INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

Back Transliteration of Romanized Assamese Social Media Texts

(Corpus, Analysis and Models)

by

Hemanta Baruah

The thesis submitted in partial fulfillment of the requirements for the award of the degree of

Doctor of Philosophy

in the

CENTRE FOR LINGUISTIC SCIENCE AND TECHNOLOGY

Under the supervision of

Prof. Sanasam Ranbir Singh and Prof. Priyankoo Sarmah

October 2025

TH-3777_186155001

TH-3777_186155001

Declaration of Authorship

I, Hemanta Baruah, hereby confirm that:

- The work contained in this thesis is original and has been done by myself under the general supervision of my supervisors.
- This work has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to the authors/researchers by citing them in the text of the thesis and giving their details in the reference.
- Whenever I have quoted from the work of others, the source is always given.

Hemanta Baruah
Research Scholar,
CENTRE FOR LINGUISTIC SCIENCE AND TECHNOLOGY,
Indian Institute of Technology Guwahati,
Guwahati, Assam, INDIA 781039,
hemanta.b@iitg.ac.in, baruahemantaofficial@gmail.com

Place: IIT Guwahati

TH-3777_186155001

TH-3777_186155001

Certificate

This is to certify that the thesis entitled **"Back Transliteration of Romanized Assamese Social Media Texts (Corpus, Analysis and Models)"** being submitted by **Mr. Hemanta Baruah** to the *Centre for Linguistic Science and Technology, Indian Institute of Technology Guwahati*, is a record of bonafide research work under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute. To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Date:
Place: IIT Guwahati

Prof. Sanasam Ranbir Singh
(Coordinating supervisor)
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati, Assam, INDIA 781039,
ranbir@iitg.ac.in

Prof. Priyankoo Sarmah
(Co-supervisor)
Department of Humanities and Social Science,
Indian Institute of Technology Guwahati,
Guwahati, Assam, INDIA 781039,
priyankoo@iitg.ac.in

TH-3777_186155001

TH-3777_186155001

DEDICATED TO
MY BELOVED FAMILY

TH-3777_186155001

TH-3777_186155001

Acknowledgements

It gives me immense pleasure to thank each individual who supported directly or indirectly towards completion of my Ph.D. journey. At first, I would like to thank my supervisors Prof. Sanasam Ranbir Singh and Prof. Priyankoo Sarmah for their exceptional and motivating guidance throughout my Ph.D. journey. Moreover, their dedication towards their duties and research always inspire me. I would always be indebted to them for several thought-provoking ideas, constructive research discussions and cultivating professional work ethics.

I would like to thank my Doctoral Committee members namely, Prof. Sukumar Nandi, Prof. Ashish Anand, and Prof. Bidisha Som for their constructive suggestions towards shaping my research goals as well as the entire thesis. I also want to thank Dr. Ashish Anand for the various research discussion and his valuable suggestions. Furthermore, my sincere thanks to Prof. Sanasam Ranbir Singh, the Head of the Centre for Linguistic Science and Technology and other faculty members for their direct and indirect support throughout my PhD process.

I humbly thank to Mr. Souvik Chowdhury, Mr. Santanu Majumdar, Mr. Raktajit Pathak, Mr. Nanu Alan Kachari, Mr. Bhriguraj Borah, and all institute's staffs for all the helps I borrowed towards making my journey smooth and productive. I specially thank Mr. Santanu Majumdar for his extreme dedications towards managing efficient computing facilities at the Centre. My thesis would not have been completed without his timely support. Furthermore, I would like to thank IIT Guwahati administration for providing on-campus hostel facility. From the core of my heart, I would like to thank the administrative staffs from Students Affairs section, Academic Section, Research and Development section, Hostel caretakers, mess staffs, canteen staffs, security personals, and housekeeping staffs for making my stay memorable and smooth.

I am especially grateful to my wife, Juri Kalita, my dear friend, Bhaswati Saikia, my brother-in-law Kangkan Kalita, and my sisters, Barnali Baruah and Deepsikha Baruah, for their invaluable support in creating the initial phase of transliteration annotation.

Having good friends is always a blessing. Fortunately, I have a large set of good and close friends with whom I have spent very quality time. I am privileged to mention Dr. Kamal Narayan Baruah, Dr. Sampreet Kalita, Nayanjyoti Sarma, Omesh Yengkhom, Arabindu Debbarma, Nayanjyoti Das, Md. Rakesh Hussain, Kalyan Boro, Rupam Dey, Sapnam Takuria, Akhil Kalita, Parashmani Sharma, Anil Kumar Singha, Bhaswati Saikia,

TH-3777_186155001

Dharitri Chamuah, Pallav Tamuly, Rinku Das, Mrinal Sharma, Rukmajit Baruah, Nilastha Goswami, Gokul Hazarika, Kongkon Bhuyan, Suranjan Baruah, Debasish Baruah, Rashmi Rekha Saikia, Abhijeet Bora, Parag Baruah and Parthajit Baruah as longtime peers and colleagues who supported my entire journey in several perspectives. I had the privilege of having a very helpful and supportive seniors, namely Dr. Durgesh Kumar, Rajib Chakrabartty, Dr. Akash Anil, Dr. Anasua Mitra, Dr. Ujjwal Biswas, Dr. Dhrubajyoti Pathak, Dr. Alakesh Kalita, Dr. Jennil Thiyam, Dr. Bornali Phukon and Dr. Loitongbam Gyanendro Singh. My stay at IIT Guwahati was made more pleasant by having many good memories with people from OSINT lab like Akash Anil, Neelakshi Sarma, Durgesh Kumar, Mala Das, Deepen Naorem, Loitongbam Gyanendro Singh, Roshan Singh, Lenin Laitonjam, Anupam Chaudhury, Jubanjan Dhar, Anurag Kushwaha, Rajdeep Borgohain, Roshan Singh, Pankaj Choudhury, Tonmoya Sarmah, Anasua Mitra, Rajib Chakrabartty, Soumyadeep Jana, Saurabh Kumar, Mridul Jyoti Roy, Tarik Mohammad Saikia, Okram Jimmy Singh, Jibon Kumar Borgoyary, and many more.

I would also like to thank the people from CLST lab like Mr. Pankaj Choudhury, Mr. Chaitanya Kirti, Mrs. Joyshree Chakraborty, Mrs. Moumita Pakrashi, Mrs. Supriya Bordoloi, Mr. Telem Joyson Singh, Mrs. Maisang Kamei Salice, Mr. Deepen Naorem, Mrs. Emily Thomas, Mrs. Meghali Deka, and many more.

My family members have played a crucial role throughout my academic career. Their constant support, love, motivation, and faith in me help me bounce back at different phases of my life. From the core of my heart, I would like to thank my father Achyut Baruah, mother Kaushalya Baruah, grand mother Labanya Baruah, wife Juri Kalita, and my brothers Diganta Baruah, Dharmendra Baruah, Jitendra Baruah, Lokendra Baruah, father-in-law Amal Kalita, mother-in-law Sudari Priya Das, brother-in-law Kangkan Kalita, sisters Kalpana Kalita, Loni Baruah, Mila baa, sister-in-laws Anuradha Baruah, Juri Baruah, Asomi Baruah, my cute little brothers Abhigyan Kalita, Kalyan Kalita, Addik Baruah and sisters Rajshree Baruah, Priyanshee Baruah and Hanshikha Baruah for showering immense love and moral support. Finally, I would like to thank all the friends, family, relatives, well wishers, and everyone who has supported my academic career.

TH-3777_186155001

Abstract

Natural Language Processing (NLP) research has focused primarily on resource-rich languages, leaving low-resource languages underrepresented. Among these is Assamese, an Indo-Aryan language spoken by millions in northeast India, which faces unique challenges due to its diverse linguistic features and lack of standardized resources. This thesis addresses back-transliteration for Romanized Assamese text, commonly found on popular social media platforms like Facebook, YouTube, and Twitter (X), where informal, noisy, and code-mixed content poses significant challenges. Transliteration, the process of converting text from one script to another while preserving its phonetic representation, whereas back-transliteration involves converting native text, represented in a non-native transliterated script, back to its original native script. Both processes have gained significant prominence with the rise of multilingual social media content, particularly in multilingual countries like India. Assamese text transliteration encounters additional complexities, including inconsistent romanization, phonetic/transliteration variations, and orthographic diversity. This thesis performs a comprehensive analysis of transliteration variations at both the grapheme and phoneme levels to address these challenges effectively. A newly developed dataset comprising 60,312 sentence pairs and 65,614 word pairs from diverse social media platforms underpins this work, enabling the benchmarking of various transliteration models spanning statistical, neural, and pre-trained transformer architectures. Focusing on back-transliteration from Romanized Assamese to its native script, this research also evaluates word-level versus sentence-level transliteration performance using state-of-the-art large language models. The findings highlight the impact of phonetic diversity, orthographic inconsistencies, and sentence level context on transliteration accuracy. Furthermore, the thesis explores how back-transliteration enhances downstream NLP tasks like sentiment analysis, demonstrating its critical role in advancing Assamese language understanding. By contributing novel datasets, methodologies, and insights, this research lays a robust foundation for addressing transliteration challenges in low-resource languages and fostering further innovation in the field.

TH-3777_186155001

TH-3777_186155001

Contents

Declaration of Authorship iii

Certificate v

Acknowledgements ix

Abstract xi

List of Figures xvii

List of Tables xix

Abbreviations xxi

1 Introduction 1
- 1.1 Overview . . . 1
- 1.2 Challenges . . . 5
- 1.3 Research Objectives and Scope of the Thesis . . . 6
- 1.4 Contributions Made in the Thesis . . . 8
  - 1.4.1 Novel Dataset Generation: . . . 8
  - 1.4.2 Analysis of Transliteration Characteristics and Variations: . . . 8
  - 1.4.3 Study on the Adaptability of Transliteration Models to a Canonical Dataset vs. a Noisy Social Media Dataset: . . . 9
  - 1.4.4 Study on the Importance of Context in Transliteration: . . . 10
  - 1.4.5 Study on the Importance of Transliteration for Downstream Applications: . . . 10
- 1.5 Organization of the Thesis . . . 11

2 Literature Survey 13
- 2.1 Analysis of Various Transliteration Methodologies . . . 14
  - 2.1.1 Traditional Machine Learning based methods . . . 14
  - 2.1.2 Traditional Machine Learning based methods for Indian Languages 17
  - 2.1.3 Neural Network based methods . . . 20

TH-3777_186155001

2.2 Brief Summary . . . 23

3 Novel Dataset Generation **25**
3.1 Introduction . . . 25
3.2 Related work . . . 27
3.3 Dataset . . . 28
3.4 Performance Metrics Used for Evaluation . . . 33
3.5 Experimental Setups . . . 36
3.6 Result and Analysis . . . 39
3.7 Summary and Future Work . . . 43

4 Analysis of Transliteration Characteristics and Variations **45**
4.1 Introduction . . . 46
4.2 Related Work . . . 47
4.2.1 Characteristic Analysis of Transliterated Social Media Text . . . 47
4.2.2 Analysis of phonetic/spelling/lexical variations in the transliterated social media text . . . 48
4.2.3 Analysis of code-mixing issues related to the romanized noisy informal social media text . . . 50
4.3 Dataset . . . 53
4.4 Transliteration variations in Assamese . . . 54
4.4.1 Transliteration variation analysis in Graphemic level . . . 54
4.4.2 Transliteration variation analysis in Phonemic level . . . 61
4.5 Comparison between Social Media Transliteration and Some Standard Assamese Transliteration Schemes . . . 68
4.6 Back Transliteration Performance Evaluation . . . 69
4.6.1 Setup for SMT Model . . . 70
4.6.2 Setup for NMT Model . . . 71
4.6.3 Error Analysis from Transliteration Output . . . 72
4.6.4 Analysis between the outputs of three different transliteration models with the same Roman input . . . 73
4.6.5 In-depth error analysis from transliteration output . . . 73
4.6.6 Summary of the three transliteration model outputs that correspond to the transliteration variations . . . 77
4.7 Attention analysis on transliterated Assamese social media data . . . 81
4.7.1 Observation on Assamese Vowels . . . 82
4.7.2 Observation on Vowel Diacritics . . . 83
4.7.3 Observation on Consonants and two Special Symbols . . . 83
4.8 Summary and Future work . . . 86

5 Study on the Adaptability of Transliteration Models to a Canonical Dataset vs. a Noisy Social Media Dataset **89**
5.1 Introduction . . . 90
5.2 Related Studies . . . 91
5.3 Dataset . . . 92

TH-3777_186155001

5.4 Experimental Setup . . . 93
5.5 Results and Discussion . . . 96
5.5.1 Error Analysis . . . 99
5.5.2 Summary and Future work . . . 100

6 Study on the Importance of Context in Transliteration 101
6.1 Introduction . . . 102
6.2 Related Work . . . 104
6.3 Dataset . . . 104
6.3.1 Dataset Statistics . . . 105
6.3.2 Dataset Statistics for Cross-Domain Adaptation Experiment . . . 106
6.4 Experimental Setup . . . 106
6.5 Results and Discussion . . . 110
6.5.1 Analysis of Word-Level vs. Sentence-Level Transliteration Models 110
6.5.2 Cross-Domain Adaptation Analysis . . . 112
6.6 Summary and Future Work . . . 114

7 Study on the Importance of Transliteration for Downstream Applications 117
7.1 Introduction . . . 117
7.2 Related Work . . . 118
7.3 Dataset . . . 119
7.3.1 Dataset Statistics . . . 120
7.3.2 Example Sentences . . . 120
7.3.3 Evaluation Metrics . . . 120
7.4 Experimental Setup . . . 121
7.5 Results and Discussion . . . 123
7.6 Error Analysis . . . 123
7.7 Summary and Future Work . . . 125

8 Conclusion and Future Work 127
8.1 Conclusion . . . 127
8.2 Limitations and Future Works . . . 128

Bibliography 129

Publications 149

TH-3777_186155001

TH-3777_186155001

List of Figures

2.1 General transliteration framework 14
2.2 Phoneme-based transliteration 16
2.3 Grapheme-based transliteration 16

3.1 Distributions of Roman Transliteration Variations for Native Assamese Words.(Semi-log Scale) 30
3.2 Distributions of Back Transliterated Native Assamese words represented by the Roman Words.(Semi-log Scale) 31
3.3 A snapshot of the annotation tool 32

4.1 Distributions of Roman characters to represent Assamese vowel graphemes. 55
4.2 Distribution of Roman characters to represent Assamese vowel diacritics. 56
4.3 Distributions of Roman characters to represent Assamese consonant graphemes 60
4.4 Distribution of Roman characters to represent Assamese vowel phonemes. 62
4.5 Distributions of Roman characters to represent Assamese consonant phonemes. 66
4.6 Venn diagram of three different transliteration model outputs for the same Roman inputs 74
4.7 Sunburst View of different types of errors observed from the experimental results. 75
4.8 Attention weight analysis on Assamese vowels. 83
4.9 Attention weight analysis of Assamese vowel diacritics. 84
4.10 Attention weight analysis of Assamese consonants. 84

TH-3777_186155001

TH-3777_186155001

List of Tables

3.1 Statistics of the collected dataset from three major social media sources along with the duration of data collection . . . 29
3.2 Two examples of both Roman and native variations along with the frequencies present in our dataset . . . 31
3.3 Transliteration results in terms of WER, CER, substitution, insertion, deletion, and BLEU (up to 4-gram) for all 10 setups. Best WER, CER, and BLEU are highlighted in **bold**. . . . 38
3.4 Comparison between the outputs of ten different transliteration model setups with the same Roman input (output of the setups that match with the ground truths are highlighted in blue) . . . 41

4.1 Dataset Statistics for the transliteration characteristics analysis experiment 54
4.2 Roman transliteration variations for Assamese vowel graphemes and diacritics . . . 55
4.3 Roman transliteration variations for Assamese consonant graphemes . . . 59
4.4 Roman transliteration variations for Assamese vowel phonemes . . . 62
4.5 Roman Transliteration Variations for Assamese Consonant Phonemes . . 64
4.6 Similarity Analysis Between Assamese Social Media Transliteration Against the Six Transliteration Standards . . . 69
4.7 Evaluation result of three baseline transliteration models on unique and whole transliteration pairs . . . 72
4.8 Comparison between the three different transliteration model outputs with the same Roman input sample . . . 74
4.9 Summary of transliteration variations from three different model outputs 81

5.1 Training, Validation and Testing dataset statistics (All transliteration pairs are word splitted into characters) . . . 93
5.2 Transliteration result in terms of Accuracy@1, Accuracy@4, Character Accuracy, Mean F-score, MRR and BLEU4 score for all the setups (Experimental setups with the highest and lowest accuracies(top-1) are highlighted in red and blue, respectively, while the accuracy of the best-performing setup is highlighted in **bold**) . . . 94
5.3 In-depth error analysis on transliteration output from setup 1, setup 3, and setup 5 . . . 99

6.1 Sentence-Level and Word-Level Dataset Statistics . . . 105
6.2 Dataset Splits and Sentence Pair Distribution . . . 107

TH-3777_186155001

6.3 Comparison of Word-level Vs Sentence-level Transliteration Models . . . 111
6.4 Cross Domain Adaptation Analysis 113

7.1 Sentiment Analysis Dataset Statistics 120
7.2 Examples from the Sentiment Analysis Dataset: Roman-Native Parallel Sentences with English Glosses 122
7.3 Performance of three state-of-the-art classification methods on our sentiment analysis dataset, separately for Romanized and Native Assamese text. Both micro and weighted scores are reported for Precision, Recall, and F1-Score. Highest and lowest performing Model–Embedding combinations are highlighted in blue and red, respectively. 123
7.4 Examples showing differences in sentiment prediction between Romanized and Native Assamese text. The native predictions align more closely with the ground truth due to normalization of variant spellings (e.g., *bohut/bhut/bahut/bhot* → বহুত, *moja/mojja/mojaa/mojjah* → মজা). Sentiment labels are denoted as: 1 = Positive, −1 = Negative and 0 = Neutral. . . 124

TH-3777_186155001

Abbreviations

NLP	Natural Language Processing
MT	Machine Translation
NER	Named Entity Recognition
OOV	Out-of-Vocabulary
LMs	Language Models
LLMs	Large Language Models
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
BiLSTM	Bidirectional Long Short-Term Memory
FIRE	Forum for Information Retrieva
T5	Text-to-Text Transfer Transformer
mT5	Multilingual Text-to-Text Transfer Transformer
ByT5	Byte-to-byte Text-to-Text Transfer Transformer
MLP	Multilayer Perceptron
BERT	Bidirectional Encoder Representations from Transformers
mBERT	Multilingual Bidirectional Encoder Representations from Transformers
BART	Bidirectional and Auto-Regressive Transformers
mBART	Multilingual Bidirectional and Auto-Regressive Transformers
NLLB	No Language Left Behind
M2M	Many To Many
LLaMA	Large Language Model Meta AI
WER	Word Error Rate
CER	Character Error Rate
BLEU	Bilingual Evaluation Understudy
CHRF	Character F-score
NLU	Natural Language Understanding

TH-3777_186155001

TH-3777_186155001

Chapter 1

Introduction

1.1 Overview

Transliteration refers to the process of converting text from one script to another while preserving the phonetic characteristics of the original language. Automatic Machine Transliteration automates this task, enabling computers to convert text across scripts while retaining phonetic and linguistic properties. Based on the direction of conversion, transliteration is classified into *forward transliteration* (or simply transliteration) and *back-transliteration* (or reverse transliteration). Forward transliteration transforms text from its native script into another script while maintaining phonetic resemblance [1]. Conversely, back-transliteration reverses this process by restoring the original script from its forward-transliterated form [2]. A well-known example of forward transliteration is romanization, where words from a native language are represented using the Roman (Latin) script of English language. This study focuses on back-transliteration of romanized Assamese text from social media. Throughout this thesis, the terms "**back-transliteration**" and "**reverse transliteration**" are used interchangeably. Furthermore, since our primary focus is on romanized Assamese text, we also use "**transliteration**" and "**romanization**" synonymously in this context.

Machine transliteration was initially explored in conjunction with Machine Translation (MT), primarily to handle named entities (NE), loanwords, and out-of-vocabulary (OOV) terms that lack direct translation equivalents across languages. However, with the increasing prevalence of social media and user-generated content, phonetic transliterations have become more widespread, posing new challenges for natural language processing (NLP) applications. This issue is particularly significant in multilingual societies like India, where a considerable portion of digital content is written in the Roman script.

TH-3777_186155001

The challenge is even greater for low-resource languages like Assamese, where informal transliterations (or romanization) on social media often lack consistency due to the absence of a standardized romanization scheme.

Assamese, an eastern Indo-Aryan language, is spoken by almost 15.3 million speakers as their first language [3]. It is also one of the 22 scheduled languages of India. Assamese orthography is an Indic script consisting of 41 consonant and 11 vowel graphemes, which correspond to 8 vowel and 23 consonant phoneme sounds, respectively [4]. The vowel phonemes in Assamese include / i, e, u, ʊ, o, ɛ, ɔ, ɑ /, while the consonant phonemes are / p, p^h, b, b^h, t, t^h, d, d^h, k, k^h, g, g^h, ɹ, l, s, z, x, h, m, n, ŋ, w / and /j/. Among the vowel phonemes, /i/, /e/ and /ɛ/ are front vowels, and the remaining are the back vowels. Each consonant phoneme represents a single sound with an inherent vowel /ɔ/ attached at the end.

While Assamese has 52 letters to represent 31 phoneme sounds, English has only 26 letters (5 vowels and 21 consonants) to produce 44 phoneme sounds [5]. This discrepancy leads to variations when transliterating Assamese words into the Roman script, especially in the absence of standardized transliteration rules. Some key orthographic differences between Assamese and English orthography are outlined below:

- **Vowel Graphemes and Diacritics:** Assamese orthography, like other Indic orthographies, uses both vowel graphemes and diacritics[1] of those graphemes. The vowel graphemes, such as <ই> are used when the vowel occurs without any consonant onset in the syllable. However, when the vowel has a preceding onset, it appears as a diacritic. For example, when the vowel grapheme <ই> appears with an onset consonant (say <ত>), the diacritic <ি> for the vowel <ই> is used to form <তি>, i.e. <ti>.

- **Vowel Length Distinctions:** Assamese written form differentiates between short and long vowels. However, this distinction is lost in spoken Assamese. Consequently, in transliteration, both short <ই> and long <ঈ> vowels are represented as <i>.

- **Retroflex and Dental Stops:** Assamese has two categories of stop consonants: retroflex and dental. In writing, these are distinct, but in spoken Assamese, both sets are produced as alveolars, making them indistinguishable. As a result, the retroflex series < ট, ঠ, ড, ঢ, ণ > and the dental series < ত, থ, দ, ধ, ন > are both romanized as < t, th, d, dh, n >.

[1] Also termed as vowel matras in many Indian languages

TH-3777_186155001

The lack of a standardized romanization guideline means that social media users do not adhere to a common standard when writing Assamese in the Roman script. Instead, they rely on their own linguistic intuition, leading to significant variations in transliterated text. Beyond linguistic challenges, social media introduces additional complexities such as code-mixing, code-switching, elongated word forms to convey emotions, and shortened word forms achieved by omitting vowels. Additionally, orthographic differences between English and Assamese contribute to a single Assamese word having multiple romanized variations and vice versa. This arises from the one-to-many and many-to-one character mappings between the two orthographies, leading to numerous transliteration variations and making the task of back-transliteration particularly challenging.

While considerable efforts have been made to develop transliteration datasets for various languages, many low-resource languages still lack comprehensive resources. Existing datasets primarily focus on named entities and out-of-vocabulary (OOV) terms ([6], [7], [8]), often overlooking the linguistic diversity present in user-generated content on platforms like Facebook, Twitter, and YouTube, where native-language words frequently appear in transliterated form. Moreover, most existing transliteration datasets emphasize forward transliteration, where native-language words are carefully transliterated into non-native scripts with human annotation. In contrast, back-transliteration datasets convert transliterated text written in a non-native script back into its native form [2]. Back-transliteration datasets are crucial as they capture transliteration variations, offering a more flexible approach. In the forward transliteration direction, annotators tend to perform precise annotation, which often lacks the natural transliteration variations observed in back transliteration datasets.

In majority of the studies, while generating the forward transliterated dataset, native language script texts are primarily sourced from Wikipedia. Annotators are tasked with producing the Romanized version of these native texts, relying on their own phonetic judgment since there are no established transliteration or romanization guidelines present for most of the indic languages. For instance, the Assamese term "শুভেচ্ছা" [pronunciation: /xub^{h}essɑ/, meaning: "**good wishes**"], only three different transliteration variations: "**xubhessa**", "**xubhescha**" and "**shubhessa**" as adopted by the annotators observed in the forward transliterated Aksharantar dataset [8]. In contrast, typically in social media different variations are observed [9] for the same native word "শুভেচ্ছা" written in Roman script, such as "**hubhesaa**", "**xuvesa**", "**huvasha**", "**huvesha**", "**subhecha**", "**huveshya**", "**subhesa**", "**xuvesha**", "**xuvessa**", "**huvasa**", "**khuvessa**", "**kuvessa**", "**khuveswa**", "**xubheisha**" and "**shubhessa**", among others, reflecting real instances from diverse users across three different social media platforms. This demonstrates the scale and challenges associated with the reverse or back-transliterated dataset compared

TH-3777_186155001

to traditional forward transliterated datasets. In the backward transliteration dataset exist in social media, users are free to express their native words using Roman script, and annotators are asked to identify the correct native word based on the context and transliterate it back to the native word in Assamese script, resulting in a richer and more diverse dataset.

From the nature of the dataset used, we may further group machine transliteration studies into two types - canonical and non-canonical parallel dataset. A canonical parallel dataset consists of legitimate text in a language's native script, transliterated into another orthography, typically by language experts, with minimal variation. In contrast, a non-canonical parallel dataset involves noisy transliterated text from unrestricted platforms like social media, which is back-transliterated into the native script by language experts. State-of-the-art multilingual Indic transliteration models, predominantly trained on canonical datasets, need to be evaluated on noisy non-canonical datasets to assess their global applicability. Similarly, transliteration models trained on noisy social media data should also be tested on clean canonical datasets to ensure their robustness. Without such cross-evaluation, the effectiveness and adaptability of machine transliteration models across different data conditions remain uncertain.

Most existing transliteration models operate at the word level, often neglecting sentence-level contextual information. However, transliteration is highly context-dependent. For example, the Romanized Assamese word "**aasa**" can mean either "আশা"(meaning: "**hope**") or "আছা" (meaning: "**you are there**"), depending on the context. Similarly, in some context the Assamese word "বল" may be romanized as "**ball**" to indicate "**playing ball**" or "**bal**" to indicate "**force**"., illustrating the limitations of word-level transliteration approaches that fail to account for sentence-level meaning. A comprehensive comparison between word-level and sentence-level transliteration approaches remains largely unexplored in the existing literature.

Despite the growing interest in reverse transliteration—converting romanized text back into its native script—its impact on downstream NLP tasks remains underexplored. While forward transliteration has been widely studied, the role of reverse transliteration in improving NLP applications is not well understood. Without investigating its broader implications, the true utility of reverse transliteration in real-world NLP applications remains uncertain.

TH-3777_186155001

1.2 Challenges

Existing transliteration research predominantly focuses on high-resource languages and structured text corpora, with limited attention given to handling noisy, Romanized social media text, particularly for low-resource Indic languages. While early studies applied statistical approaches to named entity transliteration [2, 10, 11], recent advancements have leveraged character-based neural methods [12, 13, 14, 15]. Significant contributions in Indic language transliteration include *Brahmi-Net* [16], the Dakshina dataset [6], Aksharantar [8], and evaluations on sentence-level transliteration [17]. However, despite these efforts, critical gaps remain in addressing the challenges of reverse transliteration—converting noisy, Romanized social media text back into its native Indic script. We can broadly classify the limitations and challenges in the transliteration of Assamese social media texts into two main categories.

1. **Linguistic Challenges:**
    - **Limited back-transliteration corpus and bias toward formal settings:** Resources for back-transliteration in noisy, Romanized social media text are largely unavailable. Existing transliteration datasets focus on formal, structured text, overlooking the unique challenges of informal and user-generated content.
    - **Focus on high-resource languages:** Most transliteration research has been directed toward high-resource languages, neglecting low-resource languages like Assamese, which lack large-scale annotated datasets for back-transliteration.
    - **Orthographic and phonetic inconsistencies:** Romanized Assamese words exhibit high variability due to one-to-many and many-to-one mappings between English and Indic scripts. Moreover, user spellings often reflect pronunciation rather than standardized orthographic rules, leading to significant transliteration ambiguity.
    - **Grapheme vs. phoneme-based transliteration:** Back-transliteration requires distinguishing between grapheme-based mappings (direct script conversion) and phoneme-based mappings (sound-based conversion), which can lead to inaccuracies if not modeled properly.
    - **Context-dependent transliteration:** Some words require different transliterations depending on their usage in a sentence. However, informal social media text often lacks sufficient context, making it challenging for models to resolve transliteration ambiguities accurately.

TH-3777_186155001

- **Dialectal influence and regional variation:** Assamese dialects introduce multiple valid transliterations for a single word, making rule-based systems less effective. This variation is especially problematic for low-resource settings where dialectal data is sparse.

2. **Social Media-Specific Challenges:**

- **Noisy, inconsistent romanized text:** Social media users employ highly inconsistent romanization styles, often mixing phonetic approximations, informal spelling variations, and typographical errors.
- **Code-mixing and code-switching:** Frequent alternation between Assamese, English, and sometimes Hindi complicates transliteration, as the same Romanized word can belong to multiple languages.
- **Elongated and truncated word forms:** Users extend words (e.g., "**bohhuuut**" instead of "**bohut**" for emphasis) or shorten them by omitting vowels/consonants (e.g., "**tmr**" for "**tomar**"), increasing transliteration difficulty.
- **Challenges in existing multilingual Indic transliteration models:** Most current models, such as Brahmi-Net [16], IndicXlit [8], focus on transliteration from native scripts to Romanized text, lacking robust mechanisms to handle "**noisy back-transliteration**" of social media text.
- **Lack of sentence-level context modeling:** While transliteration models typically operate at the word level, social media text often requires "**context-aware transliteration**" to disambiguate words with multiple valid back-transliterations.
- **Impact on downstream NLP applications:** Inaccurate back transliteration can negatively affect sentiment analysis, named entity recognition, and other downstream NLP tasks, highlighting the need for models that improve back-transliteration quality while preserving meaning.

1.3 Research Objectives and Scope of the Thesis

Building on the research gaps and challenges outlined earlier, this thesis sets out to address the following key objectives:

1. **Novel Dataset Generation:** Given the scarcity of back-transliteration datasets for low-resource social media text, this study aims to create a novel dataset to

TH-3777_186155001

bridge this gap. The dataset will be used to evaluate state-of-the-art transliteration models, analyzing their performance in handling noisy, Romanized social media text.

2. **Analysis of Transliteration Characteristics and Variations:** Due to the absence of standardized Romanization guidelines for many Indic languages, combined with orthographic and phonetic differences between English and Indic scripts, transliteration exhibits significant variations. This research aims to conduct a detailed analysis of these variations, examining their implications for transliteration accuracy.

3. **Study on the Adaptability of Transliteration Models to a Canonical Dataset vs. a Noisy Social Media Dataset:** Existing transliteration models are predominantly trained on clean, forward-transliterated data sourced from Wikipedia, named entities, and parallel translation corpora. This study evaluates how these models perform on noisy, back-transliterated social media data, highlighting the challenges posed by informal, user-generated text.

4. **Study on the Importance of Context in Transliteration:** Given that transliteration, particularly reverse transliteration, is highly context-dependent, this research investigates the effectiveness of context-aware, sentence-level models compared to context-less, word-level models in handling Romanized Assamese social media text.

5. **Study on the Importance of Transliteration for Downstream Applications:** This study examines how reverse transliteration influences downstream NLP applications, such as sentiment analysis, by comparing the performance of models on Romanized Assamese text versus its back-transliterated native script counterpart.

Scope of the Thesis:

This thesis focuses on the challenges and solutions related to back-transliteration of Romanized Assamese social media text. It investigates the performance of state-of-the-art transliteration models on noisy, informal text, explores linguistic variations at the grapheme and phoneme levels, and examines the role of context in transliteration tasks. The study also evaluates the impact of reverse transliteration on downstream NLP tasks, such as sentiment analysis, with a specific focus on Assamese. The findings aim to provide insights into how transliteration techniques can be adapted to handle the informal, diverse nature of social media text for low-resource languages like Assamese.

TH-3777_186155001

1.4 Contributions Made in the Thesis

This thesis presents the following significant contributions by addressing the identified gap in research and fulfilling the designated research objectives in terms of research questions.

1.4.1 Novel Dataset Generation:

Majority of the earlier transliteration studies have focused on high-resource languages like English, with a primary emphasis on forward transliteration (from native script to Roman script) and named entities in regular text, leaving the reverse transliteration direction (from Roman script back to native form) largely unexplored. Additionally, while many studies have addressed transliteration in standard texts, few have tackled the specific challenges posed by social media, where informal language, abbreviations, and context-dependent nuances are prevalent. To fill these gaps, we created a novel back-transliteration dataset, "**AssameseBackTranslit**", consisting of 60,312 sentence pairs and 65,614 unique word pairs. This dataset captures the transliteration variations of Assamese text from Roman script to its native form, with a particular focus on social media text, which often features non-standard spellings and informal language. By providing this dataset, we aim to address the scarcity of resources for reverse transliteration in Assamese, particularly in the context of social media. In addition to creating the dataset, we evaluated ten state-of-the-art transliteration methods, including statistical models such as a joint n-gram-based string transduction system and a phrase-based statistical transliteration model, along with advanced neural network models like BiLSTM with attention and a neural transformer model. We also fine-tuned three pre-trained models, including a multilingual transliteration model and two pre-trained sequence-to-sequence large language models. The results demonstrated that the neural transformer model outperformed the others, achieving the lowest Word Error Rate (WER) and Character Error Rate (CER), and the highest BLEU score of 55.05, 19.44, and 69.15, respectively. These findings highlight the effectiveness of modern neural models in handling reverse transliteration for low-resource languages, especially in the complex domain of social media.

1.4.2 Analysis of Transliteration Characteristics and Variations:

Recent reports, such as those by KPMG[2] and Statista[3], have highlighted the rapid growth of Indian language content on the internet, with projections suggesting that nearly 75%

[2] KPMG
[3] Statista

TH-3777_186155001

of internet users in India will contribute content in Indian languages by 2021. While some of this content is in native Indian scripts, a significant portion is transliterated using Roman orthography. This shift is largely due to the lack of technological support for native script input across various platforms. As a result, reverse transliteration—the process of converting Romanized content back to native scripts—has become a critical task for text processing, information retrieval, and natural language processing (NLP). However, the lack of standardized transliteration conventions and the noise introduced by informal social media language make reverse transliteration particularly challenging. To address these issues, this work examines the transliteration characteristics of Romanized Assamese text gathered from three major social media platforms: Twitter(currently X), Facebook, and YouTube. The study also evaluates three basic state-of-the-art reverse transliteration methods to better understand the nuances of Assamese transliteration in the social media domain. Our experiments utilize the in-house back-transliteration dataset, "**AssameseBackTranslit**", with approximately 5,000 posts and comments randomly selected from each platforms, focusing on Assamese content in either native or Romanized script.

1.4.3 Study on the Adaptability of Transliteration Models to a Canonical Dataset vs. a Noisy Social Media Dataset:

Most transliteration models have been developed for clean, standard text, often focusing on forward transliteration, and are generally trained on well-structured data. However, social media text presents unique challenges due to its noisy, informal nature, including the use of slang, abbreviations, and non-standard spellings, which makes it difficult for conventional models to perform effectively. Additionally, existing multilingual transliteration models typically struggle with such noise, as they are often trained on cleaner, more structured datasets. Given these challenges, there is a critical need to evaluate both monolingual and multilingual word-level transliteration models on Assamese text, specifically targeting both canonical and noisy social media data. This evaluation becomes crucial in order to determine the most effective methods for handling the complexities of social media text while ensuring robust performance across different types of data. We assessed the performance of the state-of-the-art multilingual transliteration model, IndicXlit [8], on a noisy test set derived from platforms like Twitter, Facebook, and YouTube, and compared it with a monolingual Assamese model trained on noisy social media data using a clean canonical test set. Our experiments involved a series of fifteen distinct setups, which revealed that models trained exclusively on clean data struggled to handle the noise inherent in social media, while models trained on noisy data performed

TH-3777_186155001

poorly on clean datasets. These findings underscore the importance of integrating both noisy and clean datasets to create a more robust transliteration model capable of performing well in real-world, noisy environments, particularly for Assamese social media text.

1.4.4 Study on the Importance of Context in Transliteration:

This study explores the role of context in reverse transliteration, or deromanization, of code-mixed Assamese social media text, demonstrating that context significantly impacts transliteration accuracy. For instance, the romanized term "**gai**" could mean "গাই" (cow) or "গৈ" (goes), depending on its sentence context. To address this, nine pre-trained large language models (LLMs) were fine-tuned and evaluated on both word-level and sentence-level transliteration tasks. The results showed that sentence-level transliteration, which leverages contextual information, consistently outperformed word-level approaches that lack such awareness. Additionally, a cross-domain adaptation experiment was conducted using datasets from YouTube, Facebook, and Twitter (now X) to assess how well models trained on one platform generalize to others. Models trained on Facebook and YouTube data achieved the best performance when tested on Twitter, highlighting the importance of diverse training data for improving model robustness. The "AssameseBackTranslit" dataset was used to evaluate both word-level and sentence-level models, with the sentence-level task comprising 59,783 unique sentence pairs split into training, development, and test sets. The study also employed different cross-domain training configurations to analyze transliteration adaptability across platforms. By focusing on context-aware sentence-level transliteration and cross-domain evaluation, this research underscores the necessity of incorporating contextual information for more accurate transliteration in noisy, code-mixed data, contributing to advancements in multilingual, low-resource NLP.

1.4.5 Study on the Importance of Transliteration for Downstream Applications:

To evaluate the effectiveness of reverse transliteration—converting romanized text back into its native script—we employed Sentiment Analysis as an extrinsic evaluation task. We created a sentiment analysis dataset comprising 15,076 parallel sentences in both romanized and native Assamese script, sourced equally from Facebook, YouTube, and Twitter (now X). Each sentence was annotated with Positive, Negative, or Neutral labels by nine annotators, with the final sentiment determined by majority voting. Using an

TH-3777_186155001

80-20 train-test split, we trained three sentiment classification models—IndicBERT v2, BERT-multilingual, and IndiSocialFT—and compared their performance on romanized versus back-transliterated native text. IndicBERT v2 achieved the highest accuracy, and models trained on back-transliterated native text consistently outperformed those trained on noisy romanized text. These results highlight the importance of reverse transliteration in improving downstream NLP tasks like sentiment analysis by providing cleaner and more structured input for better model performance.

1.5 Organization of the Thesis

The organization of this thesis is comprised of several distinct chapters, each of which serves a specific purpose in supporting the overall argument and structure of the work. The chapters are as follows:

- **Chapter 1 Introduction**: This chapter begins with an overview of the Assamese language, highlighting the differences between Assamese and English orthography and explaining the prevalence of back transliteration challenges in the noisy, social media-specific context of low-resource languages like Assamese in the Indian subcontinent. It is followed by the challenges involved, and the motivation of the thesis work. The research objective of this thesis work is formally discussed, followed by an overview of contributions made.
- **Chapter 2 Literature Survey**: This chapter begins with a brief review of previous studies on transliteration challenges in general, with a particular focus on the back transliteration problem in noisy, social media-specific environment.
- **Chapter 3 Novel Dataset Generation:** This chapter introduces "**AssameseBackTranslit**", a novel back-transliteration dataset with 60,312 sentence pairs and 65,614 unique word pairs, designed to address the transliteration challenges of Romanized Assamese social media text. We implemented ten state-of-the-art methods, which include two statistical models, two neural models (a BiLSTM with attention and a neural transformer), three pre-trained models, and fine-tuned versions such as the multilingual IndicXlit transliteration system. Additionally, we evaluated two sequence-to-sequence large language models, ByT5 [18] and mT5 [19].
- **Chapter 4 Analysis of Transliteration Characteristics and Variations:** This chapter explores the complexities of back transliteration in Romanized Assamese social media text, highlighting the lack of standardized rules and challenges

TH-3777_186155001

posed by informal language use. It analyzes data from Twitter, Facebook, and YouTube and evaluates three transliteration models—PBSMT, BiLSTM with attention, and a neural transformer model. The chapter provides insights into transliteration variations and an in-depth error analysis on three transliteration model outputs. Additionally, attention analysis from the BiLSTM model reveals the intricate character mappings between Assamese and Roman orthographies, offering valuable insights into transliteration variations.

- **Chapter 5 Study on the Adaptability of Transliteration Models to a Canonical Dataset vs. a Noisy Social Media Dataset:** This chapter evaluates the performance of a state-of-the-art multilingual transformer-based indic transliteration model, IndicXlit [8], on both noisy Assamese social media text and clean canonical data. It compares this model with a monolingual Assamese model trained on noisy social media data, highlighting challenges in handling ambiguity and noise in social media text. The study also explores strategies for integrating noisy and clean datasets to improve transliteration accuracy in real-world scenarios.

- **Chapter 6 Study on the Importance of Context in Transliteration:** This chapter explores the impact of context on reverse transliteration, comparing sentence-level and word-level transliteration approaches for code-mixed Assamese social media text. It evaluates nine pre-trained large language models (LLMs) on both levels, finding that sentence-level transliteration, which incorporates contextual information, performs better. The study also highlights the importance of domain diversity through a cross-domain adaptation experiment. This cross-domain experiment allowed for a comprehensive evaluation of model performance when trained on one platform and evaluated on another, providing insights into the adaptability of transliteration models across different social media domains.

- **Chapter 7 Study on the Importance of Transliteration for Downstream Applications:** This chapter highlights the importance of reverse transliteration in downstream NLP tasks, focusing on sentiment analysis, by comparing models trained on Romanized and native Assamese text. The findings reveal that models using native script outperform those using Romanized text, underscoring the significance of reverse transliteration in enhancing NLP task accuracy.

- **Chapter 8 Conclusion and Future Work**: This chapter presents conclusion of the thesis with few possible future research directions to the thesis.

TH-3777_186155001

Chapter 2

Literature Survey

As mentioned earlier, Machine Transliteration was studied alongside Machine Translation to translate proper nouns, technical terms, and out-of-vocabulary words. Most of the previous research focused on improving machine translation performance by transliterating proper nouns, technical terms, and words that are out of vocabulary. However, with the increasing popularity of social media, user-generated content has flooded the internet, and most of this content is transliterated text with lots of misspellings, abbreviations, code-mixing, etc. Thus, the complexity of processing these noisy texts increases since most of the NLP tools were developed only in formal writing. Moreover, Social media users use transliterated text not only for proper nouns, technical terms, or out-of-dictionary words but also for other dictionary terms without following standard transliteration guidelines. In fact, there is no standard transliteration guideline for most of the language pairs.

In the literature, studies on transliteration have focused primarily on two approaches, namely (i) *rule based* and (ii) *data driven*. The rule-based approach uses mainly expert knowledge to create mapping rules from the source orthography to the target orthography [[20],[21]]. The data-driven approach focuses on creating a sizeable parallel corpus and developing machine learning methods for transliteration [[16],[6]]. As most recent transliteration studies reported in the literature exploit data-driven approaches, this section briefly summarizes some of these studies. Karimi *et al.,* 2011 [22], Kaur *et al.,* 2016 [23] and Prabhakar and Pal 2018 [24], did a thorough literature survey on Machine Transliteration methodologies and challenges for Indian as well as other languages. There, they discussed the research on different transliteration approaches: rule-based, traditional machine learning-based, and neural network-based. Prabhakar and Pal 2018,[24] discuss various ways of obtaining transliteration, i.e., transliteration generation, mining, and fusion approaches. The following subsections briefly review the existing literature for these Machine Transliteration approaches.

TH-3777_186155001

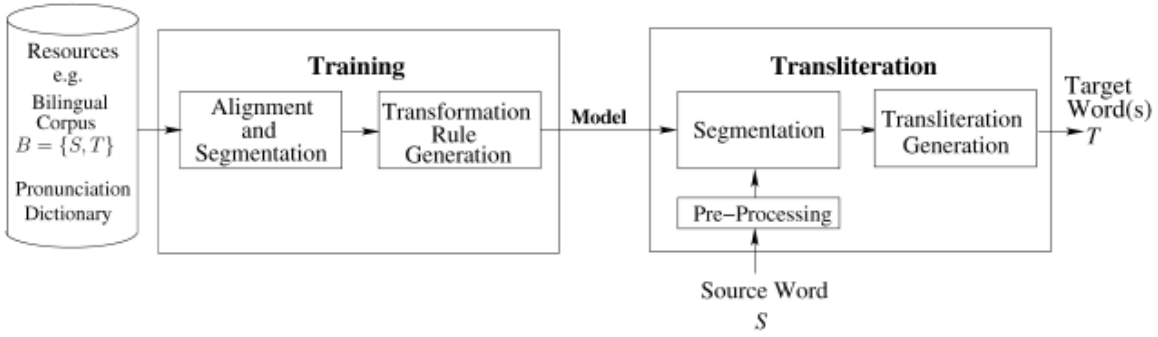

Figure 2.1: General transliteration framework

2.1 Analysis of Various Transliteration Methodologies

We encountered four different types of transliteration approaches, which are **Phoneme-based approach**, **Grapheme-based approach**, **Hybrid approach**, and **Combined approach**. Phoneme-based approaches regard the problem as a purely phonetic process. A Grapheme-based approach perceives it as an orthographic process and uses spellings; a Hybrid approach mixes these two approaches, and a Combined approach combines any spelling or phonetic methods. In the grapheme-based transliteration method, graphemes from the source word are directly mapped to graphemes in the target word. In contrast, in the phoneme-based transliteration method, the source grapheme is first converted into an intermediate phonetic representation. Next, the phonetic representation of the source word is mapped to the target word's phonetic representation, which is then converted into the target word's grapheme representation. We have shown a high-level transliteration architecture in Figure 2.1, Phoneme-based approach in Figure 2.2 and Grapheme-based approach in Figure 2.3, [22]. People explored various Rule-based methods, Traditional Machine Learning methods, and state-of-the-art Neural Network-based methods for transliteration methodologies.

2.1.1 Traditional Machine Learning based methods

Transliteration methods have primarily been proposed to transform text between English and non-English languages. This section will review dictionary-based methods and traditional machine learning techniques, focusing on these approaches for other languages and Indian languages. Apart from the shared task of transliterated search organized by the Forum for Information Retrieval (FIRE), all of the mentioned works in this section

TH-3777_186155001

deal with the transliteration of named entities (NEs), proper nouns, technical terms, and out-of-vocabulary (OOV) words.

Phoneme-based approach: Knight and Graehl, 1998 [2] used a phoneme-based transliteration approach to convert English to Japanese Katakana. They also used a weighted finite-state transducer (WFST) and weighted finite-state acceptor (WFSA) to transliterate English names and technical terms written in Japanese Katakana back to actual English. They introduced a four-step approach to transliteration. A Japanese source "S," was first converted to its phonetic representation "I_S," and then these source phonemes were mapped to target English phonemes "I_T." Finally, a phoneme-to-grapheme mapping produces the English target word "T." Stalls and Knight, 1998 [20] proposed a rule-based back-transliteration method for Arabic out-of-dictionary words into English. They also discussed various problems for transliteration in Arabic, as there is no unique pronunciation dictionary for Arabic out-of-dictionary words, a lack of electronic resources for Arabic pronunciation, and short vowels are not written in Arabic. Wan and Verspoor, 1998 [21] investigated a general phoneme-based approach to English-Chinese transliteration. They employed two steps: a grapheme-to-phoneme conversion step and a phoneme-to-grapheme transformation step based on a fixed set of rules. Lin and Chen, 2002 [25] proposed a back-transliteration method. They used a modified Widrow-Hoff learning algorithm that automatically captures the phonetic similarities from a bilingual transliteration corpus. In their approach, they used a pronunciation dictionary to transform English and Chinese names to their IPA representation and then applied a similarity measure on the phoneme. Similarly, Virga and Khudanpur, 2003 [[26],[27]] also examined English-Chinese transliteration using phoneme representation of English names. They utilized the Festival speech synthesis system to convert English names into phonemes, extracted sub-syllables to match Chinese pronunciations and converted them into Chinese. Instead of using an indirect source-channel model, Gao *et al.,* 2004 [28] investigated a direct model for English-Chinese transliteration. Jeong *et al.,*1999 [29] proposed a phoneme-based back-transliteration method for Korean out-of-dictionary phrases to English. Back-transliteration candidates were generated using a Hidden Markov Model (HMM) implemented as a feed-forward network with error propagation. In a similar study, Jung *et al.,* 2000 [30] also proposed an English-Korean transliteration method using an extended Markov window. Oh and Choi, 2002 [31] studied English-Korean transliteration using pronunciation and contextual rules. They referred to their method as *correspondence-based* transliteration. Using only monolingual resources, Ravi and Knight, 2009 [32] proposed an automatic transliteration method using a four-stage cascade of weighted finite-state transducers.

TH-3777_186155001

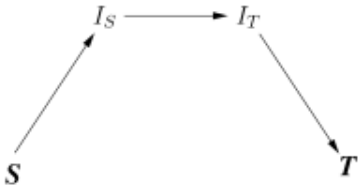

Figure 2.2: Phoneme-based transliteration

Figure 2.3: Grapheme-based transliteration

Grapheme based approach: AbdulJaleel and Larkey, 2003 [33] examined a grapheme-based English-Arabic transliteration using n-gram models. During the training phase, word pairs from a bilingual transliteration corpus are aligned using GIZA++[1]. Then, transformation rules are formed, and probabilities are assigned according to the corpus frequencies. Sheriff and Kondrak, 2007 [34] proposed Arabic-English transliteration using dynamic programming and substring-based conversion techniques. Li *et al.,* 2004 [35] proposed a direct orthographic mapping (DOM) model to generate transliteration from English-Chinese through a joint source-channel model with an n-gram transliteration model. Kang, 2000 [36] used a Hidden Markov model with a bi-gram language model, and Kang *et al.,* 2000 [37] proposed a new alignment algorithm with decision-tree learning for both forward and backward English-Korean transliteration. Lindén, 2006 [38] investigated the problem of transliteration between a set of European languages: English, Finnish, French, German, Italian, Danish, Dutch, Portuguese, and Spanish using a weighted finite-state transducer (WFST). Li *et al.,* 2007 [39] proposed a semantic transliteration method for personal names. They experimented with three languages of origin: Japanese, Chinese, and English. Names were first separated into a female given name, a male given name, and a surname. Using sequences of four characters, the origin of these names and their gender were detected. Karimi *et al.,* 2007 [40] were the first to investigate the effects of corpus construction on the English and Persian language transliteration systems. They proposed a novel algorithm for both forward and backward transliteration of English-Persian language pairs.

Hybrid approach: Al-Onaizan and Knight, 2002 [41] first introduced a hybrid transliteration approach for Arabic-English transliteration. They adopted the source channel

[1] http://www.statmt.org/moses/giza/GIZA++.html

TH-3777_186155001

model for a phonetic approach and a rule-based approach for spelling information. Since transliteration is both a spelling and a phonetic conversion process, Oh and Choi 2005[42], 2006[43] proposed a hybrid transliteration model for the English-Korean language. They compared their hybrid model with the previous model based on graphemes and phonemes, using several machine learning methods such as Decision Tree Model (DT), Maximum Entropy Model (MEM), and Memory-Based Learning Model (MBL).

Combined approach: Oh and Isahara, 2007 [44] studied English-Korean and English-Japanese transliteration using a combined transliteration approach. They proposed a method based on the Support Vector Machine (SVM) and Maximum Entropy Markov Model (MEM) to rank each transliteration system's output. These individual systems come from various grapheme-based, phoneme-based, and hybrid methods. Support vector machine-based methods provide better results for both language pairs. Karimi, 2008 [45] proposed a combined transliteration method for English-Persian language pair under the black-box framework. They aggregated multiple spell-based transliteration systems (M_i; i = 1,2,...,15) into one system, "M". The combined method is a mixture of Naïve-Bayes classifier[2] and majority voting scheme.

The study of Machine transliteration was also observed as a part of many workshops shared tasks, namely: Named Entity Workshop (NEWS) [[46],[47], [48]] and Workshop organized by Forum for Information Retrieval (FIRE) [49]. Transliteration shared task of the NEWS workshop mainly focused on named entity transliteration, whereas the FIRE workshop was solely devoted to Indian language transliteration only. In the NEWS 2015 shared task, Wang *et al.,* 2015 [50] described their grapheme-based segmentation approaches for English-Korean and English-Chinese transliteration. They used DirectTL+, M2M-aligner, and a string conversion model. Nicolai *et al.,* 2015 [51] focused on methods for combining several base systems and using transliteration from several languages.

2.1.2 Traditional Machine Learning based methods for Indian Languages

Apart from other languages, much work is being done on Indian language transliteration. In this section, we are going to review them thoroughly. There could be three different ways to get the transliteration of Indian languages: (i). From English-to-Indian Language (E-IL), (ii). From Indian Languages to English (IL-E) and (iii). From one Indian Language-to-another (IL-IL) [24]. However, compared to other languages, research on

[2] https://en.wikipedia.org/wiki/Naive_Bayes_classifier

TH-3777_186155001

Indian languages is still in its early stages. Many regional languages in India remain unaddressed. The unavailability of adequate amounts of data and the lack of more mature technology could be the main reason for this shortage.

Das and Gambäck (2013) [52] conducted an early study examining the nature of code-mixing in social media texts. Their analysis highlighted several challenges unique to this domain, including frequent occurrences of code-switching, lexical borrowing, anglicisms, and phonetic spellings, all of which contribute to the complexity of social media text processing. Ekbal *et al.,* 2006 [53] explored the revised joint-source channel approach proposed by Li *et al.,* 2004 [35] for Bengali-English transliteration. The transliteration units in the original word were selected using a regular expression based on the occurrence of consonants, vowels, and matras (Bengali language delimiter), and hand-crafted transformation rules were provided for one-to-many alignments between English and Bengali. The method they proposed also includes linguistic knowledge of possible conjuncts and diphthongs in Bengali and their English equivalent. Malik, 2006 [54] proposed a rule-based transliteration system for converting words between two scripts of Punjabi: Shahmukhi, which is based on Arabic script, to Gurumukhi, which is derived from Landa script. Malik *et al.,* 2008 [55] also proposed a Hindi-Urdu transliteration model using a finite-state transducer(FST). They identified three problems while transliterating Urdu terms into their Hindi equivalent. First, **multi-equivalence**, second, **no equivalence**, and third, **missing diacritical marks**. Vijaya *et al.,* 2009 [56] implemented their transliteration problem using the WEKA [57] j48 decision tree classifier. Their transliteration model has been experimented with and evaluated in English to Tamil transliteration. Chinnakotla *et al.,* 2010 [58] proposed a three-stage transliteration system based on the defined rules for Hindi to English, from English to Hindi, and from Persian to English. First, the origin of the word was identified by source-side character sequence modeling (CSM). Second, transliteration candidates were generated using rule-based manual character matching, and finally, transliteration candidates were ranked again using CSM on the target side. Kumaran *et al.,* 2010 [59] proposed a combination of two composite machine transliteration systems. Their "sequential composition" system combines the individual transliteration components, say X -> Y and Y -> Z, to provide transliteration functionality, X -> Z. The "parallel compositional" system aggregates multiple transliteration paths from X -> Z to improve the quality. Antony *et al.,* 2010 [60] developed a transliteration system from English to Kannada transliteration using support vector machines (SVM) classifier. Josan and Lehal, 2008 [61] proposed a transliteration approach from Punjabi to Hindi combined with a basic character mapping technique with rule-based enhancements and a Soundex algorithm. Based on the noisy channel model, Josan and Kaur, 2011 [62] further improved their baseline system, [61]. Regarding the low

TH-3777_186155001

accuracy in the baseline system, [62] identified three issues: multiple equivalent letters for a single letter (sound [sh] in Punjabi has two equivalent letters in Hindi "श" and "ष"), single letter equivalent of multiple letters and no equivalent letter in the other language (Hindi letter "ऋ" has no corresponding letter in target language). Dhore *et al.* 2012 [63] proposed a forward machine transliteration model for Hindi and Marathi named entities, NEs (both written with the Devanagari script) to English. Their phoneme-based statistical approach used phonemes and the length of NEs as features for supervised learning. Using the monosyllabic features of the Manipuri language, Thoudam Doren Singh, 2012 [64] proposed a rule-based approach for Bengali-Meetei Mayek transliteration. Using the monosyllabic characteristics of the Manipuri language, their proposed rule-based approach points out the importance of deeper integration of linguistic rules in the process. Dasgupta *et al.,* 2013 [65] proposed an English-Bengali back-transliteration system using two different models: joint source-channel model and tri-gram model. English words were phonetically analyzed using a grapheme-phoneme converter (G2P) to generate a transliteration unit, TU, according to the technique of Ekbal *et al.,* 2006 [53]. Bengali words were segmented according to linguistic rules. They evaluated their approach using a parallel corpus comprising 83,000 English-Bengali transliteration pairs. In FIRE-2013 [49] transliterated search shared task, Pakray *et al.,* 2013 [66] used four different models: Joint Source Channel Model (JSC), Modified Joint Source Channel Model (MJSC), Improved Modified Joint Source Channel Model (IMJSC) and Trigram Model (Tri) for the transliteration of English-Bangla and English-Hindi language pairs. Joshi *et al.,* 2013 [67] used a syllabification approach to perform the transliteration from Roman script(English) to Devanagari script(Hindi). Similarly, in FIRE-2014 transliterated search shared sub-task II, Bhat *et al.,* 2014 [68] submitted their transliterations approach for the following language pairs: Hindi-English, Gujarati-English, Bengali-English, Kannada-English, Malayalam-English, and Tamil-English. They first transliterated the words to WX notation [3] and then converted WX notation to the native scripts. They trained a character-based ID3 Decision tree classifier for mapping non-English words to WX using the modified word-transliterations pairs. Mukherjee *et al.,* 2014 [69] introduced back-transliteration from Roman Hindi to native Devanagari script to overcome the problem of spelling variations in transliterated search queries. They also proposed a rule-based consonant mapping system and a naive Bayes classifier was used to classify transliterated words that when viewed individually, can belong to both Hindi and English. Ganguly *et al.,* 2014 [70] tried to solve the vocabulary mismatch problem that occurs due to the multiple transliteration alternatives. Patel and Desai, 2015 [71] proposed a LIGA (Language Identification Graph Approach) approach to label the words with their language tags, followed by transliteration using rule-based syllabification to terms in their native script. Gupta *et al.,* 2014 [72]

[3] https://en.wikipedia.org/wiki/WX_notation

TH-3777_186155001

used a modified joint source-channel-based transliteration model for Bangla-English and Hindi-English language pair. Kunchukuttan and Bhattacharyya, 2015 [73] proposed a phrase-based transliteration model for Hindi, Bengali, and Tamil. One-gram and bigram characters were used to train their transliteration model, and the results were extracted using a module in the Moses[4] toolkit. In another study, Kunchukuttan *et al.*[16], 2015 also presented Brahmi-Net, an online transliteration and script conversion system for major Indian languages. Roark *et al.* [6], 2020 released the Dakshina dataset, consisting of 12 South Asian languages (mainly Indian languages) written in Latin script, and provided baseline results for transliteration and language modeling tasks. For processing Arabizi (Arabic written in Roman script) text, Al-Badrashiny *et al.* (2014) [74] and May *et al.* (2014) [75] used finite state transducers combined with language models to generate and rank candidate transliterations for Arabizi words.

2.1.3 Neural Network based methods

With the rising popularity of Neural Network-based methods among the NLP research community and also the successful implementation of the neural machine translation(NMT) framework for solving the Machine Translation problem proposed by Bahdanau *et al.,* 2014 [76], Research on Machine Transliteration also shifted towards the Neural Network based approaches from traditional Machine learning-based methods. That makes Neural approaches the state-of-the-art techniques for solving the Machine Transliteration problem. This section will review the work on different Neural Network-based proposed approaches for solving the Machine Transliteration problem.

Among neural network-based approaches, Deselaers *et al.,* 2009 [77] proposed new deep belief networks (DBNs) [78] based transliteration technique later also adopted by Sanjanaashree *et al.,* 2014 [79]. DBN is a generative graphical model. Deselaers *et al.,* 2009 [77] evaluated their proposed methods on an Arabic-English transliteration task for transcribing Arabic city names into the equivalent English spelling. Sanjanaashree *et al.,* 2014 [79] improved the bilingual machine transliteration task for Tamil and English languages with limited corpus containing proper nouns and technical terms. Their motivation behind this experiment was that DBN is entirely bi-directional, supports dimensionality reduction, supports unsupervised learning functions, and is a supervised method for transliteration. As part of the workshop shared task organized by NEWS 2015, Finch *et al.*, 2015 [80] examined a neural network sequence-to-sequence transduction model for the transliteration task. They proposed two models. In their first model, they successfully enhanced the system's performance by incorporating the neural network

[4] http://www2.statmt.org/moses/

TH-3777_186155001

score as a feature in the phrase-based statistical machine transliteration system. Their second model was the neural network model, which was used independently, and a simple beam search algorithm. They evaluated their system on 14 language pairs. Rao *et al.,* 2015 [81] proposed a Grapheme to Phoneme (G2P) model based on a Long Short-Term Memory (LSTM)[5] recurrent neural network (RNN). They experimented with unidirectional LSTM (ULSTM) with deep bidirectional LSTM (DBLSTM) and different kinds of output delays and with a connectionist temporal classification (CTC) [82] layer. Shao and Nivre, 2016 [83] presented an English-Chinese both forward and backward transliteration system as a part of their participation in the NEWS 2016 machine transliteration shared task. They used convolutional neural networks (CNN) to extract character-level information from the transliteration units and stack a simple recurrent neural network (RNN) on top for sequence processing. Rosca and Breuel, 2016 [84] described two neural-network-based sequence-to-sequence models for transliteration. Their first model is based on epsilon insertions and CTC alignment [82], and the second model is an attention-based sequence-to-sequence model commonly used in end-to-end machine translation proposed by Bahdanau *et al.,* 2014 [76]. They evaluated their proposed model on Arabic to English (AR-EN) and English to Japanese (EN-JA) transliteration and grapheme to phoneme conversion, specifically English to IPA (EN-IPA) conversion. Finch *et al.,* 2016 [85] proposed a purely neural network-based transliteration system by exploiting the agreement between a pair of bidirectional target LSTMs to generate balanced targets with both good suffixes and good prefixes. Their evaluation results surpass the current state-of-the-art on most language pairs and expose weaknesses in some tasks, motivating further study. To build their transliteration system, they used a publicly available toolkit called Janus toolkit[6]. Ameur *et al.,* 2017 [86] proposed a neural attention-based encoder-decoder (Sutskever *et al.,* 2014 [87]) system for the task of Machine Transliteration between Arabic and English language. They compared their system to several sequence-to-sequence models and demonstrated the effectiveness of their proposed approach compared to some previous research developed in this area. Abbas and Sadat (2017) [88] applied character-level neural machine transliteration to convert Arabizi into Modern Standard Arabic, focusing on Algerian dialect. Younes *et al.* (2018) [89] used sequence-to-sequence models with beam search for word-level transliteration of Tunisian Arabizi. Kundu *et al.,* 2018 [90] adopted two different neural machine translation (NMT) frameworks: convolutional sequence to sequence-based neural machine translation framework and recurrent neural network-based framework. Their machine Translation architecture is language-independent and achieved top performance for the English-Persian language pair in the NEWS2018 shared task on transliteration. Their proposed system also showed promising

[5] https://en.wikipedia.org/wiki/Long_short-term_memory
[6] https://github.com/lemaoliu/Agtarbidir

TH-3777_186155001

results while doing multilingual transliteration. Merhav and Ash, 2018 [91] compared the performance of their transliteration task using the traditional weighted finite-state transducer (WFST) approach against two neural approaches: the encoder-decoder recurrent neural network method, Sutskever *et al.,* 2014 [87] and the non-sequential Transformer method by Vaswani *et al.,* 2017 [92]. They demonstrated that the Tensor2Tensor neural transformer method produces reliable results compared to the LSTM-based encoder-decoder neural method and the WFST-based traditional method. However, they also admit that the Tensor2Tensor neural transformer is computation-intensive. For three Indian languages: Hindi, Sindhi, and Kannada, Joshi *et al.,* 2018 [93] proposed a deep learning-based scalable pipeline using LSTMs and Sequence-to-sequence models to find an optimal model by comparing the results. Their results showed that sequence-to-sequence models are better suited for this solution. They also discussed various methods for pre-processing data and post-processing the output for optimal performance. Inspired by sequence-to-sequence recurrent neural network-based translation methods, Le and Sadat, 2018 [94] proposed their transliteration approach as a part of NEWS 2018 shared task which applies an alignment representation for the input sequences and the pre-trained source and destination embeddings to overcome the transliteration problem for a low-resource language pair, English-Vietnamese. Their method was inspired by the sequence-to-sequence recurrent neural network-based translation method. During the same NEWS 2018 shared task, Grundkiewicz and Heafield, 2018 [95] adopted a deep attentional RNN encoder-decoder model for transliterating named entities. They applied several well-established neural machine translation (NMT) techniques, such as dropout regularization, model assembly, right-to-left model rescoring, and reverse translation. Laitonjam *et al.,* 2018 [96] proposed a machine transliteration system to transliterate loan words and named entities in an orthographically very different English-Manipuri language pair. They first compared the performance between cutting-edge learning models, namely RNN, LSTM, and Bi-LSTM, using seq2seq autoencoder [87] and also dictionary-based models, and observed that among all, BiLSTM was giving better performance. Similarly, they reported that grapheme-based representations outperform their phoneme-based counterparts in dictionary and learning-based methods. Dershowitz and Terner, 2020 [97] proposed a recurrent neural network (RNN) based transliteration method, which uses the connectionist temporal classification [82] loss to deal with unequal input/output lengths. Their model will automatically convert Judeo-Arabic into an Arabic script. The neural machine translation (NMT) framework introduced by [98] has gained significant popularity in the natural language processing (NLP) research community. Consequently, research on machine transliteration has also transitioned toward neural approaches, which are now considered state-of-the-art techniques for addressing transliteration challenges. Khan and Sarfaraz,

TH-3777_186155001

2019 [13] proposed an attention-based encoder-decoder model trained on a bilingual parallel corpus. They demonstrated the application of neural machine translation (NMT) in Urdu language transliteration, focusing on the contextual coverage of the language, improving transliteration accuracy. Byambadorj *et al.*[12], 2021 explored various character level sequence-to-sequence (seq2seq) models for Mongolian Cyrillic written in Roman script after normalizing the noisy text using various performance enhancement methods. Kunchukuttan *et al.* [7], 2021 evaluated neural machine transliteration for English and 10 Indian languages. Their main focus was multilingual transliteration to utilize orthographic similarity between Indian languages. Their multilingual baseline model [99] is a character-level, attention-based, encoder-decoder model with all the model components shared amongst all languages. They separately trained multi-target (English to Indian languages) and multi-source (Indian languages to English). They used *Marian* [100] to train their transliteration models. Madhani *et al.*[8], 2023 introduced Aksharantar, a large transliteration dataset for 21 Indian languages, and achieved state-of-the-art results with a single transformer-based multilingual transliteration model, IndicXlit.

2.2 Brief Summary

In this chapter, we provide an overview of the literature on various transliteration approaches. We noticed that all transliteration approaches followed a general transliteration framework. For a rule-based transliteration system, people use some handcrafted rules for language pairs. In contrast, most researchers have used traditional machine learning technologies and the latest neural network training approaches to solve the transliteration problem. Machine transliteration in the field of social media is much more complicated because people have to solve other normalization issues that are characteristic of social media together. Machine transliteration is a challenging task for other non-Roman languages compared to Roman English. Researchers have also identified various language-specific orthographic and phonetic problems with transliteration into Arabic, Urdu, Persian, and some Indian languages such as Hindi, Punjabi, Bengali, etc.

We observed that most prior research on transliteration and normalization in the social media domain did not systematically examine the characteristics of transliterated text. A thorough understanding of language-pair-specific orthographic and phonetic properties is essential for assessing compatibility between source and target languages. While earlier studies identified social media-specific challenges, they often lacked detailed linguistic analysis. Although a few works addressed Indian language social media text, our study explores a wide spectrum of transliteration methods–from statistical and neural

TH-3777_186155001

approaches to modern transformer–based models. Over the course of this research, large language models became increasingly relevant, and we incorporated them accordingly. Rather than choosing a single "most suitable" method in advance, we reviewed the full landscape of prior work to evaluate their advantages, limitations, and applicability, with a particular focus on social media contexts. All methods discussed are relevant to our work, especially in handling noisy, user-generated transliterated content.

TH-3777_186155001

Chapter 3

Novel Dataset Generation

The previous chapter provided a comprehensive literature review on transliteration methodologies, and the challenges of processing transliterated social media text. It pointed out the shortcomings of current datasets, which predominantly emphasize named entities and structured transliteration pairs, frequently neglecting the various spelling variations and phonetic inconsistencies present in user-generated content.

This chapter introduces a new back-transliteration dataset specifically designed for Assamese social media text. This dataset addresses resource scarcity and reflects real-world variations in transliteration. Unlike previous datasets focusing on exact annotations, our dataset emphasizes the diversity of informal transliterations, making it more representative of user practices.

Additionally, we assess the effectiveness of the dataset using advanced transliteration models, including statistical approaches and modern neural architectures, to set a benchmark for future research.

3.1 Introduction

This chapter introduces a novel back-transliteration dataset that showcases the linguistic diversity present in Assamese social media text and explores its effects on existing transliteration models. By making this dataset publicly available, we aim to promote future research in low-resource transliteration and improve NLP applications for Assamese and similar languages. Despite advancements in transliteration research, many low-resource languages, including Assamese, still lack comprehensive back-transliteration datasets. Assamese, an Indo-Aryan language primarily spoken in northeastern India, faces significant

TH-3777_186155001

challenges due to the limited availability of structured linguistic resources. While numerous forward transliteration datasets exist, most are designed for formal text and do not adequately capture transliterated social media content's informal and highly variable nature. This resource gap limits the effectiveness of existing transliteration models in handling real-world scenarios where users frequently write Assamese in Roman script due to the prevalence of English keyboards. The rise of digital communication in India has significantly increased social media content produced in Indian languages, often utilizing the Roman script. Reports from KPMG[1] and Statista[2] indicate a notable growth in internet users engaging with Indian languages, with a considerable portion participating in transliteration practices. Despite this, the absence of processing tools for transliterated text hampers the recognition and analysis capabilities of existing NLP models. While forward transliteration datasets are generally developed through controlled annotation methods, back-transliteration from noisy, user-generated content poses a unique challenge due to variations in spelling and phonetic inconsistencies. To illustrate the limitations of existing datasets, let's consider the Assamese word "শুভেচ্ছা" (pronounced /xub^{h}essA/), which means "**good wishes.**" Forward transliteration datasets, such as Aksharantar [8], provide only a limited set of three variations. In contrast, our collected dataset reveals a much wider range of 46 naturally occurring transliteration variations for the same word, sourced from real-world social media content. This variation underscores the need for a dataset that accommodates the flexible and informal nature of transliterated text.

We introduce a large-scale back-transliteration dataset specifically designed for Assamese social media text collected from three popular social media sites: (Facebook, YouTube and Twitter)[3] to address this gap. Our dataset differs from previous datasets that originate from structured domains like Wikipedia, as it is based on real social media interactions, highlighting actual usage patterns. We collected transliterated Assamese posts from social media, followed by expert annotation to align them with their native-script equivalents. This dataset is intended to serve as a benchmark for assessing back-transliteration models and enhancing natural language processing applications for Assamese. Beyond dataset creation, this chapter also assesses the performance of multiple state-of-the-art transliteration models. We begin with two statistical baseline models from NEWS 2018 [101]: SEQUITUR[4] [102], a joint n-gram-based string transduction system, and a phrase-based statistical model using Moses[5] [103]. Subsequently, we explore deep neural network models, including BiLSTM, with attention [98] to the transformer [92] model. Additionally, we report results from three publicly available transliteration APIs: IndicTrans [104],

[1] KPMG
[2] Statista
[3] https://www.facebook.com, https://www.youtube.com, https://www.twitter.com
[4] Sequitur
[5] Moses

TH-3777_186155001

google transliteration API for the Google Input Tools[6], and a multilingual transliteration model, IndicXlit [8]. Finally, we fine-tune three large-scale pre-trained models—IndicXlit, mT5 [19], and a tokenizer-free extension of the mT5 [19] model, ByT5 [18] on our dataset to examine their effectiveness in handling transliterated Assamese text. In summary, this chapter introduces a novel back-transliteration dataset that highlights the diverse linguistic features of Assamese social media text. We assess its impact on current transliteration models. By making this dataset publicly available, we aim to support future research in low-resource transliteration and improve NLP applications for Assamese and similar languages. The dataset can be downloaded from Github[7].

Contributions:

The key contributions of this chapter are:

1. Introduction of a novel backward transliteration dataset for Assamese language sourced from social media.
2. Emphasis on back transliteration to capture real-world variations in Assamese social media text.
3. Evaluation and public release of state-of-the-art transliteration models and dataset for the research community.
4. Fine-tuning of a pre-trained multilingual transliteration model and two state-of-the-art sequence-to-sequence large language models for Assamese transliteration tasks.

3.2 Related work

Recent years have seen notable advancements in Indic language transliteration. Kunchukuttan *et. al*, 2015 [16] proposed *Brahmi-Net*, an online statistical transliteration system for transliteration and script conversion for all major Indian language pairs (18 languages and 306 language pairs) including Assamese. Roark *et. al*, 2020 [6] introduced the Dakshina dataset, covering 12 South Asian languages in the Roman script, laying the foundation for transliteration and language modeling tasks. Extending this, Kunchukuttan *et. al*, 2021 [105] explored neural machine transliteration for English and 10 Indian languages, emphasizing multilingual transliteration. Madhani *et. al*, 2023 [8] presented

[6]Google Input Tools
[7]https://github.com/osintg-iitghy/LREC-COLING-2024-OSINTG-IITG

TH-3777_186155001

the Aksharantar dataset, the largest transliteration dataset covering 21 Indian languages, achieving state-of-the-art results with the multilingual IndicXlit model. In a recent study by Ruder *et. al*, 2023 [106], sentence-level transliteration was evaluated across 13 languages, encompassing 12 languages from the Dakshina dataset and the Amharic language across 30 distinct transliteration directions. The researchers harnessed transfer learning setups (mT5-Base [19], ByT5-Base [18], Flan-PaLM-62B [107]) to conduct their experiments. Kirov et al. (2024) investigate context-aware transliteration for South Asian languages, focusing on sentence-level processing. Their study shows that including non-parallel mono-script corpora enhances transliteration quality when large parallel datasets are absent. For social media like infomral text, the Forum for Information Retrieval (FIRE) organized pivotal shared tasks like FIRE 2013 and FIRE 2014 [108, 109], focusing on Hindi song lyrics in Roman script. Despite these achievements, current research does not explicitly address transliteration challenges in Romanized social media datasets.

3.3 Dataset

Our dataset was curated from three popular social media platforms, YouTube, Twitter (currently X), and Facebook using three publicly available APIs for systematic data extraction. Specifically, we employed the YouTube Data API[8] to harvest comments from predefined Assamese YouTube channels[9]. On Twitter (currently X), our focus was on acquiring reply tweets only from a prominent Assamese Twitter handle[10], utilizing the Tweepy API[11]. Similarly, we extracted comments from selected Assamese Facebook pages[12] using the Facebook Graph API[13]. Comprehensive details about the experimental dataset and the duration of data collection are available in Table 3.1.

The posts in Table 3.1 were selected from specific handles, pages, and channels that are most prominent for Romanized Assamese content. For Facebook, we collected comments from two popular Assamese pages at that time: the Gauhati University Confession page and the CMO Assam page. Facebook's data collection policies then allowed access only to public pages. Twitter data was collected from reply tweets to the official handle of the Assam Chief Minister, @himantabiswa, during the COVID-19 period. This handle was particularly active, and citizens frequently engaged with it to seek information, express opinions, or appeal to the government. While the general Assamese public was less active

[8] YouTube Data API
[9] Dimpu's Vlogs, News Live, Assamese Mixture
[10] @himantabiswa
[11] Tweepy
[12] GU Confession Page, CMO Assam Page
[13] Facebook Graph API

TH-3777_186155001

Table 3.1: Statistics of the collected dataset from three major social media sources along with the duration of data collection

Social Media Sources	Duration of Data Collection	#posts collected	#posts annotated	#words Assamese (total)	#unique Assamese words
Facebook	Dec-2013 to Feb-2017	409,168	5,300	71,800	79,200
YouTube	Jun-2018 to Aug-2023	385,676	50,000	426,089	
Twitter	Mar-2021 to Aug-2021	285,676	5,012	91,400	

on Twitter, this period saw a surge in replies and interactions on the Chief Minister's tweets. For YouTube, comments were obtained from three widely followed Assamese channels: Dimpu's Vlogs, News Live, and Assamese Mixture, where users commonly posted in Romanized Assamese. These sources were selected to ensure that the dataset captures authentic, user-generated transliteration content across these 3 selected social media platforms.

In our dataset, we encompass a total of 60,312 sentences, ranging from single-word sentences to those extending up to 162 words. The average sentence length is 11.14, exhibiting a standard deviation of 8.38. Moreover, we note an average code-mixing percentage of 20.1% within the dataset. Code-mixing entails incorporating authentic English words alongside Romanized Assamese words, quantified as a percentage of the total words present in a sentence. At the word level, our dataset comprises a total of 671,921 words. Among them, 67,131 words are in English, 589,289 are Assamese words, and 15,501 are mixed-script words denoting a single token expressed in multiple scripts. Furthermore, an ambiguity analysis of the word-level dataset revealed that out of 52,299 unique Romanized word forms, 9,581 have more than one possible native Assamese transliteration, resulting in an ambiguity rate of 18.32%. This indicates that nearly one-fifth of the Romanized tokens in the dataset are inherently ambiguous due to multiple plausible native mappings, primarily caused by phonetic overlap and inconsistent Roman spelling conventions.

Out of the total 589,289 Assamese words, there are only 79,200 unique Assamese words, and from this set, we extracted a total of 65,614 unique transliteration pairs for conducting our experiments. Here, "unique Assamese words" refers to the Romanized variants of the native Assamese words, since a single native word can have multiple plausible

TH-3777_186155001

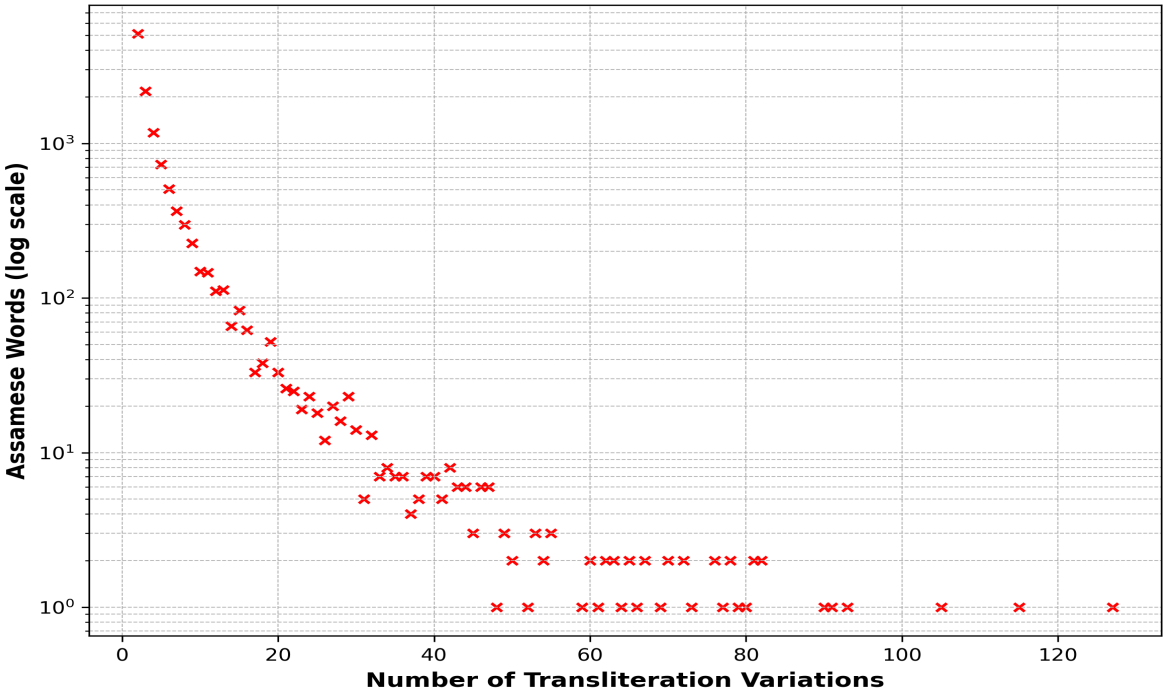

Figure 3.1: Distributions of Roman Transliteration Variations for Native Assamese Words.(Semi-log Scale)

Romanized forms. This distinction ensures clarity between the native script words and their corresponding Romanized transliterations used in the dataset. Again, as the nature of social media data, a single source token can be represented in multiple ways. i.e., a single token can exhibit multiple transliteration variations. We have noticed a maximum of 127 Roman transliteration variations for a native Assamese word in our dataset. We have also noticed that based on the context or the similarity in pronunciation, a single Roman word may represent multiple native Assamese words, with one Roman word in our dataset representing a maximum of 31 native Assamese words. Figure 3.1 visually demonstrates the connection between the number of Roman variations and the corresponding count of native Assamese words exhibiting those many variations. Again, the plot in Figure 3.2 reveals the relationship between the number of Roman words and the total count of back-transliterated native Assamese words represented by those Roman words. It's worth noting that although not many terms in our dataset have the maximum number of variations, many words display more than one variation. Two examples in Table 3.2 shows these variations found in our dataset.

Dataset Annotation

After acquiring the necessary dataset from our selected sources, we engaged 24 annotators and 3 linguistic experts as validators to annotate and verify the dataset. An online annotation tool, developed and deployed on our local server, facilitated this process.

TH-3777_186155001

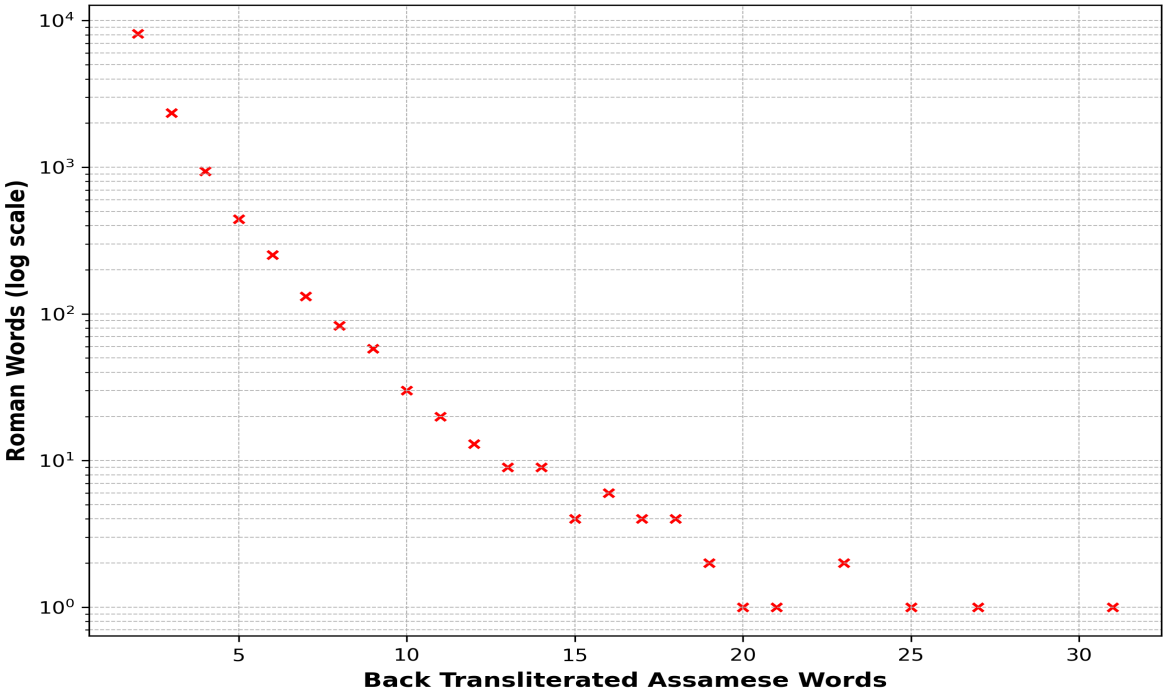

Figure 3.2: Distributions of Back Transliterated Native Assamese words represented by the Roman Words.(Semi-log Scale)

Table 3.2: Two examples of both Roman and native variations along with the frequencies present in our dataset

Term	Script Language	Underlying Language	English Meaning	Total Variations	Number of Variations with Frequencies
"বহুত"	Assamese	Assamese	Many	27	***bohut:*** *2915*, ***bhut:*** *1047*, ***bht:*** *851*, ***bahut:*** *651*, ***bohot:*** *126*, ***buhut:*** *105*, ***bhout:*** *47*, ***bhot:*** *46*, ***bahot:*** *24*, ***bhtt:*** *13*, ***boht:*** *12*, ***bohud:*** *12*, ***bhoot:*** *8*, ***bout:*** *7*, ***buhot:*** *7*, ***bhohut:*** *6*, ***bohout:*** *6*, ***bhaut:*** *6*, ***vohut:*** *6*, ***bohoot:*** *5*, ***bohuuuuuuuutttttt:*** *1*, ***bohuuuuuut:*** *1*, ***vohot:*** *1*, ***bohuuuuuuuut:*** *1*, ***bohuuuuut:*** *1*, ***bohuuuut:*** *1*, ***bhhhuut:*** *1*
"gai"	Roman	Assamese	To Sing	5	গাই: *65*, গায়: *18*, গৈ: *3*, যায়: *1*, গান: *1*

Annotators were selected based on their proficiency in both English and Assamese. A comprehensive annotation guideline was prepared in collaboration with linguistic experts. Both annotators and validators were required to register and log in to our system first. Upon logging in, each annotator received an initial set of 100 sample posts for tagging, presented one at a time after submission. During tagging, annotators adhered to the annotation guidelines. Linguistic experts validated annotations using two tags: accept and reject. Those exceeding the 80% acceptance threshold in the initial test proceeded to

TH-3777_186155001

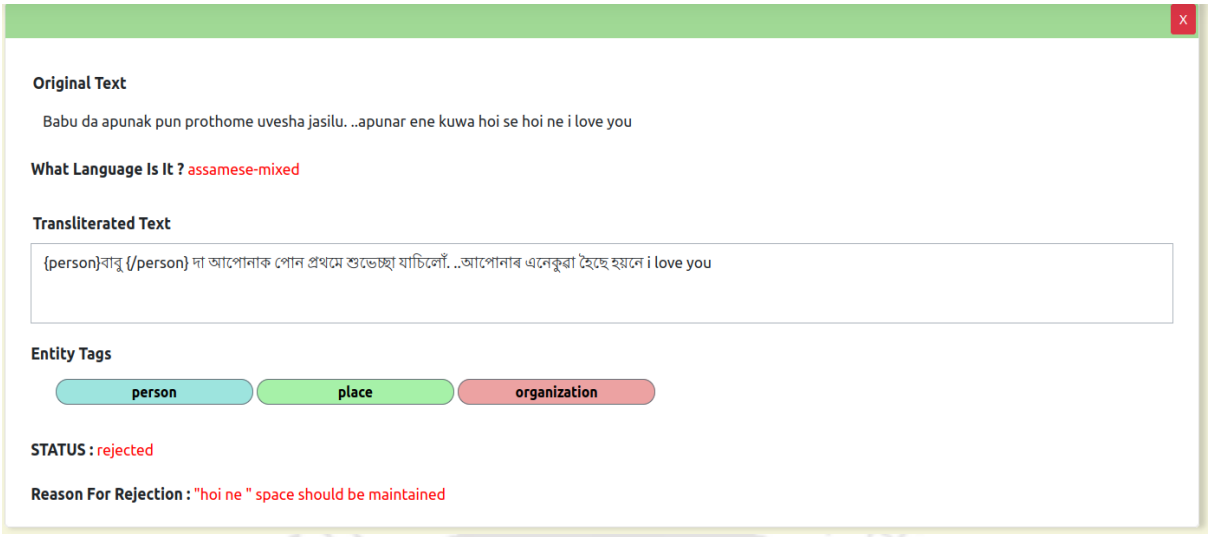

Figure 3.3: A snapshot of the annotation tool

the final annotation task. Annotators received a compensation of 3 INR for each accepted post.

The annotators were tasked with three main responsibilities. Firstly, they had to identify the language of the post at the sentence level, categorizing it as English, Assamese, Assamese-mixed, or Other. Secondly, identified Assamese words written in the Roman script were transliterated back to the corresponding Assamese words in the native script. English-origin words were retained if spelled correctly, otherwise replaced with accurate spelling. Thirdly, if a term in the sentence was recognized as a person's name, a geographical location, or the name of an organization, annotators selected and tagged the term or phrase as <person>, <place>, or <organization>, respectively, by clicking the appropriate label below the post.

Validators were responsible for reviewing and validating each post, marking it as accept or reject. In the case of rejection, validators provided explicit reasons for the rejection. Which will further reflected in the respective accounts of the annotators so that it can be tagged correctly in the subsequent attempts. The identical set of posts was allocated to two annotators to assess inter-annotator agreement in the reverse transliteration task at a later stage. We quantified inter-annotator agreement using Cohen's kappa (κ) coefficient and observed a value of 0.83. Figure 3.3 presents a snapshot of the annotation tool[14].

[14] https://www.iitg.ac.in/cseweb/osint/annotation/

TH-3777_186155001

3.4 Performance Metrics Used for Evaluation

To ensure a consistent and comprehensive evaluation framework across all experiments, a unified set of performance metrics was adopted for assessing the effectiveness of transliteration models. These metrics capture complementary aspects of model performance - from fine-grained character-level accuracy to overall sentence-level fluency and ranking quality. While not all metrics are applied in every chapter, this common evaluation framework is followed throughout this chapter and the subsequent chapters up to Chapter 6, with each experiment employing the most relevant subset of these metrics based on its specific objective.

Word Error Rate (WER)

WER measures the percentage of correctly predicted word pairs relative to the total number of word pairs and is defined as:

$$WER(\%) = \frac{\text{Number of correctly predicted word pairs}}{\text{Total number of word pairs}} \times 100\% \tag{3.1}$$

A lower WER value indicates better transliteration accuracy at the word level.

Character Error Rate (CER)

Character error is calculated based on the concept of the Levenshtein distance[110]. Levenshtein distance is measured as the minimum number of edits of a single character (substitution, deletion, or insertion) required to change a word into another word.

$$CER = \frac{\text{S} + \text{D} + \text{I}}{N} \tag{3.2}$$

- S = Number of Substitution
- D = Number of Deletion
- I = Number of Insertion
- N = Number of characters in the ground truth

To obtain the overall CER percentage, we need to take the mean of the CER values for all word pairs.

TH-3777_186155001

BLEU Score

BLEU [111] score is nothing but the multiplication of the Brevity Penalty and the Geometric Average of the n-gram precision scores. Bleu score can be calculated for different values of n gram. In our case, we have considered precision scores up to 4 gram. Thus, the Geometric Average Precision Score is the geometric average precision of n gram (4 gram in our case). The brevity penalty is used to penalize predicted words/sequences that are too small. Equations for calculating the geometric average precision score, the brevity penalty, and the BLEU score are given below.

$$\begin{aligned} Geometric\,Average\,Precision\,(N) &= \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \\ &= \prod_{n=1}^{N} p_n^{w_n} \\ &= (p_1)^{\frac{1}{4}} \times (p_2)^{\frac{1}{4}} \times (p_3)^{\frac{1}{4}} \times (p_4)^{\frac{1}{4}} \end{aligned} \tag{3.3}$$

- We take N = 4 and uniform weights $w_n = \frac{1}{4}$ in our case.

$$Brevity\,Penalty = \begin{cases} 1, & if\, c > r \\ e^{(1-r/c)}, & if\, c <= r \end{cases} \tag{3.4}$$

- c is the predicted length = number of characters in the predicted word
- r is the target length = number of characters in the target word

Finally, we will calculate the Bleu score as follows.

$$BLEU(N) = Brevity\ Penalty \times Geometric\,Average\,Precision\,Scores\,(N) \tag{3.5}$$

Character F-score (CHRF)

CHRF [112] computes the average F-score over character-level n-grams, combining precision and recall across multiple n-gram lengths (typically up to 6). It is more tolerant to

TH-3777_186155001

minor spelling variations and effectively evaluates transliteration quality at the character sequence level:

$$CHRF = 100 \times \frac{1}{N} \sum_{n=1}^{N} \frac{(1+\beta^2) \times P_n \times R_n}{\beta^2 \times P_n + R_n} \tag{3.6}$$

where P_n and R_n are the precision and recall for character n-grams, N is the maximum n-gram order (commonly 6), and β is a recall-weighting parameter (set to 2 in our case).

Other Metrics

In later chapters, additional complementary metrics such as *Accuracy@k*, *Character Accuracy*, *Mean F-score*, and *Mean Reciprocal Rank (MRR)* are used. These provide a more detailed understanding of model performance in terms of ranking, precision-recall balance, and contextual fluency.

1. Accuracy@k:

$$Accuracy@k = \frac{\text{Number of correct predictions within top } k}{\text{Total number of samples}} \times 100\% \tag{3.7}$$

Accuracy@1 considers only the topmost prediction, whereas Accuracy@4 counts a correct match if the ground truth appears among the top four outputs.

2. Character Accuracy:

$$Character\ Accuracy = \frac{\text{Number of correctly predicted characters}}{\text{Total number of characters}} \times 100\% \tag{3.8}$$

3. Mean F-score:

$$F_i = \frac{2 \times P_i \times R_i}{P_i + R_i}, \qquad Mean\ F\text{-}score = \frac{1}{N} \sum_{i=1}^{N} F_i \tag{3.9}$$

where P_i and R_i are the precision and recall for the i-th sample. Unlike CHRF, which aggregates n-gram level scores, Mean F-score operates at the instance level.

4. Mean Reciprocal Rank (MRR):

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{r_i} \tag{3.10}$$

TH-3777_186155001

Here r_i denotes the rank position of the correct transliteration among the predictions for the i-th test instance. A higher MRR value indicates that correct predictions tend to appear near the top of the ranked list.

Together, these metrics offer a comprehensive framework for evaluating transliteration performance. While *WER* and *CER* quantify fine-grained character-level and word-level errors, *CHRF* and *BLEU* measure the overall linguistic accuracy and fluency of the generated outputs. In contrast, the ranking-based metrics such as *Accuracy@k* and *MRR* capture the reliability and contextual relevance of model predictions. While the task is referred to as "word-level transliteration," each word is internally split into its constituent characters for training the sequence-to-sequence models. This allows the models to learn character-level mappings rather than treating each word as a single atomic unit. Consequently, WER is meaningful as it measures the proportion of words that are fully correctly transliterated, CER evaluates errors at the character level within each word, and n-gram BLEU is computed over the character sequences of the predicted and reference words. In other words, WER captures word-level correctness, CER captures finer-grained character-level errors, and character-level BLEU provides an n-gram overlap measure, making all these metrics relevant for assessing model performance on character-sequence representations of words.

3.5 Experimental Setups

Our word-level transliteration experiments were conducted across four distinct setups: (1) Statistical Transliteration Setup, (2) Neural Network-Based Transliteration Setup, (3) Evaluation utilizing Pre-trained Transliteration Models, and (4). Transfer learning setup by fine-tuning existing state-of-the-art large language models (LLMs) with our word-level transliteration objective. In the subsequent discussion, each of these experimental setups will be briefly outlined.

1. **Statistical Machine Transliteration Setup:** In our statistical machine transliteration configurations, we employed two state-of-the-art statistical models for transliteration. One incorporated the phrase-based statistical machine transliteration model with the Moses decoder, employing GIZA++ [113] for character alignment and KenLM [114] for language modeling. The other model employed the joint n-gram-based string transduction system, SEQUITUR [102]. Both systems were trained up to 4-grams for the language model, and their respective performances were reported.

TH-3777_186155001

2. **Neural Network Based Transliteration Setup:** We mainly employed two state-of-the-art neural network based sequence-to-sequence models. One was implemented using the sequence-to-sequence BiLSTM encoder-decoder model with attention with the help of the OpenNMT toolkit [115] and the other using the Fairseq [116] implementation of neural transformer model.

    - **Experimental Setup for the BiLSTM with Attention Model:** For the BiLSTM model, the encoder is made up of two BiLSTM layers, each with 512 hidden units. The whole network processes words (split into characters) in batches of 128 embeddings, 64 training batches, and 32 validation batches, with an initial learning rate of 1 and a decay rate of 0.5 over 10,000 steps out of 200,000. BiLSTM dropout [117] and attention dropout with early stopping criterion with values of 0.3, 0.1, and 10 were used to reduce overfitting. Similarly, the decoder network includes 512 RNN [87] hidden units with 2 stacked on the decoder side. We use the Adagrad Optimizer [118] to train the sequence-to-sequence system and normalize the gradient to prevent memory growth during training. At every 10,000 steps, we set a checkpoint.
    - **Experimental Setup for the Neural Transformer Model:** The Transformer model architecture consists of 6 encoder and 6 decoder layers, incorporating layernorm within each transformer layer for output normalization. The GELU activation function [119] is applied, and both encoder and decoder self-attention layers utilize 4 attention heads. Parameters include a batch size of 1024, 256 dimensions for encoder and decoder embeddings, and 1024 dimensions for encoder and decoder feed-forward networks (FFN). Dropout rates of 0.5 and 0.1 are used, and label-smoothed cross-entropy with a smoothing factor of 0.1 serves as the training criterion. The Adam optimizer [120] is employed with betas (0.9, 0.98) and a learning rate of 0.001. The learning rate scheduler is inverse square root with a warmup initialization learning rate of 0 and warmup updates of 4000. The model is trained for a maximum of 100 epochs.

3. **Pre-trained transliteration Model Evaluation:** We evaluated the transliteration performance of three publicly available pre-trained transliteration models directly on our test set, including the Google Transliteration API, the indic-trans transliteration model, and the multilingual transliteration model, IndicXlit.

4. **Transfer Learning Setup:** Additionally, we fine-tuned three models: the multilingual transliteration model IndicXlit, and two transformer-based large language models, the multilingual pre-trained text-to-text transformer model, mT5 and ByT5,

TH-3777_186155001

Table 3.3: Transliteration results in terms of WER, CER, substitution, insertion, deletion, and BLEU (up to 4-gram) for all 10 setups. Best WER, CER, and BLEU are highlighted in **bold**.

Model Setup	**Statistical**		**Neural**		**Pre-trained**			**Transfer Learning**		
Metric	**Setup 1: Phrase-based Statistical Moses**	**Setup 2: Joint Source-Channel Sequitur**	**Setup 3: BiLSTM-Attention**	**Setup 4: Neural Transformer**	**Setup 5: IndicXlit**	**Setup 6: Google Transliteration API**	**Setup 7: indic-trans**	**Setup 8: IndicXlit Model**	**Setup 9: Google mT5 Model**	**Setup 10: Google ByT5 Model**
WER (%)	66.78	63.99	58.90	**55.05**	73.38	58.79	95.11	63.53	76.38	66.36
CER (%)	23.04	21.34	19.76	**19.44**	29.91	24.01	54.62	21.10	31.69	22.94
Substitution	9146	8593	9728	8129	12265	9625	24507	8924	12633	8996
Insertion	3093	3038	3311	2889	2946	3289	4021	2868	4340	3085
Deletion	3825	3258	3911	3289	5639	3821	9550	2921	5119	3911
BLEU	64.41	67.33	68.60	**69.15**	54.03	67.61	24.04	66.48	55.50	65.93

a tokenizer-free variant of mT5, representing a token-free byte-to-byte pre-trained transformer model. Both pre-trained models were trained on the multilingual variant of the C4 [121] dataset, mC4 [19] covering 101 languages. The pre-trained objective of the IndicXlit model aligns with ours, focusing on **"Word Level Transliteration"** only, and our target language, Assamese, is also present in the training set. In contrast, the pre-training objective for the other two models is **"Span Corruption"**, and our target language, Assamese, is not part of the training set in either of these two pre-trained models. For our evaluation, we utilized the mT5-small[15] and ByT5-small, pre-trained transformer models with 300 million parameters from the Hugging Face library, running both models for 51 epochs each.

[15] mt5-small

TH-3777_186155001

Experimental Dataset:

In our word-level transliteration task, we maintained uniformity in the training, validation, and test sets across all ten different setups. To ensure a thorough evaluation, we employed a 70-10-20 split. Out of the total 65,614 unique transliteration pairs, the training data includes 45,934 pairs, the validation set contains 6,560 pairs, and we evaluated the models with 13,120 pairs in the testing set.

3.6 Result and Analysis

In this section, we assess the performance of various transliteration setups using specialized evaluation metrics designed for word-level transliteration tasks. Given the nature of word-level transliteration, metrics such as Word Error Rate (WER), Character Error Rate (CER), the counts of substitution, deletion, and insertion errors, along with BLEU score [111] (up to 4 grams), provide insightful measurements. WER represents the percentage of correctly predicted word pairs out of the total word pairs. CER, based on Levenshtein distance [110], measures the minimum edits (substitution, insertion, and deletion) needed to transform a predicted word into the actual ground truth word. The BLEU score combines the Brevity Penalty and the Geometric Average of n-gram precision scores. We calculated BLEU scores up to 4 grams for each setup in this paper. Among the ten selected setups, those involving neural network-based models outperformed other baselines. Specifically, the setup with the neural transformer model (setup 4) achieved the lowest Word Error Rate (WER) and Character Error Rate (CER) values, along with the highest BLEU (up to 4 gram) score of 55.05, 19.44, and 69.15, respectively. The result of all the ten experimental setups are presented in Table 3.3.

(1). Statistical Model Setup: Within the statistical model setups, the Sequitur implementation of the joint n-gram-based transliteration model (setup 2) exhibited superior performance compared to its counterpart, the Phrase-based statistical transliteration model using Moses (setup 1), as indicated in the Table 3.3.

(2). Neural Model Setup: In the configurations featuring neural network-based transliteration models, the one employing the neural transformer model (setup 4) outperformed the BiLSTM model with attention (setup 3) in terms of Word Error Rate (WER), Character Error Rate (CER), and the BLEU (up to 4 gram) accuracy score, as depicted in Table 3.3.

(3). Pre-trained Model Evaluation Setup: Among the pre-trained models, the configuration utilizing the Google Transliteration API (setup 6) achieves comparable results

TH-3777_186155001

with the best-performing neural transformer model (setup 4). It also demonstrates the highest performance among all three pre-trained model setups. However, the multilingual pre-trained transliteration model IndicXlit (setup 5), primarily trained on canonical transliteration pairs, encounters challenges with noisy transliteration pairs from social media. It is worth noting that most pre-trained transliteration models were trained on carefully annotated clean/canonical transliteration pairs, mapping from the native Assamese word to its respective romanized Assamese counterpart in the forward transliteration direction. In contrast, the pre-trained transliteration model indic-trans (setup 7) exhibits the lowest performance among all ten experimental setups, as indicated in Table 3.3.

(4). Transfer Learning Setup: In the transfer learning setup, we fine-tuned three pre-trained models. One utilized the transformer-based multilingual pre-trained transliteration model, IndicXlit, trained on 22 Indic languages, including our target language Assamese in their training set. Additionally, we employed two multilingual transformer-based pre-trained large language models trained on 101 languages from the mC4 dataset, where Assamese is not part of the training set. Furthermore, their pre-training objective, **"Span Corruption"** differs from our "**word-level transliteration**" objective. In contrast to the character-level embedding used in earlier setups, the mT5 model (setup 9) employs pre-trained SentencePiece embedding to create the vocabulary. On the other hand, the ByT5 model (setup 10) follows a language-agnostic approach, directly processing UTF-8 bytes without any text pre-processing. These bytes are embedded into the model's hidden size using a vocabulary of 256 possible byte values. Since ByT5 operates on the byte level rather than the character or sub-word level, it doesn't maintain a fixed vocabulary size, enabling it to process text in any language, given that characters in any language have unique UTF-8 byte values. Among the three transfer learning setups, the fine-tuned IndicXlit model achieves the highest performance in terms of Word Error Rate (WER), Character Error Rate (CER), and BLEU (up to 4 gram) accuracy score, as depicted in Table 3.3.

Error Analysis: Challenges in Social Media Transliteration:

Processing social media text poses several challenges, including generic transliteration issues and those arising from the inherent nature of noisy social media content. The challenges are amplified by users not adhering to standard transliteration guidelines while writing Assamese using the Roman script on social media. Additionally, English and Assamese are orthographically distinct languages, lacking a direct one-to-one correspondence between their graphemes. Many social media users are comfortable with the English

TH-3777_186155001

Table 3.4: Comparison between the outputs of ten different transliteration model setups with the same Roman input (output of the setups that match with the ground truths are highlighted in blue)

Different Setups	Roman Input and Actual Native ground truth		Statistical Model Setups		Neural Model Setups		Pre-trained Model Evaluation Setups			Transfer Learning Setups		
Different Obsevations	Roman Input	Assamese Native	*Moses output (setup1)*	*Sequitur output (setup2)*	*BiLSTM output (setup3)*	*Transformer output (setup4)*	*IndicXlit model output (setup5)*	*Google Transliteration API output (setup6)*	*indic-trans model output (setup7)*	*Fine-tune IndicXlit model output (setup8)*	*Fine-tune mT5 model output (setup9)*	*Fine-tune ByT5 model output (setup10)*
Multiple Character Mapping	hani	শনি	হানি	হানি	শনি	সানি	হানি	সানি	হানী	হানি	সানি	হানি
	xuola	শুৱলা	শুলা	সোলা	খুৱলা	খোৱালা	সোঁৱলা	শুৱলা	জুওলা	শুৱলা	সোলা	শুৱলা
Short form Representation	jrhtt	যোৱহাটত	যোৱহাতত	যোৱহাতত	যোৱহাতত	যোৱহাটত	জাৰ্ট	হৰ্ষদ	জৰত্ত	যোৱহাট	যোৱহাতত	যোৱহাতত
	bhtor	বহুতৰ	বহঁতৰ	বহঁতৰ	বহঁতৰ	বহুতৰ	ভটৰ	ভাতৰ	ভট্টৰ	ভাতৰ	ভিতৰ	ভটৰ
Long form Representation	aaauuu	আওঁ	আওঁ	আ	আওঁ	আওঁ	আওঁওও	আআওওও	আউ	আও	আও	আআও
	bapppaaoiii	বাপ্পাঐ	বাপপায়	বাপ্পাও	বাপপ্পাওঁ	বাপপ্পাওঁ	বাপ্পাআওই	বাপ্পাপায়ী	বাপ্পাওঈ	বাপ্পায়	বাপ্পাঐ	বাপ্পাঐ
Alphanumeric Word	ai2e	এইটোৱে	এইটো	এইটো	এইটোৱে	এইটোৱে	এআইআইচে	অং২য়ে	আই2এ	এইটোৱে	এইটোৱে	এইটোএ
	kn2	কিন্তু	কিন্তু	কোনতো	কিন্তু	কিন্তু	কেএনচি	কঁ২	ন2	কিন্তু	কিন্তু	কিন্তু

QWERTY keyboard, leading them to phonetically transcribe Assamese using Roman alphabets based on their own phonetic judgment. Transliteration models often struggle to address these challenges, leading to errors. In the subsequent discussion, we will discuss some of those challenges encountered while processing social media text. The observations are categorized into four different categories, outlined in Table 3.4, alongside some sample examples. Model outputs that align with the ground truths are highlighted in blue.

(1). Multiple character mapping: Due to the absence of direct one-to-one correspondence between English and Assamese graphemes, coupled with the lack of adherence to a common standard transliteration guideline, we have observed instances where a single Roman grapheme represents multiple Assamese graphemes (one-to-many mapping). Conversely, a single Assamese grapheme may exhibit various Roman variations (many-to-one mapping). In our dataset, for instance, the Roman grapheme "r" corresponds to Assamese graphemes "ৰ", "ড়", and "ঢ়" in the transliteration of "ৰং" (transliteration: "**r**ang", meaning: "Colour"), "গড়" (transliteration: "go**r**", meaning: "Rhino") and "বুঢ়া" (transliteration: "bu**r**a", meaning: "Old man"). Similarly, the Assamese grapheme "শ" is transliterated with different Roman graphemes such as "s", "sh", "h", "x", and "kh". For example, in the representation of the Assamese word "শিলাদিত্য", it is transliterated as "**s**iladitya", "**sh**iladitya", "**h**iladitya", "**x**iladitya", and "**kh**iladitto" as evidenced by our dataset. Referencing Table 3.4, we have observed two instances of Assamese words, "শনি" and "শুৱলা", where the Assamese grapheme "শ" is correspondingly represented as "h" and

TH-3777_186155001

"x" in the ground truth. Consequently, due to these multiple character mappings, our models predicted "শ" as "হ", "স", "খ" and "জ" respectively.

(2). Short form representation: In social media, it is a common practice to utilize informal short-form representations by excluding vowels between consonants in Roman transliterations. When individuals write Assamese words using Roman characters on social media, they intentionally omit vowels between consonants to create short-form versions. For instance, the Assamese word "লগত" is transliterated as **"lgt"** in a short form by removing the vowels between consonants. However, the ideal Roman transliteration of the Assamese word "লগত" should be either **"lagat"** or **"logot"** without the short form. Table 3.4 presents two illustrative examples of short-form representations derived from our datasets: **"jrhtt"** ("যোৰহাটত") and **"bhtor"** ("বহুতৰ"). Notably, only the Transformer model in setup 4 yielded outputs that accurately matched the ground truth in both cases.

(3). Long form representation: In social media text, repeating the same character multiple times to emphasize or convey emotions is common. For example, the Assamese word "বহুত" might be transliterated as **"bohuuuuuuuuttttt"** instead of its standard form **"bohut"**. Furthermore, In the Assamese language, the presence or absence of the nasalized character "ঁ" holds significant meaning, conveying different interpretations. For instance, "কাহ" (meaning: cough) and "কাঁহ" (meaning: bell metal) showing this distinction. We offer two examples of long-form representation from our dataset: **"aaauuu"** ("আওঁ") and **"bapppaaoiii"** ("বাপ্পাঐ") in Table 3.4. The presence of the nasalized character "ঁ" is correctly predicted by the statistical model with Moses (setup 1) and both the neural models, namely the BiLSTM with attention model in setup 3 and the Transformer model in setup 4. Similarly, the long-form representation of "বাপ্পাঐ" for **"bapppaaoiii"** is accurately predicted by the transfer learning setups in setup 9 (mT5) and setup 10 (ByT5).

(4). Alphanumeric word: A common practice noted in the transliteration of Assamese social media text involves representing a single word by combining both alphabetic letters and numbers within the same word. For example, the Assamese words "এইটোৰে" and "কিন্তু" are transliterated as **"ai2e"** and **"kin2"** respectively, as demonstrated in Table 3.4 of our dataset. It is worth noting that, except for the pre-trained models in setup 5, setup 6, and setup 7, all the other models effectively accommodate this alphanumeric pattern and correctly predicted the output, as evident in our dataset.

TH-3777_186155001

3.7 Summary and Future Work

In summary, this chapter has introduced a novel back-transliteration dataset for Assamese, specifically designed to address the resource scarcity in reverse transliteration for Indic languages. This dataset reflects real-world complexities beyond structured transliteration pairs by capturing diverse linguistic variations from three major social media platforms—Facebook, Twitter (currently X), and YouTube. Additionally, we conducted a comprehensive evaluation of ten state-of-the-art word-level transliteration models, demonstrating that the Neural Transformer model outperformed others with the lowest Word Error Rate (WER) and Character Error Rate (CER), along with the highest BLEU score (up to 4-gram). These results confirm the efficacy of our dataset in evaluating transliteration models and emphasize its potential to further transliteration research for low-resource languages.

Looking ahead, this research presents various opportunities for future work. Expanding the dataset to include a wider range of linguistic content and additional social media platforms would improve its overall representativeness. Addressing specific challenges—such as variations in informal short-form and long-form transliterations, different character mappings, and managing code-mixed text—could significantly enhance transliteration accuracy. From a modeling perspective, fine-tuning existing pre-trained models on social media transliterations and developing transformer-based architectures tailored for low-resource languages, like Assamese, could lead to further advancements.

This work establishes a crucial foundation for improving transliteration technology, particularly in low-resource and social media contexts. By bridging the transliteration resource gap for Assamese, we contribute to broader cross-lingual communication efforts. Future research will explore sentence-level transliteration, better handling of code-mixed text, and adaptation for other Indic languages. As multilingual digital communication grows, addressing these transliteration challenges remains essential for enhancing linguistic accessibility and inclusivity.

TH-3777_186155001

TH-3777_186155001

Chapter 4

Analysis of Transliteration Characteristics and Variations

In the previous chapter, we introduced a novel back-transliteration dataset to address resource scarcity in Assamese reverse transliteration, laying the groundwork for developing more accurate transliteration models. However, effectively handling Romanized Assamese text from social media requires a deeper understanding of its linguistic complexities. Assamese orthography differs from English orthography, resulting in many instances of one-to-many and many-to-one character mappings. Additionally, specific phonemes in Assamese do not have direct equivalents in English, and the same is true in reverse. This creates difficulties in accurately reconstructing native Assamese words from their Romanized versions. Unlike some other popular Asian languages (say **Pinyin** for Chinese), The absence of standardized transliteration or romanization guidelines further compounds this challenge, resulting in inconsistent spellings across users and platforms. This chapter investigates these issues at grapheme and phoneme levels by analyzing how Assamese vowels, vowel diacritics, and consonants are represented in Romanized form. Using our curated dataset from Facebook, YouTube, and Twitter (now X), we compare user-generated transliterations with six existing Assamese romanization schemes, highlighting variations and inconsistencies. To further explore these challenges, we develop and evaluate three character-level back-transliteration models: (1) a phrase-based statistical machine transliteration (PBSMT) model, (2) a BiLSTM-based seq2seq model with attention, and (3) a Transformer-based neural transliteration model. Through a detailed error and attention analysis, this chapter provides insights into transliteration difficulties and contributes to building more robust Assamese back-transliteration systems for social media text.

TH-3777_186155001

4.1 Introduction

In the previous chapter, we introduced a novel back-transliteration dataset to address the scarcity of resources for Assamese reverse transliteration. This dataset lays the groundwork for a more in-depth analysis of the challenges associated with transliteration. Reverse transliteration of social media text is particularly complex due to the lack of standard transliteration guidelines, as well as significant differences in orthography and phonetics between Assamese and English. Recent reports from organizations like KPMG[1] and Statista[2] indicate that the number of Indian language users contributing content online is rapidly increasing. Projections suggest that by 2021, nearly 75% of Indian Internet users would engage in content creation using Indian languages. While some content is written in native scripts, a substantial portion is transliterated into Roman script, often due to insufficient technological support for typing in Indian scripts. However, the absence of standardized transliteration rules leads to inconsistencies, making reverse transliteration a challenging task.

Like other Indian language content on social networks, Assamese text is often transliterated using Roman orthography, which introduces various linguistic complexities. Assamese and English have distinct orthographic and phonemic systems, leading to many one-to-many and many-to-one character mappings. Additionally, some phonemes in Assamese do not have exact equivalents in English, and vice versa, which makes transliteration inherently ambiguous. Moreover, the noise commonly found on social media, such as elongated forms (e.g., "khuvesshhhaa" for "শুভেচ্ছা," pron: /xub^{h}essɑ/, meaning: "good wishes")) and shortened forms (e.g., "vl" for "ভাল," pron: /b^{h}ɑl/, meaning: "good")[3], further complicates the reverse transliteration process.

Previous efforts have aimed to develop reverse transliteration systems for social media text in several Indian languages, as highlighted in the FIRE shared task 6 [[49], [109]], along with studies by Bhat *et al.* [122] and Chakma *et al.* [123]. However, none of these studies have specifically focused on Assamese. To our knowledge, this is the first study that systematically analyzes the transliteration characteristics of Assamese social media text both orthographic and phonetic levels across platforms such as Twitter, Facebook, and YouTube, and empirically evaluates state-of-the-art reverse transliteration methods. Since romanization is a specialized form of transliteration that employs the Roman script, we will use the terms "transliteration" and "romanization" interchangeably for the remainder of this chapter.

[1] KPMG
[2] Statista
[3] Each noisy Romanized Assamese term is shown with its equivalent native Assamese word, phonetic pronunciation, and meaning.

TH-3777_186155001

Contributions: The key contributions of this chapter are:

1. Analysis of transliteration variations in Assamese social media text at both the grapheme and phoneme levels.
2. Comparison of the different Assamese transliteration schemes reported in different studies with the transliteration characteristics observed in the transliterated Assamese social media texts.
3. Evaluation of state-of-the-art statistical and neural reverse transliteration methods.

4.2 Related Work

Most existing research on transliteration handles the transliteration problem for named entities, out-of-vocabulary words, or technical terms. However, with the rising popularity of social media and the unavailability of native-language keyboard support, most social media users express themselves in their native languages but use transliterated text. Besides transliteration, other social media-specific challenges like code-mixing [52], spelling variations, etc., also increase the complexity of processing social media text. However, the information in social media is vast, and the NLP tools built on plain text do not support this type of noisy social media text. Our research problem also addresses the transliteration problem for Indian social media text, so we must first analyze the characteristics of transliterated social media text. Every language has some unique orthographic and phonetic characteristics. So, before directly jumping into the transliteration problem, it is essential to understand how specific languages are represented in the transliterated social media text. Below, we analyze a few socio-linguistic studies to understand the characteristics of social media text.

4.2.1 Characteristic Analysis of Transliterated Social Media Text

In this subsection, we will review the work on analyzing different characteristics of social media text and the various challenges in processing them.

Subramaniam *et al.,* 2009 [124] reported different types of text noise and different techniques to deal with it. Clark and Araki, 2011 [125] discuss problems with the automatic normalization of social media English, various applications for its use, and their progress by developing a rule-based approach to this problem. Hu *et al.,* 2013 [126] highlighted the

TH-3777_186155001

formal nature of Twitter's language and the behavior of Twitter users compared to SMS and online chat. Asta Zelenkauskaite, 2017 [127] discussed the problem of non-standard typography (NST) in terms of abbreviations and insertions and their relationship to message length for Italian Facebook and mobile SMS. They found more abbreviations in text messages and more inserts in Facebook messages if we neglected the length limit. The listener-to-listener message contains more non-standard typography (NST) than the listener sends it directly to the broadcast station. Dua'a Abu Elhija, 2014 [128] discussed various transliteration methods and the challenges of using Latin scripts to express dialectal Arabic on Facebook. This script is often called Electronic Amiyyas (EA) and is used by young Facebook users in the Arab world. They also noted that electronic writing is common in diglossic languages[4] such as Persian, Arabic, and various languages of the Indian subcontinent such as Bengali, Tamil, Telugu, and Sinhala. They studied the consonantal system and the difference in this system when used in electronic writing among different users in different countries of the Arab world. Nenagh Kemp, 2010 [129] studied the usage and understanding of textism among young people (such as "r" for "are" and "n" for "and" in English). The writing speed of textism messages is faster than that of regular English messages, but the reading time has almost doubled, resulting in more reading errors. They also investigated the origin of textism or how it started. Abbreviations used in text messages can take several forms. Researchers have classified them in a variety of ways, including letter-number homophones (2moro, r), word shortenings (msg), and abbreviations with internal letters omitted (txt), unconventional, frequently pronounced spelling (skool), clippings (goin, hav), acronyms (ttfn: ta ta for now), accent styles (wiv, dunno, gonna) and symbols: @, :-o, :-(

4.2.2 Analysis of phonetic/spelling/lexical variations in the transliterated social media text

In this subsection, we will review the transliteration issues arising from different spelling variations present in the transliterated social media text and different approaches to handling them. Since SMS texts are also a social media text, all sorts of noisy texts are referred to here as social media texts.

Choudhury *et al.,* 2007 [130] conducted a formal investigation into the nature and type of compressions used in SMS messages using a word model based on the hidden Markov model for texting language(TL). The language used in computer-mediated communication, such as chats, emails, and text messages, is called texting language (TL). Social

[4] https://en.wikipedia.org/wiki/Diglossia

TH-3777_186155001

media contains a variety of non-standard tokens. These non-standard tokens are created unintentionally and intentionally by users for various reasons including speed requirements, ease of typing (Crystal, 2008 [131]), emotional expressions, e.g.: "coooool" (Brody and Diakopoulos, 2011 [132]), intimacy and social purpose (Thurlow, 2003 [133]), etc., which makes it more difficult to interpret social information. Liu *et al.,* 2012 [134] proposed a cognitively controlled social media text normalization system that integrates different human perspectives while normalizing the non-standard tokens. The main goal of this work is to automatically convert noisy non-standard tokens observed in a social media text into standard English words. However, they could not solve a few normalization issues: various expressions related to sentiments, such as emoticons (":-D", "X-8"), interjections ("bwahaha," "brrrr"), abbreviations ("lol," "lmao"), etc., and kept them for further investigation. Sapkal and Shrawankar, 2016 [135] proposed a transliteration system that converts transliterated SMS (short message service) into the Indian regional Marathi language. Using a hierarchical clustering technique, they first transformed the SMS slang into plain, standardized text and back-transliterated them into their native Marathi language. Sharf and Rahman, 2017 [136] proposed a model to achieve the lexical normalization of Roman text in Urdu. They also conducted a detailed survey on how social media texts in different languages, such as Chinese, Arabic, Japanese, Polish, Bangla, Dutch, and Roman Urdu, are normalized to ensure consistency. They identified some of the normalization problems present in romanized Urdu text: (1). Words are written in different ways, e.g., the word "**zindagi**" (English meaning: life) is written as **"zindagee"**, **"zindagy"**, **"zaindagee"**, and **"zndagi"** (2). Identically written words that are lexically different, for example, "**bahar**" can be used both ("**outside**" as Hindi) and ("**spring**" as English), and (3). Some spellings that also correspond to valid English words, for example, "**had**" in Roman Urdu (English meaning: limit value) for the English word "**had**." Singh *et al.,* 2018 [137] proposed a normalization system for handling the word variations in a code-mixed social media text. By exploiting the property that words and their variations share similar contexts in a large, noisy text, they captured different variations of words belonging to the same context in an unsupervised manner using distributed representations of words. They trained the Word2Vec embeddings, computed on a 300-dimensional embedding with a skip-gram-10 model [138]. They classified different variations for the Hindi-English pair. These variations are: *Informal transliterations*, *Long Vowel transliteration*, *Borrowed words' pronunciations*, *Accent and Dialect Phonetics*, *Double Consonants*, *Nonphonetic writing*. They also classified the spelling variations due to Informal speech. These variations are: *Elongation*: (e.g.: "cooooooool", "soooooo goood"), *SMS language*: (e.g.: "gud nyt"), *Abbreviations*: (e.g.: "lappy" for "laptop"). Lusetti *et al.,* 2018 [139] proposed a neural encoder-decoder (ED) framework for normalizing Swiss-German WhatsApp messages. They changed the decoding stage of the

TH-3777_186155001

ED model to include target-side language models operating at different levels of granularity: characters and words. Their proposed approach leads to an improvement over the character-level statistical machine translation (CSMT) baseline. For the romanized pair of Bengali and English (Bn-En) languages, Mandal and Nanmaran, 2018 [140] proposed a new architecture for normalizing transliterated words in code mixed data using the Seq2Seq model and the Levenshtein distance [141]. They used Indian Languages Transliteration(ITRANS)[5] normalized form to represent Bengali words. Their first normalization module does the initial normalization and tries to transform the input string closest to the standard transliteration. The first module's output is passed to the second normalization module, which attempts to match the standard transliteration in the dictionary (BN TRANS). The candidate with the best match is returned as the final normalized string. Khan *et al.* 2020 [142] presented a feature-based clustering framework for the lexical normalization of Roman Urdu corpora. Their proposed framework includes a phonetic algorithm called UrduPhone, a string-matching component, and a feature-based similarity function, and they introduced a clustering algorithm called Lex-Var.

4.2.3 Analysis of code-mixing issues related to the romanized noisy informal social media text

In this section, we will review the literature concerning code-mixing issues in social media texts, which we must address during transliteration.

By taking mixed English-Bengali and English-Hindi Facebook messages, Das and Gambäck, 2013 [52] conducted an initial survey to understand the characteristics of code-mixing in social media. They reported various aspects specific to social media, such as code-switching, code-mixing, lexical borrowing, anglicisms, and phonetic typing, making social media text processing difficult. Sakshi Gupta, 2016 [143] in her master's thesis, designed a shallow parsing pipeline for word normalization and language identification for Hindi-English code-mixed social media text (CMST). [144] conducted a study on the transliteration of code-mixed social media text. They employed Roman-to-Devanagari and Devanagari-to-Roman transliteration systems to translate Hindi-English code-mixed Facebook comments to English. To address spelling variations in Hindi words, they utilized the n-best-list generation approach of phrase-based statistical machine transliteration system(PBSMT) for Devanagari-to-Roman. They picked up 3, 5, and 7-best list transliteration of Hindi sentences into the Roman script, further increasing resources for Hindi-English code-mixed translation. Tara and Purnachandran, 2018, [145] surveyed

[5] https://en.wikipedia.org/wiki/ITRANS

TH-3777_186155001

most code-mixing applications and briefly discussed researchers' challenges in this area. They also highlighted NLP techniques and machine learning (ML) algorithms for processing code-mixed data. Sunayana Sitaram, 2019 [146], investigates computational approaches for code-switched speech and language processing in her survey. They reviewed research on code-switching in various speech and NLP applications, including language processing tools and end-to-end systems.

Irvine *et al.* 2012 [147] proposed a system that converts informal Romanized non-standard Pakistani Urdu SMS messages into standard Arabic scripts so that it can be used to an existing downstream processing tool, such as machine translation, which usually trained on well-formed, native script text only. Their model combines word-level and character-level information, enabling it to handle out-of-vocabulary terms. Al-Badrashiny *et al.,* 2014 [74] solved the problem of converting Dialectal Arabic text (DA), which is written using Latin script (called Arabizi) to modern standard Arabic script following CODA (conventional spelling for dialect) Arabic). They used a finite state transducer trained at the character level to generate all possible transliterations for the incoming Arabizi words. Then, they filtered the generated list using a Dialectal Arabic (DA) morphological analyzer. Using a language model, they picked the best choice for each input word. May *et al.,* 2014 [75] also presented a weighted finite-state transducer-based Arabizi-to-Arabic conversion module, equipped with an n-gram language model. Chakma and Das, 2014 [148] employed different supervised approaches to automatically transliterate Bengali-English and Hindi-English code-mixed Indian social media texts. Their methods were based on the joint source-channel (JSC) model and International Phonetic Alphabet (IPA[6]) model considering a trigram-model as a baseline. Experiments showed that their IPA-based approach outperforms the JSC-based counterpart. Desai and Narvekar, 2015 [149] aimed to address different noisy not-in-vocabulary (NIV) data types. The first type, referred to as "Letter" noise, involves words that are intentionally or unintentionally misplaced. This type of noise is the most common on platforms like Twitter and SMS. For instance, the use of "shud" instead of "should" The second type, **"Number Replacement,"**, occurs when numbers are used instead of phonetically similar words. For instance, "2" is often used instead of "to" The third type is **"Slang"**, which refers to the use of internet slang or abbreviations. An example of this would be using "btw" instead of "by the way" The fourth type is **"Letter and Number"**, where users combine letters and numbers to replace phonetically similar words. A typical example is "2moro" used instead of "tomorrow" Dutta *et al.* 2015 [150] used a noisy channel spelling correction model to solve the problem of text normalization in code-mixed Bengali-English

[6] https://en.wikipedia.or/wiki/International_Phonetic_Alphabet

TH-3777_186155001

social media text. The objective of their work is to correct spelling mistakes in English. Their spellchecker model can also solve wordplay, contracted words, and phonetic variations in the code-mixed text. Sharma *et al.,* 2016 [151] studied the shallow parsing problem of Hindi-English code-mixed social media text (CSMT). Using GIZA++, they obtained character alignments between noisy Hindi words in Roman script to normalized Hindi words and then trained a CRF classifier over these alignments. Finally, using the noisy channel framework, they computed the most probable normalized Hindi word in Roman script for a given Hindi word. As a sub normalizer, they used SILPA libindic spell-checker[7] to compute the top 10 normalized words for a given input word. Bhat *et al.,* 2017 [152] proposed an efficient and less resource-intensive strategy to parse Hindi-English code-mixed data. They modeled the problem of normalization and back transliteration of (noisy) Romanized Hindi words as a single transliteration problem. They aim to learn to map standard and non-standard Romanized Hindi word forms to their respective standard forms in Devanagari. Abbas and Sadat, 2017 [88] proposed a character-based neural machine transliteration (NMTR) system for transliterating the Latin or romanized dialectal Arabic known as "Arabizi" into "Modern Standard Arabic". They mainly focused on the Algerian Arabic dialect. Their transliteration method consisted of two essential steps: 1) the construction of the Arabizi corpus and 2) the neural transliteration of Arabizi into Arabic. Younes *et al.,* 2018 [89] also proposed a deep learning-based Sequence-to-Sequence approach to perform a word-level transliteration of the user-generated Tunisian dialect on the social media(Arabizi), in both Latin to Arabic and Arabic to Latin transliteration. They experimented using Vanilla Seq2Seq models and using a Beam search. Singh *et al.,* 2018 [153] presented their task of identifying the language using transliteration in English-Hindi code-mixed data following the approach used by Rosca and Breuel., 2016 [84]. They trained an attention-based sequence-to-sequence Model (Seq2Seq). Their model generates a sequence of Devanagari characters for a given sequence of Roman characters. They investigated several Seq2Seq models, and the best model they reported includes a double-layer RNN bi-directional encoder and a double-layer RNN decoder. Bhat *et al.,* 2018 [122] introduced normalization and back transliteration models with a decoding process adapted for Hindi-English code switch data. They considered both normalization and back transliteration problems as general sequence-sequence learning problems. Their main goal is to learn to map non-standard English and Romanized Hindi words to standard forms in their respective scripts. They used Luong's attention-based encoder-decoder model (Luong *et al.,* 2015 [154]) with global attention to learning. Their three-step decoding process surpasses the first-best decoding and fragment-wise decoding. Riyadh and Kondrak, 2019 [155] described the problem of

[7] https://github.com/libindic/spellchecker

TH-3777_186155001

the Deromanization of Hindi and Bengali code-mixed social media text. Deromanization converts sentences or terms written in romanized form back to their native form. Their proposed approach mainly combines three components: (1). Language identification (2). Back transliteration and (3). sequence prediction. For the Deromanization task, they used three systems: OpenNMT[8] and Sequitur[9] and DTLM[10]. The neural system, OpenNMT, has the best top-1 accuracy in Bengali, whereas Sequitur achieves the best accuracy among the top-10 predictions in both languages. Masmoudi *et al.,* 2019 [156] proposed two models to transliterate Arabizi into Arabic script for the Tunisian dialect. Their first model used a series of transliteration rules to convert the texts of the Tunisian Arabizi dialect into Arabic script. In the second model, transliteration is carried out at both word and character levels. Patel and Parikh, 2020 [157] adopted a rule-based approach for normalizing code-mixed English and Gujarati sentences. Parikh and Solorio [158], 2021, developed a method for normalization and grapheme-to-phoneme conversion of Hindi-English code-switched data in social media. They also released a script-corrected Hindi-English code-switched dataset for named entity recognition and part-of-speech tagging tasks.

We have noticed that most of the earlier research on transliteration and normalization in the social media domain did not explicitly try to understand the characteristics of transliterated text first. However, a thorough investigation of the language pair's different characteristics will help understand the compatibility between the source and target language pairs. Although previous studies reported various social media-specific challenges, language-specific orthographic and phonetic characteristic analysis was lacking in the previous literature. Again, although there are few earlier research works on processing Indian language data in social media, as far as we know, this is the first work on processing Romanized Assamese language data in social media.

4.3 Dataset

Since this chapter examines the transliteration characteristics of Romanized Assamese text collected from three major social media platforms: Twitter (X), Facebook, and YouTube. Our experiments utilize the in-house back-transliteration dataset, "Assamese-BackTranslit", described in sub-section 3.3.

[8] https://opennmt.net/
[9] http://www-i6.informatik.rwth-aachen.de/web/Software/
[10] https://code.google.com/archive/p/directl-p/

TH-3777_186155001

Table 4.1: Dataset Statistics for the transliteration characteristics analysis experiment

Social Media Sources	#posts	#words	#words (total)	#words (Assamese)	#unique Assamese words
Facebook	5304	71,775	2,26,497	1,64,713	27,765
YouTube	5019	63,331			
Twitter	5009	91,391			

For this analysis, we randomly selected approximately 5,000 posts and comments from each of the three social media platform within the "AssameseBackTranslit" dataset, ensuring that all data sources were exclusively in Assamese - either in its native script or Romanized orthography. Table 4.1 provides a detailed overview of the dataset. During transcription, non-native embedded words were retained in their original form, while common social media noise - such as spelling errors, shortened words, and elongated forms - was normalized to standard Assamese orthographic conventions.

4.4 Transliteration variations in Assamese

Assamese is an Abugida [159] writing system, whereas the English writing system is alphabetic. Furthermore, the phonemic inventories of the two languages differ substantially. Unlike Pinyin for Mandarin Chinese [[160],[159]] and the revised romanization for Korean [161], there is no standard romanization scheme for Assamese, resulting in transliteration variations. Hence, in the following subsections, we provide an analysis of transliteration variations at both graphemic and phonemic levels. This section analyses the transliteration variations observed in the dataset reported in Table 4.1.

4.4.1 Transliteration variation analysis in Graphemic level

In the grapheme level analysis, we will study how differently people transliterate each Assamese grapheme with the help of the Roman script present in our dataset. This grapheme-level analysis is carried out purely in the orthographic level without considering the phonemic properties of the graphemes. Assamese has 12 vowel graphemes, 41 consonant graphemes, and 10 vowel diacritics. Considering Assamese is an Abugida writing system, each consonant grapheme is associated with a vowel, /ɔ/ by default. To represent other vowels, diacritics are attached to consonants that replace the default vowel /ɔ/ with the vowel sound represented by the particular diacritic. However, as mentioned

TH-3777_186155001

Table 4.2: Roman transliteration variations for Assamese vowel graphemes and diacritics

Sl. No.	Vowel Graphemes and Diacritics (with total variations)	Corresponding Phonemes	Total occurrences (in our dataset)	Total Roman Transliteration variations with %
1.	অ (8)	/ɔ/	937	*'a': 55.5%, 'o': 42.46%, 'ao': 1.07%, 'r': 0.32%, 'e': 0.32%, 'oa': 0.11%, 't': 0.11% , 'k': 0.11%*
2.	অ' (6)	/o/	48	*'o': 70.83%, 'a': 12.5%, 'au': 6.25%, 'ao': 4.2%, 'e': 4.2%, 'ho': 2.02%*
3.	আ (9)	/ɑ/	1448	*'a': 76.7%, 'aa': 20.8%, 'u': 1.17%, 'r': 0.55%, 'aaa': 0.21%, 'ya': 0.21%, 'o': 0.14%, 'ea': 0.07% ,'ia': 0.07%*
4.	া (5)	/ɑ/	14440	*'a': 72.5%, 'aa': 21.2%, 'i': 4.3%, 'ah': 1.2%, 'ha': 0.8%*
5.	ই (6)	/i/	3130	*'i': 54.1%, 'e': 27.9%, 'y': 17.2%, 'ee': 0.57%, 'eeeeee': 0.03%, 'eee': 0.03%*
6.	ি (5)	/i/	8933	*'i': 62.1%, 'e': 17.3%, 'y': 11.7%, 'ee': 7.3%, 'ii': 1.6%*
7.	ঈ (4)	/i/	13	*'i': 53.8%, 'ei': 30.8%, 'eaa': 7.7%, 'e': 7.7%*
8.	ী (5)	/i/	1878	*'i': 66.1%, 'e': 19.2%, 'y': 8.7%, 'ee': 4.3%, 'ye': 1.7%*
9.	উ (6)	/u/	417	*'u': 89.7%, 'o': 6.2%, 'w': 3.4%, 'oo': 0.24%, 'uo': 0.24%, 'yo': 0.24%*
10.	ু (6)	/u/	3135	*'u': 83.1%, 'o': 11.2%, 'uu': 1.8%, 'oo': 1.5%, 'ou': 1.3%, 'w': 1.1%*
11.	ঊ (1)	/u/	1	*'u': 100% (no variations)*
12.	ূ (3)	/u/	286	*'u': 86.3%, 'o': 11.3%, 'oo': 2.4%*
13.	ঋ (1)	/ri/	2	*'ri': 100% (no variations)*
14.	ৃ (4)	/ri/	189	*'ri': 89.1%, 'i': 5.6%, 're': 2.8%, 'ir': 2.5%*
15.	এ (9)	/e/	1179	*'a': 48%, 'e': 38.3%, 'ae': 5.2%, 'aa': 5%, 'ea': 2.9%, 'i': 0.17%, 'aaa': 0.17%, 'ey': 0.08% , 'aaaaaa': 0.08%*
16.	ে (8)	/e/, /æ/	7663	*'e': 73.7%, 'a': 18.9%, 'ae': 5.1%, 'ea': 1.4%, 'ee': 0.4%, 'ey': 0.3%, 'ay': 0.11%, 'ei: 0.09%*
17.	ঐ (4)	/oi/	10	*'oi': 50%, 'oiii': 20%, 'ai': 20%, 'oii': 10%*
18.	ৈ (6)	/oi/	891	*'oi': 63.1%, 'ai': 31.3%, 'oy': 2.7%, 'e': 1.2%, 'oe': 1.1%, 'ue': 0.6%*
19.	ও (5)	/ʊ/	1321	*'u': 46.9%, 'o': 39.9%, 'w': 7.9%, 'on': 3.6%, 'wo': 1.6%*
20.	ো (3)	/ʊ/	4181	*'u': 51.9%, 'o': 43.3%, 'oo': 4.8%*
21.	ঔ (7)	/ou/	8	*'u': 25%, 'ou': 12.5%, 'ouu: 12.5%', 'ouuu': 12.5%, 'o': 12.5%, 'ow': 12.5%, 'ooou': 12.5%*
22.	ৌ (6)	/ou/	101	*'ou': 33.1%, 'ow': 22.4%, 'au': 21.4%, 'o': 11.7%, 'u': 7.2%, 'w': 4.2%*

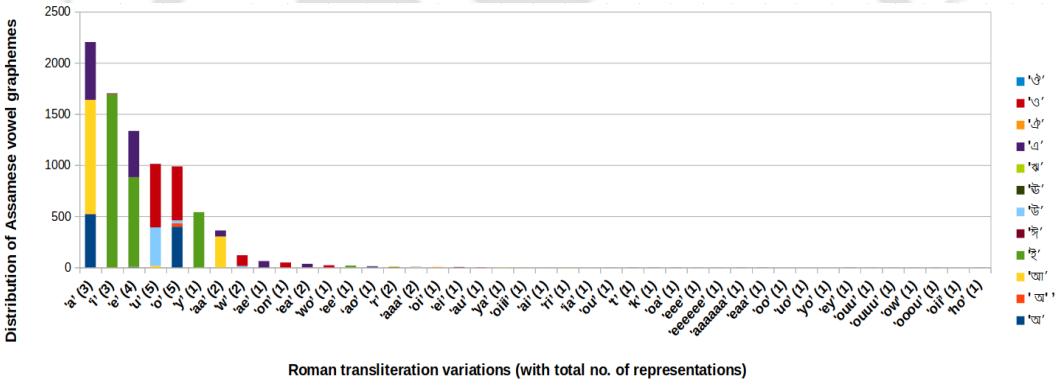

Figure 4.1: Distributions of Roman characters to represent Assamese vowel graphemes.

before, in the transliteration of Assamese texts, variations in transliteration in consonants and vowels are abundant.

Assamese Vowel Graphemes and Vowel Diacritics:

Table 4.2 shows the transliteration variations of Assamese vowel graphemes and their diacritics observed in the dataset. Some of the key observations of Table 4.2 are listed below.

1. It is observed that the occurrences of the vowel graphemes are less than their diacritics. As mentioned above, vowel graphemes appear only when there is no onset in a syllable, which is rare in Assamese.

TH-3777_186155001

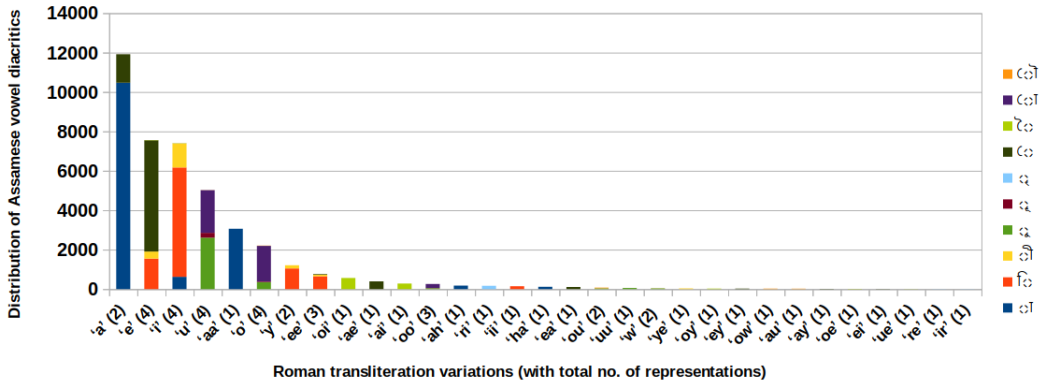

Figure 4.2: Distribution of Roman characters to represent Assamese vowel diacritics.

2. Similarly, fewer transliteration variations are also observed for the vowels compared to their diacritics.

3. Though a large number of transliteration variations are observed for both vowels and their diacritics, the frequency distributions are skewed, i.e. only a few English graphemes dominate. For example, in Table 4.2, although **"অ"** is transliterated with eight different English graphemes, almost 98% of the occurrences come from either "a" or "o". Rare use of the other graphemes may be due to unintended transliteration noise, generally observed in social media text.

4. Many to many transliteration mappings between Assamese and English graphemes are observed. For example, English graphemes "a" and "o", which are used to transliterate the Assamese grapheme **"আ"**, are also used to transliterate the Assamese graphemes **"আ"**, **"অ"**, **"এ"**, **"ও"**, and **"ঔ"**, respectively.

5. The vowel graphemes "আ", "এ", and diacritic "ে" have the maximum transliteration variations (9, 9 and 8, respectively).

6. The vowel grapheme "ই" and diacritic "ি" have the maximum occurrences with 36.76% and 34.63%, respectively.

7. Assuming that the low-frequency English transliteration graphemes are due to unintended noise, ignoring the low-frequency graphemes (10% or less in this reporting), the average number of transliteration variations is reduced from 5.5 to 3.

Although we have seen several transliteration variations for the Assamese vowel graphemes and vowel diacritics, it is also seen that the same Roman character is associated with multiple vowels and vowel diacritics. Such associations are observed in Figure 4.1 and Figure 4.2. Each vowel grapheme and diacritic has been represented by unique color codes. Each bar corresponds to usage of a Roman grapheme for transliterating different

TH-3777_186155001

Assamese vowels and diacritics, and the heights of the bar indicate their occurrences. Some of the key observations made from the figures are listed below.

1. A total of 42 different Roman graphemes are observed for the representation of the 12 Assamese vowel graphemes in Figure 4.1, whereas 31 different Roman graphemes are observed for the representation of 10 vowel diacritics in Figure 4.2.

2. On the horizontal axis of Figure 4.1 and Figure 4.2, different Roman graphemes/ digraphs that are used to represent Assamese vowel graphemes and diacritics are shown, and the number inside the bracket represents the number of Assamese graphemes transliterated using that Roman grapheme. It can be seen from the figures that an Assamese grapheme may be transliterated with different Roman graphemes, but many of the Roman graphemes are used to transliterate a single Assamese grapheme.

3. Although 42 different Roman graphemes have been used to represent all the Assamese vowel graphemes, as seen in Figure 4.1, 8 Roman graphemes/digraphs (vowels: “a”, “e”, “i”, “o”, “u”; semivowels: “y”, “w” and digraph: “aa”) cover the 12 Assamese vowel graphemes and dominate with 97.62% of the total occurrences. The overall skewed distribution is clearly visible.

4. Similarly for Assamese vowel diacritics, also, the same 8 Roman graphemes/digraphs (vowels: “a”, “e”, “i”, “o”, “u”; semi-vowels: “y”, “w” and digraph: “aa”) cover all the 10 diacritics, which dominates with 98.2% of the total occurrences.

5. Roman character “a” has the highest occurrences for both vowels and diacritics, with 2203 and 11917 total occurrences, respectively.

6. Roman characters “u” and “o” are present to represent a maximum of 5 different vowel graphemes and “e”, “i”, “u”, and “o” are present to represent a maximum of 4 different diacritics.

7. The unique Roman representation having many occurrences clearly indicates that there are a few Roman graphemes/digraphs that are only seen while representing some particular Assamese vowel graphemes and diacritics. For example, “y” (for ই), “ae” (for এ), “on” (for ঔ), “oi” and “ai” (for ঐ) in our case.

From the above discussion, it can be inferred that to represent Assamese vowel graphemes and diacritics in Roman transliterated space, the 5 English vowels along with graphemes “w”, “y” and digraph “aa” are primarily used. At the same time, one-to-many character

TH-3777_186155001

mapping between few Assamese vowel graphemes, vowel diacritics, and English also exists. Most of the time, their distribution of variations is very sparse, where the dominance of one particular Roman character for the representation of Assamese vowels and diacritics is clearly visible.

Assamese Consonant Graphemes:

Assamese consonant graphemes with their different transliteration variations are shown in Table 4.3. In this orthographic variation analysis, we mainly focus on how a consonant grapheme in isolation is represented in a Roman transliterated space without adding any vowel diacritics. Although people use multiple Roman graphemes for the representation of each consonant, the overall distribution of these variations is sparse, as we have already observed for Assamese vowels and diacritics. Thus, rare/accidental cases of Roman transliteration variations are clearly observed for the representation of Assamese consonant graphemes as well. Few of these key observations made from Table 4.3 are:

1. In addition to consonant graphemes "ঙ", "ণ", "ন", "ম", "ল", "ঃ" and "ঁ", all other 35 consonant graphemes show more than one transliteration variations. But the distributions of these variations are very sparse.

2. The dominance of one particular Roman character for the representation of a few Assamese consonant graphemes is clearly visible.

3. Few accidental or rare Roman variations are also observed for the representation of a few of the consonant graphemes. Assuming that the low-frequency Roman transliteration graphemes are due to unintended noise, ignoring the low-frequency Roman graphemes (10% or less in this report), the average number of transliteration variations is reduced from 4.85 to 1.85.

4. Consonants seem to have smaller variations compared to vowels and diacritics.

5. Among consonant graphemes, "ক্ষ" is having a maximum of 13 different variations whereas grapheme "ৰ" is having the maximum of 7082 occurrences found in our dataset.

6. Consonant graphemes: "খ", "ঘ", "চ", "ছ", "ঞ", "ঠ", "থ", "ধ", "ফ", "ভ", "ৱ", "শ", "ষ", "স", "ক্ষ", "ঢ় ", "য়" and "ং" , all are having dense distribution of variations found in our dataset.

7. While representing a few of the Assamese consonant graphemes, a direct one-to-one character mapping is observed in English. For these cases, we have not found

TH-3777_186155001

Table 4.3: Roman transliteration variations for Assamese consonant graphemes

Sl. No.	Consonant Graphemes (with total variations)	Corresponding Phonemes	Total occurrences (in our dataset)	Total Roman Transliteration variations with %
1.	ক (5)	/k/	6827	*'k': 99.01%, 'c': 0.7%, 'g': 0.25%, 'q': 0.02%, 'x': 0.02%*
2.	খ (7)	/k^h/	1340	*'kh': 88.21%, 'k': 6.94%, 'x': 3.6%, 'h': 0.82%, 'xh': 0.2%, 's': 0.15%, 'c': 0.08%*
3.	গ (2)	/g/	2116	*'g': 98.68%, 'k': 1.32%*
4.	ঘ (2)	/g^h/	272	*'gh': 83.82%, 'g': 16.18%*
5.	ঙ (5)	/ŋ/	177	*'ng': 89.83%, 'g': 7.34%, 'n': 1.15%, 'w': 0.84%, 'o': 0.84%*
6.	চ (6)	/s/	1947	*'s': 77.3%, 'ch': 12.8%, 'sh': 3.85%, 'x': 3.5%, 'c': 2.41%, 'j': 0.14%*
7.	ছ (7)	/s/	1884	*'s': 83.23%, 'ch': 6.42%, 'x': 4.46%, 'sh': 3.02%, 'c': 2.7%, 'j': 0.11%, 'h': 0.06%*
8.	জ (6)	/z/	2084	*'j': 91.3%, 'z': 6.48%, 'g': 1.2%, 's': 0.8%, 'i': 0.11%, 'sh': 0.11%*
9.	ঝ (2)	/z/	3	*'j': 66.67%, 'z': 33.33%*
10.	ঞ (7)	/n/	138	*'n': 71.75%, 'y': 19.56%, 'o': 4.35%, 'ng': 1.45%, 'a': 1.45%, 'ny': 0.72%, 'w': 0.72%*
11.	ট (2)	/t/	1364	*'t' : 93.84%, '2': 6.16%*
12.	ঠ (2)	/t^h/	279	*'th': 86.74%, 't': 13.26%*
13.	ড (1)	/d/	158	*'d': 100% (no variations)*
14.	ঢ (5)	/d^h/	26	*'dh': 76.92%, 'd': 7.7%, 'rh': 7.7%, 'r': 7.69%, 'h': 7.69%*
15.	ণ (1)	/n/	574	*'n': 100% (no variations)*
16.	ত (4)	/t/	5480	*'t': 95.05%, 'th': 4.1%, '2': 0.76%, 'd': 0.09%*
17.	থ (2)	/t^h/	817	*'th': 87.27%, 't': 12.73%*
18.	দ (6)	/d/	2968	*'d': 98.9%, 't': 0.47%, '2': 0.47%, 'th': 0.1%, 'r': 0.03%, 'j': 0.03%*
19.	ধ (2)	/d^h/	939	*'dh': 83.71%, 'd': 16.29%*
20.	ন (1)	/n/	7012	*'n': 100% (no variations)*
21.	প (3)	/p/	3143	*'p': 99.3%, 'f': 0.41%, 'b': 0.29%*
22.	ফ (3)	/p^h/	355	*'f': 85.35%, 'ph': 12.11%, 'p': 2.54%*
23.	ব (5)	/b/	5259	*'b': 98.63%, 'v': 1.06%, 'p': 0.15%, 'w': 0.11%, 'o': 0.05%*
24.	ভ (4)	/b^h/	1326	*'v': 48.42%, 'bh': 42.53%, 'b': 7.1%, 'vh': 1.95%*
25.	ম (1)	/m/	3540	*'m': 100% (no variations)*
26.	য (4)	/z/	617	*'j': 94.33%, 'y': 2.3%, 'z': 2.1%, 'g': 1.27%*
27.	ৰ (3)	/ɹ/	7082	*'r': 99.86%, 'd': 0.11%, 'w': 0.03%*
28.	ল (1)	/l/	5680	*'l': 100% (no variations)*
29.	ৱ (10)	/w/	1760	*'w': 31.6%, 'b': 30.74%, 'u': 20.85%, 'v': 7.9%, 'a': 7.9%, 'o': 0.4%, 'e': 0.23%, 'p': 0.18%, 'yo': 0.1%, 'm': 0.1%*
30.	শ (9)	/x/	1115	*'h': 28.1%, 's': 21.5%, 'kh': 18.2%, 'x': 15.7%, 'sh': 15.1%, 'xh': 1.2%, 'k': 0.18%, 'ch': 0.01%, 'ck': 0.01%*
31.	ষ (8)	/x/	295	*'kh': 48.8%, 'h': 15.2%, 's': 13.5%, 'x': 10.2%, 'sh': 9.5%, 'k': 1.7%, 'xh': 1.01%, 'ch': 0.09%*
32.	স (9)	/x/	2654	*'h': 45.7%, 's': 22.9%, 'x': 13.5%, 'kh': 12.9%, 'sh': 3.01%, 'xh': 0.9%, 'ch': 0.5%, 'k': 0.4%, 'c': 0.19%*
33.	হ (3)	/h/	2788	*'h': 99.75%, 'x': 0.22%, 'w': 0.03%*
34.	ক্ষ (13)	/k^h/	252	*'kh': 58.3%, 'khy': 22.2%, 'ksh': 4.8%, 'k': 4.8%, 'x': 2.8%, 'ky': 2.4%, 'ks': 0.8%, 'kx': 0.8%, 's': 0.8%, 'xh': 0.8%, 'kk': 0.8%, 'shy': 0.35%, 'sh': 0.35%*
35.	ড় (3)	/ɹ/	69	*'r': 88.4%, 'd': 7.2%, 'rh': 4.4%*
36.	ঢ় (5)	/ɹ/	277	*'h': 44.04%, 'rh': 29.6%, 'r': 17.7%, 'hr': 7.6%, 'd': 1.06%*
37.	য় (7)	/j/	2398	*'a': 36.53%, 'o': 21.1%, 'i': 17.9%, 'y': 13.6%, 'e': 10.7%, 'j': 0.09%, 'u': 0.08%*
38.	ৎ (2)	/t/	64	*'t': 98.44%, 'd': 1.56%*
39.	ং (3)	/ŋ/	471	*'ng': 76.4%, 'n': 21.9%, 'm': 1.7%*
40.	ঃ (1)	/h/	6	*'h': 100% (no variations)*
41.	ঁ	/~/	1011	*No specific representation found in our dataset*

any transliteration variations. People generally use an equivalent Roman character alone or a Roman character suffixed with "a" or "o" based on their position in a word. If the alphabet is present in the beginning position of a word, people mostly use equivalent Roman characters suffixed with "o", and if it is present in the middle position, people use equivalent Roman characters suffixed with "a". If the alphabet is present at the end of a word or sometimes for the purpose of word shortening, an equivalent Roman character alone without any suffixes is used.

Although Table 4.3 shows different variations in Roman transliteration for each individual consonant grapheme, we have seen that a single Roman character is also used

TH-3777_186155001

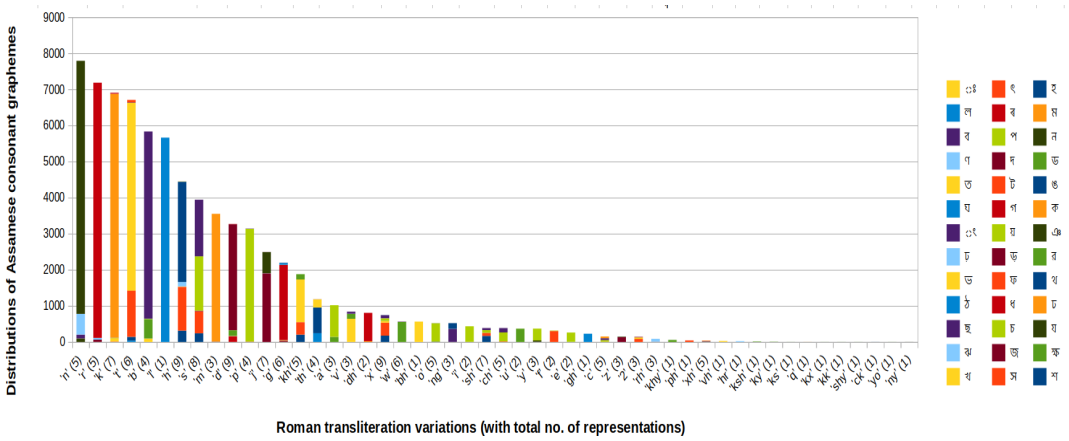

Figure 4.3: Distributions of Roman characters to represent Assamese consonant graphemes

to represent multiple consonant graphemes. Each consonant grapheme shown in Figure 4.3 is represented by different color codes. Each bar corresponds to different Roman variations used to represent Assamese consonant graphemes, and the height of each bar indicates their total occurrences found in our dataset. The different color present in each bar indicates that the same Roman character is also used to represent many different consonant graphemes. Again, the same color present in different bars also indicates that each consonant grapheme has different Roman variations. Thus, both many-to-one and one-to-many character mappings are clearly visible for representing Assamese consonant graphemes. The distribution of Roman transliteration variations are shown in Figure 4.3. Some of the key observations made from Figure 4.3 are:

1. A total of 50 different Roman transliteration variations are observed for the representation of 41 Assamese consonant graphemes.

2. Overall, skewed distributions are observed for the representation of Assamese consonant graphemes with Roman graphemes. i.e. only a few of the Roman graphemes dominate the others.

3. Roman grapheme “n” has a maximum of 7790 overall occurrences, and “h”, “d”, and “x” are present to represent a maximum of 9 different Assamese consonant graphemes.

4. While representing three Assamese consonant graphemes: **“ঞ”, “ৱ”** and**“য়”**, all the 5 English vowels: “a”, “e”, “i”, “o”, “u” and “w”, “y” are mainly used. This is similar to what we have already seen earlier in the representation of Assamese vowel graphemes and diacritics, as well.

TH-3777_186155001

4.4.2 Transliteration variation analysis in Phonemic level

Similar to the graphemic-level transliteration variation analysis, we also perform the transliteration variation analysis at the phonemic level. Although the Assamese language has many different graphemes(vowels, diacritics, and consonants), only a few distinct phoneme sounds are actually present in the language. Out of the 12 vowels, 10 diacritics, and 41 consonant graphemes, only 8 vowels and 23 consonant phoneme sounds are actually present in Assamese. This means that a single Assamese phoneme sound consists of multiple different graphemes with their inherent orthographic properties. But while writing Assamese using Roman script(i.e. transliteration), it is observed that people are mostly concerned about the sound or how a particular grapheme is pronounced. Thus, while transliterating, people are less likely to differentiate between two or more graphemes that correspond to a single phoneme sound. For an orthographically rich language like Assamese, it is actually very hard for a native speaker to distinguish between two different graphemes that correspond to a single phoneme sound. This is also reflected in their transliteration counterpart when writing Assamese using Roman orthography. However, the grapheme-to-phoneme mapping present in English is mainly one-to-one. Thus, many different transliteration variations due to many-to-one mappings along with one-to-many mappings are observed at the phonemic level. Therefore, it is very important to study this analysis of transliteration variations at the phonemic level separately. This study may also help to further build a grapheme-phoneme (G2P) converter system for the Assamese language in the future.

We have observed that people use different variations of the Roman transliteration for the representation of the phoneme sounds present in our dataset. As we can see in Table 4.4, a phoneme sound could be the combination of more than one grapheme and diacritics with similar pronunciations. So, in this phoneme-level analysis, we are basically analyzing only the phoneme sounds present in the Assamese vocabulary. In the following, we briefly discuss the different transliteration variations observed for both Assamese vowel and consonant phonemes, respectively.

Assamese Vowel Phonemes:

Assamese vowel phonemes with their different variations in Roman transliteration are shown in Table 4.4. In this phoneme-level analysis, we mainly focus on how a phoneme sound is represented in Roman transliterated space. We have noticed multiple Roman transliteration variations for the representation of each vowel phonemes, but the overall

TH-3777_186155001

Table 4.4: Roman transliteration variations for Assamese vowel phonemes

Sl. No.	Vowel Phonemes (with total variations)	Corresponding Graphemes	Total occurrences (in our dataset)	Total Roman Transliteration variations with %
1.	/i/(10)	ি, ী, ই, ঈ	13954	*'i': 60.84%, 'e': 19.94%, 'y': 12.53%, 'ee': 5.38%, 'ii': 1.03%, 'ye': 0.24%, 'ei': 0.03%, 'eee': 0.007%, 'eeeeee': 0.007%, 'eaa': 0.007%*
2.	/e/(3)	এ'	23	*'e': 82.6%, 'a': 13.04%, 'i': 4.36%*
3.	/ɛ/(12)	ে, এ	8818	*'e': 68.98%, 'a': 22.8%, 'ae': 5.11%, 'ea': 1.59%, 'aa': 0.67%, 'ee': 0.35%, 'ey': 0.27%, 'ay': 0.11%, 'ei': 0.08%, 'i': 0.02%, 'aaa': 0.02%, 'aaaaaa': 0.01%*
4.	/u/(8)	ু , ূ, উ, ঊ	3839	*'u': 84.06%, 'o': 10.65%, 'uu': 1.46%, 'oo': 1.43%, 'w': 1.30%, 'ou': 1.04%, 'uo': 0.03%, 'yo': 0.03%*
5.	/ʊ/(6)	ো, ও	5502	*'u': 50.69%, 'o': 42.48%, 'oo': 3.67%, 'w': 1.91%, 'on': 0.87%, 'wo': 0.38%*
6.	/o/(2)	অ', consonant + /o/	12576	*'o': 69.72%, 'a': 30.28%*
7.	/ɔ/(10)	অ, consonant + /ɔ/	8673	*'a': 36.06%, 'o': 31.67%, 'w': 15.9%, 'u': 8.41%, 'b': 6.24%, 'v': 1.61%, 'e': 0.05%, 'p': 0.03%, 'yo': 0.01%, 'm': 0.01%*
8.	/ɑ/(12)	আ, া	15888	*'a': 72.88%, 'aa': 21.17%, 'i': 3.91%,'ah': 1.09%, 'ha': 0.73%, 'u': 0.11%, 'r': 0.05%, 'aaa': 0.02%, 'ya': 0.02%, 'o': 0.01%, 'ea': 0.006%, 'ia': 0.006%*

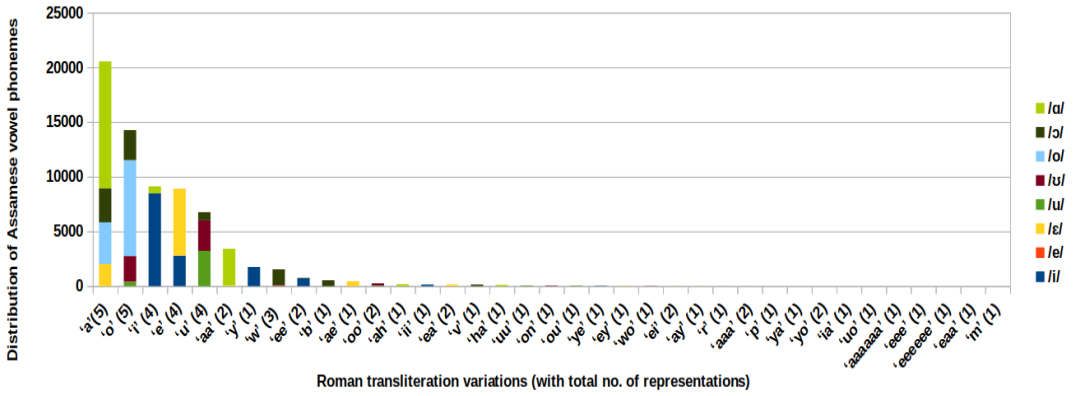

Figure 4.4: Distribution of Roman characters to represent Assamese vowel phonemes.

distributions of these variations are very sparse. Few of the key observations made from Table 4.4 are:

1. We have seen a relatively sparse distribution of variations for the representation of Assamese vowel phonemes compared to vowel and consonant graphemes.

2. The average number of Roman transliteration variations for the representation of 8 Assamese vowel phonemes are 7.88.

3. Ignoring low-frequency Roman graphemes (10% or less in this report), the average number of Roman transliteration variations is reduced from 7.88 to 2.25. Thus, the presence of many rare or accidental Roman variations are clearly visible.

4. The vowel phoneme /ɑ/ has a maximum of 15888 total occurrences.

5. The vowel phonemes /ɑ/ and /ɛ/ have a maximum of 12 different transliteration variations each found in our dataset.

6. For the representation of Assamese vowel phonemes, mainly 5 English vowel graphemes: “a”, “e”, “i”, “o”, “u” along with the English semi-vowel graphemes: “w”, “y” and the digraph “aa” are mainly used.

Like Assamese vowel graphemes and diacritics, a significant overlap is also observed in the representation of each vowel phoneme against each Roman grapheme. This can be easily observed in Figure 4.4. Each vowel phonemes are represented with different color codes. Each bar corresponds to different Roman transliteration variations used to represent Assamese vowel phonemes, and the height of each bar indicates their total occurrences found in our dataset. Similar to our earlier observation, in the case of vowel phonemes also different colors present in each bar indicate that the same Roman grapheme is also used to represent many different vowel phonemes. Again, the same color present across different bars also indicating many different Roman transliteration variations for each Assamese vowel phoneme. Some of the key observations made from Figure 4.4 are:

1. A total of 37 different Roman transliteration variations are observed for the representation of 8 Assamese vowel phonemes.

2. The general sparse distribution of variations is clearly visible. This means that only a few of the Roman graphemes/digraphs are mostly dominant while representing vowel phonemes with Roman graphemes. Other Roman variations are mostly rare or accidental, since their occurrences are also very rare.

3. Both many-to-one and one-to-many character mappings are clearly visible in the representation of Assamese vowel phonemes with Roman graphemes.

4. The unique Roman representation having many occurrences clearly indicates that there are a few Roman graphemes/digraphs that are only seen while representing some particular Assamese vowel phonemes. For example, in our case “aa”, “y”, “ae”, “ea”, and “on”.

5. The digraphs are mainly observed while representing unique Assamese phoneme sounds.

Assamese Consonant Phonemes:

In Table 4.5, we have shown different variations in Roman transliteration for the representation of each consonant phoneme sound present in Assamese from our dataset. It is observed from our dataset that to represent a consonant phoneme, different Roman graphemes, sometimes digraphs or even trigraphs, are generally used. In this section, our

TH-3777_186155001

Table 4.5: Roman Transliteration Variations for Assamese Consonant Phonemes

Sl. No.	Consonant Phonemes (with total variations)	Corresponding Graphemes	Total occurrences (in our dataset)	Total Roman Transliteration variations with %
1.	/p/ (3)	প	3143	*'p': 99.3%, 'f': 0.41%, 'b': 0.29%*
2.	/p^h/(3)	ফ	355	*'f': 85.4%, 'ph': 12.1%, 'p': 2.5%*
3.	/b/(5)	ব	5259	*'b': 98.6%, 'v': 1.1%, 'p': 0.15%, 'w': 0.11%, 'o': 0.04%*
4.	/b^h/(4)	ভ	1326	*'v': 48.4%, 'bh': 42.5%, 'b': 7.1%, 'vh': 0.02%*
5.	/t/(4)	ত, ট, ৎ	6908	*'t': 94.8%, 'th': 3.2%, '2': 1.8%, 'd': 0.2%*
6.	/t^h/(2)	থ, ঠ	1096	*'th': 87.1%, 't': 12.9%*
7.	/d/(6)	দ, ড	3126	*'d': 98.94%, 't': 0.45%, '2': 0.45%, 'th': 0.09%, 'r': 0.03%, 'j': 0.03%*
8.	/d^h/(5)	ধ, ঢ	965	*'dh': 83.5%, 'd': 16.1%, 'rh': 0.21%, 'r': 0.1%, 'h': 0.1%*
9.	/k/(5)	ক	6827	*'k': 99%, 'c': 0.7%, 'g': 0.25%, 'q': 0.05%, 'x': 0.05%*
10.	/k^h/(15)	খ, ক্ষ	1592	*'kh': 83.48%, 'k': 6.6%, 'khy': 3.5%, 'x': 3.4%, 'ksh': 0.75%, 'h': 0.7%, 'ky': 0.37%, 'xh': 0.31%, 's': 0.25%, 'ks': 0.12%, 'kr': 0.12%, 'kk': 0.12%, 'c': 0.06%, 'shy': 0.06%, 'sh': 0.06%*
11.	/g/(2)	গ	2116	*'g': 98.68%, 'k': 1.32%*
12.	/g^h/(2)	ঘ	272	*'gh': 83.82%, 'g': 16.18%*
13.	/ɹ/(6)	ৰ, ড়, ঢ়	7428	*'r': 96.69%, 'h': 1.64%, 'rh': 1.14%, 'hr': 0.28%, 'd': 0.21%, 'w': 0.04%*
14.	/l/(1)	ল	5661	*'l': 100% (no variations)*
15.	/s/(7)	চ, ছ	3831	*'s': 80.21%, 'ch': 9.7%, 'x': 3.9%, 'sh': 3.4%, 'c': 2.6%, 'j': 0.1%, 'h': 0.09%*
16.	/z/(8)	জ, য, ঝ	2704	*'j': 91.9%, 'z': 5.5%, 'g': 1.2%, 's': 0.6%, 'y': 0.52%, 'sh': 0.11%, 'i': 0.11%, 'jy': 0.06%*
17.	/x/(10)	স, শ, ষ	4067	*'h': 38.6%, 's': 21.9%, 'kh': 17.01%, 'x': 13.8%, 'sh': 6.8%, 'xh': 0.96%, 'k': 0.42%, 'ch': 0.4%, 'c': 0.09%, 'ck': 0.02%*
18.	/h/(3)	হ , ঃ	2794	*'h': 99.75%, 'x': 0.21%, 'w': 0.04%*
19.	/m/(1)	ম	3540	*'m': 100% (no variations)*
20.	/n/(7)	ন, ণ, ঞ	7724	*'n': 99.5%, 'y': 0.35%, 'o': 0.08%, 'ng': 0.02%, 'a': 0.02% , 'ny': 0.015%, 'w': 0.015%*
21.	/ŋ/(6)	ঙ, ং	647	*'ng': 80.22%, 'n': 16.23%, 'g': 2.01%, 'm': 1.24%, 'w': 0.15%, 'o': 0.15%*
22.	/w/(9)	ৱ	1780	*'w': 31.72%, 'b': 30.4%, 'u': 20.62%, 'v': 7.9%, 'a': 7.9%, 'o': 0.4%, 'e': 0.2%, 'p': 0.47%, 'yo': 0.39%*
23.	/j/(7)	য়	2398	*'a': 36.53%, 'o': 21.06%, 'i': 17.9%, 'y': 13.6%, 'e': 10.7%, 'j': 0.08%, 'u': 0.04%*

main focus is to study the different Roman transliteration variations against each Assamese consonant phoneme sounds along with their different representational variations observed across Roman transliterated space. Few of the key observations made from Table 4.5 are:

1. In addition to Assamese consonant phonemes /l/ and /m/, all other 21 phonemes show more than one transliteration variation.

2. Although phonemes show many different transliteration variations, their overall distribution of variations is much more skewed.

3. The average number of Roman transliteration variations for the representation of 23 Assamese consonant phoneme sounds is 5.26.

4. Ignoring low-frequency Roman graphemes (10% or less in this report), the average number of Roman transliteration variations is reduced from 5.26 to 1.65. This clearly indicates the presence of many rare or accidental Roman transliteration variations for the representation of the Assamese consonant phoneme sound.

TH-3777_186155001

5. Among consonant phonemes, /k^{h}/ phoneme is having a maximum of 15 different variations.

6. The consonant phoneme /n/ has a maximum of 7724 total occurrences found in our dataset.

7. A few of the unique Roman variations are also observed for the representation of a few of the consonant phoneme sounds. This clearly indicates that there are few unique Roman graphemes that are only seen while representing few of the distinct consonant phoneme sounds.

So, from the transliteration variation analysis of Assamese consonant phoneme sounds, we have found that maximum variations are observed for the representation of those phoneme sounds which do not have either an equivalent direct Roman character mapping or the phonemes which can be represented with the help of more than one character mapping either in the form of digraphs or trigraphs. The distribution of variations is clearly visible in Figure 4.5. Each consonant phoneme sounds are represented by different color codes. Each bar corresponds to different Roman variations used to represent Assamese consonant phonemes, and the height of each bar indicates their total occurrences found in the dataset. The different colors present in each bar indicate that the same Roman character is also used to represent many different vowel phonemes. Again, the same color present in different bars also signifies that each consonant phoneme has many different variations in Roman transliteration. Few of the distinct characteristics observed between Assamese consonant phonemes with their different Roman representations observed from Figure 4.5 are:

1. A total of 51 different Roman transliteration variations are observed for the representation of 23 consonant phoneme sounds.

2. Roman grapheme “n” has a maximum of 7790 overall occurrences, and the graphemes “h” and “w” are present to represent a maximum of five different Assamese consonant phonemes found in our dataset.

3. On the horizontal axis of Figure 4.5, different Roman graphemes/digraphs that are used to represent 23 Assamese consonant phonemes are shown, and the number in brackets represents the number of Assamese phonemes transliterated using a Roman grapheme alone.

4. It can be seen from the figure that an Assamese consonant phoneme may be transliterated with different Roman graphemes, but many of the Roman graphemes are used to transliterate a single consonant phoneme only.

TH-3777_186155001

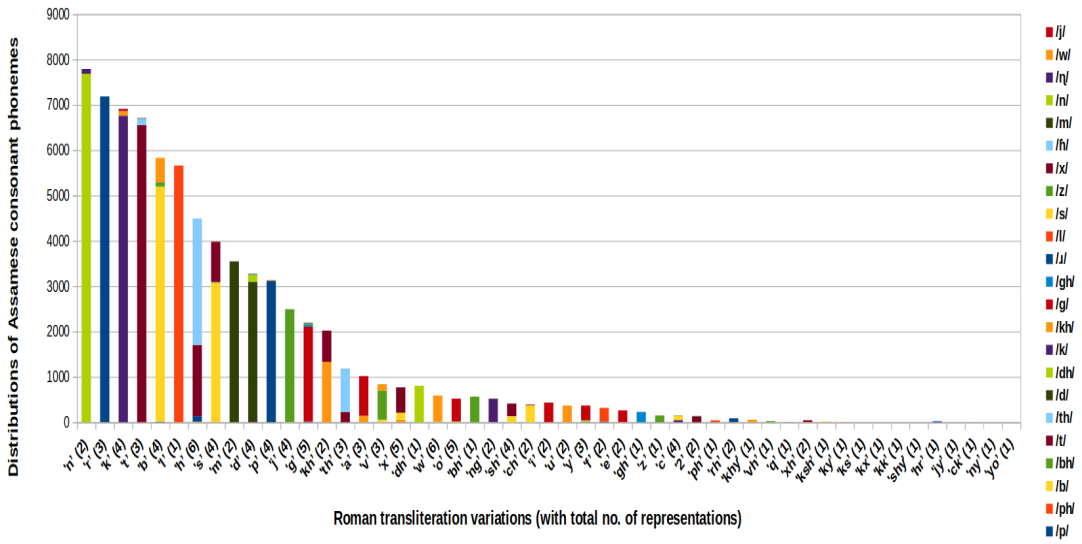

Figure 4.5: Distributions of Roman characters to represent Assamese consonant phonemes.

5. Distribution of Roman graphemes for the representation 23 Assamese consonant phonemes are relatively denser compared to vowel and consonant graphemes and also vowel phonemes. Since out of 51 different Roman graphemes, up to 39 Roman graphemes are required to represent all 23 Assamese consonant phonemes alone, which is 76.47% of the total occurrences.

After observing the variations of the Roman transliteration for the representation of Assamese consonant phonemes and also from the knowledge of the Assamese phonemic inventory, we could draw following key conclusions.

1. The phoneme sound /ʃ/ (which corresponds to the digraph “sh”) is not present in the Assamese phoneme inventory and is often simply realized as the phoneme sound /s/ and on some rare occasions as the English digraphs “sh” or “ch”.

2. Assamese phoneme inventory has two distinct groups of phonemes, aspirated stops: /p^h/, /t^h/, /k^h/, /b^h/, /d^h/, /g^h/ and unaspirated stops: /p/, /t/, /k/, /b/, /d/, /g/. However, in English, they are not phonemes, but are merely considered as allophones. Voiceless stop phonemes (/p/,/t/ and /k/) are realized as aspirated phones like ([p^h],[t^h],[k^h]) and voiced stop phonemes /b/, /d/ and /g/ are not aspirated. Hence to map these two distinct Assamese phoneme groups using the Roman script, either a single Roman grapheme or digraphs are used.

3. Assamese phoneme sound /x/ (corresponding to “শ”, “ষ” and “স”) which is not present in the phonemic inventory of English.

TH-3777_186155001

4. In English, phoneme /f/ is a voiceless labiodental fricative and phoneme /v/ is a voiced labiodental fricative. But in Assamese, these two phoneme sounds are not present. In Assamese, there are two bilabial plosive phonemes /p^h/ ("ফ") and /b^h/ ("ভ") is present. In Roman transliterated Assamese most of the time /p^h/ and /b^h/ phoneme is represented using the digraph "ph" and "bh" followed by "f" and "v" respectively.

5. In the Assamese phoneme inventory, phoneme sounds /x/ ("শ", "ষ" and "স") and /k^h/ ("খ" and "ক্ষ") are often realized as allophones of each other. Significant overlap is also observed between the phoneme sounds /x/ ("শ", "ষ" and "স") and /s/ ("চ" and "ছ"). Thus, significant transliteration variations and character overlap are observed between the representation of phoneme sounds /x/, /k^h/ and /s/ and also in the representation of these graphemes "শ", "ষ", "স", "খ", "ক্ষ", "চ" and "ছ".

6. The Assamese phoneme sound /z/ can be represented by Roman graphemes "j" or "z" based on the graphemes "জ", "য" and "ঝ" all corresponds to the phoneme sound /z/.

7. Assamese phoneme sound /ɹ/ corresponds to 3 graphemes: "ৰ", "ড়" and "ঢ়", while representing in Roman transliterated form, "ৰ" is represented by "r" only but for the representation of "ড়" and "ঢ়" both grapheme "r" and digraph "rh" is used.

Remarks: From the above graphemic and phonemic level analysis, the following points can be noted:

- Compared to graphemes, there are fewer phoneme sounds present in Assamese. Thus, the overlap between Roman transliteration variations for the grapheme representations is greater compared to phonemes.
- When transliteration is performed, people are mostly concerned with how the graphemes are actually pronounced, i.e. the phonemic properties, rather than considering their orthographic or graphemic properties.
- Graphemic and phonemic mappings shown in Table 4.2, Table 4.3, Table 4.4 and Table 4.5 may be useful for creating a common standard Assamese transliteration guidelines.

TH-3777_186155001

4.5 Comparison between Social Media Transliteration and Some Standard Assamese Transliteration Schemes

As mentioned in the previous sections, the Assamese language does not have a common standard romanization scheme for writing Assamese native text using Roman script. Therefore, while writing in social media, Assamese users also do not follow any transliteration guidelines. However, few of the earlier studies and in few applications have adopted different romanization standards to write Assamese using Roman script along with other Indian languages. But none of them is formally recognized as a standard romanization scheme for writing Assamese. We have identified a total of 6 of such standard transliteration schemes. We then performed a detailed comparative analysis between the transliteration variations observed in the social media dataset and the 6 standard transliteration schemes identified for the Assamese language. These are mainly: (1). Assamese online dictionary Xobdo[11], (2). IT3 transliteration standard is jointly administered by IISC Bangalore and CMU [162], (3). ISO 15919 indic transliteration standard [12], (4). ASR 11 transliteration standard given by [163], (5). Microsoft indic input standard [13] and (6). WX notation standard [14]. We performed a detailed similarity analysis between the standard Roman transliteration of Assamese vowels, vowel diacritics, and consonant graphemes against the social media transliteration observed in our dataset. From our observation, we have seen that each of these 6 transliteration standards follows a definite one-to-one and very rarely one-many character mappings present between the native Assamese and its Roman phonetic equivalent. However, we hardly notice any one-to-one character mappings present in our social media dataset. Out of 41 graphemes only for 6 graphemes: "ড", "ণ", "ন", "ম", "ল" and "ঃ", We have seen one-to-one character mapping between native Assamese and Roman graphemes present in our dataset. Similarly, we have not seen any one-to-one character mapping between Assamese vowels and diacritics. Again, out of the 23 Assamese consonant phonemes, only for (/m/: "ম") and (/l/: "ল"), we have seen one-to-one character mapping present in the dataset between native Assamese and Roman graphemes. Thus, while doing the similarity analysis, only for the graphemes: "k", "p", "m", "r" and "l", the equivalent Roman character mapping is the same throughout all the 6 standard transliteration systems and also with our dataset.

[11] http://www.xobdo.org/article/ts2013

[12] https://en.wikipedia.org/wiki/ISO_15919

[13] https://www.microsoft.com/en-in/images/downloads/Assamese%20Indic%20Input%203-User%20Guide.pdf

[14] http://calts.uohyd.ac.in/calts/convert-frame.html

TH-3777_186155001

Table 4.6: Similarity Analysis Between Assamese Social Media Transliteration Against the Six Transliteration Standards

Assamese Graphemes	Xobdo, Assamese Online Dictionary	IT3 standard	ISO 15919, standard	ASR11, standard	Microsoft, Indic input standard	WX, notation
Vowel	**0.4771**	0.4563	0.3053	0.5934	0.4563	0.4201
Vowel diacritics	**0.6360**	0.5934	0.2027	0.5929	0.5963	0.3743
Consonants	**0.8591**	0.7669	0.7643	0.7720	0.7713	0.5317

For the Assamese phoneme set, we have not found a single phoneme, which is the same throughout all the 6 transliteration standards and also in our dataset.

We have calculated the Support, Confidence, and Joint Probability Distribution Scores for each pair of transliterations. Support is the co-occurrence frequency of both the Assamese grapheme and its Roman transliteration equivalent in the dataset, whereas the confidence score is the conditional probability of occurrence of a Roman transliteration equivalent for a given Assamese grapheme. The joint distribution is the multiplication of these two probability scores. We have found that for all Assamese vowel graphemes, vowel diacritics, and consonant graphemes, the transliteration standard adopted by the Assamese online dictionary, Xobdo gives the maximum similarity score of 0.4771, 0.6360 and 0.8591 respectively. This can be highlighted in Table 4.6.

From our analysis above, we have found that among the 6 transliteration standards, not a single standard can alone be capable of representing the transliterated Assamese social media data. Since the similarity score is very low, we can infer that social media users do not follow a common standard transliteration guideline while writing native Assamese on social media with the help of Roman script. It has also been observed from the dataset that most of the time social media users tend to use different Roman graphemes to map the same native Assamese grapheme.

4.6 Back Transliteration Performance Evaluation

From previous discussions, it is evident that the text of transliterated social media is noisy, and at the same time they also exhibit many different transliteration variations at both the graphemic (see sub-section 4.4.1) and phonemic (see sub-section 4.4.2) levels. To develop effective automatic transliteration models, it is important to verify the performance of state-of-the-art transliteration methods and identify the challenging issues inherent in

TH-3777_186155001

the problem. In this section, we have investigated three of the state-of-the-art machine transliteration models, namely (1). Phrase-Based Statistical Machine Transliteration Model (PBSMT) using the Moses[15] Toolkit[103] and two state-of-the-art neural machine transliteration models. (2). A character-based sequence-to-sequence BiLSTM model with attention[98] and (3). A character-based neural transformer model[92]. As given in Table 4.1, there are 1,64,713 romanized Assamese words, of which 27,765 are unique. We have investigated the performance of the models on the unique set and on the complete set. For all the experiments, we follow five fold cross validation with 80-10-10 splits for training, validation, and test set. The main reasons for building these two separate setups are as follows:

- Unique Assamese word pairs across the three different datasets are very limited compared to the total dataset that combines the three sources. Thus, the transliteration models will not see many examples of different types while training with only the unique pairs.
- Transliteration models, specifically neural network-based models, generally require a large dataset to train their parameters. When increasing the size of the dataset, we also want to see whether increasing the size of the dataset also increases the model performance or not.
- The idea is to give enough examples to the models while training and also to utilize the full capability of the neural models, which actually need a lot of data to be trained properly.

4.6.1 Setup for SMT Model

For building the SMT model, we use an open source SMT framework called Moses decoder[103] along with GIZA++[113] for character alignment. To build the language model on the target side (that is, Assamese here) of a word-aligned parallel corpus, we used a Kenser-Ney trigram language model, KenLM[114]. For training the Phrase-based SMT system, we used the romanized Assamese word on the source side and its equivalent native Assamese word on the target side. The system was tuned with Minimum Error Rate Training (MERT)[164] on a dataset different from that of the train set and then tested with a different test set.

[15] https://github.com/moses-smt/mosesdecoder

TH-3777_186155001

4.6.2 Setup for NMT Model

We implemented both baseline neural models with the help of the popular open-source OpenNMT toolkit[16][115]. In the following, we discuss the experimental setup for both models.

BiLSTM sequence-to-sequence Model:

For the BiLSTM (Bidirectional Long Short-Term Memory) implementation of the neural sequence-to-sequence encoder-decoder model with attention[98], the encoder is made up of two BiLSTM layers, each with 512 hidden units. The whole network processes words (split into characters) in batches of 128 embeddings, 64 training batches, and 32 validation batches, with an initial learning rate of 1, decaying by 0.5 over 10,000 steps within a total of 200,000. To counter overfitting, BiLSTM dropout and attention dropout are applied with values set at 0.3, 0.1, and 10 respectively, employing an early stopping criterion. In the OpenNMT library, we try to keep the default parameters as high as possible. Similarly, the decoder network includes 512 RNN[87] hidden units with 2 stacked on the decoder side. We use the Adagrad Optimizer[118] to train the sequence-to-sequence system and normalize the gradient to prevent memory growth during training. At every 10,000 steps, we set a checkpoint. In the BiLSTM encoder, we feed the Roman transliterated Assamese words split into characters, while the LSTM decoder will learn from the corresponding native Assamese words and predict the native Assamese words character by character.

Neural Transformer Model:

The Transformer is a neural encoder-decoder model featuring multiple heads that leverage self-attention. It is constructed by stacking identical layers (N = 6) for both the encoder and decoder. Operating at the character level, the input characters are initially encoded into 512 dimensional embedding vectors, with an initial hidden Transformer feedforward size of 2,048. The relative or absolute position of characters in the input text sequence is then encoded by adding positional encoding to the embedding input. A residual block[165] encompasses each of the two sublayers, followed by an addition and normalization layer on both the encoder and the decoder. Prior to being added to the input of the sublayer and normalized, the output of each sublayer undergoes dropout[117] with a value of 0.1. The Transformer utilizes 4 accumulator counts, undergoes 200,000 training steps, has a maximum of 2 generator batches, a batch size of 512, a token-based batch type, and a learning rate set at 2. All hyperparameter values are detailed here. The Transformer's self-attention employs 8 heads, utilizing scaled-dot attention[154] in the decoder layer. Between the two aforementioned sub-layers, the decoder introduces a third layer that

[16] https://opennmt.net/OpenNMT-py/

TH-3777_186155001

Table 4.7: Evaluation result of three baseline transliteration models on unique and whole transliteration pairs

Language pairs	PBSMT using Moses			BiLSTM with attention Model			Neural Transformer Model		
	CER	WER	BLEU score	CER	WER	BLEU score	CER	WER	BLEU score
Assamese(roman) to Assamese(native) Unique pairs	18.96%	56.24%	69.67%	19.76%	58.90%	68.60%	**16.40%**	**50.00%**	**74.34%**
Assamese(roman) to Assamese(native) Whole dataset	8.57%	21.41%	83.9%	8.06%	21.30%	86.5%	**6.77%**	**17.03%**	**88.7%**

performs multi-headed attention over the encoder stack's output, utilizing the Adam optimizer[120] with an initial decay rate of 0.998.

4.6.3 Error Analysis from Transliteration Output

For the evaluation of the transliteration models, we have used three performance matrices. (1). Word Error Rate (WER), (2). Character Error Rate (CER) and (3). BLEU[111] accuracy score as described in section 3.4. Table 4.7 describes the result. In the first experimental setup, where we trained each of the three transliteration models with only unique transliteration pairs, the neural transformer model outperformed the other two transliteration models in terms of both the word error rate (WER) and the character error rate (CER). We also saw that the neural transformer model outperformed the other two models by a large margin in terms of the BLEU score. We can verify this from Table 4.7.

In the second experimental setup, while combining the three datasets together and running the same experiments again on the entire dataset, the neural transformer model once again outperformed the other two transliteration models in terms of both the word error rate (WER) and the character error rate (CER) which can also be seen in Table 4.7. Again, we have found in Table 4.7 that the neural transformer model outperformed the other two transliteration models by a large margin in terms of the BLEU score. Thus, the neural transformer model has outperformed the other two transliteration models in both experimental setups. Again, the performance of the second experimental setup, which was carried out on the entire dataset, was superior to that of the first experimental setup, which made use of only the distinct transliteration pairs in each of the three transliteration models.

The word error rate (WER) and the character error rate (CER) are basically a combination of three different error types, namely substitution error, insertion error, and deletion

TH-3777_186155001

error. Each of the three different error types (i.e., substitution, insertion, and deletion errors) has its own significance. Substitution errors are those errors where, instead of predicting an Assamese grapheme or diacritic for a Roman grapheme, the transliteration model predicted/substituted by a different Assamese grapheme/diacritic in the output. Again deletion errors are those errors where some of the Assamese graphemes/diacritics are missing in the final predicted output by the transliteration models. However, some extra Assamese graphemes/diacritics are being added in the final transliteration output by the transliteration model in case of insertion errors. Most of the errors present in our dataset are substitution errors followed by insertion and deletion errors in all three transliteration models in both the experimental setups.

4.6.4 Analysis between the outputs of three different transliteration models with the same Roman input

In this section, we compare the transliteration output from our three models using the same Roman input sample. We have categorized the outputs of the models into eight different cases and presented them in Table 4.8 along with some sample examples. Any output from the three models that do not match the ground truth is highlighted in red. Out of a total of 27,765 instances, we observed that all three models agreed with the ground truth 7,446 times (CASE 1), whereas all three models disagreed with the ground truth 10,314 times (CASE 2). Furthermore, we found that the BiLSTM and Transformer models agreed with the ground truth in 2,212 instances, but the SMT output did not agree with the ground truth (CASE 3). Similarly, the SMT and Transformer models agreed with the ground truth in 2,111 instances, but the BiLSTM output did not agree with the ground truth (CASE 4). Moreover, the SMT and BiLSTM models agreed with the ground truth in 777 instances, but the Transformer output did not agree with the ground truth (CASE 5). In addition, only the SMT output matched the ground truth in 1,816 instances (CASE 6), only the BiLSTM output matched the ground truth in 976 instances (CASE 7), and only the Transformer output matched the ground truth in 2,113 instances (CASE 8). Finally, we have presented a Venn diagram plot of the three different transliteration model outputs for the same Roman inputs in Figure 4.6.

4.6.5 In-depth error analysis from transliteration output

From the experimental results on the transliteration outputs of each of the three different transliteration models, we can draw a few conclusions. Since all three current

TH-3777_186155001

Table 4.8: Comparison between the three different transliteration model outputs with the same Roman input sample

Observations	Roman input	Assamese Ground Truth	SMT Output	BiLSTM Output	Transformer Output
CASE 1:	lgt	লগত	লগত	লগত	লগত
	mil	মিল	মিল	মিল	মিল
CASE 2:	kiso	কিছু	কিছো	কিছ'	কৈছো
	nabalak	নাবালক	নাবলাক	নালালে	নৱলাক
CASE 3:	aru	আৰু	অৰু	আৰু	আৰু
	saru	সৰু	চাৰো	সৰু	সৰু
CASE 4:	bangladeshi	বাংলাদেশী	বাংলাদেশী	বাংলদেশী	বাংলাদেশী
	hoise	হৈছে	হৈছে	থৈছে	হৈছে
CASE 5:	gyan	জ্ঞান	জ্ঞান	জ্ঞান	জ্ঞানী
	joma	জমা	জমা	জমা	যোমা
CASE 6:	thai	ঠাই	ঠাই	থৈ	থৈ
	jihokole	যিসকলে	যিসকলে	যিহকলে	জিসকলে
CASE 7:	ghuri	ঘূৰি	ঘূৰি	ঘূৰি	ঘূৰাই
	bakhr	বাখৰ	বেশৰ	বাখৰ	ভাষাৰ
CASE 8:	borai	বঢ়াই	বৰাই	বৰাই	বঢ়াই
	moyo	মইও	ময়	ময়ো	মইও

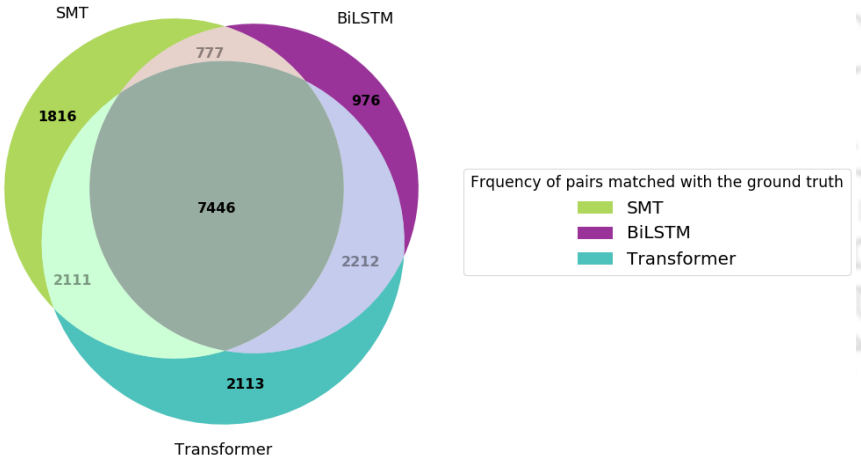

Figure 4.6: Venn diagram of three different transliteration model outputs for the same Roman inputs

transliteration models are executed directly on the noisy social media data themselves, the characteristics of these noisy social media data are also reflected in their transliteration outputs and also in the transliteration errors. We have done a thorough investigation on transliteration output at the word level and realized that each of the three different error types, i.e. insertion, deletion, and substitution errors, can be further subdivided into five different sub-error types. Figure 4.7 shows each of these three types of error along with the sub-errors. In the following, we will briefly discuss each of these five different sub-error types.

TH-3777_186155001

Figure 4.7: Sunburst View of different types of errors observed from the experimental results.

1. **Transliteration model error:** In this type of error, both the actual ground truth and the human annotation are correct, but the transliteration model still gives the wrong result. Most of the time, either one extra character is being added in the middle/last position, or one character is deleted at the middle/last position of the word. For example, the Assamese word **"ফুর্তিত"** is wrongly predicted for the Roman ground truth word "**furtite**". However, it should be predicted as **"ফুর্তিতে"**. The other two types of model error are due to the presence of consonant conjuncts and multiple mappings. Both types of errors are briefly discussed below:

   (a) **Errors due to consonant conjuncts:** Consonant conjuncts are the combination of two or more than two consonants that join together with the help of a special character halanta "্" in between. For example, the Assamese word **"কৃতজ্ঞতা"** is written as "**kitoygyota**" in the ground truth, but the transliteration model incorrectly predicted it as **"কেইটোগ্যতা"**. Here, **"জ্ঞ"** is a consonant conjunct of consonants: **"জ"**, special character "্" and consonant **"ঞ"**. But the model predicted it as **"গ্য"** which is a combination of consonant **"গ"**, special character "্" and consonant **"য"**. Since on the Roman side (ground truth), "**gyo**" is present, which is actually valid for both **"জ্ঞ"** and **"গ্য"**. But in actual Roman transliteration, **"জ্ঞ"** should be transliterated as "**jnya**" as in the person name, e.g. **"জ্ঞান ৰঞ্জন"** ("**Jnyana Ranjan**").

   (b) **Errors due to multiple mapping:** In the previous sections, We have already seen that one Roman grapheme can be used to represent many different Assamese graphemes (one-to-many mapping); similarly, one Assamese grapheme

TH-3777_186155001

can have multiple different Roman equivalents (many-to-one mapping). For example, Assamese word **"সঁচাই"** is predicted as **"সঁছাই"** for the Roman ground truth "sosai". Similarly, another Assamese word **"জনজাতীয়"** is predicted as **"জনযাতিয়"** for the Roman ground truth word "jonozatiyo". In the above two examples, two types of character mapping are observed:

i. **One-to-many mapping:** In the first example, the Roman grapheme "**s**" is used to transliterate the Assamese graphemes **"স"** and **"চ"** in the same word itself. Although the Assamese grapheme **"ছ"** can also be transliterated with the help of the Roman grapheme "**s**".

ii. **Many-to-one mapping:** In the second example, for the same Assamese grapheme **"জ"**, both Roman graphemes "**j**" and "**z**" are used in the same word. Although the Assamese grapheme **"য"** can also be transliterated with the help of the Roman grapheme "**z**".

2. **Errors due to short form in the ground truth:** Typically in social media, people use a short form omitting the vowels present between the consonants in a Roman transliterated space. While writing Assamese using Roman characters on social media, people also intentionally delete vowels present between consonants to make it short. Thus, the Roman equivalent mappings for Assamese vowel diacritics are not present in the Roman ground truth. e.g: Assamese word **"বহুত"** is transliterated as "**bht**" in the ground truth, and the transliteration model incorrectly predicted it as "ভ". However, the actual Roman transliteration of the Assamese word **"বহুত"** should be either "**bohut**" or "**bahut**".

3. **Errors due to long form in the ground truth:** Sometimes the same character is repeated multiple times to give emphasis to a word. For example, the Assamese word **"অতি"** is transliterated as "**ooottii**" instead of "**oti**" in the Roman ground truth. But the transliteration model incorrectly predicted it as **"অত্তি"**. However the word "অত্তি" is not present in the Assamese vocabulary. Here, the Roman character "**o**" is repeated **3** times, followed by "**i**" and "**t**" is repeated **2** times each.

4. **Errors in the ground truth:** Sometimes there are errors in the Roman ground truth itself. However, by going through the context of the sentence, the annotator correctly annotates the word in Assamese. For example, the Assamese word

TH-3777_186155001

"অসুবিধা" is wrongly transliterated as "**okobidha**" in the Roman ground truth. Thus, the transliteration model also predicted it incorrectly as **"অকবিধ"**. Here, the Roman character "**k**" is used to transliterate the Assamese character **"স"**. This is very rare, or rather accidental, mapping.

5. **Errors present in the human annotation:** Many times human annotators also make mistakes while annotating. Most of the time, the user forgets to put the last character. In that case, our transliteration model sometimes predicts it correctly and sometimes makes error. For example, annotators annotate the Assamese word **"তেওঁলোকক"** in place of **"তেওঁলোককো"** for the Roman word "**teolokoko**" present in the ground truth. However, in this example, our transliteration model correctly predicts it as **"তেওঁলোককো"**.

We have observed that among the five different types of sub-errors, the transliteration error due to multiple mappings is relatively high. This can also be seen in Figure 4.7. Errors due to multiple mapping occur due to the presence of multiple Roman transliteration equivalent to represent an Assamese grapheme and vice versa. To normalize the effect of these error types, we have mapped each of the phonetically equivalent Assamese graphemes to a single Assamese grapheme with the help of the phonetic mapping table present in Table 4.4 and Table 4.5 and again evaluated the performance. We have noticed that after this normalization step, the word error rate (WER) is reduced from 50.0% to 41.1% of the state-of-the-art transformer model. Furthermore, with the help of a very good language model trained in Assamese, these phonetically similar-sounding normalized words could be converted back to their original canonical form in the post-processing steps. For a more comprehensive discussion on how access to sentential context can resolve additional transliteration errors, we refer the reader to Chapter 6

4.6.6 Summary of the three transliteration model outputs that correspond to the transliteration variations

In the previous sub-sections, we evaluated the performance of three different transliteration models and conducted a detailed error analysis. In this section, we present a concise overview of the various transliteration variations observed in our dataset. To accomplish this, we leverage the outputs of the three distinct transliteration models used in our evaluation. We classify these variations into seven distinct categories, each illustrated with sample examples in Table 4.9. Among these seven categories, three have sub-categories.

TH-3777_186155001

The model outputs associated with these variations, which lead to transliteration errors are highlighted in red, while the actual native ground truths and Roman inputs are highlighted in blue. We will briefly discuss all of these variations.

1. **Variations due to word elongation :** From Table 4.9, we have observed that the Assamese word **"মজা"** is transliterated into different Roman forms, such as "**moja**", "**mojaaaa**", and "**mojah**". However, both the SMT and BiLSTM models incorrectly predicted "**mojah**" as **"মজাহ"** instead of **"মজা"**. Similarly, for the long form representation of the Roman input "**juiiiiiiiiiiiiiiii**", which corresponds to the Assamese word **"জুই"**, the SMT model predicted it as **"জুইইই"** and the BiLSTM model predicted it as **"উউই"**, instead of the correct form.

2. **Variations due to word shortening :** The Roman transliterated form "**kr**" is used as a short form representation for both the Assamese words **"কাৰণ"** and **"কৰক"**. However, the consonant conjunct **"ক্ৰ"** in **"পর্যায়ক্রমে"** is also represented by "**kr**". As a result of this variation in short-form usage, our SMT model predicted it incorrectly as **"কৰি"**, while the BiLSTM model predicted it as **"কেৰে"**, instead of the correct form as shown in Table 4.9.

3. **Variations due to writer biases :** Users often make mistakes while writing Assamese in Roman form, but human annotators correct these mistakes by considering the context, which can result in variations. As shown in Table 4.9, we have observed that the Assamese word **"সংস্কৃতি"** should be transliterated as "**khongskriti**" instead of "**khongkriti**" as there is no word like **"সংকৃতি"** in the Assamese vocabulary predicted by all three models. Additionally, "**bo**" is used to represent **"ভ"** in **"ভবা"**, but both the BiLSTM and Transformer models incorrectly predicted it as **"ববা"** and **"বোঁবা"** respectively.

4. **Variations due to character/numeric substitution:** Both "**a**" and "**ya**" are often used interchangeably to represent **"য়া"** as seen in **"দিয়া"**. However, both "**a**" and "**ya**" are also used for **"য়ে"** which creates variations that are also observed in the output of our BiLSTM model. Similarly, the number "**2**" is often used as a substitute for **"তু"**/**"তো"**/**"টু"**/**"টো"**. As shown in Table 4.9, "**n2**" is used to represent **"ন্তু"** in **"কিন্তু"**. However, both SMT and BiLSTM output use "**n2**" to represent **"নটো"** and **"ন্তো"** which creates errors due to variations.

5. **Variations due to one-to-many mapping:** It has been observed that some Roman characters/digraphs can be mapped to represent many different native Assamese characters, resulting in one-to-many mapping and creating numerous

TH-3777_186155001

transliteration variations. Below, we have categorized all of the variations observed in our dataset.

(a) **Multigrapheme mapping:** Table 4.9 shows that a single Roman character or digraph can represent multiple Assamese characters. These variations are commonly observed when representing an Assamese phoneme sound that is not present in English. For instance, the digraph "**sh**" is used to represent "স" in "সমালোচনা", "ক্ষ" in "কলাক্ষেত্র", "শ" in "নিশিকাই" and "চ" in "চৰকাৰ". However, since the digraph "**sh**" can also represent other Assamese characters, this one-to-many character mapping creates many transliteration errors due to the variations observed in all three model outputs.

(b) **Vowel phoneme triggered:** The vowel phoneme /i/, which corresponds to Assamese vowel graphemes (ি, ী, ই, ঈ), is generally represented by the Roman grapheme "**i**". Table 4.9 illustrates that in the word "ঈদ", "**i**" is used to represent the vowel grapheme "ঈ". However, due to variations in the vowel phoneme, the SMT output predicts "ী" and both the BiLSTM and Transformer outputs predict "ই". Similarly, all three model outputs predict "ৰী" for "**ri**" instead of "ৰি" in "নাগৰিক".

(c) **Consonant grapheme triggered:** The Roman grapheme "**s**" is used to represent both the consonant graphemes "চ" and "ছ" in the representation of "চাব", "চাগে" and "বিছাৰিবলে" respectively, which creates variations in their representations. As a result, sometimes "চ" is predicted as "ছ" and vice versa, as shown in Table 4.9.

(d) **Consonant phoneme triggered:** Roman graphemes "**n**" and "**t**" are used to represent the two consonant phonemes /n/(ন, ণ, ঞ) and /t/(ত, ট, ৎ), respectively. This results in variations, as demonstrated in Table 4.9, where the predicted output is "ন" instead of "ণ", and "ত" instead of "ট" and vice versa both for the representation of "কাৰণটো" and "চামতা".

(e) **Intra-word variation:** Sometimes, it is observed that a single Assamese character can be represented with different Roman characters within a single word. For example, the Roman character "**t**" is used to represent both the Assamese characters "ত" and "ট" inside the word "এটাতো" which leads to confusion and variations in the output of both SMT and Transformer models. They predict "এটাটো" instead of "এটাতো". Similarly, a similar observation is also noticed in the representation of "নাৰায়ণ" where the Roman character "**n**" is used to represent both the Assamese characters "ন" and "ণ" but the BiLSTM model predicts it as "নাৰায়ন" instead of "নাৰায়ণ" as shown in Table 4.9.

TH-3777_186155001

6. **Variations due to many-to-one mapping:** Below, we have categorized all the observed variations in our dataset where a single native Assamese character can have multiple alternative Roman representations(many-to-one mapping), either as a single Roman character or as digraphs.

   (a) **Unfamiliar phoneme sound:** The Assamese language contains several phoneme sounds that are not present in English. e.g: /x/(স, শ, ষ), /k^h/(খ, ক্ষ) etc. Consequently, multiple Roman alternatives are used to represent these unfamiliar sounds using English orthography. Table 4.9 demonstrates that the Assamese character "স" is represented as "**x**" in **"সন্মান"**, "**h**" in **"সেইটো"**, "**s**" in **"সূত্রধাৰ"**, and "**kh**" in **"সেই"**. This many-to-one character mapping leads to various output variations in all three models since the same Roman characters can also represent other native Assamese characters, as shown in the table.

   (b) **Vowel grapheme triggered:** In order to represent the vowel grapheme **"আ"** in Roman form, both "**aa**" and "**a**" are used, as seen in the representation of **"আৰু"** in Table 4.9. However, sometimes "**a**" is also used to represent the vowel grapheme **"অ"**. This variation creates errors as reflected in the SMT output, where **"অৰু"** is predicted instead of **"আৰু"**.

   (c) **Intra-word variation:** Within a single word, sometimes the same Assamese character can be mapped to different Roman characters. As shown in Table 4.9, in the word **"অসমবাসীৰ"**, both the Roman characters "**x**" and "**kh**" are used to represent the Assamese character **"স"**. This variation can lead to confusion, as seen in the SMT model output, where **"অসমবাখিৰ"** is predicted instead of **"অসমবাসীৰ"**. Similarly, in the representation of the word **"বেচিবলৈহে"**, both "**v**" and "**b**" are used to represent **"ব"**. However, since "**v**" is usually used to represent **"ভ"**, all three models predict "**v**" as **"ভ"** instead of **"ব"**.

7. **Variations due to both one-to-many and many-to-one mapping:** Sometimes, examples of both one-to-many and many-to-one mapping can be observed within the same two occurrences. Such examples have been observed in our dataset, and they are mostly triggered by vowel diacritics. An explanation of this variation example is provided below.

   (a) **Vowel diacritic triggered:** Both "**u**" and "**oo**" are used to represent the vowel diacritic " ু" in the Assamese words **"বৰষুণ"** (**barakhun**) and **"ভুল"** (**bhool**) respectively. However, this variation creates errors in the outputs of all three models, as shown in Table 4.9, since "**u**" is also used sometimes to represent the vowel diacritic **"ো"**. Furthermore, the SMT model incorrectly

TH-3777_186155001

Table 4.9: Summary of transliteration variations from three different model outputs

OBSERVATIONS		Roman input	Assamese Ground Truth	SMT Output	BiLSTM Output	Transformer Output
Word elongation		moja	মজা	মজা	মজা	মজা
		mojaaaa	মজা	মজা	মজা	মজা
		mojah	মজা	মজাহ	মজাহ	মজা
		juiiiiiiiiiiiiiii	জুই	জুইইই	উউই	জুই
Word shortening		krn	কাৰণ	কাৰণ	কাৰণে	কাৰন
		krk	কৰক	কৰক	কৰি	কৰি
		porjaykrome	পর্যায়ক্রমে	পর্যায়কৰিমে	পর্যাৰকেৰে	পর্যাক্রমে
Writer biases		khongkriti	সংস্কৃতি	সংকৃতি	সংকৃতি	সংকৃতি
		boba	ভবা	ভৱা	ববা	বোবা
Character/numeric substitution		dia	দিয়া	দিয়া	দিয়ে	দিয়া
		diya	দিয়া	দিয়া	দিয়ে	দিয়া
		kin2	কিন্তু	কিনটো	কিন্তো	কিন্তু
One-to-many mapping	Multigrapheme mapping	shomalosana	সমালোচনা	সমালোচনা	সমালোচনা	সমালোচনা
		kolashetrot	কলাক্ষেত্ৰত	ক'লেহেত্ৰত	কলস্মাৰত	কলাশ্রত
		nishikay	নিশিকাই	নিচিকেই	নিশেই	নিশ্চয়
		sharkar	চৰকাৰ	চৰকাৰ	চৰকাৰ	শৰকাৰ
	Vowel phoneme triggered	id	ঈদ	ীদ	ইদ	ইদ্
		nagrik	নাগৰিক	নাগৰীক	নাগৰীক	নাগৰীক
	Consonant grapheme triggered	saba	চাব	সব	সবে	চবে
		sage	চাগে	ছাগে	চাগে	ছাগে
		bisaribole	বিছাৰিবলে	বিচাৰিবলে	বিচাৰিবলে	বিচাৰিবলে
	Consonant phoneme triggered	karontu	কাৰণটো	কাৰনটো	কাৰন্তো	কাৰণতো
		chamata	চামতা	সমতা	সমাটা	চামতা
	Intra-word variation	atatu	এটাতো	এটাটো	এটাতো	এটাটো
		narayan	নাৰায়ণ	নাৰয়ণ	নাৰায়ন	নাৰায়ণ
Many-to-one mapping	Unfamiliar phoneme sound	xonman	সন্মান	সন্মান	সন্মান	সন্মান
		haitu	সেইটো	সেইটো	সেইটো	সেইটো
		sutradhar	সূত্ৰধাৰ	চুত্ৰধাৰ	চুদতাৰ	সুত্রদ্ধাৰ
		khai	সেই	খাই	খায়	খায়
	Vowel grapheme triggered	aaru	আৰু	আৰু	আৰু	আৰু
		aru	আৰু	অৰু	আৰু	আৰু
	Intra-word variation	axombakhir	অসমবাসীৰ	অসমবাখিৰ	অসমবাসীৰ	অসমবাসীৰ
		vesiboloihe	বেচিবলৈহে	ভেচিবলৈহে	ভেছিবলে	ভেচিবলৈহে
Both One-to-many and Many-to-one mapping	Vowel diacritic triggered	barakhun	বৰষুণ	বাৰাখোন	বৰষুণ	বৰষুণ
		maru	মাৰো	মাৰো	মাৰো	মাৰো
		bhool	ভুল	ভোল	বহ'ল	ভোল

predicts “বাৰাখোন” instead of “বৰষুণ” for “**barakhun**”, where the graphemes “ব” and “ৰ” are followed by the diacritic “ো” in that instance. However, in the very next example, “**a**” is used as “ো” in the ground truth representation of “মাৰো” (**maru**).

4.7 Attention analysis on transliterated Assamese social media data

From the previous sections, we have observed many variations in the Roman transliteration present in the Assamese social media text. We have also seen that Assamese users

TH-3777_186155001

do not follow common standard transliteration guidelines when writing Assamese using Roman script. We have also observed the performance of transliteration using three standard transliteration methods in our dataset and also performed a thorough error analysis. From the error analysis, we have seen many different error types and found that most of the errors are due to multiple character mappings present between Assamese and Roman scripts. To understand the contextual dependency between graphemes while doing the prediction, we performed an attention analysis between the source-target language pairs. In our case, the source language input is the Assamese word in Roman transliterated form, and the target language output is the corresponding native Assamese word. The word-aligned source-target pairs are split into characters first. We have built a grapheme-based bidirectional LSTM neural sequence-to-sequence model with attention. With the help of attention weight scores of the source language characters, while generating the target language characters, we can further analyze the attention between the source-target pairs. We have used the same setup discussed in sub-section 4.6.2 for building our attention-based sequence to sequence BiLSTM model.

Attention analysis actually gives us interesting insights into the character mapping present between the Romanized Assamese source words and native Assamese target words. i.e., with the help of the attention weight scores, our attention model tells us about those characters in the source word that actually get more attention while generating the characters in the target word. In other words, while generating the output string, instead of considering the entire encoded string, the decoder will focus only on a few characters in the input string determined by the attention weight scores. This attention analysis may further help us to understand the transliteration model and also the characteristics of the transliterated texts in a better way to build a transliteration model specifically for Assamese social media data in the future. In the following sub-sections, we will discuss the attention analysis of Assamese vowels, vowel diacritics, and consonants with the help of the attention weight scores while training the BiLSTM neural sequence-to-sequence transliteration model for the collected social media data.

4.7.1 Observation on Assamese Vowels

In Assamese, we have a total of **11** vowel graphemes. All the corresponding English graphemes that receive the maximum attention weight score for the generation of each Assamese vowel grapheme given by the attention model can be found in Figure 4.8. From the attention plot of the Assamese vowels, the following conclusions can be drawn.

TH-3777_186155001

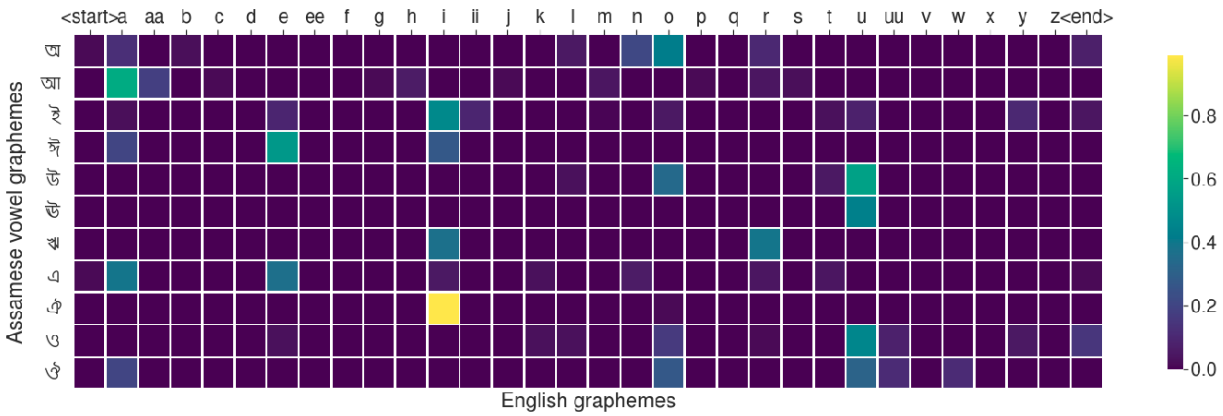

Figure 4.8: Attention weight analysis on Assamese vowels.

1. All English vowel graphemes, "**a**", "**e**", "**i**", "**o**", and "**u**", only get the highest attention weight score for the generation of all Assamese vowels.

2. English vowel alphabet "**a**" alone gets the highest attention weight score for the generation of five Assamese vowels followed by "**o**", getting the highest attention score for the generation of four vowels, and "**e**", "**i**" and "**u**" gets the maximum attention score for the generation of **3** vowels each.

3. English graphemes other than vowels that receive attention weight scores are "**n**", "**w**", "**y**", and two vowel digraphs "**aa**" and "**uu**".

4.7.2 Observation on Vowel Diacritics

In Assamese, there are a total of **10** vowel diacritics. Similar to Assamese vowels, for the generation of vowel diacritics, our attention model also gives the attention weight scores against each corresponding English grapheme. Figure 4.9 provides the attention weight analysis plot for Assamese vowel diacritics.

4.7.3 Observation on Consonants and two Special Symbols

A similar attention analysis was carried out for the generation of Assamese consonants. A total of 41 consonants are present in Assamese. Along with these consonants, two special symbols (' and ◌্) are also frequently found in Assamese. The special symbol (◌্) is actually required for the formation of consonant conjuncts in Assamese. Figure 4.10 will discuss the weight distribution of attention for the generation of all 41 Assamese consonants and two special symbols from our dataset.

TH-3777_186155001

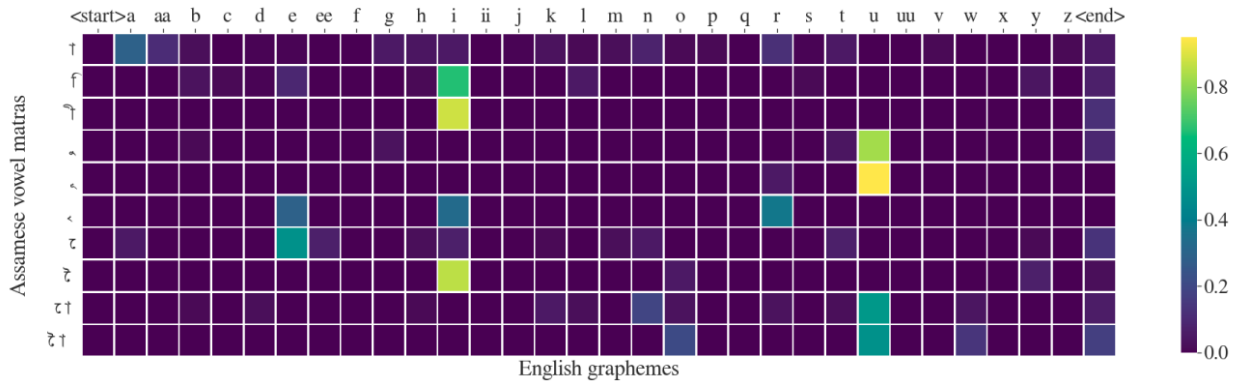

Figure 4.9: Attention weight analysis of Assamese vowel diacritics.

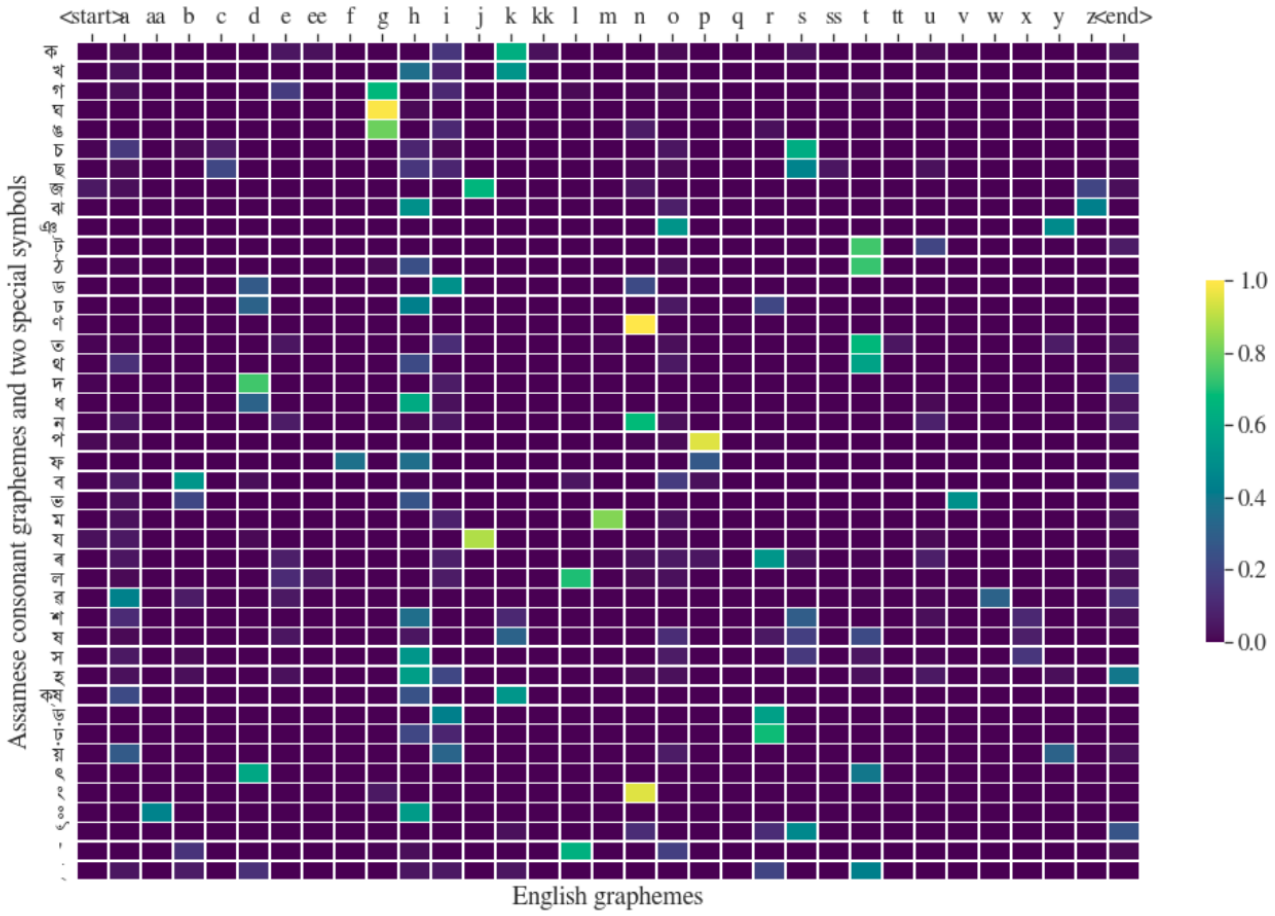

Figure 4.10: Attention weight analysis of Assamese consonants.

From the analysis of the attention on Assamese consonants given by the attention weight scores, the following critical observations can be drawn:

1. For all consonant graphemes present in the initial and final positions of a word, the attention model always gives attention to the individual Roman grapheme only.

TH-3777_186155001

2. In the case of Roman transliterated Assamese words written in their short form, we have seen that most of the time vowels are usually removed between the consonants. When we transliterate those words back to the native word, we see that the attention model pays attention to all the characters preceding and following the missing vowels.

From the attention weight analysis of English graphemes against each of the Assamese vowels, vowel matras/diacritics, and consonant graphemes, we have drawn the following key observations.

1. Roman graphemes "**a**", "**e**", "**i**", "**o**", "**u**", "**w**", "**y**" and the digraph "**aa**" gets the highest attention weight score for the representation of both vowel graphemes and diacritics. We also observed the same in the transliteration variation analysis section.

2. Interestingly, Roman graphemes "**n**", "**r**", and digraph 'uu' also receive the highest attention weight scores for the generation of Assamese vowel graphemes and diacritics. This means that there are a few Roman graphemes that are only seen for the representation of some unique Assamese vowel graphemes and diacritics.

3. Apart from the above few Roman graphemes, other Roman graphemes present in the heatmap are getting less attention weight scores. That means we need to identify the evidence of those rare/accidental transliteration variations if present.

4. Roman graphemes "**i**" and "**u**" receive the highest attention weight scores for the representation of both Assamese vowel graphemes and diacritics.

5. All Roman graphemes "**a**", "**e**", "**i**", "**o**", and "**u**" are receiving attention weight scores for the generation of more than one vowel grapheme and diacritics. This signifies the many-to-one character mapping present between Assamese vowel graphemes/-diacritics and Roman graphemes.

6. We have also observed that for the generation of almost all Assamese vowels and vowel diacritics, more than one Roman grapheme or digraph gets the attention weight scores. This clearly indicates the one-to-many character mapping present between Assamese vowel graphemes, vowel diacritics, and Roman graphemes.

7. Since both one-to-many and many-to-one character mappings are observed from the attention heatmap as well, contextual information may also need to be considered while generating rules for building any transliteration model for processing these kinds of noisy social media data in the future.

TH-3777_186155001

8. One-to-one mapping is very rare in the representation of both Assamese vowel graphemes and diacritics with Roman graphemes. However, we have observed one-to-one character mapping in the representation of a few Assamese consonant graphemes with Roman characters. This is clearly evident from the heatmap representation of attention weight scores.

9. For the generation of all Assamese consonant graphemes, which are usually represented by English digraphs in Roman transliterated space, the attention model gives attention to the second English character in the digraph irrespective of its position in the word. Most of the time, the second English character present in the digraph is found to be the Roman grapheme "**h**". For example: "খ" (**kh**), "থ" (**th**), "ঘ" (**gh**), "ধ" (**dh**), "ফ" (**ph**), "ভ" (**bh**) etc.

10. Grapheme "**h**" also receives the maximum attention weight scores for the representation of the Assamese consonant grapheme "হ". Thus, a direct one-to-one character mapping is also present between them.

4.8 Summary and Future work

In conclusion, this chapter investigated the transliteration characteristics of Romanized Assamese language text in social media. A unique dataset was curated and used for this purpose, which is suitable for building an effective machine transliteration system for the language in this domain. The analysis of graphemic and phonemic variations in transliteration provided insights into users' writing behavior while using Roman orthography. This study is the first to analyze the transliteration characteristics of Indian language social media content written in Roman orthography, with a particular focus on Assamese. The performance of three state-of-the-art machine transliteration models was evaluated, and the challenges that need to be addressed to build an effective machine transliteration system for Assamese in the social media domain were identified. The developed neural Transformer model outperformed the other two baseline models namely BiLSTM with attention model and PBSMT model and achieved **50%** word error rate(WER) and **74.34%** BLEU score accuracy. Additionally, a state-of-the-art neural network-based machine transliteration model (BiLSTM with attention) was developed to understand the attention behavior between different Assamese and English character mappings with the help of attention weight scores. This will eventually help to build a more robust machine transliteration system while handling these kinds of Romanized social media data in the future.

TH-3777_186155001

Based on the analysis of the data and experimental results, it can be inferred that users do not adhere to any established transliteration standards or rules, resulting in significant variations in both graphemic and phonemic transliterations. Moreover, users seem to prioritize the phonetic characteristics associated with Roman graphemes. The complex one-to-many and many-to-one relationships between the Assamese and Roman orthographies and other social media text-specific problems make it challenging for state-of-the-art machine transliteration techniques to produce satisfactory outcomes.

In the future, this study can be further enhanced in many ways; (i) Based on the different types of errors identified through experimental observations and the corresponding analytical observations obtained from the dataset, there is potential to expand this study and develop an effective transliteration method for Assamese language social media text. (ii) Due to the presence of certain social media-specific challenges in a social media text, it may be recommended to perform lexical normalization as a pre-processing step before applying transliteration models to Romanized Assamese social media input. (iii) As our current transliteration setup is based solely on word pairs, certain transliteration errors may arise due to the absence of contextual information. To improve the performance of the transliteration models, it may be beneficial to include sentence-level contextual information and integrate a well-trained language model in the target Assamese language.

We can further extend this study to investigate the problem of transliteration in Indian social media text across Indian languages. Given that India is a multilingual country and most of the Indian languages share similar property, this issue is considered fundamental to practically all Indian social media contexts, making it a crucial area of research.

TH-3777_186155001

TH-3777_186155001

Chapter 5

Study on the Adaptability of Transliteration Models to a Canonical Dataset vs. a Noisy Social Media Dataset

Following the creation of our novel back transliteration dataset and its in-depth transliteration characteristic and variation analysis, we now turn our focus to evaluating how well existing transliteration models handle real-world noisy transliterated text. The increasing prevalence of Romanized Assamese on social media introduces significant challenges for NLP applications, as user-generated transliterations are highly inconsistent and lack standardized conventions. Unlike manually curated canonical datasets, these noisy texts contain spelling variations, phonetic inconsistencies, and non-standard transliteration patterns, making it difficult for current models to process them effectively.

The existing state-of-the-art transformer-based multilingual transliteration models present for 22 Indic languages are trained on canonical (clean, manually transliterated) datasets and exhibit strong performance on structured data. However, it remains unclear how these models perform when applied to noisy, back-transliterated social media text—where variations are uncontrolled and unpredictable. Conversely, models trained on non-canonical data might struggle with structured transliteration tasks. Without evaluating these aspects, one might assume that machine transliteration is a fully solved problem.

TH-3777_186155001

This chapter systematically investigates whether these models can effectively handle noisy back-transliterated social media text. We conduct fifteen different experimental setups using both canonical (manually forward transliterated) and non-canonical (manually back-transliterated) datasets to identify optimal approaches for handling both structured and noisy data. Our findings reveal significant performance gaps, underscoring the need for robust transliteration models that can generalize across different transliteration scenarios.

5.1 Introduction

The increasing prevalence of phonetically transliterated user-generated content on digital platforms, particularly social media, has made machine transliteration—the process of converting text from one orthography to another—an essential task in natural language processing (NLP). In a multilingual society like India, where many languages coexist, transliteration plays a crucial role in facilitating communication and enabling accessibility. A significant portion of Assamese social media content is written in the Roman script, leading to challenges in automatic text processing. Early research in machine transliteration primarily focused on handling named entities and technical terms as part of machine translation tasks [166]. However, with the widespread adoption of informal transliteration in social media, robust transliteration models are now essential for real-time information processing.

Depending on the source and target orthographies, transliteration can be categorized into forward transliteration (converting native script text into another script while preserving phonetic integrity) and back-transliteration (converting transliterated text in a non-native script back to its original script) [2]. Additionally, transliteration datasets can be classified into canonical parallel datasets and non-canonical parallel datasets. Canonical datasets contain structured and rule-based transliterations, often manually created by language experts. In contrast, non-canonical datasets consist of noisy user-generated transliterations, which vary widely in spelling, phonetic mapping, and stylistic conventions. Due to the lack of standardization in social media transliteration, back-transliteration of such text is a significantly more challenging task.

This chapter explores various experimental setups to determine the most effective transliteration models for handling both clean and noisy transliterated data. We evaluate fifteen different configurations, leveraging both canonical (manually forward transliterated) and non-canonical (manually back-transliterated) datasets. Our investigation reveals that transliteration models trained solely on manually created canonical datasets fail to generalize to noisy social media text, while models trained exclusively on noisy transliterations

TH-3777_186155001

struggle to handle structured canonical data. To address these limitations, we examine different training strategies, including dataset merging and transfer learning. While the merged model, trained on both clean and noisy data, shows improved robustness, it remains suboptimal. Additionally, the transfer learning approach, designed to leverage knowledge from canonical transliterations for noisy text, does not significantly enhance performance. These findings underscore the necessity of developing transliteration models specifically optimized for handling the complexities of real-world, user-generated transliterations.

Given the growing importance of processing multilingual social media data, this study is particularly relevant for languages like Assamese, which has 15.3 million native speakers [3] and serves as the official language of Assam. The insights gained from this research contribute to broader transliteration challenges across Indic languages and highlight the need for more adaptable and resilient transliteration models.

In the following sections, we detail our dataset construction, experimental setups, model evaluations, and error analysis. Throughout the chapter, we refer to models trained on non-canonical datasets as "noisy models", those trained on canonical datasets as "clean models", and those trained on a combination of both as "merge models". Our primary goal is to identify effective strategies for building robust transliteration models that can seamlessly process diverse transliterated text in Assamese.

Contribution:

The major contributions of this chapter are:

1. Comprehensive evaluation of transformer-based monolingual and multilingual transliteration models on both canonical (clean) and non-canonical (noisy) transliterated Assamese social media text.
2. Exploration of dataset combinations and transfer learning approaches for enhancing transliteration performance across different types of data.
3. Identification of key challenges and limitations in existing transliteration models when applied to noisy, user-generated Assamese social media text.

5.2 Related Studies

The transliteration of informal text, particularly informal Romanized Indian language text commonly found in social media, has become a prominent research topic, especially

TH-3777_186155001

in the context of shared tasks organized by the Forum for Information Retrieval (FIRE). Notably, the FIRE 2013 and FIRE 2014 shared tasks used song lyrics data written in Roman script for evaluation and analysis. Participants were challenged to develop independent transliteration models for specific subtasks using provided fixed datasets([108], [67], [66], [104], [167], [168], [135]).

In recent years, significant progress has been made in the transliteration domain. For instance, for processing canonical Indic language transliteration pairs, [6] released the Dakshina dataset, comprising 12 South Asian languages (mainly Indian languages) written in Latin script, and provided baseline results for transliteration and language modeling tasks. [105] evaluated neural machine transliteration for English and 10 Indian languages, focusing on multilingual transliteration to leverage orthographic similarity between Indian languages. Following this work, [8] introduced the Aksharantar dataset, a large transliteration dataset for 21 Indian languages, and achieved state-of-the-art results using a single transformer-based multilingual transliteration model, IndicXlit. However, it is worth noting that none of the recent work has specifically addressed the transliteration problem for Romanized social media datasets.

5.3 Dataset

To obtain the required non-canonical Romanized Assamese dataset, we again used our in-house "**AssameseBackTranslit**" dataset for non-canonical Romanized Assamese, gathered from three major social media platforms like Twitter (currently X), Facebook, and YouTube as discussed in sub-section 3.3. To obtain the canonical Assamese dataset, we made use of the publicly available Aksharantar [8] dataset. This dataset comprises Romanized Assamese to native Assamese transliteration pairs, which were generated using a combination of strategies. These strategies encompassed extracting transliteration pairs from parallel corpora, mining from monolingual corpora, utilizing existing transliterated data sources, and carefully constructing a transliteration corpus through manual forward transliteration from native Assamese words to their corresponding Romanized forms. In total, the Aksharantar [8] dataset provided us with 1,87,924 Assamese transliteration pairs for analysis and evaluation. We performed further data cleaning and observed some errors that needed attention. Specifically, we identified a total of 8905 instances where the Bengali letter "র" was mistakenly used instead of the correct Assamese letter "ৰ". Conversely, the Assamese letter "ৰ" appeared correctly 1,33,367 times in the entire corpus. Unlike some other Indian languages, the diacritic **Nukta** (" ়") is not treated as a separate character in Assamese, resulting in individual characters like "য়", "ড়", and

TH-3777_186155001

Table 5.1: Training, Validation and Testing dataset statistics (All transliteration pairs are word splitted into characters)

	DATASET SIZE			
	TRAINING	**VALIDATION**	**TESTING**	**TOTAL**
Noisy setup	45,934	6,560	13,120	65,614
Clean setup	1,31,546	18,792	37,586	1,87,924
MergeUniqueTrans	1,74,696	19,411	Noisy and Clean test	2,44,813
MergeDuplicateTrans	4,89,266	54,363	Noisy and Clean test	5,94,335
IndicXlitEval	Multilingual IndicXlit Model		Noisy and Clean test	-

"ঢ়". These characters are not realized as combinations, such as য + ় = য়, ড + ় = ড় or ঢ + ় = ঢ়. We corrected all such instances to their proper forms as "য়", "ড়", and "ঢ়" respectively.

5.4 Experimental Setup

For conducting our experiments, we employed a transformer-based encoder-decoder architecture [92] to create a character-level monolingual transliteration model. This model operates in a one-to-one setting, taking a Romanized character sequence as input and generating the corresponding character sequence in the native Assamese script as output. To implement the monolingual transliteration task, we utilized the Fairseq implementation [116] of the transformer architecture. The model architecture included 6 encoder and 6 decoder layers, with layernorm applied within each transformer layer to normalize the outputs of the multi-head self-attention and feed-forward sub-layers. The GELU activation function was used, and the number of attention heads was set to 4 for both the encoder and decoder self-attention layers. We utilized a batch size of 1024 and set the dimensions for the encoder and decoder embeddings to 256 and the dimensions for the encoder and decoder feed-forward networks (FFN) to 1024. Dropout was applied with a rate of 0.5 and attention dropout at 0.1. We used label-smoothed cross-entropy with a smoothing factor 0.1 as the training criterion. The Adam optimizer was employed with betas (0.9, 0.98) and a learning rate of 0.001. The learning-rate scheduler followed the inverse square-root policy with a warm-up initialization learning rate of 0, meaning that the learning rate begins at zero and increases linearly during the first 4000 updates until

TH-3777_186155001

Table 5.2: Transliteration result in terms of Accuracy@1, Accuracy@4, Character Accuracy, Mean F-score, MRR and BLEU4 score for all the setups (Experimental setups with the highest and lowest accuracies(top-1) are highlighted in red and blue, respectively, while the accuracy of the best-performing setup is highlighted in **bold**)

			Acc@1	Acc@4	Char Accuracy	Mean F-score	MRR	BLEU4 Score
Setup with Noisy Test Set	Noisy Model (noisy training set unique)	*setup 1*	41.57	69.79	79.85	87.57	53.38	69.15
	Clean Model (clean training set unique)	*setup 2*	28.21	42.76	73.76	83.12	34.05	56.64
	Merge Model (with unique pairs)	*setup 3*	**41.88**	69.20	81.71	87.74	53.29	69.97
	Merge Model (with duplicate pairs)	*setup 4*	46.45	74.08	82.65	88.74	57.94	73.78
	IndicXlit Model (multilingual model)	*setup 5*	26.87	42.46	71.57	82.18	33.19	54.03
	Noisy Model (noisy training set with duplicates)	*setup 6*	46.22	73.71	82.66	88.61	57.69	73.88
Setup with Clean Test Set	Clean Model (clean training set)	*setup 7*	77.78	92.99	96.59	97.36	84.58	91.94
	Noisy Model (noisy training set)	*setup 8*	34.71	56.36	85.71	89.26	43.74	65.29
	Merge Model (with unique pairs)	*setup 9*	**75.04**	91.91	95.82	96.77	82.56	90.85
	Merge Model (with duplicate pairs)	*setup 10*	67.47	86.17	93.69	95.33	75.59	87.22
	IndicXlit Model (multilingual model)	*setup 11*	76.85	92.09	96.38	97.27	83.68	90.93
Finetunning Setup	Fine-tune Clean Model (on noisy dataset)	*setup 12*	36.63	62.11	77.43	85.66	47.24	64.40
	Fine-tune IndicXlit Model (on noisy dataset)	*setup 13*	38.05	62.42	77.26	86.19	48.16	66.48
	Fine-tune Noisy Model (on clean dataset)	*setup 14*	77.07	92.87	96.46	97.25	84.13	91.56
	Evaluate *setup 14* (on noisy testset)	*setup 15*	28.16	43.29	74.54	83.6	34.21	56.88

it reaches the target value of 0.001, after which it decays proportionally to the inverse square root of the update step(or warm-up steps). This strategy facilitates smoother convergence and prevents unstable gradients at the beginning of training. The model was trained for a maximum of 100 epochs for each setup.

Our experiments were conducted across three main setups: (1) models evaluated on the noisy test set, (2) models evaluated on the clean test set, and (3) fine-tuning experiments exploring cross-domain adaptation between the two test sets. Within the first two setups, we conducted six and five experiments respectively, while the fine-tuning setup involved four experiments. Thus, we conducted a total of fifteen distinct experimental setups. We employed a 70-10-20 split to ensure comprehensive evaluation for training, validation, and testing purposes. Among these fifteen setups, three specifically incorporated the state-of-the-art multilingual transliteration model, IndicXlit [8], for comparative analysis. Detailed dataset statistics for all fifteen setups can be found in Table 5.1. In the following sections, we will elaborate on our experimental setups.

1. **Noisy Setup:** In this setup, we specifically focused on the non-canonical transliteration pairs available in our dataset for training and validating our model. Subsequently, the trained model was employed to evaluate the performance on both the

TH-3777_186155001

noisy and clean test sets. For this setup, the input and output vocabulary sizes are 44 and 76 characters, respectively.

2. **Clean Setup:** Similarly, in this setup, we utilized the canonical transliteration pairs from Aksharantar dataset for training and validating the model. Subsequently, we evaluated the model's performance on both noisy and clean test sets using the same trained model. The input and output vocabulary sizes for this setup are 28 and 68 characters, respectively.

3. **Merge Transliteration Model with Unique Pairs (MergeUniqueTrans):** In this setup, we combined all the unique transliteration pairs from both the noisy and clean datasets for training and validation. Following training, the same trained model can be employed to assess performance using both the noisy and clean test sets.

4. **Merge Transliteration Model with Duplicate Pairs (MergeDuplicateTrans):** Similarly, in this setup, we combined all available datasets, including noisy pairs from the Noisy model setup and clean pairs from the Clean model setup, while also including duplicate transliteration pairs this time. We aim to prioritize frequently occurring transliteration pairs, thus normalizing the dataset and assigning weights to high-frequency pairs. Subsequently, the trained model was used to assess the performance on both clean and noisy test sets.

5. **IndicXlit Model (IndicXlitEval):** In this setup, we evaluated the performance of the state-of-the-art multilingual IndicXlit model using the test sets from both the noisy and clean models. The input and output vocabulary sizes for this multilingual setup are 28 and 780 characters, respectively.

6. **Finetune (noisy model):** In this setup, the model initially trained on the noisy dataset was further fine-tuned using the clean dataset from the clean model. This setup enables the model to adapt from informal and inconsistent transliteration patterns to more standardized ones, improving robustness across domains. The fine-tuning process involved continued training of the pretrained noisy model weights with the optimizer, learning rate scheduler, and dataloader reset to ensure stable adaptation. The model was evaluated on both noisy and clean test sets to assess generalization performance after adaptation.

7. **Finetune (clean model):** Here, the model initially trained on the clean dataset was fine-tuned using the noisy dataset employed in the noisy model. This setup evaluates the model's capacity to adapt from well-formed, canonical text to noisy and user-generated transliteration data. Similar to the previous setup, the optimizer

TH-3777_186155001

and learning parameters were reinitialized, and epochs were counted as complete passes over the fine-tuning corpus. This ensures the model's ability to learn domain-invariant representations without catastrophic forgetting of prior knowledge.

8. **Finetune (IndicXlit model):** In this setup, the multilingual IndicXlit model was fine-tuned on our noisy Assamese dataset to enhance its performance on informal, real-world data. The pretrained weights of IndicXlit were used as initialization, with full optimizer and scheduler resets during fine-tuning. The model was subsequently evaluated on both noisy and clean test sets. This approach allows the base multilingual model to specialize in Assamese transliteration while maintaining its general multilingual capabilities.

5.5 Results and Discussion

In this section, we discuss the performance of various transliteration setups using multiple evaluation metrics, including top-1 Accuracy (Acc@1), top-4 Accuracy (Acc@4), Character Accuracy, Mean F-score, Mean Reciprocal Rank (MRR), and BLEU4 (up to 4-grams) score as described in section 3.4. The results of all fifteen experimental setups are presented in Table 5.2. Here, Accuracy@k denotes the proportion of test instances where the correct transliteration appears within the top k generated outputs (e.g., Accuracy@1 for the topmost candidate and Accuracy@4 for any of the top four candidates). Character Accuracy measures the proportion of correctly predicted characters, Mean F-score represents the harmonic mean of precision and recall averaged over all test samples, MRR indicates the average reciprocal rank of the first correct prediction, and BLEU4 evaluates the n-gram overlap (up to four-grams) between predicted and reference transliterations. Our analysis reveals that setup 7 achieves the highest Top-1 accuracy of 77.78%, as expected, where both clean datasets are used for training and testing. Conversely, setup 5 exhibits the lowest Top-1 accuracy of 26.87%, where we evaluated the multilingual IndicXlit model on the noisy test set. Additionally, we observe that Acc@4 consistently outperforms Acc@1 across all setups. Below, we provide a detailed discussion of each observation.

1. **Performance Evaluation of Noisy and Clean Transliteration Setups:** Based on our observations, the model trained on both noisy and clean datasets and tested with the respective noisy and clean test sets in setup 1 and setup 7 achieved accuracies of 41.57% and 77.78%, respectively. However, these same models did not perform well when tested with the clean test set in setup 8, resulting in an accuracy

TH-3777_186155001

of 34.71% for the model trained on noisy data, and an accuracy of 28.21% for the model trained on clean data and tested with the noisy test set in setup 2.

2. **Performance Evaluation on Setups With Noisy and Clean Test Set:** The setup that performed the best on the noisy test set was the Merge Model (with duplicate pairs) in setup 4, achieving the highest accuracy of 46.45%. On the other hand, for the clean test set, the clean model trained on the clean dataset in setup 7 achieved the highest accuracy of 77.78% among all the setups.

3. **Impact of Combining Noisy and Clean Datasets with Unique Pairs on Transliteration Model:** In setup 3, we observed a slight increase in accuracy, from 41.57% (setup 1) to 41.88%, by combining both the noisy and clean training sets using unique pairs and testing with the noisy test set. However, when evaluating the same setup with a clean test set in setup 9, the accuracy decreased to 75.04%, which is slightly lower than the 77.78% achieved in setup 7. This suggests that although the transliteration model did not show a significant improvement from being trained with a combined (unique) noisy and clean dataset, it stands out as the single setup that can handle both the noisy and clean dataset together, compared to all other setups.

4. **Effect of Combining Noisy and Clean Datasets with Duplicate Pairs on Transliteration Model:** Combining the noisy and clean datasets with duplicate pairs in setup 4 and setup 10 yielded inconclusive results. In setup 4, the performance was the highest among all the setups tested on the noisy test set, reaching 46.45% accuracy. However, the same setup achieved only 67.47% accuracy when tested with the clean test set, which is lower than the 77.78% and 75.04% accuracy achieved in setup 7 and setup 9, respectively. One plausible reason for this discrepancy could be the variation in the number of duplicate pairs between our noisy and clean datasets. The higher number of duplicate pairs in the noisy dataset, compared to the clean dataset, might be contributing to the observed variations in performance between these two setups.

   We conducted additional investigations to determine whether merging with duplicate pairs, which includes both the clean and noisy datasets and tested on the noisy test set (setup 4), provides any advantage over using a single noisy model trained with duplicate noisy pairs in setup 6 or not. Interestingly, we observed that setup 6 achieved a comparable accuracy of 46.22% in contrast to the 46.45% accuracy achieved in setup 4. Therefore, our conclusion is that merging with duplicates can be beneficial, but it requires a balanced proportion of both the clean and noisy

TH-3777_186155001

datasets. Otherwise, a single model trained with duplicate pairs can yield similar results.

5. **Performance of Multilingual IndicXlit Model:** The state-of-the-art multilingual IndicXlit model, mainly trained on clean or canonical datasets, exhibits suboptimal performance on the noisy test set (setup 5) with an accuracy of only 26.87%, compared to 41.57% in setup 1. However, when evaluated with a clean test set, the same model demonstrates a comparable accuracy of 76.85% (setup 11), similar to the 77.78% accuracy achieved in setup 7. The slight difference in performance between setup 11 and setup 7 may be attributed to out-of-vocabulary(OOV) characters resulting from data cleaning, as discussed in Section 5.3. The presence of large out-of-vocabulary(OOV) characters in the noisy social media test set leads to the lowest accuracy of 26.87% achieved by the multilingual IndicXlit model.

6. **Enhancement through Fine-Tuning:** Finally, we observed that the fine-tuning process did not lead to any significant enhancement in transliteration performance. Specifically, when fine-tuning the model trained with the clean dataset on our noisy set, there was a decrease in accuracy from 41.57% in setup 1 to 36.63% in setup 12. On the other hand, fine-tuning the multilingual IndicXlit model with our noisy dataset resulted in a notable improvement in accuracy, increasing from 26.87% in setup 5 to 38.05% in setup 13, yet still lower than the 41.57% accuracy achieved in setup 1. Similarly, when fine-tuning the Noisy model of setup 1 on the clean dataset in setup 14, a comparable accuracy of 77.07% was attained similar to the 77.78% accuracy observed in setup 7. However, when the same model evaluated with the noisy test set in setup 15, the accuracy dropped down to 28.16%.

In summary, while most transliteration setups performed well only on either the noisy or the clean test set, the MergeUniqueTrans configuration (highlighted in bold) demonstrated consistently strong performance across both. As shown in Table 5.2, the MergeUniqueTrans setups–trained on combined unique pairs and evaluated separately on noisy test set (setup 3) and clean test set(setup 9) – achieved balanced accuracy and robustness, making them the most reliable choice for handling both data conditions effectively. Furthermore, the fine-tuned version of the multilingual IndicXlit model demonstrated clear improvements over the base IndicXlit when evaluated on the noisy test set. Fine-tuning helped the model better handle common sources of transliteration errors such as letter–number mixing within words (e.g., "**fal2**" for "ফাল্টু") and the presence of other special characters. This indicates that training a multilingual transliteration model by combining both noisy and clean datasets can potentially enhance performance and effectively handle both the noisy and clean datasets simultaneously.

Table 5.3: In-depth error analysis on transliteration output from setup 1, setup 3, and setup 5

Input (Roman)	**Native (Ground truth)**	**Merged Model Output (*setup 3*)**	**Noisy Model Output (*setup 1*)**	**IndicXlit Model Output (*setup 5*)**
dhekeri	ঢেকেৰী	ঢেকেৰী	ধেকেৰী	ঢেকেৰী
uddipona	উদ্দীপনা	উদ্দীপনা	উদিপনা	উদ্দীপনা
montra	মন্ত্র	মন্ত্র	মনত্র	মন্ত্রা
ahom	আহোম	আহোম	অসম	আহম
sandubi	চানডুবি	চানডুবি	চান্দুবি	চান্দুবি
ingland	ইংলেণ্ড	ইংলেণ্ড	ইংলান্দ	ইংলেণ্ড
2mak	তোমাক	তোমাক	তোমাক	no output
swali**2**r	ছোৱালীটোৰ	ছোৱালীটোৰ	ছোৱালীটোৰ	no output
axom'ot	অসমত	অসমত	অসমত	no output

Additionally, fine-tuning the multilingual IndicXlit model on the noisy dataset yielded better performance than evaluating the IndicXlit model directly on the noisy test set.

5.5.1 Error Analysis

This section presents key observations derived from the transliteration output, as shown in Table 5.3. Combining the noisy and clean datasets in setup 3 and training the transliteration model with unique pairs resulted in improved performance compared to the solely noisy dataset-trained model in setup 1 while tested on the noisy test set. This finding is interesting as it highlights our efforts to develop models capable of handling both noisy and clean datasets. Remarkably, the same merged model in setup 9, containing unique clean and noisy pairs, achieved comparable results to setup 7 when tested with the clean test set. To gain deeper insights, we conducted an error analysis on transliteration outputs comparing the Noisy model in setup 1 to the Merge model in setup 3. Additionally, the state-of-the-art multilingual IndicXlit model exhibited limited performance on the noisy test set in setup 5 compared to the noisy dataset-trained model in setup 1. We conducted a similar error analysis for the setup 5 transliteration output to explore this further. We marked the transliteration errors with red in Table 5.3.

Our observations revealed that the transliteration model trained solely on noisy social media text in setup 1 encountered challenges in handling *consonant conjuncts* and *named entities*, as evident in Table 5.3. Conversely, the merge model, trained with a combination

TH-3777_186155001

of both noisy and clean datasets in setup 3, demonstrated superior performance in transliterating *consonant conjuncts* and *named entities*. Additionally, both the noisy model of setup 1 and the merged model of setup 3 correctly transliterated *letter-number mixing* and the use of *apostrophes* (') inside words, which are common practices in social media texts. However, the multilingual IndicXlit model, primarily trained on clean/canonical datasets, struggled to cope with such noisy behavior and treated these as out-of-vocabulary (OOV) characters, resulting in no transliteration outputs.

5.5.2 Summary and Future work

In summary, this chapter assessed the performance of the state-of-the-art machine transliteration model in handling both non-canonical social media text and manually forward transliterated canonical data within the context of the Assamese language. Our investigations across 15 different setups revealed that models trained solely on canonical data encountered challenges with ambiguity and noise present in social media text, while models trained solely on noisy data faced limitations with clean canonical data. However, the combined model trained on both canonical and non-canonical data demonstrated comparable results, emphasizing the need for robust models that can effectively handle both clean and noisy transliterations.

In the future, our main focus will be on enhancing the transliteration model to effectively handle both canonical and non-canonical data. Our primary challenge will lie in dealing with the complexities associated with processing real-world noisy social media data in non-canonical form. Additionally, exploring the transliteration of not only non-canonical romanized Assamese data but also investigating the processing of code-mixed Assamese data represents a potential future direction. Given the linguistic diversity in India, addressing this transliteration challenge holds immense importance across various Indian social media scenarios and languages. Furthermore, extending this study to train a multilingual transliteration model that incorporates both canonical and non-canonical data in a unified system shows promising potential for advancing natural language processing across a wide range of linguistic contexts.

TH-3777_186155001

Chapter 6

Study on the Importance of Context in Transliteration

Based on the insights gathered from previous chapters, this chapter highlights the vital role of sentence-level context in improving transliteration or reverse transliteration especially for Romanized Assamese text from social media platforms. Transliteration in low-resource settings encounters challenges due to code-mixed text, where users combine Roman script Assamese with English and other languages. While word-level transliteration approaches analyze individual Romanized words, they often miss context-dependent meanings that can lead to ambiguities. For example, the word "**gai**" can mean "গাই" (meaning: "cow") or "গৈ" (meaning: "goes"), depending on the sentence structure. Without contextual cues, word-level models may struggle with accuracy, emphasizing the need for transliteration methods that consider the entire sentence.

To address this, we evaluate nine state-of-the-art large language models (LLMs) along with the Neural transformer model as the standard baseline, which gave promising results in other sequence-to-sequence tasks, fine-tuned for English-to-Assamese transliteration at both the word and sentence levels. Our evaluation framework ensures a rigorous comparison by assessing sentence-level models on both sentence-level and word-level test sets to measure their ability to retain contextual information while evaluating word-level models exclusively on a word-level test set to highlight their limitations in handling context-dependent ambiguities. Our findings indicate that models based on sentences consistently outperform word-based models, resulting in greater phonetic and semantic accuracy. Furthermore, we investigate the influence of social media data across different platforms on transliteration performance. Since linguistic patterns and writing styles vary across platforms, we analyze how models trained on data from YouTube, Facebook, and Twitter perform across different domains. The findings reinforce that context is critical

TH-3777_186155001

in transliteration, making sentence-aware models essential for handling real-world, code-mixed text more effectively.

6.1 Introduction

The increasing use of social media and digital communication has led to an influx of informal, code-mixed, and Romanized text, particularly in multilingual environments like India. On platforms such as YouTube, Facebook, and Twitter (now X), users often write in their native languages using Roman script, blending it with English and other languages. This widespread practice poses a challenge for natural language processing (NLP) systems that typically rely on native script for accurate analysis, translation, and sentiment detection.

Reverse transliteration, which involves converting Romanized text back into its native script, plays a vital role in addressing this gap. However, most existing literature on transliteration focuses on word-level models that treat individual Romanized words as isolated entities, disregarding the broader sentence context. These word-level models face limitations, especially when dealing with ambiguous words that have different meanings based on their surrounding context. For example, the Romanized Assamese word "**gai**" could be transliterated as either "গাই" (meaning: "cow") or "গৈ", depending on its usage within a sentence. Ignoring this sentence-level context often results in errors that degrade transliteration accuracy.

To address these challenges, this chapter investigates the impact of contextual information on reverse transliteration accuracy. Specifically, we compare the performance of word-level and sentence-level transliteration models to examine how incorporating surrounding words affects accuracy. While word-level models are trained on isolated word pairs, sentence-level models process entire sentences, preserving contextual information that can disambiguate multiple possible transliterations. Our experiments demonstrate that sentence-level models consistently outperform word-level models, achieving higher phonetic and semantic accuracy.

For this study, we integrate nine cutting-edge, pre-trained sequence-to-sequence large language models: mT5-small, mT5-base, ByT5-small, ByT5-base, NLLB-200-distilled-600M, NLLB-200-1.3B, mBART-large-50, mBART-large-50-many-to-many-mmt, M2M-100 along with the Neural Transformer model trained from scratch as the standard baseline. Each model has been fine-tuned for English-to-Assamese transliteration at both

TH-3777_186155001

the word and sentence levels. The evaluation strategy ensures a comprehensive analysis of contextual influence: our sentence-level transliteration models are assessed on both sentence-level and word-level test sets, whereas the word-level models are evaluated exclusively on a word-level test set. This allows us to determine the extent to which contextual information enhances transliteration performance.

Additionally, we explore how transliteration models generalize across different social media platforms. Since platform-specific linguistic patterns and stylistic variations can influence transliteration accuracy, we conduct a cross-platform evaluation using datasets from YouTube, Facebook, and Twitter. By training models on data from one platform and evaluating them on another, we assess how well models adapt to domain shifts. Our results highlight that models trained on Facebook and YouTube data achieve better performance when tested on Twitter data, emphasizing the importance of both contextual information and domain-specific adaptation in improving transliteration quality.

Through these analyses, this chapter provides strong empirical evidence supporting the integration of sentence-level context in transliteration models, demonstrating its role in resolving ambiguities and improving performance in real-world, code-mixed text.

Contribution:

The key contributions of this chapter are:

1. **Comparative Analysis of Word-Level and Sentence-Level Transliteration Models:** To examine the role of contextual information, we compare word-level and sentence-level models, evaluating their accuracy in transliteration. Sentence-level models, trained on full sentences, are tested on both sentence and word-level test sets, while word-level models, trained on isolated word pairs, are assessed solely on word-level data. This analysis highlights the advantages of incorporating surrounding context in transliteration tasks.

2. **Evaluation of Large Language Models (LLMs) for Reverse Transliteration:** We assess nine state-of-the-art LLMs for their effectiveness in converting Romanized Assamese text into its native script.

3. **Cross-Platform Evaluation of Social Media Data:** Using datasets from YouTube, Facebook, and Twitter, we analyze the impact of platform-specific linguistic variations on transliteration performance. Our sentence-level models are evaluated across these platforms to understand how social media context influences model effectiveness.

TH-3777_186155001

6.2 Related Work

Transliteration, particularly for low-resource languages, has been widely studied, with increasing attention to handling informal Romanized text. While early approaches focused primarily on word-level transliteration, recent research highlights the importance of sentence-level context for improving accuracy. Additionally, cross-domain adaptation remains a challenge, as models trained on data from one platform often struggle to generalize across different social media domains. However, systematic studies evaluating sentence-level transliteration in this context remain limited.

Kirov et al. (2022) [169] explore context-aware transliteration for South Asian languages, demonstrating that incorporating monolingual corpora improves performance when parallel data is scarce. Although their study emphasizes contextual information, it does not directly compare word-level and sentence-level transliteration models at scale. Samuel & Straka (2021) [170] presents ByT5-based lexical normalization techniques, showing strong results in handling noisy text but primarily focusing on word-level normalization rather than full-sentence transliteration. Similarly, Kuparinen et al. (2023) [171] investigate dialect-to-standard normalization using character-level transduction, finding that sequence-to-sequence models perform better when trained on full sentences. However, their work does not address cross-domain adaptation. Fahim et al. (2024) [172] introduce a large-scale Bangla transliteration dataset, highlighting the benefits of pretraining on informal Romanized text, yet without systematically assessing the role of sentence-level context or cross-platform generalization.

While these studies underscore the importance of context in transliteration, they do not conduct large-scale evaluations comparing word-level and sentence-level models across different pre-trained language models. Moreover, the impact of cross-domain adaptation across multiple social media platforms remains unexplored. This chapter addresses these gaps by systematically evaluating sentence-aware transliteration and investigating cross-platform adaptation in transliteration tasks.

6.3 Dataset

To evaluate the effectiveness of contextual information in transliteration, particularly reverse transliteration, we curated our in-house diverse back transliteration dataset, "**AssameseBackTranslit**" as describe in sub-section 3.3, consisting of Romanized Assamese text and their corresponding native Assamese script representations. This dataset

TH-3777_186155001

Table 6.1: Sentence-Level and Word-Level Dataset Statistics

Dataset Type	Total Pairs	Train Set	Development Set	Test Set
Sentence-Level	59,783	40,095	6,018	13,670
Word-Level	65,614	45,934	6,560	13,120

was systematically constructed from three major social media platforms—YouTube, Facebook, and Twitter (now X)—to ensure linguistic diversity, domain variability, and robustness against noisy, code-mixed data. The dataset supports both sentence-level and word-level transliteration, enabling a comprehensive assessment of various models across different granularities.

6.3.1 Dataset Statistics

The dataset comprises both sentence-level and word-level transliteration data from Romanized Assamese to native Assamese, systematically divided into three subsets: training, development (validation), and testing. The training set is used to train the model, the development (dev) set is used to tune hyper parameters and prevent overfitting, and the test set is reserved for evaluating the model's performance on unseen data. We have maintained a 70-10-20 train-dev-test split for both word-level and sentence-level transliteration tasks. Detailed dataset statistics are presented in Table 6.1.

The sentence-level dataset consists of 5,000 sentences each from Facebook and Twitter (now X) and approximately 50,000 sentences from YouTube, ensuring a diverse range of linguistic styles and content. The word-level dataset is derived from the same sources, comprising 65,614 unique word pairs extracted from the 59,783 sentence pairs in the sentence-level dataset, allowing for a controlled evaluation of transliteration models at the lexical level. The sentence-level dataset was carefully constructed from the word-level pairs collected from the same three sources. Each sentence includes at least one and a maximum of three representative word-level pairs, ensuring wide lexical coverage while avoiding repetition. Consequently, since multiple distinct word pairs often occur within a single sentence, which results in the sentence-level corpus being smaller in size than the word-level corpus in this experiment. While preparing the sentence-level test set, we ensure that all words from the word-level test set appear at least once in the sentence-level test set, enabling a fair assessment of contextual importance in transliteration.

TH-3777_186155001

The two datasets are designed to capture different aspects of transliteration. The word-level dataset isolates individual word mappings, focusing on direct character-level conversion without any contextual information. In contrast, the sentence-level dataset preserves the surrounding words and syntactic structure, allowing the model to leverage context for more accurate transliteration. Feeding sentence-level data word by word into a word-level model would not provide equivalent results, as the model trained only on isolated words cannot utilize contextual cues. Therefore, maintaining both datasets enables a clear comparison between context-independent and context-aware transliteration performance.

6.3.2 Dataset Statistics for Cross-Domain Adaptation Experiment

To analyze model adaptability across social media platforms, we created three distinct train-dev-test splits while maintaining a consistent distribution of 10,050 training pairs, 3,015 validation pairs, and 2,010 test pairs across all setups. Each split was designed by varying the platform combination used for training and evaluation. In each case, the model was trained on data from two platforms, while the third platform's data was used for validation and testing. Specifically, Split 1 trained on Facebook and YouTube, with Twitter (now X) for validation and testing; Split 2 trained on Facebook and Twitter, with YouTube for evaluation; and Split 3 trained on YouTube and Twitter, with Facebook as the unseen platform. The dataset splits for cross-domain analysis are summarized in Table 6.2.

By leveraging our in-house dataset "**AssameseBackTranslit**", our study provides a rigorous benchmarking framework for evaluating reverse transliteration models. The dataset's carefully curated structure ensures that both word-level and sentence-level transliteration models are tested under realistic, diverse, and noisy conditions, yielding insights into the effectiveness of context-aware approaches in transliteration tasks.

6.4 Experimental Setup

We have integrated a total of nine cutting-edge, pre-trained sequence-to-sequence large language models along with the neural transformer model as standard baseline. Each of the nine large language models has been fine-tuned for English-to-Assamese transliteration tasks at both the word and sentence levels. Our sentence-level transliteration models were evaluated on test sets for both sentence and word levels, while the word-level models

TH-3777_186155001

Table 6.2: Dataset Splits and Sentence Pair Distribution

Splits	Total Sentence Pairs	Train Source	Dev Source	Test Source
Split 1	15,075 (5025 ×3)	Facebook + YouTube (10,050 pairs)	Twitter (now X) (3,015 pairs)	Twitter (now X) (2,010 pairs)
Split 2	15,075 (5025 ×3)	Facebook + Twitter (now X) (10,050 pairs)	YouTube (3,015 pairs)	YouTube (2,010 pairs)
Split 3	15,075 (5025 ×3)	YouTube + Twitter (now X) (10,050 pairs)	Facebook (3,015 pairs)	Facebook (2,010 pairs)

were assessed solely on a word-level test set. Comprehensive details of the datasets for both word and sentence levels are presented in Table 6.1. A brief overview of the nine large language models is provided in the following section, while the neural transformer model corresponds to the one detailed earlier in Section 3.5.

1. **mT5 model:** The mT5 model is a multilingual pre-trained text-to-text transfer transformer designed for a wide range of natural language processing tasks across 101 languages [19]. It utilizes SentencePiece [173] for subword tokenization and is trained on the multilingual variant of the C4 dataset [121], known as mC4 [19]. The model follows the "Span Corruption" pre-training objective, where spans of tokens in unlabeled text are randomly masked and must be predicted. mT5 employs a balanced encoder-decoder architecture optimized for both generation and classification tasks. However, Assamese is not included in its training set, which may affect its performance for this language. We have fine-tuned two variations of the mT5 model, each differing in the total number of parameters, to better adapt it for our specific tasks.

    - **mT5-small:** The mT5-small[1] model consists of 300 million parameters, with a hidden dimension (dmodel) of 512, a feed-forward dimension (dff) of 1024, and 8 encoder-decoder layers.

[1] mt5-small

TH-3777_186155001

- **mT5-base:** The mT5-base[2] model is a larger variant, containing 582M parameters, a hidden dimension of 768, a feed-forward dimension of 2048, and 12 encoder-decoder layers.

2. **ByT5 model:** The ByT5 model is a multilingual pre-trained text-to-text transfer transformer that differs from mT5 by eliminating the need for subword tokenization [18]. Instead of using SentencePiece [173] tokenizer, ByT5 processes raw UTF-8 bytes directly, making it a fully tokenizer-free model. Like mT5, it is also trained on the mC4 [19] dataset, using a modified "Span Corruption" pre-training objective, where masked spans are replaced with learned byte representations. ByT5 employs a deep encoder architecture with a reduced decoder capacity, optimizing it for both classification and generation tasks. However, Assamese is not included in its training set. We have fine-tuned two variations of the ByT5 model, each differing in the total number of parameters, to better adapt it for our specific tasks.

   - **ByT5-small:** The ByT5-small[3] model consists of 300 million parameters, with a hidden dimension (dmodel) of 1472, a feed-forward dimension (dff) of 3584, and an encoder-decoder structure of 12 and 4 layers, respectively.
   - **ByT5-base:** The ByT5-base[4] model is a larger variant, containing 582 million parameters, a hidden dimension of 1536, a feed-forward dimension of 3968, and an encoder-decoder structure of 18 and 6 layers, respectively.

3. **NLLB-200 model:** The NLLB-200 model, developed by Meta AI [174], is a large-scale multilingual machine translation model trained on 200 languages, including Assamese, using a Sparsely Gated Mixture of Experts architecture. It was trained on extensive multilingual text, with preprocessing done using SentencePiece tokenizer, and evaluated on the Flores-200 dataset. Optimized for general-domain translation with a 512 token limit, it is primarily intended for research in low-resource language translation. We have fine-tuned two variations of NLLB-200 for Romanized Assamese to native Assamese transliteration, enhancing its ability to accurately perform reverse transliteration while maintaining linguistic consistency.

   - **NLLB-200-distilled-600M[5]:** This variant has 600 million parameters, making it a smaller and more efficient model while still maintaining strong performance. It is optimized for faster inference and lower computational requirements, making it suitable for resource-constrained environments.

[2] mT5-base
[3] ByT5-small
[4] ByT5-base
[5] NLLB-200-distilled-600M

TH-3777_186155001

- **NLLB-200-1.3B[6]:** This variant has 1.3 billion parameters, making it a larger and more capable model for multilingual translation and transliteration tasks. It provides higher accuracy, particularly for low-resource languages, due to its increased capacity and richer representations.

4. **mBART-large-50 model:** mBART-large-50[7], developed by Meta AI [175], is a multilingual Sequence-to-Sequence model pre-trained using the Multilingual Denoising Pretraining objective, which applies sentence order shuffling and masked span reconstruction to enhance language representations. It extends the original mBART model to 50 languages, utilizing large-scale unlabeled monolingual data to improve support for low-resource languages. However, Assamese is not part of the training data. Unlike models trained from scratch, mBART-large-50 undergoes multilingual fine-tuning on multiple translation directions simultaneously, leveraging both pretraining and fine-tuning techniques. The model employs SentencePiece tokenization and a language ID token (LID) to direct translations. To address this, we have fine-tuned mBART-large-50 for Romanized Assamese to native Assamese transliteration, ensuring accurate conversion while maintaining linguistic consistency.

5. **mBART-large-50-many-to-many-mmt(mBART-large-50-m2m-mmt):** mBART-large-50-many-to-many-mmt(mBART-large-50-m2m-mmt[8]) is a fine-tuned version of mBART-large-50, designed specifically for multilingual machine translation. Introduced in the Meta AI paper [176], this model enables direct translation between any pair of 50 languages without requiring pivot languages. It leverages pretrained multilingual representations and fine-tuning on parallel data to improve translation quality across diverse language pairs.

6. **M2M-100 model:** The M2M-100[9] model is a many-to-many multilingual translation model developed by Facebook AI [177], capable of translating directly between 100 languages without relying on English as an intermediary. It was trained on a large-scale dataset covering thousands of language pairs, emphasizing non-English-centric translation. The model has 418 million parameters and features a 12-layer encoder-decoder architecture with 1024 hidden dimensions, 16 attention heads, and 4096 feed-forward dimensions and it uses a special language ID token at the beginning of both source and target text, and translation is controlled by the `forced_bos_token_id` parameter. m2m-100 enables high-quality multilingual

[6] NLLB-200-1.3B
[7] mBART-large-50
[8] mBART-large-50-m2m-mmt
[9] M2M-100_418M

TH-3777_186155001

translation, particularly improving direct translations between low-resource languages.

In our experimental setup, we fine-tuned multiple models on Assamese text to evaluate their transliteration and translation performance. Among these, only the NLLB models included Assamese in their original pre-training data. The other multilingual models—such as mT5 and ByT5—did not have Assamese coverage, though they included Bengali as a pre-trained language. Since Assamese and Bengali belong to the same language family and share a nearly identical script (differing in only two characters "ৰ" and "ৱ"), we treated Assamese text as Bengali during fine-tuning. This approach allowed the models to leverage the linguistic and orthographic similarities between the two languages, partially compensating for the lack of Assamese-specific pre-training.

6.5 Results and Discussion

We have conducted a comprehensive comparative analysis of word-level and sentence-level transliteration results across all nine large language models, as presented in Table 6.3. Additionally, the results of the cross-domain adaptation analysis are provided in Table 6.4.

6.5.1 Analysis of Word-Level vs. Sentence-Level Transliteration Models

To ensure a comprehensive evaluation, four complementary metrics were used. Word Error Rate (WER) and Character Error Rate (CER) measure the proportion of insertion, deletion, and substitution errors at the word and character levels, respectively, providing insights into fine-grained transliteration accuracy. CHRF(Character F-score) computes the F-score over character n-grams and is more tolerant to minor spelling variations, while BLEU (up to 4-grams) evaluates the n-gram overlap between predicted and reference outputs, indicating fluency and overall contextual alignment. Therefore, WER and CER are particularly suitable for assessing character-level transliteration accuracy, whereas CHRF and BLEU are more appropriate for evaluating sentence-level transliteration quality. We have briefly described all the evaluation metrices in Section 3.4. The results in Table 6.3 clearly indicate that sentence-level models outperform word-level models across all metrics, including Word Error Rate (WER), Character Error Rate (CER), Character F-score (CHRF), and BLEU score. Among all models, ByT5-base (sentence-level) achieves the

TH-3777_186155001

Table 6.3: Comparison of Word-level Vs Sentence-level Transliteration Models

Sl. No.	LLM Models	Model Type	Test Set	WER	CER	CHRF	BLEU
1	Transformer	Sent. Model	Sent. Test	20.45	9.65	81.90	61.85
			Word Test	65.10	21.10	63.70	65.20
		Word Model	Word Test	67.55	23.80	62.40	64.40
2	mT5-base	Sent. Model	Sent. Test	23.43	11.62	78.40	54.63
			Word Test	75.25	46.63	43.70	44.10
		Word Model	Word Test	63.47	23.94	61.60	64.47
3	mT5-small	Sent. Model	Sent. Test	35.45	22.80	68.90	41.34
			Word Test	86.65	65.34	27.5	28.19
		Word Model	Word Test	69.73	25.30	53.50	59.58
4	ByT5-base	Sent. Model	Sent. Test	**16.23**	**7.09**	**85.20**	**67.32**
			Word Test	59.08	20.26	68.16	71.26
		Word Model	Word Test	59.88	21.49	67.60	69.37
5	ByT5-small	Sent. Model	Sent. Test	18.92	8.84	83.50	63.58
			Word Test	63.02	20.24	65.10	66.78
		Word Model	Word Test	66.36	22.94	64.30	65.93
6	NLLB-200-distilled-600M	Sent. Model	Sent. Test	22.88	11.01	79.90	56.16
			Word Test	67.90	32.95	52.9	56.21
		Word Model	Word Test	64.46	22.40	62.9	66.39
7	NLLB-200-1.3B	Sent. Model	Sent. Test	21.84	9.84	80.60	57.16
			Word Test	66.21	32.31	54.90	57.94
		Word Model	Word Test	68.48	23.39	61.00	64.98
8	mBART-large-50	Sent. Model	Sent. Test	23.39	11.12	78.70	54.95
			Word Test	68.08	34.58	51.60	54.51
		Word Model	Word Test	69.41	23.49	58.10	63.54
9	mBART-large-50-m2m-mmt	Sent. Model	Sent. Test	23.11	10.71	79.00	54.96
			Word Test	67.31	33.43	52.70	55.80
		Word Model	Word Test	69.95	23.74	57.70	63.19
10	M2M-100	Sent. Model	Sent. Test	40.30	14.29	66.70	30.04
			Word Test	79.49	37.79	41.80	47.97
		Word Model	Word Test	80.45	28.89	46.20	53.46

best results, with the lowest WER (16.23) and CER (7.09), along with the highest CHRF (85.20) and BLEU (67.32) scores. This suggests that using a sentence-based approach provides significantly better context, leading to more accurate transliteration. In contrast, word-level models, even when using the same underlying architecture, show much higher error rates, demonstrating that the lack of surrounding context negatively impacts performance.

A closer examination of the models reveals that ByT5 models (both base and small variants) consistently outperform other architectures. This can be attributed to ByT5's byte-level tokenization, which allows it to capture fine-grained subword and character-level dependencies more effectively. Unlike other models that rely on subword segmentation (e.g., SentencePiece used in mBART and NLLB-200), ByT5 operates at the byte level, making it highly effective for transliteration, where precise character-level transformations are crucial. This byte-based approach ensures that rare or unseen characters are better handled, reducing errors in transliteration.

TH-3777_186155001

The results also highlight the significant gap between sentence and word-level models. For instance, ByT5-base sentence-level model achieves a WER of 16.23, whereas its word-level counterpart records a much higher WER of 59.88. Similarly, the NLLB-200-1.3B sentence-level model records a WER of 26.41, whereas its word-level version has 68.48, showing a similar pattern. This discrepancy suggests that transliteration benefits significantly from larger context windows. In sentence-level models, the network has access to surrounding words, allowing it to make more informed predictions, whereas word-level models treat each word in isolation, losing crucial linguistic and phonetic relationships.

Looking at other architectures, the Transformer model trained from scratch performs competitively but remains below the ByT5 variants across all evaluation metrics. While it benefits from parallel attention mechanisms and performs reasonably well at the word level, its limited exposure to byte-level representations makes it less effective in capturing fine-grained orthographic variations. In contrast, ByT5 models, which operate directly on raw byte sequences, demonstrate superior robustness to character-level inconsistencies. NLLB-200 and mBART models exhibit comparatively weaker performance, which can be attributed to their reliance on subword tokenization, leading to fragmentation issues in transliteration. Additionally, M2M-100 performs the worst, with the highest error rates across both sentence and word-level tasks. This suggests that while M2M-100 is effective for multilingual translation, it is not optimized for transliteration, as its architecture prioritizes sentence-level semantic representations over precise character-level mappings.

Overall, the results strongly favor sentence-level transliteration models, particularly those using byte-level processing, such as ByT5, which excel in handling complex character transformations. The superior performance of ByT5-base and ByT5-small demonstrates that byte-based models are inherently more suitable for transliteration tasks, as they can effectively model character dependencies without being constrained by subword tokenization artifacts. Given these findings, sentence-level ByT5 models are recommended for achieving the highest transliteration accuracy.

6.5.2 Cross-Domain Adaptation Analysis

The cross-domain adaptation analysis investigates how transformer-based and other large language models (LLMs) generalize across different training and testing domains. As presented in Table 6.4, training on a combination of two domains can, in certain cases, enhance performance on an unseen third domain–particularly when the source domains exhibit stylistic or lexical similarity to the target domain. For instance, the ByT5-base model, when trained on the Facebook + YouTube combination (Split 1) and evaluated on

TH-3777_186155001

Table 6.4: Cross Domain Adaptation Analysis

Sl. No.	LLM Models	Different Training Splits	Dev + Test Set	WER	CER	CHRF	BLEU
1	Transformer	**Split 1** [Facebook + YouTube]	Twitter (now X)	12.25	6.20	87.80	72.50
		Split 2 [Facebook + Twitter(now X)]	YouTube	22.45	9.40	79.30	56.10
		Split 3 [YouTube + Twitter(now X)]	Facebook	24.10	12.85	78.90	57.20
2	mT5-base	**Split 1** [Facebook + YouTube]	Twitter (now X)	16.57	7.31	83.9	64.50
		Split 2 [Facebook + Twitter(now X)]	YouTube	29.65	15.69	72.00	44.80
		Split 3 [YouTube + Twitter(now X)]	Facebook	24.04	12.33	76.10	53.30
3	ByT5-base	**Split 1** [Facebook + YouTube]	Twitter (now X)	9.01	3.84	91.50	80.30
		Split 2 [Facebook + Twitter(now X)]	YouTube	18.55	7.62	82.40	61.90
		Split 3 [YouTube + Twitter(now X)]	Facebook	15.51	7.25	84.90	67.70
4	ByT5-small	**Split 1** [Facebook + YouTube]	Twitter (now X)	11.41	5.69	89.30	76.20
		Split 2 [Facebook + Twitter(now X)]	YouTube	21.24	8.79	80.00	57.40
		Split 3 [YouTube + Twitter(now X)]	Facebook	23.32	13.32	80.60	58.70
5	NLLB-200-distilled-600M	**Split 1** [Facebook + YouTube]	Twitter (now X)	15.34	6.60	85.60	67.10
		Split 2 [Facebook + Twitter(now X)]	YouTube	24.31	10.68	77.10	51.80
		Split 3 [YouTube + Twitter(now X)]	Facebook	21.59	11.08	79.50	57.80

Twitter, achieves the best cross-domain results with a WER of 9.01, CER of 3.84, CHRF of 91.50, and BLEU of 80.30, demonstrating strong generalization ability. In contrast, training on Facebook + Twitter (Split 2) and testing on YouTube results in a noticeable performance decline (WER 18.55, CER 7.62, CHRF 82.40, BLEU 61.90), indicating that the linguistic characteristics of YouTube data differ significantly from those of Facebook and Twitter, thereby posing greater challenges for generalization. Based on the findings in Table 6.3, among the ten evaluated models, only five–ByT5-base, ByT5-small, Transformer, NLLB-200-distilled-600M, and mT5-base–exhibited consistently strong results. Consequently, these five models were selected for the cross-domain adaptation analysis.

Among the evaluated models, ByT5 (both base and small variants) consistently achieves the best cross-domain results, obtaining the lowest WER and CER values across all domain splits. This strong performance highlights the model's robustness to orthographic variability and its capacity to handle noisy transliteration patterns. NLLB-200-distilled-600M also performs competitively, particularly on Twitter, where it achieves a WER of

TH-3777_186155001

15.34 and CHRF 85.60, demonstrating effective multilingual generalization. The mT5-base model follows closely, benefitting from large-scale pretraining but showing slightly higher error rates on domains with heavier social-media noise, such as YouTube. The Transformer baseline lags behind pre-trained models, suggesting that large-scale multilingual pretraining plays a crucial role in achieving better cross-domain adaptability.

The observed performance differences can be attributed to the tokenization and pretraining strategies used by each model. ByT5, with its byte-level tokenization, is inherently resilient to spelling inconsistencies and diverse transliteration patterns across domains. NLLB benefits from extensive multilingual pretraining, enabling it to generalize effectively even with limited Assamese data. mT5, although also pretrained on multilingual data, operates at the subword level, making it less sensitive to character-level variations in transliteration. In contrast, the standard Transformer, trained from scratch, lacks this pretraining advantage and thus exhibits reduced generalization capability. Overall, ByT5 emerges as the most adaptable model for cross-domain transliteration, while the Transformer model serves as a useful baseline reference.

6.6 Summary and Future Work

This chapter presents a comprehensive evaluation of word-level and sentence-level transliteration models, along with an analysis of cross-domain adaptation in large language models (LLMs). The results demonstrate that sentence-level models significantly outperform word-level models across all evaluation metrics, emphasizing the importance of contextual information in transliteration tasks. Among all models, ByT5-base (sentence-level) achieves the highest accuracy, benefiting from its byte-level tokenization, which effectively captures fine-grained character dependencies. In contrast, word-level models exhibit higher error rates, highlighting the limitations of isolated word processing. ByT5 models consistently outperform other architectures, particularly subword-based models like mBART and NLLB-200, which struggle with transliteration due to segmentation inconsistencies. The findings reinforce that transliteration benefits from sentence-level modeling and byte-based tokenization. In cross-domain adaptation, ByT5-base demonstrates the best adaptability across different domains like Facebook, YouTube, and Twitter. While NLLB models also perform well due to their multilingual pretraining, mBART and M2M-100 show weaker adaptation, especially with informal text from platforms like YouTube.

For future work, exploring more advanced pretrained and instruction-tuned generative foundation models such as LLaMA [178] could further enhance transliteration quality.

TH-3777_186155001

LLaMA models, known for their strong generalization capabilities and efficiency in handling multilingual data, could provide better contextual understanding and improved character-level generation, making them promising candidates for transliteration tasks. Additionally, domain-specific fine-tuning can improve adaptation across various platforms, particularly in handling informal and noisy text. Investigating hybrid tokenization techniques, combining subword and byte-level approaches, could mitigate segmentation inconsistencies and improve rare character handling. Furthermore, applying transfer learning from high-resource to low-resource languages and incorporating diverse datasets may enhance model robustness and generalization in transliteration across multiple domains.

TH-3777_186155001

TH-3777_186155001

Chapter 7

Study on the Importance of Transliteration for Downstream Applications

In this chapter, we aimed to demonstrate the importance of reverse transliteration—converting romanized text back into its native script—by employing an extrinsic evaluation using a downstream NLP task: Sentiment Analysis. This study builds upon previous chapters, where we explored the challenges of romanized text and the benefits of reverse transliteration. To validate our hypothesis, we created a novel Sentiment Analysis dataset containing 15,075 Roman-Native parallel sentences sourced from three social media platforms: Facebook, YouTube, and Twitter (now X). The dataset was annotated with three sentiment labels: Positive (+1), Negative (-1), and Neutral (0). We evaluated three state-of-the-art sentiment classification models (MLP and BERT) considering different embeddings: IndicBERT v2 [179], BERT-multilingual [180], and IndiSocialFT [181], and found that models trained and tested on native-script data consistently outperformed those using romanized data. This underscores the significance of reverse transliteration in improving sentiment classification accuracy.

7.1 Introduction

Sentiment analysis plays a crucial role in understanding public opinion, particularly in the context of social media. However, the presence of romanized text in code-mixed languages introduces noise, making sentiment classification challenging. Reverse transliteration, which converts romanized text back into its native script, has the potential to improve

TH-3777_186155001

sentiment classification by providing cleaner, more standardized textual representations. This chapter presents an extrinsic evaluation of reverse transliteration through sentiment analysis, demonstrating how it enhances the performance of NLP models.

Contribution:

The major contributions of this chapter are:

- Creation of a novel Sentiment Analysis dataset comprising 15,076 Roman-Native parallel sentences with sentiment labels.
- Evaluation of three state-of-the-art sentiment classification models on both romanized and it's native-script counter part.
- Investigation into the impact of script choice on sentiment classification, emphasizing the necessity of reverse transliteration. This explores how the representation of text in romanized versus native script affects the ability of models to accurately capture sentiment, considering linguistic nuances, orthographic structure, and contextual dependencies that may be lost or altered during romanization.

7.2 Related Work

Duwairi *et al.*, 2016 [182] conducted sentiment analysis on Arabizi tweets by first transliterating them into Arabic using a rule-based converter. Their study found that sentiment classification models (SVM and Naïve Bayes) performed better on Arabic-transliterated data than on raw Arabizi text, highlighting the significance of script normalization in downstream NLP tasks. However, their rule-based transliteration system struggled with ambiguities and variations, which negatively impacted sentiment analysis performance.

Similarly, Guellil *et al.*, 2018 [183] explored sentiment analysis on Arabizi by first converting it back to Arabic. Their findings suggest that reverse transliteration can enhance sentiment analysis performance, although challenges persist due to transliteration inaccuracies. While the study does not explicitly compare sentiment analysis results on romanized versus native-script data, it indicates that reverse transliteration could be beneficial for NLP tasks like sentiment classification.

Roychoudhury *et al.*, 2022 [184] investigated cross-lingual sentiment analysis using transliteration and character embeddings to create a metalingual representation for low-resource languages. Their work primarily focused on forward transliteration, transforming data

TH-3777_186155001

from resource-constrained languages into a standardized representation to improve sentiment analysis. Using WX-notations as the transliteration approach, their findings suggest that transliteration-based transformations can enhance sentiment analysis performance. However, they did not evaluate sentiment analysis performance by directly comparing transliterated or romanized text with its native-script counterpart.

Existing research on sentiment analysis of romanized languages relies heavily on rule-based transliteration to convert text into its native script before classification. However, these approaches struggle with inconsistencies and ambiguities in transliteration, which can negatively impact sentiment analysis performance. In contrast, we employed native human annotators for the conversion process, ensuring higher accuracy and preserving the contextual meaning of words. Additionally, most studies do not compare the effectiveness of sentiment classification on romanized versus native-script text, leaving gaps in understanding the necessity of transliteration. Furthermore, traditional machine learning methods are predominantly used, while more advanced classification techniques remain underexplored.

7.3 Dataset

The dataset used in this study consists of 15,076 Roman–Native parallel sentences, collected equally from Facebook, YouTube, and Twitter. Each Romanized Assamese sentence was paired with its corresponding Native Assamese form and annotated with a sentiment label. The annotation process involved nine native Assamese speakers who were fluent in both Assamese and English. Each sentence was annotated by three annotators, and the final sentiment label was determined by majority voting among the three. The dataset was labeled using three sentiment categories: **Positive (+1)**, **Negative (−1)**, and **Neutral (0)**.

During the creation of the native side of the corpus, the Romanized Assamese sentences were manually transliterated back into Assamese script by the same group of annotators. While transliterating, the embedded non-native (English) words were retained in their original correct forms. Additionally, social media noise: such as spelling variations, elongated or shortened native words, and informal orthography was normalized to standard Assamese script. As a result, the Native Assamese portion of the dataset effectively represents a combination of *pure native Assamese* and *mixed-code (Romanized + Assamese)* sentences. This occurs because the original Romanized Assamese content often includes both Romanized Assamese words and English words (in either correct or noisy form). The presence of this mixed-code phenomenon is also evident in Table 7.2. Therefore, the

TH-3777_186155001

Table 7.1: Sentiment Analysis Dataset Statistics

Statistic	**Value**
Total Sentences	15,075 (5025 sentences from each sources)
Sources	Facebook, YouTube, Twitter (now X)
Positive Samples (+1)	3,709
Negative Samples (-1)	6,852
Neutral Samples (0)	4,515
Number of Annotators[1]	9

sentiment analysis results reported on the Native Assamese side of the dataset inherently reflect the system's performance on *mixed-code data*, since this portion captures both purely native and mixed-code textual patterns.]

7.3.1 Dataset Statistics

We examine the dataset used for training and evaluation by analyzing key statistics, including the number of sentences, sources, sentiment distribution (positive, negative, and neutral samples), and the number of annotators across three diverse social media platforms. Table 7.1 provides a detailed breakdown, offering insights into linguistic diversity and data distribution across platforms.

7.3.2 Example Sentences

We present a set of seven example sentences in Romanized Assamese and their corresponding Native Assamese forms, each annotated with sentiment tags (positive, negative, or neutral). These examples, shown in Table 7.2, illustrate the linguistic variations between Romanized and Native Assamese text and highlight sentiment distribution in the dataset.

7.3.3 Evaluation Metrics

To evaluate the performance of the sentiment classification models, four standard metrics were used: **Accuracy**, **Precision**, **Recall**, and **F1-score**. These metrics provide a comprehensive assessment of the model's classification performance across the three sentiment classes — *Positive*, *Negative*, and *Neutral*.

[1]Each sentence was evaluated by 3 annotators, with majority voting determining the final sentiment label.

TH-3777_186155001

- **Accuracy:** The ratio of correctly classified instances to the total number of instances, providing an overall measure of model correctness.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.1}$$

- **Precision:** The ratio of correctly classified instances of a particular class to all instances predicted as belonging to that class. It reflects the model's ability to avoid false positives.

$$Precision = \frac{TP}{TP + FP} \tag{7.2}$$

- **Recall:** The ratio of correctly classified instances of a particular class to all actual instances of that class. It measures the model's ability to capture all relevant cases.

$$Recall = \frac{TP}{TP + FN} \tag{7.3}$$

- **F1-score:** The harmonic mean of Precision and Recall, balancing both metrics to provide a single measure of classification performance, particularly useful in class-imbalanced settings.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7.4}$$

For multi-class evaluation, Precision, Recall, and F1-score are **macro-averaged** across the three sentiment classes. This ensures that each class — Positive, Negative, and Neutral — contributes equally to the overall metric, regardless of class distribution.

7.4 Experimental Setup

In this section, we describe the experimental setup used in our study, including the three models evaluated using the evaluation metrics employed.

- **IndicBERT v2:** IndicBERT v2 [179] is a multilingual language model designed for Indic languages, trained on IndicCorp v2, a dataset containing 20.9 billion tokens across 24 languages. It is evaluated using IndicXTREME, a benchmark covering nine Natural Language Understanding (NLU) tasks and 105 evaluation sets in 20 Indic languages. The model is trained using Masked Language Modeling (MLM) and incorporates additional objectives like Translation Language Modeling (TLM)

TH-3777_186155001

Table 7.2: Examples from the Sentiment Analysis Dataset: Roman-Native Parallel Sentences with English Glosses

Roman Text	Native Text	English Gloss	Sentiment Tag
record r namot ulta pulta ako nkriba bhai hot	record ৰ নামত উল্তা পুল্তা একো নকৰিব ভাই হত	Don't do unnecessary things in the name of record, brothers.	-1
sir pic clear aha nai so pohibo pra nai	sir picture clear আহা নাই so পঢ়িব পৰা নাই	Sir, the picture isn't clear, can't read it.	-1
saru katha moi sash kari bahi aso ane	সৰু কথা মই শেষ কৰি বহি আছো এনে	Small task, I have finished it and sitting idle.	0
sop eibur muslim bilakar karene hoise	সৰ্ এইবোৰ মুচলিম বিলাকৰ কাৰণে হৈছে	All this happened because of Muslims.	-1
antorik dhanyabad atailoi	আন্তৰিক ধন্যবাদ আটাইলৈ	Heartfelt thanks to everyone.	1
nishsoy ami xaju asu	নিশ্চয় আমি সাজু আছো	Surely, we are ready.	1
efrom joriyote	efrom জৰিয়তে	Through Efrom (via Efrom).	0

with the Samanantar parallel corpus and back-translation via IndicTrans. Additionally, a variant called IndicBERT-SS converts Indic scripts to Devanagari before training to enhance lexical sharing.

- **mBERT (Multilingual BERT):** mBERT (Multilingual BERT) [180] is a general-purpose multilingual version of BERT, pre-trained on Wikipedia data in 104 languages, including Indic languages. Unlike IndicBERT v2, which is specifically optimized for Indic languages, mBERT employs Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) for training. Although it is trained on a smaller dataset compared to IndicBERT v2 and lacks fine-grained language-specific optimizations, its broad multilingual training makes it valuable for cross-lingual transfer learning.

- **IndiSocialFT:** IndiSocialFT [181] is a FastText-based word embedding [185] model designed to handle native scripts, transliterated text, multilingual content, code-mixing, and social media text in Indian languages. Trained on a diverse corpus combining social media data and well-formed text, it provides robust representations suited for informal and noisy text. Through intrinsic and extrinsic evaluations, IndiSocialFT outperforms other pre-trained embeddings, demonstrating superior adaptability for Indian language NLP applications.

TH-3777_186155001

Table 7.3: Performance of three state-of-the-art classification methods on our sentiment analysis dataset, separately for Romanized and Native Assamese text. Both micro and weighted scores are reported for Precision, Recall, and F1-Score. Highest and lowest performing Model–Embedding combinations are highlighted in blue and red, respectively.

Model	Embedding	Data	Accuracy	Precision		Recall		F1-Score	
				Micro	Weighted	Micro	Weighted	Micro	Weighted
MLP	IndiSocialFT	Roman	0.625	0.651	0.650	0.586	0.625	0.596	0.616
		Native	0.650	0.624	0.624	0.612	0.650	0.522	0.641
BERT	**IndicBERT v2**	**Roman**	0.641	0.633	0.638	0.612	0.641	0.619	0.635
		Native	0.676	0.660	0.680	0.667	0.676	0.663	0.678
	bert-multilingual	Roman	0.622	0.608	0.623	0.608	0.622	0.608	0.622
		Native	0.627	0.614	0.637	0.625	0.627	0.616	0.629

7.5 Results and Discussion

In this study, we aimed to demonstrate the significance of reverse transliteration using sentiment analysis as the downstream NLP task. The results, presented in Table 7.3, reveal that IndicBERT v2 outperforms all other models, achieving the highest accuracy and F1-score across both Romanized and Native Assamese datasets. IndicBERT v2 showed an accuracy of 67.6% on native-script data, while its performance dropped to 64.1% on Romanized data. Similarly, IndiSocialFT and BERT-multilingual exhibited better performance on the native-script dataset, with a clear drop in performance for the Romanized dataset. This pattern emphasizes the difficulty in working with Romanized text, which often introduces inconsistencies due to one-to-many and many-to-one transliteration variations, whereas the native-script data provides more consistency for accurate sentiment classification.

These findings highlight the critical role of reverse transliteration in improving sentiment analysis accuracy. By converting Romanized text back into its native script, models benefit from cleaner and more reliable inputs, resulting in significantly better performance. This extrinsic evaluation emphasizes the necessity of reverse transliteration in processing code-mixed and noisy social media text, showcasing its broader impact on enhancing the effectiveness of downstream NLP tasks for low-resource languages.

7.6 Error Analysis

Table 7.4 illustrates representative cases where sentiment predictions differ between Romanized and back-transliterated native Assamese text for the best performing IndicBERT v2 model. Across the complete test set of 3,016 sentences, models trained and evaluated

Table 7.4: Examples showing differences in sentiment prediction between Romanized and Native Assamese text. The native predictions align more closely with the ground truth due to normalization of variant spellings (e.g., *bohut/bhut/bahut/bhot* → বহুত, *moja/mojja/mojaa/mojjah* → মজা). Sentiment labels are denoted as: 1 = Positive, −1 = Negative and 0 = Neutral.

Romanized Text	**Native Text (Back-Transliterated)**	**English Gloss**	**True Label**	**Pred. (Roman)**	**Pred. (Native)**
ooooooo **mojaa** hoise aru kaitaman ulai ahibo lage tatia tel sai val lagibo	ওওওওও মজা হৈছে আৰু কেইটামান উলাই আহিব লাগে তেতিয়া তেল চাই ভাল লাগিব	It's so much fun; a few more should come out, then it'll look even better.	1	-1	1
mojja kotha koise olop 2 hikkhito ho olop 2 laaz pa	মজা কথা কৈছে অলপ টো শিক্ষিত হ' অলপ টো লাজ পা	You've spoken jokingly—be a little educated and have some shame.	-1	0	-1
@himantabiswa @bjp4india o mama mur baa biya 8th may **bht** kosto k biya khn gutaisu please biya khn hoi jbole debo please mama please reply debo ami bht tension ot asu	@himantabiswa @bjp4india অ' মামা মোৰ বা বিয়া 8th may বহুত কষ্ট কে বিয়া খন গোটাইছো please বিয়া খন হয় যাবলে দিব please মামা please reply দিব আমি বহুত tension অ'ত আছো	Sir, I've worked very hard to arrange my wedding on 8th May, please allow it to happen; we are in great tension.	1	-1	1
bhut dukh lgise axom t a axomia mnuh r ai obstha hoise	বহুত দুখ লাগিছে অসম অত এ অসমীয়া মানুহ অৰ এই অৱস্থা হৈছে	It's very sad to see the present condition of Assamese people in Assam itself	-1	1	-1
masjid ot namaz porhibo laage raas-taat porhibo...	মছজিদ ত নামাজ পঢ়িব লাগে ৰাস্তাত পঢ়িব কোনে কৈছে...	One should pray in the mosque, who ask to pray on the street...	1	0	1

on native text achieved 67.14 % accuracy compared with 62.43 % on Romanized text, a consistent +4.71 percentage-point improvement. This advantage largely arises from normalization effects achieved through back-transliteration. On social media, a single Assamese word often appears in multiple Roman variants–such as bohut, bhut, bahut, bhot for "বহুত" or moja, mojja, mojaa, mojjah for "মজা". Such inconsistent spellings fragment the input distribution and weaken token-level sentiment cues. Back-transliteration restores these lexical forms to a unified representation, allowing the model to correctly interpret key affective expressions and polarity indicators.

Nevertheless, residual errors persist in highly code-mixed or context-sparse posts, as well as in sarcastic or ironic remarks where lexical sentiment alone is insufficient. In rare instances, Roman predictions surpass native ones when English sentiment words dominate the message. Overall, as evidenced in Table 7.4, native script normalization substantially improves model alignment with true sentiment labels and enhances overall robustness in downstream analysis.

TH-3777_186155001

7.7 Summary and Future Work

This chapter demonstrated that reverse transliteration plays a vital role in improving the performance of downstream NLP tasks, with sentiment analysis serving as an example. Our findings showed that sentiment classification models trained and tested on native-script Assamese consistently outperformed those trained on Romanized text, highlighting the challenges posed by transliteration inconsistencies and noise in informal social media text. By leveraging reverse transliteration, we can enhance text quality, leading to more accurate and reliable NLP models for low-resource languages.

Beyond sentiment analysis, reverse transliteration can also benefit other downstream NLP tasks, such as machine translation, named entity recognition, and text classification, by providing cleaner and more structured input data. Future work can explore other pre-trained models to assess their effectiveness in handling Romanized and native-script text. Additionally, expanding the dataset with more code-mixed and informal text from diverse social media platforms could help develop robust NLP systems for real-world applications.

TH-3777_186155001

TH-3777_186155001

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This thesis explored various aspects of transliteration and reverse transliteration, particularly in the context of Assamese language processing. Beginning with the creation of a novel back transliteration dataset, we established a foundation for machine transliteration research in Assamese, capturing linguistic variations from social media platforms. We then analyzed the transliteration characteristics of Romanized Assamese text, providing insights into the challenges of informal and non-standard transliteration. Through empirical evaluations of monolingual and multilingual transliteration models, we examined their ability to handle both canonical and noisy social media data, identifying the strengths and limitations of different modeling approaches, from statistical and neural methods to advanced transformer-based pre-trained large language models (LLMs)..

The context-aware lexical deromanization study demonstrated that sentence-level models significantly outperform word-level models, highlighting the necessity of contextual information in transliteration tasks. Furthermore, the extrinsic evaluation in sentiment analysis confirmed that reverse transliteration enhances downstream NLP tasks, as models trained on native-script Assamese performed better than those trained on Romanized text. Overall, this research contributes to the development of robust transliteration models that can handle real-world noisy data and improve NLP applications in low-resource languages.

TH-3777_186155001

8.2 Limitations and Future Works

Despite the significant contributions of this thesis, several limitations highlight areas for future improvement. One key limitation is the restricted dataset scope, which, while diverse, can be expanded to include more sources beyond Facebook, Twitter, and YouTube. Collecting larger, more diverse datasets covering additional social media platforms and informal online interactions can improve the generalizability of transliteration models. Additionally, while this research focused on word-level and sentence-level transliteration, further exploration of phrase-level and context-aware transliteration could enhance accuracy, especially in code-mixed scenarios. Cross-lingual influences and phonetic variations remain a challenge, requiring models that can learn from multilingual and code-mixed data for better transliteration performance.

The detailed analysis of transliteration variations presented in Chapter 3 was not directly incorporated into the model design but served as an important linguistic investigation. This analysis systematically examined the non-canonical and noisy nature of Assamese social media text and motivated the development of the curated back-transliteration dataset. It also revealed several limitations of existing transliteration methods when processing user-generated content characterized by inconsistent spellings and mixed-script usage. While this thesis primarily focused on dataset construction and empirical evaluation, the findings from the transliteration variation study highlight the importance of modeling context and variability in real-world text. Building on these insights, future research may explore integrating such linguistic variation analysis into model architectures to develop more robust, context-aware transliteration systems.

Another important direction is the development of more effective multilingual transliteration models. This thesis demonstrated that models trained on both canonical and non-canonical data outperform those trained on either type alone. However, leveraging advanced multilingual frameworks and cross-lingual transfer learning could further improve transliteration accuracy. Beyond transliteration, this research established that reverse transliteration benefits downstream NLP tasks. Future work can explore its impact on machine translation, named entity recognition, and information retrieval, as well as experiment with alternative pre-trained models to assess their effectiveness in handling Romanized and native-script text together. Ultimately, this research lays the groundwork for context-aware, domain-adaptive, and scalable transliteration models, contributing to the broader field of low-resource language processing in real-world applications.

TH-3777_186155001

Bibliography

[1] G. Vyshnavi, G. Sridevi, P. K. Reddy, and M. Ritesh, "Syllabified sequence-to-sequence attention for transliteration," in *Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation*, 2022, pp. 617–625.

[2] K. Knight and J. Graehl, "Machine transliteration," *Computational Linguistics*, vol. 24, no. 4, pp. 599–612, 1998.

[3] C. Chandramouli and R. General, "Census of india," *Rural Urban Distribution of Population, Provisional Population Total. New Delhi: Office of the Registrar General and Census Commissioner, India*, 2011.

[4] S. Mahanta, "Assamese," *Journal of the International Phonetic Association*, vol. 42, no. 2, p. 217–224, 2012.

[5] A. L. Bizzocchi, "How many phonemes does the english language have?" *International Journal on Studies in English Language and Literature (IJSELL)*, vol. 5, no. 10, pp. 36–46, 2017.

[6] B. Roark, L. Wolf-Sonkin, C. Kirov, S. J. Mielke, C. Johny, I. Demirşahin, and K. Hall, "Processing South Asian languages written in the Latin script: the Dakshina dataset," in *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*, 2020, pp. 2413–2423.

[7] A. Kunchukuttan, S. Jain, and R. Kejriwal, "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*, 2021, pp. 3469–3475.

[8] Y. Madhani, S. Parthan, P. Bedekar, G. Nc, R. Khapra, A. Kunchukuttan, P. Kumar, and M. M. Khapra, "Aksharantar: Open indic-language transliteration datasets and models for the next billion users," in *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, Dec. 2023, pp. 40–57.

TH-3777_186155001

[9] H. Baruah, S. R. Singh, and P. Sarmah, "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)*, 2024, pp. 1627–1637.

[10] A. Irvine, J. Weese, and C. Callison-Burch, "Processing informal, romanized pakistani text messages," in *Proceedings of the Second Workshop on Language in Social Media*, 2012, pp. 75–78.

[11] M. Al-Badrashiny, R. Eskander, N. Habash, and O. Rambow, "Automatic transliteration of romanized dialectal arabic," in *Proceedings of the eighteenth conference on computational natural language learning*, 2014, pp. 30–38.

[12] Z. Byambadorj, R. Nishimura, A. Ayush, and N. Kitaoka, "Normalization of transliterated mongolian words using seq2seq model with limited data," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 20, no. 6, aug 2021.

[13] A. Khan and A. Sarfaraz, "Rnn-lstm-gru based language transformation," *Soft Computing*, vol. 23, 08 2019.

[14] J. Younes, E. Souissi, H. Achour, and A. Ferchichi, "A sequence-to-sequence based approach for the double transliteration of tunisian dialect," *Procedia computer science*, vol. 142, pp. 238–245, 2018.

[15] Y. Merhav and S. Ash, "Design challenges in named entity transliteration," in *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Aug. 2018, pp. 630–640.

[16] A. Kunchukuttan, R. Puduppully, and P. Bhattacharyya, "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*. Association for Computational Linguistics, Jun. 2015, pp. 81–85.

[17] S. Ruder, J. H. Clark, A. Gutkin, M. Kale, M. Ma, M. Nicosia, S. Rijhwani, P. Riley, J.-M. A. Sarr, X. Wang, J. Wieting, N. Gupta, A. Katanova, C. Kirov, D. L. Dickinson, B. Roark, B. Samanta, C. Tao, D. I. Adelani, V. Axelrod, I. Caswell, C. Cherry, D. Garrette, R. Ingle, M. Johnson, D. Panteleev, and P. Talukdar, "XTREME-UP: A user-centric scarce-data benchmark for under-represented languages," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Association for Computational Linguistics, Dec. 2023, pp. 1856–1884.

TH-3777_186155001

[18] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel, "Byt5: Towards a token-free future with pre-trained byte-to-byte models," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 291–306, 2022.

[19] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "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*. Association for Computational Linguistics, Jun. 2021, pp. 483–498.

[20] B. G. Stalls and K. Knight, "Translating names and technical terms in arabic text," in *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, ser. Semitic '98. Association for Computational Linguistics, 1998, p. 34–41.

[21] S. Wan and C. M. Verspoor, "Automatic english-chinese name transliteration for development of multilingual resources," in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*. Association for Computational Linguistics, 1998, p. 1352–1356.

[22] S. Karimi, F. Scholer, and A. Turpin, "Machine transliteration survey," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, pp. 1–46, 2011.

[23] K. Kaur, "Machine transliteration: A review of literature," *International Journal of Engineering Trends and Technology (IJETT) _*, vol. 37, 2016.

[24] D. K. Prabhakar and S. Pal, "Machine transliteration and transliterated text retrieval: a survey," *Sādhanā*, vol. 43, no. 6, p. 93, 2018.

[25] W.-H. Lin and H.-H. Chen, "Backward machine transliteration by learning phonetic similarity," in *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002.

[26] P. Virga and S. Khudanpur, "Transliteration of proper names in cross-language applications," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, 2003, pp. 365–366, *(a)*.

[27] ——, "Transliteration of proper names in cross-lingual information retrieval," in *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition*, 2003, pp. 57–64, *(b)*.

TH-3777_186155001

[28] W. Gao, K.-F. Wong, and W. Lam, "Phoneme-based transliteration of foreign names for oov problem," in *International Conference on Natural Language Processing*. Springer, 2004, pp. 110–119.

[29] K. S. Jeong, S.-H. Myaeng, J. S. Lee, and K. Choi, "Automatic identification and back-transliteration of foreign words for information retrieval," *Inf. Process. Manag.*, vol. 35, pp. 523–540, 1999.

[30] S. Y. Jung, S. Hong, and E. Paek, "An english to korean transliteration model of extended markov window," in *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics, 2000, p. 383–389.

[31] J.-H. Oh and K.-S. Choi, "An english-korean transliteration model using pronunciation and contextual rules," in *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[32] S. Ravi and K. Knight, "Learning phoneme mappings for transliteration without parallel data," in *Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics*, 2009, pp. 37–45.

[33] N. AbdulJaleel and L. S. Larkey, "Statistical transliteration for english-arabic cross language information retrieval," in *Proceedings of the twelfth international conference on Information and knowledge management*, 2003, pp. 139–146.

[34] T. Sherif and G. Kondrak, "Substring-based transliteration," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007, pp. 944–951.

[35] G. Li, M. Zhang, and J. Su, "A joint source-channel model for machine transliteration," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, 2004, pp. 159–166.

[36] I. KANG, "English-to-korean transliteration using multiple unbounded overlapping phoneme chunks," in *Proc. 18th International Conf. on Computational Linguistics*, vol. 1, 2000, pp. 418–424.

[37] B. KANG, "Automatic transliteration and back-transliteration by decision tree learning," in *Proc. 2nd International Conference on Language Resources & Evaluation*, 2000.

TH-3777_186155001

[38] K. Lindén, "Multilingual modeling of cross-lingual spelling variants," *Information Retrieval*, vol. 9, no. 3, pp. 295–310, 2006.

[39] H. Li, K. C. Sim, J.-S. Kuo, and M. Dong, "Semantic transliteration of personal names," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007, pp. 120–127.

[40] S. Karimi, F. Scholer, and A. Turpin, "Collapsed consonant and vowel models: New approaches for english-persian transliteration and back-transliteration," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007, pp. 648–655.

[41] Y. Al-Onaizan and K. Knight, "Machine transliteration of names in arabic texts," in *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, 2002a.

[42] J.-H. Oh and K.-S. Choi, "Machine learning based english-to-korean transliteration using grapheme and phoneme information," *IEICE transactions on information and systems*, vol. 88, no. 7, pp. 1737–1748, 2005.

[43] ——, "An ensemble of transliteration models for information retrieval," *Information processing & management*, vol. 42, no. 4, pp. 980–1002, 2006.

[44] J.-H. Oh and H. Isahara, "Machine transliteration using multiple transliteration engines and hypothesis re-ranking," *Proceedings of MT Summit XI*, pp. 353–360, 2007.

[45] S. Karimi, "Machine transliteration of proper names between english and persian," 2008.

[46] M. Zhang, H. Li, R. E. Banchs, and A. Kumaran, "Whitepaper of news 2015 shared task on machine transliteration," in *Proceedings of the Fifth Named Entity Workshop*, 2015, pp. 1–9.

[47] X. Duan, M. Zhang, H. Li, R. E. Banchs, and A. Kumaran, "Whitepaper of news 2016 shared task on machine transliteration," in *Proceedings of the Sixth Named Entity Workshop*, 2016, pp. 49–57.

[48] N. Chen, R. E. Banchs, M. Zhang, X. Duan, and H. Li, "Report of news 2018 named entity transliteration shared task," in *Proceedings of the seventh named entities workshop*, 2018, pp. 55–73.

TH-3777_186155001

[49] R. S. Roy, M. Choudhury, P. Majumder, and K. Agarwal, "Overview of the fire 2013 track on transliterated search," in *Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation*, 2013, pp. 1–7.

[50] Y.-C. Wang, C.-K. Wu, and R. T.-H. Tsai, "Ncu iisr english-korean and english-chinese named entity transliteration using different grapheme segmentation approaches," in *Proceedings of the Fifth Named Entity Workshop*, 2015, pp. 83–87.

[51] G. Nicolai, B. Hauer, M. Salameh, A. St Arnaud, Y. Xu, L. Yao, and G. Kondrak, "Multiple system combination for transliteration," in *Proceedings of the Fifth Named Entity Workshop*, 2015, pp. 72–77.

[52] A. Das and B. Gambäck, "Code-mixing in social media text. the last language identification frontier?" *TAL*, vol. 54, pp. 41–64, 2013.

[53] A. Ekbal, S. K. Naskar, and S. Bandyopadhyay, "A modified joint source-channel model for transliteration," in *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 2006, pp. 191–198.

[54] M. G. A. Malik, "Punjabi machine transliteration," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 2006, pp. 1137–1144.

[55] M. G. A. Malik, C. Boitet, and P. Bhattacharyya, "Hindi urdu machine transliteration using finite-state transducers," in *22nd International Conference on Computational Linguistics (COLING)*. ICCL, 2008, pp. 537–544.

[56] M. Vijaya, V. Ajith, G. Shivapratap, and K. Soman, "English to tamil transliteration using weka," *International Journal of Recent Trends in Engineering*, vol. 1, no. 1, p. 498, 2009.

[57] I. H. Witten, E. Frank, L. E. Trigg, M. A. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical machine learning tools and techniques with java implementations," 1999.

[58] M. K. Chinnakotla, O. P. Damani, and A. Satoskar, "Transliteration for resource-scarce languages," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 9, no. 4, pp. 1–30, 2010.

[59] A. Kumaran, M. M. Khapra, and P. Bhattacharyya, "Compositional machine transliteration," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 9, no. 4, pp. 1–29, 2010.

TH-3777_186155001

[60] A. P.J., A. V.P., and S. K.P., "Kernel method for english to kannada transliteration," in *Proceedings of the 2010 International Conference on Recent Trends in Information, Telecommunication and Computing*. IEEE Computer Society, 2010, p. 336–338.

[61] G. S. Josan and G. S. Lehal, "A punjabi to hindi machine translation system," in *Coling 2008: Companion volume: Demonstrations*, 2008, pp. 157–160.

[62] G. S. Josan and J. Kaur, "Punjabi to hindi statistical machine transliteration," *International Journal of Information Technology and Knowledge Management*, vol. 4, no. 2, pp. 459–463, 2011.

[63] M. Dhore, S. Dixit, and R. Dhore, "Optimizing transliteration for hindi/marathi to english using only two weights," in *Proceedings of the First International Workshop on Optimization Techniques for Human Language Technology*, 2012, pp. 31–48.

[64] T. D. Singh, "Bidirectional bengali script and meetei mayek transliteration of web based manipuri news corpus," in *Proceedings of the 3rd Workshop on South and Southeast Asian Natural Language Processing*, 2012, pp. 181–190.

[65] T. Dasgupta, M. Sinha, and A. Basu, "A joint source channel model for the english to bengali back transliteration," in *Proceedings of the First International Conference on Mining Intelligence and Knowledge Exploration - Volume 8284*. Springer-Verlag, 2013, p. 751–760.

[66] P. Pakray and P. Bhaskar, "Transliterated search system for indian languages," in *Pre-proceedings of the 5th FIRE-2013 workshop, forum for information retrieval evaluation (FIRE)*, 2013.

[67] H. Joshi, A. Bhatt, and H. Patel, "Transliterated search using syllabification approach," in *Forum for information retrieval evaluation*, 2013.

[68] I. A. Bhat, V. Mujadia, A. Tammewar, R. A. Bhat, and M. Shrivastava, "Iiit-h system submission for fire2014 shared task on transliterated search," in *Proceedings of the Forum for Information Retrieval Evaluation*, 2014, pp. 48–53.

[69] A. Mukherjee, A. Ravi, and K. Datta, "Mixed-script query labelling using supervised learning and ad hoc retrieval using sub word indexing," in *Proceedings of the Forum for Information Retrieval Evaluation*, 2014, pp. 86–90.

[70] D. Ganguly, S. Pal, and G. J. Jones, "Dcu@ fire-2014: fuzzy queries with rule-based normalization for mixed script information retrieval," in *Proceedings of the Forum for Information Retrieval Evaluation*, 2014, pp. 80–85.

TH-3777_186155001

[71] S. Patel and V. Desai, "Liga and syllabification approach for language identification and back transliteration: A shared task report by da-iict," 01 2015, pp. 43–47.

[72] D. K. Gupta, S. Kumar, and A. Ekbal, "Machine learning approach for language identification transliteration," in *Proceedings of the Forum for Information Retrieval Evaluation*, ser. FIRE '14. Association for Computing Machinery, 2014, p. 60–64.

[73] A. Kunchukuttan and P. Bhattacharyya, "Data representation methods and use of mined corpora for indian language transliteration," in *Proceedings of the Fifth Named Entity Workshop*, 2015, pp. 78–82.

[74] M. Al-Badrashiny, R. Eskander, N. Habash, and O. Rambow, "Automatic transliteration of romanized dialectal arabic," in *CoNLL 2014 - 18th Conference on Computational Natural Language Learning, Proceedings*, ser. CoNLL 2014 - 18th Conference on Computational Natural Language Learning, Proceedings. Association for Computational Linguistics (ACL), Jan. 2014, pp. 30–38, 18th Conference on Computational Natural Language Learning, CoNLL 2014 ; Conference date: 26-06-2014 Through 27-06-2014.

[75] J. May, Y. Benjira, and A. Echihabi, "An arabizi-english social media statistical machine translation system," in *Proceedings of the Eleventh Biennial Conference of the Association for Machine Translation in the Americas*, Association for Machine Translation in the Americas. Vancouver, Canada: Association for Machine Translation in the Americas, October 2014.

[76] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[77] T. Deselaers, S. Hasan, O. Bender, and H. Ney, "A deep learning approach to machine transliteration," in *Proceedings of the Fourth Workshop on Statistical Machine Translation*, 2009, pp. 233–241.

[78] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[79] P. Sanjanaashree *et al.*, "Joint layer based deep learning framework for bilingual machine transliteration," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2014, pp. 1737–1743.

[80] A. Finch, L. Liu, X. Wang, and E. Sumita, "Neural network transduction models in transliteration generation," in *Proceedings of the fifth named entity workshop*, 2015, pp. 61–66.

TH-3777_186155001

[81] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4225–4229.

[82] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[83] Y. Shao and J. Nivre, "Applying neural networks to english-chinese named entity transliteration," in *Proceedings of the sixth named entity workshop*, 2016, pp. 73–77.

[84] M. Rosca and T. Breuel, "Sequence-to-sequence neural network models for transliteration," *arXiv preprint arXiv:1610.09565*, 2016.

[85] A. Finch, L. Liu, X. Wang, and E. Sumita, "Target-bidirectional neural models for machine transliteration," in *Proceedings of the sixth named entity workshop*, 2016, pp. 78–82.

[86] M. S. H. Ameur, F. Meziane, and A. Guessoum, "Arabic machine transliteration using an attention-based encoder-decoder model," *Procedia Computer Science*, vol. 117, pp. 287–297, 2017.

[87] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[88] M. Abbas and F. Sadat, "Transliteration of algerian arabic dialect into modern standard arabic," 12 2017.

[89] J. Younes, E. Souissi, H. Achour, and A. Ferchichi, "A sequence-to-sequence based approach for the double transliteration of tunisian dialect," *Procedia Computer Science*, vol. 142, pp. 238 – 245, 2018, arabic Computational Linguistics.

[90] S. Kundu, S. Paul, and S. Pal, "A deep learning based approach to transliteration," in *Proceedings of the seventh named entities workshop*, 2018, pp. 79–83.

[91] Y. Merhav and S. Ash, "Design challenges in named entity transliteration," in *Proceedings of the 27th international conference on computational linguistics*, 2018, pp. 630–640.

[92] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

TH-3777_186155001

[93] A. Joshi, K. Mehta, N. Gupta, and V. K. Valloli, "Indian language transliteration using deep learning," in *2018 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*. IEEE, 2018, pp. 103–107.

[94] N. T. Le and F. Sadat, "Low-resource machine transliteration using recurrent neural networks of asian languages," in *Proceedings of the Seventh Named Entities Workshop*, 2018, pp. 95–100.

[95] R. Grundkiewicz and K. Heafield, "Neural machine translation techniques for named entity transliteration," in *Proceedings of the Seventh Named Entities Workshop*, 2018, pp. 89–94.

[96] L. Laitonjam, L. G. Singh, and S. R. Singh, "Transliteration of english loanwords and named-entities to manipuri: Phoneme vs grapheme representation," in *2018 International Conference on Asian Language Processing (IALP)*. IEEE, 2018, pp. 255–260.

[97] O. Terner, K. Bar, and N. Dershowitz, "Transliteration of judeo-arabic texts into arabic script using recurrent neural networks," in *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, 2020, pp. 85–96.

[98] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[99] A. Kunchukuttan, M. Khapra, G. Singh, and P. Bhattacharyya, "Leveraging orthographic similarity for multilingual neural transliteration," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 303–316, 2018.

[100] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. F. Aji, N. Bogoychev *et al.*, "Marian: Fast neural machine translation in c++," in *ACL (4)*, 2018.

[101] S. Singhania, M. Nguyen, G. H. Ngo, and N. Chen, "Statistical machine transliteration baselines for news 2018," in *Proceedings of the Seventh Named Entities Workshop*, 2018, pp. 74–78.

[102] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech communication*, vol. 50, no. 5, pp. 434–451, 2008.

[103] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and

TH-3777_186155001

E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Association for Computational Linguistics, Jun. 2007, pp. 177–180.

[104] I. A. Bhat, V. Mujadia, A. Tammewar, R. A. Bhat, and M. Shrivastava, "Iiit-h system submission for fire2014 shared task on transliterated search," in *Proceedings of the 6th Annual Meeting of the Forum for Information Retrieval Evaluation*, ser. FIRE '14. Association for Computing Machinery, 2014, p. 48–53.

[105] A. Kunchukuttan, S. Jain, and R. Kejriwal, "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*. Association for Computational Linguistics, Apr. 2021, pp. 3469–3475.

[106] S. Ruder, J. H. Clark, A. Gutkin, M. Kale, M. Ma, M. Nicosia, S. Rijhwani, P. Riley, J.-M. Sarr, X. Wang *et al.*, "Xtreme-up: A user-centric scarce-data benchmark for under-represented languages," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 1856–1884.

[107] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, "Scaling instruction-finetuned language models," *Journal of Machine Learning Research*, vol. 25, no. 70, pp. 1–53, 2024.

[108] R. S. Roy, M. Choudhury, P. Majumder, and K. Agarwal, "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*, ser. FIRE '12 & '13. New York, NY, USA: Association for Computing Machinery, 2013.

[109] M. Choudhury, G. Chittaranjan, P. Gupta, and A. Das, "Overview of fire 2014 track on transliterated search," *in Proceedings of FIRE*, pp. 68–89, 2014.

[110] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet physics. Doklady*, vol. 10, pp. 707–710, 1965.

[111] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[112] M. Popović, "chrf: character n-gram f-score for automatic mt evaluation," in *Proceedings of the tenth workshop on statistical machine translation*, 2015, pp. 392–395.

TH-3777_186155001

[113] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational linguistics*, vol. 29, no. 1, pp. 19–51, 2003.

[114] K. Heafield, "Kenlm: Faster and smaller language model queries," in *Proceedings of the sixth workshop on statistical machine translation*, 2011, pp. 187–197.

[115] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation," in *Proceedings of ACL 2017, System Demonstrations*, 2017, pp. 67–72.

[116] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Association for Computational Linguistics, Jun. 2019, pp. 48–53.

[117] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[118] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[119] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[120] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[121] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[122] I. Bhat, R. A. Bhat, M. Shrivastava, and D. M. Sharma, "Universal dependency parsing for hindi-english code-switching," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 987–998.

[123] K. Chakma and A. Das, "Revisiting automatic transliteration problem for code-mixed romanized indian social media text," *Social-India*, vol. 2014, p. 42, 2014.

TH-3777_186155001

[124] L. V. Subramaniam, S. Roy, T. A. Faruquie, and S. Negi, "A survey of types of text noise and techniques to handle noisy text," in *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, ser. AND '09. Association for Computing Machinery, 2009, p. 115–122.

[125] E. Clark and K. Araki, "Text normalization in social media: Progress, problems and applications for a pre-processing system of casual english," *Procedia - Social and Behavioral Sciences*, vol. 27, pp. 2 – 11, 2011, computational Linguistics and Related Fields.

[126] Y. Hu, K. Talamadupula, and S. Kambhampati, "Dude, srsly?: The surprisingly formal nature of twitter's language," in *Seventh International AAAI Conference on Weblogs and Social Media*, 2013.

[127] A. Zelenkauskaite, "Abbreviate and insert? message length, addressee and non-standard writing in italian mobile texting and facebook," *Writing Systems Research*, vol. 9, no. 2, pp. 123–136, 2017.

[128] D. Abu Elhija, "A new writing system? developing orthographies for writing arabic dialects in electronic media," *Writing Systems Research*, vol. 6, no. 2, pp. 190–214, 2014.

[129] N. Kemp, "Texting versus txtng: reading and writing text messages, and links with other linguistic skills," *Writing Systems Research*, vol. 2, no. 1, pp. 53–71, 2010.

[130] M. Choudhury, R. Saraf, V. Jain, A. Mukherjee, S. Sarkar, and A. Basu, "Investigation and modeling of the structure of texting language," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 10, no. 3-4, pp. 157–174, 2007.

[131] D. Crystal, *Txtng: The gr8 db8*. OUP Oxford, 2008.

[132] S. Brody and N. Diakopoulos, "Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! using word lengthening to detect sentiment in microblogs," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 562–570.

[133] C. Thurlow and A. Brown, "Generation txt? the sociolinguistics of young people's text-messaging," *Discourse analysis online*, vol. 1, no. 1, p. 30, 2003.

[134] F. Liu, F. Weng, and X. Jiang, "A broad-coverage normalization system for social media language," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2012, pp. 1035–1044.

[135] K. Sapkal and U. Shrawankar, "Transliteration of secured sms to indian regional language," *Procedia Computer Science*, vol. 78, pp. 748–755, 2016.

TH-3777_186155001

[136] Z. Sharf and S. U. Rahman, "Lexical normalization of roman urdu text," *International Journal of Computer Science and Network Security*, vol. 17, no. 12, pp. 213–221, 2017.

[137] R. Singh, N. Choudhary, and M. Shrivastava, "Automatic normalization of word variations in code-mixed social media text," in *International Conference on Computational Linguistics and Intelligent Text Processing*. Springer, 2018, pp. 371–381.

[138] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[139] M. Lusetti, T. Ruzsics, A. Göhring, T. Samardžić, and E. Stark, "Encoder-decoder methods for text normalization." Association for Computational Linguistics, 2018.

[140] S. Mandal and K. Nanmaran, "Normalization of transliterated words in code-mixed data using seq2seq model & levenshtein distance," *W-NUT 2018*, p. 49, 2018.

[141] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[142] A. R. Khan, A. Karim, H. Sajjad, F. Kamiran, and J. Xu, "A clustering framework for lexical normalization of roman urdu," *Natural Language Engineering*, vol. 28, no. 1, pp. 93–123, 2022.

[143] S. Gupta, "Towards understanding code-mixed social media texts," Ph.D. dissertation, MA thesis, International Institute of Information Technology, India, 2016.

[144] T. D. Singh and T. Solorio, "Towards translating mixed-code comments from social media," in *Computational Linguistics and Intelligent Text Processing: 18th International Conference, CICLing 2017, Budapest, Hungary, April 17–23, 2017, Revised Selected Papers, Part II 18*. Springer, 2018, pp. 457–468.

[145] S. Thara and P. Poornachandran, "Code-mixing: A brief survey," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2018, pp. 2382–2388.

[146] S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black, "A survey of code-switched speech and language processing," *arXiv preprint arXiv:1904.00784*, 2019.

[147] A. Irvine, J. Weese, and C. Callison-Burch, "Processing informal, romanized pakistani text messages," in *Proceedings of the Second Workshop on Language in Social Media*, ser. LSM '12. USA: Association for Computational Linguistics, 2012, p. 75–78.

TH-3777_186155001

[148] K. Chakma and A. Das, "Revisiting automatic transliteration problem for code-mixed romanized indian social media text," 12 2014.

[149] N. Desai and M. Narvekar, "Normalization of noisy text data," *Procedia Computer Science*, vol. 45, pp. 127 – 132, 2015, international Conference on Advanced Computing Technologies and Applications (ICACTA).

[150] S. Dutta, T. Saha, S. Banerjee, and S. K. Naskar, "Text normalization in code-mixed social media text," in *2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS)*, 2015, pp. 378–382.

[151] A. Sharma, S. Gupta, R. Motlani, P. Bansal, M. Shrivastava, R. Mamidi, and D. M. Sharma, "Shallow parsing pipeline - Hindi-English code-mixed social media text," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Jun. 2016, pp. 1340–1345.

[152] I. Bhat, R. A. Bhat, M. Shrivastava, and D. Sharma, "Joining hands: Exploiting monolingual treebanks for parsing of code-mixing data," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Apr. 2017, pp. 324–330.

[153] K. Singh, I. Sen, and P. Kumaraguru, "Language identification and named entity recognition in Hinglish code mixed tweets," in *Proceedings of ACL 2018, Student Research Workshop*. Association for Computational Linguistics, Jul. 2018, pp. 52–58.

[154] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1412–1421.

[155] R. R. Riyadh and G. Kondrak, "Joint approach to deromanization of code-mixed texts," in *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*. Association for Computational Linguistics, Jun. 2019, pp. 26–34.

[156] A. Masmoudi, M. Khmekhem, M. Khrouf, and L. Belguith, "Transliteration of arabizi into arabic script for tunisian dialect," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 19, pp. 1–21, 11 2019.

[157] D. Patel and R. Parikh, "Language identification and translation of english and gujarati code-mixed data," 02 2020, pp. 1–4.

TH-3777_186155001

[158] D. Parikh and T. Solorio, "Normalization and back-transliteration for code-switched data," in *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, 2021, pp. 119–124.

[159] L. Wolf-Sonkin, V. Schogol, B. Roark, and M. Riley, "Latin script keyboards for south asian languages with finite-state normalization," in *Proceedings of the 14th International Conference on Finite-State Methods and Natural Language Processing*, 2019, pp. 108–117.

[160] S. Wan and K. Verspoor, "Automatic english-chinese name transliteration for development of multilingual resources," in *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, 1998, pp. 1352–1356.

[161] G. M. McCune and E. O. Reischauer, *The Romanization of the Korean language based upon its phonetic structure*. Korea Branch of the Royal Asiatic Society, 1939.

[162] P. Lavanya, P. Kishore, and G. T. Madhavi, "A simple approach for building transliteration editors for indian languages," *Journal of Zhejiang University-SCIENCE A*, vol. 6, no. 11, pp. 1354–1361, 2005.

[163] T. Hussain and K. Samudravijaya, "Comparison and usefulness of asr11 scheme over previous schemes for transliteration and label set purposes for indian languages," in *39th All India DLA conference, Punjabi University, Patiala*, 2011.

[164] F. J. Och, "Minimum error rate training in statistical machine translation," in *Proceedings of the 41st annual meeting of the Association for Computational Linguistics*, 2003, pp. 160–167.

[165] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[166] R. E. Banchs, M. Zhang, X. Duan, H. Li, and A. Kumaran, "Report of NEWS 2015 machine transliteration shared task," in *Proceedings of the Fifth Named Entity Workshop*. Association for Computational Linguistics, Jul. 2015, pp. 10–23.

[167] A. Mukherjee, A. Ravi, and K. Datta, "Mixed-script query labelling using supervised learning and ad hoc retrieval using sub word indexing," in *Proceedings of the 6th Annual Meeting of the Forum for Information Retrieval Evaluation*, ser. FIRE '14. Association for Computing Machinery, 2014, p. 86–90.

TH-3777_186155001

[168] D. Ganguly, S. Pal, and G. J. F. Jones, "Dcu@fire-2014: Fuzzy queries with rule-based normalization for mixed script information retrieval," in *Proceedings of the 6th Annual Meeting of the Forum for Information Retrieval Evaluation*. Association for Computing Machinery, 2014, p. 80–85.

[169] C. Kirov, C. Johny, A. Katanova, A. Gutkin, and B. Roark, "Context-aware transliteration of romanized south asian languages," *Computational Linguistics*, vol. 50, no. 2, pp. 475–534, 06 2024.

[170] D. Samuel and M. Straka, "ÚFAL at MultiLexNorm 2021: Improving multilingual lexical normalization by fine-tuning ByT5," in *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, W. Xu, A. Ritter, T. Baldwin, and A. Rahimi, Eds. Association for Computational Linguistics, Nov. 2021, pp. 483–492.

[171] O. Kuparinen, A. Miletić, and Y. Scherrer, "Dialect-to-standard normalization: A large-scale multilingual evaluation," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Association for Computational Linguistics, Dec. 2023, pp. 13 814–13 828.

[172] M. Fahim, F. Shifat, F. Haider, D. Barua, M. Sourove, M. Ishmam, and M. Bhuiyan, "Banglatlit: A benchmark dataset for back-transliteration of romanized bangla," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 14 656–14 672.

[173] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, E. Blanco and W. Lu, Eds. Association for Computational Linguistics, Nov. 2018, pp. 66–71.

[174] M. R. Costa-Jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard *et al.*, "No language left behind: Scaling human-centered machine translation," *arXiv preprint arXiv:2207.04672*, 2022.

[175] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer, "Multilingual denoising pre-training for neural machine translation," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 726–742, 2020.

TH-3777_186155001

[176] Y. Tang, C. Tran, X. Li, P.-J. Chen, N. Goyal, V. Chaudhary, J. Gu, and A. Fan, "Multilingual translation with extensible multilingual pretraining and finetuning," *arXiv preprint arXiv:2008.00401*, 2020.

[177] A. Fan, S. Bhosale, H. Schwenk, Z. Ma, A. El-Kishky, S. Goyal, M. Baines, O. Celebi, G. Wenzek, V. Chaudhary *et al.*, "Beyond english-centric multilingual machine translation," *Journal of Machine Learning Research*, vol. 22, no. 107, pp. 1–48, 2021.

[178] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[179] S. Doddapaneni, R. Aralikatte, G. Ramesh, S. Goyal, M. M. Khapra, A. Kunchukuttan, and P. Kumar, "Towards leaving no indic language behind: Building monolingual corpora, benchmark and models for indic languages," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 12 402–12 426.

[180] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[181] S. Kumar, R. Sanasam, and S. Nandi, "IndiSocialFT: Multilingual word representation for Indian languages in code-mixed environment," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Association for Computational Linguistics, Dec. 2023, pp. 3866–3871.

[182] R. M. Duwairi, M. Alfaqeh, M. Wardat, and A. Alrabadi, "Sentiment analysis for arabizi text," in *2016 7th International Conference on Information and Communication Systems (ICICS)*, 2016, pp. 127–132.

[183] I. Guellil, A. Adeel, F. Azouaou, F. Benali, A.-e. Hachani, and A. Hussain, "Arabizi sentiment analysis based on transliteration and automatic corpus annotation," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, A. Balahur, S. M. Mohammad, V. Hoste, and R. Klinger, Eds. Association for Computational Linguistics, Oct. 2018, pp. 335–341.

TH-3777_186155001

[184] R. Roychoudhury, S. Dey, M. Akhtar, A. Das, and S. Naskar, "A novel approach towards cross lingual sentiment analysis using transliteration and character embedding," in *Proceedings of the 19th International Conference on Natural Language Processing (ICON)*, M. S. Akhtar and T. Chakraborty, Eds. Association for Computational Linguistics, Dec. 2022, pp. 260–268.

[185] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

TH-3777_186155001

TH-3777_186155001

Publications

Conference:

1. **Hemanta Baruah**, Sanasam Ranbir Singh, and Priyankoo Sarmah. **Assamese-BackTranslit: 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*

2. **Hemanta Baruah**, Sanasam Ranbir Singh, and Priyankoo Sarmah. **Assamese Back Transliteration - An Empirical Study Over Canonical and Non-canonical Datasets.**. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation, pages 801–808, Hong Kong, China. Association for Computational Linguistics*

Journal:

1. **Hemanta Baruah**, Sanasam Ranbir Singh, and Priyankoo Sarmah. **Transliteration Characteristics in Romanized Assamese Language Social Media Text and Machine Transliteration** In *ACM Trans. Asian Low-Resour. Lang. Inf. Process (TALLIP)* 23, 2, Article 33 (February 2024), Pages 1 - 36, ://doi.org/10.1145/3639565

2. **Hemanta Baruah**, Sanasam Ranbir Singh, and Priyankoo Sarmah. **LexDeRom: Context-Aware Lexical Deromanization with Large Language Models, A Comparison between Sentence and Word-Level Transliteration in Code-Mixed Assamese Social Media Tex**. In *ACM Trans. Asian Low-Resour. Lang. Inf. Process (TALLIP) (Ready to submit)*

3. **Hemanta Baruah**, Sanasam Ranbir Singh, and Priyankoo Sarmah. **Evaluation and Dataset Creation for Downstream NLP Tasks: Sentiment Analysis on Romanized vs Back-Transliterated Native Assamese Social Media Text**. In *Lang Resources and Evaluation (LREV) (Ready to submit)*

TH-3777_186155001

Brief Biography of the Author

Hemanta Baruah was born on January 17, 1991, in Nort Guwahati, Assam, India. He earned Bachelor of Engineering (B.E.) degree in Computer Science and Engineering from Jorhat Engineering College, Jorhat, Assam, India in 2013. He completed his M.Tech at NIT, Silchar, Assam, India in May 2016. Afterward, he worked as a Junior Project Fellow (JPF) in the Department of Computer Science and Engineering at IIT Guwahati from August 2018 to July 2018. Hemanta later enrolled as a Ph.D. research scholar at the Center for Linguistic Science and Technology, Indian Institute of Technology (IIT) Guwahati, Assam, India under the joint supervision of Prof. Sanasam Ranbir Singh and Prof. Priyankoo Sarmah. His research interests include Machine Transliteration, Machine Translation, Natural Language Processing (NLP), Low-Resource Social Media Analysis, Sentiment Analysis, Machine Learning, and Deep Learning.

TH-3777_186155001