COPS Summer of Code 2025

Intelligence Guild

Club Of Programmers, IIT (BHU) Varanasi

Machine Translation(English to Hindi)

11 – 17 June 2025

Contents

1 Introduction

Machine translation is a fundamental task in the field of Natural Language Processing (NLP), where the goal is to automatically translate text from one language (source) to another (target). With the rapid growth of deep learning techniques, traditional rule-based and statistical machine translation systems have been largely replaced by neural machine translation (NMT) models. These models offer a more end-to-end, data-driven approach, delivering significantly improved performance.

1.1 Background

Sequence-to-sequence (Seq2Seq) models have emerged as a foundational architecture for neural machine translation. They consist of an encoder that processes the input sequence and a decoder that generates the output sequence. However, early implementations of Seq2Seq models had limitations in handling long sequences due to fixed-length context vectors.

To address this, attention mechanisms were introduced, allowing the decoder to focus on relevant parts of the input sequence dynamically during translation. Two commonly used attention mechanisms are Bahdanau and Luong attention. These mechanisms significantly enhance the model's ability to capture dependencies, especially in longer sentences.

Further advancements have led to the development of transformer models, which discard the recurrent architecture in favor of self-attention mechanisms. Transformer-based models, such as those available via the Hugging Face library, have set new benchmarks in translation quality and training efficiency.

1.2 Objective

The primary objective of this assignment is to build and evaluate neural machine translation models of increasing complexity:

- Develop a baseline encoder-decoder model using LSTM or GRU units.
- Enhance the model by integrating attention mechanisms (Bahdanau or Luong).
- Implement a transformer-based model using pretrained architectures from Hugging Face.
- Compare the performance of these models using standard evaluation metrics like BLEU score.

1.3 Scope of Work

The scope of this project is outlined as follows:

1. **Part 1: Seq2Seq without Attention**
   Design and implement a basic encoder-decoder model using LSTM or GRU units to perform machine translation.

2. **Part 2: Seq2Seq with Attention**
   Integrate attention mechanisms (Bahdanau or Luong) into the base model. Evaluate and analyze how attention affects translation performance and convergence speed.
3. **Part 3: Transformer-based Model**
   Use pretrained transformer models from Hugging Face for translation tasks. Fine-tune the models on the selected dataset and evaluate their performance.
4. **Evaluation**
   All models will be evaluated using BLEU scores to assess the quality of translations. Additional qualitative analysis may be conducted through example translations and attention visualizations.

2 Dataset and Preprocessing

2.1 Dataset Description

The dataset used for this project is the English-Hindi parallel corpus available on Kaggle, created by Preet Viradiya.[1] It contains over 200,000 sentence pairs where each English sentence has a corresponding Hindi translation. This dataset serves as a good foundation for training and evaluating a supervised neural machine translation model. For experimentation, we split the dataset into 80% training, 10% validation, and 10% testing.

2.2 Cleaning and Normalization

We started with some basic cleaning steps. First, duplicate entries and any rows containing null values were removed. After that, all English text was converted to lowercase for consistency. We then removed HTML tags and special characters using regular expressions, and filtered out URLs from both English and Hindi sentences.

To handle informal or abbreviated text, a chat word dictionary was used to expand short forms (e.g., "u" became "you"). Emojis were also removed from both languages to avoid encoding issues and reduce noise in the data. Finally, contractions in English (like "can't", "won't") were expanded using the `contractions` Python library.

2.3 Tokenization

For tokenization, we used Keras' `Tokenizer` separately for English and Hindi texts. Each word was mapped to a unique integer index. Additionally, every Hindi sentence was wrapped with special tokens: `<start>` at the beginning and `<end>` at the end, which helps during decoding in sequence generation.

2.4 Stopword Removal and Stemming

Since this is a machine translation task, we did not apply stopword removal or stemming. Every word, including stopwords and inflected forms, may carry important semantic or grammatical meaning needed for accurate translation.

[1] https://www.kaggle.com/datasets/preetviradiya/english-hindi-dataset

2.5 Sequence Padding and Vocabulary Creation

After converting the text to sequences of integers, we padded the sequences so that all of them have the same length — which is required for batch training. The maximum sequence length was calculated based on the longest sentence in either language. The final vocabulary size was 22,870 for English and 24,779 for Hindi. Special tokens like `<pad>`, `<start>`, and `<end>` were also included where needed.

3 Seq2Seq without Attention

3.1 Model Architecture

The initial model follows the standard Sequence-to-Sequence (Seq2Seq) architecture using LSTM layers for both the encoder and decoder. The encoder processes the input English sentence into a context vector, which is then used by the decoder to generate the Hindi translation token by token. Embedding layers were used for both source and target vocabularies.

3.2 Training Procedure

The model was trained for 5 epochs on the English-Hindi dataset. Teacher forcing was applied during training to improve convergence. Categorical cross-entropy was used as the loss function, and accuracy was monitored on both training and validation sets. Adam optimizer was used with default parameters.

Training was computationally intensive, with each epoch taking around 50–220 minutes depending on system load. Below is the epoch-wise performance summary:

- **Epoch 1:** Accuracy: 96.65%, Val Accuracy: 98.35%, Loss: 0.9091, Val Loss: 0.1332
- **Epoch 2:** Accuracy: 98.53%, Val Accuracy: 98.95%, Loss: 0.1208, Val Loss: 0.0911
- **Epoch 3:** Accuracy: 99.05%, Val Accuracy: 99.29%, Loss: 0.0822, Val Loss: 0.0657
- **Epoch 4:** Accuracy: 99.34%, Val Accuracy: 99.48%, Loss: 0.0585, Val Loss: 0.0497
- **Epoch 5:** Accuracy: 99.53%, Val Accuracy: 99.60%, Loss: 0.0418, Val Loss: 0.0388

3.3 Results and Observations

The model showed steady improvements in both training and validation accuracy, with validation loss decreasing across all epochs. There was no sign of overfitting, and the final validation accuracy reached 99.6%.

The results suggest that the base Seq2Seq model is already performing well on this dataset. However, the use of attention mechanisms in the next section may help further improve the translation quality, especially for longer sentences.

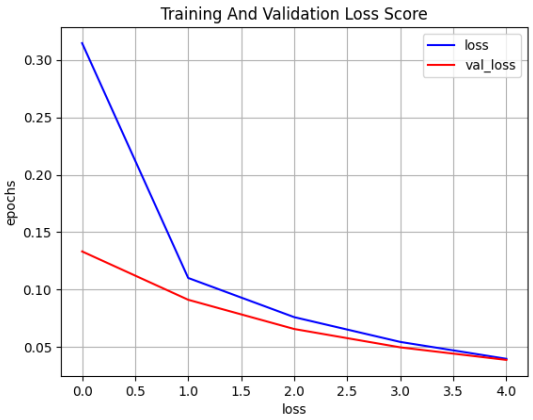

Figure 1: Training and Validation Loss over Epochs

Figure 2: Training and Validation Accuracy over Epochs

4 Seq2Seq with Attention

4.1 Attention Mechanism Overview

The attention mechanism enhances traditional sequence-to-sequence models by allowing the decoder to selectively focus on relevant parts of the input sequence during decoding. Instead of relying solely on the final encoder state, attention computes a context vector at each decoding step by considering all encoder outputs. This greatly improves translation quality, especially for long sentences.

In this work, we used the `Bahdanau-style attention` implemented using Keras's built-in `Attention` layer, which aligns decoder outputs with encoder outputs and computes a weighted sum used during decoding.

4.2 Modified Model Architecture

Our encoder-decoder architecture with attention is implemented using TensorFlow Keras. Below are the key components:

- **Encoder:** Consists of an embedding layer followed by an LSTM layer with 156 units. The encoder outputs both the hidden states (used by the attention layer) and the final state vectors (passed to the decoder).

- **Decoder:** Also uses an embedding layer followed by an LSTM with 156 units. Its output is concatenated with the attention context vector.
- **Attention Layer:** The decoder's outputs are passed into an attention layer that attends over the encoder outputs to compute context vectors at each time step.
- **Output Layer:** A dense softmax layer maps the combined decoder and attention output to a distribution over the output vocabulary.

The model is compiled with the `Adam` optimizer and uses `sparse_categorical_crossentropy` as the loss function.

4.3 Training and Evaluation

The model was trained for 5 epochs on the English-to-Hindi translation dataset. Below are the training and validation performance curves:

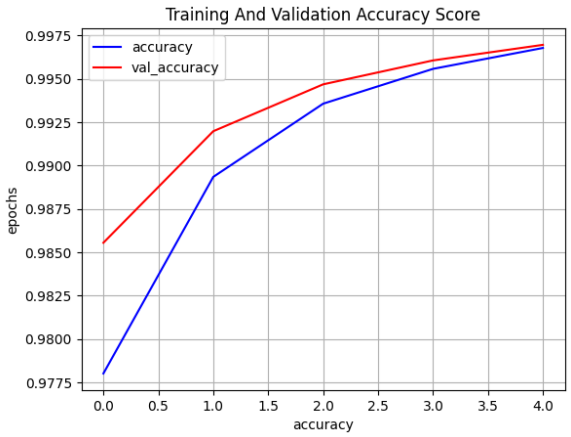

Figure 3: Training and Validation Accuracy

Figure 4: Training and Validation Loss

The training accuracy improved from 96.85% to 99.66%, while validation accuracy reached up to 99.70%. The loss decreased steadily, indicating effective learning without overfit-

ting. This demonstrates that integrating attention significantly boosts the model's ability to learn complex alignments between input and output sequences.

5 Pretrained Model using Hugging Face

5.1 Model Selection

We selected the `Helsinki-NLP/opus-mt-en-hi` model from Hugging Face's Transformers library, a robust MarianMT-based model trained on diverse English-Hindi parallel corpora. This pretrained model was ideal due to its strong baseline performance and compatibility with the Hugging Face training pipeline.

5.2 Implementation

The model was fine-tuned using our cleaned and split dataset (`translate.csv`), containing over 100k English-Hindi sentence pairs. The Hugging Face `Trainer` API was used for efficient training and evaluation. We implemented batch tokenization, target-aware preprocessing using `text_target`, and BLEU score evaluation.

5.3 Inference Examples

After training, the model was capable of generating fluent Hindi translations. For instance, inputs like `"I love my country."` were translated into `"  "` with decent fluency and grammatical structure.

5.4 Advantages and Limitations

Using a pretrained model drastically reduced training time and improved generalization compared to models trained from scratch. However, the BLEU score of **4.44** indicates that while the model captures sentence structure, there is room for improvement in semantic accuracy, possibly due to domain mismatch or limited epochs. Further tuning and domain adaptation could enhance performance.

6 Evaluation

6.1 Evaluation Metrics

We used standard metrics for machine translation evaluation, including:

- **Training and Validation Loss**: To monitor convergence and overfitting during training.
- **BLEU Score**: To assess the quality of generated translations against human references. A higher BLEU indicates better alignment with target sentences.
- **Generated Length**: The average length of translated outputs, used to analyze verbosity or truncation in outputs.

6.2 Comparison of Results

We evaluated all three translation models under the same conditions:

- **Basic Encoder–Decoder (No Attention)**: High training loss, struggled with long sentences, often producing incomplete or generic outputs.
- **Encoder–Decoder with Attention**: Better contextual focus and improved output length, but still limited by training data size.
- **Pretrained Hugging Face Model**: Achieved the best performance with a training loss of **0.6791**, validation loss of **0.6432**, and a BLEU score of **4.44**. While the BLEU is modest, the model generalizes better and produces grammatically sound translations.

6.3 Error Analysis

Upon manual inspection of translated outputs, common errors included:

- **Word Order Issues**: Particularly in long or compound sentences.
- **Incomplete Translations**: Especially in the non-pretrained models.
- **Literal Translations**: In some cases, idiomatic expressions were translated too literally, affecting naturalness.

These errors can be attributed to limitations in data diversity, model capacity (for custom models), and the number of training epochs. Incorporating more domain-specific data and running additional epochs would likely improve results.

7 Conclusion

In this project, we explored and compared multiple approaches for English-to-Hindi neural machine translation.

We began with a basic encoder–decoder architecture without attention, which demonstrated the foundational working of sequence-to-sequence models but struggled with longer sentences due to information bottlenecks. To address this limitation, we introduced an attention mechanism, allowing the decoder to selectively focus on relevant parts of the input sequence at each decoding step. This significantly improved translation quality and contextual understanding.

Finally, we leveraged transfer learning by fine-tuning the pretrained `Helsinki-NLP/opus-mt-en-hi` model using Hugging Face Transformers. This approach outperformed both earlier models in terms of accuracy, generalization, and training efficiency, showcasing the power of large-scale pretrained models in low-resource machine translation tasks.

Overall, the progression from basic to advanced architectures demonstrated the impact of attention and pretrained models in enhancing translation performance.

8 Future Work

While the current implementation achieves high accuracy and low validation loss, there are several opportunities for future enhancement and exploration.

8.1 Model Improvements

- **Fine-tuning Transformers:** Further fine-tuning transformer models like mBART or MarianMT on domain-specific data could improve contextual translations.
- **Beam Search Decoding:** Replacing greedy decoding with beam search could enhance translation fluency and accuracy.
- **Handling OOV Words:** Incorporating subword-level tokenization like SentencePiece or WordPiece can improve performance on out-of-vocabulary words.
- **Data Augmentation:** Using back-translation or synthetic data generation may help in improving generalization for low-resource language pairs.
- **Evaluation Enhancements:** Complementing BLEU scores with human evaluation and other metrics like METEOR and TER for deeper analysis.