Team IndiDataMiner at IndoNLP 2025: Hindi Back Transliteration - Roman to Devanagari using LLaMa

Saurabh Kumar, Dhruvkumar Babubhai Kakadiya, and Sanasam Ranbir Singh
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
{saurabh1003, d.kakadiya, ranbir}@iitg.ac.in

Abstract

The increasing use of Romanized typing for Indo-Aryan languages on social media poses challenges due to its lack of standardization and loss of linguistic richness. To address this, we propose a sentence-level back-transliteration approach using the LLaMa 3.1 model for Hindi. Leveraging fine-tuning with the Dakshina dataset, our approach effectively resolves ambiguities in Romanized Hindi text, offering a robust solution for converting it into the native Devanagari script.

1 Introduction

The widespread use of social media platforms and the prevalence of English keyboards have led to a significant rise in the use of Romanized typing for Indo-Aryan languages, primarily for quick and informal communication. However, Romanized text on social media often lacks consistency, with variations in spelling, phonetic representation, and vowel omission. This lack of standardization introduces ambiguity, as the same word can be written in multiple ways, such as नमस्ते (*Namaste*) appearing as *Namste*, *Nmst*, or *Namastey*. Romanized text also involves one-to-many mappings based on context, such as Romanized text *sir* can correspond to सिर(English: head) or सर(English: Sir) based on the context. Such inconsistencies lead to misunderstandings in human communication and errors in NLP applications like machine translation.

In addition to standardization issues, Romanized scripts fail to preserve the linguistic richness and phonetic nuances of native scripts, often losing cultural and linguistic expression. Certain sounds in Hindi and other Indo-Aryan languages lack precise representation in Roman script, resulting in phonetic ambiguities. For example, the Hindi letters ट (retroflex T) and त (dental T) are both commonly written as *T* or *Ta* in Romanized text, ignoring the critical distinction between retroflex and dental sounds in native pronunciation. Similarly, English sounds do not always map neatly to Hindi phonetics. For instance, the English sounds *v* and *w* are often transliterated as व (v), which can be confused with *b*-like sounds such as ब or भ. Such limitations underscore the challenges of relying solely on Romanized text for meaningful communication and accurate linguistic representation.

These challenges emphasize the need for robust back-transliteration systems to convert Romanized Indo-Aryan text into native scripts. Back-transliteration maps Romanized text to its native script based on phonetic representation, addressing the absence of standardization and variability in typing habits. Accurate back-transliteration enhances digital communication by promoting cultural preservation, improving readability, and reducing miscommunication. Furthermore, it facilitates the integration of Romanized content into automated systems such as machine translation, text-to-speech, and text mining, significantly boosting their effectiveness and utility.

Transliteration can be approached at both the word level and the sentence level. Word-level transliteration models often fall short due to their inability to account for contextual information, which is essential for accurately resolving ambiguities in Romanized text. This study explores sentence-level transliteration for Hindi, leveraging the LLaMa 3.1(8B) (Dubey et al., 2024) model. The experiments include both zero-shot learning and fine-tuning approaches. For fine-tuning, the Dakshina dataset (Roark et al., 2020a) is employed.

Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages (IndoNLP2025), pages 155–160
January 20, 2025. ©2025 Association for Computational Linguistics

The fine-tuned LLaMa 3.1 model achieves significant improvements in transliteration accuracy, as demonstrated by the BLEU scores on the Hindi Test dataset[1]. On Test Set 1, the model achieves a BLEU score of 0.8866 for character overlap and 0.6288 for word overlap. On Test Set 2, the BLEU scores are 0.8176 for character overlap and 0.5105 for word overlap. These results underscore the effectiveness of fine-tuning in improving transliteration performance, providing a robust solution for the challenges associated with Romanized Hindi text conversion.

Split	Script	#Data	#Word	#Char
Train	Roman	10041	17.50	102.08
	Native	10041	17.50	92.42
Test Set1	Roman	9998	15.30	89.09
	Native	9998	15.30	80.63
Test Set2	Roman	4998	15.29	80.11
	Native	4998	15.28	80.46

Table 1: Statistics of the Training and Testing Dataset. Here **#Data** represents the number of text samples, **#Word** denotes the average number of words per text sample, and **#Char** indicates the average number of characters per text sample.

2 Related Works

In recent years, significant progress has been made in transliteration for Indo-Aryan languages. Notable contributions include Kunchukuttan et al. (2015), who introduced Brahmi-Net, a statistical transliteration system capable of handling script conversion across 18 Indo-Aryan languages, resulting in 306 language pairs, including Hindi. Similarly, Roark et al. (2020b) developed the Dakshina dataset, supporting transliteration and language modeling tasks for 12 South Asian languages written in Roman script, providing a foundational resource for this domain.

Building on these efforts, Kunchukuttan et al. (2021) explored multilingual neural machine transliteration for English and 10 Indian languages, demonstrating the potential of multilingual systems. Another significant milestone is the Aksharantar dataset presented by Madhani et al. (2023), which covers 21 Indian languages and achieved state-of-the-art results using the IndicXlit model. Additionally, Ruder et al. (2023) evaluated sentence-level transliteration across 13 languages, including 12 from the Dakshina dataset and Amharic, using transfer learning models like mT5-Base, ByT5-Base, and FlanPaLM-62B.

Transliteration for informal and social media text has also been addressed in shared tasks organized by the Forum for Information Retrieval (FIRE). For instance, FIRE 2013 and FIRE 2014 (Roy et al., 2013; Choudhury et al., 2014) focused on transliterating Hindi song lyrics written in Roman script, shedding light on the challenges of informal text processing. Transliteration of Romanized Assamese text on social media environment is explored in the study (Baruah et al., 2024b) and recently back transliteration of Romanized Assamese social media text is explored by Baruah et al. (2024a) using BiLSTM, Neural Transformer Model, mT5, and ByT5.

Despite these advancements, existing research does not specifically address the transliteration challenges posed by Romanized social media datasets, characterized by inconsistencies, non-standard typing patterns, and ad-hoc transliterations. This highlights the need for further research tailored to the complexities of social media communication.

3 Approach

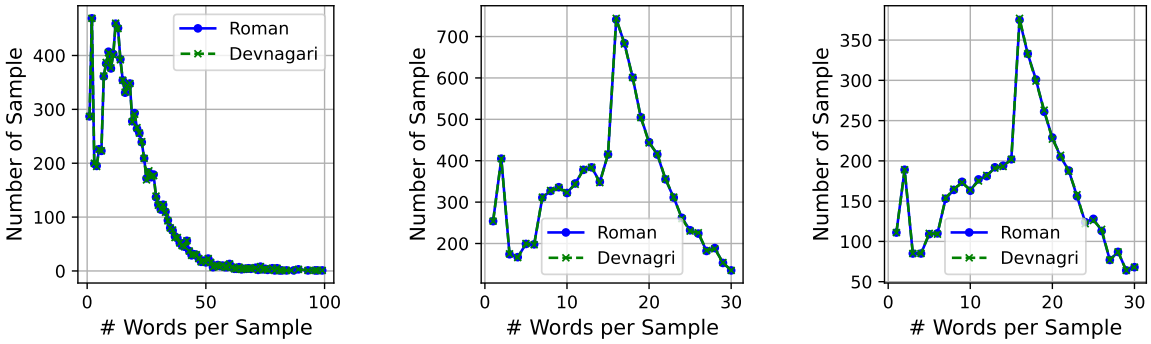
In this experiment, we focus on training a back-transliteration model to convert Romanized Hindi text into Devanagari script using a sentence-level model. The architecture used is the LLaMa 3.1 model, which is fine-tuned using a pre-defined set of instructions and inputs. The model training process includes both zero-shot and fine-tuning techniques to enhance the model's transliteration capabilities. The code for training is available at this GitHub repo[2].

3.1 Dataset

For training our model, we use the Dakshina dataset(Roark et al., 2020a), which provides a transliteration parallel corpus of 12 Indian languages, including Hindi. All the samples whose lengths are greater than 100 words are manually broken into smaller sentences. For

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

[2] https://github.com/saurabhdbz/LlaMa_Translit



(a) Train Set (b) Test Set-1 (c) Test Set-2

Figure 1: Word distribution of the Training and the Testing dataset

the testing, we have used the two sets of the dataset provided in the shared task[3]. The statistics of the Hindi dataset used for our training are tabulated in Table 1. The romanized text in Test Set 2 has most of the sample with the vowel omission. The same is reflected in Table 1 as well. The average character count for romanized text in Test Set 2 is less than that of Test Set 1. The word distribution of each dataset is shown in Fig. 1. It is observed that most of the samples in training data fall under the sample length of 50 words, and for the testing data, the sample length is limited to 30 words.

The dataset is formatted to fit the Alpaca prompt structure, where the instruction is to transliterate the Romanized Hindi input back into Devanagari script. The dataset is processed by creating training examples that combine instructions, inputs, and outputs, with the end-of-sequence token (EOS) added to each instance to guide the model in generating complete sequences. The fixed instruction, "*Transliterate the given Romanized Hindi text back to Devanagari script.*" is consistently used across both the training and testing phases.

3.2 Model Architecture

The foundation of our system is the LLaMA 3.1 8B model (Dubey et al., 2024), a large-scale transformer-based architecture with 8 billion parameters. This model is multilingual and supports a significantly extended context length of 128K, making it suitable for advanced use cases such as long-form text summarization, multilingual conversational agents, and coding assistants. The fine-tuned variant of LLaMA employed in this work is optimized for causal language modeling and enhanced with Low-Rank Adaptation (LoRA) and 4-bit quantization. LoRA is applied with a rank of 16, enabling efficient adaptation by training lightweight low-rank matrices while freezing the original model weights, significantly reducing the number of trainable parameters. The model consists of 32 decoder layers, each comprising self-attention and feed-forward modules. All projections (query, key, value, and output) within the self-attention mechanism leverage low-rank matrices, with rotary embeddings incorporated for positional encoding. The use of 4-bit quantization further minimizes memory and computational overhead, making the model highly efficient for resource-constrained environments while maintaining its performance quality.

3.3 Training Method

The training process utilizes the `SFTTrainer` class from the `trl` library, designed explicitly for supervised fine-tuning of language models. To improve memory efficiency, we integrated the Unsloth[4] framework, which supports 4-bit quantization by loading the pre-trained model in a compressed format. This approach accelerates training and inference while significantly reducing the memory footprint.

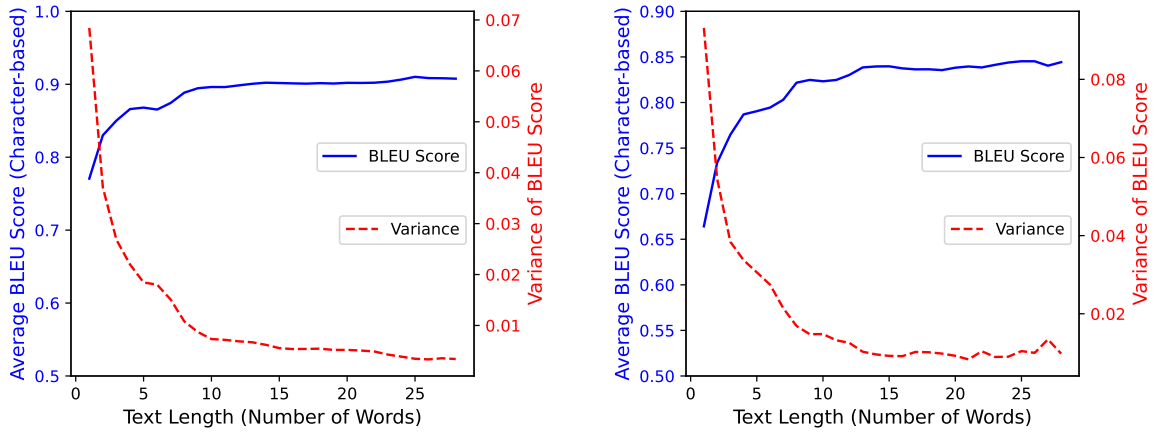
The model is fine-tuned for one epoch with a batch size of 2, using gradient accumulation steps set to 4 to manage the training of the

[3]IndoNLP Workshop 2025: https://indonlp-workshop.github.io/IndoNLP-Workshop/

[4]Unsloth: https://github.com/unslothai/unsloth

Model	Test Set 1				Test Set 2			
	WER	CER	BLUE$_C$	BLUE$_W$	WER	CER	BLUE$_C$	BLUE$_W$
IndicXlit	0.4552	0.1785	0.7319	0.2505	0.5320	0.2313	0.6567	0.1689
LLaMa3.1	0.2154	0.0881	0.8675	0.5996	0.2851	0.1339	0.8029	0.4879
Proposed	**0.1892**	**0.0684**	**0.8866**	**0.6288**	**0.2640**	**0.1183**	**0.8176**	**0.5105**

Table 2: Model performance on both test sets, evaluated using Word Error Rate (WER), Character Error Rate (CER), BLEU score for character overlap (BLEU$_C$), and BLEU score for word overlap (BLEU$_W$). The proposed model is the fine-tuned version of LLaMa 3.1.



(a)Performance on Test Set-1

(b)Performance on Test Set-2

Figure 2: Average BLEU score and the variance of the BLUE score across different text lengths of the sample from both Test Set 1 and Test Set 2.

large model size. The learning rate was configured to 2×10^{-4}, and the AdamW optimizer was employed with 8-bit precision to further reduce memory usage. Additionally, the training process incorporated a warm-up phase followed by linear learning rate decay to ensure stable convergence.

3.4 Back-transliteration

We employ both the pre-trained LLaMa model and the fine-tuned model to perform back-transliteration of Romanized Hindi text. In both cases, the same prompt, i.e. "*Transliterate the given Romanized Hindi text back to Devanagari script.*" is used. During text generation in both cases, the default temperature value of 1.0 is used, which strikes a balance between randomness and determinism, ensuring natural and coherent output.

4 Results and Discussion

We evaluate the model's performance in two scenarios: a zero-shot setting, where responses are generated directly from the pre-trained model using prompts, and after the model fine-tuning, by analyzing its responses on two test datasets: Test Set 1 and Test Set 2. The performance metrics include Word Error Rate (WER), Character Error Rate (CER), and BLEU score.

For the BLEU score, we compute two distinct types of overlap: Character-Level Overlap and Word-Level (or Token-Level) Overlap. The BLEU score for Character-Level Overlap evaluates the precision of individual characters in the generated output compared to the reference, making it particularly useful for fine-grained tasks such as transliteration. On the other hand, the BLEU score for Word-Level Overlap measures the precision of word-level tokens in the generated output, which is more suited for tasks emphasizing semantic accuracy and fluency. The BLEU score is calculated by assigning equal weight to unigrams, bigrams, trigrams, and fourgrams to ensure a balanced evaluation across different n-gram

levels.

We compare our model against IndicXlit (Madhani et al., 2023), considering it as baseline. It is a transformer-based state-of-the-art multilingual transliteration model with 11 million parameters, supporting 21 Indian languages for Roman-to-native and native-to-Roman script conversions. Using IndicXlit, the Romanized Hindi text was converted into Devanagari and compared with the outputs of our trained models.

Table 2 summarizes the performance of the models on Test Set 1 and Test Set 2. The pre-trained LLaMa model outperforms the baseline IndicXlit model on both test sets, achieving significant reductions in WER and CER. On Test Set 1, the WER and CER are reduced by 24% and 9%, respectively, while on Test Set 2, the reductions are 25% and 10%, respectively. Additionally, the Character-Level BLEU score shows a gain of 13%, and the Word-Level BLEU score improves by 34% on Test Set 1, with similar improvements observed on Test Set 2.

The fine-tuned model demonstrates the best performance overall. On Test Set 1, it achieves a WER of 18.92% and a CER of 6.84%. For BLEU scores, the fine-tuned model achieves 88.66% for Character-Level Overlap and 62.88% for Word-Level Overlap, representing gains of 15.47% and 37.83%, respectively. Similarly, on Test Set 2, the model significantly reduces the WER and CER by 41.37% and 11.3%, respectively, compared to the IndicXlit baseline. Furthermore, it achieves a BLEU score of 81.76% on Character-Level Overlap for Test Set 2, underscoring its effectiveness in transliteration tasks.

Additionally, we analyze the relationship between text length and model performance by plotting line graphs of the average BLEU score and its variance against text length for both Test Set 1 and Test Set 2, as shown in Fig. 2. From the graphs, we observe that the model's performance remains relatively consistent for texts longer than 8 words across both test sets. However, a slightly higher variance in BLEU scores for smaller text indicates that the model's performance is less stable on text of smaller length.

5 Conclusion and Future Work

This paper addresses the challenges of back-transliteration of Romanized Hindi text, which often suffers from inconsistencies in spelling, phonetic representation, and the omission of vowels. We explore the use of the LLaMa 3.1 (8B) model for back-transliteration, employing both prompting and fine-tuning methods. For fine-tuning, the Dakshina dataset was utilized. Our results demonstrate significant improvements in transliteration accuracy, as measured by Word Error Rate (WER), Character Error Rate (CER), and BLEU score, providing an effective solution for handling the variability in Romanized text and enhancing the performance of NLP applications such as machine translation and text mining.

In future work, we plan to extend our approach to other Indo-Aryan languages, incorporating larger and more diverse datasets. We also aim to refine the model to handle even greater text variability and improve transliteration accuracy further. Additionally, exploring domain-specific adaptations and integrating the model into real-time applications will be key directions for advancing back-transliteration systems in the future.

Limitations

This work is primarily limited to the Hindi language and focuses on more structured text. The training data used for model development lacks the nuances of social media text, such as abbreviations, short forms, and vowel omissions. As a result, the model's performance declines for shorter sentences and on datasets like Test Set 2, which include texts with vowel omissions.

Additionally, the study is restricted to transformer-based models, specifically the encoder-decoder architecture and the LLaMa model. While large language models (LLMs) like LLaMa demonstrate superior performance, their significant size makes them less suitable for deployment on resource-constrained devices, such as mobile phones, for real-time transliteration. To address this, future work should explore model compression techniques to reduce the computational footprint and enhance applicability in such environments.

References

Hemanta Baruah, Sanasam Ranbir Singh, and Priyankoo Sarmah. 2024a. AssameseBackTranslit: Back transliteration of Romanized Assamese social media text. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1627–1637, Torino, Italia. ELRA and ICCL.

Hemanta Baruah, Sanasam Ranbir Singh, and Priyankoo Sarmah. 2024b. Transliteration characteristics in romanized assamese language social media text and machine transliteration. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 23(2).

Monojit Choudhury, Gokul Chittaranjan, Parth Gupta, and Amitava Das. 2014. Overview of fire 2014 track on transliterated search. *Proceedings of FIRE*, pages 68–89.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Anoop Kunchukuttan, 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.

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

Brian Roark, Lawrence Wolf-Sonkin, Christo Kirov, Sabrina J Mielke, Cibu Johny, Isin Demirsahin, and Keith Hall. 2020a. Processing south asian languages written in the latin script: the dakshina dataset. *arXiv preprint arXiv:2007.01176*.

Brian Roark, Lawrence Wolf-Sonkin, Christo Kirov, Sabrina J. Mielke, Cibu Johny, Isin Demirsahin, and Keith Hall. 2020b. Processing South Asian languages written in the Latin script: the Dakshina dataset. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2413–2423, Marseille, France. European Language Resources Association.

Rishiraj Saha Roy, Monojit Choudhury, Prasenjit Majumder, and Komal Agarwal. 2013. Overview of the fire 2013 track on transliterated search. In *Proceedings of the 4th and 5th Annual Meetings of the Forum for Information Retrieval Evaluation*, FIRE '12 & '13, New York, NY, USA. Association for Computing Machinery.

Sebastian Ruder, Jonathan Clark, Alexander Gutkin, Mihir Kale, Min Ma, Massimo Nicosia, Shruti Rijhwani, Parker Riley, Jean-Michel Sarr, Xinyi Wang, John Wieting, Nitish Gupta, Anna Katanova, Christo Kirov, Dana Dickinson, Brian Roark, Bidisha Samanta, Connie Tao, David Adelani, Vera Axelrod, Isaac Caswell, Colin Cherry, Dan Garrette, Reeve Ingle, Melvin Johnson, Dmitry Panteleev, and Partha Talukdar. 2023. XTREME-UP: A user-centric scarce-data benchmark for under-represented languages. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1856–1884, Singapore. Association for Computational Linguistics.