Attentional German-English Neural Machine Translation Model

Xin Qian
xinq@andrew.cmu.edu

Abstract

This report describes the implementation detail and experimental analysis on the dynet-based attentional neural machine translation model. Experiment was performed on the IWSLT2016 workshop for German-English translation. With a hidden dimension of 512, embedding dimension 512 and attention size of 256, we achieved a BLEU score of 27.042 (46.02% improvement over reference baseline) on the development set and a BLEU score of 27.143 on the test set.

1 Introduction

Maintaining a referece input word vector around at each decoding step, attentional Neural Machine Translation (NMT) model resolves long-distance dependency problem between words within both single direction encoder-decoder or bidirectional encoder-decoder model. The model also overcomes the difficulty of encoding arbitrary length sentences into fixed size of hidden vector.

As one of the major advantages of Dynet, minibatching avails fast-training for training our attentional Neural Machine Translation model, both on CPU and GPU.

2 Dataset Specification and Statistics

Experiment was done over the dataset from the IWSLT2016 workshop for German-English translation. The dataset was divided into into three batches, including 90K parallel lines of German-English sentences as a training set, 887 lines as a validation set, and a 1.5K lines as a test set. An additional German blind test without English translation was also given. Each batch has at most three pre-processed versions, a NLTK tokenizer preprocessed tokenized version (.tok), a simple lowercased version(.low) and a sentence length filtered version (.filt).

3 Technical Detail and Experimental Analysis

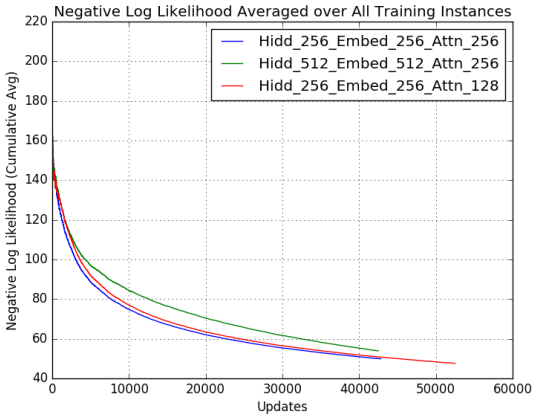
3.1 General Pipeline

The general pipeline divided into the training module and the test module. At the start of both module, training dataset was read to build a parallel vocabulary dictionary. Each word is associated with an index for future embedding lookup operation and a reverse dictionary is also made available for translation from index. Training dataset was trunked into fixed-length batches.

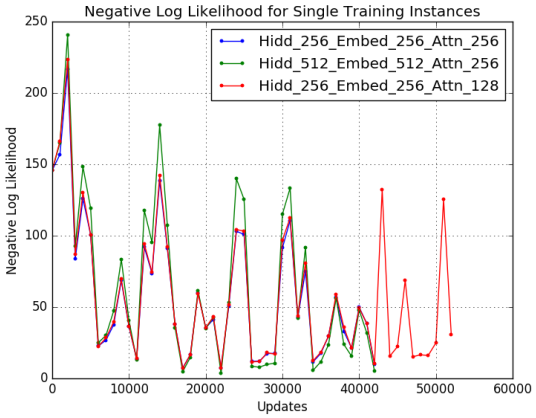
After looking up word index in each position in each sentence to create, we feed the embeddings into two different encoders, one forward and one backward. Resulting vectors are simply concatednated together until it becomes the last output as to become the initial vector for the decoder. This requrired us to add a parameterized hidden layer to convert to half of the size of encoder output vector. Along the decoding process, we calculated attention score based on the previous hidden state and previous context in each timestep. A softmax weighted attention vector allows the calculation of the context vector at current timestep.

At decoding stage, we read out a embedding size vector at each word position. A tanh function followed after affine transformation is performed on the hidden vector, context vector and current input word produced a vector of embedding size. after an softmax softmax on the affine transformation of the readout vector, we obtain the probability of a word suits the scenario.

The training process has a default setting of batch size 32 (3K updates per epoch). We examine the BLEU score (our primary evaluation metric) every 2500 update and for every update we keep track of the negative log likelihood and single/average perplexity for monitoring purpose, as plotted in Figure 1, 2 and 3.

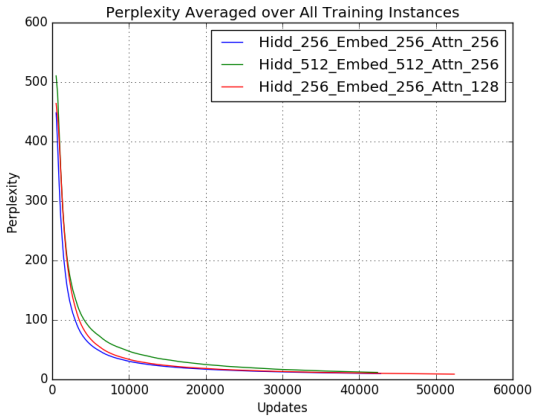


(a) Negative Log Likelihood Cumulatively Averaged

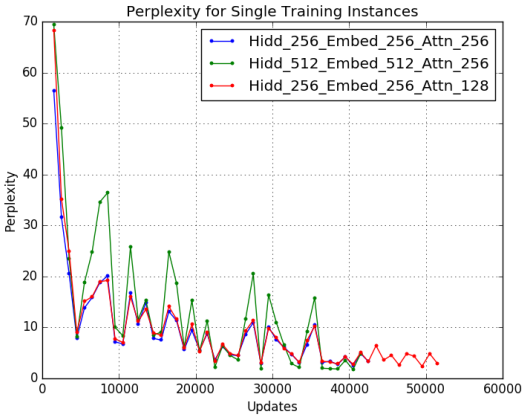


(b) Negative Log Likelihood Sampled over each 1000 Update

Figure 1: Negative Log Likelihood throughout Training



(a) Perplexity Cumulatively Averaged



(b) Perplexity Sampled over each 1000 Update (Starting from 1500^{th} Update)

Figure 2: Perplexity throughout Training

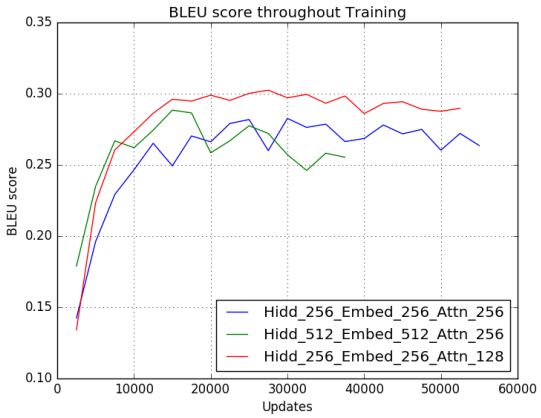


Figure 3: BLEU score throughout Training

Table 1: Impact of Beam Size over BLEU score, H=E=512, A=256

Beam Size	1	3	5	7	10
Dev Set	0.24893	0.26776	0.26990	**0.27040**	**0.27042**
Test Set	0.24604	**0.27077**	**0.27143**	0.26861	0.26902

3.2 Mini batching with fixed length

Despite that in reference implementation, we are encouraged to sort sentences by length and evenly trunk sentences into batches. We applied a stricter trunking approach to further reduce coding and computation complexity. We divide source sentences into different length bins and further divides sentences into fixed size of batches, this step eliminates unnecessary padding within batch and therefore reduces computation complexity behind the scene.

3.3 Dropout

Dropout is one of the effective technique for addressing the problem of parameter overfitting by randomly drop units from the neural network during training. With integrated Dynet dropout functionality, we apply dropout with a fixed dropout rate as 0.5 from the empirical practice.

3.4 Beam Search

At translation stage (both during validation within training and throughout the testing stage), we applied beam search as our translation generation scheme as an alternative to greedy search (a special case where beam size = 1), as greedy search does not guarantee to find the most probable translation. Table 1 demonstrates results from different beam size. Here a beam size of 5 wins on test set while a beam size of 10 wins on the development set. This mismatch could be due to test/dev sample difference, or it might be because of the length bias, that increasing beam size will early stop hypothesis searching after finding enough short sentences. We selected 5 as our final choice in view of the larger sample size in test set than in the development set (Section 2.). Future work will be to by introducing a prior probability from the length of source sentence into the hypothesis scoring.

3.5 Impact of parameter dimension

Figure 1, 2 and 3 demonstrates the difference of using different parameter dimensions. Cumulative average of both negative log likelihood and perplexity drops during training and converges at a small value. Individual negative log likelihood and perplexity have an overall trend of decreasing but cannot avoid fluctuation. This could be due to difference in each batch in terms of source sentence length and difficulty. From figure 3, we see that the parameter setting of Hdim=Edim=256 and Attn_size=128 achieves the highest BLEU score during training.

3.6 End of Training

Our final output was obtained with early stopping scheme after 10 times of BLEU score non-increasing attempt. The model after 18 epoches stopped at an average negative log likelihood of 43.981552 (over all updates and all lines within each batch).