Multilingual Machine Translation for Code-Mixed Hinglish and Tanglish Sentences to English: Leveraging NMT for Informal Communication

Nithish Krishna Shreenevasan
Johns Hopkins University

Yashaskar Aryan
Johns Hopkins University

Maria Movsheva
Johns Hopkins University

1 Introduction

In multilingual societies like India, informal digital communication often involves *code-mixing*, where individuals blend words and phrases from multiple languages within a single sentence. A notable example of this phenomenon is *Hinglish*, a combination of Hindi and English, where Hindi words are written using the Roman script alongside English words. Similarly, *Tanglish*, a blend of Tamil and English, is widely used in everyday conversations, particularly in digital communication. These code-mixed forms are not only prevalent in social media posts, messaging apps, and online forums, but have also become a defining feature of digital communication in the region. However, traditional machine translation (MT) systems, which are primarily designed for formal language, struggle with such code-mixed text due to its informal nature and lack of consistent grammar or punctuation.

1.1 Problem Statement

This project aims to develop a *transliteration system* capable of accurately handling code-mixed *Hinglish* and *Tanglish* texts. The system first transliterates the mixed-language text into its native script (Hindi or Tamil), before translating it into formal English. The primary challenge lies in handling the complex mixture of languages, where words from different scripts are interspersed, often with no clear separation between languages. To address this, we utilize the *Google Transliterate model*, which is a well-established tool for transforming Roman-scripted Indian languages into their native scripts. Once the code-mixed input is transliterated into its native script, we will test out translation models to convert the native text into English, offering a robust solution for this multilingual challenge.

1.2 Example

To illustrate the task, consider the following code-mixed sentences in Hinglish and Tanglish:

- **Input (Hinglish):** "Kal main market jaunga to buy some groceries."
- **Desired Output (English):** "Tomorrow I will go to the market to buy some groceries."
- **Input (Tanglish):** "Naan inga irundhaal, shopla products paarththuduven."
- **Desired Output (English):** "If I am here, I will check the products in the shop."

In these examples, the sentences contain a mix of Hindi/English (Hinglish) and Tamil/English (Tanglish), making it challenging to translate directly using traditional MT models. The goal is to first transliterate the mixed-language input into its respective native script and then translate it into fluent English.

2 Importance of the Problem

The accurate transliteration and translation of *code-mixed Hinglish and Tanglish* is not just a linguistic challenge, but a critical task for several real-world applications. By addressing this problem, we can unlock a range of benefits:

2.1 Improved Communication in Digital Platforms

As code-mixed languages are becoming increasingly dominant in online communication, enhancing translation accuracy on platforms such as social media, messaging apps, and customer support systems is essential. By improving the transliteration and subsequent translation of Hinglish and Tanglish, we provide better communication tools for millions of people who commonly use these hybrid languages.

2.2 Wider Access to Information

In India and other multilingual societies, many digital users communicate in code-mixed languages, which can be a barrier for those who only speak formal languages like English. By effectively transliterating and translating these code-mixed sentences, we can open up access to information for a wider audience, making content more accessible to those who are not fluent in formal English but engage in code-mixing.

2.3 Advancing Language Technologies

The ability to process code-mixed languages has broader implications for advancing language processing technologies, such as sentiment analysis, chatbots, and language learning tools. By developing a robust system that handles both transliteration and translation of Hinglish and Tanglish, we contribute to the enhancement of these tools, making them better equipped to handle informal, hybrid text that mirrors real-world communication.

Through this approach, we aim to create a powerful system that not only enhances the quality of transliteration and translation but also provides a foundation for future applications in multilingual natural language processing. By focusing on code-mixed texts like Hinglish and Tanglish, we are building more inclusive, effective language technologies that are tailored to the linguistic realities of multilingual societies.

3 Related Work

3.1 Aksharantar: Open Indic-language Transliteration Datasets and Models

Aksharantar is the largest publicly available transliteration dataset for Indian languages, containing 26 million transliteration pairs for 21 languages. It introduces the IndicXlit model (Madhani et al., 2023), which improves accuracy by 15% on the Dakshina test set. This dataset is key for advancing transliteration for Indian languages using romanized inputs.

3.2 Corpus Creation for Sentiment Analysis in Code-Mixed Tamil-English Text

This work (Chakravarthi et al., 2020) introduces a gold-standard Tamil-English code-mixed sentiment-annotated corpus of 15,744 YouTube comments. The paper details corpus creation, polarity assignment, and inter-annotator agreement. It establishes benchmark sentiment analysis results using this dataset, addressing the lack of annotated data for low-resource languages like Tamil.

3.3 TAMLI: Shorthand Romanized Tamil to Tamil Reverse Transliteration

This work (Mudiyanselage and Sumanathilaka, 2024) proposes a reverse transliteration approach for Tamil, addressing the challenges of vowel omission in Romanized Tamil. The method, incorporating N-gram analysis and a rule-based model, achieves a character-level accuracy of 0.93, outperforming existing transliteration techniques, making it a significant advancement in Tamil to Roman script transliteration.

3.4 IIIT-H System Submission for FIRE2014 Shared Task

This paper (Bhat et al., 2014) addresses two tasks in the FIRE 2014 Shared Task on Transliterated Search: query word labeling and mixed-script ad hoc retrieval for Hindi song lyrics. The system employs letter-based language models for token-level language identification and uses edit distance-based query expansion for song retrieval in mixed scripts.

3.5 Hinglish to English Machine Translation Using Multilingual Transformers

This study (Agarwal et al., 2021) explores the use of multilingual transformers (mBART, mT5) for Hinglish-to-English machine translation. By leveraging large pretrained models, the approach shows a substantial improvement in BLEU scores from 15.3 to 29.5 on the PHINC dataset, demonstrating the effectiveness of transformers for code-mixed language translation.

4 Challenges

Translating code-mixed Hinglish and Tanglish presents several challenges, primarily due to the informal nature of these languages. These challenges include the following:

4.1 Informal Language Structure

Code-mixed texts often lack grammatical consistency, with sentence fragments, missing punctuation, and unconventional word order. Informal expressions, slang, and idioms further complicate translation. For example, Hinglish sentences like

”Mujhe pata tha ki ye ho jaayega” (I knew this would happen) are difficult to translate due to their informal structure and idiomatic usage.

4.2 Inconsistent Spelling

Another challenge is the inconsistent spelling of words in Roman script. Variants like ”jaunga,” ”jaoonga,” and ”jauunga” for the same word create difficulties in mapping to native scripts. This inconsistency makes it hard for transliteration systems to correctly identify and process these variations, complicating the training of models for accurate recognition.

4.3 Why Obvious Solutions Fail

4.3.1 Rule-Based Systems

Rule-based systems, which rely on predefined linguistic rules, fail to handle the variability and informal nature of code-mixed texts. They cannot account for slang, idiomatic phrases, or spelling variations, leading to poor performance.

4.3.2 Standard NMT Models

Standard neural machine translation models, trained on formal corpora, struggle to translate code-mixed inputs accurately. These models fail to generalize to informal, code-mixed texts, often producing inaccurate or incomplete translations.

In conclusion, the informal structure, spelling inconsistencies, and lack of resources make translating Hinglish and Tanglish particularly challenging. Traditional approaches like rule-based systems and standard NMT models do not perform well due to these issues.

5 Experiments

5.1 Dataset

For this project, two datasets were primarily used to test the performance of the models: the **Dakshina dataset** (Roark et al., 2020) and the **HINGE dataset**. The Dakshina dataset was utilized to evaluate the models on Hindi and Tamil transliteration tasks. The development (dev) set from the Dakshina dataset for both languages contained approximately 5000 sentences. From this, 100 sentences were randomly sampled for each evaluation run. This approach ensured a consistent and representative test set for model evaluation across different languages.

In addition to Dakshina, the **HINGE dataset** (Srivastava and Singh, 2021) was employed to test the performance of the best-performing model identified in earlier experiments. The HINGE dataset is specifically focused on Hindi transliteration and includes a diverse set of sentences. The dataset was used to evaluate the best-performing model identified in earlier experiments. The validation set of the HINGE dataset contains around 380 sentences, which were used for the evaluation of the model. The HINGE dataset also includes an average rating for each annotated translation. For the evaluation, only sentences with an average rating of 5 or higher were considered to ensure high-quality transliterations.

Both datasets were critical in analyzing the effectiveness and robustness of the transliteration models across different languages, using consistent sampling for each language's task.

5.2 Models Used for Transliteration

Three models were used to perform the transliteration tasks and evaluate their performance. Below is a brief description of each model:

5.2.1 Model 1: IndicXlit

IndicXlit (Madhani et al., 2023) is a transformer-based multilingual transliteration model with approximately 11 million parameters. It supports 21 Indic languages for both Roman to native script and native to Roman script conversions. This model was utilized for the transliteration tasks on Hindi and Tamil in our experiments.

5.2.2 Model 2: INDIC-TRANS

INDIC-TRANS (Bhat et al., 2014) is a model developed for the FIRE 2014 Shared Task on Transliterated Search, which includes two subtasks: Query Word Labeling and Mixed-script Ad hoc Retrieval for Hindi Song Lyrics. The model uses letter-based language models for token-level language identification and a structured perceptron model for back-transliteration. Additionally, it employs edit distance-based query expansion and language modeling for retrieval tasks.

5.2.3 Model 3: Google Transliterate

Google Transliterate is a popular tool for transliterating text between different languages. It was used to transliterate Hindi and Tamil text from Roman script to native script and vice versa in our experiments.

5.3 Results

The models were evaluated using three standard metrics in machine translation and transliteration: **BLEU**, **GLEU**, and **METEOR**. Each of these metrics assesses the quality of transliterated text by comparing it to reference translations.

5.3.1 Metrics

BLEU (Bilingual Evaluation Understudy Score) is a precision-based metric that compares n-grams of the machine-generated output to a reference translation. A higher score indicates better quality.

GLEU (General Language Understanding Evaluation) is similar to BLEU but has a different approach to penalizing errors. It emphasizes matching smaller units (like bigrams) rather than relying on exact matches.

METEOR (Metric for Evaluation of Translation with Explicit ORdering) incorporates synonyms and stemming along with precision and recall to provide a more holistic assessment of translation quality.

5.4 Performance Review

The following tables present the evaluation scores for the models tested on the Dakshina Hindi and Dakshina Tamil datasets. For each dataset, 100 sentences were randomly sampled from the development set to compute the scores.

Model	BLEU	GLEU	METEOR
INDIC-TRANS (Hindi)	0.332	0.445	0.619
Google Transliterator (Hindi)	**0.487**	**0.596**	**0.732**
AI4Bharat-IndicXlit (Hindi)	0.251	0.376	0.550

Table 1: Performance of Transliteration Models on the Dakshina Hindi Dev Set

From Tables 1 and 2, we can infer that the Google Transliterate model works best for both romanized Tamil and Hindi scripts. It is interesting to note that INDIC-TRANS performs poorly when it comes to Tamil indicating that the model may not be fine tuned or trained properly for other regional languages apart from Hindi. The paper did focus mainly on working with Hindi songs and text.
We now evaluate Google Transliterate model on the HINGE dataset and the test set of Dakshina dataset. The results can be found in Table 3.
We can see that the Google model works quite well with the Dakshina test sets. For the HINGE set, the sampled sentences also had a corresponding disagreement score ranging from 0 to 10. Evaluating the model on sentences with disagreement scores less than 3 could give better results.

Model	BLEU	GLEU	METEOR
INDIC-TRANS (Tamil)	6.868×10^{-80}	0.0362	0.0517
Google Transliterator (Tamil)	**0.3957**	**0.6149**	**0.7205**
AI4Bharat-IndicXlit (Tamil)	0.3032	0.5522	0.6722

Table 2: Performance of Transliteration Models on the Dakshina Tamil Dev Set

Model	BLEU	GLEU	METEOR
Google Transliterator (HINGE)	0.2957	0.3974	0.5928
Google Transliterator (Dakshina Hindi Test Set)	0.5083	0.6059	0.7250
Google Transliterator (Dakshina Tamil Test Set)	0.4249	0.5991	0.7268

Table 3: Performance of Transliteration Models on the HINGE Dataset

References

Vibhav Agarwal, Pooja Rao, and Dinesh Babu Jayagopi. 2021. Hinglish to english machine translation using multilingual transformers. In *Proceedings of the Student Research Workshop Associated with RANLP 2021*, pages 16–21.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2014. Iiit-h system submission for fire2014 shared task on transliterated search. FIRE '14, New York, NY, USA. Association for Computing Machinery.

Bharathi Raja Chakravarthi, Vigneshwaran Muralidaran, Ruba Priyadharshini, and John Philip McCrae. 2020. Corpus creation for sentiment analysis in code-mixed Tamil-English text. In Dorothee Beermann, Laurent Besacier, Sakriani Sakti, and Claudia Soria, editors, *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 202–210, Marseille, France, May. European Language Resources association.

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

Anuja Dilrukshi Herath Herath Mudiyanselage and TG Deshan K Sumanathilaka. 2024. Tam: Shorthand romanized tamil to tamil reverse transliteration using novel hybrid approach. *The International Journal on Advances in ICT for Emerging Regions*, 17(1).

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

Vivek Srivastava and Mayank Singh. 2021. Hinge: A dataset for generation and evaluation of code-mixed hinglish text. *arXiv preprint arXiv:2107.03760*.