Su, and Arya D McCarthy. 2022. Pre-trained multilingual sequence-to-sequence models: A hope for low-resource language translation? In Findings of the Association for Computational Linguistics: ACL 2022, pages 58--67.

M Lewis. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

Y Liu. 2020. Multilingual denoising pre-training for neural machine translation. arXiv preprint arXiv:2001.08210.

WMP 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), volume 1, pages 1--5. IEEE.

Chamila Liyanage, Randil Pushpananda, Dulip Lakmal Herath, and Ruvan Weerasinghe. 2012. A computational grammar of sinhala. In Computational Linguistics and Intelligent Text Processing: 13th International Conference, CICLing 2012, New Delhi, India, March 11-17, 2012, Proceedings, Part I 13, pages 188--200. Springer.

Mohammad Mahdi Mahsuli and Reza Safabakhsh. 2017. English to persian transliteration using attention-based approach in deep learning. In 2017 Iranian Conference on Electrical Engineering (ICEE), pages 174--178.

Molly Moran and Constantine Lignos. 2020. Effective architectures for low resource multilingual named entity transliteration. In Proceedings of the 3rd Workshop on Technologies for MT of Low Resource Languages, pages 79--86, Suzhou, China. Association for Computational Linguistics.

Kashyapa Niyarepola, Dineth Athapaththu, Savindu Ekanayake, and Surangika Ranathunga. 2022. Math word problem generation with multilingual language models. In Proceedings of the 15th International Conference on Natural Language Generation, pages 144--155.

Ernst Pulgram. 1951. Phoneme and grapheme: A parallel. Word, 7(1):15--20.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140):1--67.

Surangika Ranathunga, Nisansa De Silva, Velayuthan Menan, Aloka Fernando, and Charitha Rathnayake. 2024a. Quality does matter: A detailed look at the quality and utility of web-mined parallel corpora. In Proceedings of the 18th Conference of the European

Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 860--880, St. Julian's, Malta. Association for Computational Linguistics.

Surangika Ranathunga, En-Shiun Annie Lee, Marjana Prifti Skenduli, Ravi Shekhar, Mehreen Alam, and Rishemjit Kaur. 2023. Neural machine translation for low-resource languages: A survey. ACM Computing Surveys, 55(11):1--37.

Surangika Ranathunga, Rumesh Sirithunga, Himashi Rathnayake, Lahiru De Silva, Thamindu Aluthwala, Saman Peramuna, and Ravi Shekhar. 2024b. Sitse: Sinhala text simplification dataset and evaluation. arXiv preprint arXiv:2412.01293.

Himashi Rathnayake, Janani Sumanapala, Raveesha Rukshani, and Surangika Ranathunga. 2022. Adapter-based fine-tuning of pre-trained multilingual language models for code-mixed and code-switched text classification. Knowledge and Information Systems, 64(7):1937--1966.

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.

Mihaela Rosca and Thomas Breuel. 2016. Sequence-to-sequence neural network models for transliteration. arXiv preprint arXiv:1610.09565.

Marianne Santaholma. 2007. Grammar sharing techniques for rule-based multilingual NLP systems. In Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007), pages 253--260, Tartu, Estonia. University of Tartu, Estonia.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, Armand Joulin, and Angela Fan. 2021. CCMatrix: Mining billions of high-quality parallel sentences on the web. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 6490--6500, Online. Association for Computational Linguistics.

Yan Shao and Joakim Nivre. 2016. Applying neural networks to English-Chinese named entity transliteration. In Proceedings of the Sixth Named Entity Workshop, pages 73--77, Berlin, Germany. Association for Computational Linguistics.

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. IEEE.

TGDK Sumanathilaka. 2023. Romanized sinhala to sinhala reverse transliteration using a hybrid approach. Ph.D. thesis.

TGDK Sumanathilaka, Ruvan Weerasinghe, and YHPP 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. IEEE.

George Tambouratzis. 2021. Alignment verification to improve NMT translation towards highly inflectional languages with limited resources. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 1841--1851, Online. Association for Computational Linguistics.

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. IEEE.

Sarubi Thillainathan, Surangika Ranathunga, and Sanath Jayasena. 2021. Fine-tuning self-supervised multilingual sequence-to-sequence models for extremely low-resource nmt. In 2021 Moratuwa Engineering Research Conference (MERCon), pages 432--437. IEEE.

Pasindu Udawatta, Indunil Udayangana, Chathulanka Gamage, Ravi Shekhar, and Surangika Ranathunga. 2024. Use of prompt-based learning for code-mixed and code-switched text classification. World Wide Web, 27(5):63.

A Vaswani. 2017. Attention is all you need. Advances in Neural Information Processing Systems.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 483--498, Online. Association for Computational Linguistics.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020. Improving massively multilingual neural machine translation and zero-shot translation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1628--1639, Online. Association for Computational Linguistics.

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, Nusa Dua, Bali. Association for Computational Linguistics.