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 competition's goal is to develop a Neural Machine Translation (NMT) model that can translate from English to seven Indian languages (Bengali, Gujarati, Hindi, Kannada, Malayalam, Tamil, and Telugu). As part of the competition, an LSTM (Long Short Term Memory) based Sequence to Sequence (seq2seq) model was implemented which uses an encoder-decoder-based architecture. On the test data, a charF++ score of **0.208**, a ROGUE score of **0.269**, and a BLEU score of **0.028** were achieved.

1 Competition Result

Codalab Username: Y_201152
Final leaderboard rank on the test set: 14
charF++ Score wrt to the final rank: 0.208
ROGUE Score wrt to the final rank: 0.269
BLEU Score wrt to the final rank: 0.028

2 Problem Description

The primary objective of this report is to demonstrate solutions for the task of translation from English to 7 Indian Languages. The task is accomplished using an LSTM based Seq2Seq model by taking inspiration from this official Pytorch tutorial blog [1]. The Sequence to Sequence model uses an encoder-decoder based architecture. These models are evaluated using the charF++, ROGUE, and BLEU Score metrics.

3 Data Analysis

The training data comprised of **4,01,246** source sentences in English and their corresponding Indian Language 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 **1,14,647** English source sentences, which included **23,085** from the "English-Hindi" language pair, **13,371** from the "English-Kannada" language pair, **19,672** from the "English-Bengali" language pair, **13,567** from the "English-Gujarati" language pair, **15,446** from the "English-Malayalam" language pair, **16,675** from the "English-Tamil" language pair, and **12,831** from the "English-Telugu" language pair.

S.No.	Indian Language	Sentence pairs	Mixed Sentence pairs
1.	Hindi	80,797	6,193
2.	Kannada	46,795	681
3.	Bengali	68,849	790
4.	Gujarati	47,482	634
5.	Malayalam	54,057	1,495
6.	Tamil	58,361	1,296
7.	Telugu	44,905	614

Table 1: Training Data Distribution

4 Model Description

The LSTM Sequence to Sequence model uses an encoder-decoder architecture where the encoder encodes the input language sequence into a single vector, that is the Context Vector. The context vector contains an abstract representation of the input sequence. The vector is then passed to the decoder, which outputs the corresponding output translation, generating one word at a time. The model architecture for translation to all 7 languages remains the same. The model was developed by taking inspiration from this blog [2]

4.1 Encoder Model Architecture

The input size is set to the length of English vocabulary since it represents the size of one-hot vector that will be input to the encoder. Encoder embedding size is set to 300. The dimension of hidden states in the LSTM cells is set to 1024. The encoder consists of 2 LSTM layers stacked on top of each other. A dropout rate of 50% is applied to the LSTM layers.

4.2 Decoder Model Architecture

The input size of the decoder is set to the vocabulary size of the target Indian language. The decoder embedding size is set to 300. The dimension of hidden states in the LSTM cells is set to 1024. The decoder consists of 2 LSTM layers stacked on top of each other. A dropout rate of 50% is applied to the LSTM layers. The output size equals the length of the target vocabulary.

4.3 Model Evolution

During the development phase, punctuations were not eliminated from the training data at first, and fewer epochs were employed, resulting in a low charF++ score of **0.147**. During the training process, I discovered that the target Indian languages had some English words which were removed in the later iterations. The number of epochs was also raised to 10, after which the charF++ score was calculated raised to **0.206**.

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. Both the source and target sentences were converted to lowercase to ensure uniformity, followed by which punctuations were removed. Sentences in the target language were filtered to exclude those containing English words, so as to exclude the mixed language sentences.

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

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.

S.No.	Optimizer	Learning Rate	Training Time	Epochs
1.	Adam	0.01	21 mins	3
2.	Adam	0.001	1 hr 40 mins	10

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 training phase, I discovered that some of the target sentences contained English words, which reduced the translation quality as the predicted translations contained a certain amount of English words. In the test phase, the mixed-word sentences were discarded resulting in an increase in performance.

Phase	charF++ score	ROGUE score	BLEU score
Dev Iteration 1	0.147	0.258	0.018
Dev Iteration 2	0.206	0.268	0.027
Test	0.208	0.269	0.028

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

7 Error Analysis

In Figure 1, which depicts the loss vs. epoch plot, we observe the results of training an English-to-Kannada translation model for 10 epochs. The plot reveals that the loss metric does not exhibit a continuous decrease as the number of training epochs increases. This suggests that the model's training process may not be converging smoothly, indicating the need to either increase the number of epochs or modify the hyper-parameters.

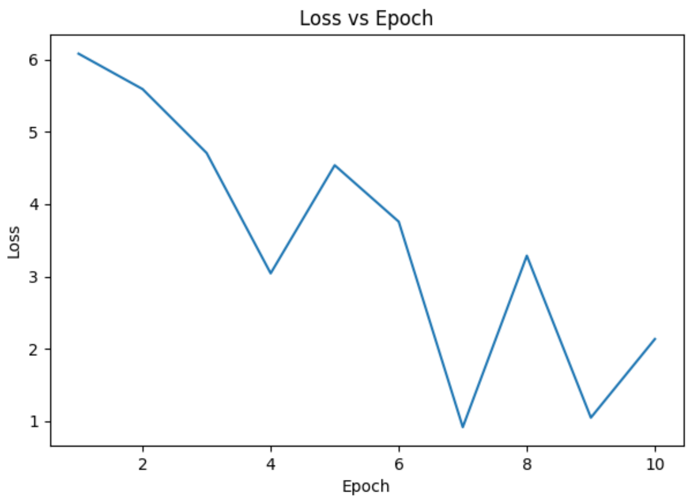

Figure 1: Loss vs Epoch plot for English to Kannada translation

8 Conclusion

Developing the LSTM Seq2Seq model pipeline and debugging the code consumed a significant amount of time during the project's development phase. This time investment limited the opportunity to explore a broader range of models and strategies. Subsequent iterations primarily focused on fine-tuning the existing LSTM Seq2Seq model to improve its performance.

As described in the "Data Analysis" section, the removal of source sentences that contain English words in their Indian language sentences seems to have an impact on the overall score obtained. This also may be one of the reasons behind not achieving an improved score on the test data, and reduced competition rank as compared to the training phase. In the next phase of the competition, I plan to fix this by converting the English words in the respective Indian language sentences into that language itself so as to improve the translation quality.

Therefore, the future directions involve experimenting with more models and strategies, including introducing attention mechanism in the Seq2Seq architecture, trying out different decoder strategies, such as beam search, and implementing transformers.

References

[1] S. Robertson, "Nlp from scratch: Translation with a sequence to sequence network and attention — pytorch tutorials 1.2.0 documentation." `https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html`, 2017. Accessed: Oct. 3, 2023.

[2] B. V, "A comprehensive guide to neural machine translation using seq2sequence modelling using pytorch." `http://bit.ly/towards-data-science-blog-seq2seq-nmt`, 2020. Accessed: Oct. 3, 2023.