CS779 Competition: Machine Translation System for India

Moinuddin
roll no. 231110030
moinuddin23@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

The project aimed to build a translation model from scratch for seven Indian languages to English using PyTorch, without pre-trained models. A subset of the large training data was used for faster training. We tested various sequence-to-sequence models with different hyperparameters. The best model, featuring attention mechanism in sequence-to-sequence, achieved a rank of 23 on the leaderboard, with a charF++ Score of 0.097, ROGUE Score of 0.097, and BLEU Score of 0.000. Challenges included incorrect translations in the dataset.

1 Competition Result

Codalab Username: M_231110030
Final leaderboard rank on the test set: 23
charF++ Score wrt to the final rank: 0.097
ROGUE Score wrt to the final rank: 0.097
BLEU Score wrt to the final rank: 0.000

2 Problem Description

The task of this competition was to develop a model that can translate from 7 Indian languages to English language. those Indian languages are Hindi, Bengali, Telugu, Malayalam, Gujarati, Kannada and Tamil.
We were not allowed to use any pre-trained model. we had to develop it from scratch using PyTorch and had to train it on Google Colab.
We can use any approach such as sequence to sequence with attention or without attention, Transformers, etc. which gives better results and accuracy on validation data and test data. we had to make improvements in our model by doing different experiments on various components of the Model. We had to make 3 submissions per week. and for each submission, The model was evaluated based on corpus-level BLEU-4 score, ROUGE-L score, and chrF++ score.
Taking feedback from the evaluation we had to improve the Model, and in this way, we came up with a final model.

3 Data Analysis

initially, Training and Validation data were provided for this task, Both were in JSON format.
Training data:
Each training example consists of a pair of the source text and its corresponding translations each pair has a unique ID number.
An example of training data for English-Bengali is given in fig.
Similar data is given for all other remaining languages.

```
{
    "English-Bengali": {
        "Train": {
            "0": {
                "source": "তেট্টেক্কাড়ুতে অনেক প্রজাতির পাখি ছাড়াও ২৮ প্রকার বৃক্ষ , ভীমাকৃতি বৃক্ষ , নয় ধরণের সরীসৃপ ইত্যাদি পাওয়া যায় ।",
                "target": "The Bird Sanctuary of Tettekkad Idukki is world famous ."
            },
            "1": {
                "source": "আমাদের দেশে প্রাচীনকালে বিদ্যা শেখানোর পরে বিদ্যার্থীদের পরীক্ষা নেওয়া হত ।",
                "target": "The exam was taken of students after learning education in our country during ancient age ."
            },
            "2": {
                "source": "১৩৩৬ খ্রিষ্টাব্দে কাজটি সম্পন্ন হলেও, পৃথ্বীরাজের উল্লেখ করা অংশটি ১২৫০ খ্রিষ্টাব্দের আশেপাশে লেখা হয়েছিল।",
                "target": "While the work was completed in 1336 CE, the part that mentions Prithviraj was written around 1250 CE."
            },
            "3": {
                "source": "ব্যবসায়ীরা সম্ভবত মঠগুলিতে দান করতেন, কারণ এই স্থানগুলি বিশ্রামাগার হিসাবে কাজ করত এবং সম্ভবত সরাসরি বাণিজ্যে অংশ নিয়ে বাণিজ্যকে সহজতর করত।",
                "target": "The merchants probably donated to the monasteries, because these sites facilitated trade by serving as rest houses and possibly by directly
            },
            "4": {
                "source": "ধানই এক মাত্র স্টার্চ যেটা গ্যাস সৃষ্টি করে না ।",
                "target": "Rice is the only starch which is not responsible for producing gas ."
            },
            "5": {
                "source": "সংক্রামক বস্তু ধরার পর হাত ধুয়ে নিতে হবে।",
                "target": "Wash your hands after handling the infectious material."
            },
```


	Language-pair	Number of sentence pairs
1	English-Hindi	80797
2	English-Malayalam	54057
3	English-Telugu	44904
4	English-Tamil	58361
5	English-Kannada	46794
6	English-Gujarati	47482
7	English-Bengali	68848
	Total	401243

Table 1: Training data

The size of the Training data provided is given in the above table:

Validation data: it consists of a source sentence for each language.
each sentence has a unique ID.
The Size of the dataset for each language is given below :

	Language	Number of source(English) sentences
1	English-Hindi	11542
2	English-Malayalam	7722
3	English-Telugu	6415
4	English-Tamil	8337
5	English-Kannada	6685
6	English-Gujarati	6783
7	English-Bengali	9835
	Total	57319

Table 2: Validation data

Test data : Test data for this task contains Source sentences for each language

example :


```
{
    "English-Bengali": {
        "Test": {
            "157368": {
                "source": "আটা বানাতে এন্দস্পার্ম টিকে খুব সূক্ষ্মভাবে বপন করতে হয়।"
            },
            "157369": {
                "source": "আগেকার পুকুরের মাঝখানে ইহা অবস্থিত।"
            },
            "157370": {
                "source": "এই কারণগুলির মধ্যে রয়েছে: ভালো যোগাযোগকারী, খুঁটিনাটির প্রতি মনোযোগ, বাতিক্রমী ক্রেতা, ভাল বিকল্পদাতা।"
            },
            "157371": {
                "source": "১৭০১ সালে নির্মিত তোপ ` সিংহবাণ ` সভাই জয়সিংহ দ্বিতীয় ১৭২২ - ২৩ সালের দুন ও ১৭৩০ সালের বুন্দীর লড়াইয়ে রাও বুদ্ধ সিংহের বিরুদ্ধে ব্যবহৃত করেছিলেন ।"
            },
            "157372": {
                "source": "যাইহোক, আপনি সচরাচর আগের ভালো অনুভব করবেন চিকিৎসার ২থেকে ৪সপ্তাহের মধ্যে।"
            },
```


The Size of Test dataset for each language is given below :

	Language	Number of source(English) sentences
1	English-Hindi	23085
2	English-Malayalam	15445
3	English-Telugu	12830
4	English-Tamil	16675
5	English-Kannada	13370
6	English-Gujarati	13567
7	English-Bengali	19671
	Total	114643

Table 3: Test data

As we have a large Training dataset our Model was taking a large amount of time to train so chose random sentences from the large dataset and trained on that subset.

4 Model Description $_{reference}$[1]

I developed the sequence-to-sequence model consisting of two RNNs called the encoder and decoder. The encoder reads an input sequence and outputs a single vector, and the decoder reads that vector to produce an output sequence.

Structure of Encoder and Decoder:

The given figure shows the structure of the encoder and decoder.

[1] `https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html`

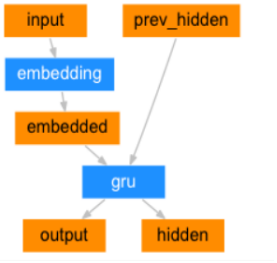


Figure 1: Encoder

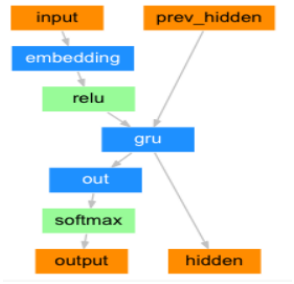
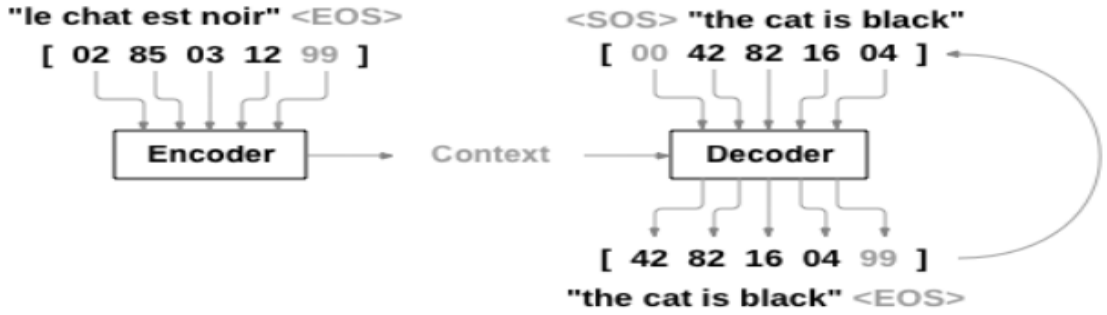


Figure 2: Decoder

Figure 3: Structure of Encoder and Decoder

As given in the figure we are using the GRU layer in both the encoder and decoder. I did translations with this model and tried to get good results by changing hyperparameters.
After this, I replaced the GRU layer of the encoder with the LSTM layer.
Next, I also used the attention mechanism in sequence-to-sequence models.

Workflow of Encoder and Decoder Model:



All images taken from the pytorch tutorial [2]

5 Experiments

I extracted the training data from the JSON file and stored each pair in a list as a nested list. As given training data was too large our model was taking a long time due to the large size of the training data. hence, we extract 5000 sentence pairs from the training data and train our Gru-Gru model on that subset.

Outcomes :

1. Using single GRU layer with 1 gru unit in both encoder and decoder :

In all cases, I chose hyperparameters randomly and used the Adam optimizer to optimize the parameters of an encoder neural network and decoder neural network.And used NNL(Negative Log Likelihood Loss) also known as cross-entropy loss.

Results are given in following table :

[2] https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

Model	Batch-size	Hidden size	Number of epoch	Learning rate	Training data size	Score
M1	128	32	80	0.001	5000	0.089
M2	128	16	80	0.001	5000	0.218

Table 4: variation in scores on different Hyperparameters

2.using a single LSTM layer with 1 LSTM unit in the encoder and a single GRU layer with 1 GRU unit in Decoder

Model	Batch-size	Hidden size	Number of epoch	Learning rate	Training data size	Score
M3	128	32	80	0.001	5000	0.226
M4	64	32	10	0.001	5000	0.215

Table 5: variation in scores on different Hyperparameters

Here also I chose hyperparameters randomly I used the Adam optimizer to optimize the parameters of an encoder neural network and decoder neural network and used NNL(Negative Log Likelihood Loss)

3.I also implemented the attention mechanism in the seq-to-seq model, but I could not submit the result in the training phase.

6 Results

Scores on Validation data :

Model	chrf_score	rouge_score	bleu_score	Total_score
M1	0.058	0.030	0.001	0.089
M2	0.142	0.075	0.001	0.218
M3	0.124	0.101	0.001	0.226
M4	0.139	0.075	0.001	0.215

Table 6: Scores of different models

From experiments, we have various results:
For training purposes size of the training data set is very important as we increase the size our models learn a lot but on increasing data-set size models take a long time to train.
If we compare M1 and M2 both have the exact same hyperparameters except batch size, so from this, we get when we reduce batch size Model performs better.
By comparing M3 and M4 we observe that if we reduce hidden size then the performance of our model is reduced. The best-performing model on validation data is M3 which uses the LSTM layer in the encoder. I could not check for model with attention mechanism in the training phase. all these results are based on my experiments.
For test data, I used a sequence-to-sequence model with attention mechanism.

chrf_score	rouge_score	bleu_score	Total_score
0.097	0.097	0.000	0.194

Table 7: Scores on test data

7 Error Analysis

Traditional RNNs are prone to the vanishing gradient problem, which makes it challenging to capture long-range dependencies.
So we use the GRU and LSTM layers to improve its ability to capture long-term dependencies.
I trained the model on a small subset of the training set which was extracted using random sampling so it might be possible our training subset consisted of similar sentences and our model could not capture or learn complex things.
Also, Our training process has encountered challenges due to the presence of incorrect translation pairs within our training dataset, which has hindered the effective training of our model. These inaccuracies in the training data have had a detrimental impact on the model's ability to learn and perform optimally.

8 Conclusion

We tried the RNN model for translation and found that RNNs can give good results if we set optimal hyperparameters. Our best model gives a score of Test 0.226 in the training phase even though we have inconsistency in our training data and we train on a small subset of training data because of resource constraints.
later we also used sequence-to-sequence model with Attention mechanism for test data . it can focus on specific relevant words using attention mechanism. we get score of 0.194.

References

For model implementation, I refer tutorial from pytorch official documentation TRANSLATION WITH A SEQUENCE TO SEQUENCE NETWORK AND ATTENTION