CS779 Competition: Machine Translation System for India

Aniket Saha
21204403
{aniketsa21}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

Machine translators are very important and state of the art machine translators are available for English and European languages. When it comes to developing machine translators for Indian languages to English and vice versa it becomes very challenging because of the richness in the Indian languages and scripts. In this phase 2, I have experimented with encoder-decoder architectures based on GRU with attention and then with transformer based architecture. The main challenge was the limited computation for training the model given the fact that transformer models are computation hungry. This reports describes in detail, the experiments and challenges faced during phase 2.

1 Competition Result

Codalab Username: A_21204403
Final leaderboard rank on the test set: put the rank
charF++ Score wrt to the final rank: put the score
ROGUE Score wrt to the final rank: put the score
BLEU Score wrt to the final rank: put the score

2 Problem Description

The problem which has been assigned to us as an assignment for the course CS-779 is machine translation for Indian languages. This is the phase 2 of the problem statement where the task is to come up with a machine translation system which can translate from seven Indian languages to English. There are existing architectures that perform well on English to European languages such as English to French, English to German, English to Spanish, and vice versa, but machine translation proves particularly challenging when it comes to English to Indian languages due to the richness of the Indian languages. The source languages for this assignment are Hindi, Bengali, Gujarati, Tamil, Kannada, Malayalam, Telugu. Some of the challenges for Indian to English languages are code mixing (eg. Using few English words in Hindi), complex morphological structure, variation of the same language with change in geography and demography which translates to lack of dataset and less standardization in the dataset. I have approached the problem by experimenting initially with a GRU based encoder-decoder architecture with attention then transformer based encoder-decoder architecture.

3 Data Analysis

The train dataset (JSON file) given to us contains 401243 data points. The analysis has been done on the Hindi and Bengali translations. I have observed that a lot of the translations in both the languages are not proper translation but an inference from the source language. For example we consider the English-Hindi data the the translation for "yaha pahuch kar mujhe laga mai Kumauni se bahar nahi aaya hoon" is given as "Cassettes of Kumauni songs were playing in the house ." Another observation which was made is that the English translations are poor in many of the cases. For example "Mukesh bonds with an undertaker named Johnny over their mutual love of chess." is grammatically wrong

Language Pair	Number of source target pairs (Train)	Number of source target pairs (Val)	Number of source target pairs (Test)
English-Bengali	68848	9835	19671
English-Gujarati	47482	6783	13567
English-Hindi	80797	11542	23085
English-Kannada	46794	6685	13370
English-Tamil	58361	8337	16675
English-Telugu	44904	6415	12830
English-Malayalam	54057	7722	15445
Total	401243	57319	114643

and it should be "love for chess." Moreover in multiple cases English words and western digits (for writing numbers) have been used. There are spelling mistakes both in source and target sentences in the training data both in Bengali and Hindi lines.

While analysing the Bengali to English dataset I came across a lot of ill translations, spelling mistakes in Bengali words and grammatically incorrect translations. For example the very first source target pair has a very very wrong translation. The second pair is grammatically wrong. The fifth pair has the first Bengali word in the source sentence wrong. In fact in the 8th sentence they have included Bengali word from Bangladeshi Bengali which is very different from the Indian Bengali.

The table gives the entire statistics of the train, validation and test dataset.

4 Model Description

I have experimented with two model architectures. The first one is GRU based encoder-decoder architecture with Bahdanau attention[1] and the second one is transformer [2] based encoder-decoder architecture. Both are based on PyTorch based online tutorial PyTorch GRU PyTorch Transformers.

For the GRU based architecture I did not use any pre-trained tokenizer and used a simple rule based pre-processing and tokenization. The encoder-decoder based architecture consists of GRU based encoder and a GRU based decoder which is useful rather that using a single RNN realising us from the burden of length dependency of a sequence. RNN based encoder-decoder architecture are good for translations as Deep Neural Networks (DNN) are useful when the input and output can be encoded to a vector of a fixed dimension and for a machine translation task the input sequence can be of variable length.

In my experiments I have used a pytorch GRU module (both in encoder and decoder) with a input_size twice the hidden size and and experimented with hidden size of 32, 64, 128, 256. For the Bahdanau attention module a feed forward neural network of three linear layers is used to get the attention scores. The encoded vector with maximum dimension is used for the purpose such that all the attention weights are learnt properly.

The training time required for this GRU based architecture was way less than the transformer based sequence 2 sequence model and the translation results for this architecture were really good.

The transformer based encoder-decoder architecture has been followed from a pytorch tutorial. In this exploration with transformer based architectures I have used a Spacy tokenizer form English language and Indic tokenizer for Indian languages. A custom iterable dataset has been used. The architecture first takes the input and converts input to input embeddings. These embeddings are then combined with cosine positional encoding which gives out the positional information of the input tokens. A mask module is incorporated during the training phase which stops the model from looking into the future words while giving predictions.

I have experimented with an embedding size of 64, 128, 512. The number of attention heads used are 8 and I started with 1 encoder decoder layer then I used 2 encoder-decoder layers and finally 3 encoder-decoder layers. The computing was constrained so I could not experiment with larger encoder-decoder layers.

5 Experiments

Starting with the GRU based architecture, the pre-processing was done manually. A class Lang was implemented for creating the dictionary. This helper class contains a word counter, a word to index map and an index to word map. List is converted to text files to give the model. The text data is

taken from Unicode to ASCII. For this model sentences with max length of 15 words are selected. Adam optimizer is used along with a learning rate of 0.01, then 0.001 and then 0.0001. staring from 5 epochs the model have been trained for 15 epochs, then 70 epochs (from scratch) then 100 epochs (from scratch). Each epoch was taking around 2 minutes. I have tuned the hidden size of GRU with the values 32, 64, 128, 256.

For the transformer based architecture I have used the Spacy tokenizer for English and Indic-NLP tokenizer for Indian based languages. For optimizing the cross entropy loss, the Adam optimizer has been used by me. The learning rates with which I experimented while training different models for different languages are 0.01,0.001, 0.0001. While training a single model for all the languages I used a learning rate of 0.0001. While training different models for different language pairs the loss was decreasing till 7th or 8th epoch and then slightly started increasing. Each epoch took around 9 minutes. While training a single model for all the languages it took around 2hr 20 minutes for each epoch but the loss was decreasing drastically.

6 Results

7 Error Analysis

I made two major error analysis while working on the assignment. While using GRU based architecture I was unable to evaluate the validation data using the model. Later I realized that I did not use any token for unknown words which was creating problem. I did not face the same problem while I was working in the first phase and I do not understand why. The model was always able to translate any random English sentence to a sentence in Indian language. I feel that the vocabulary for English is less that the Indian languages so it was always able to translate from English to Indian language although the translation might be wrong.

While working with the transformer based architecture the major mistake which I came across was using pytorch based train test split function. Logically I thought that since the the pytorch train test split uses two modules for source and target sentences so the model might learn wrong representation and give wrong translations but in reality the model was giving same translation for every sentence in Indian Language. Then I changed train test split function and started using the sklearn train test split function and to my surprise the model giving different translation for different source language sentences.

8 Conclusion

The entire assignment had two phases. Phase 1 and phase 2. The phase 1 dealt with with the machine translation for English to seven Indian languages which were Bengali, Gujarati, Hindi, Kannada, Tamil, Telugu and Malayalam while the phase 2 deals with designing a machine translator for the mentioned seven Indian languages to English. While seq2seq [3] models are good for sequential data like sentences but without attention the performance of these seq2seq models were not that good. This is why I started with attention based models during phase 2 and the translations were good. While transformer based seq2seq models perform very well but they are take a lot of time and computation for training and in contrast the GRU based architecture with attention performs very well and takes very less time for training compared to transformer based seq2seq architecture.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.