CODE MIX GENERATION

Automated Generation of Code-Mixed Text (Hinglish)

Author
TEAM- 29
Uday Bindal (2022114015)
Rijul Saigal (2022114001)
Nikhil Sriram (2022114003)
International Institute of Information Technology Hyderabad

Contents

1 Introduction

Code-mixing refers to the blending of two or more languages within a sentence or discourse, a phenomenon increasingly observed in multilingual societies and across digital communication platforms. This linguistic behavior reflects the dynamic and fluid nature of language use in diverse communities, often driven by factors such as social context, speaker identity, and communicative purpose. An example of code-mixing widely recognized in the Indian subcontinent is Hinglish, a blend of Hindi and English.

2 Problem Statement

In multilingual contexts, especially within the Indian subcontinent, there is a frequent interchange of English and Hindi within a single discourse, creating a language blend known as Hinglish. The computational challenge is to devise a system capable of generating sentences that accurately represent this linguistic fusion, using input data consisting of English sentences. Specifically, the system must take as input sentences in English (L1) and produce a code-mixed sentence that combines elements of both languages while maintaining grammatical integrity and semantic coherence.
For example:
The two given sentences are
L1: A girl is holding a red umbrella.
The system should generate a code-mixed sentence like: Ek girl ek red umbrella liye hue hai.

3 Scope of The Project

The aim is to develop a machine learning model capable of generating coherent and contextually appropriate code-mixed text from bilingual input data in English and Hindi. This model will assist in enhancing the natural language processing capabilities for the linguistically diverse user-generated content on digital platforms.

3.1 Research and Development Focus

- **Bilingual Corpus Compilation:** Collection and preprocessing of English-Hindi bilingual corpora that includes both monolingual and code-mixed examples. The corpus will be annotated to mark code-switch points and language tags.
- **Code-Mixing Pattern Identification:** The model will learn typical code-mixing patterns within the corpus, focusing on intrasentential switches and establishing criteria for where and how languages are mixed.

3.2 Technical Boundaries

- **Language Pair Limitation:** The project is limited to the English-Hindi language pair, also known as Hinglish, and the algorithms will be fine-tuned to cater specifically to the nuances of these languages.

- **Scope of Code-Mixing:** The model will be designed to understand and generate common patterns of code-mixing but may not cover all sociolinguistic variations.
- **Resource Dependence:** Model performance is dependent on the quality and size of the bilingual corpus used for training.

The model will be implemented in Python using prominent machine learning frameworks such as TensorFlow or PyTorch. Key stages of implementation include:

- **Data Preprocessing:** Cleaning, tokenizing, and annotating the dataset for training.
- **Model Development:** Designing and training statistical models and neural networks to identify and generate code-mixing patterns.
- **Evaluation:** Assessing the performance of the model using metrics like BLEU score for language generation tasks and accuracy for pattern recognition tasks.

4 Datasets

For the scope of this project, we will use the HinGE dataset. HinGE has Hinglish sentences generated by humans as well as two rule-based algorithms corresponding to the parallel Hindi-English sentences.
The dataset contains the following columns:

- English, Hindi: The parallel source sentences from the IITB English-Hindi parallel corpus.
- Human-generated Hinglish: A list of Hinglish sentences generated by the human annotators.
- WAC: Hinglish sentence generated by the WAC algorithm.
- WAC rating1, WAC rating2: Quality rating to the Hinglish sentence generated by the WAC algorithm. The quality rating ranges from 1-10.
- PAC: Hinglish sentence generated by the PAC algorithm.
- PAC rating1, PAC rating2: Quality rating to the Hinglish sentence generated by the PAC algorithm. The quality rating ranges from 1-10.

5 Literature Review

1. **A Semi-supervised Approach to Generate the Code-Mixed Text using Pre-trained Encoder and Transfer Learning by Deepak Gupta, Asif Ekbal, Pushpak Bhattacharyya**
   This paper presents a study on generating code-mixed Hinglish dialogues, focusing on incorporating linguistic constraints and context awareness into the model. The authors propose a model that leverages pre-trained language models and neural architecture search techniques to generate code-mixed Hinglish dialogues. The study highlights the importance of context and linguistic constraints in improving the quality of code-mixed dialogue generation.

2. **A Comprehensive Understanding of Code-mixed Language Semantics using Hierarchical Transformer by Ayan Sengupta*, Tharun Suresh*, Md Shad Akhtar, and Tanmoy Chakraborty**
   This paper advances the understanding of code-mixed language semantics by employing a hierarchical transformer model. This model captures the complexities of code-switching at different linguistic levels, offering insights into the semantic coherence of code-mixed sentences and facilitating more accurate text generation

3. **Marathi-English Code-mixed Text Generation and L3Cube-HingCorpus and HingBERT**
   These papers contribute significantly to the field by developing and utilizing specialized datasets and models tailored for code-mixed language processing. The creation of HingCorpus and HingBERT, for instance, marks a pivotal step towards building resources dedicated to Hinglish text analysis and generation.

Baseline Models

Model	en-es			en-de			en-fr			en-hi		
	B	**R**	**M**	**B**	**R**	**M**	**B**	**R**	**M**	**B**	**R**	**M**
Seq2Seq	16.42	36.03	24.23	19.19	36.19	24.87	19.28	38.54	26.41	15.49	35.29	23.72
Attentive-Seq2Seq	17.21	36.83	25.41	20.12	37.14	25.64	20.12	39.30	27.54	16.55	36.25	24.97
Pointer Generator	18.98	37.81	26.13	21.45	38.22	26.14	21.41	40.42	28.76	17.62	37.32	25.61

Here, B: BLEU, R: Rouge-L and M: METEOR

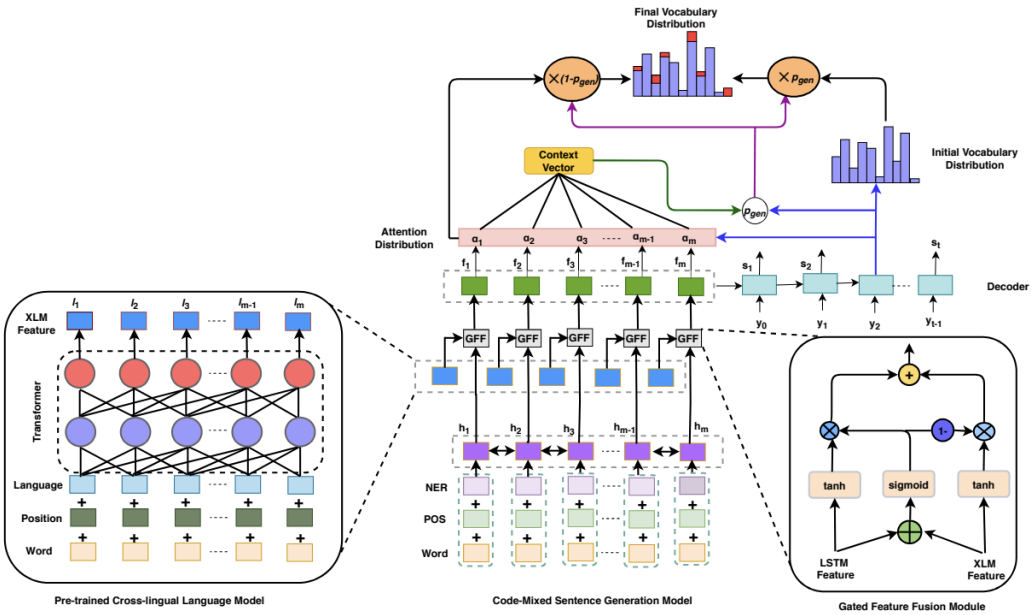
These models serve as baselines due to their effectiveness in sequence-to-sequence tasks, with Seq2Seq providing a basic framework, Attentive Seq2Seq improving attention mechanisms, and Pointer Generator addressing vocabulary limitations by enabling direct copying from the input. They collectively offer a solid foundation for codemixed language generation tasks.

6 Implementation

Our implementation involves two main components:

- A text encoder-decoder model with attention mechanism to translate English sentences into codemixed Hinglish.
- Integration of a pre-trained XLM model to capture cross-lingual semantic representations and improve translation accuracy.

The text encoder encodes the input English sentence into a fixed-length vector representation, which is then decoded by the text decoder to generate the corresponding Hinglish sentence. The attention mechanism enables the model to focus on relevant parts of the input sentence during decoding, enhancing translation quality. The feature-rich encoder assists the model to capture the linguistic phenomenon of code-mixing, especially to decide when to switch between the two languages.
Additionally we also perform transfer learning to learn the prior distribution from the pre-trained NMT. The pre-trained NMT weights are used to initialize the code-mixed generation



network.

The XLM model provides contextualized embeddings for both English and Hinglish tokens, enabling the model to capture language-specific features and improve translation accuracy for codemixed sentences.

6.1 Step By Step Implementation:

Importing Libraries: The code begins with importing necessary libraries such as PyTorch modules for neural network operations, spaCy for natural language processing tasks, BERT tokenizer, XLM-RoBERTa model from Hugging Face's Transformers library, and pickle for loading data from a file.

Custom Modules:

- Attention: Defines the attention mechanism used in the LSTM-Attention module.
- LSTM-Attention: Implements the LSTM-based sequence-to-sequence model with attention.
- TextDataset: Custom dataset class for loading and processing text data.
- TextEncoder: Encodes text inputs into embeddings using an LSTM-based encoder.
- XLMEncoder: Encodes text inputs using a pre-trained XLM-RoBERTa model.
- GatedFeatureFusion: Integrates the outputs of TextEncoder and XLMEncoder using a gating mechanism.

Data Preparation: A dataset (TextDataset) is created with English sentences, tokenized using BERT tokenizer, and annotated with part-of-speech tags and named entity recognition. DataLoader is used to create batches of data for training.

Training Loop:

- The training loop runs for a specified number of epochs. Inside the loop, the optimizer's gradients are zeroed. Forward pass:
- The input text is passed through the GatedFeatureFusion model, which combines features from TextEncoder and XLMEncoder.
- The fused output is then passed through the LSTM-Attention decoder model to generate predictions.
- Loss calculation: The loss is computed using a cross-entropy loss function.
- Backpropagation: Gradients are calculated and propagated back through the network. Model parameters are updated using the optimizer.

7 Metrics For Evaluation

The evaluation of code-mixed text generation can involve several metrics:

1. **METEOR (Metric for Evaluation of Translation with Explicit Ordering):** An evaluation metric that involves matching generated text to reference translations at the level of words and phrases, which can be adapted for code-mixed text.
2. **BLEU Score (Bilingual Evaluation Understudy):** A metric for evaluating the quality of text which has been machine translated from one language to another. It measures the similarity between the machine-generated text and one or more reference texts