CS779 Competition: Machine Translation System for India

Yashika Malhotra
201152
{yashika20}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

Machine Translation (MT) is a domain of computational linguistics, which explores the use of software to translate text or speech from one language to another. Neural machine translation (NMT) is an approach to machine translation that uses an artificial neural network to predict the likelihood of a sequence of words, typically modeling entire sentences in a single integrated model. The goal of competition's second phase is to develop a Neural Machine Translation (NMT) model that can translate from seven Indian languages (Bengali, Gujarati, Hindi, Kannada, Malayalam, Tamil, and Telugu) to English. The implementation in Phase 2 involved a sequence-to-sequence transformer-based architecture. The model demonstrated substantial improvements in performance metrics over the LSTM sequence-to-sequence architecture employed in Phase 1, achieving notable scores, including a charF++ score of **0.374**, a ROGUE score of **0.380**, and a BLEU score of **0.093** on the test data

1 Competition Result

Codalab Username: Y_201152
Final leaderboard rank on the test set: 11
charF++ Score wrt to the final rank: 0.374
ROGUE Score wrt to the final rank: 0.380
BLEU Score wrt to the final rank: 0.093

2 Problem Description

The primary objective of this report is to demonstrate solutions for the task of translation from 7 Indian Languages to English. The task is accomplished using a transformer-based model by taking inspiration from this official Pytorch tutorial blog [1]. Transformers, known for their self-attention mechanism and bidirectional processing, form the basis of state-of-the-art machine translation models, employing encoder-decoder architecture, multi-head attention, and positional encoding for efficient and accurate translation across various language pairs. The models are evaluated using the charF++, ROGUE, and BLEU Score metrics.

3 Data Analysis

The training data comprised of **4,01,243** source sentences in the seven Indian languages and their corresponding English translations. Upon careful examination, it was observed that some of the Indian language sentences contained English words in them. Such sentences were discarded from the training dataset. Table 1 shows the distribution of training data for the 7 languages.

While the test data comprised of **1,14,645** source sentences, which included **23,085** from the "English-Hindi" language pair, **13,370** from the "English-Kannada" language pair, **19,671** from the "English-Bengali" language pair, **13,567** from the "English-Gujarati" language pair, **15,445** from the "English-Malayalam" language pair, **16,675** from the "English-Tamil" language pair, and **12,830** from the "English-Telugu" language pair.

S.No.	Indian Language	Sentence pairs
1.	Hindi	80,797
2.	Kannada	46,794
3.	Bengali	68,848
4.	Gujarati	47,482
5.	Malayalam	54,057
6.	Tamil	58,361
7.	Telugu	44,904

Table 1: Training Data Distribution

4 Model Description

The Transformer Sequence to Sequence model uses an encoder-decoder structure where the encoder processes the input language sequence, transforming it into a set of vectors. These vectors are collectively referred to as "Encoder Output" and capture the contextual information of the input sequence. The encoder employs self-attention mechanisms to understand the relationships between words in the input sentence, enabling it to represent the source sentence comprehensively. The decoder, on the other hand, takes the encoder's output and generates the target translation one word at a time. It uses a similar self-attention mechanism, but it pays attention to both the input sentence and the words it has generated so far.

The network consists of three parts. First part is the embedding layer. This layer converts tensor of input indices into corresponding tensor of input embeddings. These embedding are further augmented with positional encodings to provide position information of input tokens to the model. The second part is the actual Transformer model. Finally, the output of the Transformer model is passed through linear layer that gives unnormalized probabilities for each token in the target language.

4.1 Model Evolution

The model consisted of 3 encoder and decoder layers, 8 multi-head attention mechanisms, and embeddings of 512 dimensions. The word embeddings were of dimension 512, and the source and target vocabularies were built keeping the minimum frequency as 1. In the development phase, I experimented with minimum frequencies 2 and 3, though bouth of them resulted in an increased training loss. Hyperparameters such as the number of attention heads, hidden layer dimension for the feedforward neural network, dropout rate, and the number of training epochs were tuned. Data was processed in batches of size 32, with specific vocabulary indices for padding (pad_idx), beginning of sentence (bos_idx), and end of sentence (eos_idx). The model optimization was achieved using the Adam optimizer with a defined learning rate of 0.0001. During the inference phase, a greedy approach was employed. This approach involves making predictions one token at a time by selecting the most likely token at each step, based on the model's output and previous predictions.

5 Experiments

5.1 Data pre-processing

7 models were separately trained corresponding to each of the target languages. A subset of the training data comprising the "English-target language" pair was extracted. The English sentences were converted to lowercase to ensure uniformity, followed by which punctuations were removed. Sentences in the source language were filtered to exclude those containing English words, so as to exclude the mixed language sentences.

The source Indian language sentences were tokenized using the Indic NLP Library. While the target English sentences were tokenized using the spaCy library with the *'en_core_web_sm'* model.

5.2 Training Procedure

The hyperparameters used for different iterations in the training phase are presented in Table 2. It was found that increasing the number of epochs and decreasing the learning rate led to an increase in the charF++, rogue, and bleu scores. The optimal learning rate was found to be 0.0001, which resulted in the minimum training loss. Also, the transformer model was seen to perform much better than LSTM Seq2Seq model.

S.No.	Model	Optimizer	Learning Rate	Training Time	Epochs
1.	LSTM	Adam	0.01	21 mins	3
2.	LSTM	Adam	0.001	1 hr 40 mins	10
2.	Transformer	Adam	0.0001	1 hr 23 mins	20
2.	Transformer	Adam	0.0001	3 hr 09 mins	40

Table 2: Hyperparameters for different models

6 Results

The results for different iterations in the training phase and the test phase are presented in Table 3. It was observed that the performance increased by raising the number of epochs. In the development phase, an initial observation revealed that the inference process was consuming an increasingly significant amount of time. To resolve this issue, a strategy was implemented, rather than sending complete source sentences into the translation function, pre-tokenized sentences were used. This adjustment resulted in a substantial reduction in inference time, by a fraction of 0.4.

Phase	charF++ score	ROGUE score	BLEU score	Rank
Development	0.366	0.376	0.089	9
Test	0.374	0.380	0.093	11

Table 3: charF++, ROGUE and BLEU scores obtained during different phases

7 Error Analysis

Given below figures show the loss vs epoch plot for a range of epochs for translation from Indian languages to English. It is evident from the plot that as the number of epochs increases, the slope of the curve decreases Therefore, not much variation was observed in the BLEU score when the number of epochs was raised from 30 to 40. Increasing the number of epochs beyond the saturation limit can lead to overfitting of data resulting in reduced performance

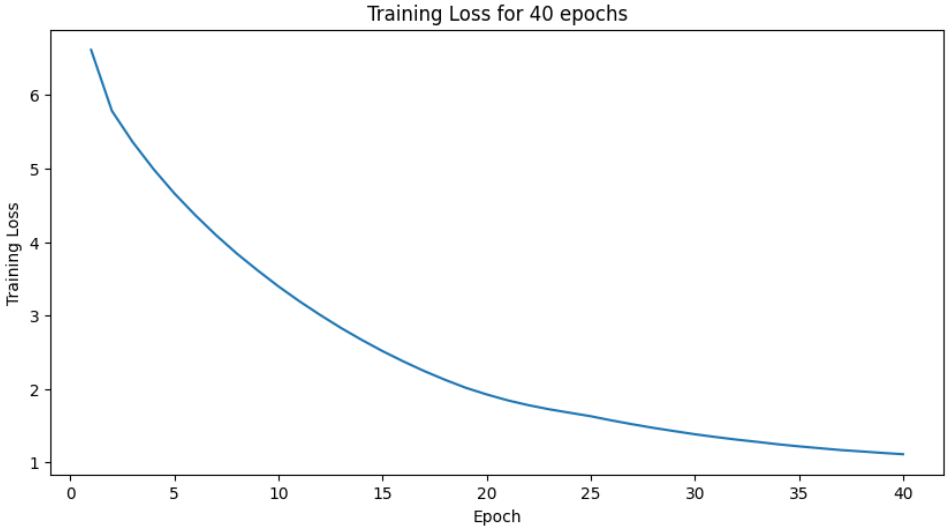

Figure 1: Loss vs Epoch plot for Telugu to English translation

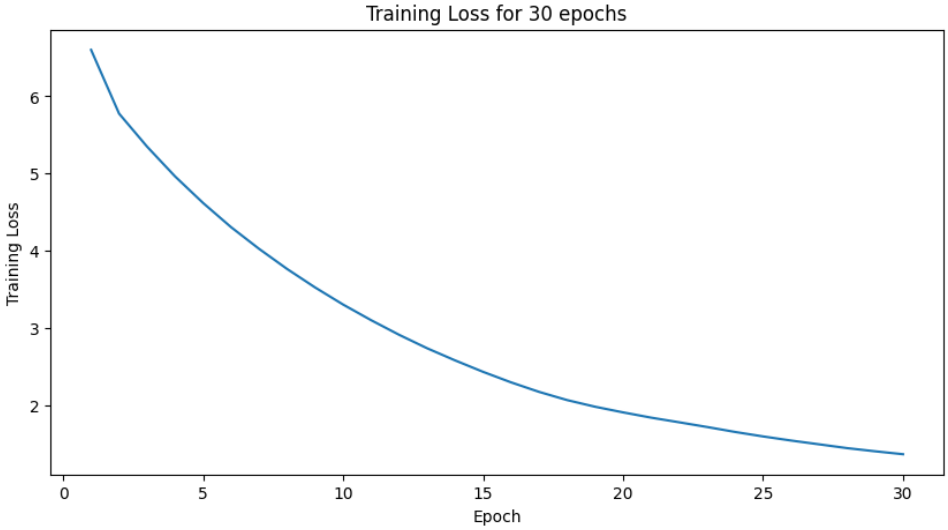

Figure 2: Loss vs Epoch plot for Kannada to English translation

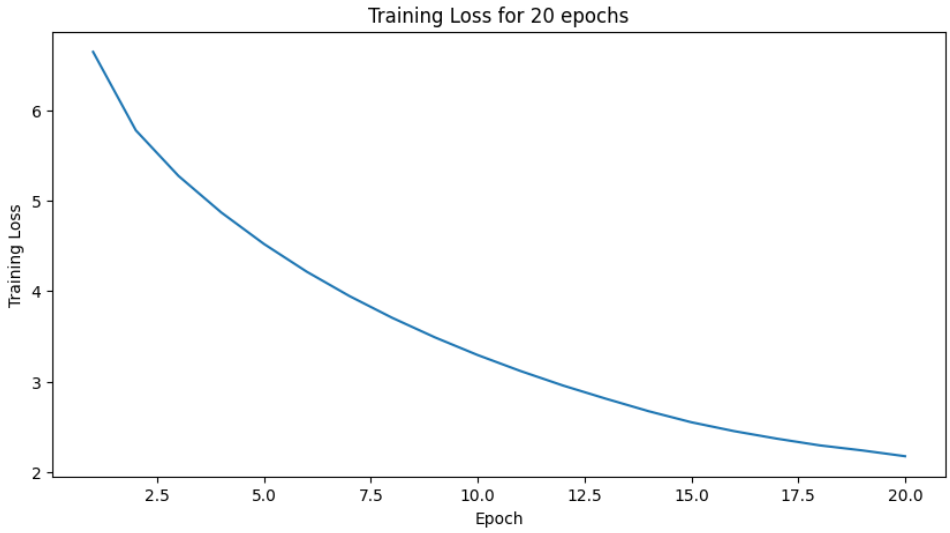

Figure 3: Loss vs Epoch plot for Bengali to English translation

8 Conclusion

I observed that the transformer model demonstrated superior performance compared to other models on both the development and test datasets. Given the relatively lower complexity of the dataset, exploring simpler models or employing fewer encoder layers, a reduced number of multi-head attention mechanisms, and a lower embedding dimension may lead to improved model performance.

References

[1] "Language translation with nn.transformer and torchtext — pytorch tutorials 1.12.1+cu102 documentation." `https://pytorch.org/tutorials/beginner/translation_transformer.html`. Accessed: Oct. 29, 2023.

[2] R. Andre, "Japanese-english machine translation model with transformer & pytorch." `https://arusl.medium.com/243738146806`, 2021. Accessed: Oct. 29, 2023.