11731 Assignment-1 Report: Seq2Seq Machine Translation Model

Hongliang Yu, Keyang Xu
Language Technologies Institute
{hongliay, keyangx}@andrew.cmu.edu

1 Overview

In this assignment, we created a German-English neural machine translation model using Seq2Seq model with some modifications, including attention mechanism, bidirectional encoder, minibatch training, first-word translation and beam-search. Compared with simple Seq2Seq model, we achieve a significant improvement for BLEU score on the IWSLT test dataset.

The implementations can be view on our Github: `https://github.com/rivercold/MT-Seq2Seq`. The folder "assign1" places all our codes. And the "output" folder contains our results for official evaluation on "test" and "blind" sets.

2 Methods and Implementations

In this section, we will describe our specific implementations for the Seq2Seq model. Our implementation is based on the open-source package DyNet and Stochastic Gradient Descent(SGD) is utilized for optimization.

2.1 Basic Implementations (Hongliang, Keyang)

We implement a vanilla sequence-to-sequence translation model with LSTM [2] encoder-decoder. The size of hidden state is set as 128 and the size of look up embeddings is 200. In order to increase the accuracy of translation, we use a stacked version of LSTM where the number layer is 2.

2.1.1 Attention Model (Hongliang, Keyang)

We further implement a Seq2Seq model with attention mechanism [1]. A multi-layer perceptron (MLP) is trained to compute the attention score between the i^{th} hidden encoder state vector $h_i^{(f)}$ and the j^{th} hidden decoder state vector $h_j^{(e)}$ as:

$$attn\_score(h_i^{(f)}, h_j^{(e)}) = w_2^T \tanh(W_1^T[h_i^{(f)}; h_j^{(e)}]) \tag{1}$$

where W_1^T and w_2 are the weight matrix and vector for the first and second layers of MLP.

With the attention score at each position, we get the context vector for t^{th} state decoder as a weighted average of all encoder hidden states. For the decoder, we concatenate the context vector c_t with the last hidden state vector h_{t-1} to compute the hidden state vector of current state: $h_t^{(e)} = dec([embed(e_{t-1}); c_{t-1}], h_{t-1}^{(e)})$.

10807 Class Project (2016).

2.2 Improvements

2.2.1 Bi-directional Encoder (Hongliang, Keyang)

To have a better encoding model, we implement the bi-directional encoder. Namely, each token in the target sentence is represented into the concatenation of two LSTM outputs: $h_i^{(f)} = [\overrightarrow{h}_i^{(f)}; \overleftarrow{h}_i^{(f)}]$.

2.2.2 Minibatch Implementation (Hongliang)

As bi-directional attention model is computationally expensive, we use the minibatch trick to speed up the algorithm.

Padding In the minibatch optimization, we consider the general case that sentences are not the same lengths. Compared with the padding tricks in vanilla bidirectional encoder-decoder model, the main challenge is the *alignment* problem in bidirectional attention model.

The normal way for padding bidirectional encoder-decoder model is to add `</S>` at the end of the forward LSTM while add `<S>` at the beginning of the backward LSTM. However, the attention model requires two LSTMs align in each step for concatenating $\overrightarrow{h}_i^{(f)}$ and $\overleftarrow{h}_i^{(f)}$. Assume the maximum length of the batch is 10, and we pad a sentence with length 6. The following is an example of the misalignment problem:

```
Forward Padding:  t1  t2  t3  t4  t5  t6  </S> </S> </S> </S>
Backward Padding: <S> <S> <S> <S> t1  t2   t3   t4   t5   t6
```

In our implementation, for both forward and backward LSTMs, we pad the sentence at the beginning and the end. Formally, assume the sentence is represented as the sequence of the token $s = \{t_1, \cdots, t_L\}$, and the maximum length of the batch is M. We pad the sentence as $\{\texttt{<S>}_1, \cdots, \texttt{<S>}_{\frac{M-L}{2}}, t_1, \cdots, t_L, \texttt{</S>}_1, \cdots, \texttt{</S>}_{\frac{M-L}{2}}\}$. The subscripts of `<S>` and `</S>` are used for counting. Therefore, the padding for the example above becomes:

```
Forward Padding:  <S> <S> t1  t2  t3  t4  t5  t6 </S> </S>
Backward Padding: <S> <S> t1  t2  t3  t4  t5  t6 </S> </S>
```

Sentence Length-based Sorting and Shuffling To avoid the inefficiency of padding of length variations within the batch, we do the length-based minibatch optimization. We use two tricks here to avoid the length bias:

(1) Instead of sorting the target sentences by length, we add a second key, a random number, in sorting. We found that, if we sort the sentences by length, the sentences with the same length would be sorted with the alphabet order. This reduce the diversity within a batch, because sentences with similar prefixes are gathered. Adding a second key rather than sentence length, the sentences with the same length are actually shuffled.

(2) After step (1), we can partition the minibatches according to the pre-defined batch size, and shuffle the batches. In this way, we avoid the bias that the optimizer meets the short sentences first and falls in the the local optimum.

2.2.3 First Word Translation (Hongliang)

The translation of the first word is important. Instead of feeding in `<S>` as the first token of the decoder, we project the output of the last step of the encoder to the decoder. We argue that such representation is more informative than `<S>`. Formally, we define:

$$embed_0^{(e)} = W_e h_{|F|}^{(f)}, \quad h_0^{(e)} = W_h h_{|F|}^{(f)} \tag{2}$$

where W_e and W_h are matrices that project $h_{|F|}^{(f)}$ to the required dimensionality.

2.2.4 Beam Search (Keyang)

We employ a beam search to improve the greedy search for the decoder, wehere we set beam wdith equals to 3. At each step of decoder, we will keep 3 states with highest log probabilities. Once there

predicts a </S> token in these states, we will save the sequence with its corresponding log probability and then reduce the beam width by 1. In order to overcome the length bias to short sentences of beam search, we take divide the log likelihood by the sequence length. Finally, we output the sequence with the highest average log probability as follows:

$$\hat{E} = \arg\max_{E} \log P(E|F)/|E| \tag{3}$$

3 Evaluations

		overall-BLEU	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Validation	Beam	16.95	53.2	24.4	12.3	6.2
	Greedy	**19.32**	52.9	25.6	13.7	7.5
Test	Beam	17.57	52.9	24.5	12.2	6.3
	Greedy	**19.98**	53.6	26.2	14.2	8.0

Table 1: BLEU scores on validation and test sets. The overall BLEU score is calculated as $BLEU = brev\_penalty \cdot \exp(\frac{1}{4} \log \sum_{i=1}^{4} BLEU_i)$

We train the sequence-to-sequence model with all our modifications on "*train.en-de.low.filt.de*" and "*train.en-de.low.filt.en*" files for 40 epochs with simple SGD. Table 1 reports our performances on validation and test sets separately, evaluated by BLEU scores. Compared with the baseline model, which achieves 18.6% BLEU score on test set, we improves it by 7.42% on the same set.

However, we surprisingly find that utilizing beam search drops the BLEU scores on both sets. Especially, beam search performs poorly in terms of higher order metrics (BLEU-3 and BLEU-4). We guess that this is caused by a bad strategy of normalizing length bias. Due to the lack of time for this assignment, we will keep improving beam search in our further studies.

4 Work Division

Hongliang: Basic Implementation, Attention Model, Bi-directional Encoder, Minibatch Implementation, and First Word Translation

Keyang: Basic Implementation, Attention Model, Bi-directional Encoder, and Beam Search

Conclusion

In this assignment, we finish a sequence-to-sequence model for the German-English translation task. After completing a vanilla LSTM encoder-decoder model, we further modify the model with attention mechanism, bi-directional encoder and first word translation, which proves to be effective in increasing the BLEU scores on both valid and test sets. Also, we implement a minibatch training for our Seq2Seq model, which enhances the speed of training and is more efficient for GPU environment. Although we explore leveraging beam search on the decoder for translation, we observe no improvement on aspect of the BLEU score.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.