Code-Mixed Machine Translation Project Final Report

IIIT Hyderabad
Team Name - The Triad : Yash Bhaskar, Sankalp Bahad, Utsav Shekhar
Course: ANLP (Advanced Natural Language Processing)

Abstract

This study explores the efficacy of various machine translation models on the task of translating English to Hinglish, a prevalent code-mixed language in multilingual societies. Focusing on the limitations posed by the informal and varied nature of code-mixed languages, we experiment with different architectures and strategies, including GRU and LSTM with attention mechanisms, sequence-to-sequence transformers, and few-shot learning on large language models (LLMs) such as Mistral 7B and mT5-small. Additionally, we implement pivot translation techniques utilizing Hindi in both Devanagari and Roman scripts to facilitate the translation process. Our comprehensive comparison, as detailed in the tables and figures within this paper, identifies models that surpass the BLEU scores reported in the "CoMeT: Towards Code-Mixed Translation Using Parallel Monolingual" paper. The results indicate that fine-tuning on high-capacity LLMs and Transformer-based models with attention mechanisms show significant promise in handling the intricacies of code-mixed translation. This paper contributes to the advancement of machine translation for low-resource and linguistically diverse language pairs by providing a nuanced understanding of model performance in the context of code-mixing.

1 Introduction

1.1 Problem Statement

Niche problem: Enhanced Code-Mixed Machine Translation

In the realm of natural language processing, the project addresses the complex problem of machine translation within the domain of code-mixed languages. Specifically, the project focuses on the translation from English to Hinglish (code mixed variant of language used by bilingual speakers of Hindi and English) , representing a significant niche in computational linguistics. The intrinsic complexity of code-mixing poses unique challenges for machine translation systems due to the syntactic and semantic intricacies involved. This project aims to delve into the development of an advanced machine translation model that can navigate the nuances of code-mixed language with greater accuracy and fluency than current state-of-the-art models, thereby setting a new benchmark in the field.

1.2 Scope of the Project

The scope of this project is ambitious, targeting the refinement of code-mixed machine translation technology. The foundational model, mBART, which serves as a benchmark with a BLEU score of 15.3, offers a baseline for enhancements. Our goal is to transcend this baseline by developing a machine translation system that is not only more proficient in interpreting and translating the subtleties of Hinglish but also in doing so with a higher degree of precision and naturalness as reflected in improved BLEU scores. This undertaking encompasses the exploration of novel methodologies, the augmentation of existing datasets, and the potential integration of linguistically informed models. It aims to substantially advance the capabilities of machine translation for code-mixed languages, leveraging the wealth of monolingual data to enrich the quality and extend the accessibility of translation services for Hinglish speakers worldwide. The culmination of this project aspires to achieve superior performance metrics, surpassing those reported in the "CoMeT: Towards Code-Mixed Translation Using Parallel Monolingual" paper, thereby contributing a significant leap forward in the field of multilingual computational linguistics.

1.3 Project Objectives

The objectives of this project, as outlined in the interim submission, are as follows:

1. **Dataset Selection:** Identify and select an appropriate dataset that accurately represents the challenges of translating Hinglish to English.

2. **Dataset Analysis:** Conduct a thorough analysis of the chosen dataset to gain insights into the characteristics, language patterns, and complexities present in code-mixed content.

3. **Model Definition:** Define an Encoder-Decoder Machine Translation (MT) Transformer model architecture that is tailored to the code-mixed translation task.

4. **Model Preparation:** Prepare the defined model for training and experimentation in the post-interim submission phase, ensuring all necessary components are in place.

5. **Model Selection:** Choose a pre-existing model as a starting point for the project, which will serve as the foundation for further development and fine-tuning.

6. **Initial Performance Assessment:** Evaluate the chosen pre-existing model's initial performance on the code-mixed translation task, establishing a baseline for future improvements.

7. **Fine-Tuning Plan:** Develop a comprehensive plan for fine-tuning the selected model, outlining the strategies and techniques that will be employed to enhance its translation capabilities.

Table 1: The statistics of the dataset. We use the language tags predicted by the CSNLI library. Since the target sentences of the test set are not public, we do not provide its statistics.

	Train	**Valid**	**Test**
# of sentences	8060	942	960
# of tokens in source sentences	98 080	12 275	12 557
# of tokens in target sentences	101 752	12 611	-
# of Hindi tokens in target sentences	68 054	8310	-
# of English tokens in target sentences	21 502	2767	-
# of 'Other' tokens in target sentences	12 196	1534	-

2 Background

Code-mixing occurs when a speaker switches between two or more languages in the context of the same conversation. It has become popular in multilingual societies with the rise of social media applications and messaging platforms.

In attempts to progress the field of code-mixed data, several code-switching workshops diab2014, diab2016, aguilar2018b have been organized in notable conferences. Most of the workshops include shared tasks on various of the language understanding tasks like language identification solorio2014, molina2016, NER aguilar2018a, rao2016, IR roy2013, banerjee2018, PoS tagging jamatia2016, sentiment analysis patra2018, patwa2020, and question answering chandu2018.

Although these workshops have gained traction, the field lacks standard datasets to build robust systems. The small size of the datasets is a major factor that limits the scope of code-mixed systems.

Machine Translation refers to the use of software to translate text from one language to another. In the current state of globalization, translation systems have widespread applications and are consequently an active area of research.

Neural machine translation has gained popularity only in the last decade, while earlier works focused on statistical or rule-based approaches. Kalchbrenner and Blunsom (2013) first proposed a DNN model for translation, following which transformer-based approaches vaswani2017 have taken the stage. Some approaches utilize multilingual pre-training song2019, conneau2019, edunov2019, liu2020; however, these works focus only on monolingual language pairs.

Although a large number of multilingual speakers in a highly populous country like India use English-Hindi code-mixed language, only a few studies srivastava2020, singh2018, dhar2018 have attempted the problem. Enabling translation systems in the following pair can bridge the communication gap between several people and further improve the state of globalization in the world.

3 Project Significance

The significance of this project lies in its potential to revolutionize code-mixed machine translation. By addressing the challenges specific to translating Hinglish to English, we aim to substantially improve the quality and accessibility of translation services in this domain. This advancement can have far-reaching implications, facilitating crosslingual communication, and opening doors to a broader audience.

4 Data Preparation

We use the dataset provided by the task organizers for our systems, and the statistics of the datasets are provided in Table 1. Since the target sentences in the dataset contain Hindi words in Roman script, We use the CSNLI library (Bhat et al., 2017, 2018) as a preprocessing step.

It transliterates the Hindi words to Devanagari and also performs text normalization. We use the provided train:validation:test split, which is in the ratio 8:1:1.

5 Exploratory Data Analysis

5.1 Dataset Overview

The dataset used is the human-annotated dataset consisting of 10000 sentences in the English language and their code-mixed translations to Hinglish language. You can access the dataset here.

5.1.1 Dataset Statistics

The dataset comprises 10000 human-annotated sentences in the English language and their translations into Hinglish.

5.2 Insights Gained

These histograms (Figure 1) provide insights into the variation in sentence lengths within the Hinglish and English text data.

This scatter plot (Figure 2) allows visualization of how sentence lengths in the two languages relate to each other.

These word clouds (Figure 3) visually represent the frequency of words in the Hinglish and English corpora.

1. Findings from the Exploratory Analysis

During the exploratory analysis, the following key findings were observed:

- Average code-mixing percentage: 35.35

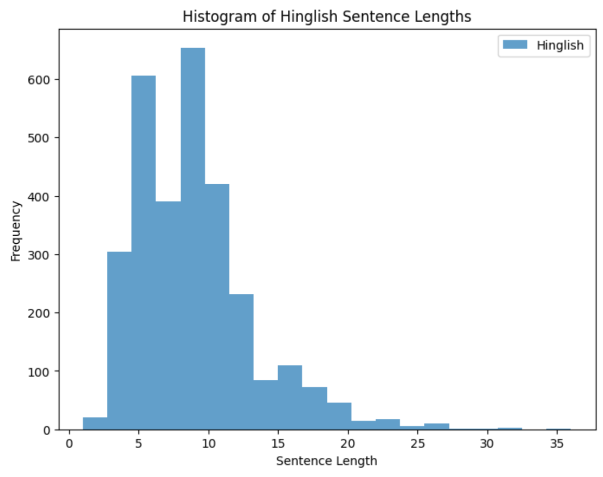
2. Implications for the Project

The exploratory analysis has important implications for the project:

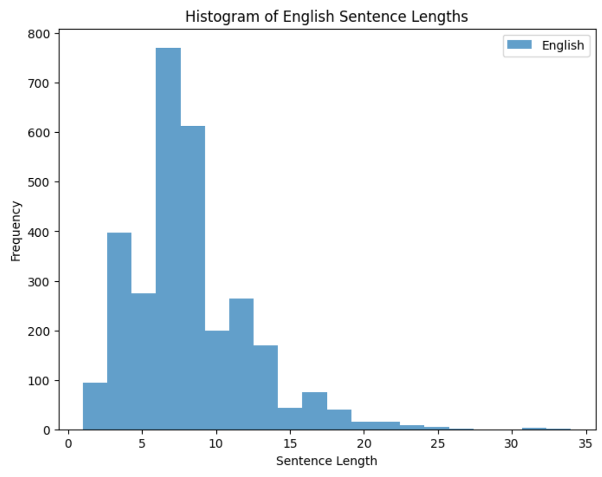
- Improved Accessibility: Converting English to Hinglish can make content more accessible to a wider audience in India, where Hinglish is commonly spoken and understood.

- Increased User Engagement: If your target audience primarily communicates in Hinglish, providing content in this language variant can lead to higher user engagement and better user experiences.

- Market Expansion: Adapting your products or services to the Hinglish-speaking population in India can open up new market opportunities and increase customer reach.

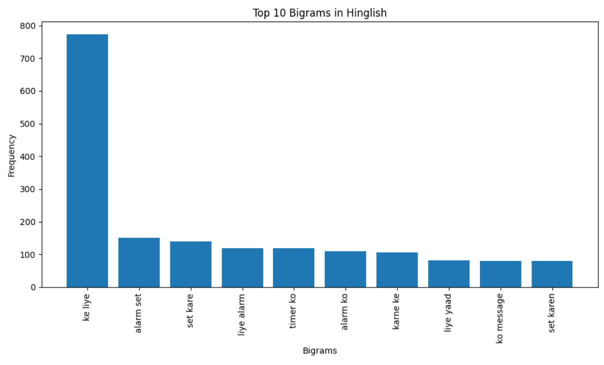


(a) Distribution of Sentence Lengths in Hinglish

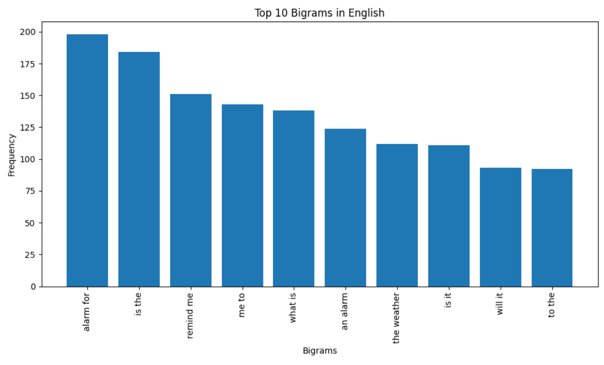


(b) Distribution of Sentence Lengths in English

Figure 1: Histograms displaying sentence length distributions in Hinglish and English texts.

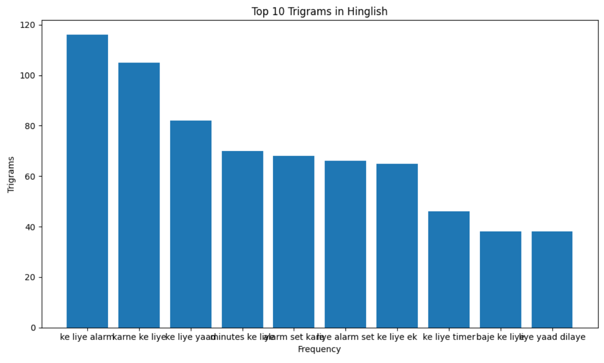


(a) Top 10 most frequent hinglish bigrams

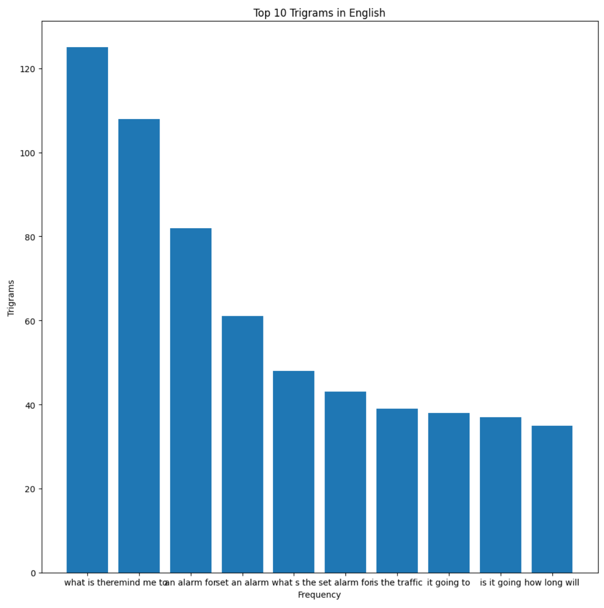


(b) Top 10 most frequent english bigrams

Figure 2: Bar graphs showing the frequencies of bigrams in english and hinglish.



(a) Top 10 most frequent hinglish trigrams



(b) Top 10 most frequent english trigrams

Figure 3: Bar graphs showing the frequencies of trigrams in english and hinglish.

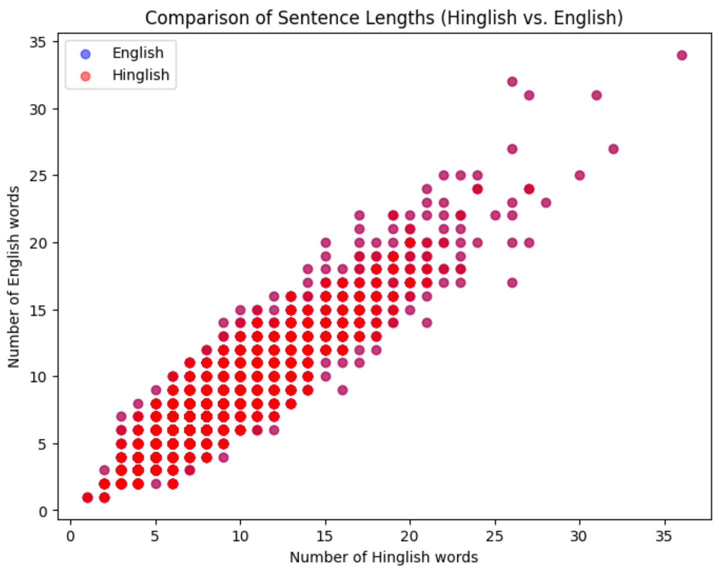


Figure 4: Scatter plot comparing sentence lengths in English and Hinglish.

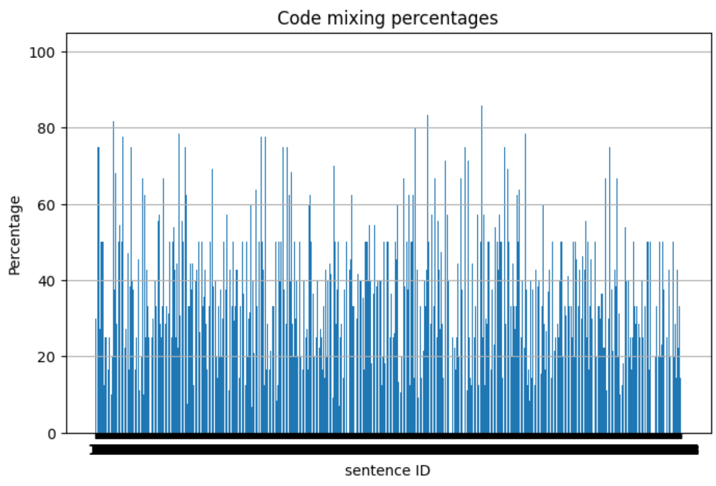
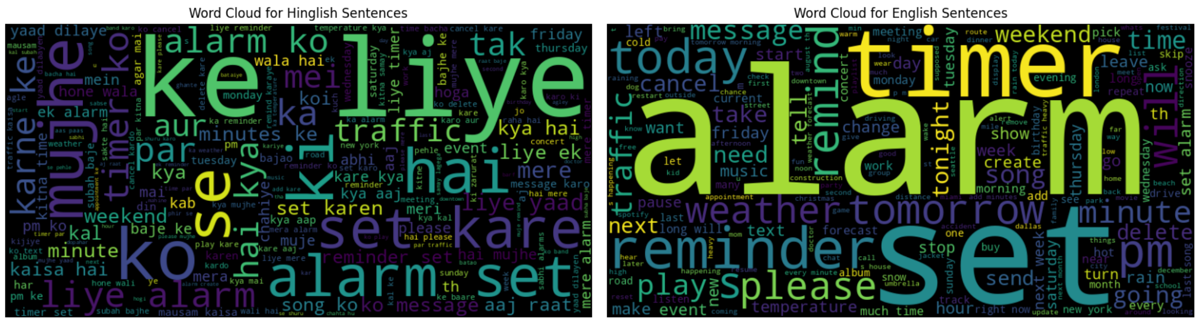


Figure 5: The Percentage of codemixing found in the given dataset in the hinglish sentences. Average code mixing percentage : 35.34908149380431



(a) Word Cloud for Hinglish (b) Word Cloud for English

Figure 6: Word clouds representing word frequency in Hinglish and English texts.

6 Comparison between Models

Table 2: Comparison of Machine Translation Models with Pivot Translations
M1: English → Hinglish,
M2: English → Hindi (Pivot Language) (Devanagari) → Hinglish,
M3: English → Hindi (Pivot Language) (Roman) → Hinglish.

Model	M1	M2	M3	Paper Score
GRU with Attention Mechanism	3.5	6.4	5.37	18.9
LSTM Sequence to Sequence	2.1	4.8	4.3	18.9
Seq to Seq Transformers with Attention	8.8	14.16	12.62	18.9
Few Shot promoting on Mistral 7B LLM	6.25	13.17	6.28	18.9
Fine tune Mistral 7B llm for MT	15.2	23.73	17.5	18.9
Fine tune mT5-small	10.63	13.42	10.49	18.9
Indic Bard	15.72	20.5	13.67	18.9
Fine tune Opus Model	6.8	0.26	1.2	18.9
Seamless	11.46	—	—	18.9

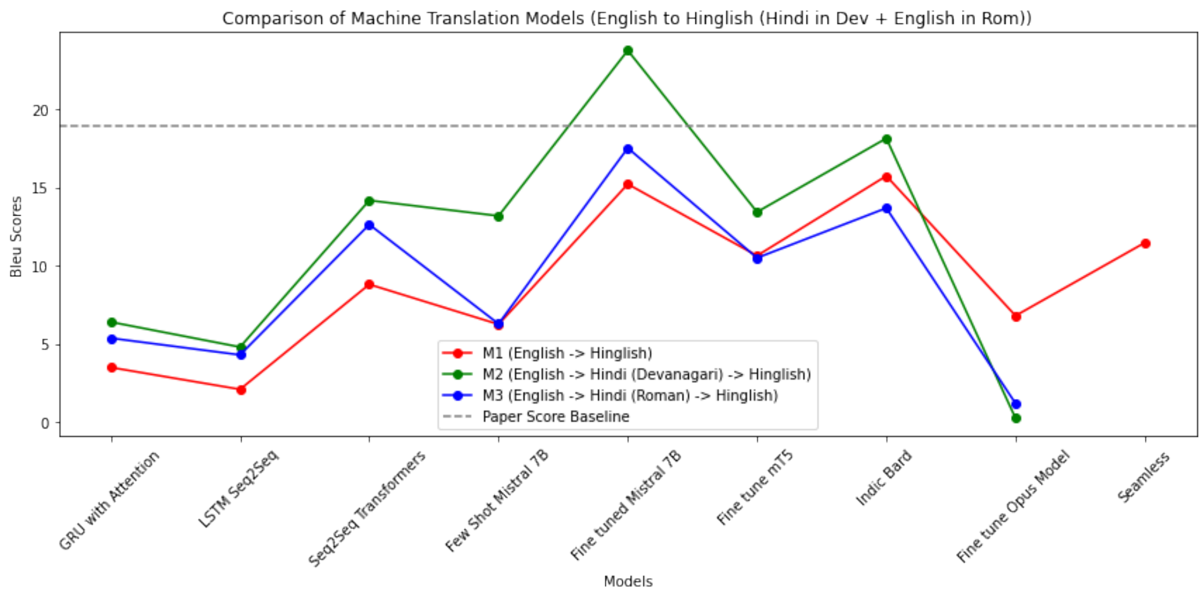


Figure 7: M1: English → Hinglish,
M2: English → Hindi (Pivot Language) (Devanagari) → Hinglish,
M3: English → Hindi (Pivot Language) (Roman) → Hinglish.

Table 3: Comparison of Machine Translation Models with Pivot Translations
M1: English → Hinglish,
M2: English → Hindi (Pivot Language) (Devanagari) → Hinglish,
M3: English → Hindi (Pivot Language) (Roman) → Hinglish.

Model	M1	M2	M3	Paper Score
GRU with Attention Mechanism	4.1	4.3	3.1	15.3
LSTM Sequence to Sequence	2.7	2.6	2.5	15.3
Seq to Seq Transformers with Attention	7.8	7.89	8.1	15.3
Few Shot promoting on Mistral 7B LLM	3.2	9.56	6.1	15.3
Fine tune Mistral 7B llm for MT	11.3	14.1	16.2	15.3
Fine tune mT5-small	7.1	11.46	9.2	15.3
Indic Bard	12.52	15.84	11.5	15.3
Fine tune Opus Model	0.14	0.37	0.27	15.3
Seamless	11.46	—	—	15.3

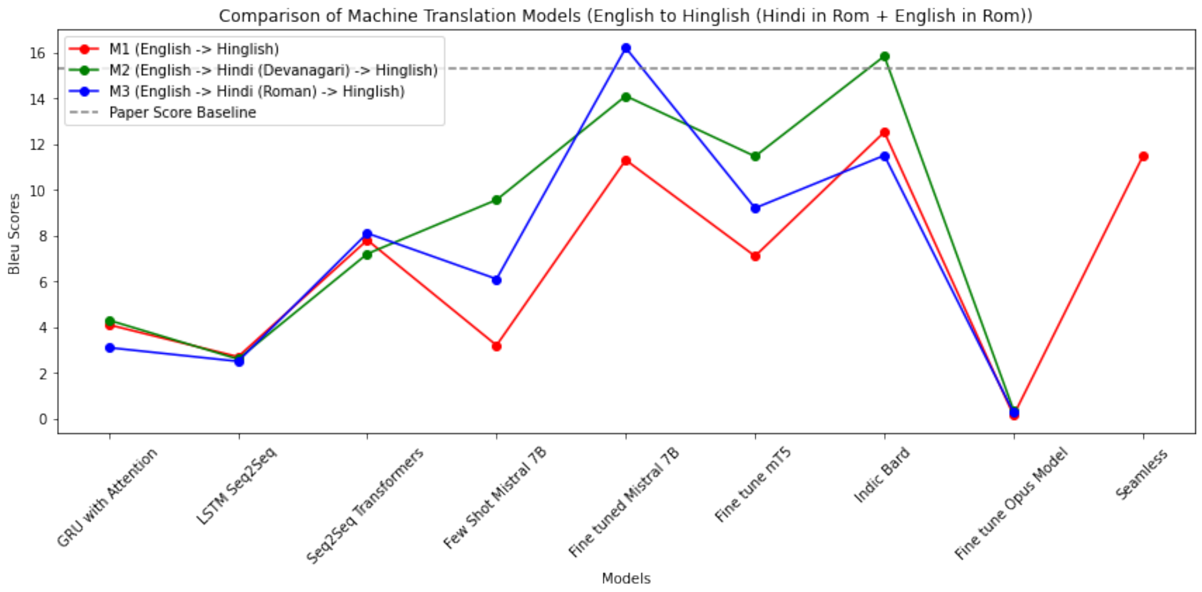


Figure 8: M1: English → Hinglish,
M2: English → Hindi (Pivot Language) (Devanagari) → Hinglish,
M3: English → Hindi (Pivot Language) (Roman) → Hinglish.

7 Evaluation Metrics

We use the following two evaluation metrics for comparing our systems:

1. **BLEU**: The BLEU score (Papineni et al., 2002) is the official metric used in the leaderboard. We calculate the score using the SacreBLEU library[1] (Post, 2018) after lowercasing and tokenization using the TweetTokenizer available with the NLTK library[2] (Bird et al., 2009).

2. **BLEU$_{\textbf{normalized}}$**: Instead of calculating the BLEU scores on the texts where the Hindi words are transliterated to Roman, we calculate the score on texts where Hindi words are in Devanagari and English words in Roman. We transliterate the target sentences using the CSNLI library and we use the outputs of our system before performing the post-processing (Section 3.3). We again use the SacreBLEU library after lowercasing and tokenization using the TweetTokenizer available with the NLTK library.

8 Model Performance Analysis

The performance of various machine translation models when translating from English to Hinglish has been evaluated using BLEU scores. Below is an analysis of each model's performance:

- **GRU with Attention Mechanism**:
  - M1's moderate performance may stem from the attention mechanism's ability to focus on relevant parts of the input sentence, while the better scores for M2 and M3 suggest that using Hindi as a pivot language improves understanding of sentence structure and semantics.
- **LSTM Sequence to Sequence**:
  - The consistently low scores across configurations indicate challenges in managing the complexity of code-switching in Hinglish, pointing towards the limitations of LSTM models in this context.
- **Seq to Seq Transformers with Attention**:
  - This model shows significant improvements, likely due to the self-attention mechanism's efficiency in capturing long-range dependencies, allowing for better modeling of language switching complexities.
- **Few Shot prompting on Mistral 7B LLM**:
  - Varied performance with notably higher scores for M2 indicates that the model, with few-shot learning, can better grasp the translation task when given examples with Hindi as a pivot.
- **Fine-tune Mistral 7B LLM for MT**:
  - Fine-tuning yields the best results, particularly for translations via Hindi pivot, suggesting that task-specific tuning significantly enhances the model's linguistic and contextual comprehension.
- **Fine-tune mT5-small**:
  - When using Hindi as a pivot language for English to Hinglish translation, two approaches exist: Target language Hindi being in Devanagari script and Romanized Hinglish data. Devanagari script poses script differences, resulting in challenging translations and lower BLEU scores. Romanized Hinglish, sharing a script with English, may yield better results. Without Hindi as a pivot, direct Hinglish translation is challenging, resulting in ordinary performance.
- **Indic BERT**:
  - The high BLEU scores, especially in M1 and M2, reflect Indic BERT's inherent proficiency with Indian languages, suggesting an advantage in understanding Hinglish nuances.
- **Fine-tune Opus Model**:
  - The Opus model, primarily trained for English to Hindi Task, does not give good results when we use pivot language. It does give decent result for English to Hinglish direct translation because target language has devanagari words, but at times fails to retain the English words in Code Mixed Hinglish output.
- **Seamless**:

[1] https://github.com/mjpost/sacreBLEU
[2] https://www.nltk.org/

  - The model performs well for M1, with no data for M2 and M3, implying optimization for direct translation without pivot languages.

8.1 Challenges Faced During Implementation

- **Sparse Data Resources:** High-quality and large-scale datasets for Hinglish are scarce, making it challenging to train models with good generalization capabilities.
- **Computational Constraints:** The fine-tuning of large language models (LLMs) requires significant computational power, often exceeding the capacity of standard GPUs and making it prohibitive without access to high-end computational resources.
- **Code-Switching Complexity:** Hinglish frequently involves code-switching, where speakers alternate between Hindi and English within a single conversation or even a sentence. This increases complexity for models that need to understand and generate text in such a hybrid context.
- **Cultural Nuances:** Capturing the cultural nuances embedded in Hinglish expressions is a complex task for machine translation systems, which may lack the subtlety required for accurate translation.
- **Transliteration Inconsistencies:** There is no standard transliteration scheme from Devanagari (Hindi script) to the Roman alphabet, leading to inconsistencies in representing Hinglish words across different datasets and systems.
- **Ambiguity and Polysemy:** Hinglish, like many natural languages, contains words that are polysemous – having multiple meanings based on context. This ambiguity poses a significant challenge for natural language understanding and generation.
- **Evaluation Metrics:** Standard evaluation metrics for language tasks may not be fully appropriate for Hinglish due to its unique characteristics, necessitating the development of specialized metrics.
- **Annotated Corpus Quality:** The quality of annotations in Hinglish corpora can be variable, impacting the training of models that rely on high-quality annotated data for tasks such as part-of-speech tagging, named entity recognition, and sentiment analysis.

9 Conclusion

The comparative analysis of various machine translation models has highlighted several insights into the translation from English to Hinglish. Our evaluation underscores that the most effective translation methodology is pivoting through Hindi in the Devanagari script before translating to Hinglish, where Hindi remains in Devanagari and English in Roman script. This approach was most effectively implemented by fine-tuning the Mistral 7B LLM 4-bit quantized model, which yielded superior BLEU scores.

The fine-tuning process allows for a more nuanced understanding of the linguistic intricacies involved in the code-switching between English and Hinglish, as well as the structural and semantic nuances introduced by the Hindi pivot language. The quantization of the model not only helps in maintaining a balance between efficiency and performance but also proves to be an effective strategy for dealing with the complexity of language translation tasks that involve a mixture of scripts and linguistic systems.

10 Future Work

Moving forward, there is a substantial scope for improvement in the English to Hindi translation segment. In this project, we utilized the Meta Seamless large model to handle this part of the translation task. Future work could focus on enhancing this stage of the translation process by exploring advanced models and fine-tuning techniques, especially those tailored for Indian languages.

Such efforts would likely yield improvements in the initial translation phase, which could in turn enhance the overall quality and fluency of the pivot-based Hinglish translation system. By addressing this, we can push the boundaries of machine translation performance closer to the nuanced understanding that is characteristic of human translation.

11 Neural Machine Translation model GRU with Attention Mechanism

11.1 Model Architecture

The NMT model with GRUs and attention can be divided into two main components: the encoder and the decoder.

11.1.1 Encoder

The encoder is a neural network that processes the input sentence, one word at a time, and encodes the information into a set of vectors. In our model, we use a bidirectional GRU architecture, which consists of two GRUs that process the input sentence in opposite directions. The forward GRU reads the sentence from the first word to the last, while the backward GRU reads it from the last word to the first. Mathematically, the forward hidden state $\overrightarrow{h}_t$ and the backward hidden state $\overleftarrow{h}_t$ at time step t are computed as follows: $\overrightarrow{h}_t = \overrightarrow{GRU}(x_t, \overrightarrow{h}_{t-1})$
$\overleftarrow{h}_t = \overleftarrow{GRU}(x_t, \overleftarrow{h}_{t+1})$ The final encoder hidden state for each word is obtained by concatenating the forward and backward hidden states: $h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$.

11.1.2 Decoder

The decoder is another GRU network that generates the translated sentence one word at a time. At each time step t, the decoder GRU calculates the hidden state s_t using the previous hidden state s_{t-1} and the context vector c_t as inputs: $\mathrm{s}_t = GRU(y_{t-1}, s_{t-1}, c_t)$ The context vector c_t is a weighted sum of the encoder hidden states and is calculated using the attention mechanism.

11.1.3 Attention Mechanism

The attention mechanism allows the model to focus on different parts of the input sentence at each step of the decoding process. It computes a set of attention weights α_{tj} that indicates the importance of the encoder's j-th hidden state to the current time step t in the decoder. These weights are computed as follows: $\mathrm{e}_{tj} = v^\top \tanh(W s_{t-1} + U h_j)$
$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$
$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j$ where v, W, and U are learnable parameters of the attention mechanism. The context vector c_t is then used to generate the output word at each time step.

11.1.4 Output Layer

The output layer converts the decoder hidden state s_t and the context vector c_t into a probability distribution over the target vocabulary. This is typically done using a linear layer followed by a softmax activation: $\mathrm{o}_t = \mathrm{softmax}(V[s_t; c_t] + b)$ where V is a weight matrix and b is a bias vector.

11.2 Training

The model is trained end-to-end using a parallel corpus of source-target sentence pairs. The objective is to minimize the cross-entropy loss between the predicted and actual target sequences. During training, teacher forcing is often used, where the true target word is provided as input to the decoder at the next time step.

11.3 Inference

During inference, the translated sentence is generated one word at a time. At each step, the model selects the word with the highest probability from the softmax layer, and this word is fed into the decoder at the next time step. Beam search is commonly used to improve the quality of the output by considering multiple translation hypotheses at once.

11.4 Conclusion

NMT models with GRUs and attention have shown state-of-the-art performance in various language pairs. The attention mechanism, in particular, has been instrumental in handling long-distance dependencies and ensuring that the model can focus on relevant parts of the input sentence at each step of the translation process.

12 LSTM Sequence to Sequence

12.1 Description

This section aims to develop a neural machine translation model using an LSTM-based encoder-decoder architecture for English-to-Hinglish translation. The model is trained using a teacher-forcing approach, which involves feeding the correct output sequence from the previous time step as input to the decoder during training. Using teacher forcing can lead to faster convergence and better model accuracy. This section explores the effectiveness of this approach in improving the performance of the English-to-Hinglish translation model.

12.2 Introduction

Machine Translation is a complex task that involves the automatic conversion of text from one language to another. With the advent of deep learning, LSTM-based seq2seq models have emerged as a powerful approach to this task. They are capable of handling variable-length input and output sequences, making them well-suited for language processing tasks such as machine translation.

12.3 LSTM Sequence-to-Sequence Architecture

The LSTM seq2seq architecture comprises two main components: the encoder and the decoder, both of which are built using LSTM layers. This architecture is designed to read an input sequence and generate an output sequence, making it ideal for translation tasks.

12.3.1 Encoder

The Encoder is responsible for processing the input sequences and summarizing them into a fixed-length context vector. In this specific code, the Encoder is implemented using Long Short-Term Memory (LSTM) layers. Here's how the Encoder works: $\mathrm{f}_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
$\tilde{C_t} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
$C_t = f_t * C_{t-1} + i_t * \tilde{C_t}$
$h_t = o_t * \tanh(C_t)$ where x_t is the input at the current time step, h_t is the hidden state, C_t is the cell state, σ denotes the sigmoid function, and W and b represent the weights and biases.

12.3.2 Embedding layer

The input sequences, typically sentences in the source language, are first passed through an embedding layer. The embedding layer maps each word in the input sequence to a continuous vector space, where similar words have similar embeddings. This step is essential for converting discrete words into continuous representations that the model can work with. LSTM Layers:

The embedded input sequences are then processed through one or more LSTM layers. LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) layer that is capable of capturing long-range dependencies in sequences. LSTM layers are bidirectional, meaning they process the input sequences in both forward and backward directions. This bidirectional processing allows the model to capture context information effectively.

12.3.3 Hidden state

The final hidden states from the bidirectional LSTM layers serve as a representation of the entire input sequence. These hidden states capture the context and important information from the source

12.3.4 Decoder

The Decoder takes the context vector produced by the Encoder and generates the target sequences, which are translations in the target language. Here's how the Decoder works:

12.3.5 Embedding layer

Similar to the Encoder, the input to the Decoder, which is the target sequence, goes through an embedding layer. This layer converts the target words into continuous vectors. LSTM Layers:

The embedded target sequence is processed through one or more LSTM layers in the Decoder. These LSTM layers are unidirectional, meaning they process the target sequence from left to right.

12.3.6 Sequence Generation

At each time step, the Decoder produces an output token (word or symbol) using the LSTM state and information from the Encoder. This token is typically selected based on the highest predicted probability among all possible tokens in the target vocabulary. The predicted token is then fed back into the model as input for the next time step, and the process continues until an end-of-sequence token is generated or a predefined maximum sequence length is reached. The combination of the Encoder and Decoder allows the model to learn to map input sequences to output sequences, making it suitable for sequence-to-sequence tasks like machine translation. The bidirectional Encoder captures context from the source language, while the unidirectional Decoder generates translations in the target language.

p(y$_t|y_{t-1}, ..., y_1$, context) = softmax($W_s h_t + b_s$) where y_t is the output word at time step t, and W_s and b_s are the softmax layer parameters.

12.4 Training

The seq2seq model is trained to maximize the likelihood of the correct translation given the input sequence. This is typically achieved by minimizing the cross-entropy loss between the predicted word probabilities and the actual words in the target sequence. During training, teacher forcing is used where the correct word is fed as the next input to the decoder instead of the model's prediction.

12.5 Attention Mechanism

Modern seq2seq models often integrate an attention mechanism, which allows the decoder to focus on different parts of the input sequence during each step of the translation. This is particularly beneficial for longer sequences, where the context vector may not be sufficient to capture all the necessary information.

12.5.1 Attention Calculation

The attention weights are calculated based on the decoder's current hidden state and all the hidden states of the encoder: $\alpha_{tj} = \frac{\exp(score(h_t, \bar{h}_j))}{\sum_{k=1}^{T} \exp(score(h_t, \bar{h}_k))}$
where $score(h_t, \bar{h}_j) = h_t^\top W_a \bar{h}_j$ The context vector for each time step in the decoder is then computed as a weighted sum of the encoder's hidden states.

12.6 Inference

During inference, the model uses beam search or greedy decoding to generate the translation. Beam search maintains a fixed number of hypotheses at each step, selecting the most probable sequence as the final translation.

12.7 Evaluation

Seq2seq models are evaluated using metrics like BLEU, METEOR, and TER, which compare the machine-generated translation to a set of reference translations. These metrics measure the quality and fluency of the translations.

12.8 Conclusion

LSTM-based seq2seq models have significantly advanced the field of machine translation. Their ability to model sequences in a context-aware manner has led to remarkable improvements in translation quality. Ongoing research continues to refine these models, making them even more efficient and accurate.

13 Seq to Seq Transformers Machine Translation model with Attention

13.1 Introduction

Machine Translation has historically been a challenging domain within Natural Language Processing. The introduction of Transformer models has revolutionized this field by dispensing with recurrent architectures in favor of attention mechanisms, enabling parallel processing and capturing long-distance dependencies more effectively.

13.2 Transformer Architecture

The Transformer adopts an encoder-decoder structure. However, unlike its LSTM and GRU predecessors, it relies entirely on self-attention mechanisms to weigh the influence of different parts of the input data.

13.2.1 Encoder

The encoder consists of a stack of identical layers, each containing two main sub-layers: a multi-head self-attention mechanism, and a position-wise fully connected feed-forward network. Residual connections around each of these sub-layers followed by layer normalization are also a critical part of the architecture.

13.2.2 Decoder

The decoder also contains a stack of identical layers but adds a third sub-layer, which performs multi-head attention over the encoder's output. Similar to the encoder, residual connections and layer normalization are employed.

13.2.3 Attention Mechanisms

Attention mechanisms are the heart of the Transformer model. They come in three variants: self-attention in the encoder, encoder-decoder attention in the decoder, and self-attention within the decoder. These mechanisms allow the model to focus on different parts of the input sequence as needed.

Scaled Dot-Product Attention The attention function takes as input a query Q, keys K, and values V, and computes the dot products of the query with all keys, divides each by $\sqrt{d_k}$, and applies a softmax function to obtain the weights on the values.

Multi-Head Attention Instead of performing a single attention function, the model learns to project the queries, keys, and values multiple times with different, learned linear projections. This allows the model to jointly attend to information from different representation subspaces at different positions.

13.3 Positional Encoding

Since the model contains no recurrence or convolution, positional encodings are added to the input embeddings to provide some information about the order of the sequence.

13.4 Training

Training a Transformer model for Machine Translation involves optimizing the parameters to minimize the loss function, typically cross-entropy, over a parallel corpus of source-target sentence pairs.

13.5 Inference

For inference, the Transformer model employs beam search to generate translations by selecting the most probable next word at each step in the sequence.

13.6 Evaluation

The quality of translations is evaluated using standard metrics such as BLEU, METEOR, and ROUGE, which compare the predicted sequences to a set of reference translations.

13.7 Conclusion

The Transformer model represents a significant advance in machine translation. It demonstrates the power of the attention mechanism, providing improved translation quality with efficient training and inference processes.

14 Mistral 7B Model: Architectural Report

14.1 Introduction

Mistral 7B is a 7-billion-parameter language model designed for high performance and efficiency. It outperforms existing models in benchmarks and incorporates advanced attention mechanisms for faster inference and longer sequence handling without increased computational cost.
The Mistral 7B model addresses the computational barriers posed by large language models through a balanced design that achieves high performance with efficient inference. It leverages innovative attention mechanisms to outperform larger models while maintaining efficiency, making it suitable for real-world applications.

14.2 Architectural Details

Mistral 7B's architecture is built on a transformer framework with key innovations in attention mechanisms. Below, we highlight the architecture's main components and the strategic optimizations that enable its enhanced performance.

14.2.1 Sliding Window Attention

Mistral 7B uses Sliding Window Attention (SWA) to allow each token to attend to a window of tokens from the previous layer, permitting information flow through the network over longer sequences with a lower computational overhead.

14.2.2 Rolling Buffer Cache

The model implements a Rolling Buffer Cache to maintain a fixed cache size for keys and values, thus optimizing memory usage during sequence generation.

14.2.3 Pre-fill and Chunking

For efficient handling of long sequences, Mistral 7B uses a pre-fill strategy with chunking, processing prompts in manageable sizes to optimize memory and computational resources.

14.3 Model Specifications

Here we outline the specific parameters that define the Mistral 7B architecture:

14.4 Conclusion

Mistral 7B is poised to set a new standard in large language models by providing a synergistic balance between performance and efficiency. Its innovations in attention mechanisms serve as a blueprint for future advancements in the field of NLP.

Parameter	Value
Dimension (dim)	4096
Number of Layers (n_layers)	32
Head Dimension (head_dim)	128
Hidden Dimension (hidden_dim)	14336
Number of Heads (n_heads)	32
Number of Key/Value Heads (n_kv_heads)	8
Window Size (window_size)	4096
Context Length (context_len)	8192
Vocabulary Size (vocab_size)	32000

Table 4: Architectural parameters of Mistral 7B.

15 Fine tune mT5

15.1 mT5 Architecture

mT5 extends the T5 architecture to over 100 languages, utilizing a modified pre-training regimen based on the Common Crawl (mC4) dataset. The model incorporates GeGLU nonlinearities and adapts its sampling strategy to balance language representation with an optimal $\alpha = 0.3$. Accommodating diverse languages, mT5 increases its vocabulary to 250,000 wordpieces and ensures comprehensive character representation with a high character coverage and SentencePiece's "byte-fallback" feature.

16 Fine tune Opus Model

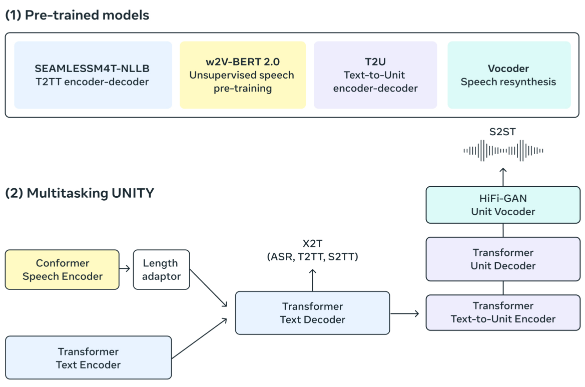
16.1 Opus Model Overview

The Opus Model epitomizes a fine-tuned NMT solution, utilizing a transformer-based encoder-decoder structure, enriched by the expansive Opus corpus. It undergoes domain-specific refinement, employing transfer learning and meticulous hyperparameter adjustment to enhance translation precision. Through tokenization and normalization in preprocessing, followed by strategic training with regularization, the model is adeptly adapted for specialized text types. While it excels in certain areas, challenges in English-to-Hinglish translation tasks suggest further fine-tuning and optimization are needed. Its efficacy is quantified through metrics like BLEU, with the goal of surpassing established NMT benchmarks.

17 Seamless

17.1 Model Architecture

The SeamlessM4T leverages the UnitY model architecture within fairseq2, a sequence modeling toolkit tailored for the PyTorch ecosystem. It incorporates dual encoders: w2v-BERT 2.0 for self-supervised speech processing and the NLLB model for text encoding, covering nearly 100 languages. A text decoder translates these representations into text, while a text-to-unit model generates discrete acoustic units for speech synthesis in 36 languages, followed by a HiFi-GAN unit vocoder for speech output. This architecture enables multitask functionality including speech recognition, text translation, and speech generation, underpinned by pre-training for improved performance and stability.



(a) Meta Seamless

A Hyperparameters of Machine Translation Models

This appendix provides details about the hyperparameters used for each of the machine translation models discussed in the main content of the paper.

A.1 GRU with Attention Mechanism

Table 5: Hyperparameters for GRU with Attention Mechanism

Hyperparameter	Value
Embedding Size	256
GRU Units	128
Batch Size	64
Learning Rate	0.01
Dropout Rate	0.1

A.2 LSTM Sequence to Sequence

Table 6: Hyperparameters for LSTM Sequence to Sequence

Hyperparameter	Value
Embedding Size	300
LSTM Units	3
Batch Size	64
Learning Rate	0.01
Dropout Rate	0.1

A.3 Seq to Seq Transformers with Attention

Table 7: Hyperparameters for Seq to Seq Transformers with Attention

Hyperparameter	Value
Embedding Size	512
Transformer Units	2
Number of Heads	8
Batch Size	8
Learning Rate	10e-4
Feed Forward	2048
Dropout Rate	0.1

A.4 Few Shot Promoting on Mistral 7B LLM

Table 8: Hyperparameters for Few Shot Promoting on Mistral 7B LLM

Hyperparameter	Value
Quantization	16bit
Do Sample	False
Temperature	0.5
Top k	50
Top p	1
Max New Token	100

A.5 Fine Tune Mistral 7B LLM - 4bit Qlora for MT

Table 9: Hyperparameters for Fine Tune Mistral 7B LLM for MT

Hyperparameter	Value
Method	Qlora
Quantization	4bit
lora alpha	16
lora dropout	0.1
r	64
bias	None
Learning Rate	2e-4
Batch Size	4
Number of Epochs	3
Weight Decay	0.001
Fine-tuning Data Size	7000 Samples

A.6 Fine Tune mT5

Table 10: Hyperparameters for Fine Tune mT5

Hyperparameter	Value
Learning Rate	0.001
Batch Size	2
Number of Epochs	10

A.7 Indic Bard

Table 11: Hyperparameters for Indic Bard

Hyperparameter	Value
Learning Rate	0.0001
Batch Size	64
Number of Epochs	80

A.8 Fine Tune Opus Model

Table 12: Hyperparameters for Fine Tune Opus Model

Hyperparameter	Value
Learning Rate	0.001
Batch Size	8
Number of Epochs	10

A.9 Seamless

Used at Default Parameters for Inference from Hugging Face

References

[1] Vaswani, A., et al. "Attention is All You Need." *Advances in neural information processing systems*, 2017.

[2] Bahdanau, D., et al. "Neural Machine Translation by Jointly Learning to Align and Translate." *ICLR*, 2015.

[3] Sutskever, I., et al. "Sequence to Sequence Learning with Neural Networks." *NIPS*, 2014.

[4] Devlin, J., et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *NAACL-HLT*, 2019.

[5] Liu, Y., et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." *arXiv preprint arXiv:1907.11692*, 2019.

[6] Pires, T., Schlinger, E., and Garrette, D. "How multilingual is Multilingual BERT?" *ACL*, 2019.

[7] CoMeT: Towards Code-Mixed Translation Using Parallel Monolingual. (2021).

[8] CST5: Code-Switched Semantic Parsing using T5

[9] Adapting Multilingual Models for Code-Mixed Translation

[10] https://link.springer.com/chapter/10.1007/978-3-642-16558-0_4

[11] https://ieeexplore.ieee.org/abstract/document/97300/

[12] https://arxiv.org/pdf/1606.04164.pdf

[13] SeamlessM4T: AI Translation Model

[14] mT5: A massively multilingual pre-trained text-to-text transformer

[15] How to train an MT5 model for translation with simple transformers

[16] https://arxiv.org/abs/1706.03762

[17] https://arxiv.org/abs/2310.06825

[18] IndicBERT - a multilingual ALBERT model

[19] https://iopscience.iop.org/article/10.1088/1757-899X/569/5/052037

[20] https://www.hindawi.com/journals/sp/2022/3909726/

[21] Sequence to Sequence Modeling using LSTM for Language Translation

[22] Universal Dependency Parsing for Hindi-English Code-Switching