Decoding with Value Networks for Neural Machine Translation

Di He[1]
di_he@pku.edu.cn

Hanqing Lu[2]
hanqinglu@cmu.edu

Yingce Xia[3]
xiayingc@mail.ustc.edu.cn

Tao Qin[4]
taoqin@microsoft.com

Liwei Wang[1,5]
wanglw@cis.pku.edu.cn

Tie-Yan Liu[4]
tie-yan.liu@microsoft.com

[1]Key Laboratory of Machine Perception, MOE, School of EECS, Peking University
[2]Carnegie Mellon University   [3]University of Science and Technology of China
[4]Microsoft Research
[5]Center for Data Science, Peking University, Beijing Institute of Big Data Research

Abstract

Neural Machine Translation (NMT) has become a popular technology in recent years, and beam search is its de facto decoding method due to the shrunk search space and reduced computational complexity. However, since it only searches for local optima at each time step through one-step forward looking, it usually cannot output the best target sentence. Inspired by the success and methodology of AlphaGo, in this paper we propose using a prediction network to improve beam search, which takes the source sentence x, the currently available decoding output $y_1, \cdots, y_{t-1}$ and a candidate word w at step t as inputs and predicts the long-term value (e.g., BLEU score) of the partial target sentence if it is completed by the NMT model. Following the practice in reinforcement learning, we call this prediction network *value network*. Specifically, we propose a recurrent structure for the value network, and train its parameters from bilingual data. During the test time, when choosing a word w for decoding, we consider both its conditional probability given by the NMT model and its long-term value predicted by the value network. Experiments show that such an approach can significantly improve the translation accuracy on several translation tasks.

1 Introduction

Neural Machine Translation (NMT), which is based on deep neural networks and provides an end-to-end solution to machine translation, has attracted much attention from the research community [2, 6, 12, 20] and gradually been adopted by industry in past several years [18, 22]. NMT uses an RNN-based encoder-decoder framework to model the entire translation process. In training, it maximizes the likelihood of a target sentence given a source sentence. In testing, given a source sentence x, it tries to find a sentence y^* in the target language that maximizes the conditional probability $P(y|x)$. Since the number of possible target sentences is exponentially large, finding the optimal y^* is NP-hard. Thus beam search is commonly employed to find a reasonably good y.

Beam search is a heuristic search algorithm that maintains the top-scoring partial sequences expanded in a left-to-right fashion. In particular, it keeps a pool of candidates each of which is a partial sequence. At each time step, the algorithm expands each candidate by appending a new word, and then keeps

the top-ranked new candidates scored by the NMT model. The algorithm terminates if it meets the maximum decoding depth or all sentences are completely generated, i.e., all sentences are ended with the end-of-sentence (EOS) symbol.

While NMT with beam search has been proved to be successful, it has several obvious issues, including exposure bias [9], loss-evaluation mismatch [9] and label bias [16], which have been studied. However, we observe that there is still an important issue associated with beam search of NMT, the myopic bias, which unfortunately is largely ignored, to the best of our knowledge. Beam search tends to focus more on short-term reward. At iteration t, for a candidate $y_1, \cdots, y_{t-1}$ (refers to $y_{<t}$) and two words w and w', we denote $y_{<t} + w$ if we append w to $y_{<t}$. If $P(y_{<t} + w|x) > P(y_{<t} + w'|x)$, new candidate $y_{<t} + w$ is more likely to be kept, even if w' is the ground truth translation at step t or can offer a better score in future decodings. Such search errors coming from short sighted actions sometimes provide a bad translation even if the translation model is good.

To address the myopic bias, for each word w and each candidate $y_{<t}$, we propose to design a prediction model to estimate the long-term reward if we append w to $y_{<t}$ and follow the current NMT model until the decoding finishes. Then we can leverage the predicted score from this model during each decoding step to help find a better w that can contribute to the long-term translation performance. This prediction model, which predicts long-term reward we will receive in the future, is exactly the concept of value function in Reinforcement Learning (RL).

In this work, we develop a neural network-based prediction model, which is called *value network for NMT*. The value network takes the source sentence and any partial target sequence as input, and outputs a predicted value to estimate the expected total reward (e.g. BLEU) generated from this partial sequence by the NMT model. In any decoding step, we select the best candidates not only based on the conditional probability of the partial sequence outputted by the NMT model, but also based on the estimated long-term reward outputted by the value network.

The main contributions of this work are summarized as follows. First, we develop a decoding scheme that considers long-term reward while generating words one by one for machine translation, which is new in NMT literature. At each step, the new decoding scheme not only considers the probability of the word sequence conditioned on the source sentence, but also relies on the predicted future reward. We believe that considering the two aspects can lead to better final translation.

Second, we design a novel structure for the value network. On the top of the encoder-decoder layer of NMT, we develop another two modules for the value network, a *semantic matching module* and a *context-coverage module*. The semantic matching module aims at estimating the similarity between the source and target sentences, which can contribute to the quality of the translation. It is often observed that the more context used in the attention mechanism, the better translation we will generate [14, 15]. Thus we build a context-coverage module to measure the coverage of context used in the encoder-decoder layer. With the outputs of the two modules, the value prediction is done via fully connected layers.

We conduct a set of experiments on several translation tasks. All the results demonstrate the effectiveness and robustness of the new decoding mechanism compared to several baseline algorithms.

The remaining parts of the paper are organized as follows. In Section 2, we briefly review the literature of neural machine translation. After that, we describe the myopic bias problem of NMT in Section 3 and introduce our method for value network learning in Section 4. Experimental results are provided and analyzed in Section 5. We discuss future directions in the last section.

2 Neural Machine Translation

Neural machine translation systems are typically implemented with a Recurrent Neural Network (RNN)-based encoder-decoder framework. Such a framework directly models the probability $P(y|x)$ of a target sentence $y = \{y_1, y_2, ..., y_{T_y}\}$ conditioned on the source sentence $x = \{x_1, x_2, ..., x_{T_x}\}$, where T_x and T_y are the length of sentence x and y.

The encoder of NMT reads the source sentence x word by word and generates a hidden representation for each word x_i:
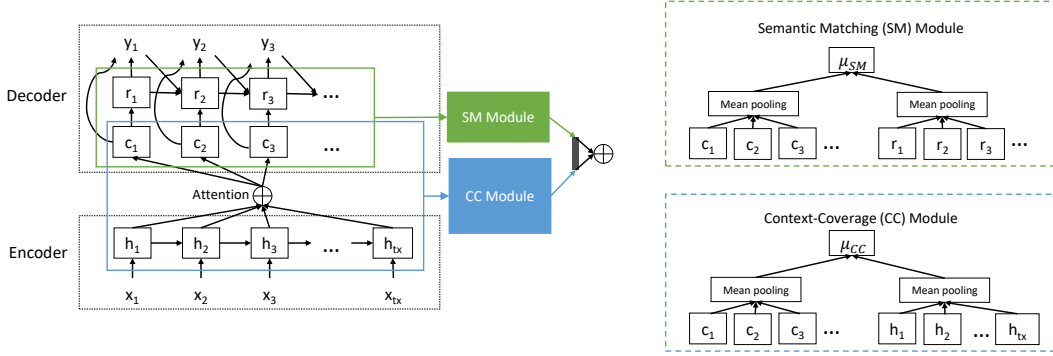
$$h_i = f(h_{i-1}, x_i), \tag{1}$$

Figure 1: Architecture of Value Network

in which function f is the recurrent unit such as Long Short-Term Memory (LSTM) unit [12] or Gated Recurrent Unit (GRU) [4]. Afterwards, the decoder of NMT computes the conditional probability of each target word y_t conditioned on its proceeding words $y_{<t}$ as well as the source sentence:

$$P(y_t|y_{<t}, x) \propto \exp(y_t; r_t, c_t), \tag{2}$$

$$r_t = g(r_{t-1}, y_{t-1}, c_t), \tag{3}$$

$$c_t = q(r_{t-1}, h_1, \cdots, h_{T_x}), \tag{4}$$

where r_t is the decoder RNN hidden representation at step t, similarly computed by an LSTM or GRU, and c_t denotes the weighted contextual information summarizing the source sentence x using some attention mechanism [4]. Denote all the parameters to be learned in the encoder-decoder framework as Θ. For ease of reference, we also use π_Θ to represent the translation model with parameter Θ.

Denote D as the training dataset that contains source-target sentence pairs. The training process aims at seeking the optimal parameters Θ^* to correctly encode source sentence and decode it into the target sentence. While there are different objectives to achieve this [2, 10, 1, 5, 19, 17], maximum likelihood estimation is the most popular one [2]:

$$\begin{aligned}
\Theta^* &= \underset{\Theta}{\operatorname{argmax}} \prod_{(x,y) \in D} P(y|x; \Theta) \\
&= \underset{\Theta}{\operatorname{argmax}} \prod_{(x,y) \in D} \prod_{t=1}^{T_y} P(y_t|y_{<t}, x; \Theta).
\end{aligned} \tag{5}$$

3 The Myopic Bias

Since during training, one aims to find the conditional probability $P(y|x)$, ideally in testing, the translation of a source sentence x should be the target sentence y with the maximum conditional probability $P(y|x)$. However, as there are exponentially many candidates in the target language, one cannot compute the probability for every candidate and find the maximum one. Thus, beam search is widely used to find a reasonable good target sentence [4, 14, 12].

Note that the training objective of NMT is usually defined on the full target sentence y instead of partial sentences. One issue with beam search is that a locally good word might not lead to a good complete sentence. From the example mentioned in the introduction, we can see that such search errors from short sighted actions can provide a bad translation even if we hold a perfect translation model. We call such errors the *myopic bias*. To reduce myopic bias, we hope to predict the long-term value of each action and use the value in decoding, which is the exact motivation of our work.

There exist several works weakly related to this issue. [3] develops the scheduled sampling approach, which takes the generated outputs from the model as well as the golden truth sentence in training, to help the model learn from its own errors. Although it can (to some extent) handle the negative

3

impact of choosing an incorrect word at middle steps, it still follows beam search during testing, which cannot avoid the myopic bias. Another related work is [16]. It learns a predictor to predict the ranking score of a certain word at step t, and use this score to replace the conditional probability outputted by the NMT model for beam search during testing. Unfortunately, this work still looks only one step forward and cannot address the problem.

4 Value Network for NMT

As discussed in the previous section, it is not reasonable to fully rely on the conditional probability in beam search. This motivates us to estimate the expected performance of any sequence during decoding, which is exactly the concept of value function in reinforcement learning.

4.1 Value Network Structure

In conventional reinforcement learning, a value function describes how much cumulated reward could be collected from state s by following certain policy π. In machine translation, we can consider any input sentence x paired with partial output sentence $y_{<t}$ as the state, and consider the translation model π_Θ as policy which can generate a word (action) given any state. Given policy π_Θ, the value function characterizes what the expected translation performance (e.g. BLEU score) is if we use π_Θ to translate x with the first $t-1$ words being $y_{<t}$. Denote $v(x, y_{<t})$ as the value function and $y^*(x)$ as the ground truth translation, and then $v(x, y_{<t}) = \sum_{y' \in \mathcal{Y}: y'_{<t} = y_{<t}} \text{BLEU}(y^*(x), y') P(y'|x; \Theta)$, where $\mathcal{Y}$ is the space of complete sentences.

The first important problem is how to design the input and the parametric form of the value function. As the translation model is built up on an encoder-decoder framework, we also build up our value network on the top of this architecture. To fully exploit the information in the encoder-decoder framework, we develop a value network with two new modules, the semantic matching module and the context-coverage module.

Semantic Matching (SM) Module: In the semantic matching module, at time step t, we use mean pooling over the decoder RNN hidden states $\bar{r}_t = \frac{1}{t} \sum_{l=1}^{t} r_l$ as a summarization of the partial target sentence, and use mean pooling over context states $\bar{c}_t = \frac{1}{t} \sum_{l=1}^{t} c_l$ as a summarization of the context in source language. We concatenate $\bar{r}_t$ and $\bar{c}_t$, and use a feed-forward network $\mu_{SM} = f_{SM}([\bar{r}_t, \bar{c}_t])$ to evaluate semantic information between the source sentence and the target sentence.

Context-Coverage (CC) Module: It is often observed that the more context covered in the attention model, the better translation we will generate [14, 15]. Thus we build a context-coverage module to measure the coverage of information used in the encoder-decoder framework. We argue that using mean pooling over the context layer and the encoding states should give some effective knowledge. Similarly, denote $\bar{h} = \frac{1}{T_x} \sum_{l=1}^{T_x} h_l$, we use another feed-forward network $\mu_{CC} = f_{CC}([\bar{c}_t, \bar{h}])$ to process such information.

In the end, we concatenate both μ_{SM} and μ_{CC} and then use another fully connected layer with sigmoid activation function to output a scalar as the value prediction. The whole architecture is shown in Figure 1.

4.2 Training Data Generation

Based on the designed value network structure, we aim at finding a model that can correctly predict the performance after the decoding ends. Popular value function learning algorithms include Monte-Carlo methods and Temple-Difference methods, and both of them have been adopted in many challenging tasks [13, 7, 11]. In this paper, we adopt the Monte-Carlo method to learn the value function. Given a well-learnt NMT model π_Θ, the training of the value network for π_Θ is shown in Algorithm 1. For randomly picked source sentence x in the training corpus, we generate a partial target sentence y_p using π_Θ with random early stop, i.e., we randomly terminate the decoding process before its end. Then for the pair (x, y_p), we use π_Θ to finish the translation starting from y_p and obtain a set $S(y_p)$ of K complete target sentences, e.g., using beam search. In the end, we compute the BLEU score of

each complete target sentence and calculate the averaged BLEU score of (x, y_p):

$$\text{avg\_bleu}(x, y_p) = \frac{1}{K} \sum_{y \in S(y_p)} \text{BLEU}(y^*(x), y). \tag{6}$$

avg_bleu(x, y_p) can be considered as an estimation of the long-term reward of state (x, y_p) used in value network training.

4.3 Learning

Algorithm 1 Value network training

1: **Input**: Bilingual corpus, a trained neural machine translation model π_Θ, hyperparameter K.
2: **repeat**
3:      $t = t + 1$.
4:      Randomly pick a source sentence x from the training dataset.
5:      Generate two partial translations $y_{p,1}, y_{p,2}$ for x using π_Θ with random early stop.
6:      Generate K complete translations for each partial translation using π_Θ and beam search.
     Denote this set of complete target sentences as $S(y_{p,1})$ and $S(y_{p,2})$.
7:      Compute the BLEU score for each sentence in $S(y_{p,1})$ and $S(y_{p,2})$.
8:      Calculate the average BLEU score for each partial translation according to Eqn.(6)
9:      Gradient Decent on stochastic loss defined in Eqn.(7).
10: **until** converge
11: **Output**: Value network with parameter ω

In conventional Monte-Carlo method for value function estimation, people usually use a regression model to approximate the value function, i.e., learn a mapping from $(x, y_p) \rightarrow$ avg_bleu(x, y_p) by minimizing the mean square error (MSE). In this paper, we take an alternative objective function which is shown to be more effective in experiments. We hope the value network we learn is accurate and useful in differentiating good and bad examples. Thus we use pairwise ranking loss instead of MSE loss.

To be concrete, we sample two partial sentences $y_{p,1}$ and $y_{p,2}$ for each x. We hope the predicted score of $(x, y_{p,1})$ can be larger than that of $(x, y_{p,2})$ by certain margin if avg_bleu$(x, y_{p,1}) >$ avg_bleu$(x, y_{p,2})$. Denote ω as the parameter of the value function described in Section 4.1. We design the loss function as follows:

$$L(\omega) = \sum_{(x, y_{p,1}, y_{p,2})} e^{v_\omega(x, y_{p,2}) - v_\omega(x, y_{p,1})}, \tag{7}$$

where avg_bleu$(x, y_{p,1}) >$ avg_bleu$(x, y_{p,2})$.

Algorithm 2 Beam search with value network in NMT

1: **Input**: Testing example x, neural machine translation model $P(y|x)$ with target vocabulary V, value network model $v(x, y)$, beam search size K, maximum search depth L, weight α.
2: Set $S = \emptyset, U = \emptyset$ as candidate sets.
3: **repeat**
4:      $t = t + 1$.
5:      $U_{expand} \leftarrow \{y_i + \{w\} | y_i \in U, w \in V\}$.
6:      $U \leftarrow \{$top $(K - |S|)$ candidates that maximize
       $\alpha \times \frac{1}{t} \log P(y|x) + (1 - \alpha) \times \log v(x, y) | y \in U_{expand}\}$
7:      $U_{complete} \leftarrow \{y | y \in U, y_t = \text{EOS}\}$
8:      $U \leftarrow U \setminus U_{complete}$
9:      $S \leftarrow S \cup U_{complete}$
10: **until** $|S| = K$ or $t = L$
11: **Output**: $y = \text{argmax}_{y \in S \cup U} \alpha \times \frac{1}{|y|} \log P(y|x) + (1 - \alpha) \times \log v(x, y)$

4.4 Inference

Since the value network estimates the long-term reward of a state, it will be helpful to enhance the decoding process of NMT. For example, in a certain decoding step, the NMT model prefers word w_1 over w_2 according to the conditional probability, but it does not know that picking w_2 will be a better choice for future decoding. As the value network provides sufficient information on the future reward, if the value network outputs show that picking w_2 is better than picking w_1, we can take both NMT probability and future reward into consideration to choose a better action.

In this paper, we simply linearly combine the outputs of the NMT model and the value network, which is motivated by the success of AlphaGo [11]. We first compute the normalized log probability of each candidate, and then linearly combine it with the logarithmic value of the reward. In detail, given a translation model $P(y|x)$, a value network $v(x, y)$ and a hyperparameter $\alpha \in (0, 1)$, the score of partial sequence y for x is computed by

$$\alpha \times \frac{1}{|y|} \log P(y|x) + (1 - \alpha) \times \log v(x, y), \tag{8}$$

where $|y|$ is the length of y. The details of the decoding process are presented in Algorithm 2, and we call our neural network-based decoding algorithm NMT-VNN for short.

5 Experiments

5.1 Settings

We compare our proposed NMT-VNN with two baselines. The first one is classic NMT with beam search [2] (NMT-BS). The second one [16] trains a predictor that can evaluate the quality of any partial sequence, e.g., partial BLEU score [1], and then it uses the predictor to select words instead of the probability. The main difference between [16] and ours is that they predict the local improvement of BLEU for any single word, while ours aims at predicting the final BLEU score and use the predicted score to select words. We refer their work as beam search optimization (we call it NMT-BSO). For NMT-BS, we directly used the open source code [2]. NMT-BSO was implemented by ourselves based on the open source code [2].

We tested our proposed algorithms and the baselines on three pairs of languages: English→French (En→Fr), English→German (En→De), and Chinese→English (Zh→En). In detail, we used the same bilingual corpora from WMT' 14 as used in [2] , which contains 12M, 4.5M and 10M training data for each task. Following common practices, for En→Fr and En→De, we concatenated newstest2012 and newstest2013 as the validation set, and used newstest2014 as the testing set. For Zh→En, we used NIST 2006 and NIST 2008 datasets for testing, and use NIST 2004 dataset for validation. For all datasets in Chinese, we used a public tool for word segmentation. In all experiments, validation sets were only used for early-stopping and hyperparameter tuning.

For NMT-VNN and NMT-BS, we need to train an NMT model first. We followed [2] to set experimental parameters to train the NMT model. For each language, we constructed the vocabulary with the most common 30K words in the parallel corpora, and out-of-vocabulary words were replaced with a special token "UNK". Each word was embedded into a vector space of 620 dimensions, and the dimension of the recurrent unit was 1000. We removed sentences with more than 50 words from the training set. Batch size was set as 80 with 20 batches pre-fetched and sorted by sentence lengths. The NMT model was trained with asynchronized SGD on four K40m GPUs for about seven days. For NMT-BSO, we implemented the algorithm and the model was trained in the same environment.

For the value network used in NMT-VNN, we set the same parameters for the encoder-decoder layers as the NMT model. Additionally, in the SM module and CC module, we set function μ_{SM} and μ_{CC} as single-layer feed forward networks with 1000 output nodes. In Algorithm 1, we set the hyperparameter $K = 20$ to estimate the value of any partial sequence. We adapted mini-batch training with batch size to be 80, and the value network model was trained with AdaDelta [21] on one K40m GPU for about three days.

---

[1]If the ground truth is y^*, the partial bleu on the partial sequence $y_{<t}$ at step t is defined as the BLEU score on $y_{<t}$ and $y^*_{<t}$.

During testing, the hyperparameter α for NMT-VNN was set by cross validation. For En→Fr, En→De and Zh→En tasks, we found setting α to be 0.85, 0.9 and 0.8 respectively are the best choices. We used the BLEU score [8] as the evaluation metric, which is computed by the *multi-bleu.perl* script[2]. We set the beam search size to be 12 for all the algorithms following the common practice [12].
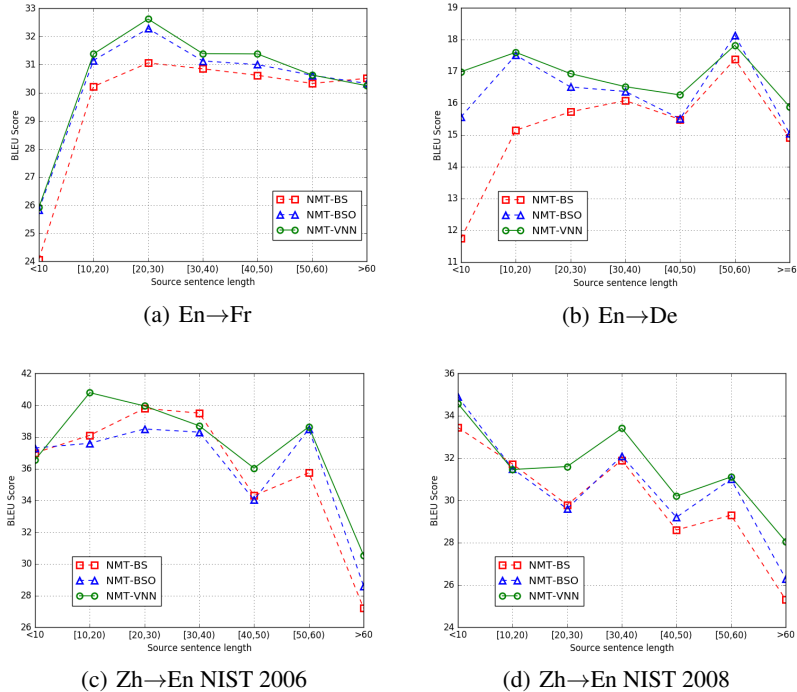

(a) En→Fr

(b) En→De

(c) Zh→En NIST 2006

(d) Zh→En NIST 2008

Figure 2: Translation results on the test sets of three tasks

5.2 Overall Results

We report the experimental results in this subsection. From Table 1 we can see that our NMT-VNN algorithm outperforms the baseline algorithms on all tasks.

For English→French task and English→German task, NMT-VNN outperforms the baseline NMT-BS by about $1.03/1.3$ points. As the only difference between the two algorithms is that our NMT-VNN additionally uses the outputs of value network to enhance decoding, we can conclude that such additional knowledge provides useful information to help the NMT model. Our method outperforms NMT-BSO by about $0.31/0.33$ points. Since NMT-BSO only uses a local BLEU predictor to estimate the partial BLEU score while ours predicts the future performance, our proposed value network which considers long-term benefit is more powerful.

The performance of our NMT-VNN is much better than NMT-BS and NMT-BSO for Chinese→English tasks. NMT-VNN outperforms the baseline NMT-BS by about $1.4/1.82$ points on NIST 2006 and NIST 2008, and outperforms NMT-BSO by about $1.01/0.72$ points. We plot BLEU scores with respect to the length of source sentences in Figure 2 for all the tasks. From the figures, we can see that our NMT-VNN algorithm outperforms the baseline algorithms in almost all the ranges of length.

Furthermore, we also test our value network on a deep NMT model in which the encoder and decoder are both stacked 4-layer LSTMs. The result also shows that we can get 0.33 points improvement on English→French task. These results demonstrate the effectiveness and robustness of our NMT-VNN algorithm.

---

[2]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl. For final evaluation we use corpus-level BLEU, while for the value network training we use sentence-level BLEU as in [1].

Table 1: Overall Performance

	En→Fr	En→De	Zh→En NIST06	Zh→En NIST08	En→Fr Deep
NMT-BS	30.51	15.67	36.2	29.4	37.86
NMT-BSO	31.23	16.64	36.59	30.5	–
NMT-VNN	**31.54**	**16.97**	**37.6**	**31.22**	**38.19**

5.3 Analysis on Value Network

We further look into the learnt value network and conduct some analysis to better understand it.

First, as we use an additional component during decoding, it will affect the efficiency of the translation process. As the designed value network architecture is similar to the basic NMT model, the computational complexity is similar to the NMT model and the two processes can be run in parallel.

Second, it has been observed that the accuracy of NMT is sometimes very sensitive to the size of beam search on certain tasks. As the beam size grows, the accuracy will drop drastically. [14] argues this is because the training of NMT favors short but inadequate translation candidates. We also observe this phenomenon on English→German translation. However, we show that by using value network, such shortage can be largely avoided. We tested the accuracy of our algorithm with different beam sizes, as shown in Figure 3.(a). It can be seen that NMT-VNN is much more stable than the original NMT without value network: its accuracy only differs a little for different beam sizes while NMT-BS drops more than 0.5 point when the beam size is large.
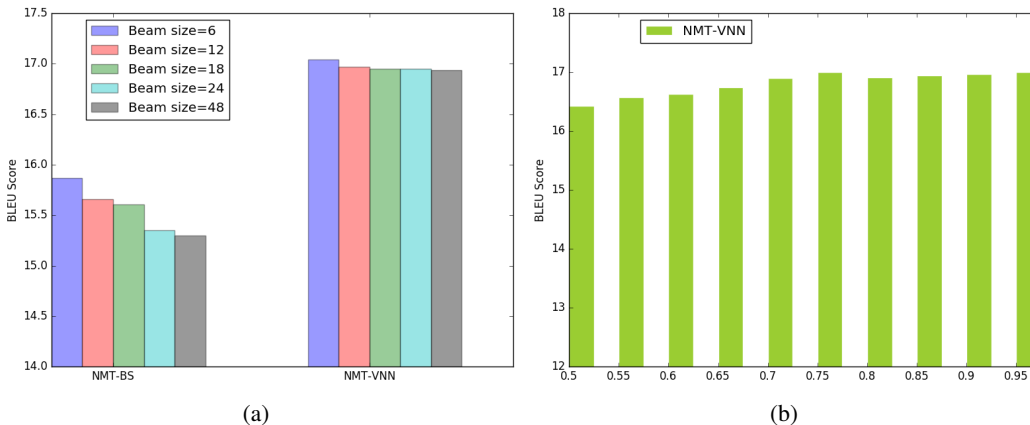

Figure 3: (a). BLEU scores of En→De task w.r.t different beam size. (b). BLEU scores of En→De task w.r.t different hyperparameter α.

Third, we tested the performances of NMT-VNN using different hyperparameter α during decoding for English→German task. As can be seen from the figure, the performance is stable for the α ranging from 0.7 to 0.95, and slightly drops for a smaller α. This shows that our proposed algorithm is robust to the hyperparameter.

6 Conclusions and Future Work

In this work we developed a new decoding scheme that incorporates value networks for neural machine translation. By introducing the value network, the new decoding scheme considers not only the local conditional probability of a candidate word, but also its long-term reward for future decoding. Experiments on three translation tasks verify the effectiveness of the new scheme. We plan to explore the following directions in the future. First, it is interesting to investigate how to design better structures for the value network. Second, the idea of using value networks is quite general, and we will extend it to other sequence-to-sequence learning tasks, such as image captioning and dialog systems.

8

Acknowledgments

References

[1] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An actor-critic algorithm for sequence prediction. *ICLR*, 2017.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[3] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[4] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[5] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T. Liu, and W.-Y. Ma. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828, 2016.

[6] S. Jean, K. Cho, R. Memisevic, and Y. Bengio. 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 2015. Association for Computational Linguistics.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[8] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[9] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *ICLR*, 2016.

[10] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. Minimum risk training for neural machine translation. In *ACL*, 2016.

[11] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[12] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[13] G. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219, 1994.

[14] Z. Tu, Y. Liu, L. Shang, X. Liu, and H. Li. Neural machine translation with reconstruction. In *AAAI*, pages 3097–3103, 2017.

[15] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li. Coverage-based neural machine translation. *CoRR, abs/1601.04811*, 2016.

[16] S. Wiseman and A. M. Rush. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*, 2016.

[17] L. Wu, Y. Xia, L. Zhao, F. Tian, T. Qin, J. Lai, and T.-Y. Liu. Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*, 2017.

[18] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[19] Y. Xia, T. Qin, W. Chen, J. Bian, N. Yu, and T.-Y. Liu. Dual supervised learning. In *ICML*, 2017.

[20] Y. Xia, F. Tian, L. Wu, J. Lin, T. Qin, N. Yu, and T. Liu. Deliberation networks: Sequence generation beyond one-pass decoding. In *31st Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

[21] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[22] J. Zhou, Y. Cao, X. Wang, P. Li, and W. Xu. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199*, 2016.