Implementing Bdahdanau Seq2Seq with Optimizations

Mingyang Zhou and Kevin Jesse

August 17, 2018

1 Introduction

In recent years, sequence to sequence neural model has been proved to be a powerful method for machine translation task. It is a large recurrent neural network, which consists of a encoder to model the source language and a decoder to generate the translation in the target language. This method is originally proposed by Sustskever[11] and Cho[2] in 2014. Later, Bdahdanau[1] optimizes the original sequence to sequence model by proposing a learnable attention mechanism that will learn to focus on an aligned word from source sentence when generate a translation word in the decoder. After that, various researches has been applied to improve Bdahdanau's approach, but the most common structure that is adopted by most of the neural machine translation research is still Bdahdanau's attention based sequence to sequence neural model.

For the final project of Stats 208, we implemented the Bdahdanau's sequence to sequence neural model with various optimizations including using conditional GRU for decoder, better beam search decoding design, and better training process design. We will discuss in details in the following sections

2 Method

Given a set of parallel sentences in language X and Y, the model aims to translate sentences $\{x_i\}_{i=1}^{N} \in X$ in language X to sentences $\{y_i\}_{i=1}^{N} \in Y$ in language Y. We first apply encoder of our neural model to take in the source sentences and learn a representation vector. Then, we use the decoder to predict the translation of the source sentence word by word with the help from the learned representation. We will introduce in details how the encoder and decoder worked in the following sections.

2.1 Encoder

We first encode an N-length source sentence $\{x\}$ as a sequence of tokens $x = \{x_1, x_2, \ldots, x_n\}$ with a bidirectional RNN [9]. Each token is represented by aone-hot vector, which is then mapped into an embedding e_i through a embedding matrix. The bidirectional RNN processes the enbedded tokens in two directions: left-to-right (forward) and right-to-left (backward). At every time step, the encoder's RNN cell generates two corresponding hidden state vector: $\overrightarrow{h_i} = \overrightarrow{\text{RNN}(h_{i-1}, e_i)}$ and $\overleftarrow{h_i} = \overleftarrow{\text{RNN}}(h_{i-1}, e_i)$. The two hidden state vectors are then concatenated together to serve as the encoder hidden state vector of the source token at step i: $h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$.

2.2 Decoder

We initialize the decoder with the mean of encoder hidden states which is the representation of the source sentence:

$$s_0 = \tanh(W_{init} \frac{1}{N} \sum_{i}^{N} h_i) \tag{1}$$

At each time step j, the decoder generates a decoder hidden state s_j from a conditional GRU cell. We follow Nemantus toolkit to implement the conditional GRU cell with attention[10].

The conditional GRU has a more complex internal structure than a normal GRU decoder, which consists of two GRU layers and an attention mechanism. At layer cGRU_1, we generate a intermediate hidden state $s_j^{'}$ from the hidden state vector from the previous step s_{j-1} and the embedding of the previously decoded word y_{j-1}:

$$s_j^{'} = \text{cGRU}_1(s_{j-1}, y_{j-1}) \tag{2}$$

Then, at the attention mechanism, we learn the context vector $c_j = \sum_{i=1}^{N} \alpha_{ji} h_i$, which capture the relevant words in the source sentence that the decoder should focus on when generating current translated token y_j. The weight α_{ji} is computed as a softmax of a score value p_{ji}, which is computed from previous decoder hidden state d_{j-1} and the set of encoder hidden state vectors. v_a,W_a, and U_a are learnable parameters associated with attention mechanism to compute the appropriate e_{ji}.

$$\alpha_{ji} = \frac{\exp(p_{ji})}{\sum_{l=1}^{N} \exp(p_{ji})} \tag{3}$$

$$p_{ji} = v_a . \tanh(W_a d_{j-1} + U_a h_i) \tag{4}$$

Finally, with layer cGRU_2, we generate s_j from the intermediate representation $s_j^{'}$ and the learned context vector c_j from the attention mechanism.

$$s_j = \text{cGRU}_2(s_j^{'}, c_j) \tag{5}$$

From the hidden state s_j that we generate from the conditional GRU, we can predict the conditional distribution of the next token y_j with a fully-connected layer W_o given the previous token's language embedding e_{j-1}, the current hidden state d_j and the context vector for current step c_j:

$$p(y_j | y_{j-1}, x) = \text{softmax}(W_o o_t) \tag{6}$$

where $o_t = \tanh(W_e e_{j-1} + W_s s_j + W_c c_j)$. The three inputs are transformed with W_e, W_s, W_c, respectively and then summed before being fed into the output layer.

The objective function for our neural model is a cross entropy loss function: $L_T = -\sum_j \log p(y_j|y_{j-1}, x)$

3 Experiment

In this section we are going to explain how we evaluate the implementation and how we make different design options. We will also share the training details.

3.1 Dataset and Evaluation Metrics

We applied our neural translation model to a small spoken domain translation dataset IWSLT 2015 English to Vietnamese dataset which consist about 130K training data, 2K validation data, and 2K test data. We pick this small dataset so that we can quickly evaluate the performance of our model and attempt different hyperparameters.

We use the Automatic Evaluation metrics BLEU[7] to evaluate our model, which is the most common evaluation metrics applied in the research field of machine translation.

	stanford nmt	our model
BLEU	23.3	**25.1**

Table 1: The BLEU score comparison between stanford neural machine translation system with our system.

3.2 Training Details

In this subsection, we share details on the hyper parameter setting to train our model. The word embedding size for both encoder and decoder are set to 128. The Encoder is a bidirectional recurrent neural network with GRU[3], which has a hidden size of 256. The decoder shares the same size as Encoder.

During training, we use Adam [4] to optimize our model with a learning rate of $4e-4$. The batch size is 32. We clip the gradient norm to 1 to avoid explosive gradient problem for recurrent neural network[8]. Dropout is applied at the embedding layer in the encoder, context vectors extracted from encoder and the output layer of decoder, where the dropout rates are set as (0.3,0.5,0.5) respectively. We initialize all the weight parameters with Kaiming's method[5]. During training, we also apply learning rate decay by a factor of 0.2 whenever the validation BLEU score does not increase for 10 evaluation rounds. We evaluate our model every 1000 iterations.

3.3 Result and Design Option Discussion

We compare the BLEU score of our model to stanford neural machine translation system [6] on IWSLT 2015 dataset. The comparison is summarized in Table 1.

It is observed that our model has outperformed Stanford system by a large margin of 1.8 Bleu score for a single run. We think this advantage is resulted from a couple of optimization that we have applied to our model. The first optimization is better beam search decoding to generate translation. A known issue for beam search decoding is that it will favor shorter sentences over longer sentences. We apply length normalization to resolve this problem i. Additionally, we force that no repeated words should be generated at consecutive positions to relieve the repetitive word generation problem in recurrent neural network. The second optimization is using an adaptive learning rate decay strategy, where we decay the learning rate until we didn't get improvement on BLEU score. The third optimization is that, we occasionally simulate the test stage translation process during training, where we use the predicted token from the previous step to generate the next word. This training strategy helps to prevent overfitting on training dataset.We validate each of the design option by disabling them, and compare the performance with the final model. gs means that we use greedy search decoding instead of beam search decoding. no-decay means that we disable the learning rate decay. full-gt means that we always use the ground truth to guide the translation during the training stage. The result is summarized in Table 2.

	gs	no-decay	full-gt	Our Model
BLEU	17.7	22.2	23.9	**25.1**

Table 2: The BLEU score comparison to validate different design options: beam search decoding, learning rate decay and applying the test decoding process during training

As can be seen, using greedy search decoder leads to a big drop of the performance, compared to using beam search decoder. Training our model without learning rate decay will lead to a smaller decrease of the performance, but it is still quite noticeble compared to the BLEU score of the full model. It seems that using full groundtruth to guide the training has the smallest impact on the performance of the model, but the result is still worse than the full model. With this result, we can see that all the optimization that we have applied to our neural model is valid and help improve the translation result.

4 Conclusion

We implementation the sequence to sequence neural model introduced by Bahdanau and apply it to IWLST 2015 Englisht to Vietnam dataset. Our implementation outperforms Standford neural machine translation system by a noticeable margin due to the optimization on the decoder and training process.

References

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

[2] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[3] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[6] Minh-Thang Luong and Christopher D. Manning. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam, 2015.

[7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[8] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[9] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997.

[10] Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. Nematus: a toolkit for neural machine translation. *CoRR*, abs/1703.04357, 2017.

[11] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.