Sentence-Level Back-Transliteration of Romanized Indian Languages: Performance Analysis and Challenges

Saurabh Kumar, Dhruvkumar Babubhai Kakadiya, Sanasam Ranbir Singh, Sukumar Nandi
Indian Institute of Technology Guwahati, Assam, India
{saurabh1003, d.kakadiya, ranbir, sukumar}@iitg.ac.in

Abstract

The widespread use of Romanized text for Indian languages, particularly on social media platforms, poses significant challenges for natural language processing due to the lack of standardized orthography and the presence of contextual ambiguities. In this study, we explore sentence-level back-transliteration for 13 Indian languages, focusing on addressing the limitations of word-level models that fail to capture contextual dependencies. We evaluate state-of-the-art models, including fine-tuned LLaMA, mT5, and Multilingual Transformer models, comparing their performance against the baseline IndicXlit model. In addition, we conduct a comprehensive error analysis to gain deeper insights into model performance. Our results demonstrate that fine-tuned LLaMA and the proposed IndiXform model, specifically designed to leverage sentence-level context, significantly outperform zero-shot LLaMA and the IndicXlit baseline. These findings provide valuable insights into handling contextual ambiguities and enhancing the accuracy of back-transliteration systems for Indian languages.

Keywords: Transliteration, Evaluation, Multilinguality

1. Introduction

The use of Romanized text for Indian languages has become widespread due to the dominance of mobile communication and social media platforms. While native-script keyboards are available, users often default to the Roman script for faster input and ease of use. However, the absence of a standardized orthography for Romanized text for Indian languages introduces significant challenges in natural language processing (NLP). Unlike languages with established Romanization systems (e.g., Pinyin for Chinese), Indian languages exhibit highly inconsistent spelling conventions, influenced by phonetic variations, user preferences, and regional dialects.

The lack of standardization in Roman-to-native transliteration introduces significant ambiguity in both many-to-one and one-to-many mappings. For example, as illustrated in Fig. 1, the Romanized words *Main*, *Mei*, and *Mai* are all mapped to the same Hindi word **मैं** in the Devanagari script, despite their distinct phonetic variations in different contexts. In the Assamese language, it is found that a total of 37 different Roman transliteration variations are available (Baruah et al., 2024b). Examples from all the languages are listed in Table 1. Conversely, a single Romanized word may correspond to multiple possible transliterations in the native script depending on the contextual meaning. For instance, the Roman word *sir* can be transliterated as either **सर** or **सिर**, and *kar* may map to either **कर** or **कार** in Devanagari. This variability underscores the challenges of developing robust transliteration systems capable of effectively handling such linguistic ambiguities.

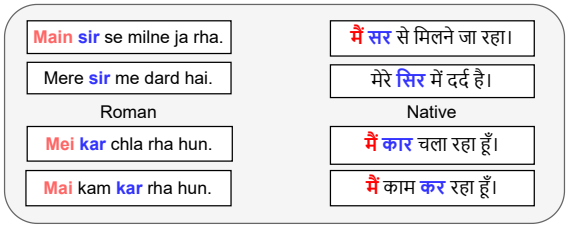

Figure 1: Example of Hindi text written in Roman and Devanagari script.

Most existing transliteration systems rely on word-pair or character-to-character mappings, which inherently lack contextual information (Kirov et al., 2024). These approaches treat transliteration as a direct mapping task, disregarding linguistic dependencies within a sentence. As a result, they struggle with disambiguation, leading to errors in cases where multiple possible transliterations exist for a given word. Since word-level models operate in isolation without considering the surrounding context, they often produce inconsistent or incorrect output. To address these limitations, there is a need to explore sentence-level transliteration approaches that incorporate contextual cues, enabling more accurate and reliable transliteration. Despite the growing interest in Indic transliteration, several questions remain unexplored:

RQ1: How do state-of-the-art models perform on sentence-level back-transliteration for Indian languages, particularly in terms of handling contextual

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 818–827
11–16 May 2026.

Lang	Native	Multiple romanized representations
as	অসমীয়া	achomiya, ahomiya, akhamia, akhmia, akhomia, akhomiya, akxomiya, anaxomiya, aokhomia, asamia, asamiya, ashaima, asomia, asomiya, asomiyoi, assamese, assamia, assamiya, assomiya, axamia, axamiya, axhomiya, axmia, axmiya, axomeaa, axomi, axomia, axomiaaaa, axomiay, axomiya, axonia, ohomia, okhamiya, okhomeya, okhomia, okhomiya, okomiya, osomia, oxomia, oxomiya
bn	মাধ্যমে	maddhom, maddhome, maddhomme, maddhoyme, maddhyme, maddhyome, maddome, madhom, madhome, madhomea, madhyom-e, madhyoma, madhyome, mddhyme, modhe, modhyome
gu	સુહાનાને	suhaanaane, suhanane
hi	भारतीय	Bharatiy, Bharatiya, Bharatoy, Bharity, Bhartiy, Bhartiya, Bhrtiy, Bhrtiya, bhaaratiy, bhaartiy, bhaartiya, bhaatiy, bharatiy, bharatiya, bhartiy, bhartiya, bhrtiy, bhrtiya
kn	ಭಾರತದ	Bhaarata, Bhaaratha, Bharata, Bharatha, India, bhaarata, bhaaratha, bharata, bharatha, vibhajaneyindaa
ml	ഭരത്താവ്	Bhartthaavu, Bhrtthavu, bhartthaav, bhartthaavu, bhrtthav, bhrtthavu
mr	एकदिवसीय	Ekadivashiya, Ekadivasiy, Ekdivasiya, eakdiwasya, ekadivasiy, ekadivasiya, ekdivaseeya, ekdivasiya, ekdivsiya
pa	ਪੰਜਾਬੀ	Panjabi, Punjab, Punjabbi, Punjabi, Punjbai, pajabi, panjaabi, punjab, punjabi, punjai, punjani, punjbai
sd	پنهنجي	Panhjay, Panhje, Pehnji, pahinje, pahnjay, pahnji, panhjay, panhje, panhjee, panhji, panhnje, paye, pea(father), pehje, pehji, pehnje, pehnji, penhainjen, penhainji, penhaji, penhanje, penhanji, penhinje, penhje, penhnje, penhnji, penjahi
si	සහෝදරියක්	sahodariyak, shodriyk
ta	மூன்று	Moondru, mondru, moodru, moondru, moondu, moonru, mundru, munru, muundru, muunru
te	జాతీయ	Jaateeya, Jateeya, Jathiya, Jatiya, gaateeya, gitam, jaateeya, jaatheeya, jaathiya, jateeya, jathiya, jeeteeya
ur	علیہ	Alaihi, Alaihis, aalehi, alaih, alaih-e-, alaihe, alaihey, alaihi, alaihis, alaihissalaam, aleh, alyhi, alyhis

Table 1: Multiple representation for native word into roman in Assamese(as), Bengli(bn), Gujarati(gu), Hindi(hi), Kannada(kn), Malayalam(ml), Marathi(mr), Punjabi(pa), Sindhi(sd), Sinhala(si), Tamil(ta), Telugu(te), and Urdu(ur)

ambiguities?

RQ2: To what extent can modern large language models (LLMs) be effectively utilized for sentence-level back-transliteration without task-specific fine-tuning and with fine-tuning?

RQ3: How effective are existing publicly available parallel corpora in supporting sentence-level back-transliteration for Indian languages, especially in the context of informal social media text?

RQ4: How do social media-specific text characteristics (e.g., informal spellings, code-mixing) affect the performance of back-transliteration models?

To address these questions, this study presents a comprehensive analysis of sentence-level back-transliteration for Indian languages, focusing on state-of-the-art models and publicly available datasets. We investigate the effectiveness of existing parallel corpora, including the Dakshina (Roark et al., 2020a) and IndoNLP (Weerasinghe et al., 2025) datasets, in supporting sentence-level transliteration tasks. Our evaluation encompasses advanced models such as LLaMA 3.1, IndicXlit, and mT5, assessing their capabilities in handling contextual ambiguities. Additionally, we introduce IndiXform, a transformer-based transliteration model designed specifically to leverage sentence-level context, aiming to improve accuracy and disambiguation in back-transliteration.

Our results show that sentence-level models leveraging contextual information significantly outperform word-level approaches, with IndiXform and fine-tuned LLaMA achieving the lowest error rates. However, challenges persist with code-mixed text and diacritic generation. These findings highlight the importance of contextual modeling and the need for enhanced training strategies to improve transliteration accuracy across diverse linguistic scenarios.

2. Related Work

Research on transliteration for Indo-Aryan languages has witnessed considerable progress over the years, with numerous important contributions. One early effort, Brahmi-Net, introduced by Kunchukuttan et al. (2015), provided a statistical approach capable of handling script conversion across 18 Indo-Aryan languages, covering a total of 306 language pairs, including Hindi. This work laid the groundwork for subsequent large-scale multilingual transliteration studies. Beyond these foundational works, other noteworthy corpora have been developed for transliteration-specific tasks. The NEWS 2015 Machine Transliteration Shared Task (Banchs et al., 2015) focused on transliteration of proper names across 14 language pairs. Additionally, Gupta et al. (2012) proposed an approach to mine Hindi-English transliteration pairs from Hindi song lyrics, creating a dataset of 30,823 word pairs. Another pivotal resource, the Dakshina dataset developed by Roark et al. (2020b), supports both transliteration and language modeling tasks across 12 South Asian languages. By providing data for Roman-scripted text, this dataset has become essential for research on Indian languages represented in non-native scripts.

Building on these foundational efforts, Kunchukuttan et al. (2021) demonstrated the effectiveness of multilingual neural machine transliteration for English and 10 Indian languages,

Statistic	as	bn	gu	hi	kn	ml	mr	pa	sd	si	ta	te	ur
#Total	15076	14991	10358	14985	9905	15000	9998	9739	9658	15066	9865	9909	9569
#Train	12060	11992	8286	11988	7924	12000	7999	7791	7726	12004	7892	7927	7591
#Dev	1507	1499	1035	1499	990	1500	1000	973	965	1500	986	990	948
#Test	1509	1500	1037	1498	991	1500	999	975	967	1502	987	992	1030
#Token(R)	32.0k	46.4k	27.0k	41.6k	55.3k	42.9k	39.7k	40.8k	40.8k	15.5k	52.8k	48.9k	30.3k
#Token(N)	20.2k	31.6k	25.5k	26.3k	49.3k	30.3k	34.2k	29.4k	28.1k	11.7k	42.7k	41.6k	22.8k

Table 2: Dataset Statistics. **#Total**, **#Train**, **#Dev**, and **#Test** represent the total number of samples in the entire dataset, the training split, the development split, and the test split, respectively. **#Token(R)** and **#Token(N)** indicate the unique number of tokens present in Romanized and Native text across the entire dataset for each language.

showcasing how neural models can generalize across multiple languages where they created a corpus of 600K word pairs mined from parallel and monolingual corpora. Similarly, Madhani et al. (2023) presented the Aksharantar dataset, which spans 21 Indian languages and achieved state-of-the-art results with the IndicXlit model. Expanding the scope further, Ruder et al. (2023) examined sentence-level transliteration across 13 languages using transfer learning models such as mT5-Base, ByT5-Base, and FlanPaLM-62B, including languages from the Dakshina dataset and Amharic.

The challenges of transliterating informal and non-standard text, particularly from social media platforms, have also gained attention in recent years. Shared tasks organized by the Forum for Information Retrieval (FIRE) have specifically focused on such text. For example, tasks held in FIRE 2013 and FIRE 2014 (Roy et al., 2013; Choudhury et al., 2014) tackled the problem of transliterating Romanized Hindi song lyrics, thereby highlighting the complexities of informal text processing. Baruah et al. (2024b) explored the transliteration of Romanized Assamese text found in social media content, while Baruah et al. (2024a) investigated back-transliteration of similar text using BiLSTM, Neural Transformer models, mT5, and ByT5. Recently, the IndoNLP workshop (Weerasinghe et al., 2025) introduced a back-transliteration task for five Indo-Aryan languages, aiming to develop a real-time reverse transliterator to enhance the typing experience for users of these languages. In response Baiju et al. (2025); De Mel et al. (2025); Perera et al. (2025), and Kumar et al. (2025) have explored various methods including rule-based, deep learning, transformer-based encoder-decoder model and LLMs like LLaMA 3 for different Indo-Aryan languages.

3. Proposed Approach

We have considered the sentence level transliteration of 13 Indian languages written in Roman script back to their native script. These languages include Assamese (as), Bengali (bn), Gujarati (gu), Hindi (hi), Kannada (kn), Malayalam (ml), Marathi (mr), Punjabi (pa), Sindhi (sd), Sinhala (si), Tamil (ta), Telugu (te), and Urdu (ur). We have investigated different state-of-the-art sequential models, including LLaMA 3.1 (Dubey et al., 2024), Transformer-based encoder-decoder model (Vaswani, 2017), and mT5 (Xue et al., 2021) for the task. We have also investigated zero-shot learning approach with LLaMA 3.1 model.

3.1. Dataset

We use the dataset from two major sources: one is Dakshina (Roark et al., 2020a), and the other is from the IndoNLP Workshop (Weerasinghe et al., 2025) shared task[1]. IndoNLP dataset includes data for Bengali, Gujarati, Hindi, Malayalam, and Sinhala and is characterized by complex, irregular writing patterns such as vowel omissions, and shortened words resembling informal text from social media. Assamese language dataset is taken from the work (Baruah et al., 2024a). The remaining languages, including Kannada, Marathi, Punjabi, Sindhi, Tamil, Telugu, and Urdu, are taken from Dakshina dataset. Unlike IndoNLP dataset, Dakshina follows more structured orthographic rules with well-formed romanized text, making it suitable for systematic transliteration tasks.

To ensure a balanced evaluation of the model, the dataset is split into training (Train) (80%), development (Dev) (10%), and test (Test) (10%) sets for each language. Table 2 provides detailed statistics of each set across different languages, including the total number of samples and unique tokens per language.

With this dataset we investigate the back transliteration work with different models. We investigate zero-shot and prompt fine-tuning approach with LLaMA 3.1 along with state-of-the-art transliteration model IndicXlit.

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

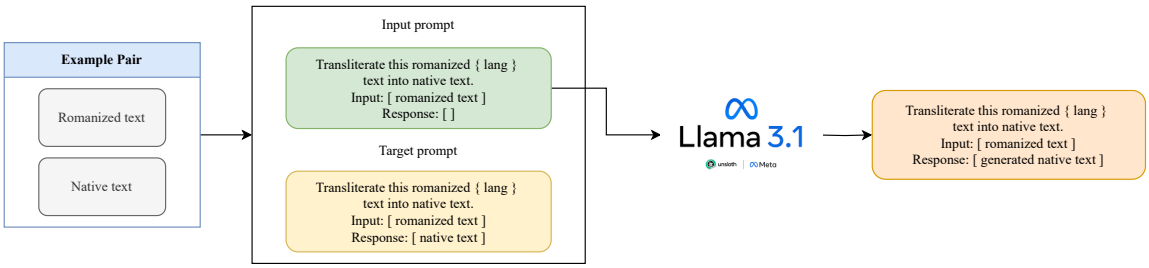

Figure 2: LLaMA Prompt Fine-tuning

3.2. Back-Transliteration with LLaMA 3.1

We explore both zero-shot and prompt fine-tuning methods to evaluate the LLaMA 3.1 8B model (Dubey et al., 2024) for the back-transliteration task across 13 Indian languages. The LLaMA 3.1 8B model is a large transformer-based architecture with 8 billion parameters. It is designed to be multilingual and supports an extended context length of 128K, making it well-suited for applications such as long-form text summarization, multilingual dialogue systems, and coding assistance.

3.2.1. Zero-Shot Testing

In zero-shot testing, the model is evaluated without any task-specific fine-tuning. We have applied an Alpaca-style prompt to assess the model's capability to perform back-transliteration from romanized text to native script. The prompt provided to the model followed the following format:

"Back transliterate this romanized {language} text into native {language} text."

For example, the prompt for Hindi text is structured as:

> `Instruction:`
> Back transliterate this romanized Hindi text into native Hindi text:

This method enabled us to measure the model's inherent performance in transliteration tasks without additional training.

3.2.2. Fine-Tuning with Alpaca Prompts

The second experiment involves fine-tuning LLaMA 3.1 using a task-specific dataset. We use the dataset consisting of pairs of romanized and native script texts for each language. Two different prompting approaches are used to fine-tune the model. In the first approach, the model is fine-tuned with individual languages, where the language tag is included in the prompt. In the second approach, the model learns to identify the language and perform back-transliteration. For this approach, we merge the datasets from each language and prompt-tune the model accordingly.

The fine-tuning process follows these steps:

- **Formatting Prompts**: Each example is converted into an Alpaca-style prompt. The instruction specifies the transliteration task, the input contains romanized text, and the response provides the native script. The prompt template is:

> Below is an instruction that describes a task, paired with an input. Write a response that appropriately completes the request.
>
> `Instruction:`
> Transliterate this romanized {language} text into native {language} script.
> `Input:` {romanized text}
> `Response:` { }

- **Model Adaptation**: The FastLanguageModel API from Unsloth loads and prepares the base model. LoRA (Low-Rank Adaptation) is applied to fine-tune the model efficiently with minimal computational overhead.

- **Optimization Parameters**: Hyperparameters include a learning rate of 2×10^{-4}, a batch size of 4, 400 training steps with a linear learning rate scheduler, LoRA-specific parameters (rank of 16, alpha value of 16, zero dropout), 4-bit model loading for memory savings, and gradient checkpointing for efficiency.

- **Training Process**: Mixed precision is used to accelerate computation and reduce memory usage. The dataset is tokenized and processed in batches with gradient accumulation to support smaller batch sizes.

In another prompt-tuning approach, the instruction first identifies the language of the given sample and then performs back-transliteration. All other parameters for prompt tuning remain the same as in the previous training. The following instruction is used for this approach:

> `Instruction:`
> 1. Identify the language from the given romanized text.

2. Transliterate this romanized text into the identified native script.

After fine-tuning, the model is evaluated with the same Alpaca prompt format to ensure consistency. The fine-tuned model shows improved accuracy and fluency, especially for complex or low-frequency word forms.

3.3. IndiXform: Back-transliteration with Transformer

We further train IndiXform, a transformer-based encoder-decoder model for sentence-level back-transliteration. The model consists of 6 layers in both the encoder and decoder, with 256-dimensional input embeddings, a 1024-dimensional feedforward network, and 8 attention heads. We implement the model using Facebook's Fairseq toolkit (Ott et al., 2019), a robust and scalable sequence-to-sequence model for multilingual NLP tasks.

We explore both a multilingual character-level transliteration model (Kunchukuttan et al., 2021) and a multilingual subword-level model. For the subword-level model, we use Byte Pair Encoding (BPE) for training.

3.3.1. Training Procedure

For training the IndiXform Multilingual Transformer, we employ two tokenization methodologies: character-level tokenization and byte pair encoding (BPE) subword tokenization. We name these two models as **IndiXform$_{ct}$** and **IndiXform$_{swt}$** respectively.

Character-Level Tokenization In character-level tokenization, each character is treated as an individual token. All the training data across the 13 languages were combined into a single corpus file. During tokenization, white spaces are not treated as tokens. Instead, the special token `__WB__` (Word Break token) was introduced to separate words within sentences. A dictionary of 852 and 232 unique characters and tokens is formed from the dataset combining all text in native script and Roman script respectively.

Byte Pair Encoding (BPE) Subword Tokenization For BPE tokenization, we use the `subword-nmt` toolkit to learn subword merges. The BPE algorithm starts by splitting each word into its smallest units, usually individual characters or bytes. Frequent pairs of characters or subwords are then iteratively merged based on their frequency in the dataset. We have experimented with various sizes of frequent merges, including 8000, 16000, 24000, 32000, and 39000. After extensive testing, a merge size of 32000 provided the most meaningful results, balancing token coverage and model performance.

Training : The training process configures various hyperparameters to optimize the performance of the transformer model. The model uses a balanced set of encoder and decoder hyperparameters, with 6 layers for both, each layer containing 8 attention heads. The encoder and decoder have embedding dimensions of 256 and feedforward dimensions of 1024. Dropout regularization is set to 0.5 to prevent overfitting. GELU (Gaussian Error Linear Unit) serves as the activation function, and the Adam optimizer with betas $(0.9, 0.98)$ is used to update the model weights. The learning rate starts at 0.001 and follows an inverse square root scheduling strategy with 4000 warm-up updates. The batch size is set to 16, and the training runs for a maximum of 41 epochs. Multilingual dictionaries and encoder language tokens are used to handle different languages efficiently.

3.3.2. Evaluation Procedure

For evaluation, we use a beam search mechanism to generate transliterations. Beam search is a heuristic search algorithm that expands multiple hypotheses (*beams=4*) at each step and selects the top candidates based on their cumulative probabilities. In this setup, we generate the top four predictions (*nbest=4*) for each input sentence, allowing users to choose from multiple high-probability outputs. For evaluation, we select the best prediction from the top four generated outputs, choosing the one with the highest probability score. This approach ensures flexibility and robustness in real-world applications, as users can select the transliteration that best fits their needs from these high-confidence options.

3.4. Back-Transliteration with mT5

We further investigate back-transliteration using the Multilingual T5 (mT5) model (Xue et al., 2021). Developed by Google, mT5 extends the T5 (Text-to-Text Transfer Transformer) architecture and is pre-trained on the mC4 dataset covering 101+ languages. Its multilingual pre-training enables robust language representation, making it suitable for transliteration, language translation, and low-resource scenarios.

3.4.1. Fine-Tuning Process

We fine-tune mT5 on Romanized and native script text pairs across 13 languages using the following steps:

		as	bn	gu	hi	kn	ml	mr	pa	sd	si	ta	te	ur	avg
IndicXlit	**cer**	0.41	0.20	0.27	0.20	0.14	0.18	0.19	0.21	0.26	-	0.13	0.15	0.24	0.27
	wer	0.60	0.55	0.64	0.48	0.47	0.60	0.56	0.53	0.59	-	0.51	0.50	0.56	0.56
	bleu$_c$	47.7	72.1	58.8	70.5	81.8	74.6	72.4	68.1	61.7	-	81.4	79.0	63.2	63.1
	bleu$_w$	11.9	18.0	8.5	22.9	28.4	12.9	17.9	17.3	12.3	-	21.4	24.7	19.0	18.0
LLaMA$_{zs}$	**cer**	0.53	0.30	0.43	0.24	0.42	0.46	0.67	0.38	0.74	0.45	0.37	0.43	0.94	0.47
	wer	0.77	0.52	0.64	0.31	0.61	0.76	0.76	0.54	0.98	0.73	0.64	0.65	0.89	0.64
	bleu$_c$	36.8	67.7	50.0	80.9	54.3	47.7	63.1	58.1	40.9	43.2	63.0	53.3	54.4	55.2
	bleu$_w$	5.5	22.9	14.5	54.3	17.5	7.6	27.1	24.4	6.8	7.7	14.2	14.8	34.0	19.9
LLaMA$_{fts}$	**cer**	0.11	0.11	0.07	0.12	**0.04**	**0.02**	**0.06**	**0.06**	0.10	**0.02**	0.09	**0.06**	0.17	0.08
	wer	0.21	0.23	0.08	0.21	**0.19**	**0.11**	**0.21**	**0.19**	0.25	0.09	**0.29**	**0.25**	0.33	**0.20**
	bleu$_c$	85.4	**89.0**	95.6	90.5	**92.2**	96.1	**90.0**	**88.9**	84.6	94.9	**89.8**	**90.2**	81.5	**89.4**
	bleu$_w$	58.6	55.3	80.9	66.1	**58.7**	**73.5**	**53.4**	**58.9**	52.7	76.2	**48.9**	**49.2**	56.5	**60.1**
LLaMA$_{ftm}$	**cer**	0.16	0.19	0.08	0.15	0.14	0.06	0.14	0.14	0.31	0.06	0.17	0.17	0.23	0.15
	wer	0.28	0.34	0.21	0.25	0.27	0.24	0.32	0.29	0.55	0.20	0.40	0.37	0.36	0.31
	bleu$_c$	80.3	81.1	87.1	82.5	85.6	90.1	80.7	79.6	66.1	88.1	82.0	78.0	72.3	80.2
	bleu$_w$	44.5	42.8	57.4	58.6	48.6	48.6	41.3	48.3	25.9	54.6	35.8	38.4	49.2	45.5
IndiXform$_{ct}$	**cer**	**0.09**	**0.06**	**0.02**	**0.06**	0.06	**0.02**	**0.06**	0.07	**0.09**	**0.02**	**0.07**	0.07	0.13	**0.07**
	wer	**0.18**	0.24	**0.07**	0.19	0.24	**0.11**	0.27	0.23	**0.22**	**0.08**	0.31	0.30	0.27	**0.20**
	bleu$_c$	**86.8**	88.4	**96.1**	89.0	90.1	**96.4**	87.3	86.3	**85.6**	**95.7**	87.6	86.8	80.4	88.2
	bleu$_w$	**60.0**	48.1	**81.6**	58.1	52.0	72.5	43.3	52.9	**55.1**	78.2	38.3	40.3	51.4	55.0
IndiXform$_{swt}$	**cer**	0.14	0.17	0.09	0.13	0.17	0.11	0.15	0.16	0.16	0.07	0.16	0.20	0.24	0.15
	wer	0.25	0.40	0.21	0.30	0.48	0.35	0.42	0.36	0.37	0.17	0.51	0.57	0.40	0.37
	bleu$_c$	81.1	75.1	86.1	79.1	75.7	83.9	75.6	75.1	74.5	88.3	75.9	69.7	70.0	77.4
	bleu$_w$	48.6	28.1	55.1	42.4	22.0	33.6	25.9	34.1	35.0	61.9	18.7	15.5	35.8	36.3
mT5-base	**cer**	0.12	0.07	0.03	**0.06**	0.09	0.04	0.08	0.13	0.13	**0.02**	0.09	0.10	**0.10**	0.08
	wer	0.23	**0.20**	0.09	**0.15**	0.27	0.19	0.26	0.26	0.25	0.10	0.30	0.33	**0.22**	0.22
	bleu$_c$	82.9	88.7	94.8	**90.6**	86.7	93.4	86.1	81.0	80.7	95.4	86.0	83.6	**84.1**	87.1
	bleu$_w$	54.2	**57.7**	79.8	**67.8**	49.4	63.0	47.0	52.5	53.9	**78.3**	41.4	39.5	**57.9**	56.3

Table 3: Evaluation Results Across Models and Languages, with best (lowest) **CER**/**WER** and best (highest) **BLEU$_c$**/**BLEU$_w$** in bold for each language. The symbol '-' indicates that the metric is not applicable.

– **Prompt Formatting**: Each example follows a task-specific prompt:

Transliterate {language}: {romanized text}

– **Tokenization**: The mT5 tokenizer processes input and target sequences with a maximum length of 64 tokens, ensuring uniform batch sizes.

– **Model Adaptation**: We use the "mT5ForConditionalGeneration" model with pre-trained weights from 'google/mT5-base'.

– **Optimization**: Fine-tuning employs a learning rate of 2×10^{-4}, a batch size of 8, 6 training epochs, and a weight decay of 0.01. Logging occurs every 100 steps, with checkpoints saved every 500 steps.

– **Training**: The Hugging Face 'Trainer' class handles training and evaluation, selecting the best-performing checkpoint.

– **Inference**: The model generates native script text from romanized input using the same prompt format:

Transliterate {language}: {romanized text}

The fine-tuned mT5 model demonstrates strong performance in back-transliteration tasks, effectively handling multiple languages and accurately generating native script text from romanized inputs. Its ability to generalize across languages further validates the benefits of multilingual pre-training.

4. Result and Evaluation

We evaluate the performance of all the six trained transliteration models at sentence level across 13 Indian languages i.e. Assamese (as), Bengali (bn), Gujarati (gu), Hindi (hi), Kannada (kn), Malayalam (ml), Marathi (mr), Punjabi (pa), Sindhi (sd), Sinhala (si), Tamil (ta), Telugu (te), and Urdu (ur). We consider (i) Back-transliteration with LLaMA 3.1: Zero-Shot approach (**LLaMA$_{zs}$**), Prompt-fine-tuning using individual languages (**LLaMA$_{fts}$**), and Prompt-fine-tuning combining all languages with language identification and back-transliteration (**LLaMA$_{ftm}$**), (ii) Back-transliteration with Transformer based encoder-decoder model: using character-level tokenization (**IndiXform$_{ct}$**) and using sub-word level tokenization with BPE (**IndiXform$_{swt}$**), and (iii) Back-Transliteration with mT5 (**mT5-base**). We consider the **IndicXlit** model as the baseline. IndicXlit is a transformer-

Error	Lang	Roman	Reference	Prediction
Deletion	hi	Matngi devi samta ka suchak hai .	**मा**तंगी देवी समता का सूचक है ।	**म**तंगी देवी समता का सूचक है ।
	gu	pr bhooleene pNa srch n kro aa paaanch vstu, phsaai jsho mushkeleemaa	પર ભૂલીને પણ સર્ચ ન કરો આ પાંચ વ**સ્તું**, ફસાઇ જશો મુશ્કેલીમાં	પર ભૂલીને પણ સરચ ન કરો આ પાંચ વ**સ્તુ**, ફસાઇ જશો મુશ્કેલીમાં
Insertion	hi	Baad men chalkar inhi logon me naujawan Bharat sabha ki sthapana ki.	बाद में चलकर इ**न्ही** लो**गो** ने नौजवान भारत सभा की स्थापना की।	बाद में चलकर इ**न्हीं** लो**गों** में नौजवान भारत सभा की स्थापना की।
	gu	pr aavee rhyaa chhe aa chaar jbrdst phichrs	પર આવી ર**હ્યા** છે આ ચાર જબરદસ્ત ફિચર્સ	પર આવી ર**હ્યાં** છે આ ચાર જબરદસ્ત ફિચર્સ
Substituted	hi	1994 ke baad se yah dal Chamling ke netrutwa men rajya men nirantar satta men hai.	**१९९४** के बाद से यह दल **चा**मलिंग के नेतृत्व में राज्य में निरंतर सत्ता में है।	**1994** के बाद से यह दल **चै**मलिंग के नेतृत्व में राज्य में निरंतर सत्ता में है।
	gu	slip. subh divs, srs ratri, aramdayk pthari.	**સ્લી**પ. શુભ દિવસ, સરસ રાત્રિ, આ**રામ**દાયક પથારી.	**સ્લિ**પ. શુભ દિવસ, સરસ રાત્રિ, આ**રાં**દાયક પથારી.
		alsno divs adbhut che prmtu mare khrekhr bhar jvum joie	આ**ળ**સનો દિવસ અદ્ભુત છે પરંતુ મારે ખરેખર બહાર જવું જોઈએ	અ**લ**સનો દિવસ અદ્ભુત છે પરંતુ મારે ખરેખર બહાર જવું જોઈએ

Table 4: Examples of different errors due to the mismatch in character count.

		as	bn	gu	hi	kn	ml	mr	pa	sd	si	ta	te	ur	avg
IndiXform_{ct}	ins%	3.38	3.04	2.25	2.93	2.82	2.14	3.21	3.02	6.04	2.5	3.46	3.46	3.97	3.24
	$\text{bleu}_{c,\ ins}$	79.1	83.6	89.7	84.6	85.4	90.6	83.5	81.7	77.3	90.2	84.1	81.5	82.0	84.1
	del%	6.89	3.92	2.29	3.46	5.1	2.18	3.87	5.54	6.21	2.76	5.01	6.48	12.33	4.93
	$\text{bleu}_{c,\ del}$	76.8	83.7	90.6	84.0	83.5	90.0	82.9	78.7	79.0	90.6	83.7	79.5	67.2	82.3
	$\text{bleu}_{c,\ sub}$	93.2	92.9	97.6	92.5	93.9	97.1	90.3	90.2	91.0	97.6	92.2	92.1	90.2	93.1
LLaMA_{fts}	ins%	4.2	4.63	5.44	4.29	2.76	2.24	3.87	2.95	3.99	2.62	4.35	3.78	4.26	3.77
	$\text{bleu}_{c,\ ins}$	75.0	83.0	86.8	84.0	86.8	89.9	84.3	85.1	78.1	89.5	86.0	84.8	82.3	84.3
	del%	3.59	3.99	2.4	3.44	2.89	2.69	3.56	3.64	4.07	2.69	3.79	3.39	8.22	3.73
	$\text{bleu}_{c,\ del}$	78.5	84.4	90.4	85.8	87.2	89.8	85.3	82.3	78.5	89.3	86.5	85.2	65.5	83.7
	$\text{bleu}_{c,\ sub}$	91.3	94.2	97.6	94.6	95.0	97.4	94.2	92.2	88.6	96.4	93.7	93.9	91.7	93.9

Table 5: Error analysis and performance of models for difference class of character mismatch.

Lang	Diacritics Marks
as	◌া, ◌ি, ◌ী, ◌ু, ◌ূ, ◌ৃ, ◌ে, ◌ৈ, ◌ো, ◌ৌ, ◌ং, ◌ঃ, ◌ঁ
bn	◌া, ◌ি, ◌ী, ◌ু, ◌ূ, ◌ৃ, ◌ে, ◌ৈ, ◌ো, ◌ৌ, ◌ং, ◌ঃ, ◌ঁ
gu	◌ા, ◌િ, ◌ી, ◌ુ, ◌ૂ, ◌ૃ, ◌ે, ◌ૈ, ◌ો, ◌ૌ, ◌ૅ, ◌ં, ◌ઃ, ◌ઁ
hi	◌ा, ◌ि, ◌ी, ◌ु, ◌ू, ◌ृ, ◌े, ◌ै, ◌ो, ◌ौ, ◌ॅ, ◌ं, ◌ः, ◌ँ
kn	◌ಾ, ◌ಿ, ◌ೀ, ◌ು, ◌ೂ, ◌ೃ, ◌ೇ, ◌ೈ, ◌ೋ, ◌ೌ, ◌ೆ, ◌ಂ, ◌ಃ
ml	◌ാ, ◌ി, ◌ീ, ◌ു, ◌ൂ, ◌ൃ, ◌െ, ◌േ, ◌ൊ, ◌ൌ, ◌ൂം, ◌ഃ
mr	◌ा, ◌ि, ◌ी, ◌ु, ◌ू, ◌ृ, ◌े, ◌ै, ◌ो, ◌ौ, ◌ॅ, ◌ं, ◌ः, ◌ँ
pa	◌ਾ, ◌ਿ, ◌ੀ, ◌ੁ, ◌ੂ, ◌ੇ, ◌ੈ, ◌ੋ, ◌ੌ, ◌ੂਂ
sd	ˆ, ٘, ٖ, ˜, ö, ُ, ٍ, اَ
si	◌ා, ◌ි, ◌ී, ◌ු, ◌ූ, ◌ෛ, ◌ෙ, ◌ෝ, ◌ෞ, ◌ෟ
ta	◌ா, ◌ி, ◌ீ, ◌ு, ◌ூ, ◌ெ, ◌ே, ◌ொ, ◌ௌ, ◌்
te	◌ా, ◌ి, ◌ీ, ◌ు, ◌ూ, ◌ృ, ◌ె, ◌ే, ◌ొ, ◌ో, ◌ం, ◌ః
ur	(No specific matras listed)

Table 6: Diacritics across different languages.

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. This model was trained on the parallel token corpus. For sentence-level transliteration using this model, we have first broken down the sentence into tokens, performed the transliteration on tokens, and then joined the transliteration to get the transliterated sentence.

For evaluation, we consider character error rate (CER), word error rate (WER), character-level BLEU (BLEU_c), and word-level BLEU (BLEU_w) as performance metrics.

Table 3 summarizes these results, comparing our experimental models (Transformer-based, LLaMA-based, and mT5-base) with the IndicXlit model. Our analysis highlights the strengths and weaknesses of each model across different languages and sentence structures.

The IndicXlit model performs significantly low across all metrics, with an average CER of 0.27 and a high WER of 0.56. The reason behind this low performance is that this model was trained on the word pair which potentially lacking the context. The same romanized word can have different interpretations depending on the context. More over this model is trained mostly on well structured word pair lacking the real life or social medial text annomolies like vowel commission, misspelling, shorten forms etc.

In contrast, IndiXform_{ct} and fine-tuned LLaMA_{fts} demonstrates considerable improvements, achieving an average CER of 0.07, a WER of 0.20 and an average CER of 0.08, a WER of 0.20 respec-

		as	bn	gu	hi	kn	ml	mr	pa	sd	si	ta	te	ur	avg
IndiXform$_{ct}$	**Diact.**	70.3	77.2	92.4	76.6	86.2	94.7	77.7	73.7	81.1	81.7	82.2	78.4	-	81.0
	Char.	84.4	87.7	95.6	89.8	89.9	96.5	88.1	86.0	81.3	95.5	86.6	84.4	78.4	88.0
LLaMA$_{fts}$	**Diact.**	69.0	79.0	92.0	81.1	89.1	94.0	81.3	78.2	79.1	80.9	84.9	82.3	-	82.5
	Char.	83.2	88.4	95.7	90.1	92.1	96.4	90.1	88.1	80.0	94.5	88.7	87.6	80.0	88.8

Table 7: Diacritic Marks analysis. The symbol '-' indicates that the metric is not applicable.

tively. The IndiXform$_{ct}$ with low error rates for Gujarati (gu) at 0.02 (CER) and 0.07 (WER). Similarly, IndiXform model show strong performance on language Hindi (hi) (CER: 0.06, WER: 0.19). The mT5 model achieves the significantly good performance overall, with CER and WER averages of 0.08 and 0.22, respectively.

When comparing BLEU$_c$ scores, the LLaMA model trained for each language, LLaMA$_{fts}$ achieves a high average of 89.4, excelling in Kannada, Marathi, Punjabi, Tamil, and Telugu. In contrast, IndicXlit reports a significantly lower BLEU$_c$ of 63.1, indicating poor transliteration accuracy. The IndiXform$_{ct}$ excelling in some languages such as Bengali (bn) at 96.1 and Sinhala (si) at 95.7. The mT5 model delivers similarly competitive results with an average BLEU$_c$ of 87.1, demonstrating its effectiveness in handling multiple languages.

Word-level BLEU (BLEU$_w$) scores clearly demonstrate the enhanced performance of our experimental models. The LLaMA zero-shot model shows limited effectiveness, achieving an average BLEU$_w$ of only 19.9, highlighting the necessity of fine-tuning. In contrast, the fine-tuned LLaMA (LLaMA$_{fts}$) achieves a remarkable average BLEU$_w$ of 60.1, significantly outperforming the baseline score of 18.0. This improvement illustrates the importance of incorporating sentence-level context for more accurate back-transliteration. Additionally, IndiXform$_{ct}$ shows exceptional performance in specific languages, achieving BLEU$_w$ scores of 60.0 in Assamese (as) and 81.6 in Gujarati (gu). To gain deeper insights into model performance, we conducted detailed error analyses, examining language-specific challenges.

4.1. Error Analysis

We analyze the errors of the two best-performing models, **LLaMA$_{fts}$** and **IndiXform$_{ct}$**, considering several error categories: character count mismatches, diacritic generation errors, sentence length effects, and code-mixing challenges.

Error due to Character Mismatch: We categorize character mismatches into three classes: (i) **Insertion Class**, where the predicted output has more characters than the reference, (ii) **Deletion Class**, where the predicted output has fewer characters, and (iii) **Substitution Class**, where the character count is the same but incorrect characters are substituted. These errors are primarily caused by the insertion, deletion, or substitution of vowels (diacritics). Additionally, numbers, digits, and confusion between similar consonants also contribute to substitution errors. The examples for each case are shown in Table 4. Due to space constraints, we include samples from Hindi and Gujarati, although the analysis covers 13 languages.

We calculate the percentage of character insertions and deletions for each language and their corresponding BLEU scores, as shown in Table 5. On average, IndiXform has an insertion rate of 3.24% and a deletion rate of 4.93%, while LLaMA has rates of 3.77% and 3.73%, respectively.

To assess the impact on performance, we compute the correlation between character error rates and BLEU scores. For IndiXform, the correlation is -0.81 for insertion and -0.96 for deletion, indicating strong negative impacts. For LLaMA, the correlations are -0.44 and -0.93, respectively.

Error due to diacritics : To gain a deeper understanding of the errors, we analyze the models' performance in generating diacritics. Diacritics from different languages are tabulated in Table 6 We separate the diacritics and characters from each sample in both the references and predictions for both models. We then calculate the BLEU scores for diacritics and characters separately, as shown in Table 7.

For IndiXform, the average BLEU score is 81.0 for diacritics and 88.0 for characters. Similarly, LLaMA achieves scores of 82.5 for diacritics and 88.8 for characters. These findings indicate that both models struggle more with generating diacritics than characters, highlighting a key challenge in transliteration accuracy.

Effect of Sentence Length: We have also done the error analysis of all models based on the sentence length effects. The **LLaMA$_{fts}$** model outperformed others on long sentences ($\geq$20 words) with an average BLEU score of 88.8, while **IndiXform$_{ct}$** excelled in Gujarati and Sinhala with BLEU$_c$ scores of 93.8 and 94.8. For short sentences ($\leq$ 5 words), both models showed strong performance (average BLEU$_c$ = 84.4), with **mT5-base** achieving the highest score of 95.6 for Gujarati.

5. Conclusion

This paper presents a comprehensive analysis of sentence-level back-transliteration for 13 Indian languages, addressing challenges posed by informal Romanized text, including irregular spelling, vowel omissions, code-mixing, and contextual ambiguities. By evaluating state-of-the-art models such as LLaMA 3.1 (in zero-shot and fine-tuned modes), mT5, and the proposed IndiXform, we demonstrated the importance of sentence-level context in enhancing transliteration accuracy. Fine-tuned LLaMA and IndiXform consistently outperformed the baseline IndicXlit model, achieving lower character error rates (CER) and word error rates (WER), while mT5 exhibited competitive performance due to its robust multilingual pre-training.

However, challenges remain with code-mixed text, short sentences, and unique character sets, highlighting model-specific weaknesses. The findings emphasize the need for advanced models with better contextual understanding and handling of linguistic diversity. Future research should focus on improved pre-training strategies, data augmentation for informal text, and hybrid approaches, ultimately contributing to more accurate and robust back-transliteration systems for Indian languages.

Limitations

Despite achieving promising results, our study faced several limitations, particularly in terms of data availability and performance in challenging linguistic scenarios. One of the primary challenges was the limited availability of high-quality annotated data for several Indian languages. Many low-resource languages in India have minimal digitized corpora, restricting the scope of comprehensive training and evaluation.

Another significant limitation was the scarcity of data containing code-mixed text, such as that found in SMS, social media, and informal communications. These forms of communication often blend English and native languages, presenting complex transliteration tasks. Due to insufficient examples in the training data, the models struggled to handle code-mixed sentences accurately, frequently misinterpreting or incorrectly transliterating English words within these sentences. Consequently, model performance in this category was noticeably lower than in others.

Additionally, transliteration performance for sentences with long and complex word structures proved to be another area of difficulty. Certain long and morphologically rich words, particularly in highly inflected languages, posed challenges to the models, occasionally leading to errors in output generation. These limitations highlight the need for further data augmentation efforts, particularly in gathering code-mixed and long-form content, to enhance the robustness of transliteration models across diverse linguistic contexts.

6. Bibliographical References

Bajiyo Baiju, Kavya Manohar, Leena G. Pillai, and Elizabeth Sherly. 2025. Romanized to native Malayalam script transliteration using an encoder-decoder framework. In *Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages*, pages 174–178, Abu Dhabi. Association for Computational Linguistics.

Rafael E Banchs, Min Zhang, Xiangyu Duan, Haizhou Li, and A Kumaran. 2015. Report of news 2015 machine transliteration shared task. In *Proceedings of the Fifth Named Entity Workshop*, pages 10–23.

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.

Widanalage Mario Yomal De Mel, Kasun Imesha Wickramasinghe, Nisansa de Silva, and Surangika Dayani Ranathunga. 2025. Sinhala transliteration: A comparative analysis between rule-based and Seq2Seq approaches. In *Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages*, pages 166–173, Abu Dhabi. Association for Computational Linguistics.

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*.

Kanika Gupta, Monojit Choudhury, and Kalika Bali. 2012. Mining hindi-english transliteration pairs from online hindi lyrics. In *LREC*, pages 2459–2465.

Christo Kirov, Cibu Johny, Anna Katanova, Alexander Gutkin, and Brian Roark. 2024. Context-aware transliteration of Romanized South Asian languages. *Computational Linguistics*, 50(2):475–534.

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, Abu Dhabi. Association for Computational Linguistics.

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.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

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, Abu Dhabi. 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.

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

Ruvan Weerasinghe, Isuri Anuradha, and Deshan Sumanathilaka, editors. 2025. *Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages*. Association for Computational Linguistics, Abu Dhabi.

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.