Enhancing Machine Translation for Code-Mixed Language Text

Kishansinh Jitendrasinh Rathod
krathod@usc.edu

Nikhil Bola Kamath
nikhilbo@usc.edu

Pavle Medvidovic
medvidov@usc.edu

Steven Melgar
stmelgar@usc.edu

Tejas Jambhale
tjambhal@usc.edu

Xingyu Zhao
xzhao911@usc.edu

Abstract

Machine translation is a rapidly evolving field in Natural Language Processing (NLP). Machine translation has applications ranging from improving cross-lingual communication to assisting global business operations. In this project, we address the complex challenge of translating code-mixed language (Hinglish) text. We used existing mixed language corpora as as well as self curated ones as our data. We developed a robust system using BART that accurately translates mixed language text into English. This system also recognizes the nuances of language switching within a sentence. With bilingualism being a significant aspect of many team members' lives, this project has a personal and practical relevance that extends beyond the research realm. Moreover, this work serves as a proof of concept, paving the way for enhanced mixed-language translation in various applications.

1 Introduction

Multilingualism and code-mixing have become standard communication features in our increasingly interconnected world. Code-mixing refers to seamless integration of elements from multiple languages within a single sentence. "Spanglish," a common example, is a hybrid language used by bilingual speakers of English and Spanish. These instances pose a unique challenge for machine translation systems, requiring both translation and language identification within a sentence. Multiple works motivated this project. In their work, (Gautam et al., 2021) use mBART and fine-tune it to achieve Hinglish to English translation. (Huber et al., 2022) introduces an end-to-end speech translation model called LAST. LAST converts a code-mixed audio signal into a target language. (Pratapa et al., 2018) shows how embeddings can help improve syntactic and semantic code-mixed processing tasks. (Sitaram et al., 2019) discusses various computational approaches for solving problems involving code-switched text. Papers by (Dhar et al., 2018) and (Srivastava and Singh, 2020) explain methods for creating/annotating datasets, a crucial step toward training the model. (Samanta et al., 2019) is another novel work that aims at generating large volumes of language-tagged code-switched text.

The rising prevalence of code-mixed languages globally highlights a need for more research in this area. One of the largest challenges in this domain is the intricacy of code-switching within bilingual speech. The nuances are manifold. Not only do some words exist in both languages, but their usage and meaning often depend on the context in which they occur. This duality necessitates a deep dive into the tone and intended message of the communication. Another challenge is the computing power. The original paper on mBART used 256 Nvidia V100 GPUs (32GB) for 500K steps, with a training time of nearly 2 weeks.

In this project, we aim to leverage large language models along with improved contextual embeddings to achieve better code-mixed translations. We focus primarily on the translation of English-Hindi (also known as Hinglish) to English. Upon completing this project, we can extrapolate this idea to create an end-to-end translation model for audio instead of text. From an application point of view, we can eliminate the language barrier prevalent in multicultural conversations. This project would also serve as a foundational model for cross-linguistic translation.

The rest of the paper is organized as follows. First we discuss the methods used to prepare the dataset followed by the pre-processing and tokenization steps. We outline the details of our code-mixed text translation models. This includes the pre-training and fine-tuning setup and evaluating the effectiveness of our methods by comparing their performance. Finally we examine any potential is-

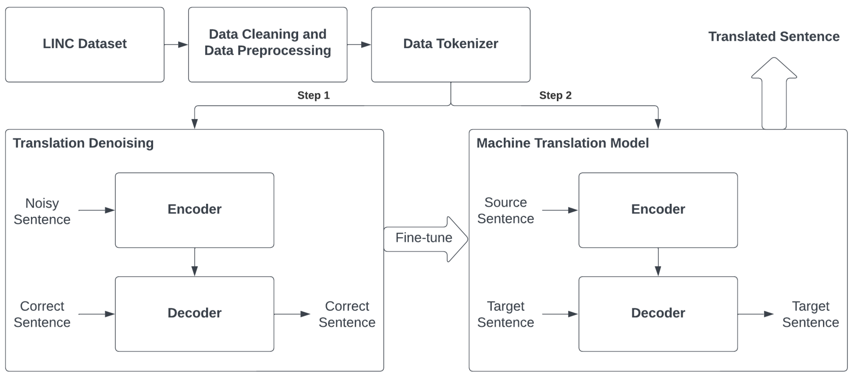

Figure 1: Code-Mixed Translation architecture.

sues with our methodology and conclude by providing guidance for future research while emphasizing our key findings.

2 Methodology

In this section, we describe our methodology for translating Hinglish to English using BART.

2.1 Data Acquisition

Data plays a vital role in driving these large model to give expected results. We know that transformers suffer from inductive bias. To address this problem of generalization, we curated a collection of data points from different sources, in addition to the existing dataset. We intend to make this available online soon. We primarily acquired the data used from three sources: the LinCE dataset from (Aguilar et al., 2020), the Hugging Face datasets from (Zhou et al., 2018), and the Hinglish-TOP dataset, a combination of synthetically generated (as explained in (Agarwal et al., 2022)) and 10,000 human annotated code-switched Hinglish sentences.

2.2 Data Preprocessing

We primarily derived our data from X, a platform defined by special characters and a colloquial linguistic style. We implemented a targeted data cleaning/preprocessing pipeline to handle this challenge. This included removing HTML, reference links, and retweet indicators, handling hashtags and usernames found on X and standardizing punctuation. Other common cleaning techniques like emoji removal, trimming redundant spaces, removing digits, and lowercase conversions were also part of the cleaning/preprocessing stage.

2.3 Data Tokenization

Before we passed our data to the model for translation, we had to break it down into meaningful units. Because we worked on code-mixed language, we had to derive a new set of tokens for this language. We employed Byte-Pair Encoding to achieve this. Our main goal was to break down the raw text into smaller, manageable tokens. These are often smaller than a word but larger than a character. We started by conducting a frequency analysis of tokens and building our initial vocabulary accordingly. Next, we computed Byte-Pairs by iterating through the vocabulary, with a goal of identifying pairs that frequently occur together. These were prime candidates for the subsequent merging process. Once we identified such pairs, they were merged and we updated our vocabulary set. This step was iterative and continued until we achieve the most efficient merging of byte pairs, optimizing the vocabulary for further processing. We finally bound our custom BPE with our Bart tokenizer to generate model-ready tokens. This output represented the optimized set of token pairs, which significantly enhanced the performance of the BART model in processing and understanding our data.

2.4 Model Training and Fine-tuning

The model training process was split into two steps as outlined in Figure 1. The first step was denoising and the second dealt with translation. Our approach leveraged the BART (Bidirectional and Auto-Regressive Transformers) model, renowned for its effectiveness in various natural language processing tasks.

The first stage began with the acquisition of the pre-trained BART model and the custom Bart tokenizer. Following this, we performed advanced text processing techniques on the transformed data, including masking/infilling and shuffling. These techniques enabled the model to develop an understanding of context and to generate predictions for missing tokens. Shuffling also helped in reducing bias and improving generalization.

We performed these techniques by using Huggingface data collators. "DataCollatorForWholeWordMask" masks whole words (group of tokens constitute a word), making the task more challenging and closer to actual language understanding. "DataCollatorForPermutationLanguageModeling" permutes the order of the input tokens, allowing the model to learn bidirectional context (i.e., both left and right context of each token). These augmented tokens were then fed as input to the BART model with the objective of generating the original sentence before augmentation. Due to resource constraints this task could be expanded upon in future work. The fine-tuning of the BART model using the transformed code-mixed text completed the first stage of training.

In the second stage, we built upon the foundation laid in the first stage by leveraging the facebook/bart-large model. We again employed the custom BART tokenizer, as input to the fine-tuned model from the previous step. We further tuned the BART model using the teacher forcing technique to achieve machine translation. This ensured that the model was finely tuned to the linguistic characteristics of these language pairs. An important feature of this stage was the model's capability for implicit language detection within a sentence or document at the word level. This was essential for accurate translation of code-mixed content. This step ensured that the model not only understood but also accurately translated the subtleties inherent in code-mixed languages. The final output of this stage was a model that could take code-mixed text as input and produce its equivalent in standard English, demonstrating its effectiveness in managing and translating complex code-mixed language scenarios. As a part of the inference step, we also performed a beam search operation to generate accurate translation for our Hinglish code-mixed text.

3 Experiment Settings

With the main objective of code-mixed translation, we hypothesized that we could create a model that translates a sentence with multi-lingual texts written in a given language into a desired language.

Table 1: Code-mixed data statistics.

	Train	Valid	Test
# of sentences	174 443	3243	7072
# of tokens in source sentences	1 764 738	38 502	70 895
# of tokens in target sentences	1 807 859	38 366	–

We proved this point by building an LLM that translates Hinglish to English. As mentioned in the data acquisition section, we used an assortment of data points from various sources. Table 1 gives an idea of the organization of our dataset. For our experiments, we defined the vocabulary size for our custom BPE tokenizer to be 50265. This is because the BART model implemented by the Hugging Face has a shared embedding between the encoder and the decoder. This forced us to have a vocabulary size same across the encoder and the decoder. We then bound the custom BPE with the Bart tokenizer to make tokens model-ready. Usage of various optimizers and schedulers had similar effects on training. In this work, we proceeded with the AdamW optimizer, with initial learning rate of 5e-5. Everything except the epsilon (1e-8) was set to default values. The scheduler also handled our learning rate throughout the training process. We followed a custom LR scheduler that allowed the learning rate to grow and decay as shown in the figure 2. Training the translation model took approximately twelve hours, with additional time for validation and testing.

We evaluated the model based on the SacreBLEU, CHRF and the BERTScore (Zhang et al., 2019). The BLEU (Bilingual Evaluation Understudy) Score is a metric that measures how closely the machine's translation aligns with one or more human translations. SacreBLEU Score is a standardized implementation of BLEU by Hugging Face that calculates the frequency of n-grams in translations that are also present in reference texts. This count is adjusted based on the length of the translations to avoid favoring overly terse or verbose translations.

Character n-gram F-score (chrF) is computed as an F-score, and it primarily evaluates the extent of character n-gram overlap between translations and reference texts. Additionally, BERTScore assesses the cosine similarity between BERT embeddings of individual tokens in translations and reference texts. Subsequently, it calculates precision, recall, and F1 scores across the dataset. BERTScore aligns

Table 2: Results: Metrics reported are SacreBLEU, ChrF, BERT-P: BERT Precision, BERT-R: BERT Recall, BERT-F1: BERT F1 score

	Our Model					GPT4				
Dataset	SacreBLEU	ChrF	BERT-P	BERT-R	BERT-F1	SacreBLEU	ChrF	BERT-P	BERT-R	BERT-F1
huggingface/cmu_hinglish_dog	15.087	33.217	0.904	0.898	0.901	40.411	58.106	0.929	0.928	0.928
hinglish-TOP-dataset	58.648	76.296	0.961	0.957	0.959	51.333	74.389	0.950	0.948	0.949
Overall dataset	**50.378**	**68.449**	**0.954**	**0.949**	**0.952**	**49.354**	**71.169**	**0.948**	**0.946**	**0.947**

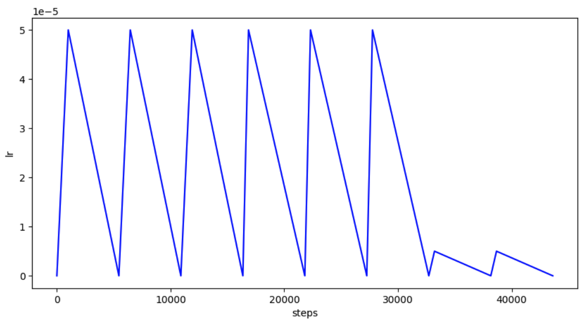

Figure 2: Custom LR Scheduler

more closely with human evaluations of text quality, proving particularly effective in scenarios where maintaining semantic accuracy and coherence is important.

Finally, we compared our results with the popular GPT-4 Turbo model (gpt-4-1106-preview) using the OpenAI API. We evaluated our test dataset against this API without any prior training/fine-tuning. We fed our test data sentence-wise without any tokenization using the prompt "Translate the following sentences from Hindi-English (Hinglish) to English".

4 Results

Table 2 shows that we were able to successfully translate any given code-mixed Hinglish (hi_en) text into English (en). This proves our hypothesis of creating one model for end-to-end translation without explicit language detection. The results as seen in Table 2, show positive results for our proposed model. We were able to achieve an overall SacreBleu score of 50.378 similar to the SacreBleu score by GPT4-Turbo of 49.354. A score above 60 is equivalent to human level translation. Moving on to the chrF score, GPT4-Turbo is slightly better in this metric. Finally, both models achieved a high BERT score, which is often considered a better metric than the BLEU score.

The following are the output of our model. All results and test data outputs can be viewed in our GitHub repository.

- ***hi_en**: kya mujhe forecast milsakta hai please*
  ***en**: can i have the forecast please*
- ***hi_en**: miami se west palm tak kitni der lagegi*
  ***en**: how long will it take from miami to west palm*
- ***hi_en**: kya aap kal ke liye mera alarm set kar sakte he*
  ***en**: can you set my alarm for tomorrow*

It is important to note that almost no literature in this domain, except on OpenAI's GPT family, has been able to replicate these levels of metrics on code-mixed data. In addition, our model achieved these results using limited compute resources in only twelve hours of training time. We believe that upon further training the model and with access to more data, we can surpass our current scores.

5 Conclusion

In this project, we proposed a unique methodology of translating code-mixed Hinglish text to English text using a Bart model. We were able to achieve an overall BLEU score of 50.378 and a BERTScore-F1 of 0.952. We then used the GPT4-Turbo model and compared the results with our implementation. We conclude that although both implementations perform similarly, our model is slightly better at BLEU and BERTScore evaluations while GPT4-Turbo leads the way in ChrF metric. By leveraging the strengths of the BART model and the BartTokenizer, we can address the challenges in processing and understanding code-mixed languages. For further work, we propose using a Speech to Speech model such as Wav2Vec to create an end to end audio Machine Translation system. Such an application would have many use cases in global business, politics, and education. Another potential research goal in this domain is to improve execution speed and reduce reliance on large compute resources for training the model and making inferences.

References

Anmol Agarwal, Jigar Gupta, Rahul Goel, Shyam Upadhyay, Pankaj Joshi, and Rengarajan Aravamudhan. 2022. Cst5: Data augmentation for code-switched semantic parsing. *arXiv preprint arXiv:2211.07514*.

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. LinCE: A Centralized Benchmark for Linguistic Code-switching Evaluation. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 1803–1813, Marseille, France. European Language Resources Association.

Mrinal Dhar, Vaibhav Kumar, and Manish Shrivastava. 2018. Enabling code-mixed translation: Parallel corpus creation and mt augmentation approach. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, pages 131–140.

Devansh Gautam, Prashant Kodali, Kshitij Gupta, Anmol Goel, Manish Shrivastava, and Ponnurangam Kumaraguru. 2021. Comet: Towards code-mixed translation using parallel monolingual sentences. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 47–55.

Christian Huber, Enes Yavuz Ugan, and Alexander Waibel. 2022. Code-switching without switching: Language agnostic end-to-end speech translation. *arXiv preprint arXiv:2210.01512*.

Adithya Pratapa, Monojit Choudhury, and Sunayana Sitaram. 2018. Word embeddings for code-mixed language processing. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3067–3072.

Bidisha Samanta, Sharmila Reddy, Hussain Jagirdar, Niloy Ganguly, and Soumen Chakrabarti. 2019. A deep generative model for code-switched text. *arXiv preprint arXiv:1906.08972*.

Sunayana Sitaram, Khyathi Raghavi Chandu, Sai Krishna Rallabandi, and Alan W Black. 2019. A survey of code-switched speech and language processing. *arXiv preprint arXiv:1904.00784*.

Vivek Srivastava and Mayank Singh. 2020. Phinc: A parallel hinglish social media code-mixed corpus for machine translation. *arXiv preprint arXiv:2004.09447*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Kangyan Zhou, Shrimai Prabhumoye, and Alan W Black. 2018. A dataset for document grounded conversations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Appendix

A Github Link

Link to our GitHub repository.

B Contributions

Nikhil was responsible for coding data accquisition scripts, proposed model implementation, custom BPE and Bart Tokenizer, Model training and inference, report documentation. Tejas mainly worked on implementing GPT4-Turbo for our comparative analysis and led report documentation. He also supported the team during the data acquisition phase and model training phase. Pavle contributed to the data acquisition and pre-processing phase. He was also responsible for designing custom BPE and BART Tokenizer. Pavle was also involved in report documentation. Steven, mainly led the data acquisition phase, followed by this he was also involved in writing scripts for data cleaning, pre-processing and custom BPE implementation. Xingyu and Kishan mainly worked on the scripts for augmenting texts for addressing the de-noising phase of training. Alongside, they also helped in report documentation.