CS779 Competition: Machine Translation System for India

Deepak Chaurasia
220330.
deepakc22@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

The work described in this paper is focused on the implementation of neural machine translation systems for the English-Hindi and English-Bengali pairs using Seq2Seq, BiLSTM with Attention, and Transformer architectures. It achieved notable improvements by systematically experimenting with data augmentation, sub-word tokenization(bpe), and mechanisms of attention. The best-performing 4-layer Transformer with 8 attention heads achieved a score of 0.32 on the validation set and 0.29 on the test set, ranking among the top submissions. These results highlight that high-quality data, effective regularization, and attention-based architectures significantly enhance translation performance for low-resource Indic languages.

1 Competition Result

Codalab Username: d_220330
Final Leaderboard Rank on the Test Set: 78
Final Leaderboard Rank on the Train phase: 67
chrF++ Score corresponding to the Final Rank: 0.319
ROUGE Score corresponding to the Final Rank: 0.317
BLEU Score corresponding to the Final Rank: 0.073
Total Number of Submissions in the Training Phase: 16
Total Number of Submissions in the Testing Phase: 5

Week	Number of submissions
Week 1	0(was relaxed to 2 submission only)
Week 2	2
Week 3	6
Week 4	4
Week 5	4
Total Submissions	**16**

Table 1: Weekly submission statistics during the training phase (Oct 5 – Nov 4, 2025).

2 Problem Description

This project, in particular, focuses on building an NMT system for translations from English to Hindi and Bengali, which are low-resource and morphologically rich languages. This task is framed as a sequence-to-sequence(model) learning problem where the model learns to map English input sequences to target-language outputs.

Classic statistical or rule-based translation systems face difficulties in handling such languages regarding context, grammar, and long-range dependencies. Hence, this work explores the use of deep

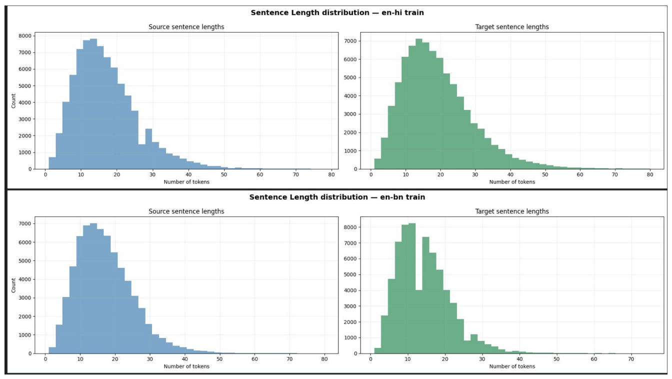


Figure 1: Sentence length distribution for train data

learning-based architectures such as Vanila LSTM,BiLSTM with Attention and Transformer models to increase translation accuracy.

The objective is to develop a translation framework that minimizes translation loss and improves BLEU scores, thus yielding grammatically and semantically coherent translations for Indian languages.

3 Data Analysis

1. **Train datasets**.

   English-Bengalis: There are 68849 rows of source(English)-target(Bengali) pairs of sentences.

   English-Hindi: There are 80797 rows of source(English)-target(Hindi) pairs of sentences.

2. **Corpus stats**:

   English-Bengali:

   Source length stats: 'mean': 16.86, 'median': 16.0, 'p95': 32.0, 'max': 80

   Target length stats: 'mean': 14.34, 'median': 13.0, 'p95': 27.0, 'max': 75

   Above stats clearly shows that the source and target sentences in English and Bengali/Hindi, respectively, are balanced and structurally consistent. The sentence length averages and medians for both sides are very close: English 16.9 / 16.0, Target 14.3 / 13.0, which shows that the majority of translation pairs keep roughly the same level of detail, without noticeable compression or expansion. The values at the 95th percentile(p95) are 32 versus 27, and the maximum lengths are 80 versus 75, which suggests that even longer sentences are proportionally matched, with few outliers.

   The average source-to-target length ratio is 1.20, with a median of 1.17, and it lies within the optimal range of 0.8–1.5, indicating well-aligned bilingual pairs with a balanced linguistic density. No script inconsistencies or empty samples were detected, confirming that all English segments use Latin characters while all target sentences are written in their respective Indic scripts. This ensures clean tokenization and minimizes cross-script noise during model training.

   On the whole, its shows overall data is balanced and well to avoid major pre processing.

   English-Hindi:

   Source length stats: 'mean': 17.12, 'median': 16.0, 'p95': 34.0, 'max': 78

   Target length stats: 'mean': 18.96, 'median': 17.0, 'p95': 38.0, 'max': 80

   Even for English-Hindi,similar case except the different numbers for mean,median and max.

3. **Validation datasets**(during the training phase):

   English-Bengalis: There are 9836 rows of source(English) sentences and we have to predict target columns corresponding to it.

   Source length stats: 'mean': 16.93, 'median': 16.0, 'p95': 32.0, 'max': 99

   English-Hindi: There are 11543 rows of source(English) sentences and we have to predict target colums corresponding to it.

   Source length stats: 'mean': 17.08, 'median': 16.0, 'p95': 34.0, 'max': 155

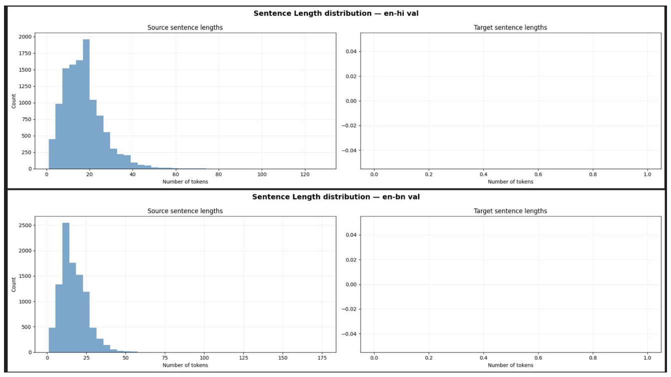


Figure 2: Sentence length distribution for val data

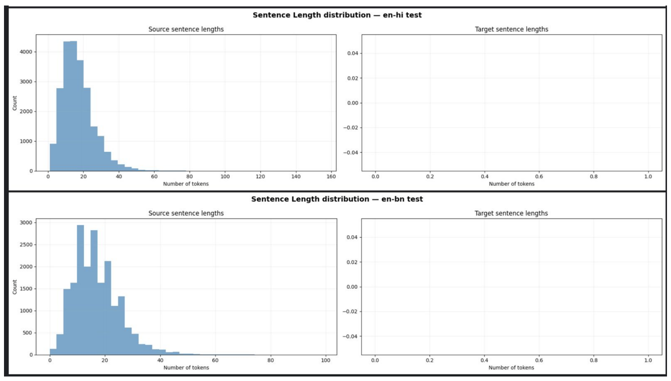


Figure 3: Sentence length distribution for test data(Target is vacant for val and test data)

4. **Test data**(during test phase):

   English-Bengalis: There are 19672 rows of source(English) sentences and we have to predict target columns corresponding to it.

   Source length stats: 'mean': 16.93, 'median': 16.0, 'p95': 32.0, 'max': 99

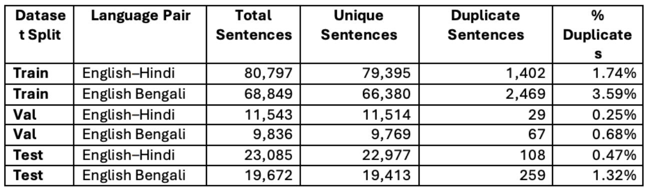
   English-Hindi: There are 23085 rows of source(English) sentences and we have to predict target colums corresponding to it.

   Source length stats: 'mean': 17.08, 'median': 16.0, 'p95': 34.0, 'max': 155

   The overall statistical analysis of the corpus clearly shows that the dataset is clean and well-balanced. For both language pairs, namely, English–Hindi and English–Bengali, the sentence length mean for each category falls approximately in the range of 14.5 to 17.5 tokens, with a similar distribution pattern of median and percentiles. This may suggest that both source and target languages are quite well-aligned with each other and do not contain major noise or any irregularities. The comparable lengths of sentences indicate there would not be any serious truncation issues or large alignment mismatches. Hence, the maximum sequence lengths set between 30 and 40 will easily catch most of the sentences without information loss.

   These findings confirm that the dataset does not require any aggressive filtering or complex preprocessing. Only minimal text normalization steps will be necessary: fixing punctuation spacing, lowercasing English text, trimming extra whitespace, and standardizing quotation marks. Indic languages like Hindi and Bengali should be in their original scripts without normalization. This cleaning will prepare the corpus for model training and learning.

5. Some interesting insights about the datasets(Train,Val,Test).



Datase t Split	Language Pair	Total Sentences	Unique Sentences	Duplicate Sentences	% Duplicate s
Train	English–Hindi	80,797	79,395	1,402	1.74%
Train	English Bengali	68,849	66,380	2,469	3.59%
Val	English–Hindi	11,543	11,514	29	0.25%
Val	English Bengali	9,836	9,769	67	0.68%
Test	English–Hindi	23,085	22,977	108	0.47%
Test	English Bengali	19,672	19,413	259	1.32%

Figure 4: Data Analysis

Overall Data sets is balanced,clean, diverse, and well-structured, only 1-3 % data is duplicate so its not major chunk of data .Small fraction is duplicate data may be that is reason of not getting very good result even after trying with Transformer and deep BiLSTM.

4 Model Description

Table 2: Model Performance on val data during Training phase

Seq No.	Model Name	Score
1	Baseline Seq2Seq (GRU)	0.08
2	LSTM	0.16
3	1 Layer BiLSTM	0.18
4	2 Layer BiLSTM	0.22
5	3 Layer BiLSTM + Attention + Dropout(0.3)	0.23
6	2 Layer BiLSTM + Attention + Dropout(0.2) + 20 epochs	0.20
7	3 Layer BiLSTM + Attention + Dropout(0.3) + 30 epochs	0.21
8	BiLSTM (4 Layers, Dropout = 0.3, overfit)	0.15
9	2 Layer LSTM + Attention + 15 epochs + beam search k=5	0.15
10	BiLSTM(2 layer) + Attention + Dropout(0.3) + Data Augmentation + epoch = 20	0.23
11	BiLSTM(2 layer) + Attention + Dropout(0.1) + Data Augmentation + epoch=30	0.28
12	Transformer (4-Blocks in Encoder and Decoder) + Data Augmentation epoch=20	0.32
13	Transformer (5-Blocks in Encoder and Decoder) + Data Augmentation epoch=20	0.28
14	BiLSTM + Attention (different parameter) + Data Augmentation	0.25
15	BiLSTM + Self Attention Encoder + Data Augmentation	0.26
16	Transformer (6-layer,Dropout=0.1, overfit,epoch=20)	0.20

Best 3 Model Description:

1.BiLSTM(2 layer) + Attention + Dropout(0.1) + Data Augmentation + epoch=30
As I have explained in Experiment may be due to less data my model was not performing well because its 2 layer stacked BiLSTM Model with high parameter(hidden size=512,Dropout 0.1,tokenizer=bpe,num layer=2,emb dropout=0.1,gradient clipping to 1,Lr scheduler),so I think may be addition of more training data will increase the model accuracy and it indeed works because for less data and more model parameter our model was overfitting.

2.Transformer (4-Blocks in Encoder and Decoder) + Data Augmentation + epoch=20:

D model(embedding dimension or hidden size)=256

N head(Number of heads in multiheaded self attention)=8

num layer(Number of Blocks in Encoder and Decoder)=4

dropout=0.1(for regularization random drop 10% neuron durin training

d ff(Feed forward network inside each Transformer block)=1024

Vocab size(Total vocab size in training of Vocabulary)=40K

Seq length(max length sequence) =50 (set to 50 to avoid major truncation) epochs=20

LR(Learning Rate) = 5e-4

This model was performing best of all I have tried in training phase and taking less time than 3 layer BiLstm model because lstm being seq2seq model which takes lots of time due to sequential computation,It best performing because self attention in Encoder and multiheaded attention able to capture multiple perspective in sentence.

5 Experiments

1. Data Pre-processing you did both for source and target language. What was the reason for doing this kind of pre-processing.

   As we have already discussed that the data is quite clean and well so we don't have to do much text cleaning for both source and target for both Train and val data. Majorly we have applying following pre-processing:

   1.Whitespace normalization:

   I have replaced multiple or irregular spaces and newline characters with a single space using regular expressions, `re.sub(r'+', ' ', text)`. This ensured that the sentences had uniform spacing, avoiding unwanted token breaks during subword segmentation in bpe tokenizer.

   2.Quotation Mark Standardization:

   Unicode quotes such as " " and ' ' were replaced with standard ASCII quotes ("), as this ensure uniform text formatting across different languages and sources.

   3.Specific script based cleaning :

   For Hindi, only characters in the Devanagari script `(u0900-u097F)`, spaces, and basic punctuation were retained.

   Only Bengali script characters `(u0980 - u09FF)`, spaces and punctuation were retained for Bengali.

   For English, all non-alphanumeric characters except punctuation were removed; text was converted to lowercase to maintain consistency.

   4.Lower case normalisation(only for english):

   Lower casing english will reduce the vocab size for bpe(byte pair encoding tokenizer) tokenizer and will improved generalisation by merging "India" and "india" as same token.

2. Training procedure: Optimizer, learning rates, epochs, training time, etc for different models you tried.

   Optimizer:

   The optimizer used here for all the models is Adam, because generally it gives faster convergence and better stability than its alternatives, such as RMSProp and SGD. Adam combines the advantages of momentum with adaptive learning rates; hence, it works very well with sequence models, like LSTMs and Transformers.

   Criterion :

   CrossEntropyLoss was used as the loss function, with an ignore index of 0 set for padding. By doing this, the model will not consider padded tokens of the input sequences for calculating the overall loss and hence will only optimize over meaningful tokens because it does not contribute in meaning.

   Learning Rate:

   The learning rate varied between 0.0001 to 0.0005 for all experiments, ensuring both stable convergence and faster learning. Furthermore, a learning rate scheduler was employed to handle the automatic reduction of LR in case of a plateau in validation loss to allow the models to fine-tune the weights at later epochs without overshooting the minima.

   Teacher Forcing:

   Teacher forcing with the correct previous token was used during training for a fraction of the steps. The most usual range in which the teacher forcing ratio was maintained was between 0.6 and 0.85 a trade-off between stability and generalization: higher ratios helped faster convergence in earlier epochs, while lower ratios made the model learn to predict based on its own predictions for better results in inference.

Epochs :

The number of epochs varied with the model complexity. For simpler architectures, such as vanilla LSTMs, models were trained for approximately 50 epochs to ensure enough exposure to the data. In contrast, deeper or more complex models like multi-layer BiLSTMs or attention-based networks converged after approximately 10 to 20 epochs due to their higher learning capacity and faster representation power and speciallly Transfomer based was guving good enough result in 10 epoch.

3. Details about different hyper-parameters for different models.

   Model Hyperparameter:

   Extra Experiments:

   1.Data Augmentation(With sir permission) :

   I included 100,000 extra English–Bengali and English–Hindi sentence pairs from cleaned public datasets in order to reduce overfitting for deeper models such as 2–4 layer BiLSTMs.

   This augmentation improved generalization; for example, the BLEU score of the 2-layer BiLSTM with attention model increased from 0.20 to 0.23, and the Transformer models also performed better due to increased linguistic diversity and sentence variety.

   2.Reverse Training:

   I also tried reversing the source sentences before training. This method usually helps in languages like French, where much of the context appears at the beginning of the sentence. However, for Hindi and Bengali, sentence meaning is more spread across the entire sequence. As a result, reversing didn't improve performance and sometimes made training slower, so I did not use it in the final models.This is given in one research paper.

   3.BPE vs SP:

   Both BPE and SP tokenization methods gave roughly similar results, but SP had some advantage for larger vocabulary sizes of about 32K–40K vocab size. SP's flexibility in handling subword segmentation and working directly on raw text made it slightly more effective, especially for morphologically rich languages like Hindi and Bengali.

   4.Change in Initial Bpe to handle spaces:

   This modified tokenizer encodes the space as part of the subword token (e.g., `("I")`, `("_love")`, `("_to")`, `("_eat")`). By adding a leading `"_"`, BPE treats the space as part of the next word and provides the Transformer with an explicit boundary marker indicating that a new word or phrase is going to starts from here. This boundary signal helps attention heads reliably identify syntactic units such as `_to`, `_the`, or `_and`, otherwise it would have been pickup sequence of characters.

   5.Tried to implement self Attention in Encoder(BiLSTM) along with Loung attention in Decoder:

   In this implementation, self-attention is applied within the BiLSTM encoder so that each word representation is contextually enriched through attending to other tokens in the same sequence. That helps the encoder capture global contextual relationships beyond sequential limitations of standard LSTMs.

6 Results

1. I have already shown with different model for Training phase in model descriton section.

Table 3: Model Performance on test data in testing phase

Seq No.	Model Name	Score
1	Seq2Seq (GRU) (very less epoch just for submission checking)	0.06
2	2 Layer BiLSTM + Attention + epoch=10	0.18
3	2 Layer BiLSTM + Attention + Data Augmentation + epoch=20	0.18
4	2 Layer BiLSTM + Attention + Data Augmentation + epoch=25	0.20
5	Transformer (4-Blocks in Encoder and Decoder) + Data Augmentation + epoch=20:	0.29

2. Again same logic goes as I have discussed in model discription part.
   D model(embedding dimension or hidden size)=256
   N head(Number of heads in multiheaded self attention)=8
   num layer(Number of Blocks in Encoder and Decoder)=4
   dropout=0.1(for regularization random drop 10
   d ff(Feed forward network inside each Transformer block)=1024
   Vocab size(Total vocab size in training of Vocabulary)=40K
   Seq length(max length sequence) =50 (set to 50 to avoid major truncation)
   epochs=10
   LR(Learning Rate) = 5e-4
   Total trainable parameter :38121536(approx 40 million)
   This model was performing best of all I have tried in training phase and taking less time than 3 layer BiLstm model because lstm being seq2seq model which takes lots of time due to sequential computation,It best performing because multi headed self attention in Encoder and cross attention in Decoder able to capture multiple perspective in sentence.

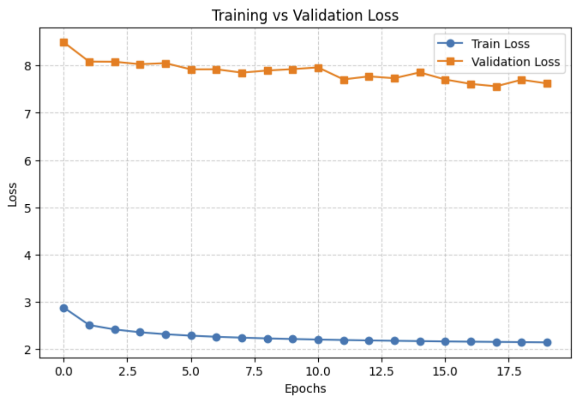


Figure 5: Loss for Eng-Ben(similar for Eng-hi)

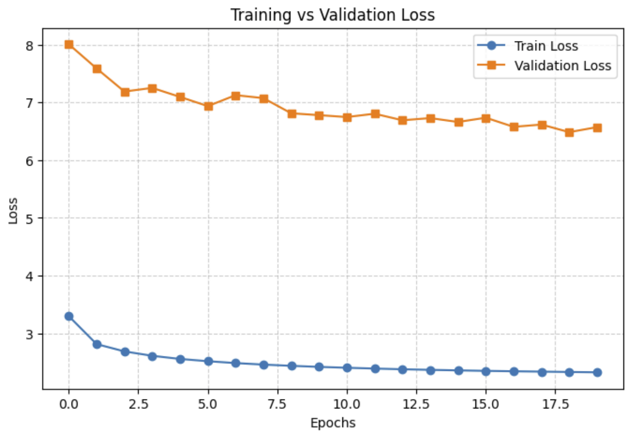


Figure 6: Loss for Eng-Hin

7 Error Analysis

1. Error analysis of your different models both on the dev set and the test set. What models worked in general and in what kind of setting? What was the reason for it?

Ans:

Basically for my case Transformer with 4-blocks with 8 heads and dropout=0.1 is working best model but it was also overfitting little bit as we can see in above loss diagram also in my case beam search k=5 was not performing good (it should perform better as compare to greedy but also it was lots to time in decoding).

Main reason for working is :

1.Balanced number of Blocks i.e. 4 and heads i.e. 8

2.Good Dropouts for regularization i.e. 0.1 if model is not so complex.

3.Multi headed self attention able to capture more contextual meaning in sentence.

2. Analyze why models are not perfect? E.g., what kind of mistakes are made by the best model, how could these be overcome?

   Ans:

   Main reason for not being perfect is :

   1.First and for most reason is overfitting since we do not have much data ,so it very hard to choose appropriate parameter setting to avoid overfitting.

   2.Second is, if datasets have idoms or polysemous words(eg take off) then model struggle to train properly.

   3.Rare words or unknown words marked to UNK,so if data sets have large number of rare words in test data then also it will not perform well.

   4.In my case beam search was producing repetitive data.

   5.In my case high(0.6 to 0.85) value to teacher forcing could be one reason for bad performance.Traning was fast but it was too rely on ground truth lebel.

3. Any interesting insights!

   1.Data Quality Matters More Than Model Complexity

   Because as I added data to bit complex model it start performing well for same model.

   2.Reverse Training Didn't Help here due to non front end loading of context in hindi/bengali language.

8 Conclusion

Key Finding for my case.

1. Adding more data to model if there large number of parameter then We should be Add more data to avoid better training and avoid overfitting.

2.For Language like french which has more context in front part of sentence reversing the source sentence works well and improve accuracy.

3.Also sentence piece tokenizer works better then Byte pair encoding.

4.For very complex model like BiLstm and 3-4 layer keep Dropout 0.2 to 0.4 works better for highly stacked modle.

5.For Transformer 4-Blocks and num heads=8 works well(although I didn't experiment much this) but layer like 6 is overfitting with Dropout 0.1 ,if we are keeping layer 5+ then we should regularize heavily like dropouts 0.2+.

6.Particularly for my case I am not able to figure it out why beam search is giving bad result than greedy.

Some Challenges that I faced during project :

1.GPU compute, since we didn't had enough GPU compute (we were only relies on Google colab(runtime disconnect main issue) and Kaggle)

2.This is particularly for my case only,since I was using additional data there was storage(RAM crashing and disconnection) problem specially on Google colab which I handle with Kaggle.

References

1.arxiv.org/pdf/1409.3215
2.YouTube videos
3.Pytorch Tutorial
4.arxiv.org/pdf/1703.01619
5.arxiv.org/ftp/arxiv/papers/2003/2003.08925.pdf