Accelerated Reinforcement Learning for Sentence Generation by Vocabulary Prediction

Kazuma Hashimoto*
Salesforce Research
k.hashimoto@salesforce.com

Yoshimasa Tsuruoka
The University of Tokyo
tsuruoka@logos.t.u-tokyo.ac.jp

Abstract

A major obstacle in reinforcement learning-based sentence generation is the large action space whose size is equal to the vocabulary size of the target-side language. To improve the efficiency of reinforcement learning, we present a novel approach for reducing the action space based on dynamic vocabulary prediction. Our method first predicts a fixed-size small vocabulary for each input to generate its target sentence. The input-specific vocabularies are then used at supervised and reinforcement learning steps, and also at test time. In our experiments on six machine translation and two image captioning datasets, our method achieves faster reinforcement learning (∼2.7x faster) with much less GPU memory (∼10x less) than the full-vocabulary counterpart. The reinforcement learning with our method consistently leads to significant improvement of BLEU scores, and the scores are equal to or better than those of baselines using the full vocabularies, with faster decoding time (∼3x faster) on CPUs.

1 Introduction

Sentence generation with neural networks plays a key role in many language processing tasks, including machine translation (Sutskever et al., 2014), image captioning (Lin et al., 2014), and abstractive summarization (Rush et al., 2015). The most common approach for learning the sentence generation models is maximizing the likelihood of the model on the gold-standard target sentences. Recently, approaches based on reinforcement learning have attracted much attention to bridge the gap between training and test time and to directly incorporate task-specific evaluation metrics such as BLEU scores (Papineni et al., 2002) into the optimization (Ranzato et al., 2016).

While reinforcement learning-based sentence generation is appealing, it is often too computationally demanding to be used with large training data. In reinforcement learning for sentence generation, selecting an action corresponds to selecting a target word in generating a whole sentence. The number of possible actions at each time step is thus equal to the vocabulary size, which often exceeds tens of thousands. Such an action space is much larger than those of typical settings of reinforcement learning such as the game of Go (the size of action space $|A| = 362$ (Silver et al., 2017)) and Atari games ($|A| = 4 \sim 18$ (Mnih et al., 2013)). Among such a large set of possible actions, at most N actions are selected if the length of the generated sentence is N, where we can assume $N \ll |A|$. In other words, most of the possible actions are not selected, and the large action space slows down reinforcement learning and consumes a large amount of GPU memory.

In this paper, we propose to accelerate reinforcement learning by reducing the large action space. The reduction of action space is achieved by predicting a small vocabulary for each source input. Our method first constructs the small input-specific vocabulary by selecting K ($= |A| \leq 1000$) relevant words, and then the small vocabulary is used at both training and test time.

Our experiments on six machine translation and two image captioning datasets show that our method enables faster reinforcement learning with much less GPU memory than the standard full softmax method, without degrading the accuracy of the sentence generation tasks. Our method also works faster at test time, especially on CPUs. The PyTorch implementation of our method is publicly available.[1]

* Work was done while the first author was working at the University of Tokyo.

[1] https://github.com/hassyGo/NLG-RL

2 Reinforcement Learning with Input-Specific Vocabularies

We first describe a neural machine translation model and an image captioning model as examples of sentence generation models used in this paper. Machine translation is a text-to-text task, and image captioning is an image-to-text task. We then review how reinforcement learning is used, and present a simple and efficient method to reduce the computational cost.

2.1 Sentence Generation

Recurrent Neural Networks (RNNs) are widely used to generate sentences by outputting words one by one (Sutskever et al., 2014; Luong et al., 2015). To generate a sentence $Y = (y_1, y_2, \ldots, y_N)$, where N is its length, given a source input X, a hidden state $h_t \in \mathbb{R}^d$ is computed for each time step t (≥ 1) by using its previous information:

$$h_t = \mathrm{RNN}\left(h_{t-1}, e(y_{t-1}), s_{t-1}\right), \quad (1)$$

where $\mathrm{RNN}(\cdot)$ is an RNN function, $e(y_{t-1}) \in \mathbb{R}^d$ is a word embedding of y_{t-1}, and $s_{t-1} \in \mathbb{R}^d$ is a hidden state optionally used to explicitly incorporate the information about the source input X into the transition. We employ Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997) for the RNN function, and cell memories are internally computed. The task here is to predict the t-th word y_t by computing a target word distribution $p(y|y_{<t}, X) \in \mathbb{R}^{|V|}$, where $|V|$ represents the vocabulary size of the target language. $p(y|y_{<t}, X)$ is used to generate sentences by either greedy/beam search or random sampling.

To learn the model parameters, the following cross entropy loss is usually employed:

$$L_c(Y_g, X) = -\sum_{t=1}^{N_g} \log p\left(y = y_t | y_{<t}, X\right), \quad (2)$$

where we assume that the target sentence Y_g is the gold sequence. Once we train the model, we can use it to generate unseen sentences.

Machine translation In the context of machine translation, the source input X corresponds to a source sentence $(x_1, x_2, \ldots, x_M)$ of length M. Each word x_i is also associated with a word embedding $\tilde{e}(x_i) \in \mathbb{R}^d$. We assume that a hidden state $\tilde{h}_i \in \mathbb{R}^{2d}$ is computed for each x_i by using a bi-directional RNN with LSTM units (Graves and Schmidhuber, 2005). That is, $\tilde{h}_i$ is the concatenation of x_i's d-dimensional hidden states $[\overrightarrow{h}_i; \overleftarrow{h}_i]$ computed by a pair of forward and backward RNNs. We set the initial hidden state of the sentence generator as $h_0 = \overrightarrow{h}_M + \overleftarrow{h}_1$. Following an attention mechanism proposed in Luong et al. (2015), s_t for predicting y_t is computed as follows:

$$a_i = \frac{\exp\left(h_t \cdot \left(W_a \tilde{h}_i\right)\right)}{\sum_{j=1}^{M} \exp\left(h_t \cdot \left(W_a \tilde{h}_j\right)\right)}, \quad (3)$$

$$c_t = \sum_{i=1}^{M} a_i \tilde{h}_i, \quad (4)$$

$$s_t = \tanh\left(W_s[h_t; c_t] + b_s\right), \quad (5)$$

where $W_s \in \mathbb{R}^{d \times 3d}$ and $W_a \in \mathbb{R}^{d \times 2d}$ are weight matrices, and $b_s \in \mathbb{R}^d$ is a bias vector. s_t is then used to compute the target word distribution:

$$p(y|y_{<t}, X) = \mathrm{softmax}(W_p s_t + b_p), \quad (6)$$

where $W_p \in \mathbb{R}^{|V| \times d}$ is a weight matrix, and $b_p \in \mathbb{R}^{|V|}$ is a bias vector.

Image captioning In the case of image captioning, the source input X corresponds to an image to be described. We assume that in our preprocessing step, each input image is fed into a convolutional neural network to extract its fixed-length feature vector $f \in \mathbb{R}^{d_f}$. More specifically, we use the pre-computed feature vectors provided by Kiros et al. (2014), and the feature vectors are never updated in any model training processes. The input feature vector is transformed into the initial hidden state $h_0 = \tanh\left(W_f f + b_f\right)$, where $W_f \in \mathbb{R}^{d \times d_f}$ is a weight matrix, and $b_f \in \mathbb{R}^d$ is a bias vector. In contrast to machine translation, we do not use s_{t-1} in Equation (1); more concretely, we do not use any attention mechanisms for image captioning. Therefore, we directly use the hidden state h_t to compute the target word distribution:

$$p(y|y_{<t}, X) = \mathrm{softmax}(W_p h_t + b_p), \quad (7)$$

where the weight and bias parameters are analogous to the ones in Equation (6).

For both of the tasks, we use the weight-tying technique (Inan et al., 2017; Press and Wolf, 2017) by using W_p as the word embedding matrix. That is, $e(y_t)$ is the y_t-th row vector in W_p, and the technique has shown to be effective in machine translation (Hashimoto and Tsuruoka, 2017) and text summarization (Paulus et al., 2018).

2.2 Applying Reinforcement Learning

One well-known limitation of using the cross entropy loss in Equation (2) is that the sentence generation models work differently at the training and test time. More concretely, the models only observe gold sequences at the training time, whereas the models have to handle unseen sequences to generate sentences at the test time.

To bridge the gap, reinforcement learning has started gaining much attention (Ranzato et al., 2016; Wu et al., 2016; Rennie et al., 2017; Zhang and Lapata, 2017; Paulus et al., 2018; Yang et al., 2018). We employ one of the simplest methods, REINFORCE (Williams, 1992).[2] In REINFORCE, the sentence generation model sets an initial state given a source input, and then iterates an action selection and its corresponding state transition. The action selection corresponds to randomly sampling a target word from Equation (6) and (7), and the state transition corresponds to the RNN transition in Equation (1).

Once a sentence is generated, an approximated loss function is defined as follows:

$$L_r(Y, X) = -\sum_{t=1}^{N} R_t \log p(y = y_t | y_{<t}, X), \quad (8)$$

where R_t is the reward at the time step t, and the loss is approximated by the single example Y. R_t is used to evaluate how good the t-th action selection is. The reward function is defined by using task-specific evaluation scores like BLEU for machine translation. In this paper, we employ GLEU proposed by Wu et al. (2016), a variant of sentence-level BLEU. Following the implementation in Ranzato et al. (2016), we define R_t as follows:

$$R_t = \mathrm{GLEU}(Y, Y_g) - b_t, \quad (9)$$

where b_t is a baseline value estimating the future reward from the next time step to reduce the variance of the gradients. To estimate b_t, we jointly train a linear regression model by minimizing $\|b_t - \mathrm{GLEU}(Y, Y_g)\|^2$, and b_t is computed as follows:

$$b_t = \sigma(W_r \cdot s_t + b_r), \quad (10)$$

where $W_r \in \mathbb{R}^d$ is a weight vector, b_r is a bias, $\sigma(\cdot)$ is the logistic sigmoid function[3], and in the case of image captioning, h_t is used instead of s_t.

[2] We tried the self-critical method in Rennie et al. (2017), but did not observe significant improvement over the REINFORCE baseline.

[3] A GLEU score is in the range of $[0.0, 1.0]$.

Overall model training The reinforcement learning step is usually applied after pre-training the models with the cross entropy loss in Equation (2). At the REINFORCE phase, we define the following joint loss function:

$$L = \lambda L_c + (1 - \lambda) L_r, \quad (11)$$

where $\lambda \in [0.0, 1.0]$ is a hyperparameter, and $\lambda = 0.0$ usually leads to unstable training (Wu et al., 2016).

2.3 Large Action-Space Reduction

The vocabulary size $|V|$ is usually more than ten thousands for datasets covering many sentences with a variety of topics. However, for example, at most 100 unique words are selected when generating a sentence of length 100. That is, the output length N is much smaller than the vocabulary size $|V|$, and this fact motivated us to reduce the large action space. Moreover, we have in practice found that REINFORCE runs several times slower than the supervised learning with the cross entropy loss.

To accelerate the training, we propose to construct a small action space for each source input. In other words, our method selects a small vocabulary V' of size K for each source input in advance to the model training. In this section, we assume that V' is given and represented with a sparse binary matrix $M_X \in \mathbb{R}^{K \times |V|}$, where there are only K non-zero elements at position (i, w_i) for $1 \leq i \leq K$. w_i is a unique word index in V. M_X is used to construct a small subset of the parameters in the softmax layer:

$$W'_p = M_X W_p, \quad (12)$$
$$b'_p = M_X b_p, \quad (13)$$

and $W'_p \in \mathbb{R}^{K \times d}$ and $b'_p \in \mathbb{R}^K$ are used instead of W_p and b_p in Equation (6) and (7). Therefore, in mini-batched processes with a mini-batch size B, our method constructs B different sets of (W'_p, b'_p).

Relationship to previous work Sampling-based approximation methods have previously been studied to reduce the computational cost at the large softmax layer in probabilistic language modeling (Ji et al., 2016; Zoph et al., 2016), and such methods are also used to enable one to train neural machine translation models on CPUs (Eriguchi et al., 2016). The construction of (W'_p, b'_p) in our method is similar to these

softmax approximation methods in that they also sample small vocabularies either at the word level (Ji et al., 2016), sentence level (Hashimoto and Tsuruoka, 2017), or mini-batch level (Zoph et al., 2016). However, one significant difference is that the approximation methods work only at training time using the cross entropy loss, and full softmax computations are still required at test time. The difference is crucial because a sentence generation model needs to simulate its test-time behavior in reinforcement learning.

3 Target Vocabulary Prediction

The remaining question is how to construct the input-specific vocabulary V' for each source input X. This section describes our method to construct V' by using a vocabulary prediction model which is separated from the sentence generation models.

3.1 Input Representations

In the vocabulary prediction task, the input is the source input X (source sentences or images) to be described, and the output is V'. Note that we should be careful *not* to make the prediction model computationally expensive; otherwise the computational efficiency by our method would be canceled out.

To feed the information about X into our vocabulary prediction model, we define an input vector $v(X) \in \mathbb{R}^{d_v}$. For image captioning, we use the feature vector f described in Section 2.1:

$$v(X) = W_v f + b_v, \tag{14}$$

where $W_v \in \mathbb{R}^{d_v \times d_f}$ is a weight matrix, and $b_v \in \mathbb{R}^{d_v}$ is a bias vector. For machine translation, we employ a bag-of-embeddings representation:[4]

$$v(X) = \frac{1}{M} \sum_{i=1}^{M} \tilde{e}_v(x_i), \tag{15}$$

where the d_v-dimensional word embedding $\tilde{e}_v(x_i) \in \mathbb{R}^{d_v}$ is different from $\tilde{e}(x_i)$ used in the machine translation model. By using the different set of the model parameters, we avoid the situation that our vocabulary prediction model is affected during training the sentence generation models.

[4]We observed that more complex representations using RNNs significantly slow down the vocabulary prediction process with no significant accuracy improvement.

Relationship to previous work Vocabulary prediction has gained attention for training sequence-to-sequence models with the cross entropy loss (Weng et al., 2017; Wu et al., 2017), but not for reinforcement learning. Compared to our method, previous methods jointly train a vocabulary predictor by directly using source encoders as input to the predictor. One may expect joint learning to improve both of the vocabulary predictor and the sentence generator, but in practice such positive effects are not clearly observed. Weng et al. (2017) reported that the joint learning improves the accuracy of their machine translation models, but our preliminary experiments did not indicate such accuracy gain. Such a joint training approach requires the model to continuously update the vocabulary predictor during the REINFORCE step, because the encoder is shared. That is, the action space for each input changes during reinforcement learning, and we observed unstable training. Therefore, for the sake of simplicity, we separately model the vocabulary predictor and focus on the effects of using the small vocabularies in reinforcement learning in this work.

Another note is that Jean et al. (2015) and L'Hostis et al. (2016) also proposed to construct small vocabularies in advance to the cross entropy-based training. They suggest that the use of word alignment works well, but using the word alignment is not general enough, considering that there exist different types of source input. By contrast, our method can be straightforwardly applied to the two sentence generation tasks with the different input modalities (i.e. image and text).

3.2 Vocabulary Prediction as Multi-Label Classification

Once the input representation $v(X)$ is computed, we further transform it by a single residual block (He et al., 2016): $r(X) = \mathrm{Res}\left(v(X)\right) \in \mathbb{R}^{d_v}$.[5] Then $r(X)$ is fed into a prediction layer:

$$o = \sigma\left(W_o r(X) + b_o\right), \tag{16}$$

where $W_o \in \mathbb{R}^{|V| \times d_v}$ is a weight matrix, and $b_o \in \mathbb{R}^{|V|}$ is a bias vector. The i-th element o_i corresponds to the probability that the i-th word in the target vocabulary V appears in the target sentence Y given its source X.

[5]We can use arbitrary types of hidden layers, but our preliminary experiments found this one performed the best in terms of the accuracy and convergence speed. We describe the details of the residual block in the supplemental material.

We use the training data for the sentence generations tasks to train the vocabulary predictor. For each X in the training data, we have its gold target sentence Y_g. We train the vocabulary predictor as a multi-label classification model by the following loss function:

$$-\sum_{i=1}^{|V|}\left(t_i \log o_i + (1-t_i)\log(1-o_i)\right), \quad (17)$$

where t_i is equal to 1.0 if the i-th word in V is included in Y_g, and otherwise t_i is 0.0. In practice, we apply the label smoothing technique (Szegedy et al., 2016) to the loss function.

We evaluate the accuracy of the vocabulary predictor by using a separate development split D:

$$\frac{\#\text{ of correctly predicted words in } D}{\#\text{ of words in } D}, \quad (18)$$

where we select the top-K predictions in Equation (16) for each source input X in D, and the evaluation metric is a recall score. We use the top-K words to construct the input-specific vocabularies V' for the sentence generation models, and we restrict that the recall is 100% for the training data.

4 Experimental Settings

We describe our experimental settings, and the details can be found in the supplemental material. We used PyTorch 0.2.0 to implement our models.

4.1 Datasets

We used machine translation datasets of four different language pairs: English-to-German (En-De), English-to-Japanese (En-Ja), English-to-Vietnamese (En-Vi), and Chinese-to-Japanese (Ch-Ja). For image captioning, we used two datasets: MS COCO (Lin et al., 2014) and Flickr8K. Table 1 summarizes the statistics of the training datasets, where the number of training examples ("Size"), the target vocabulary size ($|V|$), and the maximum length of the target sentences ($\max(N)$) are shown. For the machine translation datasets, we manually set $\max(N)$ and omitted training examples which violate the constraints.

En-De We used 100,000 training sentence pairs from `news commentary` and `newstest2015` as our development set, following Eriguchi et al. (2017).

Dataset	Size	$\lvert V\rvert$	$\max(N)$
En-De	100,000	24,482	50
En-Ja (100K)	100,000	23,536	50
En-Ja (2M)	1,998,821	70,668	100
En-Ja (2M, SW)	1,998,816	37,905	200
En-Vi	132,406	14,321	100
Ch-Ja	100,000	23,383	50
MS COCO	413,915	14,543	57
Flickr8K	30,000	4,521	38

Table 1: Statistics of the training datasets.

En-Ja We used parallel sentences in `ASPEC` (Nakazawa et al., 2016) and constructed three types of datasets: En-Ja (100K), En-Ja (2M), and En-Ja (2M, SW). The 100K and 2M datasets were constructed with the first 100,000 and 2,000,000 sentence pairs, respectively. To test our method using subword units, we further pre-processed the 2M dataset by using the `SentencePiece` toolkit[6] to construct the En-Ja (2M, SW) dataset.

En-Vi We used the pre-processed datasets provided by Luong and Manning (2015).[7] Our development dataset is the `tst2012` dataset.

Ch-Ja We constructed the Ch-Ja dataset by using the first 100,000 sentences from `ASPEC`.

MS COCO and Flickr8K We used the pre-processed datasets provided by Kiros et al. (2014).[8] We can also download the 4096-dimensional feature vectors f (i.e., $d_f = 4096$).

4.2 Settings of Vocabulary Prediction

We set $d_v = 512$ for all the experiments. We used `AdaGrad` (Duchi et al., 2011) to train the vocabulary predictor with a learning rate of 0.08 and a mini-batch size of 128. The model for each setting was tuned based on recall scores (with $K = 1000$) for the development split.

4.3 Settings of Sentence Generation

We set $d = 256$ with single-layer LSTMs for all the experiments, except for the En-Ja (2M) and (2M, SW) datasets. For the larger En-Ja datasets, we set $d = 512$ with two-layer LSTMs. We used stochastic gradient decent with momentum, with a learning rate of 1.0, a momentum rate of 0.75,

[6] https://github.com/google/sentencepiece

[7] https://nlp.stanford.edu/projects/nmt/

[8] https://github.com/ryankiros/visual-semantic-embedding.

		Cross entropy		REINFORCE w/ cross entropy	
		Small softmax	Full softmax	Small softmax	Full softmax
Translation	En-De	11.09±0.51	10.84±0.37	12.13±0.33	11.73±0.23
	En-Ja (100K)	28.26±0.15	28.05±0.40	29.14±0.13	29.01±0.35
	En-Vi	24.56±0.14	24.53±0.18	24.98±0.11	24.92±0.09
	Ch-Ja	29.27±0.08	28.97±0.15	30.10±0.12	29.80±0.15
Image captioning	MS COCO	24.88±0.25	24.75±0.36	26.43±0.32	25.74±0.13
	Flickr8K	16.45±0.28	16.52±0.11	19.04±0.43	19.17±0.24

Table 2: BLEU scores for the development splits of the six datasets. "Small softmax" corresponds to our method.

and a mini-batch size of 128. The model for each setting was tuned based on BLEU scores for the development split. All of the models achieved the best BLEU scores for all the datasets within 15 to 20 training epochs. For REINFORCE, we set $\lambda = 0.005$, and the learning rate was set to 0.01. The REINFORCE steps required around 5 epochs to significantly improve the BLEU scores.

4.4 Computational Resources

We used a single GPU of `NVIDIA GeForce GTX 1080`[9] to run experiments for the En-De, En-Ja (100K), En-Vi, Ch-Ja, MS COCO, and Flickr8K datasets. For the En-Ja (2M) and En-Ja (2M, SW) datasets, we used GPUs of `NVIDIA Tesla V100`[10] to speedup our experiments.

Multiple GPUs for full softmax It should be noted that our small softmax method can be run even on the single `GTX 1080` GPU for the larger translation datasets, whereas the full softmax method runs out of the GPU memory and requires multiple GPUs. We used four or eight `V100` GPUs on demand to run experiments for the larger datasets with the full softmax computations (i.e., the baseline results). When using multiple GPUs, we split the mini-batch computations in the softmax layer.

5 Results of Sentence Generation Tasks

5.1 Accuracy of Vocabulary Prediction

Figure 1 shows recall scores with respect to different values of the small vocabulary size K for each dataset. We can see that the recall scores reach 95% with $K = 1000$ for most of the datasets. One exception is the En-De dataset, and this is not surprising because a German vocabulary would become sparse by many compound nouns.

[9]The GPU memory capacity is 11,178MiB.

[10]The GPU memory capacity is 16,152MiB, and we used the `AWS p3` instances.

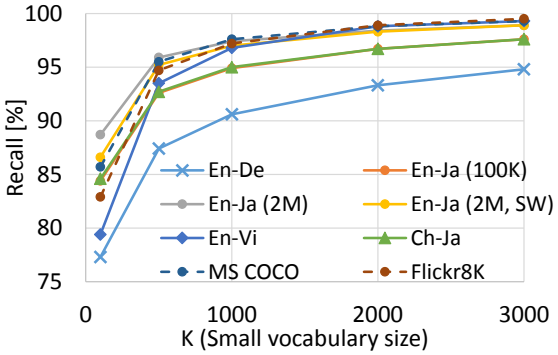

Figure 1: Recall scores of our vocabulary predictor.

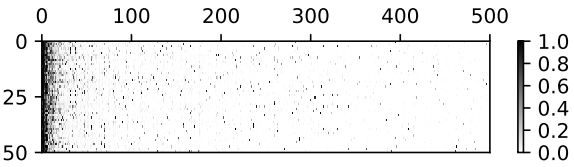

Figure 2: Visualization of a heatmap of the vocabulary prediction values for the En-Ja (100K) dataset. The vertical axis corresponds to 50 English sentences in the development split, and the horizontal axis corresponds to 500 most frequent words in the Japanese vocabulary.

These results show that our vocabulary predictor works well for source inputs of different modalities (text and image) and their corresponding different target languages. Our method also works at the subword level as well as at the standard word level. Figure 2 shows how our method works in the En-Ja (100K) dataset. In the heatmap, we can see that our method covers highly frequent target words as well as input-specific words.

For training the sentence generation models, we set $K = 500$ for the Flickr8K dataset and $K = 1000$ for the other datasets. Our empirical recommendation is $K = 1000$ if $|V|$ is larger than 10,000 and otherwise $K = 500$.

5.2 Accuracy of Sentence Generation

The goal of this paper is achieving efficient reinforcement learning for sentence generation to en-

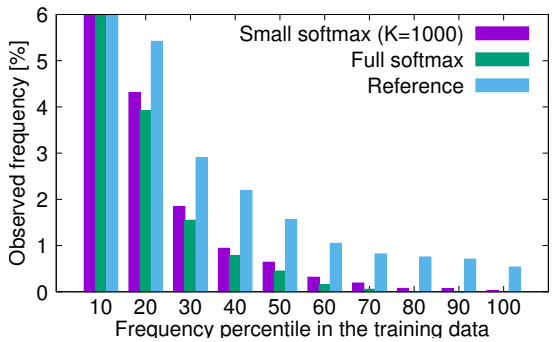

Figure 3: An analysis on the En-De translation results. The small and full softmax results are based on the REINFORCE setting, and "Reference" corresponds to the En-De reference translations of the development split.

courage future research in this are, but before evaluating the efficiency of our method, we show that the use of the small vocabularies does not degrade the accuracy of the sentence generation models. Table 2 shows BLEU scores for the development splits of the four machine translation and two image captioning datasets. The BLEU scores are averaged over five different runs with different random seeds, and the standard deviations are also reported. All the scores are obtained by greedy generations.

We can see in Table 2 that our method (Small softmax) keeps the BLEU scores as high as those of "Full softmax". For some datasets, the BLEU scores of our method are even better than those of the full softmax method. The trend is consistent in both of the cross entropy training phase and the REINFORCE phase. These results indicate that our method works well for different machine translation and image captioning datasets. We also confirmed that our experimental results are competitive with previously reported results when using the same training datasets; for example, our En-Vi test set result on `tst2013` is 27.87±0.21 (cf. 26.9 in Luong and Manning (2015)).

These BLEU scores suggest that our method for reinforcement learning has the potential to outperform the full softmax baseline. However, it is still unclear what is the potential advantage in terms of generation quality. We therefore analyzed the differences between output sentences of the small and full softmax methods, following Ott et al. (2018). Figure 3 shows the results of the En-De translation dataset, and we observed the same trend for all the other datasets. Each entry is computed as follows:

$$\frac{\#\text{ of output words in each percentile}}{\#\text{ of output words}}, \quad (19)$$

where the "10" percentile includes the top 10% of the most *frequent* words, and the "100" percentile includes the top 10% of the most *infrequent* words. We can see that our small softmax method better outputs rare words, and these results suggest that using input-specific vocabularies is useful in controlling action spaces for reinforcement learning.

	2M	2M, SW
Cross entropy	38.76	39.15
w/ beam search	39.88	40.35
REINFORCE w/ cross entropy	40.10	40.26
w/ beam search	40.36	40.38
w/ beam search (K= 500)	40.07	40.07
w/ beam search (K=2000)	40.30	40.50
w/ beam search (K=3000)	40.27	40.41

Table 3: BLEU scores for the development split of the En-Ja (2M) and En-Ja (2M, SW) datasets.

REINFORCE w/ cross entropy (K=1000)	40.16
w/ beam search	40.50
- Cross entropy (1.3M) w/ beam search (Hashimoto and Tsuruoka, 2017)	39.42
- Cross entropy (2M) w/ beam search (Oda et al., 2017b)	40.29
- Cross entropy (2M+1M back-trans.) w/ beam search (Morishita et al., 2017)	41.42

Table 4: En-Ja BLEU scores for the test split. Our training dataset is the En-Ja (2M, SW) dataset. The 95% confidence interval by bootstrap resampling (Noreen, 1989) is [39.61, 41.47] for our beam search result.

Results on larger datasets To see whether our method works in larger scales, Table 3 shows BLEU scores for the development split when using the En-Ja (2M) and En-Ja (2M, SW) datasets.[11] These results show that our method consistently works even on these larger datasets at the word and subword levels. In this table we also report how our method works with beam search, and the greedy-based BLEU scores are very close to those of beam search after the REINFORCE phase. When performing a beam search, we can optionally use different sizes of the small vocabulary, but we observe that our method is robust to the changes, whereas Wu et al. (2017) reported that their dynamic vocabulary selection method is sensitive to such changes.

[11] For the 2M dataset, the full softmax baseline achieves BLEU scores of 38.67 and 39.84 for the "Cross entropy" and "REINFORCE w/ cross entropy" settings, respectively.

	$\|V\|$	$\max(N)$	Cross entropy		REINFORCE w/ cross entropy	
			Small softmax	Full softmax	Small softmax	Full softmax
En-Ja (100K)	23,536	50	4.6	4.8	10.1	21.2
En-Ja (2M)	70,668	100	157.5†	196.5‡	364.0†	979.5††
En-Vi	14,321	100	10.5	10.7	23.2	38.4
MS COCO	14,543	57	4.4	4.2	11.6	22.9
Flickr8K	4,521	38	0.3	0.3	0.8	0.9

Table 5: Training time [minutes/epoch] for the text generation models. †: one `V100` GPU, ‡: four `V100` GPUs, ††: eight `V100` GPUs, others: one `GTX 1080` GPU.

			CPU		GPU	
Data size	$\|V\|$	Model size	Small softmax	Full softmax	Small softmax	Full softmax
100K	23,536	1-L, 256-D	54.4	113.8	71.9	78.4
2M	70,668	2-L, 512-D	156.2	503.2	80.5	105.7
2M, SW	37,905	2-L, 512-D	161.0	369.2	84.8	99.2

Table 6: Average time [milliseconds] to obtain a translation for each sentence in the En-Ja development split.

	$\|V\|$	$\max(N)$	Cross entropy		REINFORCE w/ cross entropy	
			Small softmax	Full softmax	Small softmax	Full softmax
En-Ja (100K)	23,536	50	1,781	6,061	2,193	7,443
En-Ja (2M)	70,668	100	6,540†	62,708‡	8,760†	91,750††
En-Vi	14,321	100	2,149	10,645	2,909	10,807
MS COCO	14,543	57	1,419	8,587	1,785	10,651
Flickr8K	4,521	38	911	2,031	1,031	3,197

Table 7: Maximum memory consumption [MiB] on our GPU devices when training the text generation models. The results in this table correspond to those in Table 5.

For reference, we report the test set results in Table 4. We cite BLEU scores from previously published papers which reported results of single models (i.e., without ensemble). Our method with greedy translation achieves a competitive score. It should be noted that Morishita et al. (2017) achieve a better score presumably because they used additional in-domain one million parallel sentences obtained by the back-translation technique (Sennrich et al., 2016).

6 Efficiency of the Proposed Method

This section discusses our main contribution: how efficient our method is in accelerating reinforcement learning for sentence generation.

6.1 Speedup at Training and Test Time

We have examined the training-time efficiency of our method. Table 5 shows the training time [minutes/epoch] for five different datasets. We selected the five datasets to show results with different vocabulary sizes and different maximum sentence lengths, and we observed the same trend on the other datasets. The vocabulary size $|V|$ and the maximum sentence length $\max(N)$ are shown for each training dataset. In the training with the standard cross entropy loss, the speedup by our method is not impressive as long as the vocabulary size $|V|$ can be easily handled by the GPUs. Note, however that the "Full softmax" method for the En-Ja (2M) dataset uses multiple GPUs and our "Small softmax" method works on a single GPU as described in Section 4.4, which is the advantage of our method.

In the training with the REINFORCE algorithm, the speedup by our method is enhanced. In particular, in the En-Ja (2M) experiments, our method with a single GPU gains a factor of 2.7 speedup compared with the full softmax baseline with multiple GPUs. For most of the experimental settings, the speedup significantly accelerates our research and development cycles when working on reinforcement learning for sentence generation tasks. One exception is the Flickr8K dataset whose orig-

inal vocabulary size $|V|$ is already very small, and the lengths of the target sentences are short.

By the fact that our method works efficiently with reinforcement learning, we expect that our method also works well at test time. Table 6 shows the average decoding time [milliseconds] to generate a Japanese sentence given an English sentence for the En-Ja development split. For reference, the vocabulary size and the model size are also shown for each setting. We note that the decoding time of our method *includes* the time for constructing an input-specific vocabulary for each source input.

We can see that our method runs faster than "Full softmax"; in particular, the speedup is significant on CPUs, and the decoding time by our method is less sensitive to changing $|V|$ than that of "Full softmax". This is because our method handles the full vocabulary only once for each source input, whereas "Full softmax" needs to handle the full vocabulary every time the model predicts a target word.

6.2 GPU Memory Consumption

Our method is also efficient in terms of GPU memory consumption at training time. Table 7 shows the maximum GPU memory consumption during the training. These results show that our method easily fits in the memory of the single `GTX 1080` GPU, whereas "Full softmax" is very sensitive to the vocabulary size $|V|$ and the sentence lengths. In particular, we observe about 90% reduction in memory usage when using the En-Ja (2M) dataset, where we note that the maximum GPU memory consumption for "Full softmax" is computed as the total GPU memory consumption across multiple GPUs. By saving the memory usage, one could try using larger models, larger mini-batches, larger vocabularies, and longer target sentences without relying on multiple GPUs.

7 Related Work

Reducing the computational cost at the large softmax layer in language modeling/generation is actively studied (Jean et al., 2015; Ji et al., 2016; Eriguchi et al., 2016; L'Hostis et al., 2016; Zoph et al., 2016; Wu et al., 2017). Most of the existing methods try to reduce the vocabulary size by either negative sampling or vocabulary prediction. One exception is that Oda et al. (2017a) propose to predict a binary code of its corresponding target word. Although such a sophisticated method is promising, we focused on the vocabulary reduction method to apply policy-based reinforcement learning in a straightforward way.

Our method relies on the vocabulary predictor, and therefore the accuracy of the predictor is important. By further improving the recall of the predictor with a much smaller value of K, it is expected that our method enables reinforcement learning with much smaller action spaces. In that sense, using more advanced regularization or data augmentation methods such as subword regularization (Kudo, 2018) is an interesting direction for future work.

As reported in this paper, one simple way to define a reward function for reinforcement learning is to use task-specific automatic evaluation metrics (Ranzato et al., 2016; Wu et al., 2016; Rennie et al., 2017; Zhang and Lapata, 2017; Paulus et al., 2018), but this is limited in that we can only use training data with gold target sentences. An alternative approach is to use a *discriminator* in generative adversarial networks (Goodfellow et al., 2014), and Yang et al. (2018) showed that REINFORCE with such a discriminator improves translation accuracy. However, Yang et al. (2018) only used the training data, and thus the potential of the generative adversarial networks is not fully realized. One promising direction is to improve the use of the generative adversarial networks for the sentence generation tasks by using our method, because our method can also accelerate the combination of REINFORCE and the discriminator.

8 Conclusion

This paper has presented how to accelerate reinforcement learning for sentence generation tasks by reducing large action spaces. Our method is as accurate as, is faster than, and uses much less GPU memory than the standard full softmax counterpart, on sentence generation tasks of different modalities. In future work, it is interesting to use our method in generative adversarial networks to further improve the sentence generation models.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR1513, Japan.

References

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-Sequence Attentional Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to Parse and Translate Improves Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Alex Graves and Jurgen Schmidhuber. 2005. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, 18(5):602–610.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural Machine Translation with Source-Side Latent Graph Parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 125–135.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *Proceedings of the 14th European Conference on Computer Vision*, pages 630–645.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

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

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. In *Proceedings of the 5th International Conference on Learning Representations*.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456.

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.

Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. 2016. BlackOut: Speeding up Recurrent Neural Network Language Models With Very Large Vocabularies. In *Proceedings of the 4th International Conference on Learning Representations*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2342–2350.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv*, cs.LG 1411.2539.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75.

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary Selection Strategies for Neural Machine Translation. *arXiv*, cs.CL 1610.00072.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollar. 2014. Microsoft COCO: Common Objects in Context. *arXiv*, cs.CV 1405.0312.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domain. In *International Workshop on Spoken Language Translation*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. In *Proceedings of NIPS Deep Learning Workshop*.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2017. NTT Neural Machine Translation Systems at WAT 2017. In *Proceedings of the 4th Workshop on Asian Translation*, pages 89–94.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian Scientific Paper Excerpt Corpus. In *Proceedings of the 10th Conference on International Language Resources and Evaluation*.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley-Interscience.

Yusuke Oda, Philip Arthur, Graham Neubig, Koichiro Yoshino, and Satoshi Nakamura. 2017a. Neural Machine Translation via Binary Code Prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 850–860.

Yusuke Oda, Katsuhito Sudoh, Satoshi Nakamura, Masao Utiyama, and Eiichiro Sumita. 2017b. A Simple and Strong Baseline: NAIST-NICT Neural Machine Translation System for WAT2017 English-Japanese Translation Task. In *Proceedings of the 4th Workshop on Asian Translation*, pages 135–139.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing Uncertainty in Neural Machine Translation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3956–3965.

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*, pages 311–318.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A Deep Reinforced Model for Abstractive Summarization. In *Proceedings of the 6th International Conference on Learning Representations*.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

Marc?fAurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *Proceedings of the 4th International Conference on Learning Representations*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical Sequence Training for Image Captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

David Silver et al. 2017. Mastering the game of Go without human knowledge. *Nature*, 550:354–359.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jon Shlens. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Rongxiang Weng, Shujian Huang, Zaixiang Zheng, Xin-Yu Dai, and Jiajun Chen. 2017. Neural Machine Translation with Word Predictions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 136–145.

Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8:229–256.

Yonghui Wu et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv*, cs.CL 1609.08144.

Yu Wu, Wei Wu, Dejian Yang, Can Xu, Zhoujun Li, and Ming Zhou. 2017. Neural Response Generation with Dynamic Vocabularies. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1346–1355.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent Neural Network Regularization. *arXiv*, cs.NE 1409.2329.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Barret Zoph, Ashish Vaswani, Jonathan May, and Kevin Knight. 2016. Simple, Fast Noise-Contrastive Estimation for Large RNN Vocabularies. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1217–1222.

Supplemental Material

A Vocabulary Prediction Model

Residual block In Section 3.2, we used a residual block $r(X) = \mathrm{Res}(v(X)) \in \mathbb{R}^{d_v}$ inspired by He et al. (2016) to transform the input vector $v(X) \in \mathbb{R}^{d_v}$:

$$\begin{aligned} r_1 &= \mathrm{BN}_{r_1}(v(X)), \; r_2 = \tanh(r_1), \\ r_3 &= W_{r_3} r_2 + b_{r_3}, \; r_4 = \mathrm{BN}_{r_4}(r_3), \\ r_5 &= \tanh(r_4), \qquad r_6 = W_{r_6} r_5 + b_{r_6}, \\ r(X) &= r_6 + v(X), \end{aligned} \tag{20}$$

where $\mathrm{BN}_{r_1}(\cdot)$ and $\mathrm{BN}_{r_4}(\cdot)$ correspond to batch normalization (Ioffe and Szegedy, 2015), $W_{r_3} \in \mathbb{R}^{d_v \times d_v}$ and $W_{r_6} \in \mathbb{R}^{d_v \times d_v}$ are weight matrices, and $b_{r_3} \in \mathbb{R}^{d_v}$ and $b_{r_6} \in \mathbb{R}^{d_v}$ are bias vectors. We apply dropout (Hinton et al., 2012) to r_5 with a dropout rate of 0.4.

Label smoothing In Section 3.2, we applied label smoothing (Szegedy et al., 2016) to the loss function in Equation (17). More concretely, we modify the gold label t_i for the i-th target word as follows:

$$t_i \leftarrow (1.0 - \varepsilon) t_i + \varepsilon p(i), \tag{21}$$

where ε is a hyperparameter, and $p(i)$ is a prior probability that the i-th word appears in a target sentence. $p(i)$ is computed for each dataset:

$$p(i) = \frac{\sum_{j=1}^{|T|} t_i^j}{|T|}, \tag{22}$$

where $|T|$ is the size of the training dataset, and t_i^j is the gold label for the i-th target word in the j-th training example. Therefore, $p(i)$ roughly reflects the unigram frequency. We have empirically found that the recommended value $\varepsilon = 0.1$ consistently improves the recall of the predictor.

B Detailed Experimental Settings

Word segmentation The sentences in the En-Vi, MS COCO, and Flickr8K datasets were pre-tokenized. We used the `Kytea` toolkit for Japanese and the `Stanford Core NLP` toolkit for Chinese. In the other cases, we used the `Moses` word tokenizer. We lowercased all the English sentences. The En-Ja (2M, SW) dataset was obtained by the `SentencePiece` toolkit so that the vocabulary size becomes around 32,000.

Vocabulary construction We built the target language vocabularies with words appearing at least five times for the En-De dataset, seven times for the En-Ja (2M) dataset, three times for the Ch-Ja dataset, and twice for the other datasets.

Optimization We initialized all the weight and embedding matrices with uniform random values in $[-0.1, +0.1]$, and all the bias vectors with zeros, except for the LSTM forget-gate biases which were initialized with ones (Jozefowicz et al., 2015). For all the models, we used gradient-norm clipping (Pascanu et al., 2013) with a clipping value of 1.0. We applied dropout to Equation (5), (6), and (7) with a dropout rate of 0.2, and we further used dropout in the vertical connections of the two-layer LSTMs (Zaremba et al., 2014) for the En-Ja (2M) and (2M, SW) datasets. As regularization, we also used weight decay with a coefficient of 10^{-6}. When training the vocabulary predictor and the sentence generation models, we checked the corresponding evaluation scores at every half epoch, and halved the learning rate if the evaluation scores were not improved. We stabilized the training of the sentence generation models by not decreasing the learning rate in the first six epochs. These training settings were tuned for the En-Ja (100K) dataset, but we empirically found that the same settings lead to the consistent results for all the datasets.

Baseline Estimator We used the `Adam` optimizer with a learning rate of 10^{-3} and the other default settings, to optimize the baseline estimator in Section 2.2. We have found that our results are not sensitive to the training settings of the baseline estimator.

Beam search For the results in Table 3 and 4, we tried two beam search methods in Hashimoto and Tsuruoka (2017) and Oda et al. (2017b), and selected better scores for each setting. In general, these length normalization methods lead to the best BLEU scores with a beam size of 10 to 20.