CS779 Competition: Machine Translation System for India

Sachin <Vinod> Bhadang
200831
{sachinb20}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

This report provides an analysis of the methodologies employed in the creation of a Machine Translation Network for the second phase of the assignment. In Phase 2 of the assignment, the objective was to design a model capable of translating Indic Languages(Bengali, Gujarati, Malayalam, Kannada, Telgu, and Tamil) text into English. A single transformer model-based approach has been implemented. The final model developed achieved 9th place on the test dataset and 3rd place on the validation dataset. The Blue Score, Rough Score, and ChrF Score metrics were used to gauge the performance of a model.

1 Competition Result

Codalab Username: S200831
Final leaderboard rank on the test set: 9
charF++ Score wrt to the final rank: 0.331
ROGUE Score wrt to the final rank: 0.390
BLEU Score wrt to the final rank: 0.115

2 Problem Description

In this assignment, the objective is to develop a Machine Translation System customized for India. The Phase 2 problem statement involves creating a model capable of translating Indic languages input text(Bengali, Gujarati, Malayalam, Kannada, Telugu, and Tamil) into English. This challenging task is critical due to India's rich linguistic diversity.

Independent data has been provided for each of the seven English-Indian language pairs to facilitate the evaluation of translation performance. The model's effectiveness is assessed using three key metrics: Blue Score, Rough Score, and ChrF Score. The competition and leaderboard for this assignment were hosted on CodaLab.

Phase 2 of the assignment was further subdivided into two essential parts: the training phase, where the model was developed and refined, and the test phase, where the models' performance was evaluated on the test data.

3 Data Analysis

The training dataset is structured as a JSON file containing uniquely indexed data points for English text and its corresponding local language text. The distribution of data points for each language pair is summarized in the table below:

The table provides an overview of the quantity of data available for training the machine translation model for each English-to-local language pair. Additionally, the validation and test sets were introduced during the training and testing phases, respectively. A 70:10:20 split was created for train, val and test sets respectively.

Language Pair	**Train Data Points**	**Val Data Points**	**Test Data Points**
English-Bengali	68848	9835	19671
English-Gujrati	47482	6783	13567
English-Hindi	80797	11543	23085
English-Malayalam	54057	7723	15446
English-Kannada	46795	6685	13371
English-Telugu	44905	6415	12381
English-Tamil	58361	8338	16675

Table 1: Number of Data Points for Each Language Pair

In the dataset, we encountered several instances where English text was erroneously found beneath the local language text, both in the training set and the test and validation sets.

We also noted occurrences where English text was inaccurately placed within the local language text in the training set. The maximum number of wrongly placed data points was observed in the English-Malayalam language pair.

Moreover, we identified noisy data points within the training set, including examples like ";", "$10^3 - 10^5$ CFU/ml" and "2.1.3.2. eCIM".

These instances of noise and misplaced data required careful handling and cleaning during the model development and evaluation processes. A detailed description of the number of noisy and misplaced data points is described in Table 2.

Language Pair	**Train Set (English)**	**Train Set (Indian L)**	**Val Set**	**Test Set**
English to Bengali	2	8	-	4
English to Gujarati	2	-	-	-
English to Hindi	4	30	5	5
English to Malayalam	787	788	102	215
English to Kannada	6	8	2	2
English to Telugu	-	3	-	-
English to Tamil	11	5	-	3

Table 2: Noise and Misplaced Data Points in Data Sets

4 Model Description

The models developed are focused on the Transformer architecture [1]. All experiments and results were achieved on a **single model**. Three approaches were tried as follows:

1. Single Encoder - Single Decoder:
   (a) Modelling Entire Corpus
   (b) Transliterating Similar Languages
2. Multi Encoder - Single Decoder
3. Mean Parameter Model

In the initial model, the embeddings are designed to learn distinct representations for each language, preserving the unique characteristics of each language when modeling the entire corpus in its original form. The rationale behind using transliteration is that Northern Indian languages like Gujarati, Hindi, and Bengali share many words with the same phonetics and meanings. Therefore, by transliterating, we aim to establish a unified representation for these common words across all languages.

The Mean Parameter model represents a straightforward approach, where we simply average the individual language pair models.

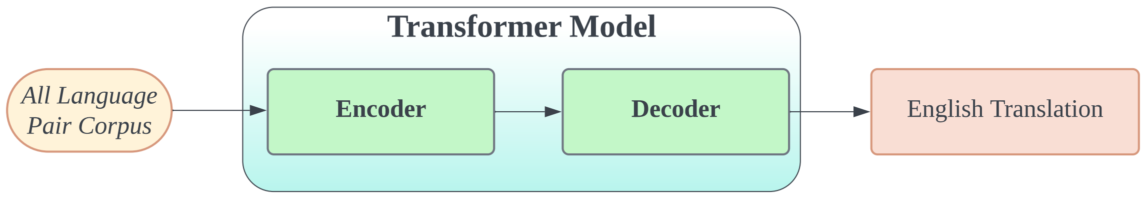

Figure 1: Design of the Single Encoder - Single Decoder Transformer Model on Entire Corpus

For the Multi Encoder - Single Decoder model, we follow a different path. Here, we train multiple encoders, each specific to a particular language, while sharing a common decoder for all. Further details for each of these approaches are provided below.

The model architecture for all the approaches outlined below is similar, with minor differences as listed:

- EMBEDDING SIZE: 400
- FEEDFORWARD NETWORK DIM: 512
- ENCODER LAYERS: 4
- DECODER LAYERS: 4
- DECODER INPUT: TGT_VOCAB_SIZE
- ENCODER INPUT: SRC_VOCAB_SIZE

4.1 Single Encoder - Single Decoder

4.1.1 Modelling Entire Corpus

In this specific approach, we harness the remarkable capabilities of a transformer network, renowned for its proficiency in extrapolating patterns across a diverse spectrum. As illustrated in Figure 1, we engage in modeling the entire corpus, a task at which the Transformer excels. This process leads us to the inference that the embeddings successfully acquire a robust representation spanning multiple languages. We use the Cross-Entropy loss as the network's training objective.

4.1.2 Transliterating Similar Languages

The motivation for employing transliteration lies in the fact that Northern Indian languages such as Gujarati, Hindi, and Bengali exhibit a significant number of words that share phonetic similarities and carry identical meanings. This is substantiated by the observed reduction in vocabulary when constructing a unified vocabulary post-transliteration. Hence, the primary objective of transliteration is to forge a common representation for these shared words across all languages as shown in Figure 2. We used the Cross-Entropy loss as the training objective for the network

In this pursuit, two distinct models are crafted—one for South Indian languages and another for their Northern counterparts. Intriguingly, these models outperform the approach of modeling the entire corpus as a single entity.

4.2 Mean Parameter Model

For this strategy, our initial step involves training distinct models for each language pair using similar Transformer Encoder-Decoder architectures. In our ultimate single model, we perform an averaging operation on the trained parameters of the Transformer model, excluding the Generator, Source, and Target Embedding weights, which we concatenate. We used the Cross-Entropy loss as the training objective for the network

However, it's important to note that this approach, while straightforward, does not yield satisfactory results. It demands a more thoughtful consideration of which parameters should be subject to averaging, as well as a thorough evaluation of the pre-processing methods employed.

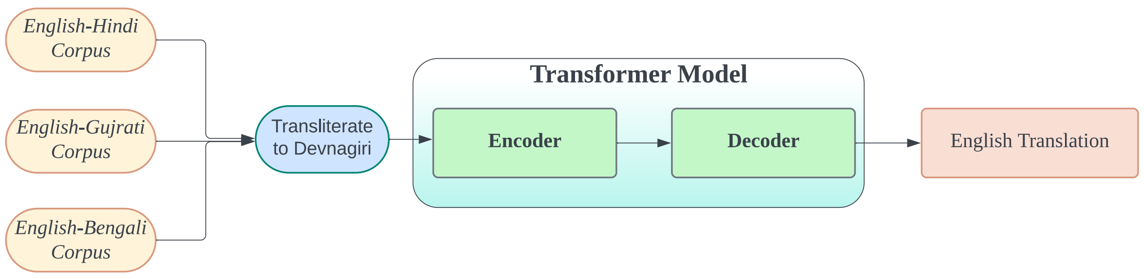

Figure 2: Design of the Single Encoder - Single Decoder Transformer Model for Transliteration Model

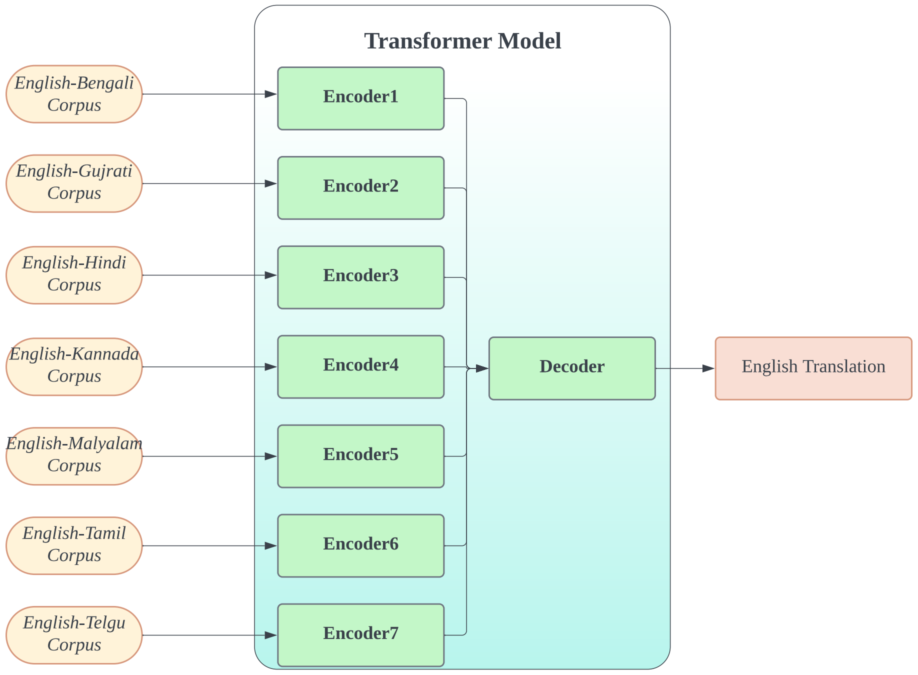

Figure 3: Design of the Multi Encoder - Single Decoder Transformer Model

4.3 Multi Encoder - Single Decoder

In the case of the Multi Encoder - Single Decoder model, we adopt a distinct approach. In this configuration, we train a set of multiple encoders, each tailored to a specific language, while simultaneously employing a shared decoder for all. Each of these encoders adheres to a uniform architectural structure, differing solely in their input data sources, but maintaining identical input dimensions. The performance of this model is on a par with that of the Single Encoder-Single Decoder Model.

We used the Cross-Entropy loss as the training objective for the network. The model underwent training with the application of two distinct Early Stopping criteria. In one scenario, training continued until the cumulative difference in validation loss across all languages dropped below 0.1. In the other case, the Early Stopping criterion was met when the validation loss for each individual language reached a value less than 0.01. To ensure optimal model performance, we created checkpoints after each epoch and selected the best model weights based on an analysis of the loss.

5 Inference

Two inference methods were implemented in our machine translation model: greedy decoding and beam-search.

In greedy decoding, the model selects the token with the highest probability at each step, generating the output sequence iteratively until an end-of-sequence token is reached. Beam-search, on the other hand, maintains a set of partially decoded sequences, or "beam," and expands candidates based on conditional probabilities. The beam width limits the number of candidates considered at each step, retaining those with the highest probabilities.

Beam-search explores a wider range of possibilities compared to greedy decoding, potentially leading to more accurate translations. The choice between these methods depends on specific translation requirements and trade-offs. Greedy decoding is faster but may sacrifice quality, while beam-search offers greater translation accuracy at the cost of increased computation.

Experiments were run on both greedy decoding and beam-search, but better results were obtained on Greedy Decoding. Results shown hereafter are using Greedy Decoding.

6 Experiments

6.1 Pre-Processing

The initial step in the data preprocessing pipeline involved cleaning the dataset to remove any noise and misplaced data points. Once the data was cleaned, two distinct tokenization processes were applied: one for the English text and another for the Hindi data.

For the English text, the Spacy library[2] was employed for tokenization. For tokenizing the Indian Language data, the IndicNLP library[3] was utilized. By using these specific libraries for tokenization, the preprocessing pipeline ensured that both the English and Hindi text data were appropriately segmented into tokens, setting the stage for subsequent processing and modeling steps in the machine translation task.

Following the tokenization of the input data, the next step in the data preprocessing pipeline involved constructing a vocabulary.

To create the vocabulary for Modelling the Entire Corpus and Transliterating Similar Languages, tokens with a minimum frequency of occurrence of 2 or more were included in the vocabulary. After Transliterating we notice the vocabulary reduces substantially, since a lot of words are common in Indic languages.

For the Mean Parameter model, we take the top 30,000 tokens, with a minimum frequency of occurrence of 2, for each language. This is done to ease the averaging of parameters.

For the Multi Encoder-Single Decoder model we take the top 30,000, with a minimum frequency of occurrence of 2, tokens for Indic Languages and the top 70,000 for English.

6.2 Training Procedure and Hyperparameters

The final models were trained on the same hyperparameters

- Optimizer : Adam
- Learning Rate : 1e-4
- Epochs : 36
- Training Time : Each epoch took about 30-40 min (for training entire corpus for Multi-Encoder and Single-Encoder)

7 Results

Here, we present the outcomes of our diverse approaches. With the exception of the Transliteration Model, the results are derived from the entire Test/Validation Set. The Transliteration model, in contrast, is exclusively evaluated within the context of the English-Hindi/Gujarati/Bengali language pairs.

To demonstrate the efficacy of transliteration, we conduct a comparative evaluation with a Control Model specifically on Northern languages. The Control Model is designed to possess matching architectural parameters and vocabulary size, mirroring the setup of the Transliteration Model trained on the English-Hindi/Gujarati/Bengali language pairs in their unaltered form.

Model (Test/Val Set)	**ChrF Score**	**ROUGE Score**	**BLEU Score**
Single Encoder and Decoder (Entire Corpus)	0.331	0.390	0.115
Single Encoder and Decoder (Transliteration)	0.393	0.421	0.143
Multi Encoder- Single Decoder	0.355	0.403	0.135
Mean Parameter Model	0.096	0.158	0.003

Table 3: Model Evaluation Scores

Model (Test Set)	**ChrF Score**	**ROUGE Score**	**BLEU Score**
Transliteration Model	0.393	0.421	0.143
Control Model	0.340	0.411	0.134

Table 4: Efficacy Of Transliteration

8 Error Analysis

During the training phase, an early stopping mechanism was implemented to monitor the change in validation loss between epochs. If the change in validation loss fell below a threshold of 0.001, the training process was halted. It's important to note that a significant portion of the trained models did not reach the maximum of 36 epochs but instead stopped earlier, typically around the 10-12 epoch range for Modelling the Entire corpus and 6-7 for Multi Encoder-Singel Decoder. The validation loss observed at this point in training typically ranged from 3.5 to 4.

This early stopping strategy is a common practice in machine learning to prevent overfitting and ensure that the model doesn't continue training when improvements in performance on the validation set become marginal. By stopping training when the validation loss stabilizes, we aim to strike a balance between achieving good model performance and avoiding excessive training that could lead to overfitting on the training data.

One of the challenges encountered during translation was the frequent appearance of the "unk" token, which typically indicates an unknown or out-of-vocabulary word. A practical solution to address this issue was to disregard sentences with more than 40 percent "unk" tokens.

In the absence of released ground truths for the test and validation datasets, I employed an alternative approach by using the Google Translate API to generate reference translations. This creative solution allows to evaluate and test the judge the performance of each language pair model individually.

9 Conclusion

In conclusion, the report outlines the approaches used for the machine translation system, highlights model performance, and provides insights into the challenges encountered during the process. The Transliteration Model demonstrated its effectiveness in handling Similar Indian languages, showcasing the value of transliteration as a technique in machine translation tasks.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[2] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." To appear, 2017.

[3] A. Kunchukuttan, "The IndicNLP Library." `https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf`, 2020.