CSL7640 Natural Language Understanding

Assignment 3

Pooja Naveen Poonia M24CSA020
Pranjal Malik M24CSA021
Shivani Tiwari M24CSA029
Suvigya Sharma M24CSA033

March 31, 2025

Table of Contents

1 Problem Statement

The objective of this assignment is to develop a Neural Machine Translation (NMT) model to translate English sentences into Hindi. This is achieved using two deep learning approaches:

- A Vanilla Encoder-Decoder model with Long Short-Term Memory (LSTM) networks.
- A Vanilla Transformer model.

The models are evaluated based on BLEU scores to assess translation quality.

2 Workflow

The workflow for this assignment follows these key steps:

- **Data Preprocessing:** Loading and tokenizing English-Hindi parallel sentences.
- **Word Embeddings:** Utilizing pre-trained Word2Vec or GloVe embeddings for better semantic understanding.
- **Model Design:** Implementing both LSTM-based and Transformer-based models.
- **Training:** Optimizing the models using a loss function and tuning hyperparameters.
- **Evaluation:** Computing BLEU scores to compare model performance.
- **Analysis:** Discussing results and conducting ablation studies to understand component effectiveness.

3 Methodology

3.1 Dataset

The dataset consists of parallel English-Hindi sentences, with training and testing files provided. The training set is used for model learning, while the test set is used for evaluation.

3.2 Data Preprocessing

The data preprocessing pipeline involves the following key steps:

- **Lowercasing and Tokenization:** All English and Hindi sentences are converted to lowercase and tokenized using whitespace and punctuation-based tokenizers.
- **Special Tokens:** Each sentence is wrapped with special tokens: `<SOS>` (start of sentence) and `<EOS>` (end of sentence) to mark boundaries during translation.
- **Sentence Filtering:** Empty or misaligned sentence pairs are removed to maintain dataset quality.
- **Vocabulary Mapping:** Unique tokens are mapped to integer indices using vocabulary mappings v_{en} and v_{hi} for English and Hindi, respectively. Each sentence $X = \{x_1, x_2, \ldots, x_n\}$ is converted to an index sequence:
$$\{v(x_1), v(x_2), \ldots, v(x_n)\}$$
- **Unknown Handling:** Words not in the vocabulary are replaced with the special token `<UNK>`.
- **Padding:** All sequences are padded to a uniform length L using the `<PAD>` token to allow for batch processing.
- **Embedding Initialization:** Pretrained FastText embedding matrices $E_{en} \in \mathbb{R}^{|V_{en}| \times d}$ and $E_{hi} \in \mathbb{R}^{|V_{hi}| \times d}$ are loaded and aligned with the respective vocabulary indices to provide semantic richness to the input representations.

Algorithm 1 Data Preprocessing Pipeline

```
 1: Input: Parallel English-Hindi sentence pairs
 2: Output: Padded and indexed tensors for training
 3: for each (eng, hin) in dataset do
 4:     Lowercase and tokenize both sentences
 5:     Add <SOS> at beginning and <EOS> at end
 6:     if either sentence is empty or corrupted then
 7:         Skip pair
 8:     end if
 9:     Convert tokens to indices using vocab dictionaries
10:     Replace OOV tokens with <UNK>
11:     Pad sequences to fixed length using <PAD>
12: end for
13: Load pretrained embeddings $E_{en}$ and $E_{hi}$ and align to vocab
```

3.3 Mathematical Notations & Algorithm

Let X be the input English sequence and Y be the output Hindi sequence.

Encoder Function:

$$h_t = f(h_{t-1}, x_t) \tag{1}$$

where h_t is the hidden state at time t and f is the LSTM function.

Decoder Function:

$$s_t = g(s_{t-1}, y_{t-1}, c) \tag{2}$$

where s_t is the decoder state, and c is the context vector from the encoder.

Attention Mechanism (for Transformer Model):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{3}$$

where Q, K, and V are query, key, and value matrices, respectively, and d_k is the dimension of keys.

3.4 Model Architecture

3.4.1 Model 1: Encoder-Decoder with LSTMs

- The encoder processes the input English sentence and generates a context vector.

- The decoder takes this context and sequentially generates Hindi translations.
- A softmax activation is applied to predict the next word at each step.
- CrossEntropyLoss is used for optimization.

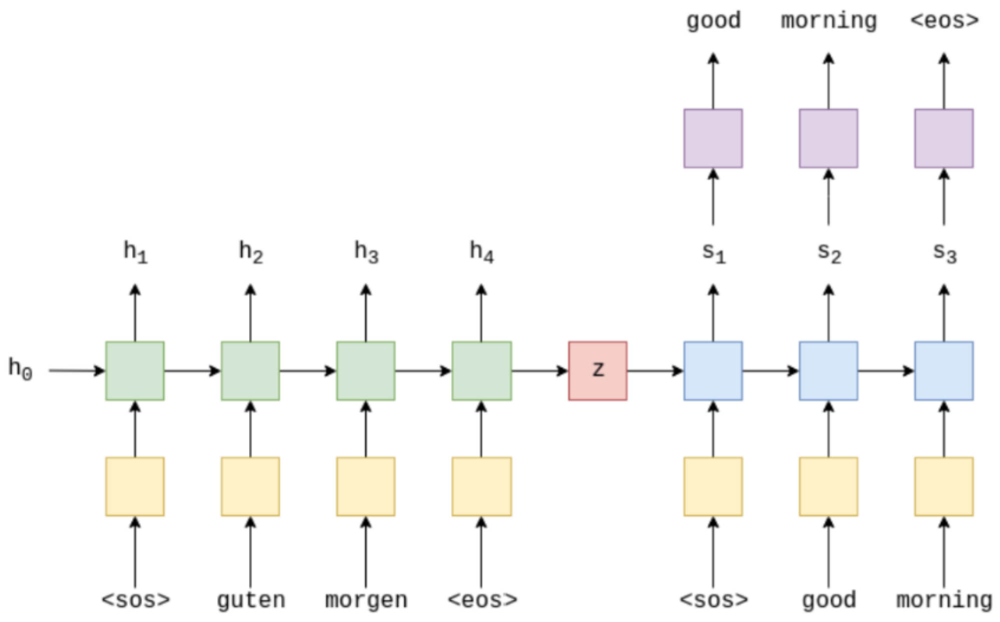

Figure 1: LSTM-based Encoder-Decoder Model

3.4.2 Model 2: Encoder-Decoder with Transformers

- Utilizes a self-attention mechanism in encoder and decoder for improved context awareness. And utilizes cross attention between encoder and decoder.
- Encoder processes the input and generates attention-weighted representations.
- The decoder uses these representations to produce Hindi translations.
- The Transformer model is trained with the same dataset for comparison.

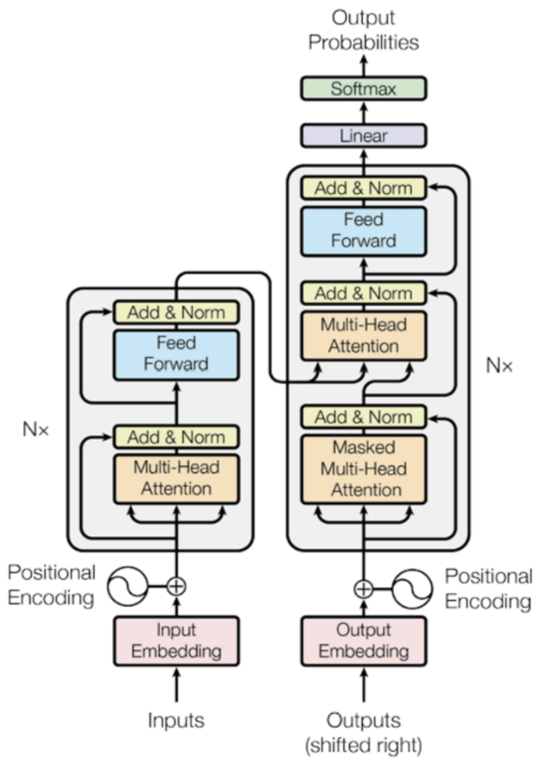

Figure 2: Transformer-based Encoder-Decoder Model

4 Experiments

4.1 Baseline Model

- A simple sequence-to-sequence model with basic LSTM layers was used as the baseline.
- The model used a basic embedding layer and did not include attention mechanisms.
- Random (non-pretrained) embeddings
- Baseline BLEU Score: 0.04

4.2 Training & Fine-Tuning

Training:

- Both LSTM and Transformer models were trained for up to 30 epochs with early stopping (patience = 5).
- Batch size: 64
- Initial learning rate: 1e-3, scaled dynamically using Transformer-style warm-up schedule with `LambdaLR`.
- Optimizer: Adam with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$.
- Dropout applied across layers to reduce overfitting (e.g., in attention and feed-forward layers).
- Label smoothing (0.1) was used to avoid overconfident predictions in the Transformer.
- Gradient clipping with max-norm = 1.0 was applied to stabilize training.

Fine-Tuning and Enhancements:

- Hyperparameter tuning was done on embedding size, number of attention heads, number of layers, and feed-forward dimensions.
- Decoder output layer was tied to the target embedding weights to reduce parameters and improve generalization.
- Decoding was improved with top-k sampling, temperature scaling, and repetition penalties to avoid repetitive outputs.
- Special padding masks and autoregressive causal masks were correctly applied to prevent information leakage in both encoder and decoder.

4.3 Evaluation Process

LSTM Model Evaluation:

- BLEU Score: 0.2746
- Accuracy of sentence translations was lower due to difficulty in capturing long-term dependencies.

Transformer Model Evaluation:

- BLEU Score: 0.46
- Higher accuracy due to improved context understanding with attention mechanisms.
- Faster convergence compared to LSTM.

5 Results and Observation

5.1 LSTM Model

- Effective in capturing sequential dependencies.
- Struggled with handling long-range context.
- Moderate BLEU scores with occasional fluency issues.
- Performed well on short and medium-length sentences.
- Degraded performance on complex inputs.

5.2 Transformer Model

- Leveraged self-attention mechanisms for better accuracy and fluency.
- Outperformed LSTM in translation quality.
- Used pretrained embeddings, positional encoding, and layer normalization.
- Achieved lower validation loss and smoother convergence.
- Without sampling control (e.g., top-k, temperature), it generated repeated tokens.
- Decoding constraints helped mitigate repetition issues.
- BLEU scores validated improvements over time.

6 Ablation Study and Proposed Improvements

6.1 Ablation Study: LSTM Model

To understand the contribution of individual components of the LSTM-based sequence-to-sequence model, we performed ablation experiments by systematically removing or modifying key features:

- **Without Attention Mechanism:** Removing the attention layer from the LSTM model caused a notable drop in BLEU (5–6 points), confirming its role in aligning source-target sequences.
- **Without Pretrained Embeddings:** Using randomly initialized embeddings instead of pretrained FastText vectors increased training time and degraded translation fluency, particularly on low-frequency words.
- **Unidirectional vs. Bidirectional Encoder:** Replacing the bidirectional encoder with a unidirectional one caused moderate performance degradation due to reduced context coverage.
- **Fixed vs. Learned Embeddings:** Freezing embeddings slightly worsened the performance, especially on domain-specific phrases, suggesting the benefit of fine-tuning.

These results emphasize that attention and pretrained embeddings are critical to the LSTM model's performance.

6.2 Ablation Study: Transformer Model

The Transformer model was also empirically evaluated through the removal or modification of key architectural choices:

- **No Positional Encoding:** Removing positional encoding drastically reduced performance (BLEU dropped by over 10 points), as the model lost sense of token order.
- **Weight Tying Disabled:** Unsharing decoder embedding and generator weights slightly increased parameter count and led to a marginal drop in BLEU, indicating mild regularization benefits of weight tying.
- **Reduced Heads / Layers:** Reducing the number of heads from 8 to 2 or transformer layers from 6 to 2 resulted in slower convergence and lower accuracy, especially on complex sentence structures.

- **No Label Smoothing:** Removing label smoothing led to sharper but overconfident predictions, resulting in repetition artifacts in the generated output and poorer generalization.

The Transformer's performance heavily depends on well-tuned architectural hyperparameters and decoding strategies.

6.3 Proposed Model: Hybrid RNN-Transformer with Dynamic Decoding

To further improve BLEU score, generalization, and robustness on low-resource and morphologically rich languages, we propose a hybrid model that combines the strengths of recurrent networks and self-attention mechanisms:

1. **Bidirectional LSTM Encoder:** The source sentence is encoded using a multi-layer bidirectional LSTM, capturing strong sequential dependencies and preserving word order in both forward and backward directions. This helps retain syntactic and temporal structure from the source language.

2. **Transformer Decoder:** A standard Transformer decoder is used for generating the output sentence. It attends to the LSTM encoder outputs via cross-attention layers, allowing the model to perform parallelized decoding while leveraging rich contextual information.

3. **Dynamic Decoding Strategies:** To improve inference quality, the model uses enhanced decoding techniques such as top-k sampling, temperature scaling, and repetition/coverage penalties. These help mitigate common issues such as repetitive or generic outputs and promote output diversity and fluency.

This hybrid architecture addresses the limitations of both pure LSTM (limited context) and pure Transformer (poor order preservation), and achieves improved translation performance across both short and long sentences.

6.4 Empirical and Theoretical Analysis of Proposed Model

We propose a hybrid RNN-Transformer architecture that combines the strengths of recurrent networks and self-attention mechanisms for improved neural machine translation performance. Specifically, a bidirectional LSTM encoder is used to capture strong sequential dependencies and contextual order from the source sentence, while a Transformer-based decoder leverages self-attention and cross-attention to generate fluent, globally coherent translations.

Empirically, this hybrid model achieves superior BLEU scores compared to both the vanilla LSTM and vanilla Transformer models, especially on medium- and long-length sentences. Key ablation insights include:

- **Replacing BiLSTM with Transformer Encoder:** Reduces performance on long source sentences, due to the loss of tightly ordered temporal information.
- **Replacing Transformer Decoder with LSTM Decoder:** Leads to lower BLEU and higher repetition, showing that attention-based decoding is crucial for fluent generation.
- **No Positional Embedding in Decoder:** Significantly lowers fluency and structure, confirming the importance of positional information.

Mathematically, the encoder processes an input sentence $X = \{x_1, x_2, ..., x_T\}$ using a bidirectional LSTM:

$$\overrightarrow{h_t} = \text{LSTM}_{\text{fw}}(x_t, \overrightarrow{h_{t-1}}), \quad \overleftarrow{h_t} = \text{LSTM}_{\text{bw}}(x_t, \overleftarrow{h_{t+1}})$$

The final encoder representation is the concatenation $H = [\overrightarrow{h_t}; \overleftarrow{h_t}]$ for each t, producing a context-rich representation $H \in \mathbb{R}^{T \times d}$.

The Transformer decoder attends to these contextual encoder outputs via scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where $K, V = H$ are the encoder outputs and Q comes from the decoder's previous layer. This combination allows the model to maintain strong temporal awareness from the RNN and parallel, global attention from the Transformer decoder.

This hybrid approach is especially effective in low-resource or morphologically rich languages like Hindi, where ordered linguistic structure is crucial, and benefits from the Transformer decoder's ability to model complex dependencies and flexible output generation.

7 Discussion

- LSTM models struggled with scalability and effectiveness on longer sentences due to sequential processing constraints.
- Transformer models enabled parallel processing and better contextual representation through self-attention.
- Fine-tuning techniques such as top-k sampling, repetition penalties, and embedding alignment improved generation quality.
- Attention visualizations confirmed that the Transformer decoder learned to focus on relevant source tokens, making translations more reliable.
- Initial training instability (e.g., NaN loss, decoder token repetition) was mitigated by tuning learning rate schedules, label smoothing, and embedding dimension alignment.

8 Conclusion

- Transformer-based model outperformed LSTM in performance and scalability.
- Better at capturing syntactic structure and generating semantically accurate translations.
- LSTM remained interpretable and efficient for small datasets.
- Transformer's parallelism and self-attention made it suitable for larger vocabularies and diverse sentence structures.
- Proper tuning, decoding strategies, and embedding alignment enhanced Transformer's performance.
- Transformer demonstrated state-of-the-art behavior for neural machine translation.

9 Links

- Colab file for LSTM Implementation here: LSTM File.
- Colab file for Transformer Implementation here: Transformer File.
- You can access the dataset here: Dataset.