One Sentence One Model for Neural Machine Translation

Xiaoqing Li, Jiajun Zhang and Chengqing Zong
National Laboratory of Pattern Recognition, Institute of Automation
Chinese Academy of Sciences
xqli,jjzhang,cqzong@nlpr.ia.ac.cn

Abstract

Current neural machine translation (NMT) practice separates the stages of training and testing. After training, the model is fixed and applied to every testing sentence. In this paper, we propose to learn a specific model for each testing sentence. This is done by learning a general model on the training data first, and then fine-tuning the model on a small set of data extracted from the training data which is similar to the testing sentence. Experimental results demonstrate this method can effectively improve the translation performance, even in the case of low similarity. And the cost of efficiency is acceptable.

1 Introduction

Neural machine translation achieved great success recently(Sutskever et al., 2014; Bahdanau et al., 2015). Thanks to the end-to-end training paradigm and the powerful modeling capacity of neural network, NMT can produce comparable or even better than traditional statistical machine translation, only after a few years of development. However, it also raises some new problems, such as how to use open vocabulary and how to avoid repeating and missing translations. These problems have been addressed by various approaches (Luong et al., 2015; Jean et al., 2015; Tu et al., 2016; Mi et al., 2016).

How to learn a good set of parameters is another challenge for nowdays deep neural networks. There has been some work in the field of NMT. (Shen et al., 2015) propose to use task specific optimization function. Specially, they propose to directly optimize BLEU score instead of likelihood o the training data. (Bengio et al., 2015) takes search into consideration during training. In common practice, the decoder use gold reference as history during training, but it has to use generated output as history during testing. To fix this discrepancy between training and testing, the authors propose to moderately replace gold reference with generated output during training. (Wiseman and Rush, 2016) takes a similar approach and regards training as beam search optimization.

In this paper, we take a new perspective and aim to learn a specific model for each testing sentence. The optimization objective of current NMT systems is minimizing the cost of the whole training data. The training data used for NMT usually consists of millions of sentence pairs, varying from topic, genre and style. Obviously they are not equally useful for a given testing sentence. So we propose a learning on-the-fly strategy for parameter fine-tuning. First, a general model is learnt from the whole training data. Then, for each testing sentence, we find some similar sentence pairs from the training data and use them to adjust the parameters. The offline and online learning part of this method is illustrated in Figure 1.

There are two key aspects for the method. One is how to define similarity and the other is how to find similar sentence pairs efficiently. For similarity measure, we tried string based similarity and hidden representation based similarity, including edit distance, average word embedding and sentence embedding. There are two additional steps compared with plain decoding: finding similar sentence pairs and fine tuning. To improve the efficiency, we used the technique of inverted index for fast retrieval. We also studied how the size of similar data influences the decoding time.

Experimental results show our approach can effectively improve the translation performance, even if data with high similarity is not available. In addition, the time cost of out approach is also acceptable.

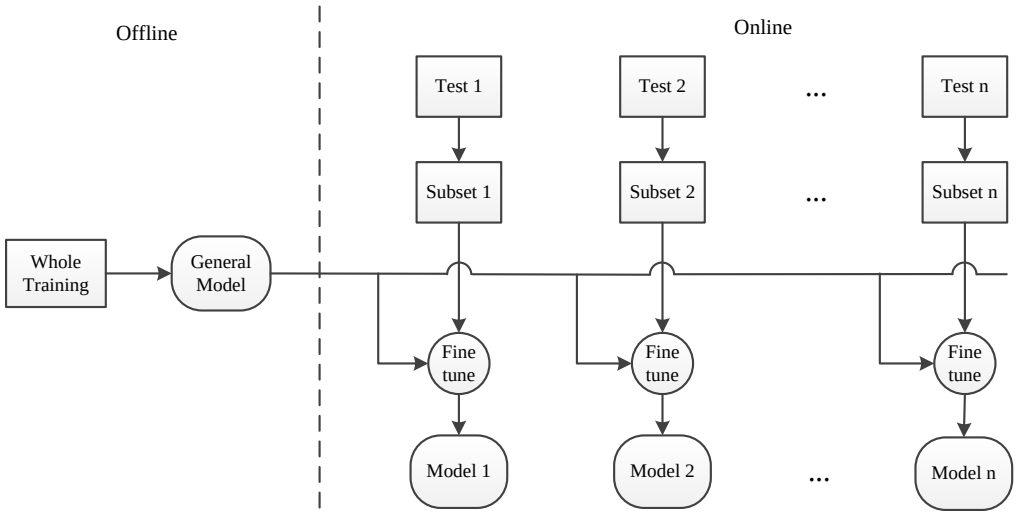

Figure 1: Learn a specific model for each testing sentence

2 Background

In this section, we will briefly introduce the NMT system from Bahdanau et al. (2015), which we will use later in the experiments. However, our approach is model independent and can be applied to other NMT systems.

Given a source sentence $s = (s_1, s_2, ...s_m)$ and its translation $t = (t_1, t_2, ..., t_n)$, NMT models the translation probability with a single neural network as follows,

$$p(t|s) = \prod_{i=1}^{n} p(t_i|t_{<i}, s) \tag{1}$$

where the conditional probability is often parameterized with the encoder-decoder framework. The encoder reads the source sentence and encodes it into a sequence of hidden states $h = h_1, h_2, ..., h_m$. We use bidirectional GRU network as the encoder in this paper.

$$h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i] \tag{2}$$

$$\overrightarrow{h}_i = \overrightarrow{\phi}(\overrightarrow{h}_{i-1}, x_i) \tag{3}$$

$$\overleftarrow{h}_i = \overleftarrow{\phi}(\overleftarrow{h}_{i+1}, x_i) \tag{4}$$

where x_i is the embedding of current word, and the recurrent activation functions $\overrightarrow{\phi}$ and $\overleftarrow{\phi}$ are gated recurrent units.

The decoder consists of a recurrent neural network and an attention mechanism. The recurrent neural network computes a hidden state for each target position as follows,

$$z_j = \phi(z_{j-1}, y_{j-1}, c_j) \tag{5}$$

where z_{j-1} is the previous hidden state, y_{j-1} is the embedding of previous word and c_j is the context vector obtained by the attention machenism, it decides which source words to look at when predicting current target word.

$$c_j = \sum_{i=1}^{m} \alpha_{i,j} h_i \tag{6}$$

and the weight $\alpha_{i,j}$ is calculated as follows,

$$\alpha_{i,j} = \frac{exp(e_{i,j})}{\sum_{k=1}^{m} exp(e_{k,j})} \tag{7}$$

$$e_{i,j} = f_{ATT}(z_{j-1}, h_i) \tag{8}$$

Then the probability of generating a specific target word w will be computed by

$$p(t_j = w|t_{<i}, s) = softmax(z_j^\top y_w) \tag{9}$$

where y_w is the embedding of target word w.

The parameters in the model are trained by maximizing the log-likelihood of the whole training corpus. And stochastic gradient descent or its variant is often adopted to for the update.

3 Tuning on-the-fly

As illustrated in Figure 1, the learning strategy of our approach is very simple. First, we learn a general model from the whole training corpus. Then, for each testing sentence, we extract a small subset from the training data, consisting of sentence pairs whose source sides are similar to the testing sentence. This subset is used fine-tune the general model and a specific model is obtained for the testing sentence. We will discuss how to evaluate sentence similarity and how to find similar sentences quickly.

3.1 Similarity Measure

There are many methods to evaluate the similarity between two sentences. In this paper, we consider three of them. The first is edit distance, which counts at least how many operations do we need to convert one sequence to another. The operations include insertion, deletion and substitution. Edit distance reflects the surface similarity of two sentences, and it does not consider the meaning of the sentence.

$$sim_{ed}(s1, s2) = \frac{ed(s1, s2)}{max(len(s1), len(s2))}$$

The second measure is based on average word embedding(Mikolov et al., 2013) of the sentence. Although this sentence representation is simple, it has been shown competitive to many complex sentence representations in many tasks.

$$sim_{vec}(s1, s2) = cos(\frac{\sum_{i=1}^{|s1|} vec(s1[i])}{len(s1)}, \frac{\sum_{j=1}^{|s2|} vec(s2[j])}{len(s2)})$$

The third measure is based on the hidden states of the encoder in NMT. Unlike word embedding, the hidden states of the encoder contains context information. What's more, the hidden states is learnt in the translation task, so we expect this representation may be better. For this similarity measure, we need to run the encoder first with the general model to get the representation of the testing sentence, then find similar sentences in the training data, finally the sentence will be encode again and decode with the new model.

$$sim_{enc}(s1, s2) = cos(\frac{\sum_{i=1}^{|s1|} h1[i]}{len(s1)}, \frac{\sum_{j=1}^{|s2|} h2[j]}{len(s2)})$$

where $h1[i]$ and $h2[j]$ are the hidden states of the two sentences, which are calculated according to equations (2) - (4).

3.2 Finding similar sentences efficiently

The training corpus for neural machine translation usually contains millions of sentences. For a given testing sentence, comparing it with every training sentence will be too time comsuming. So we propose to filter the training corpus first by only considering those which have common words with the testing sentence, and then compute similarity with the filtered set.

We use inverted index for fast retrieval. Each training sentence is given a unique index. And we maintain a word to indexes map, recording the sentence indexes where each word appears. For efficiency consideration, we ignore the most frequent words, which usually are function words and punctuations. Then for each word in a testing sentence, we find all sentence indexes which contains the word. And the union of these indexes are used as the filterd set.

However, calculating edit distance between the testing sentence and each sentence in the filtered set is still not fast enough. So we propose to further reduce the set with a simpler similarity measure, i.e. dice coefficients.

$$sim_{dice} = \frac{2|set(w \in s1) \cap set(w \in s2)|}{|set(w \in s1)| + |set(s \in s2)|}$$

We first calculate the dice coefficients between the testing sentence and each sentence in the filtered set, the reduce the size of the set to a given threshold, e.g. 1000, by keeping the sentences with the highest dice coefficients. Finally, we will calculate edit distance for the reduced set.

For the other two similarity measures, calculating cosine similarity can be done efficiently with linear algebra library. So there is no need to further reduce the filtered set.

4 Experiments

We evaluate our method on the Chinese to English translation task. Translation quality is measured by the BLEU metric (Papineni et al., 2002).

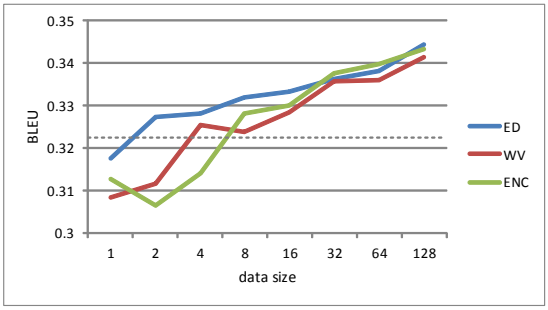

Figure 2: How data size influence translation performance.
ED: edit distance; WV: average word vectors; ENC: average of encoder hidden states.

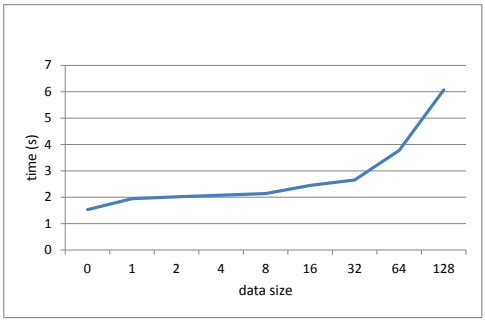

Figure 3: How data size influence efficiency

4.1 Settings

The bilingual data to train the NMT model is selected from LDC, which contains about 2M sentence pairs. To avoid spending too much training time on long sentences, all sentence pairs longer than 50 words either on the source side or on the target side are discarded. The NIST 03 dataset is chosen as the development set, which is used to monitoring the training process and decide the early stop condition. We also use it to decide how much data should be added for fine-tuning. And the NIST 04 to 06 are used as our testing set.

4.2 Training Details

The hyperparameters used in our network are described as follows. We limit both the source and target vocabulary to 30k in our experiments. The number of hidden units is 1,000 for both the encoder and decoder. And the embedding dimension is 500 for all source and target tokens. The network parameters are updated with the adadelta algorithm.

4.3 Influence of Data Size for Fine-Tuning

We first investigate how much data should be added for fine-tuning. For each testing sentence, we try to add at most 128 similar sentences found in the training data. Every time we double the data, i.e. the sizes we choose are 1, 2, 4, etc. For all the three similarity measures, the performances on the development set are shown in Figure 2.

We have the following observations according to the figure. If equal or less than 16 sentences are used for fine-tuning, the data found by the edit distance measure performs consistently better than the other two measures. We think the reason is that the sentences found by the edit distance measure have more words in common with the testing sentence, so the embeddings of more words will be updated. And if equal or more than 32 sentences are used, there is not big difference between the three measures. Generally, adding more data leads to better performance for all the three measures. If we only add one or two sentence pair for fine-tuning, the performance may be even worse than the baseline without any fine-tuning (denoted by dashed line in the figure). This is because the model overfits the data. If we can find very similar sentence to the testing sentence, overfitting is a good thing. However, the average edit distance similarity is only 0.28 for the top 1 sentence we find for each testing sentence.

The influence of data size on efficiency is shown in Figure 3. The time cost in the figure only includes fine-tuning time and decoding time. The retrieval time, i.e., time of finding similar sentences, is not shown because it is relatively small compared to the other two. Retrieval with edit distance measure is the slowest one. But it is still less than 1/3 of the decoding time. We can see from the figure if less or equal than 32 sentences are used for fine-tuning for each testing sentence, the time cost is controlled within two times of the baseline. If we use 128 sentences, the time cost increases to 4 times.

4.4 Testing Set Performance

We evaluate our method on the testing data with the following settings. The similarity measure is edit distance.And the data size used for each sentence is 128. Table 1 shows the experimental results.

Our method gives an improvement of 1.2 BLEU score on average for the testing sets. This improvement is smaller than the one we get on the development set. We think the difference is caused

System	04	05	06	Avg.
RNNSearch	36.06	32.74	34.85	34.55
online-tune	37.43	34.01	35.77	35.74

Table 1: Results for online tuning

by how similar the data we can find from the training corpus.

5 Conclusion

In this paper, we propose a novel method to tune network parameters on the fly for each testing sentence. Experiment results demonstrate our method can effectively improve the translation performance at a low efficiency cost. The source code of our system is available at https://github.com/jiajunzhangnlp/EUREKA-MangoNMT.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July. Association for Computational Linguistics.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July. Association for Computational Linguistics.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. A coverage embedding model for neural machine translation. *arXiv preprint arXiv:1605.03148*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. *arXiv preprint arXiv:1601.04811*.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*.