CS779 Competition: Machine Translation System for India

Pranjali Singh
220796
pranjalis22@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

This competition focused on building multilingual Neural Machine Translation (NMT) systems for English–Hindi and English–Bengali language pairs under a low-resource setting. Our best model, a Transformer-based encoder–decoder with 4 encoder and 4 decoder layers, achieved a **chrF++ score of 0.413**, **ROUGE score of 0.471**, and **BLEU score of 0.212**, ranking **1st** on the final leaderboard. Key improvements included subword tokenization using BPE, label smoothing, and beam search decoding with repetition and length penalties. The multilingual setup with language tags enabled shared learning across Indic scripts, resulting in significant gains over recurrent architectures.

1 Competition Result

Codalab Username: p_220796
Final leaderboard rank on the test set: 1
charF++ Score wrt to the final rank: 0.413
ROGUE Score wrt to the final rank: 0.471
BLEU Score wrt to the final rank: 0.212
Total Number of Submissions in the Training Phase: 18
Total Number of Submissions in the Testing Phase: 6
Table showing the Number of Submissions made each Week during the Training Phase:

Week	Number of Submissions
Week 1	0
Week 2	8
Week 3	4
Week 4	6

2 Problem Description

2.1 Overview

This project is about multilingual Neural Machine Translation (NMT) from English to two major Indian languages: Hindi and Bengali. The objective is to develop a machine translation system that can automatically translate English text into the target Indian languages.

2.2 Problem Setting

Given a source sentence in English, the task is to generate its corresponding translation in either Hindi (Devanagari script) or Bengali (Bengali script). This is a sequence-to-sequence learning problem where:

- **Input:** English sentence (source language)

- **Output:** Translated sentence in Hindi or Bengali (target language)
- **Constraint:** The model must be trained from scratch without using pre-trained models.
- **Allowed resources:** Only static word embeddings (Word2Vec, GloVe, FastText) are permitted.

The dataset presents several challenges:

- **Multilingual complexity:** Supporting two target languages with distinct scripts and grammatical structures
- **Cross-lingual semantic alignment:** Preserving meaning across languages with different word orders (English: SVO vs Hindi/Bengali: SOV)
- **Morphological richness:** Indian languages are morphologically richer than English, requiring the model to handle complex word formations
- **Script diversity:** Devanagari (Hindi) and Bengali scripts require proper Unicode handling

3 Data Analysis

3.1 Training Dataset Description

The training corpus comprises **149,646 parallel sentence pairs** for English to Indian language translation, distributed across two target languages:

- **Hindi**: 80,797 samples (54.0%)
- **Bengali**: 68,849 samples (46.0%)

This constitutes a **low-resource NMT scenario** with fewer than 100K samples per language pair, presenting challenges in generalization and requiring careful regularization strategies.

3.2 Corpus Statistics

3.2.1 Sentence Length Analysis

Table 1 reveals striking differences in translation characteristics between Hindi and Bengali.

Table 1: Sentence Length Statistics (Training Data)

Language	Mean	Median	Std
English	16.96	16.0	8.96
Hindi	18.93	17.0	10.75
Bengali	14.27	13.0	7.21

Critical Finding: Hindi translations are consistently **13.2% longer** than English source, while Bengali translations are **12.5% shorter**.

3.2.2 Vocabulary Analysis

Table 2: Vocabulary Statistics by Language

Language	Unique Words	Total Tokens	Tokens/Type
English	69,833	2,476,893	35.5
Hindi	12,125	2,717,637	**224.1**
Bengali	11,841	2,318,886	**195.8**

Key Observation: Despite nearly identical vocabulary sizes (12,125 vs 11,841), Hindi exhibits:

- **17.2% more tokens** than Bengali (2.72M vs 2.32M)
- **14.5% higher type-token ratio** (224.1 vs 195.8)

This suggests Hindi text in our corpus has higher lexical repetition. The English vocabulary is **5.76× larger** than Hindi/Bengali combined, reflecting the agglutinative nature of Indic languages where morphological complexity is encoded through word formation rather than separate tokens.

Table 3: Noise Analysis in Training Corpus

Issue Type	Count	Percentage
Duplicate English sentences	27,082	**18.10%**
Duplicate target sentences	2,206	1.47%
Very short English (<3 words)	1,202	0.80%
Very short Indian (<3 words)	1,043	0.70%
Very long English (>100 words)	10	0.01%
Very long Indian (>100 words)	22	0.01%
Unusual characters (English)	9,666	6.46%
Unusual characters (Hindi/Bengali)	19,520	13.04%

3.3 Test Dataset Analysis

The test set contains **42,757 samples** (28.6% of training size) with identical language distribution (54% Hindi, 46% Bengali), ensuring representative evaluation.

Table 4: Training vs Test Comparison

Metric	Training	Test
Sample count	149,646	42,757
Avg sentence length (words)	16.96	17.01
English vocabulary	69,833	40,634
Vocab overlap	**100%**	**83.8%**

The **83.8% vocabulary overlap** indicates that **16.2% of test vocabulary is OOV**, necessitating robust subword tokenization.

4 Model Description

4.1 Model Evolution

The model evolved iteratively across three weeks , with major architectural transitions and corresponding insights summarized below. The development followed an empirical approach, starting from recurrent

architectures and transitioning to attention-based models.

4.1.1 Week 1: Baseline LSTM with Attention

Implemented a 2-layer unidirectional LSTM encoder-decoder with Bahdanau attention mechanism [1]. The encoder(2 layers) processed English input sequentially, and the decoder(1 layer) generated Hindi/Bengali translations token-by-token with attention over encoder hidden states. Separate models were trained for Hindi and Bengali.
Key Learning: The encoder only captured left-to-right context, missing bidirectional information important for morphologically rich languages like Hindi and Bengali.

4.1.2 Week 2: Bidirectional LSTM Exploration

The bidirectional LSTM encoder captures both past and future context. This architecture is particularly beneficial for morphologically rich languages [2]. A bidirectional decoder was not used, as it would increase model complexity while providing minimal improvement in performance, given that the encoder plays the primary role in capturing context. A deeper architecture with a 2-layer bidirectional encoder and a 1-layer decoder was also tested, under the hypothesis that increased depth would capture more complex syntactic patterns necessary for cross-lingual translation.

Key Learning: The 2-layer encoder configuration degraded performance despite having more parameters. Analysis revealed slower convergence, likely due to increased model complexity/gradient vanishing without sufficient regularization or maybe a fault in my implementation.

Critical Realization: RNN-based architectures are fundamentally limited by sequential processing and vanishing gradients. The marginal gains from BiLSTM enhancements did not justify the added architectural complexity.

4.1.3 Week 3: Transformer Exploration

Motivated by the limitations of recurrent architectures, the RNN-based model was replaced entirely with a **Transformer encoder-decoder** [3]. This architecture leverages self-attention mechanisms to model global dependencies without recurrence, enabling parallel processing across the sequence. Weight tying, curriculum learning were also tried upon.

Key Learnings: Subword tokenization reduced vocabulary by 6× while significantly improving rare word handling, essential for morphologically rich Indic languages. The single multilingual model leveraged cross-lingual patterns and shared representations, improving performance on both Hindi and Bengali compared to separate models. Beam search performs better than greedy search.

Performance: This architecture achieved the **best results** on the Codabench leaderboard, with substantial improvements over RNN-based models, particularly on longer sequences and rare word translation.

4.2 Final Model Architecture

4.2.1 Model Specifications

Component	Configuration
Encoder Layers	4
Decoder Layers	4
Model Dimension (d_{model})	512
Attention Heads	8
Feed-Forward Dimension (d_{ff})	2048
Dropout Rate	0.1
Activation Function	GELU
Max Sequence Length	512
Source Vocabulary Size	8,792
Target Vocabulary Size	13,286 (includes language tags)
Total Parameters	$\sim$47.5M

Table 5: Transformer Model Configuration

4.2.2 Language Control Mechanism

We adopted a language tag prefix strategy where the target language is specified via a special token at decoder start: [⟨s⟩, ⟨2hi⟩, ...] or [⟨s⟩, ⟨2bn⟩, ...]. This approach, inspired by Johnson et al. [1], allows a single shared decoder to handle both Hindi and Bengali without mixing language tags in the training data text

4.2.3 Model Objective(Loss) Functions

The model is trained using cross-entropy loss with label smoothing [4] to prevent overconfidence:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{|V|} \tilde{y}_i \log(p_i) \tag{1}$$

where the smoothed labels are:

$$\tilde{y}_i = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|V|} & \text{if } i = y^* \\ \frac{\epsilon}{|V|} & \text{otherwise} \end{cases} \tag{2}$$

Parameters:

- Label smoothing: $\epsilon = 0.1$ (optimal value found through experimentation)
- Padding index: ignored in loss computation

Motivation: Label smoothing reduces overfitting and improves generalization, particularly important for low-resource languages like Hindi and Bengali [5].

4.2.4 Inference: Decoding Strategies

The project explored two decoding strategies, evolving from simple to more sophisticated methods:

1. Greedy Decoding (Initial Baseline)

Algorithm:

$$y_t = \arg\max_{w \in V} P(w \mid y_{<t}, x) \tag{3}$$

2. Beam Search (Primary Method)

Algorithm: Maintains k most probable partial hypotheses at each step [6].

Repetition Penalty: To reduce repetitive outputs (common in Indic language generation):

$$\log P(w_t) \leftarrow \log P(w_t) - \log(\gamma) \cdot \delta(w_t \in y_{<t}) \tag{4}$$

where $\gamma = 1.8$ and δ is an indicator function.

5 Experiments

5.1 Data Preprocessing

5.1.1 Tokenization

The model employs a custom **SentencePiece-style Byte Pair Encoding (BPE)** tokenizer [7, 8]:

- Source (English): 32,000 BPE merges → 8,792 tokens
- Target (Hindi + Bengali): 50,000 BPE merges → 13,284 tokens + 2 language tags
- **Normalization:** Unicode NFKC normalization
- **Word Boundary Marker:** (U+2581) prefix for subword units
- **Special Tokens:** `<pad>`, `<unk>`, `<s>`, `</s>`

Rationale: BPE reduces vocabulary size while handling rare words through subword composition [7]. This is crucial for low-resource settings where OOV (out-of-vocabulary) words are common.

5.2 Training Procedure

Table 6: Training Configuration

Parameter	Value
Optimizer	Adam [9]
β_1, β_2	0.9, 0.98
ϵ	10^{-9}
Learning Rate	Noam Scheduler [3]
Initial LR	0 (warmup from 0)
Warmup Steps	4,000
Peak LR	$\sim 10^{-3}$ (at step 4000)
Formula	$lr = d_{\text{model}}^{-0.5} \cdot \min(\text{step}^{-0.5}, \text{step} \cdot 4000^{-1.5})$
Regularization	
Dropout	0.1
Gradient Clipping	1.0 (max norm)
Training	
Batch Size	64
Epochs	15/25 (4+4)
Mixed Precision	FP16 with AMP
Gradient Accumulation	No

5.3 Hyperparameter Selection

5.3.1 Hyperparameter Tuning Strategy

1. **Architecture:** Started with Transformer-Base (4+4) from [3]
2. **Dropout:** Used standard 0.1 (no tuning needed for base config)
3. **Label Smoothing:** Set to 0.1 based on [5] (optimal for NMT)
4. **Learning Rate:** Noam scheduler with warmup=4000
5. **Beam Size:** Beam size = 5 (balance between time and accuracy)
6. **Length Penalty:** Used $\alpha = 0.6$
7. **Repetition Penalty:** $\gamma = 1.8$
8. **Maximum Sequence Length:** 85 tokens

6 Results

I took part in development phase.

Table 7: Experimental Result on Development Phase

Model	Decoding	chrF++	ROUGE-L	BLEU
Transformer (4 Encoder + 4 Decoder)	Beam	0.39	0.45	0.192
LSTM with Attention	Beam	0.191	0.311	0.046
Bi-LSTM	Beam	0.207	0.332	0.061
Stacked Bi-LSTM	Beam	0.137	0.294	0.04

Due to space constraints, I cannot put the full table here.

Table 8: Experimental Results on Test Data: Model Variations and Decoding Strategies

Model	Decoding	Date	Score ID	chrF++	ROUGE-L	BLEU
Transformer (4 Encoder + 4 Decoder)						
4+4 Base	Greedy	2025-11-05	414068	0.393	0.457	0.194
4+4 Base	Beam Search	2025-11-05	414165	**0.413**	**0.471**	**0.212**
Transformer Deep Encoder (7 Encoder + 3 Decoder) with additions						
7+3 Complex (more Epochs)	Greedy	2025-11-06	415088	0.426	0.452	0.183
7+3 Complex	Beam Search	2025-11-06	415292	0.413	0.448	0.189
Transformer Deep Encoder (7 Encoder + 3 Decoder) with additions						
7+3 Complex	Greedy	2025-11-06	415327	0.412	0.437	0.173
7+3 Complex(more Epochs)	Beam Search	2025-11-06	416131	0.402	0.452	0.204

7 Conclusion

7.1 Key Findings

This work developed a Transformer-based multilingual NMT system for English→Hindi/Bengali translation. Key results:

1. **Best Model:** 4E+4D Transformer (47.5M parameters) with beam search achieved **BLEU=0.413, chrF++=0.471, ROUGE-L=0.212**.
2. **Decoding Critical:** Beam search provided **+5.1% BLEU** over greedy with 0% output agreement, proving decoding strategy is as important as architecture.
3. **Major Challenges:** Repetition loops (25% → 5% with penalties), technical term errors, and exposure bias during training.

7.2 Future Directions

Short-term: Add copy mechanism for proper nouns; back-translation for data augmentation
Long-term: Leverage pre-trained models (mBART, IndicBART); extend to other Indian languages

References

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations (ICLR)*, (San Diego, CA, USA), 2015.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, (Montreal, Canada), pp. 3104–3112, 2014.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NIPS)*, (Long Beach, CA, USA), pp. 5998–6008, 2017.

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, NV, USA), pp. 2818–2826, 2016.

[5] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?," in *Advances in Neural Information Processing Systems (NeurIPS)*, (Vancouver, Canada), pp. 4696–4705, 2019.

[6] M. Freitag and Y. Al-Onaizan, "Beam search strategies for neural machine translation," in *Proceedings of the First Workshop on Neural Machine Translation*, (Vancouver, Canada), pp. 56–60, 2017.

[7] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, (Berlin, Germany), pp. 1715–1725, 2016.

[8] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent approach to subword tokenization," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Brussels, Belgium), pp. 66–71, 2018.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, 2015.