Competition: Hindi to English Machine Translation System

Arpit Agarwal
180139
{arpitagr}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

This is the report of a NMT project implemented for the course CS779 under Prof. Ashutosh Modi. The project requires to make a NMT model for hindi to english translation. The model described in the report uses a 2 layer LSTM Seq2Seq model trained using Adam Optimizer. The final BLEU score of the model was 0.0217 and course rank of 44.

1 Competition Result

Codalab Username: 180139
Final leaderboard rank on the test set: 44
METEOR Score wrt to the best rank: 0.182
BLEU Score wrt to the best rank: 0.0217
Link to the CoLab notebook: `https://colab.research.google.com/drive/1pm4o5Gdf7USP2qnbo6as7ogn9U__PAPS?pli=1#scrollTo=wyY0jGcSwyk5`

2 Problem Description

The problem statement is about building a neural machine translation model which translates hindi statements to english. We are provided with a test data set requiring pre-processing which basically consists of around 100 thousand pairs to hindi and corresponding english text strings.

3 Data Analysis

1. Training dataset consists of 1 lakh pair of hindi and corresponding english sentences.
2. Noises in the training corpus can be summarized as:
   - Some data points in hindi column consists of english sentences and vice versa.
   - Some data points consists of partial hindi and partial english sentences in both the columns.
   - Some data points contain characters which are neither in hindi nor in english.
   - Some data points are just blank
3. On the other hand test data consist of only hindi sentences but there are several data points in test data consisting of empty strings.

4 Model Description

Model implemented for the final phase is a 2 layer LSTM Seq2Seq model. [1]

4.1 Introduction

We will be using an encoder decoder model which makes use of recurrent neural network. The model basically takes hindi string as input and converts it into a context vector which is a basically a tensor representation of the input string used for translating it. We then use that obtained context vector to evaluate the english sentence. More on the same has been clarified in the subsequent sections.

4.2 Preparing Data

Preparing Data basically involves everything from converting a input string to a tensor which can be given to the model as an input. We do it in the following ways:

1. We first tackle the noises in the training corpus mentioned in the previous section.
2. We then tokenize the hindi and english strings using simple word tokenizer during which we add a <sos> and <eos> token at the start and end respectively.
3. We then build a vocabulary to train our model on. We only use those tokens whose frequency are >= 2. We also keep an unknown token to account for unknown words in the test corpus.
4. We then numericalize the data using the generated vocabulary.

4.3 Building the Seq2Seq model

1. Encoder: The encoder is a 2 layer LSTM model in which model takes an input the embedding into the first hidden layer, output of which are used as input to the second hidden layer.

$$h_t^1, c_t^1 = Encoder(x_t, (h_{t-1}^1, c_{t-1}^1))$$

$$h_t^2, c_t^2 = Encoder(h_t^1, (h_{t-1}^2, c_{t-1}^2))$$

   Here h_t^1 and h_t^2 represent components of t^{th} first and second hidden layer and the same goes for cell states as well whereas x_t represent the t^{th} token.
   Encoder gives as output the values of hidden layer and cell state for the final layer which are the only 2 things required for constructing the final context vector as all the rest of the information in encoded in the context vector itself.

2. Decoder: Similar to the Encoder, the decoder takes as input embedding corresponding to the token one by one and uses h and c values of previous layer to find h and c for the first layer which are provided as an input to the second layer. However note that the output of decoder at each level produces a tensor of dimension of enlish vocabulary which can be used to synthesize the corresponding token using the preconstructed vocabulary and eventually evaluate the final english text.

$$h_t^1, c_t^1 = Decoder(x_t, (h_{t-1}^1, c_{t-1}^1))$$

$$h_t^2, c_t^2 = Decoder(h_t^1, (h_{t-1}^2, c_{t-1}^2))$$

3. Seq2Seq Model: This component recieves the input and output texts, uses encoder to produce the context vectors and then uses the decoder to produce the final english text. A few things to highlight here:
   - We use a technique called teacher-forging ratio which basically means that we assign a probability of teacher-forging ratio to the the actual evaluated token and 1-teacher-forging ratio the the token afterwards.
   - <eos> token is never an input to the decoder.

4.4 Training the Seq2Seq Model

In the paper [1], weights has been uniformly between -0.08 and +0.08 which has performed better as compared to everything else and the same has been done for the model. We then used Adam Optimizer to update the parameters of our model. We calculate gradients using loss.backwards().

5 Experiments

1. Tried using 4 layer and 1 layer LSTM but were giving poorer results as compared to 2 layer LSTM probably because of overfitting and underfitting.
2. Tried using xavier weight initialization method but the one mentioned in paper worked better.
3. teacher-forging method works better as compared to the normal method of computing.
4. Adam Optimization works better as compared to softmax optimizer.
5. Final model on 40 epoches take 14633 seconds.

6 Results

Phase	BLEU score	METEOR
2	0.003	0.016
3	0.0055	0.110
Final	0.0217	0.182

1. 2 layer LSTM works better than 1 and 4 layer for the chosen vocabulary, batch size and optimizer.
2. Adam Optimizer works better than Softmax loss.

7 Error Analysis

We can clearly observe in the predicted English sentences the large number of unknown tokens. This can be easily removed using sub word tokenizer as it would basically be able to translate almost every hindi word to something.

8 Conclusion

We learned to appreciate the practical implementation of machine learning model. One of the most prominent problem in the current implementation is that it produces unknown token for a lot of hindi words. One possible solution to tackle this could be the use of subword tokenization as mentioned in [2]. This would drastically improve the models performance as compared to the status quo. We can also use transformers in place of Seq2Seq implementation which might give a bettwe BLUE score.

References

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *arXiv preprint arXiv:1409.3215*, 2014.

[2] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.