Supplementary Material: 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

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 $v(X) \in \mathbb{R}^{d_v}$:

$$\begin{aligned} r_1 &= \mathrm{BN}_{r_1}(v(X)), \; r_2 = \tanh(r_1), \\ r_3 &= W_{r_3} r_2 + b_{r_3}, \; r_4 = \mathrm{BN}_{r_4}(r_3), \\ r_5 &= \tanh(r_4), \qquad r_6 = W_{r_6} r_5 + b_{r_6}, \\ r(X) &= r_6 + v(X), \end{aligned} \tag{13}$$

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 (10). More concretely, we modify the gold label t_i for the i-th target word as follows:

$$t_i \leftarrow (1.0 - \varepsilon) t_i + \varepsilon p(i), \tag{14}$$

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:

$$p(i) = \frac{\sum_{j=1}^{|T|} t_i^j}{|T|}, \tag{15}$$

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.

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

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 (3), (4), and (5) 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

			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.

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. (2017), 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.

C Test Time Efficiency

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.

References

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.

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.

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.

Yusuke Oda, Katsuhito Sudoh, Satoshi Nakamura, Masao Utiyama, and Eiichiro Sumita. 2017. 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.

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.

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*.

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