Neural Machine Translation from Chinese to English

Yan Sun yas108@ucsd.edu
Yu Shen yus122@ucsd.edu
Xin Li xil222@ucsd.edu
Fangyu Liu fal053@ucsd.edu

Abstract

Neural machine translation(NMT) is an technique that uses neural network to predict the likelihood of a sequence of words. In our project, we implemented a NMT with various tricks such as attention, teacher forcing, etc to see how those tricks can improve the performance of NMT. In addition, we also did experiments about how hyper-parameter can influence the NMT.

1 Introduction

Machine Translation is one of the hot topics in the area of linguistic and artificial intelligence, which requires to build a model for translating a sequence of text from one language to another. Usually, there is not one absolutely best translation for a given sentence due to the ambiguity and flexibility of human language. Initially, some statistical machine translation (SMT) technique is utilized for this task such as modelling by Hidden Markov Model (Vogel et al., 1996). As the development of neural networks, Neural Machine Translation (NMT) techniques are also introduced to translation task. Different from traditional statistical machine translation model, the neural network could train a single model from input text and label directly without tuning the sub-components inside the traditional phrase-based systems (Bahdanau et al., 2014). Nowadays, one of the most common used neural networks is Recurrent Neural Network (RNN) and its improved version such as Gate Recurrent Unit (GRU) (Chung et al., 2014) and Long Short-term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997).

In this project, Bidirectional GRU combined with attention technique is used as primary architecture for translation from Chinese to English. Attention is used to improve the performance of decoder. Teacher forcing trick is used for efficient training. Beam Search technique is implemented for model inference and Bilingual Evaluation Understudy (BLEU) is used for model evaluation.

2 Method

Neural Machine Translation (NMT) is a sequence to sequence task. The encoder-decoder model is utilized for this task. Specifically, bidirectional GRU is used for building the model for translation. Although LSTM may performs better than GRU, GRU cell is used in consideration for the training efficiency. For further consideration for optimization, bidirectional model is selected for implementation. Bidirectional GRU allows the model to look at both directions at the time for encoding for each word to resolve possible ambiguity caused by only looking at previous step. Apart from these approaches, more optimization tricks are also utilized for training, inference and evaluation, including teacher forcing, attention decoder, beam search and Bilingual Evaluation Understudy score.

2.1 Teacher Forcing

Teacher forcing(Williams and Zipser, 1989) is one technique to accelerate the training process. During the training process, if the prediction of one word if bad and it is used for the following input, the error will be accumulated and the prediction for the following prediction will go far away from correct ones. After applying teacher forcing, there would be probability that the correct prediction will be used for the next step input in order to avoid the accumulation of prediction error, which also accelerate the training process. The probability to use teacher forcing will also be a hyper-parameter for the configuration.

2.2 Attention

In the paper (Bahdanau et al., 2014), attention mechanism was first applied to the task of NMT. Prior to the development of attention mechanism, one of the most common problems for sequence to sequence task is caused by the sequence length. The performance for most models will go down as the increase of sequence length (Vaswani et al., 2017). To solve this problem, adding attention mechanism to the decoder is an effective solution. In the progress of sequence-to-sequence and attention, the first step is to encode the text by embedding and then feed the embedding into Bi-GRU to get the encoder outputs and hidden states h_t. h_t can be calculated as following:

$$\begin{aligned} h_f^t &= f(h_f^{t-1}, x_t) \\ h_b^t &= f(h_b^{t+1}, x_t) \\ h_t &= [h_f^t, h_b^t] \\ h_t &= f_e(x_t, h_{t-1}) \end{aligned} \tag{1}$$

f_e indicates the process of encoder using Bi-GRU. h_f means the hidden states for the forward direction in Bi-GRU while h_b indicates the hidden states in the opposite direction. The second step is to decode encoder outputs and get prediction sequence. In order to get a sequence of word, we can use this formula to generate one word each time by iteration.

$$\begin{aligned} p(y_t|y<t,x) &= f(y_{t-1}, s_t, c_t)) \\ s_t &= f_d(y_{t-1}, s_{t-1}, c_t) \\ c_t &= \sum \alpha_{ij} h_j \\ \alpha_{ij} &= softmax(e_{ij}) \\ e_{ij} = a(s_{t-1}, h_j) &= v_a^T \tanh(W s_{t-1} + U h_j) \end{aligned} \tag{2}$$

f is the prediction process, including dimension mapping and an activation function. y_{t-1} is the predicted word vector of previous step, which will be used as input of current step with the context c_t. s_t is the hidden decoder state of current sequence. c_t is a weighted average of h_j, which is called attention distribution. The weight coefficient is α_{ij}. v_a, W and U are trainable parameters. This is just a general type of attention mechanism. There exist many other improved version of attention but the general one is utilized here.

2.3 Beam Search

Beam Search (Freitag and Al-Onaizan, 2017) is a search algorithm mostly used in extending a node and finding the most promising nodes in a limited set. From this paper fill this in, we noticed Beam Search is an efficient algorithm in improving translation integrity as it considers the Topk possible sentence segments instead of a single word as a factor in predicting the next word. Therefore, sentences translated as a whole has less grammar mistakes compared to baseline greedy translation model.

2.4 Bilingual Evaluation Understudy

Bilingual Evaluation Understudy(Papineni et al., 2002) is a method for automatic evaluation for machine translation. Compared with human evaluation, BLEU is quick, inexpensive and language-independent. By using BLEU, we can measure the closeness between machine's output and professional human translation, which is called the quality of text translated by machine. In the algorithm of BLEU, the first step is to build modified forms of precision between the candidate translation and many reference translation from unigram to n-gram. And using the average logarithm with uniform weights is to combine all the modified n-gram precision. Moreover, when candidate translations are shorter than their reference translations, an exponential brevity penalty factor will be added. The brevity penalty BP is

$$BP = \begin{cases} 1 & \text{if } c > r \\ \exp(1 - r/c) & \text{if } c \leq r \end{cases} \tag{3}$$

c is the length of the candidate translation and r is the effective reference corpus length.

$$BLEU = BP * exp(\sum_{n=1}^{N} w_n \log p_n) \tag{4}$$

As we mentioned before, $w_n = 1/N$ and we use $N = 4$.

3 Data

The dataset used in this project is the pre-segmented Chinese-English corpus obtained from Xinhua News Agency and People's Daily newspaper, for which the primary content is about economic, political and cultural news. There are 100 thousand sentences in the original dataset.

3.1 Pre-processing

The pre-processing consists of two steps, trimming sentences and building vocabularies. As

lr	hidden layer	teacher forcing ratio
0.0001	64	0.5

Table 1: Default parameters

epoch lr	1	2	3	4	5
0.0001	7.89	6.65	6.61	6.53	6.47
0.001	6.62	6.02	5.68	5.46	5.30
0.01	6.77	6.51	6.15	5.85	5.67
0.1	8.08	8.45	8.38	8.44	8.87

Table 2: Loss value of different lr in first 5 epoch

for trimming sentences, sentences with number of words between 3 and 50 are selected as the data used for training and validation in order to make the task more specific and accelerate training process. After trimming the sentences, vocabulary is built for the word that appeared more than once. In addition, the start of sentence sign $< sos >$, end of sentence sign $< eos >$, padding sign $< pad >$ for different sequence length and unknown word $< unk >$ are also created for specific index in the vocabulary. The vocabulary will be used for the remaining part of the project.

4 Experiments

In this section, we modified some hyper-parameters to see how they would influence the performance of our model. We will focus on the effects about learning rate, number of hidden layers and teacher forcing ratio. Their default values are in table 1. The loss function is mask loss. Since it is very time consuming to train an network for NMT, we will use fixed number of epoches and see the change of loss value.

4.1 Learning Rate

We did different experiments when learning rate = 0.0001, 0.001, 0.01 and 0.1. The results are in table 2. From the results we can see that the value of learning rate shuold be neither too high nor too low. When lr is 0.0001 and 0.0001, the loss decreases very slow. When lr is 0.1, the loss is even increasing. From this results, we could say it's critical to choose a proper value of learning rate.

4.2 Number of Hidden Layers

We change the number of hidden layers to 16, 32, 64, 128 to test their performance, results are in table 3. Theoretically, the deeper your model, the stronger it will be during training. In other word, a deeper model will have more chance to converge to a lower loss comparing with a shallow model. Table 3 proofs this, we can see that as the number of epoch increases, the model with more layers has low loss. Of course, having a deep network is not a always good idea, because there will be other problem such as over-fitting or gradient vanishing.

epoch layers	1	2	3	4	5
16	8.46	6.68	6.65	6.63	6.61
32	8.90	7.12	6.73	6.67	6.66
64	7.89	6.65	6.61	6.53	6.47
128	7.10	6.54	6.45	6.34	6.24

Table 3: Loss value of different numbers of layers in first 5 epoch

4.3 Teacher Forcing Ratio

epoch ratio	1	2	3	4	5
0.3	8.06	6.70	6.66	6.65	6.63
0.4	7.22	6.59	6.52	6.48	6.44
0.5	7.89	6.65	6.61	6.53	6.47
0.6	7.24	6.59	6.51	6.46	6.41

Table 4: Loss value of different teacher forcing ratio in first 5 epoch

We tested different values of teacher forcing ratio in 0.3, 0.4, 0.5, 0.6. Results are in table 4. From the results we can see that when teacher forcing ratio is small, the training speed is slow in the beginning, which is same as expected. However, as epoch increases, the loss value becomes closes since other parameters are the same. If the teacher forcing value is high, the model will be fragile, it will be easier to over fit on the training sample, but this experiment can't show that.

4.4 Training Process

One set of default parameters is selected to train the model for 24 hours. The batch size is 64, hidden size is 512, embedding dimension is 256, encoder has 2 layers, decoder has 1 layer and dropout rate is 0.5. After training on NVIDIA GTX 1080 Ti for one day, the model converged and reasonable prediction result start to occurred. Some inference result is shown in Figure 1 and Figure 2. Then the trained model is used for the following

evaluation part.

```
INPUT:
<sos> 中非 友好 源远流长 . <eos>
REF:
<sos> friendship between china and africa enjoys a long history . <eos>
PREDICTION:
<sos> friendship between china and africa enjoys a long history . <eos>
INPUT:
<sos> 21世纪 即将 到来 . <eos>
REF:
<sos> the 21 st century is just around the corner . <eos>
PREDICTION:
<sos> the 21 st century is just around the corner . <eos>
```

Figure 1: Good Translation by Trained Model

```
INPUT:
<sos> 三 是 经济 机遇 . <eos>
REF:
<sos> three , economic opportunity . <eos>
PREDICTION:
<sos> third is the opportunity and economic . <eos>
```

Figure 2: Bad Translation by Trained Model

5 Evaluation

5.1 Beam Search

In this part, the main goal in implementing Beam Search is realizing the multiplication of conditional probability. For example, to translate to a sentence $Y_i = Y_1, Y_2, ...$ with i words, we calculate the probability of sentence segments following the pattern below:

$P(Y_1, Y_2|X) = P(Y_1|X) * P(Y_2|X, Y_1)$

The only parameter in Beam Search is the Beam Width denoted as β, deciding the most possible β sentence segments we used to predict for next word. After predicting Y_i word, we had β most possible sentence segments for each of most possible β segments at Y_{i-1} word, an overall of β^2 options. By taking topK, we retrieved most possible β sentence segments of length i. We kept this record until $\langle eos \rangle$ got reached.

In our experiment, we first implemented Bleu Score on part of training set around 50 sentences with $\beta = \{1, 2, 3, 4, 5\}$.

Beam Width	Bleu Score
Width = 1	0.7168
Width = 2	0.7487
Width = 3	0.7466
Width = 4	0.7963
Width = 5	0.7930

Table 5: Bleu score on sample set

From result above, there was significant improvement for Beam Width from $1 - 2$ and from $3 - 4$. Beam Width implementation for width = 1 generated exact the same result as our implementation of greedy search which validated our Beam Search Model.

After that, we implemented $\beta = \{1, 2, 3\}$ on test set to evaluate the performance based on Bleu score using GPU 1080-Ti.

Beam Width	Bleu Score	Time(mins)
Width = 1	0.2797	210
Width = 2	0.2642	400
Width = 3	0.2686	580

Table 6: Bleu score on entire test set

Overall, the bleu score was much smaller compared to sample set. Since we only trained 70k sentence-pairs, there existed over-fitting when we did the test part. That described a significant drop in the bleu score.

Generally, the larger the Beam Width, the greater chances for better output sequences, but would consume more resources and computation power. The smaller the B, less desired results can be derived, but much fast and memory efficient.

In our experiments there exists potential to make improvements. Though we terminated Beam Search when reaching final word $\langle eos \rangle$. However, we did not store global optimal solutions when some of predictions reached end, some were not. As we kept updating new predictions, some of the optimal solutions were overwritten by the optimal solutions of incomplete sentence segmentation.

6 Conclusion and Future Work

Bidirectional GRU is implemented for encoder and general attention decoder combined with teacher forcing is used for training a neural machine translation model to translate news report from Chinese to English. After applying optimization tricks for model training, inference and evaluation, reasonable translation occurred by trained model. ultimately, the BLEU score is 0.571 for the training set and 0.280 for the testing set, respectively.

There also exist many aspects that remained optimization and other state-of-the-art method for neural machine translation. First, data augmentation will increase the ability of model trained in this project to translate more accurate. Second, us-

ing other types of attention may make the training process more efficient. Currently, the training is slow even if the dataset just consists of 100 thousand sentences. The new type of attention such as the one mentioned in (Luong et al., 2015) could be an alternative. Third, using other type of sequence to sequence model may also help such as the model in character level, for which the word segmentation during the preprocessing is not required and the model performance will not be affected by the quality of word segmentation. Many other techniques may also be added to optimize the existed model for better translation from Chinese to English.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. *CoRR* abs/1702.01806. http://arxiv.org/abs/1702.01806.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR* abs/1508.04025. http://arxiv.org/abs/1508.04025.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '02, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR* abs/1706.03762. http://arxiv.org/abs/1706.03762.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 836–841.

R. J. Williams and D. Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation* 1(2):270–280. https://doi.org/10.1162/neco.1989.1.2.270.