Key-value Attention Mechanism for Neural Machine Translation

Hideya Mino [1,2*] **Masao Utiyama** [3] **Eiichiro Sumita** [3] **Takenobu Tokunaga** [2]
[1]NHK Science & Technology Research Laboratories
[2]Tokyo Institute of Technology
[3]National Institute of Information and Communication Technology
mino.h-gq@nhk.or.jp {mutiyama, eiichiro.sumita}@nict.go.jp
take@c.titech.ac.jp

Abstract

In this paper, we propose a neural machine translation (NMT) with a key-value attention mechanism on the source-side encoder. The key-value attention mechanism separates the source-side content vector into two types of memory known as the key and the value. The key is used for calculating the attention distribution, and the value is used for encoding the context representation. Experiments on three different tasks indicate that our model outperforms an NMT model with a conventional attention mechanism. Furthermore, we perform experiments with a conventional NMT framework, in which a part of the initial value of a weight matrix is set to zero so that the matrix is at the same initial-state as the key-value attention mechanism. As a result, we obtain comparable results with the key-value attention mechanism without changing the network structure.

1 Introduction

Recently, neural machine translation (NMT) (Sutskever et al., 2014; Cho et al., 2014) has achieved impressive results owing to its capacity to model the translation process end-to-end within a single probabilistic model. The unique features of the most popular approaches to NMT comprise a encoder-decoder architecture comprising recurrent neural networks (RNNs) and an attention mechanism, whereby the decoder can attend directly to localized information from source sequence tokens for generating a target sequence (Bahdanau et al., 2015; Luong et al., 2015). The encoder-decoder architecture predicts the target word with a target hidden-state and a context vector. This context vector is calculated as a weighted average over all source hidden-states. The weight of a source hidden-state is calculated as the inner product of the source hidden-state and the target word hidden-state. Note that the source hidden-state acts as the key to weight itself. It also acts as the value to predict the target word through the context vector. Daniluk et al. (2017) suppose that the dual use of a single vector makes training the model difficult and propose the use of a key-value paired structure, which is a generalized way of storing content in the vector.

In this paper, we propose splitting the matrix of the source hidden-states into two parts, an approach suggested by Daniluk et al. (2017) and Miller et al. (2016). The first part refers to the key used to calculate the attention distribution or weights. The second part refers to the value for the source-side context representation.

We empirically demonstrate that the separation of the source-side context vector into the key and value significantly improves the performance of an NMT using three different English-to-Japanese translation tasks.

2 Related Work

A significant amount of research has been performed on the use of memory within neural networks. For instance, an RNN features an implicit memory in the form of recurring hidden states. However, a vanilla RNN is known to have difficulties in storing information for long time-spans (Bengio et al., 1994). To overcome this problem, LSTM (Hochreiter and Schmidhuber, 1997), or GRU (Cho et al., 2014), which contain memory cells with a recurrently self-connected linear unit

[*] This work was performed while the first author was affiliated with National Institute of Information and Communication Technology, Kyoto, Japan.

Proceedings of the The 8th International Joint Conference on Natural Language Processing, pages 290–295,
Taipei, Taiwan, November 27 – December 1, 2017

have been proposed.

Attention-based neural networks with soft or hard attention over an input have shown successful results in a wide range of tasks including machine translation (Bahdanau et al., 2015), sentence summarization (Rush et al., 2015), and image captioning (Xu et al., 2015). These attention-based networks use an encoded memory for both as the key and value as described in the Introduction to calculate the output.

In contrast to the dual use of a single memory vector, Miller et al. (2016) have proposed key-value memory networks with key- and value-memory vectors to solve question-answering tasks, which use a generalized approach to store content in the memory. The key-memory vectors are used to calculate the attention weights, which address relevant memories with respect to the question, whereas the value-memory vectors are used to calculate the contextual representation to predict the answer. Daniluk et al. (2017) introduce a key-value attention model for neural language modeling that separates output vectors into keys to calculate the attention distribution and values for encoding the next-word distribution and context representation. We also focus on the key-value attention model. Our approach differs from the approach of Daniluk et al. (2017) in that they use it for the language model only; in contrast we use the key-value attention to encode the source-side context and predict the target-side word for translation.

3 Method

3.1 NMT with Attention

Our work is based on an attention-based NMT (Luong et al., 2015), which generates a target sentence $\boldsymbol{y} = (y_1, ..., y_N) \in \mathbb{R}^{V_t \times N}$ from the source sentence $\boldsymbol{x} = (x_1, ..., x_M) \in \mathbb{R}^{V_s \times M}$. V_s and V_t denote the vocabulary size of the source and target side, respectively. The attention-based model comprises two components, an encoder and a decoder. The encoder embeds the source sentence $\boldsymbol{x}$ into vectors through an embedding matrix and produces the hidden states using a bidirectional RNN, which represents a forward and a backward sequence. Thus, we have

$$\overrightarrow{h}_i = \text{enc}_1(\boldsymbol{W}_s x_i, \overrightarrow{h}_{i-1}), \quad (1)$$

$$\overleftarrow{h}_i = \text{enc}_2(\boldsymbol{W}_s x_i, \overleftarrow{h}_{i+1}). \quad (2)$$

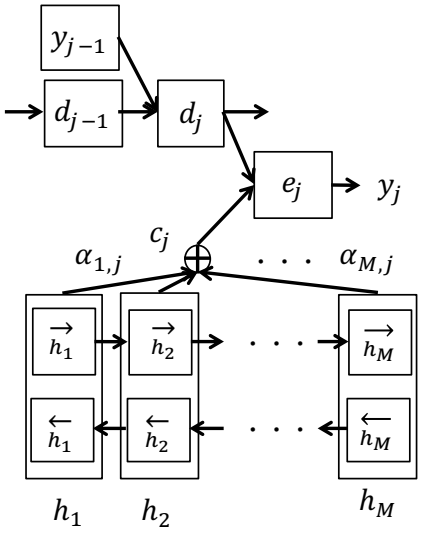

Figure 1: Encoder-decoder NMT architecture

$\boldsymbol{W}_s \in \mathbb{R}^{K \times V_s}$ is an embedding matrix where K is the word embedding size, and enc_1 and enc_2 are nonlinear functions as in LSTM. Then, as illustrated in Figure 1, the forward and backward hidden states $\overrightarrow{\boldsymbol{h}}$ and $\overleftarrow{\boldsymbol{h}}$ are concatenated into the hidden states $\boldsymbol{h} = (h_1, ..., h_M) \in \mathbb{R}^{K \times M}$ as

$$h_i = \boldsymbol{W}_e[\overrightarrow{h}_i^\top; \overleftarrow{h}_i^\top]^\top, \quad (3)$$

where $\boldsymbol{W}_e \in \mathbb{R}^{K \times 2K}$ is a matrix for the affine transform. Each hidden state, represented as a single vector, can be seen a memory vector that includes not only the lexical information at its source position, but also information about the left and right contexts. Then, the decoder predicts the target sentence $\boldsymbol{y}$ using a conditional probability calculated as bellow:

$$p(y_j|\boldsymbol{y}_{1,j-1}, \boldsymbol{x}) = \text{softmax}(\boldsymbol{W}_o e_j + b_o), \quad (4)$$

where $\boldsymbol{W}_o \in \mathbb{R}^{V_t \times K}$ and $b_o \in \mathbb{R}^{V_t}$ are implemented as a matrix and a bias of a feedforward neural network with a softmax output layer. $e_j \in \mathbb{R}^K$ is calculated by concatenating a hidden state with a context vector, and performing an affine transform with $\tanh$ function as

$$e_j = \tanh(\boldsymbol{W}_d[d_j; c_j]^\top), \quad (5)$$

where $\boldsymbol{W}_d \in \mathbb{R}^{K \times 2K}$ is a matrix for the affine transform; $d_j \in \mathbb{R}^K$ is the hidden state of the decoder RNN; and $c_j \in \mathbb{R}^K$ is the context vector derived from the source sentence. d_j is a fixed-length continuous vector computed by

$$d_j = \text{dec}(d_{j-1}, y_{j-1}). \quad (6)$$

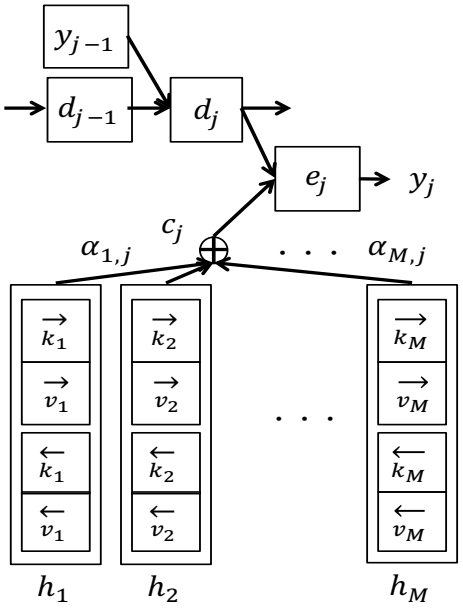

Figure 2: Encoder-decoder NMT architecture with key-value attention

Here dec is a nonlinear function analogous to enc1 or enc2; d_1 is set to a matrix of an affine transformation of the last hidden state h_M. The context vector c_j is computed as a convex sum of the hidden states h_i of Equation (3):

$$c_j = \sum_{i=1}^{M} \alpha_{i,j} h_i, \tag{7}$$

where $\alpha_{i,j}$, known as the attention weight, is a scalar weight computed by

$$\alpha_{i,j} = \frac{\exp\{\text{score}(d_{j-1}, h_i)\}}{\sum_{l=1}^{M} \exp\{\text{score}(d_{j-1}, h_l)\}}, \tag{8}$$

where the score function is referred as a content-based function and can be an arbitrary similarity function. We use the dot product, following Luong et al. (2015).

3.2 NMT with Key-value Attention

Attention-based NMT encodes an arbitrary sequence of source-side words into fixed-length dense vectors as in $\boldsymbol{h}$ in Eq. (3), which are used to calculate the attention weights and the context vectors as in Equations 8 and (7). However, the requirement to compress all necessary information into a single memory vector in each memory slot is likely to cause performance deficiencies. Therefore, to alleviate this problem, Miller et al. (2016) and Daniluk et al. (2017) propose the use of a separate vector depending on the purpose. Inspired by them, we introduce a key-value attention mechanism into NMT to calculate the context vector with explicit separate vectors as shown in Figure 2. The encoder embeds the source sentence $\boldsymbol{x}$ and produces hidden states $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ as in Equations (1) and (2). Then, the two hidden states are decomposed into two respective parts, which are a key and a value, as

$$\overrightarrow{h_i} = \begin{bmatrix} \overrightarrow{k}_i \\ \overrightarrow{v}_i \end{bmatrix}, \overleftarrow{h_i} = \begin{bmatrix} \overleftarrow{k}_i \\ \overleftarrow{v}_i \end{bmatrix}, \tag{9}$$

where the number of dimensions of the keys $\overrightarrow{k}_i$ and $\overleftarrow{k}_j$ and the values $\overrightarrow{v}_i$ and $\overleftarrow{v}_i$ is $K/2$. The forward and backward hidden-states $\overrightarrow{\boldsymbol{k}}$ $\overleftarrow{\boldsymbol{k}}$ $\overrightarrow{\boldsymbol{v}}$ and $\overleftarrow{\boldsymbol{v}}$ are concatenated into the hidden states $\boldsymbol{h}$ as

$$h_i = \begin{bmatrix} k_i \\ v_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{W}_f \begin{bmatrix} \overrightarrow{k}_i \\ \overleftarrow{k}_i \end{bmatrix} \\ \boldsymbol{W}_g \begin{bmatrix} \overrightarrow{v}_i \\ \overleftarrow{v}_i \end{bmatrix} \end{bmatrix}, \tag{10}$$

where $\boldsymbol{W}_f \in \mathbb{R}^{K/2 \times K}$ and $\boldsymbol{W}_g \in \mathbb{R}^{K/2 \times K}$ are matrices for the affine transform. The hidden states $\boldsymbol{k}$ and $\boldsymbol{v}$ indicate the key- and value-memory vector, respectively. Then, the decoder predicts the target sentence $\boldsymbol{y}$ using a conditional probability calculated as in Equations (4), (5), and (6). The context vector c_j in Eq. (5) is computed as a convex sum of the value memory $\boldsymbol{v}$ in Equation (10):

$$c_j = \sum_{i=1}^{M} \alpha_{i,j} v_i \tag{11}$$

where $\alpha_{i,j}$ is calculated with the key memory k_i as

$$\alpha_{i,j} = \frac{\exp\{\text{score}(d_{j-1}, k_i)\}}{\sum_{l=1}^{M} \exp\{\text{score}(d_{j-1}, k_l)\}}. \tag{12}$$

We also use the dot product for the score.

3.3 NMT Modifying Initial Weight

Another approach to alleviating this problem is modifying the score function in Eq. (8) and the initial value of the weight $\boldsymbol{W}_d$ in Eq. (5) of Section 3.1. The benefit of this approach is that the modification to the source code is minimal and the $\boldsymbol{W}_d$ may be tuned for better values. We suppose

Corpus	Training					Development			Test		
	Sents.	Word types		Avg. length		Sents.	Word types		Sents.	Word types	
		en	ja	en	ja		en	ja		en	ja
IWSLT'07	40K	9K	10K	9.3	12.7	0.5K	1.2K	1.3K	0.5K	0.8K	0.9K
NTCIR-10	717K	105K	79K	23.3	27.7	2.0K	5.0K	4.4K	0.5K	2.4K	2.1K
ASPEC	843K	288K	143K	22.1	23.9	1.8K	7.1K	6.3K	1.8K	7.0K	6.4K

Table 1: Datasets

that the upper half of the hidden states $\boldsymbol{h}$ in Eq. (3) produced by the encoder is used to calculate the alignment weight and the lower half of $\boldsymbol{h}$ is used to encode the source-side context vector. Then, we present two modifications. Firstly, since the score function in Eq. (8) calculates the alignment weight, we modify Eq. (8) to be zero for the lower half of the output of the score function as

$$\alpha_{i,j} = \frac{\exp\{\text{score}(d_{j-1}, h_i \odot u)\}}{\sum_{l=1}^{M} \exp\{\text{score}(d_{j-1}, h_l \odot u)\}}, \quad (13)$$

where $u \in \mathbb{R}^K$ is a vector for masking of which the upper half is one and the lower half is zero, and $\odot$ denotes the element-wise multiplication operation of the two vectors. Secondly, e_j in Eq. (5) is calculated with the context vector c_j, of which the upper half should not be used hereafter. Therefore, we set the initial weight of $\boldsymbol{W}_d$ to

$$\begin{bmatrix} w_{1,1} & \dots & w_{1,k} & 0 & \dots & 0 & w_{1,3K/2} & \dots & w_{1,2K} \\ & & & & \vdots & & & & \\ w_{K/2,1} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & w_{K/2,2K} \\ & & & & \vdots & & & & \\ w_{K,1} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & w_{K,2K} \end{bmatrix}.$$

The particular concern is that, unlike Section 3.2, the upper and lower halves of $\boldsymbol{h}$ in the model are not completely independent though the upper and lower halves of the initial state are independently used to train the model.

The objective of the three methods in this section is to jointly maximize the conditional probability for each generated target word as

$$\theta^* = \arg\max_{\theta} \sum_{t=1}^{T} \sum_{j=1}^{L_t} \log p(y_j^t | \mathbf{y}_{1,j-1}^t, \mathbf{x}^t, \theta), \quad (14)$$

where $(\mathbf{x}^t, \mathbf{y}^t)$ is the t-th training pair of sentences, and L_t is the length of the t-th target sentence $\mathbf{y}^t$.

4 Experiments

We evaluate the proposed method using three different English-to-Japanese translation tasks.

4.1 Data and Model Parameters

The corpora used were IWSLT'07 (Fordyce, 2007), NTCIR-10 (Goto et al., 2013), and ASPEC (Nakazawa et al., 2016) shown in Table 1. We constrained training sentences to a maximum length of 40 words to speed up the training. Each test sentence had a single reference translation.

4.2 Settings

The inputs and outputs of our model are sequences of one-hot vectors with dimensionality corresponding to the sizes of the source and target vocabularies. For NTCIR-10 and ASPEC, we replaced words with frequencies less than 3 with the [*UNK*] symbol and excluded them from the vocabularies. Each source and target word was projected into a 540-dimensional continuous Euclidean space to reduce the dimensionality. The depth of the stacking LSTMs was 2 and the hidden-layer size was set to 540. Each model was optimized using Adam (Kingma and Ba, 2014) with the following parameters: $\alpha = 1e-3$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$. To prevent overfitting we used dropout (Srivastava et al., 2014) with a drop rate of $r = 0.5$ to the last layer of each stacking LSTM. All weight matrices of each model were initialized by sampling from a normal distribution of 0 mean and 0.05 standard deviation. The gradient at each update was calculated using a minibatch of at most 64 sentence pairs which was run for a maximum of 20 iterations for the entire training data. Training was early-stopped to maximize the performance on the development set measured by BLEU. We used a single Tesla K80 GPU with 12 GB memory for training. For decoding, we used a beam search with a beam size of 10. The beam search was terminated when an end-of-sentence [*EOS*] symbol was generated. We used Chainer 1.21.0 (Tokui et al., 2015) to implement all the models.

System	
source	This makes it difficult to reproduce by a thin film multi-reproducing head .
reference	この ため **薄膜 マルチ 再生 ヘッド (a thin film multi-reproducing head)** に よ る 再生 が 困難 と な る 。
attn	これ に よ り 、 **薄 い (a thin) 薄膜 磁気 ヘッド (thin film reproducing head)** に よ る 再生 が 困難 に な る 。
key-value	これ に よ り 、 **薄膜 磁気 ヘッド (a thin film reproducing head)** に よ る 再生 は 困難 で あ る 。
modifying-IW	よ っ て 、 **薄膜 磁気 ヘッド (a thin film reproducing head)** に よ る 再生 が 困難 で あ る 。

Figure 3: Examples of the outputs

System	IWSLT'07	NTCIR-10	ASPEC
attn	49.1	31.1	29.6
key-value	49.3	33.8 †	30.7 †
modifying-IW	49.6	32.6 †	30.0

Table 2: BLEU scores for the attention-based NMT (*attn*), NMT with the key-value attention (*key-value*), and NMT modifying initial weight (*modifying-IW*) (†: significantly better than *attn* ($p < 0.05$).

5 Results

Table 2 summarizes the results for all the three tasks. NMT with the key-value attention achieved statistically significant results for the experiments with NTCIR-10 and ASPEC, though the experiments with IWSLT07 showed no such statistically significant results. The reason for our model's small difference in BLEU for IWSLT07 is likely due to the low number of word types used. The number of word types used in IWSLT07 was much lower than in the others, as presented in Table 1. Our model can be considered to be more effective for tasks with a vast vocabulary size. The results with NMT modifying initial weight are almost comparable to NMT with the key-value attention. Figure 3 shows the example of the outputs with each model. Though the attention-based NMT translates the same part of the sentence ("thin") twice, the NMT with the key-value attention and the NMT modifying initial weight translate correctly. These results show that the use of the separate memories for every different purpose improve the NMT translation quality and the initial weights of the hidden layers are likely to be able to control a single memory to keep dealing with the key and the value as separate as possible. For the training and translation speed, we did not observe large difference between these three models, since the three models have almost same number of parameters.

6 Conclusion

We propose a new method with the key-value attention mechanism in order to make the attention mechanism simpler. Our empirical evaluation shows that the proposed method is effective in achieving substantial improvements in terms of translation quality consistently across three different tasks.

Acknowledgments

This work was partially supported by the program "Promotion of Global Communications Plan: Research, Development, and Social Demonstration of Multilingual Speech Translation Technology" of MIC, Japan.

References

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

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5(2):157–166.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. 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)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734.

Michal Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. *CoRR* abs/1702.04521.

Cameron Shaw Fordyce. 2007. Overview of the 4th international workshop on spoken language translation iwslt 2007 evaluation campaign. In *In Proceedings of IWSLT 2007*. Trento, Italy, pages 1–12.

Isao Goto, Ka-Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K. Tsou. 2013. Overview of the patent machine translation task at the NTCIR-10 workshop. In *Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-10, National Center of Sciences, Tokyo, Japan, June 18-21, 2013*.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

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*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 1400–1409.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Portoro, Slovenia, pages 2204–2208.

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*. Association for Computational Linguistics, Lisbon, Portugal, pages 379–389.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 3104–3112.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. pages 2048–2057.