Deep Learning Approach for English to Hindi Language Translation

Amal Thomas
Department of Mathematics, School of Advanced Studies
Vellore Institute of Technology Chennai, Tamil Nadu-600127
amal.thomas2023@vitstudent.ac.in

1 abstract

This project explores a deep learning approach for translating text from English to Hindi. It focuses on building a neural machine translation model using sequence-to-sequence (Seq2Seq) models, encoder-decoder architecture, and attention mechanisms to improve translation accuracy and fluency.

2 Introduction

Machine translation is a critical component in the field of natural language processing, with applications ranging from cross-lingual communication to content localization. This project presents a deep learning approach for English to Hindi language translation using neural machine translation (NMT) models. Leveraging a sequence-to-sequence (Seq2Seq) framework with an encoder-decoder architecture, we incorporate an attention mechanism to capture linguistic nuances between English and Hindi. The model is trained on a bilingual corpus, utilizing recurrent neural networks (RNNs) and long short-term memory (LSTM) units to handle sequential dependencies. Evaluation metrics such as BLEU score are used to assess translation quality, with results demonstrating the model's capability in generating accurate and contextually relevant Hindi translations. This study highlights both the challenges and potential of deep learning in language translation, particularly for complex, morphologically rich languages like Hindi.

3 Problem Statement

This project aims to develop a deep learning-based solution for translating English text into Hindi, addressing challenges arising from the distinct grammatical structures and linguistic nuances of the two languages. By using a sequence-to-sequence model with attention mechanisms, the goal is to improve translation accuracy and produce coherent, contextually appropriate Hindi translations.

4 Evaluation

The effectiveness of the English to Hindi translation model is evaluated using the BLEU (Bilingual Evaluation Understudy) score, a standard metric for machine translation. BLEU score measures the similarity between the model-generated translation and a set of reference translations, capturing both precision and n-gram overlap.

4.1 BLEU Score Calculation

The BLEU score is computed by comparing n-grams (typically 1-gram to 4-gram) in the generated translation with those in the reference translation. The formula for calculating the BLEU score is:

$$\text{BLEU} = \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \times BP \tag{1}$$

where p_n is the precision of n-grams, w_n is the weight assigned to each n-gram (often uniform, so $w_n = \frac{1}{N}$), and BP is the brevity penalty to penalize translations that are shorter than the reference translation.

4.2 Brevity Penalty (BP)

The brevity penalty (BP) is defined as:

$$BP = \begin{cases} 1 & \text{if } c > r \\ \exp\left(1 - \frac{r}{c}\right) & \text{if } c \leq r \end{cases} \tag{2}$$

where c is the length of the candidate translation and r is the effective reference length. This penalty encourages translations that are closer in length to the reference.

4.3 Interpretation of BLEU Score

BLEU scores range from 0 to 1, where a higher BLEU score indicates greater similarity to the reference translations. Typically, BLEU scores for machine translation models fall in the range of 0.3 to 0.5, with scores closer to 1 indicating high-quality translations.

5 Model Architecture

This project utilizes a neural machine translation model based on a sequence-to-sequence (Seq2Seq) architecture with an encoder-decoder framework, enhanced by an attention mechanism. The following subsections provide a detailed description of each component.

5.1 Encoder

The encoder is responsible for processing the input English sentence and converting it into a context vector. This context vector represents the semantic information of the sentence. In this model, a recurrent neural network (RNN) with long short-term memory (LSTM) units is used to capture the sequential dependencies in the sentence. The LSTM units effectively retain long-term dependencies, making them suitable for handling complex language structures.

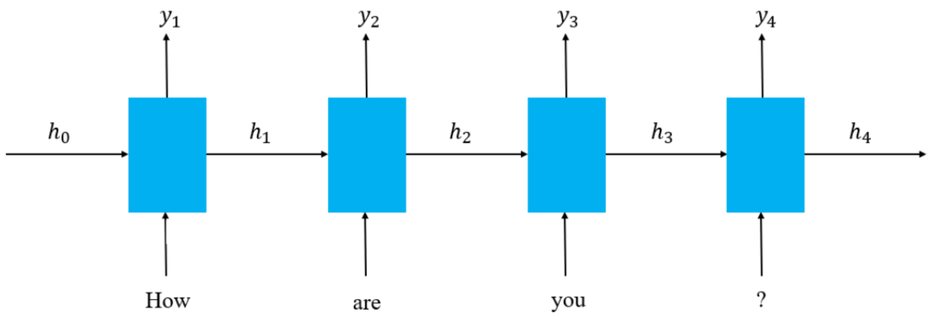


Figure 1: Encoder architecture used in the English-to-Hindi translation model. The encoder transforms the input sequence into a context vector, capturing essential information needed for translation.

5.2 Decoder

The decoder generates the Hindi translation by taking the context vector from the encoder and producing the output sentence word by word. Each output word is influenced by the context vector as well as the previously generated words, allowing for coherent sentence construction. The decoder also utilizes LSTM units to manage sequential dependencies in the target language.

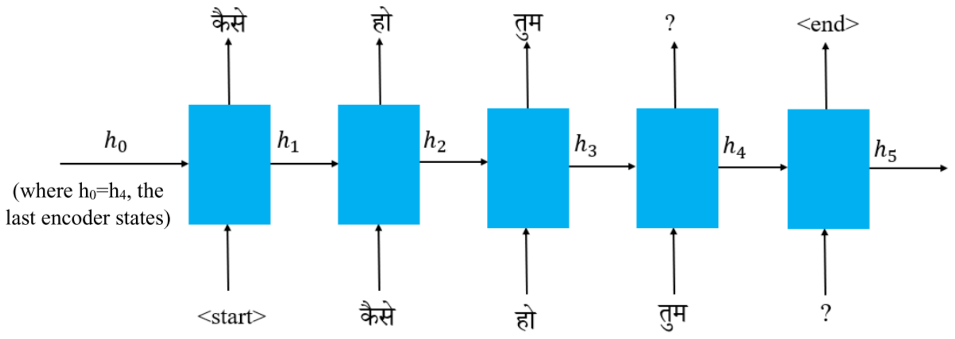


Figure 2: Encoder architecture used in the English-to-Hindi translation model. The encoder transforms the input sequence into a context vector, capturing essential information needed for translation.

5.3 Attention Mechanism

To improve translation accuracy, an attention mechanism is incorporated into the model. Instead of relying solely on the fixed-length context vector, the attention mechanism allows the decoder to access all hidden states of the encoder, enabling it to focus on the most relevant parts of the input sentence for generating each word. This approach enhances the model's ability to handle long and complex sentences by aligning specific words between the source and target languages.

5.4 LSTM Units

Both the encoder and decoder use long short-term memory (LSTM) units to manage the sequential nature of language data. LSTM networks are capable of retaining long-term dependencies, which are essential for capturing the linguistic nuances between English and Hindi, two languages with significant structural differences.

6 Experiment Setup

6.1 About the Dataset

The dataset used in this project consists of parallel English and Hindi text pairs intended for sequence-to-sequence language translation. Basic statistics and key details of the dataset are as follows:

Character and Token Counts

The dataset provides counts for characters and tokens (words) in both English and Hindi sentences.

- **English Character Count**: Ranges from 0 to 2132 characters, with an average of about 87 characters.
- **Hindi Character Count**: Ranges from 0 to 2025 characters, averaging around 86 characters.
- **Token Counts**: English token counts average at approximately 15 tokens per sentence, while Hindi averages around 17 tokens.

Summary Statistics

Statistic	**eng_char_count**	**hindi_char_count**	**hindi_tok_count**	**eng_tok_count**
Count	20000.000	20000.000	20000.000	20000.000
Mean	87.2839	86.4757	17.1618	15.3340
Standard Deviation	81.7711	83.1602	16.0653	14.0926
Minimum	0.0000	0.0000	1.0000	1.0000
25th Percentile	38.0000	37.0000	8.0000	7.0000
Median (50%)	64.0000	64.0000	13.0000	12.0000
75th Percentile	115.0000	112.0000	22.0000	20.0000
Maximum	2132.0000	2025.0000	417.0000	398.0000

Table 1: Basic statistics of character and token counts for English and Hindi sentences

Missing Values

There are no missing values in the dataset across the following fields:

- `source` (indicating the language pair source)

- `eng_char_count` (English character count)
- `hindi_char_count` (Hindi character count)
- `hindi_tok_count` (Hindi token count)
- `eng_tok_count` (English token count)

This information provides insight into the dataset's structure, which helps inform preprocessing and model design choices, such as handling variable sentence lengths and character counts in both languages.

6.2 Data Preprocessing

The dataset for this project consists of paired English and Hindi sentences. The preprocessing steps involve:

- **Tokenization**: Splitting sentences into words (tokens) and converting them into numerical representations.
- **Padding**: Ensuring all sentences have a uniform length by padding shorter sentences, which allows for batch processing.
- **Vocabulary Building**: Creating a vocabulary of unique words for both the English and Hindi datasets, allowing each word to be mapped to an integer.
- **Data Splitting**: Dividing the dataset into training and validation sets to facilitate model training and evaluation.

6.3 Model Configuration

The neural translation model follows a sequence-to-sequence (Seq2Seq) architecture with an encoder-decoder framework, using long short-term memory (LSTM) units for both the encoder and decoder. Key parameters include:

- **Embedding Dimensions**: Set to capture meaningful word representations in a continuous vector space.
- **Hidden Units**: Determines the size of the LSTM hidden layers, impacting the model's capacity to capture dependencies.
- **Attention Mechanism**: Incorporated to improve translation quality by focusing on relevant parts of the input sentence during decoding.
- **Batch Size and Learning Rate**: Batch size controls the number of sentences processed at once, and learning rate dictates the step size in the optimization process.

6.4 Training Procedure

The model is trained by feeding the English sentences into the encoder and passing the generated context vector to the decoder, which outputs the translated Hindi sentences. The training process minimizes the cross-entropy loss between predicted and target outputs over multiple epochs. During training, the BLEU score is calculated on the validation set to monitor translation quality.

6.5 Evaluation Metrics

After completing the training, the model's translation performance is evaluated using the BLEU score, which compares the model's output to the reference Hindi translations. A higher BLEU score indicates better alignment with the reference translations, reflecting improved translation accuracy.

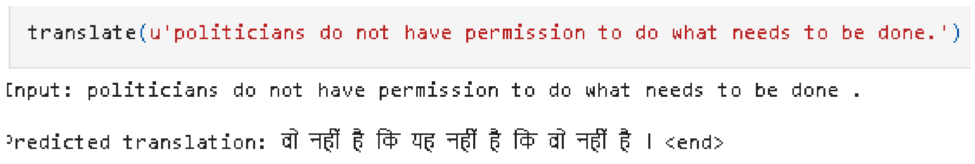
7 Translation

7.1 Translation Process

Once the model has been trained, it is used to translate English sentences into Hindi. The process involves passing the English input sentence through the encoder, which generates a context vector. The decoder then utilizes this context vector, along with the attention mechanism, to produce the Hindi sentence one word at a time. During this decoding process, the attention mechanism helps the decoder focus on specific parts of the input sentence, improving translation accuracy.

To assess the quality of translations, a few examples from the test set are displayed with both the model-generated translations and the reference translations. This qualitative evaluation provides insights into the model's performance on various sentence structures and contexts.

8 Result



```
translate(u'politicians do not have permission to do what needs to be done.')
```

```
Input: politicians do not have permission to do what needs to be done .

Predicted translation: वो नहीं है कि यह नहीं है कि वो नहीं है । <end>
```

Figure 3: Encoder architecture used in the English-to-Hindi translation model. The encoder transforms the input sequence into a context vector, capturing essential information needed for translation.

9 Coclusion

In this project, we developed a model to translate English to Hindi using a deep learning approach with an encoder-decoder structure and attention mechanism. This model captures patterns between English and Hindi sentences, allowing it to produce Hindi translations based on English input. Although we achieved promising initial results, the translation quality could improve with further training. Due to the time required for processing, we trained the model for only 10 epochs. Running additional training cycles would likely enhance the model's accuracy and make the translations more reliable, showing the potential of deep learning for language translation tasks.

10 Reference

- Towards Data Science: Recurrent Neural Networks, `https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce`
- Towards Data Science: Illustrated Guide to LSTMs and GRUs, `https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21`
- Sequence to Sequence Learning with Neural Networks (Arxiv), `https://arxiv.org/pdf/1508.04025`
- Sequence-to-Sequence Learning by Guillaume Genthial, `https://guillaumegenthial.github.io/sequence-to-sequence.html`
- Towards Data Science: Understanding GRU Networks, `https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be`
- Andrej Karpathy: The Unreasonable Effectiveness of Recurrent Neural Networks, `https://karpathy.github.io/2015/05/21/rnn-effectiveness/`