Neural Machine Translation using Bahdanau Attention Mechanism

Navneetha Rajan
PES1201700161
Dept. of Computer Science
PES University
Bangalore,India
navneetha.rajan1999@gmail.com

Sirisha Lanka
PES1201700294
Dept.of Computer Science
Pes University
Bangalore,India
siri181.lanka@gmail.com

Malavikka Rajmohan
PES1201700794
Dept. of Computer Science
PES University
Bangalore, India
malu2809@gmail.com

Abstract*—Machine translation is the task of automatically converting source text in one language to text in another language. In recent years, Neural Machine Translation has emerged as a way of addressing this task. In this project, we explore different configurations for setting up a Neural Machine Translation System for language Hindi. Here, we are building a sequence to sequence translator using the Bahdanau Attention mechanism, to convert hindi sentences to english using various concepts like autoencoders and basic NLP text tokenization techniques

Keywords*—Neural Machine Translation, sequence to sequence translator,Bahdanau Attention Mechanism,Autoencoder,Deep Learning,Machine Learning

I. INTRODUCTION

Given the large number of diverse languages in a country like India, language translation is a key part of communication within various communities present in the country and worldwide. Machine translation helps us in this task . It is particularly useful when you have large amounts of user-generated content that needs to be translated quickly. It also serves extremely in commercial sectors. For example, machine translation makes it possible to quickly translate and review customer reviews, online comments and social media posts thereby increasing the depth of market study. Human translators tend to be more time consuming, exhausting and more expensive.

II. PREVIOUS SOLUTIONS

A. Word to Word Translation

The conventional method followed was where each word was translated to its corresponding word in the translated language. However, this method was ineffective since the grammatical rules followed by different languages aren't necessarily the same. Another issue, faced with this technique was the number of words in the output was the same as the number of words as the given input which does not have to be the case for all languages.

B. Using RNNs

RNNs using the encoder-decoder architecture are used where variable length output from the input may be input. This is possible due to the ability of the model to encode the source text into an internal fixed-length representation called the context vector. Interestingly, once encoded, different decoding systems could be used, in principle, to translate the context into different languages. Although effective, the Encoder-Decoder architecture has problems with long sequences of text to be translated. The problem stems from the fixed-length internal representation that must be used to decode each word in the output sequence.

C. Seq2Seq without Attention Mechanism

Encoder takes input and converts it into a fixed-size vector and then the decoder makes a prediction and gives output sequence. It works fine for short sequence but it fails when we have a long sequence bcoz it becomes difficult for the encoder to memorize the entire sequence into a fixed-sized vector and to compress all the contextual information from the sequence. As we observed that as the sequence size increases model performance starts getting degrading.

III. DATASET

The dataset used for training the model contains 2867 tab de-limited hindi sentences and their corresponding translations in english. This dataset covers almost all types of variability in sentences which are used on a daily basis.

```
Input Language; index to word mapping
1 ----> <start>
356 ----> दुनिया
5 ----> में
418 ----> ऐसी
53 ----> कोई
37 ----> भी
298 ----> चीज़
6 ----> नहीं
11 ----> है
1470 ----> जिसपर
936 ----> सूरज
19 ----> का
2944 ----> असर
98 ----> न
765 ----> पड़ता
102 ----> हो।
2 ----> <end>
```

Fig. 1. Input tokens generated after pre-processing

IV. IMPLEMENTATION

A. *Pre-processing*

1) Added a start and end token to each sentence.
2) Made dataset homogeneous by converting all upper-case letters to lower-case
3) Cleaned the sentences by removing special characters and stop words using the regular expression library
4) Created a word index and reverse word index by applying integer encoding for every token generated in the input language(hindi) and the target language(english) and outputted as tensors
5) Padded each sentence to a maximum length (30 in our case)

B. *Dataset Preparation*

1) Opened the file, strip from beginning and end and finally split the line when it sees line separator("\n")
2) Generated hindi-english word pair separated by tab("\t")
3) Divided the dataset into train and validation sets using 80-20 split rule

The inputted text files are pre-processed and ready to be fed into the model as shown in Fig.1.

C. *Model*

1) Encoder:
The input was put through an encoder model which gave us the encoder output of shape (batch_size, max_length, hidden_size) and the encoder hidden state of shape (batch_size, hidden_size).

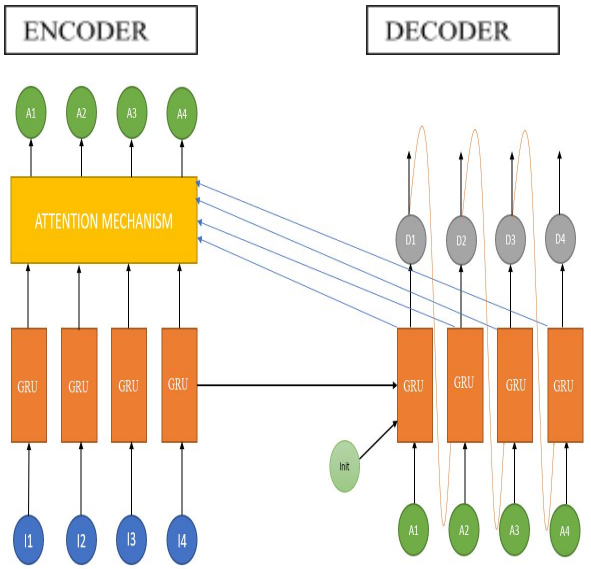

Fig.2. Sequence to Sequence Model with Attention Mechanism

2) Decoder:
A decoder outputs a translation from the encoded vector. Decoder considers input from previous state of decoder and present input from encoder followed by Attention.
3) Bahdanau Attention:
at time ***t*** we consider about ***t-1*** hidden state of the decoder. Then we calculate alignment , context vectors. And then we concatenate this context with hidden state of the decoder at t-1. So before the softmax this concatenated vector goes inside the GRU.
4) Loss Function:
Here, the loss chosen was SparseCategoricalCrossentropy. The reason was because it computes cross-entropy loss when we have 2 or more labels.
5) Optimizer :
Adam Optimizer is used.

The model used in this project is depicted in Fig.2.

D. *Training*

1) Passed the input through the encoder which returned encoder output and the encoder hidden state.
2) The encoder output, encoder hidden state and the decoder input (which is the <start> token) was passed to the decoder.
3) The decoder returned the predictions and the decoder hidden state.
4) The decoder's hidden state was then passed back into the model and the predictions were used to calculate the loss.
5) Used teacher forcing - technique where the target word was passed as the next input to the decoder
6) The final step was to calculate the gradients and apply it to the optimizer and backpropagate.

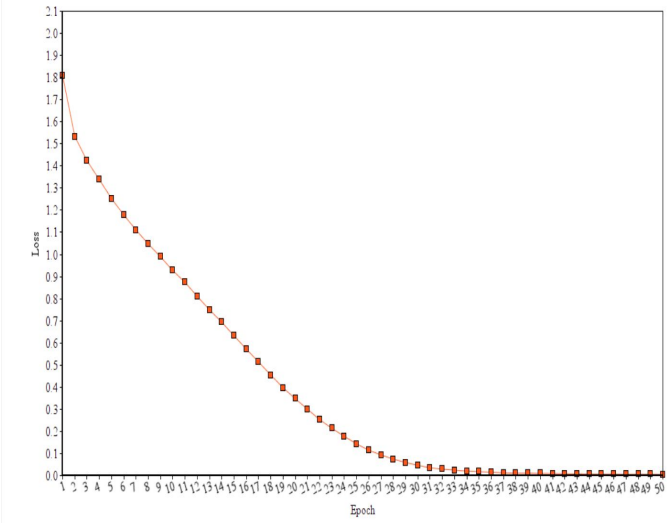

Fig.3. Graph depicting the convergence of the loss function(cross-entropy) to 0. On the x-axis is the number of epochs and on the y-axis is the total loss after each epoch

E. Evaluation

1) The evaluation function was similar to the training loop, except the input to the decoder at each time step was its previous predictions along with the hidden state and the encoder output.
2) Prediction stopped when the model predicted the <end> token .
3) Attention weights were stored at every time step.

F. Equations

1) Input:

$$\{x_i = \text{source}_i\ ,\ y_i = \text{target}_i\ \}^{N}_{i=1} \quad (1)$$

2) Encoder:

$$h_t = \text{GRU}(h_{t-1},\ x_t) \quad (2)$$

$$s_0 = h_T \quad (3)$$

3) Decoder:

$$e_{jt} = V^{T}_{attn}\ \tanh(U_{attn}h_j + W_{attn}s_t) \quad (4)$$

$$\alpha_{jt} = \text{softmax}(e_{jt}) \quad (5)$$

$$c_t = \sum_{j=1}^{T} \alpha_{jt}h_t \quad (6)$$

$$s_t = \text{GRU}(s_{t-1}\ ,\ [e(y'_{t-1}),c_t]) \quad (7)$$

$$l_t = \text{softmax}(Vs_t + b) \quad (8)$$

4) Loss function:

Sum of cross-entropy

$$L(\theta) = \sum_{t=1}^{T} L_t(\theta) \quad (9)$$

$$L_t(\theta) = -\log P(y_t = l_t \mid y^{t-1}_1) \quad (10)$$

```
translate(u"मैं और नहीं चल सकती।")

Input: <start> मैं और नहीं चल सकती। <end>
Predicted translation: i can't walk any further . <end>

translate(u"उसने अपनी गाड़ी बिना हिचकिचाहट हे बेच दी।")

Input: <start> उसने अपनी गाड़ी बिना हिचकिचाहट हे बेच दी। <end>
Predicted translation: she called me a car . <end>
```

Fig.4. Experiment results (a)Top- correct result (b)Bottom- wrong result

V. RESULTS AND CONCLUSIONS

A. Epochs

The model was trained for 50 epochs. 50 was ideal as the loss converged to 0 by the 50th epoch as shown in Fig.3.

B. Translation

Out of 10 randomly sampled input statements, 7 were correctly translated, while 3 were wrong. This proves that the model was not overfitted. Fig.4.(a,b) shows the results of our experiment.

VI. SCOPE OF FUTURE WORK

A. The performance of the model can be increased by increasing the training dataset and using Bi-directional LSTM for better context vector.

B. Use the beam search strategy for decoding the test sequence instead of using the greedy approach (argmax).

C. One drawback of attention is that it's time-consuming. To overcome this problem Google introduced "Transformer Model"

VII. ACKNOWLEDGMENT

We would like to thank Prof. Srinivas K. S. for providing us an opportunity to work on this project as a part of our course of Topics of Deep Learning.

VIII. REFERENCES

[1] NPTEL Online course on encoder-decoder model with attention

[2] towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee

[3] towardsdatascience.com/neural-machine-translation-15ecf6b0b