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) with 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 text. We aim to develop a robust system that accurately translates mixed language text (Spanglish and Hinglish) into English, recognizing 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, potentially paving the way for enhanced mixed-language translation in various applications. This document is the status report.

1 Tasks that have been performed

The primary goal of the project's initial phase was to establish a complete end-to-end pipeline. This phase encompassed a series of critical tasks, including data acquisition, cleansing, processing, and the design and implementation of the language model. Figure 1 depicts the overall architecture adhered to, which we aim to enhance further. The following section provides an in-depth explanation of the expanded version of the proposed end-to-end architecture.

For this project, a comprehensive approach was essential in ensuring the quality and variety of our dataset. To that end, our team integrated data from LinCE, a well-known source known for its rich repository of linguistic code-switching content. Recognizing the need for more diverse data (for the next phase), we also undertook the meticulous task of manually curating datasets that provided more naturally occurring language interchanges, bridging the gap between automated and real-world data. To diversify further, we scoured the internet, extracting datasets that varied in their content — from the casual tone seen in typical Twitter posts to more structured and formal content.

Given the primary derivation of our datasets from Twitter, a platform defined by its concise and colloquial linguistic style, our team implemented a detailed data cleaning/preprocessing protocol, to mention a few of them: HTML tags and reference links removal, retweet indicator elimination, handling hashtags and mentions found across the Twitter space and punctuation standardization. Of course, other miscellaneous activities, such as username elimination, emojis/emoticons removal, trimming of redundant spaces, handling of digits, and lowercase conversions, were also part of the cleaning/preprocessing stage. Yet another problem we aimed to solve was the un-normalized form - for instance, common shorthand forms, such as 'u' being used for 'you' or '4' as 'for' had to be expanded.

Tokenization, an integral process in our workflow, involves breaking down each text input into individual components, aka tokens, setting the stage for more detailed processing in translation. With this, we focus on tackling unconventional spellings and shorthand nuances. Tokenization entails crafting a dedicated module that accurately discerns and retains the essence of code-switching evident within tweets and colloquial forms of textual communication. We achieve this using a custom tokenizer trained on the input Spanglish and Hinglish data and subsequently used to transform sentences/documents into designated tokens. Text Tokenizer libraries like SentencePiece are for this purpose, which uses BPE and unigram language model to create tokens.

Following this is model training. Motivated by Gautam et al. (2021), we use mBART (Liu et al., 2020), a seq-to-seq denoising auto-encoder pretrained on a large-scale corpus. mBART performs well on supervised and unsupervised machine translation both at the sentence and document level. We leverage this power of mBART to perform code-

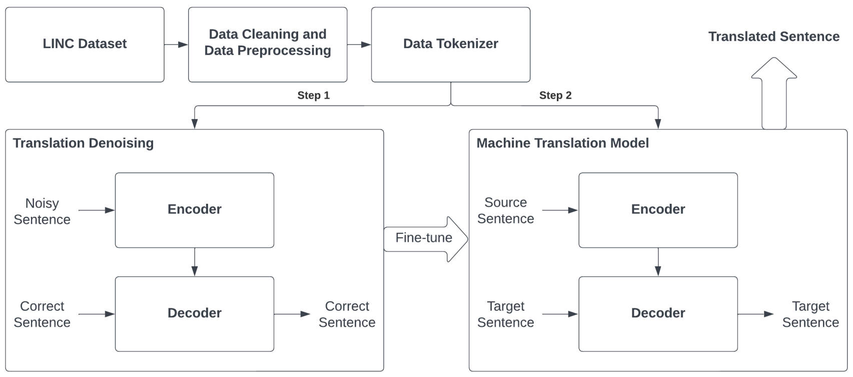

Figure 1: Code-Mixed Translation architecture.

mixed translation. The fine-tuning of mBART on the input Spanglish and Hinglish tokens takes place in two steps, as seen in Figure 1. The first step involves translation denoising, transforming the input space by adding noise. This is achieved by shuffling tokens, random token removals/masking, etc. This altered sentence goes into the encoder of the mBART model. The learned embeddings from the encoder, along with the expected input tokens sequence, are fed to the generative decoder of the mBART to generate the correct sentence. The objective is to ensure that the predicted correct sentence is similar to the true correct sentence. This mechanism captures sentence-level semantics and contextual information at a better scale. The second step is the machine translation phase. Here, again, we feed the Spanglish/Hinglish tokens to the fine-tuned mBART model from the previous step. We further fine-tune the mBART model using the teacher forcing technique to achieve machine translation. As of now, we were successfully able to create the tokens and feed them into the two-staged training phase. The forward-pass led to the generation of undesired translations due to the lack of training at this moment. As a part of the next phase, our team focuses on fine-tuning the model to fit our use case.

2 Risks and challenges that you think you need to address by the project deadline

One of the most formidable challenges our project might face 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 can heavily depend on the context in which they occur. This duality necessitates a deep dive into the tone and intended message of the communication. Our current setup for code-switching detection, while functional, is introductory. Another challenge is the computing power, as the original paper on mBART used 256 Nvidia V100 GPUs (32GB) for 500K steps, with a training time of nearly 2 weeks.

3 Your plan to mitigate the risks and address the challenges

We intend to mitigate the issues by improving the diversity of our dataset by gathering them from different sources available and fine-tuning the model for a longer duration. We first aim to leverage transfer learning for code-mixed translation by selectively freezing model layers to achieve the desired results. As we approach our project deadline, refining and enhancing our approach to code-switching remains a priority, ensuring our system comprehensively understands and respects the subtleties of bilingual communication.

4 Individual Contributions

Steven and Pavle were engaged in acquiring/curating data. Following this, they were also involved in cleaning/pre-processing stage. Xingyu and Kishansinh designed the custom dataset modules and data loaders supporting various transforms (augmentation) needed to train the model. Nikhil and Tejas worked on researching and implementing the custom data tokenizer, mBART model, and training pipeline in Pytorch. Finally, everyone equally participated in creating this project status report.

References

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.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pretraining for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.