

# Neural Machine Translation with Key-Value Memory-Augmented Attention

Fandong Meng, Zhaopeng Tu, Yong Cheng, Haiyang Wu, Junjie Zhai, Yuekui Yang, Di Wang

Tencent AI Lab

{fandongmeng,zptu,yongcheng,gavinwu,jasonzhai,yuekuiyang,diwang}@tencent.com

## Abstract

Although attention-based Neural Machine Translation (NMT) has achieved remarkable progress in recent years, it still suffers from issues of repeating and dropping translations. To alleviate these issues, we propose a novel *key-value memory-augmented attention* model for NMT, called KVMEMATT. Specifically, we maintain a timely updated key-memory to keep track of attention history and a fixed value-memory to store the representation of source sentence throughout the whole translation process. Via nontrivial transformations and iterative interactions between the two memories, the decoder focuses on more appropriate source word(s) for predicting the next target word at each decoding step, therefore can improve the adequacy of translations. Experimental results on Chinese $\Rightarrow$ English and WMT17 German $\Leftrightarrow$ English translation tasks demonstrate the superiority of the proposed model.

## 1 Introduction

The past several years have witnessed promising progress in Neural Machine Translation (NMT) [Cho *et al.*, 2014; Sutskever *et al.*, 2014], in which attention model plays an increasingly important role [Bahdanau *et al.*, 2015; Luong *et al.*, 2015; Vaswani *et al.*, 2017]. Conventional attention-based NMT encodes the source sentence as a sequence of vectors after bi-directional recurrent neural networks (RNN) [Schuster and Paliwal, 1997], and then generates a variable-length target sentence with another RNN and an attention mechanism. The attention mechanism plays a crucial role in NMT, as it shows which source word(s) the decoder should focus on in order to predict the next target word. However, there is no mechanism to effectively keep track of attention history in conventional attention-based NMT. This may let the decoder tend to ignore past attention information, and lead to the issues of repeating or dropping translations [Tu *et al.*, 2016]. For example, conventional attention-based NMT may repeatedly translate some source words while mistakenly ignore other words.

A number of recent efforts have explored ways to alleviate the inadequate translation problem. For example, Tu *et al.* [2016] employ coverage vector as a lexical-level indicator

to indicate whether a source word is translated or not. Meng *et al.* [2016] and Zheng *et al.* [2018] take the idea one step further, and directly model translated and untranslated source contents by operating on the attention context (i.e., the partial source content being translated) instead of on the attention probability (i.e., the chance of the corresponding source word is translated). Specifically, Meng *et al.* [2016] capture translation status with an interactive attention augmented with a NTM [Graves *et al.*, 2014] memory. Zheng *et al.* [2018] separate the modeling of translated (PAST) and untranslated (FUTURE) source content from decoder states by introducing two additional decoder adaptive layers.

Meng *et al.* [2016] propose a generic framework of memory-augmented attention, which is independent from the specific architectures of the NMT models. However, the original mechanism takes only a single memory to both represent the source sentence and track attention history. Such overloaded usage of memory representations makes training the model difficult [Rocktäschel *et al.*, 2017]. In contrast, Zheng *et al.* [2018] try to ease the difficulty of representation learning by separating the PAST and FUTURE functions from the decoder states. However, it is designed specifically for the precise architecture of NMT models.

In this work, we combine the advantages of both models by leveraging the generic memory-augmented attention framework, while easing the memory training by maintaining separate representations for the expected two functions. Partially inspired by [Miller *et al.*, 2016], we split the memory into two parts: a *dynamical key-memory* along with the update-chain of the decoder state to keep track of attention history, and a *fixed value-memory* to store the representation of source sentence throughout the whole translation process. In each decoding step, we conduct multi-rounds of memory operations repeatedly layer by layer, which may let the decoder have a chance of re-attention by considering the “intermediate” attention results achieved in early stages. This structure allows the model to leverage possibly complex transformations and interactions between 1) the key-value memory pair in the same layer, as well as 2) the key (and value) memory across different layers.

Experimental results on Chinese $\Rightarrow$ English translation task show that attention model augmented with a single-layer key-value memory improves both translation and attention performances over not only a standard attention model, but also

over the existing NTM-augmented attention model [Meng *et al.*, 2016]. Its multi-layer counterpart further improves model performances consistently. We also validate our model on bidirectional German $\leftrightarrow$ English translation tasks, which demonstrates the effectiveness and generalizability of our approach.

## 2 Background

**Attention-based NMT** Given a source sentence  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$  and a target sentence  $\mathbf{y} = \{y_1, y_2, \dots, y_m\}$ , NMT models the translation probability word by word:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \prod_{t=1}^m P(y_t | \mathbf{y}_{<t}, \mathbf{x}; \theta) \\ &= \prod_{t=1}^m \text{softmax}(f(\mathbf{c}_t, y_{t-1}, \mathbf{s}_t)) \end{aligned} \quad (1)$$

where  $f(\cdot)$  is a non-linear function, and  $\mathbf{s}_t$  is the hidden state of decoder RNN at time step  $t$ :

$$\mathbf{s}_t = g(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c}_t) \quad (2)$$

$\mathbf{c}_t$  is a distinct source representation for time  $t$ , calculated as a weighted sum of the source annotations:

$$\mathbf{c}_t = \sum_{j=1}^n a_{t,j} \mathbf{h}_j \quad (3)$$

where  $\mathbf{h}_j$  is the encoder annotation of the source word  $x_j$ , and the weight  $a_{t,j}$  is computed by

$$a_{t,j} = \frac{\exp(e_{t,j})}{\sum_{k=1}^N \exp(e_{t,k})} \quad (4)$$

where  $e_{t,j} = \mathbf{v}_a^T \tanh(\mathbf{W}_a \tilde{\mathbf{s}}_{t-1} + \mathbf{U}_a \mathbf{h}_j)$  scores how much  $\tilde{\mathbf{s}}_{t-1}$  attends to  $\mathbf{h}_j$ , where  $\tilde{\mathbf{s}}_{t-1} = g(\mathbf{s}_{t-1}, y_{t-1})$  is an intermediate state tailored for computing the attention score with the information of  $y_{t-1}$ .

The training objective is to maximize the log-likelihood of the training instances  $(\mathbf{x}^s, \mathbf{y}^s)$ :

$$\theta^* = \arg \max_{\theta} \sum_{s=1}^S \log p(\mathbf{y}^s | \mathbf{x}^s) \quad (5)$$

**Memory-Augmented Attention** Meng *et al.* [2016] propose to augment the attention model with a memory in the form of NTM, which aims at tracking the attention history during decoding process, as shown in Figure 1. At each decoding step, the NTM employs a network-based reader to read from the encoder annotations and output a distinct memory representation, which is subsequently used to update the decoder state. After predicting the target word, the updated decoder state is written back to the memory, which is controlled by a network-based writer. As seen, the interactive read-write operations can timely update the representation of source sentence along with the update-chain of the decoder state and let the decoder keep track of the attention history.

However, this mechanism takes only a single memory to both represent the source sentence and track attention history. Such overloaded usage of memory representations makes training the model difficult [Rocktäschel *et al.*, 2017].

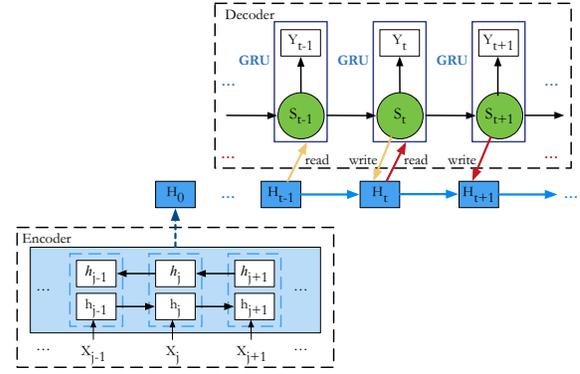


Figure 1: Illustration for the NTM-augmented attention. The yellow and red arrows indicate interactive read-write operations.

## 3 Model Description

Figure 2 shows the architecture of the proposed **Key-Value Memory-augmented Attention** model (KVMEMATT). It consists of three components: 1) an encoder (on the left), which encodes the entire source sentence and outputs its annotations as the initialization of the **KEY-MEMORY** and the **VALUE-MEMORY**; 2) the key-value memory-augmented attention model (in the middle), which generates the context representation of source sentence appropriate for predicting the next target word with iterative memory access operations conducted on the **KEY-MEMORY** and the **VALUE-MEMORY**; and 3) the decoder (on the right), which predicts the next target word step by step.

Specifically, the **KEY-MEMORY** and **VALUE-MEMORY** consists of  $n$  slots, which are initialized with the annotations of the source sentence  $\{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n\}$ . KVMEMATT-based NMT maintains these two memories throughout the whole decoding process, with the **KEY-MEMORY** keeping updated to track the attention history, and the **VALUE-MEMORY** keeping fixed to store the representation of source sentence. For example, the  $j$ -th slot ( $v_j$ ) in **VALUE-MEMORY** stores the representation of the  $j$ -th source word (fixed after generated), and the  $j$ -th slot ( $k_j$ ) in **KEY-MEMORY** stores the attention (or translation) status (updated as translation goes) corresponding to the  $j$ -th source word. At step  $t$ , the decoder state  $\mathbf{s}_{t-1}$  first meets the previous prediction  $y_{t-1}$  to form a “query” state  $\mathbf{q}_t$ , which can be calculated as follows

$$\mathbf{q}_t = \text{GRU}(\mathbf{s}_{t-1}, \mathbf{e}_{y_{t-1}}) \quad (6)$$

where  $\mathbf{e}_{y_{t-1}}$  is the word-embedding of the previous word  $y_{t-1}$ . The decoder uses the “query” state  $\mathbf{q}_t$  to *address* from the **KEY-MEMORY** looking for an accurate attention vector  $\mathbf{a}_t$ , and *reads* from the **VALUE-MEMORY** with the guidance of  $\mathbf{a}_t$  to generate the source context representation  $\mathbf{c}_t$ . After that the **KEY-MEMORY** is *updated* with  $\mathbf{q}_t$  and  $\mathbf{c}_t$ .

The memory access (i.e. *address*, *read* and *update*) in one decoding step can be conducted repeatedly, which may let the decoder have a chance of re-attention (with new information added) before making the final prediction. Suppose there are  $R$  rounds of memory access in each decoding step. The detailed operations from round  $r-1$  to round  $r$  are as follows:

First, we use the “query” state  $\mathbf{q}_t$  to address from  $\mathbf{K}_{(t)}^{(r-1)}$

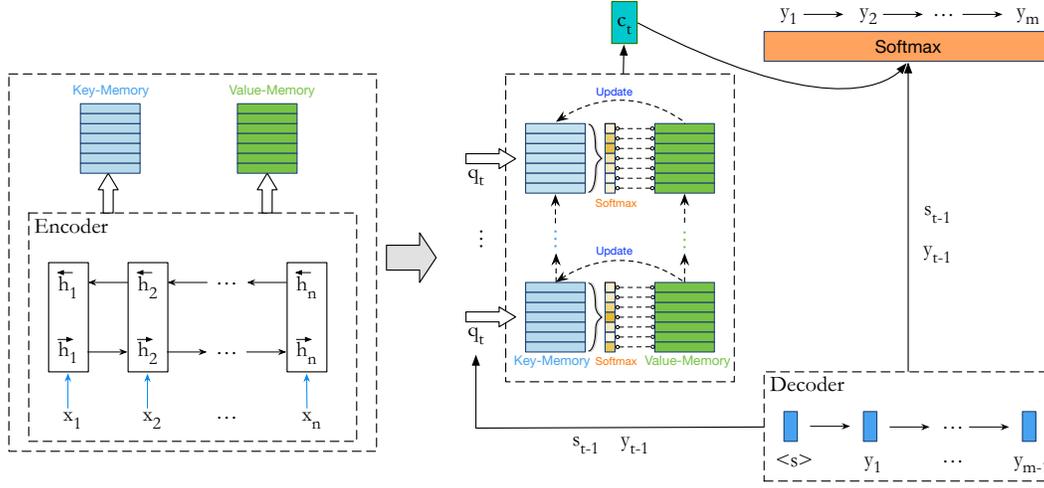


Figure 2: The architecture of KVMEMATT-based NMT. The encoder (on the left) and the decoder (on the right) are connected by the KVMEMATT (in the middle). In particular, we show the KVMEMATT at decoding time step  $t$  with multi-rounds of memory access.

to generate the “intermediate” attention vector  $\tilde{\mathbf{a}}_t^r$

$$\tilde{\mathbf{a}}_t^r = \mathbf{Address}(\mathbf{q}_t, \mathbf{K}_t^{(r-1)}) \quad (7)$$

which is subsequently used as the guidance for reading from the value memory  $\mathbf{V}$  to get the “intermediate” context representation  $\tilde{\mathbf{c}}_t^r$  of source sentence

$$\tilde{\mathbf{c}}_t^r = \mathbf{Read}(\tilde{\mathbf{a}}_t^r, \mathbf{V}) \quad (8)$$

which works together with the “query” state  $\mathbf{q}_t$  are used to get the “intermediate” hidden state:

$$\tilde{\mathbf{s}}_t^r = \mathbf{GRU}(\mathbf{q}_t, \tilde{\mathbf{c}}_t^r) \quad (9)$$

Finally, we use the “intermediate” hidden state to update  $\mathbf{K}_t^{(r-1)}$  as recording the “intermediate” attention status, to finish a round of operations,

$$\mathbf{K}_t^{(r)} = \mathbf{Update}(\tilde{\mathbf{s}}_t^r, \mathbf{K}_t^{(r-1)}) \quad (10)$$

After the last round ( $R$ ) of the operations, we use  $\{\tilde{\mathbf{s}}_t^R, \tilde{\mathbf{c}}_t^R, \tilde{\mathbf{a}}_t^R\}$  as the resulting states to compute a final prediction via Eq. 1. Then the KEY-MEMORY  $\mathbf{K}_t^{(R)}$  will be transited to the next decoding step  $t+1$ , being  $\mathbf{K}_{t+1}^{(0)}$ . The details of **Address**, **Read** and **Update** operations will be described later in next section.

As seen, the KVMEMATT mechanism can update the KEY-MEMORY along with the update-chain of the decoder state to keep track of attention status and also maintains a fixed VALUE-MEMORY to store the representation of source sentence. At each decoding step, the KVMEMATT generates the context representation of source sentence via nontrivial transformations between the KEY-MEMORY and the VALUE-MEMORY, and records the attention status via interactions between the two memories. This structure allows the model to leverage possibly complex transformations and interactions between two memories, and lets the decoder choose more appropriate source context for the word prediction at each

step. Clearly, KVMEMATT can subsume the coverage models [Tu *et al.*, 2016; Mi *et al.*, 2016] and the interactive attention model [Meng *et al.*, 2016] as special cases, while more generic and powerful, as empirically verified in the experiment section.

### 3.1 Memory Access Operations

In this section, we will detail the memory access operations from round  $r-1$  to round  $r$  at decoding time step  $t$ .

**Key-Memory Addressing** Formally,  $\mathbf{K}_t^{(r)} \in \mathbb{R}^{n \times d}$  is the KEY-MEMORY in round  $r$  at decoding time step  $t$  before the decoder RNN state update, where  $n$  is the number of memory slots and  $d$  is the dimension of vector in each slot. The addressed attention vector is given by

$$\tilde{\mathbf{a}}_t^r = \mathbf{Address}(\mathbf{q}_t, \mathbf{K}_t^{(r-1)}) \quad (11)$$

where  $\tilde{\mathbf{a}}_t^r \in \mathbb{R}^n$  specifies the normalized weights assigned to the slots in  $\mathbf{K}_t^{(r-1)}$ , with the  $j$ -th slot being  $\mathbf{k}_{(t,j)}^{(r-1)}$ . We can use content-based addressing to determine  $\tilde{\mathbf{a}}_t^r$  as described in [Graves *et al.*, 2014] or (quite similarly) use the soft-alignment as in Eq. 4. In this paper, for convenience we adopt the latter one. And the  $j$ -th cell of  $\tilde{\mathbf{a}}_t^r$  is

$$\tilde{\mathbf{a}}_{t,j}^r = \frac{\exp(\tilde{e}_{t,j}^r)}{\sum_{i=1}^n \exp(\tilde{e}_{t,i}^r)} \quad (12)$$

where  $\tilde{e}_{t,j}^r = \mathbf{v}_a^r T \tanh(\mathbf{W}_a^r \mathbf{q}_t + \mathbf{U}_a^r \mathbf{k}_{(t,j)}^{(r-1)})$ .

**Value-Memory Reading** Formally,  $\mathbf{V} \in \mathbb{R}^{n \times d}$  is the VALUE-MEMORY, where  $n$  is the number of memory slots and  $d$  is the dimension of vector in each slot. Before the decoder state update at time  $t$ , the output of reading at round  $r$  is  $\tilde{\mathbf{c}}_t^r$  given by

$$\tilde{\mathbf{c}}_t^r = \sum_{j=1}^n \tilde{\mathbf{a}}_{t,j}^r \mathbf{v}_j \quad (13)$$

where  $\tilde{\mathbf{a}}_t^r \in \mathbb{R}^n$  specifies the normalized weights assigned to the slots in  $\mathbf{V}$ .

**Key-Memory Updating** Inspired by the attentive writing operation of neural Turing machines [Graves *et al.*, 2014], we define two types of operation for updating the KEY-MEMORY: FORGET and ADD.

FORGET determines the content to be removed from memory slots. More specifically, the vector  $\mathbf{F}_t^r \in \mathbb{R}^d$  specifies the values to be forgotten or removed on each dimension in memory slots, which is then assigned to each slot through normalized weights  $\mathbf{w}_t^r$ . Formally, the memory (“intermediate”) after FORGET operation is given by

$$\tilde{\mathbf{K}}_{t,i}^{(r)} = \mathbf{K}_{t,i}^{(r-1)}(1 - \mathbf{w}_{t,i}^r \cdot \mathbf{F}_t^r), \quad i = 1, 2, \dots, n \quad (14)$$

where

- $\mathbf{F}_t^r = \sigma(\mathbf{W}_F^r \tilde{\mathbf{s}}_t^r)$  is parameterized with  $\mathbf{W}_F^r \in \mathbb{R}^{d \times d}$ , and  $\sigma$  stands for the *Sigmoid* activation function, and  $\mathbf{F}_t^r \in \mathbb{R}^d$ ;
- $\mathbf{w}_t^r \in \mathbb{R}^n$  specifies the normalized weights assigned to the slots in  $\mathbf{K}_{(t)}^{(r)}$ , and  $\mathbf{w}_{t,i}^r$  specifies the weight associated with the  $i$ -th slot.  $\mathbf{w}_t^r$  is determined by

$$\mathbf{w}_t^r = \text{Address}(\tilde{\mathbf{s}}_t^r, \mathbf{K}_{(t)}^{(r-1)}) \quad (15)$$

ADD decides how much current information should be written to the memory as the added content

$$\mathbf{K}_{t,i}^{(r)} = \tilde{\mathbf{K}}_{t,i}^{(r)} + \mathbf{w}_{t,i}^r \cdot \mathbf{A}_t^r, \quad i = 1, 2, \dots, n \quad (16)$$

where  $\mathbf{A}_t^r = \sigma(\mathbf{W}_A^r \tilde{\mathbf{s}}_t^r)$  is parameterized with  $\mathbf{W}_A^r \in \mathbb{R}^{d \times d}$ , and  $\mathbf{A}_t^r \in \mathbb{R}^d$ . Clearly, with FORGET and ADD operations, KVMEMATT potentially can modify and add to the KEY-MEMORY more than just history of attention.

### 3.2 Training

**EOS-attention Objective:** The translation process is finished when the decoder generates  $eos_{trg}$  (a special token that stands for the end of target sentence). Therefore, accurately generating  $eos_{trg}$  is crucial for producing correct translation. Intuitively, accurate attention for the last word (i.e.  $eos_{src}$ ) of source sentence will help the decoder accurately predict  $eos_{trg}$ . When to predict  $eos_{trg}$  (i.e.  $y_m$ ), the decoder should highly focus on  $eos_{src}$  (i.e.  $x_n$ ). That is to say the attention probability of  $a_{m,n}$  should be closed to 1.0. And when to generate other target words, such as  $y_1, y_2, \dots, y_{m-1}$ , the decoder should not focus on  $eos_{src}$  too much. Therefore we define

$$\text{ATTEOS} = \sum_{t=1}^m \text{Att}_{eos}(t, n) \quad (17)$$

where

$$\text{Att}_{eos}(t, n) = \begin{cases} a_{t,n}, & t < m \\ 1.0 - a_{t,n}, & t = m \end{cases} \quad (18)$$

**NMT Objective:** We plug Eq. 17 to Eq. 5, we have

$$\theta^* = \arg \max_{\theta} \sum_{s=1}^S \left( \log p(\mathbf{y}^s | \mathbf{x}^s) - \lambda * \text{ATTEOS}^s \right) \quad (19)$$

The EOS-attention objective can assist the learning of KVMEMATT and guiding the parameter training, which will be verified in the experiment section.

## 4 Experiments

### 4.1 Setup

We carry out experiments on Chinese $\Rightarrow$ English (Zh $\Rightarrow$ En) and German $\Leftrightarrow$ English (De $\Leftrightarrow$ En) translation tasks. For Zh $\Rightarrow$ En, the training data consist of 1.25M sentence pairs extracted from LDC corpora. We choose NIST 2002 (MT02) dataset as our valid set, and NIST 2003-2006 (MT03-06) datasets as our test sets. For De $\Leftrightarrow$ En, we perform our experiments on the corpus provided by WMT17, which contains 5.6M sentence pairs. We use newstest2016 as the development set, and newstest2017 as the testset. We measure the translation quality with BLEU scores [Papineni *et al.*, 2002].<sup>1</sup>

In training the neural networks, we limit the source and target vocabulary to the most frequent 30K words for both sides in Zh $\Rightarrow$ En task, covering approximately 97.7% and 99.3% of two corpus respectively. For De $\Leftrightarrow$ En, sentences are encoded using byte-pair encoding [Sennrich *et al.*, 2016], which has a shared source-target vocabulary of about 36000 tokens. The parameters are updated by SGD and mini-batch (size 80) with learning rate controlled by AdaDelta [Zeiler, 2012] ( $\epsilon = 1e^{-6}$  and  $\rho = 0.95$ ). We limit the length of sentences in training to 80 words for Zh $\Rightarrow$ En and 100 sub-words for De $\Leftrightarrow$ En. The dimension of word embedding and hidden layer is 512, and the beam size in testing is 10. We apply dropout on the output layer to avoid over-fitting [Hinton *et al.*, 2012], with dropout rate being 0.5. Hyper parameter  $\lambda$  in Eq. 19 is set to 1.0. The parameters of our KVMEMATT (i.e., encoder and decoder, except for those related to KVMEMATT) are initialized by the pre-trained baseline model.

### 4.2 Results on Chinese-English

We compare our KVMEMATT with two strong baselines: 1) RNNSEARCH, which is our in-house implementation of the attention-based NMT as described in Section 2; and 2) RNNSEARCH+MEMATT, which is our implementation of interactive attention [Meng *et al.*, 2016] on top of our RNNSEARCH. Table 1 shows the translation performance.

**Model Complexity** KVMEMATT brings in little parameter increase. Compared with RNNSEARCH, KVMEMATT-R1 and KVMEMATT-R2 only bring in 1.95% and 7.78% parameter increase. Additionally, introducing KVMEMATT slows down the training speed to a certain extent (18.39%~39.56%). When running on a single GPU device Tesla P40, the speed of the RNNSEARCH model is 2773 target words per second, while the speed of the proposed models is 1676~2263 target words per second.

**Translation Quality** KVMEMATT with one round memory access (KVMEMATT-R1) achieves significant improvements over RNNSEARCH by 1.92 BLEU and over RNNSEARCH+MEMATT by 0.65 BLEU averagely on four test sets. It indicates that our key-value memory-augmented attention mechanism can give more effective power for the

<sup>1</sup>For Zh $\Rightarrow$ En task, we apply case-insensitive NIST BLEU *meval-v11b.pl*. For De $\Leftrightarrow$ En tasks, we tokenized the reference and evaluated the performance with case-sensitive *multi-bleu.pl*. The metrics are exactly the same as in previous work.

SYSTEM	ARCHITECTURE	# Para.	Speed	MT03	MT04	MT05	MT06	Ave.
<i>Existing end-to-end NMT systems</i>								
[Tu <i>et al.</i> , 2016]	COVERAGE	–	–	33.69	38.05	35.01	34.83	35.40
[Meng <i>et al.</i> , 2016]	MEMATT	–	–	35.69	39.24	35.74	35.10	36.44
[Wang <i>et al.</i> , 2016]	MEMDEC	–	–	36.16	39.81	35.91	35.98	36.95
[Zhang <i>et al.</i> , 2017]	DISTORTION	–	–	37.93	40.40	36.81	35.77	37.73
<i>Our end-to-end NMT systems</i>								
this work	RNNSEARCH	53.98M	2773	36.02	38.65	35.64	35.80	36.53
	+ MEMATT	55.03M	2319	36.93*+	40.06*+	36.81*+	37.39*+	37.80
	+ KVMEMATT-R1	55.03M	2263	37.56*+	40.56*+	37.83*+	37.84*+	38.45
	+ ATTEOSOBJ	55.03M	1992	37.87*+	40.64+	38.01+	38.13*+	38.66
	+ KVMEMATT-R2	58.18M	1804	38.08+	40.73+	38.09+	38.80*+	38.93
	+ ATTEOSOBJ	58.18M	1676	38.40*+	41.10*+	38.73*+	39.08*+	39.33

Table 1: Case-insensitive BLEU scores (%) on Chinese⇒English translation task. “Speed” denotes the training speed (words/second). “R1” and “R2” denotes KVMEMATT with one and two rounds, respectively. “+ATTEOSOBJ” stands for adding the EOS-attention objective. The “\*” and “+” indicate that the results are significantly ( $p < 0.01$  with *sign-test*) better than the adjacent above system and the “RNNSEARCH”.

ARCHITECTURE	BLEU	△	AER	▽
RNNSEARCH	24.65	-	39.97	-
+ MEMATT	25.36	+0.71	37.98	-1.99
+ KVMEMATT-R1	25.55	+0.90	37.70	-2.27
+ ATTEOSOBJ	25.93	+1.28	36.89	-3.08
+ KVMEMATT-R2	26.43	+1.78	35.70	-4.27
+ ATTEOSOBJ	27.19	+2.54	35.39	-4.58

Table 2: Performance on manually aligned Chinese-English test set. Higher BLEU score and lower AER score indicate better quality.

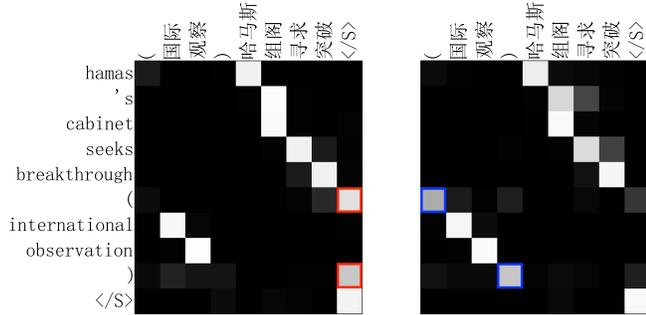


Figure 3: Example attention matrices of KVMEMATT-R2 generated from the first round (left) and the second round (right). We highlight some bad links with red frame and good links with blue frame.

attention via nontrivial transformations and interactions between the KEY-MEMORY and the VALUE-MEMORY. The two rounds counterpart (KVMEMATT-R2) can further improve the performance by 0.48 BLEU on average. It confirms our hypothesis that the decoder can benefit from the re-attention process, which considers the “intermediate” attention result achieved in the early stage and then makes a more accurate decision. However, we find that adding more than two rounds of memory access operations into KVMEMATT does not lead to better translation performance (not shown in the table). One possible reason is that memory access with more rounds leads to more updating operations (i.e. attentive writing) on the KEY-MEMORY, which may be difficult to op-

SYSTEMS	UNDERTRAN	OVERTRAN
RNNSEARCH	13.1%	2.7%
+ KVMEMATT-R2	9.7%	1.3%

Table 3: Subjective evaluation of translation adequacy. Numbers denote percentages of source words.

imize within our current architecture design. We will leave it as future work.

The contribution of adding EOS-attention Objective is to assist the learning of attention, and guiding the parameter training. It consistently improves translation performance over different variants of KVMEMATT. It gives a further improvement of 0.40 BLEU points over KVMEMATT-R2, which is 2.80 BLEU points better than RNNSEARCH.

**Alignment Quality** Intuitively, our KVMEMATT can enhance the attention and therefore improve the word alignment quality. To confirm our hypothesis, we carry out experiments of the alignment task on the evaluation dataset from [Liu and Sun, 2015], which contains 900 manually aligned Chinese-English sentence pairs. We use the alignment error rate (AER) [Och and Ney, 2003] as the evaluation metric for the alignment task. Table 2 lists the BLEU and AER scores. As expected, our KVMEMATT achieves better BLEU and AER scores (the lower the AER score, the better the alignment quality) than the strong baseline systems. Additionally, the results also indicate that the EOS-attention objective can assist the learning of attention-based NMT, since adding this objective yields better alignment performance. By visualizing the attention matrices, we found that the attention qualities are improved from the first round to the second round as expected, as shown in Figure 3.

**Subjective Evaluation** We did a subjective evaluation to investigate the benefit of incorporating KVMEMATT to NMT, especially on alleviating the issues of over- and under-translations. Table 3 lists the translation adequacy of the RNNSEARCH baseline and our KVMEMATT-R2 on the randomly selected 100 sentences from test sets. From Table 3

SYSTEM	ARCHITECTURE	DE⇒EN	EN⇒DE
<i>Existing end-to-end NMT systems</i>			
[Rikters <i>et al.</i> , 2017]	cGRU + dropout + named entity forcing + <i>synthetic data</i>	29.00	22.70
[Escolano <i>et al.</i> , 2017]	Char2Char + rescoring with inverse model + <i>synthetic data</i>	28.10	21.20
[Sennrich <i>et al.</i> , 2017]	cGRU + <i>synthetic data</i>	32.00	26.10
[Tu <i>et al.</i> , 2016]	RNNSEARCH + COVERAGE	28.70	23.60
[Zheng <i>et al.</i> , 2018]	RNNSEARCH + PAST-FUTURE-LAYERS	29.70	24.30
<i>Our end-to-end NMT systems</i>			
this work	RNNSEARCH	29.33	23.83
	+ MEMATT	30.13	24.62
	+ KVMEMATT-R2 + ATTEOSOBJ	30.98	25.39

Table 4: Case-sensitive BLEU scores (%) on German⇔English translation task. “*synthetic data*” denotes additional 10M monolingual sentences, which is not used in this work.

we can see that, compared with the baseline system, our approach can decline the percentages of source words which are under-translated from 13.1% to 9.7% and which are over-translated from 2.7% to 1.3%. The main reason is that our KVMEMATT can keep track of attention status and generate more appropriate source context for predicting the next target word at each decoding step.

### 4.3 Results on German-English

We also evaluate our model on the WMT17 benchmarks on bidirectional German⇔English translation tasks, as listed in Table 4. Our baseline achieves even higher BLEU scores to the state-of-the-art NMT systems of WMT17, which do not use additional synthetic data.<sup>2</sup> Our proposed model consistently outperforms two strong baselines (i.e., standard and memory-augmented attention models) on both De⇒En and En⇒De translation tasks. These results demonstrate that our model works well across different language pairs.

## 5 Related Work

Our work is inspired by the key-value memory networks [Miller *et al.*, 2016] originally proposed for the question answering, and has been successfully applied to machine translation [Gu *et al.*, 2017; Gehring *et al.*, 2017; Vaswani *et al.*, 2017; Tu *et al.*, 2018]. In these works, both the key-memory and value-memory are fixed during translation. Different from these works, we update the KEY-MEMORY along with the update-chain of the decoder state via attentive writing operations (e.g. FORGET and ADD).

Our work is related to recent studies that focus on designing better attention models [Luong *et al.*, 2015; Cohn *et al.*, 2016; Feng *et al.*, 2016; Tu *et al.*, 2016; Mi *et al.*, 2016; Zhang *et al.*, 2017]. [Luong *et al.*, 2015] proposed to use a global attention to attend to all source words and a local attention model to look at a subset of source words. [Cohn *et al.*, 2016] extended the attention-based NMT to include structural biases from word-based alignment models. [Feng *et al.*, 2016] added implicit distortion and fertility models to attention-based NMT. [Zhang *et al.*, 2017] incorporated

<sup>2</sup>[Sennrich *et al.*, 2017] obtains better BLEU scores than our model, since they use large scaled synthetic data (about 10M). It maybe unfair to compare our model to theirs directly.

distortion knowledge into the attention-based NMT. [Tu *et al.*, 2016; Mi *et al.*, 2016] proposed coverage mechanisms to encourage the decoder to consider more untranslated source words during translation. These works are different from our KVMEMATT, since we use a rather generic key-value memory-augmented framework with memory access (i.e. *address, read* and *update*).

Our work is also related to recent efforts on attaching a memory to neural networks [Graves *et al.*, 2014] and exploiting memory [Tang *et al.*, 2016; Wang *et al.*, 2016; Feng *et al.*, 2017; Meng *et al.*, 2016; Wang *et al.*, 2017] during translation. [Tang *et al.*, 2016] exploited a phrase memory stored in symbolic form for NMT. [Wang *et al.*, 2016] extended the NMT decoder by maintaining an external memory, which is operated by reading and writing operations. [Feng *et al.*, 2017] proposed a neural-symbolic architecture, which exploits a memory to provide knowledge for infrequently used words. Our work differ at that we augment attention with a specially designed interactive key-value memory, which allows the model to leverage possibly complex transformations and interactions between the two memories via single- or multi-rounds of memory access in each decoding step.

## 6 Conclusion and Future Work

We propose an effective KVMEMATT model for NMT, which maintains a timely updated key-memory to track attention history and a fixed value-memory to store the representation of source sentence during translation. Via nontrivial transformations and iterative interactions between the two memories, our KVMEMATT can focus on more appropriate source context for predicting the next target word at each decoding step. Additionally, to further enhance the attention, we propose a simple yet effective attention-oriented objective in a weakly supervised manner. Our empirical study on Chinese⇒English, German⇒English and English⇒German translation tasks shows that KVMEMATT can significantly improve the performance of NMT.

For future work, we will consider to explore more rounds of memory access with more powerful operations on key-value memories to further enhance the attention. Another interesting direction is to apply the proposed approach to Transformer [Vaswani *et al.*, 2017], in which the attention model plays a more important role.

## References

- [Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.
- [Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.
- [Cohn *et al.*, 2016] Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. Incorporating structural alignment biases into an attentional neural translation model. In *NAACL*, 2016.
- [Escolano *et al.*, 2017] Carlos Escolano, Marta R. Costajussà, and José A. R. Fonollosa. The talp-upc neural machine translation system for german/finnish-english using the inverse direction model in rescoring. In *WMT*, 2017.
- [Feng *et al.*, 2016] Shi Feng, Shujie Liu, Mu Li, and Ming Zhou. Implicit distortion and fertility models for attention-based encoder-decoder NMT model. In *COLING*, 2016.
- [Feng *et al.*, 2017] Yang Feng, Shiyue Zhang, Andi Zhang, Dong Wang, and Andrew Abel. Memory-augmented neural machine translation. *arXiv*, 2017.
- [Gehring *et al.*, 2017] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv*, 2017.
- [Graves *et al.*, 2014] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing machines. *arXiv*, 2014.
- [Gu *et al.*, 2017] Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. Search engine guided non-parametric neural machine translation. *arXiv*, 2017.
- [Hinton *et al.*, 2012] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv*, 2012.
- [Liu and Sun, 2015] Yang Liu and Maosong Sun. Contrastive unsupervised word alignment with non-local features. In *AAAI*, 2015.
- [Luong *et al.*, 2015] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.
- [Meng *et al.*, 2016] Fandong Meng, Zhengdong Lu, Hang Li, and Qun Liu. Interactive attention for neural machine translation. In *COLING*, 2016.
- [Mi *et al.*, 2016] Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. Coverage embedding models for neural machine translation. In *EMNLP*, 2016.
- [Miller *et al.*, 2016] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *EMNLP*, 2016.
- [Och and Ney, 2003] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *CL*, 2003.
- [Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.
- [Riktors *et al.*, 2017] Matīss Riktors, Chantal Amrhein, Maksym Del, and Mark Fishel. C-3ma: Tartu-riga-zurich translation systems for wmt17. In *WMT*, 2017.
- [Rocktäschel *et al.*, 2017] Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. Frustratingly short attention spans in neural language modeling. In *ICLR*, 2017.
- [Schuster and Paliwal, 1997] Mike Schuster and Kuldeep K Paliwal. Bidirectional recurrent neural networks. *TSP*, 1997.
- [Sennrich *et al.*, 2016] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL*, 2016.
- [Sennrich *et al.*, 2017] Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. The university of edinburgh’s neural MT systems for WMT17. *arXiv*, 2017.
- [Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.
- [Tang *et al.*, 2016] Yaohua Tang, Fandong Meng, Zhengdong Lu, Hang Li, and Philip L. H. Yu. Neural machine translation with external phrase memory. *arXiv*, 2016.
- [Tu *et al.*, 2016] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *ACL*, 2016.
- [Tu *et al.*, 2018] Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. Learning to remember translation history with a continuous cache. *TACL*, 2018.
- [Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.
- [Wang *et al.*, 2016] Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. Memory-enhanced decoder for neural machine translation. In *EMNLP*, 2016.
- [Wang *et al.*, 2017] Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. Neural machine translation advised by statistical machine translation. In *AAAI*, 2017.
- [Zeiler, 2012] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv*, 2012.
- [Zhang *et al.*, 2017] Jinchao Zhang, Mingxuan Wang, Qun Liu, and Jie Zhou. Incorporating word reordering knowledge into attention-based neural machine translation. In *ACL*, 2017.
- [Zheng *et al.*, 2018] Zaixiang Zheng, Hao Zhou, Shujian Huang, Lili Mou, Dai Xinyu, Jiajun Chen, and Zhaopeng Tu. Modeling past and future for neural machine translation. *TACL*, 2018.