Improving Neural Machine Translation with Attention and Beam Search

matthew@cinnamon.is

Feb 2020

1 Introduction

So far, we have learnt how to using Seq2seq model to solve the problem of Machine Translation. However, our model is still have much more rooms for improvement. Two significant ways to improve our model's performance is beam search and attention mechanisms. In the next sections, Sec.2 will show how to improve the decoding process of the seq2seq model and Sec.3 will introduce a additional module in our seq2seq model to help with the alignment of texts in translation.

Your task is to implement the strategies mentioned below and show their effectiveness in your report.

2 Beam search

Until now, we have used a greedy algorithm to generate output sequences from the probabilities prediction of Seq2seq model. However, it will not result in a optimal translation for our system. Another way to find the optimal sequence to maximize $argmax_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x})$ is to try all possible $\boldsymbol{y}$ sequences which will make our decoding process extremely show with time complexity of $O(V^T)$.

From those problems, the most common strategy for us to generate output sequence is using *beam search* where we only keep the best k results at a decoding timestep. You can find more details on beam search in Sec. 7.2.3 (Neubig, 2017)

3 Attention mechanism

In the simple version of seq2seq model, we only use last hidden state of the encoder as the *context vector* for the decoding process. However, in translation, we have the intuition that different words of an input have different level of significance on the each output word. The attention mechanism help the decode have a look at the *whole input sentence* at every decoding step.

In this assignment, we will use the Bahdanau style attention (Bahdanau, Cho, & Bengio, 2014). As illustrated in Fig.1, with the input sequence $\boldsymbol{x} =$

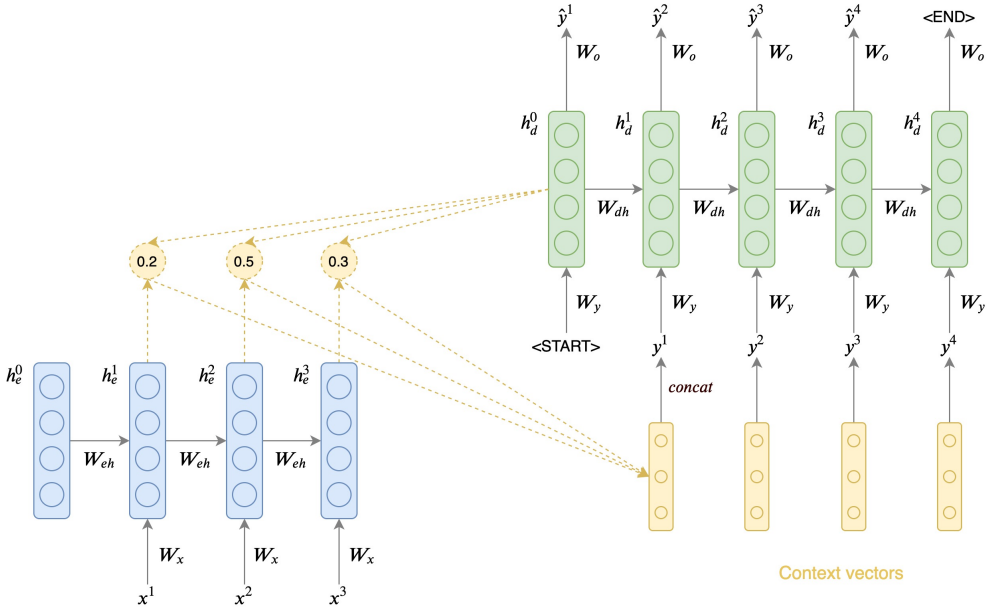

Figure 1: Seq2Seq with Bahdanau attention

$\{\boldsymbol{x}^1, ..., \boldsymbol{x}^T\}$ and the output sequence $\boldsymbol{y} = \{\boldsymbol{y}^1, ..., \boldsymbol{y}^{T'}\}$ the attention vector for decoding timestep t is computed as below:

$$\alpha_{ti} = \frac{exp(score(\boldsymbol{h}_t^d, \boldsymbol{h}_i^e))}{\sum_{i'=1}^{T} exp(score(\boldsymbol{h}_t^d, \boldsymbol{h}_{i'}^e))} \qquad \textit{Attention weights} \qquad (1)$$

$$\boldsymbol{c}^t = \sum_i \alpha_{ti} \boldsymbol{h}_i^e \qquad \textit{Context vector} \qquad (2)$$

where the $score(.)$ function can be modeled with a fully-connected layer with a non-linear activation. The context vector will be concatenated with the input word embedding of the next timestep and then fed to the decoder RNN cell.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Neubig, G. (2017). Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*.