Memory-enhanced Decoder for Neural Machine Translation

Mingxuan Wang[1] **Zhengdong Lu**[2] **Hang Li**[2] **Qun Liu**[3,1]
[1]Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
{wangmingxuan,liuqun}@ict.ac.cn
[2]Noah's Ark Lab, Huawei Technologies
{Lu.Zhengdong,HangLi.HL}@huawei.com
[3]ADAPT Centre, School of Computing, Dublin City University

Abstract

We propose to enhance the RNN decoder in a neural machine translator (NMT) with external memory, as a natural but powerful extension to the state in the decoding RNN. This memory-enhanced RNN decoder is called MEMDEC. At each time during decoding, MEMDEC will read from this memory and write to this memory once, both with content-based addressing. Unlike the unbounded memory in previous work(Bahdanau et al., 2014) to store the representation of source sentence, the memory in MEMDEC is a matrix with pre-determined size designed to better capture the information important for the decoding process at each time step. Our empirical study on Chinese-English translation shows that it can improve by 4.8 BLEU upon Groundhog and 5.3 BLEU upon on Moses, yielding the best performance achieved with the same training set.

1 Introduction

The introduction of external memory has greatly expanded the representational capability of neural network-based model on modeling sequences(Graves et al., 2014), by providing flexible ways of storing and accessing information. More specifically, in neural machine translation, one great improvement came from using an array of vectors to represent the source in a sentence-level memory and dynamically accessing relevant segments of them (alignment) (Bahdanau et al., 2014) through content-based addressing (Graves et al., 2014). The success of RNNsearch demonstrated the advantage of saving the entire sentence of arbitrary length in an unbounded memory for operations of next stage (e.g., decoding).

In this paper, we show that an external memory can be used to facilitate the decoding/generation process thorough a memory-enhanced RNN decoder, called MEMDEC. The memory in MEMDEC is a direct extension to the state in the decoding, therefore functionally closer to the memory cell in LSTM(Hochreiter and Schmidhuber, 1997). It takes the form of a matrix with pre-determined size, each column ("a memory cell") can be accessed by the decoding RNN with content-based addressing for both reading and writing during the decoding process. This memory is designed to provide a more flexible way to select, represent and synthesize the information of source sentence and previously generated words of target relevant to the decoding. This is in contrast to the set of hidden states of the entire source sentence (which can viewed as another form of memory) in (Bahdanau et al., 2014) for attentive read, but can be combined with it to greatly improve the performance of neural machine translator. We apply our model on English-Chinese translation tasks, achieving performance superior to any published results, SMT or NMT, on the same training data (Xie et al., 2011; Meng et al., 2015; Tu et al., 2016; Hu et al., 2015)

Our contributions are mainly two-folds

- we propose a memory-enhanced decoder for

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 278–286,
Austin, Texas, November 1-5, 2016.

neural machine translator which naturally extends the RNN with vector state.

- our empirical study on Chinese-English translation tasks show the efficacy of the proposed model.

Roadmap In the remainder of this paper, we will first give a brief introduction to attention-based neural machine translation in Section 2, presented from the view of encoder-decoder, which treats the hidden states of source as an unbounded memory and the attention model as a content-based reading. In Section 3, we will elaborate on the memory-enhanced decoder MEMDEC. In Section 4, we will apply NMT with MEMDEC to a Chinese-English task. Then in Section 5 and 6, we will give related work and conclude the paper.

2 Neural machine translation with attention

Our work is built on attention-based NMT(Bahdanau et al., 2014), which represents the source sentence as a sequence of vectors after being processed by RNN or bi-directional RNNs, and then conducts dynamic alignment and generation of the target sentence with another RNN simultaneously.

Attention-based NMT, with RNNsearch as its most popular representative, generalizes the conventional notion of encoder-decoder in using a unbounded memory for the intermediate representation of source sentence and content-based addressing read in decoding, as illustrated in Figure 1. More specifically, at time step t, RNNsearch first get context vector $\mathbf{c}_t$ after reading from the source representation $\mathbf{M}^{\mathrm{s}}$, which is then used to update the state, and generate the word y_t (along with the current hidden state $\mathbf{s}_t$, and the previously generated word y_{i-1}).

Formally, given an input sequence $\mathbf{x} = [x_1, x_2, \ldots, x_{T_x}]$ and the previously generated sequence $\mathbf{y}_{<t} = [y_1, y_2, \ldots, y_{t-1}]$, the probability of next word y_t is

$$p(y_t|\mathbf{y}_{<t}; \mathbf{x}) = f(\mathbf{c}_t, y_{t-1}, \mathbf{s}_t), \qquad (1)$$

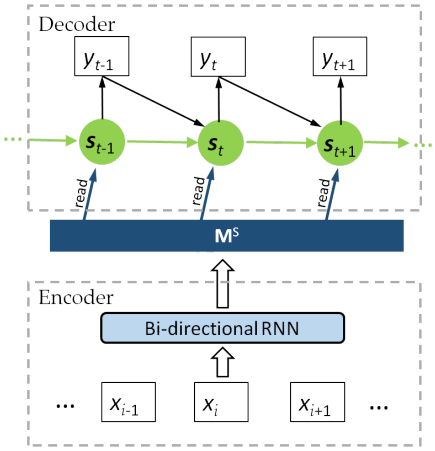

Figure 1: RNNsearch in the encoder-decoder view.

where $\mathbf{s}_t$ is state of decoder RNN at time step t calculated as

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

where $g(\cdot)$ can be an be any activation function, here we adopt a more sophisticated dynamic operator as in Gated Recurrent Unit (GRU, (Cho et al., 2014)). In the remainder of the paper, we will also use GRU to stand for the operator. The reading $\mathbf{c}_t$ is calculated as

$$\mathbf{c}_t = \sum_{j=1}^{j=T_x} \alpha_{t,j}\mathbf{h}_j, \qquad (3)$$

where $\mathbf{h}_j$ is the j^{th} cell in memory $\mathbf{M}^{\mathrm{s}}$. More formally, $\mathbf{h}_j = [\overleftarrow{\mathbf{h}_j}^\top, \overrightarrow{\mathbf{h}_j}^\top]^\top$ is the annotations of x_j and contains information about the whole input sequence with a strong focus on the parts surrounding x_j, which is computed by a bidirectional RNN. The weight $\alpha_{t,j}$ is computed by

$$\alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_{k=1}^{k=T_x} \exp(e_{t,k})}.$$

where $e_{i,j} = \mathbf{v}_a^T \tanh(\mathbf{W}_a\mathbf{s}_{t-1} + \mathbf{U}_a\mathbf{h}_j)$ scores how well $\mathbf{s}_{t-1}$ and the memory cell $\mathbf{h}_j$ match. This is called automatic alignment (Bahdanau et al., 2014) or attention model (Luong et al., 2015), but it is essentially reading with content-based addressing defined in (Graves et al., 2014). With this addressing strategy the decoder can attend to the source representation that is most relevant to the stage of decoding.

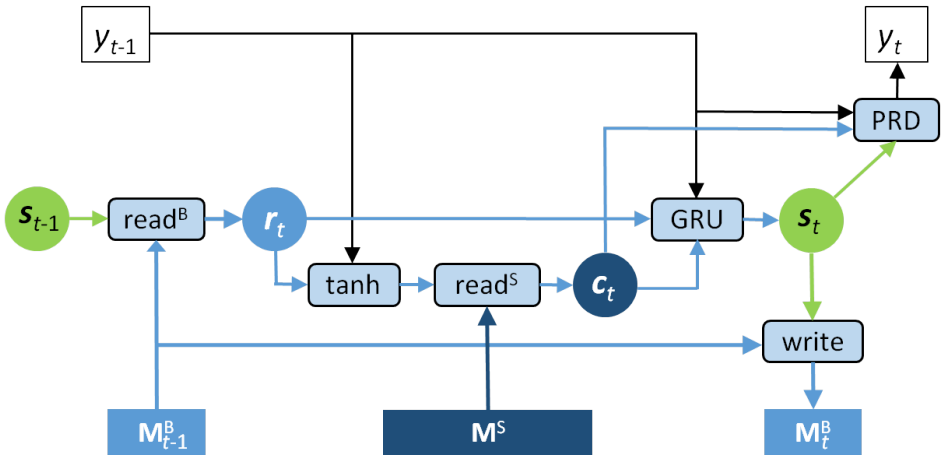

Figure 2: Diagram of the proposed decoder MEMDEC with details.

2.1 Improved Attention Model

The alignment model $\alpha_{t,j}$ scores how well the output at position t matches the inputs around position j based on $\mathbf{s}_{t-1}$ and h_j. It is intuitively beneficial to exploit the information of y_{t-1} when reading from $\mathbf{M}^S$, which is missing from the implementation of attention-based NMT in (Bahdanau et al., 2014). In this work, we build a more effective alignment path by feeding both previous hidden state $\mathbf{s}_{t-1}$ and the context word y_{t-1} to the attention model, inspired by the recent implementation of attention-based NMT[1]. Formally, the calculation of $e_{t,j}$ becomes

$$e_{t,j} = \mathbf{v}_a^T \tanh(\mathbf{W}_a \tilde{\mathbf{s}}_{t-1} + \mathbf{U}_a \mathbf{h}_j),$$

where

- $\tilde{\mathbf{s}}_{t-1} = \mathcal{H}(\mathbf{s}_{t-1}, \mathbf{e}_{y_{t-1}})$ is an intermediate state tailored for reading from $\mathbf{M}^S$ with the information of y_{t-1} (its word embedding being $\mathbf{e}_{y_{t-1}}$) added;

- $\mathcal{H}$ is a nonlinear function, which can be as simple as tanh or as complex as GRU. In our preliminary experiments, we found GRU works slightly better than tanh function, but we chose the latter for simplicity.

[1] github.com/nyu-dl/dl4mt-tutorial/tree/master/session2

3 Decoder with External Memory

In this section we will elaborate on the proposed memory-enhanced decoder MEMDEC. In addition to the source memory $\mathbf{M}^S$, MEMDEC is equipped with a buffer memory $\mathbf{M}^B$ as an extension to the conventional state vector. Figure 3 contrasts MEMDEC with the decoder in RNNsearch (Figure 1) on a high level.

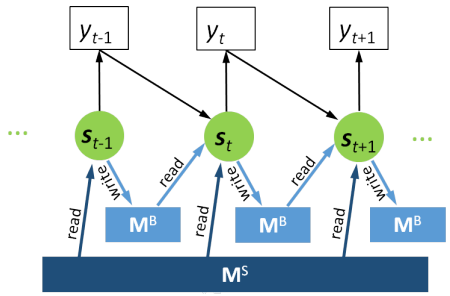

Figure 3: High level digram of MEMDEC.

In the remainder of the paper, we will refer to the conventional state as vector-state (denoted $\mathbf{s}_t$) and its memory extension as memory-state (denoted as $\mathbf{M}^B_t$). Both states are updated at each time step in a interweaving fashion, while the output symbol y_t is predicted based solely on vector-state $\mathbf{s}_t$ (along with $\mathbf{c}_t$ and y_{t-1}). The diagram of this memory-enhanced decoder is given in Figure 2.

Vector-State Update At time t, the vector-state $\mathbf{s}_t$ is first used to read $\mathbf{M}^{\text{B}}$

$$\mathbf{r}_{t-1} = \text{read}^{\text{B}}(\mathbf{s}_{t-1}, \mathbf{M}^{\text{B}}_{t-1}) \quad (4)$$

which then meets the previous prediction y_{t-1} to form an "intermediate" state-vector

$$\widetilde{\mathbf{s}}_t = \tanh(\mathbf{W}^r \mathbf{r}_{t-1} + \mathbf{W}^y \mathbf{e}_{y_{t-1}}). \quad (5)$$

where $\mathbf{e}_{y_{t-1}}$ is the word-embedding associated with the previous prediction y_{t-1}. This pre-state $\widetilde{\mathbf{s}}_t$ is used to read the source memory $\mathbf{M}^{\text{S}}$

$$\mathbf{c}_t = \text{read}^{\text{S}}(\widetilde{\mathbf{s}}_t, \mathbf{M}^{\text{S}}). \quad (6)$$

Both readings in Eq. (4) & (6) follow content-based addressing(Graves et al., 2014) (details later in Section 3.1). After that, $\mathbf{r}_{t-1}$ is combined with output symbol y_{t-1} and $\mathbf{c}_t$ to update the new vector-state

$$\mathbf{s}_t = \text{GRU}(\mathbf{r}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c}_t) \quad (7)$$

The update of vector-state is illustrated in Figure 4.

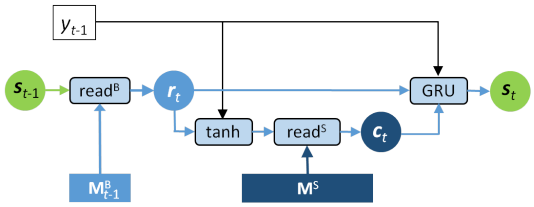

Figure 4: Vector-state update at time t.

Memory-State Update As illustrated in Figure 5, the update for memory-state is simple after the update of vector-state: with the vector-state $\mathbf{s}_{t+1}$ the updated memory-state will be

$$\mathbf{M}^{\text{B}}_t = \text{write}(\mathbf{s}_t, \mathbf{M}^{\text{B}}_{t-1}) \quad (8)$$

The writing to the memory-state is also content-based, with same forgetting mechanism suggested in (Graves et al., 2014), which we will elaborate with more details later in this section.

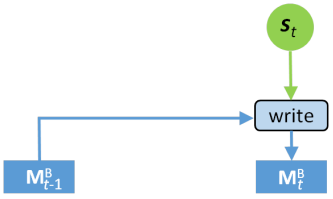

Figure 5: Memory-state update at time t.

Prediction As illustrated in Figure 6, the prediction model is same as in (Bahdanau et al., 2014), where the score for word y is given by

$$\text{score}(y) = \text{DNN}([\mathbf{s}_t, \mathbf{c}_t, \mathbf{e}_{y_{t-1}}])^\top \omega_y \quad (9)$$

where ω_y is the parameters associated with the word y. The probability of generating word y at time t is then given by a softmax over the scores

$$p(y|\mathbf{s}_t, \mathbf{c}_t, y_{t-1}) = \frac{\exp(\text{score}(y))}{\sum_{y'} \exp(\text{score}(y'))}.$$

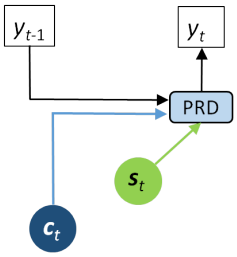

Figure 6: Prediction at time t.

3.1 Reading Memory-State

Formally $\mathbf{M}^{\text{B}}_{t'} \in \mathbb{R}^{n \times m}$ is the memory-state at time t' after the memory-state update, where n is the number of memory cells and m is the dimension of vector in each cell. Before the vector-state update at time t, the output of reading $\mathbf{r}_t$ is given by

$$\mathbf{r}_t = \sum_{j=1}^{j=n} \mathbf{w}^{\text{R}}_t(j) \mathbf{M}^{\text{B}}_{t-1}(j)$$

where $\mathbf{w}^{\text{R}}_t \in \mathbb{R}^n$ specifies the *normalized* weights assigned to the cells in $\mathbf{M}^{\text{B}}_t$. Similar with the reading from $\mathbf{M}^{\text{S}}$ (a.k.a. attention model), we use content-based addressing in determining $\mathbf{w}^{\text{R}}_t$.

More specifically, $\mathbf{w}_t^{\text{R}}$ is also updated from the one from previous time $\mathbf{w}_{t-1}^{\text{R}}$ as

$$\mathbf{w}_t^{\text{R}} = g_t^{\text{R}} \mathbf{w}_{t-1}^{\text{R}} + (1 - g_t^{\text{R}}) \widetilde{\mathbf{w}}_t^{\text{R}}, \qquad (10)$$

where

- $g_t^{\text{R}} = \sigma(\mathbf{w}_g^{\text{R}} \mathbf{s}_t)$ is the gate function, with parameters $\mathbf{w}_g^{\text{R}} \in \mathbb{R}^m$;
- $\widetilde{\mathbf{w}}_t$ gives the contribution based on the current vector-state $\mathbf{s}_t$

$$\widetilde{\mathbf{w}}_t^{\text{R}} = \text{softmax}(\mathbf{a}_t^{\text{R}}) \qquad (11)$$

$$\mathbf{a}_t^{\text{R}}(i) = \mathbf{v}^\top (\mathbf{W}_a^{\text{R}} \mathbf{M}_{t-1}^{\text{B}}(i) + \mathbf{U}_a^{\text{R}} \mathbf{s}_{t-1}), \qquad (12)$$

with parameters $\mathbf{W}_a^{\text{R}}, \mathbf{U}_a^{\text{R}} \in \mathbb{R}^{m \times m}$ and $\mathbf{v} \in \mathbb{R}^m$.

3.2 Writing to Memory-State

There are two types of operation on writing to memory-state: ERASE and ADD. Erasion is similar to the forget gate in LSTM or GRU, which determines the content to be remove from memory cells. More specifically, the vector $\mu_t^{\text{ERS}} \in \mathbb{R}^m$ specifies the values to be removed on each dimension in memory cells, which is than assigned to each cell through normalized weights $\mathbf{w}_t^{\text{W}}$. Formally, the memory-state after ERASE is given by

$$\widetilde{\mathbf{M}}_t^{\text{B}}(i) = \mathbf{M}_{t-1}^{\text{B}}(i)(1 - \mathbf{w}_t^{\text{W}}(i) \cdot \mu_t^{\text{ERS}}) \qquad (13)$$

$$i = 1, \cdots, n$$

where

- $\mu_t^{\text{ERS}} = \sigma(\mathbf{W}^{\text{ERS}} \mathbf{s}_t)$ is parametrized with $\mathbf{W}^{\text{ERS}} \in \mathbb{R}^{m \times m}$;
- $\mathbf{w}_t^{\text{W}}(i)$ specifies the weight associated with the i^{th} cell in the same parametric form as in Eq. (10)-(12) with generally different parameters.

ADD operation is similar with the update gate in LSTM or GRU, deciding how much current information should be written to the memory.

$$\mathbf{M}_t^{\text{B}}(i) = \widetilde{\mathbf{M}}_t^{\text{B}}(i) + \mathbf{w}_t^{\text{W}}(i) \mu_t^{\text{ADD}}$$

$$\mu_t^{\text{ADD}} = \sigma(\mathbf{W}^{\text{ADD}} \mathbf{s}_t)$$

where $\mu_t^{\text{ADD}} \in \mathbb{R}^m$ and $\mathbf{W}^{\text{ADD}} \in \mathbb{R}^{m \times m}$.

In our experiments, we have a peculiar but interesting observation: it is often beneficial to use the same weights for both reading (i.e., $\mathbf{w}_t^{\text{R}}$ in Section 3.1) and writing (i.e., $\mathbf{w}_t^{\text{W}}$ in Section 3.2) for the same vector-state $\mathbf{s}_t$. We conjecture that this acts like a regularization mechanism to encourage the content of reading and writing to be similar to each other.

3.3 Some Analysis

The writing operation in Eq. (13) at time t can be viewed as an nonlinear way to combine the previous memory-state $\mathbf{M}_{t-1}^{\text{B}}$ and the newly updated vector-state $\mathbf{s}_t$, where the nonlinearity comes from both the content-based addressing and the gating. This is in a way similar to the update of states in regular RNN, while we conjecture that the addressing strategy in MEMDEC makes it easier to selectively change some content updated (e.g., the relatively short-term content) while keeping other content less modified (e.g., the relatively long-term content).

The reading operation in Eq. (10) can "extract" the content from $\mathbf{M}_t^{\text{B}}$ relevant to the alignment (reading from $\mathbf{M}^{\text{S}}$) and prediction task at time t. This is in contrast with the regular RNN decoder including its gated variants, which takes the entire state vector to for this purpose. As one advantage, although only part of the information in $\mathbf{M}_t^{\text{B}}$ is used at t, the entire memory-state, which may store other information useful for later, will be carry over to time $t+1$ for memory-state update (writing).

4 Experiments on Chinese-English Translation

We test the memory-enhanced decoder to task of Chinese-to-English translation, where MEMDEC is put on the top of encoder same as in (Bahdanau et al., 2014).

4.1 Datasets and Evaluation metrics

Our training data for the translation task consists of 1.25M sentence pairs extracted from LDC corpora[2], with 27.9M Chinese words and 34.5M

[2]The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07,

English words respectively. We choose NIST 2002 (MT02) dataset as our development set, and the NIST 2003 (MT03), 2004 (MT04) 2005 (MT05) and 2006 (MT06) datasets as our test sets. We use the case-insensitive 4-gram NIST BLEU score as our evaluation metric as our evaluation metric (Papineni et al., 2002).

4.2 Experiment settings

Hyper parameters In training of the neural networks, we limit the source and target vocabularies to the most frequent 30K words in both Chinese and English, covering approximately 97.7% and 99.3% of the two corpora respectively. The dimensions of word embedding is 512 and the size of the hidden layer is 1024. The dimemsion of each cell in $\mathbf{M}^{\text{B}}$ is set to 1024 and the number of cells n is set to 8.

Training details We initialize the recurrent weight matrices as random orthogonal matrices. All the bias vectors were initialize to zero. For other parameters, we initialize them by sampling each element from the Gaussian distribution of mean 0 and variance 0.01^2. Parameter optimization is performed using stochastic gradient descent. Adadelta (Zeiler, 2012) is used to automatically adapt the learning rate of each parameter ($\epsilon = 10^{-6}$ and $\rho = 0.95$). To avoid gradients explosion, the gradients of the cost function which had ℓ_2 norm larger than a predefined threshold 1.0 was normalized to the threshold (Pascanu et al., 2013). Each SGD is of a minibatch of 80 sentences. We train our NMT model with the sentences of length up to 50 words in training data, while for moses system we use the full training data.

Memory Initialization Each memory cell is initialized with the source sentence hidden state computed as

$$\mathbf{M}^{\text{B}}(i) = \mathbf{m} + \nu_i \quad (14)$$

$$\mathbf{m} = \sigma(\mathbf{W}_{\text{INI}} \sum_{i=0}^{i=T_x} \mathbf{h}_i)/T_x \quad (15)$$

where $\mathbf{W}_{\text{INI}} \in \mathbb{R}^{m \times 2 \cdot m}$; σ is $\tanh$ function. $\mathbf{m}$ makes a nonlinear transformation of the source sentence information. ν_i is a random vector sampled from $\mathcal{N}(0, 0.1)$.

Dropout we also use dropout for our NMT baseline model and MEMDEC to avoid overfitting (Hinton et al., 2012). The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. In the simplest case, each unit is omitted with a fixed probability p, namely dropout rate. In our experiments, dropout was applied only on the output layer and the dropout rate is set to 0.5. We also try other strategy such as dropout at word embeddings or RNN hidden states but fail to get further improvements.

Pre-training For MEMDEC, the objective function is a highly non-convex function of the parameters with more complicated landscape than that for decoder without external memory, rendering direct optimization over all the parameters rather difficult. Inspired by the effort on easing the training of very deep architectures (Hinton and Salakhutdinov, 2006), we propose a simple pre-training strategyFirst we train a regular attention-based NMT model without external memory. Then we use the trained NMT model to initialize the parameters of encoder and parameters of MEMDEC, except those related to memory-state (i.e., $\{\mathbf{W}^{\text{R}}_a, \mathbf{U}^{\text{R}}_a, \mathbf{v}, \mathbf{w}^{\text{R}}_g, \mathbf{W}^{\text{ERS}}, \mathbf{W}^{\text{ADD}}\}$). After that, we fine-tune all the parameters of NMT with MEMDEC decoder, including the parameters initialized with pre-training and those associated with accessing memory-state.

4.3 Comparison systems

We compare our method with three state-of-the-art systems:

- **Moses:** an open source phrase-based translation system [3]: with default configuration and a 4-gram language model trained on the target portion of training data.

LDC2004T08 and LDC2005T06.

[3] http://www.statmt.org/moses/

SYSTEM	MT03	MT04	MT05	MT06	AVE.
Groundhog	31.92	34.09	31.56	31.12	32.17
RNNsearch⋆	33.11	37.11	33.04	32.99	34.06
RNNsearch⋆ + coverage	34.49	38.34	34.91	34.25	35.49
MEMDEC	**36.16**	**39.81**	**35.91**	**35.98**	**36.95**
Moses	31.61	33.48	30.75	30.85	31.67

Table 1: Case-insensitive BLEU scores on Chinese-English translation. Moses is the state-of-the-art phrase-based statistical machine translation system. For RNNsearch, we use the open source system Groundhog as our baseline. The strong baseline, denoted RNNsearch⋆, also adopts *feedback attention* and *dropout*. The *coverage* model on top of RNNsearch⋆ has significantly improved upon its published version (Tu et al., 2016), which achieves the best published result on this training set. For MEMDEC the number of cells is set to 8.

pre-training	n	MT03	MT04	MT05	MT06	Ave.
N	4	35.29	37.36	34.58	33.32	35.11
Y	4	35.39	39.16	35.33	35.02	36.22
Y	6	35.63	39.29	35.61	34.92	36.58
Y	8	36.16	39.81	35.91	35.98	36.95
Y	10	36.46	38.86	34.46	35.00	36.19
Y	12	35.92	39.09	35.31	35.12	36.37

Table 2: MEMDEC performances of different memory size.

- **RNNSearch:** an attention-based NMT model with default settings. We use the open source system GroundHog as our NMT baseline[4].
- **Coverage model**: a state-of-the-art variant of attention-based NMT model (Tu et al., 2016) which improves the attention mechanism through modelling a soft coverage on the source representation.

4.4 Results

The main results of different models are given in Table 1. Clearly MEMDEC leads to remarkable improvement over Moses (+5.28 BLEU) and Groundhog (+4.78 BLEU). The *feedback attention* gains +1.06 BLEU score on top of Groundhog on average, while together with *dropout* adds another +0.83 BLEU score, which constitute the 1.89 BLEU gain of RNNsearch⋆ over Groundhog. Compared to RNNsearch⋆ MEMDEC is +2.89 BLEU score higher, showing the modeling power gained from the external memory. Finally, we also compare MEMDEC with the state-of-the-art attention-based NMT with COVERAGE mechanism(Tu et al., 2016), which is about 2 BLEU over than the published result after adding fast attention and dropout. In this comparison MEMDEC wins with big margin (+1.46 BLEU score).

[4] https://github.com/lisa-groundhog/GroundHog

4.5 Model selection

Pre-training plays an important role in optimizing the memory model. As can be seen in Tab.2, pre-training improves upon our baseline +1.11 BLEU score on average, but even without pre-training our model still gains +1.04 BLEU score on average. Our model is rather robust to the memory size: with merely four cells, our model will be over 2 BLEU higher than RNNsearch⋆. This further verifies our conjecture the the external memory is mostly used to store part of the source and history of target sentence.

4.6 Case study

We show in Table 5 sample translations from Chinese to English, comparing mainly MEMDEC

src	恩达依兹耶说:“签署(2003年11月停火)协定的各方,最迟必须在元月五日以前把战士的驻扎地点安顿完毕 。”
ref	“All *parties* that signed the *(November 2003 ceasefire)* accord should finish the cantoning of their fighters by January 5, 2004, at the latest,” Ndayizeye said.
MEMDEC	UNK said, “ the *parties involved in the ceasefire agreement on November 2003* will have to be completed by January 5, 2004. ”
base	“The signing of the **agreement (UNK-fire) agreement in the November 2003 ceasefire** must be completed by January 5, 2004.
src	代表团成员告诉今日美国报说,布希政府已批准美国代表团预定元月六日至十日展开的北韩之行。
ref	Members of the delegation told *US Today* that the Bush administration had *approved the US delegation' s visit* to North Korea from January 6 to 10.
MEMDEC	The delegation told the *US today* that the Bush administration has *approved the US delegation's visit* to north Korea from 6 to 10 january .
base	The delegation told the **US** that the Bush administration has **approved the US to begin his visit** to north Korea from 6 to 10 January.

Table 3: Sample translations-for each example, we show the source(src), the human translation (ref),the translation from our memory model MEMDEC and the translation from RNNsearch(equipped with fast attention and dropout).We italicise some *correct* translation segments and highlight a few **wrong** ones in bold.

and the RNNsearch model for its pre-training. It is appealing to observe that MEMDEC can produce more fluent translation results and better grasp the semantic information of the sentence.

5 Related Work

There is a long thread of work aiming to improve the ability of RNN in remembering long sequences, with the long short-term memory RNN (LSTM) (Hochreiter and Schmidhuber, 1997) being the most salient examples and GRU (Cho et al., 2014) being the most recent one. Those works focus on designing the dynamics of the RNN through new dynamic operators and appropriate gating, while still keeping vector form RNN states. MEMDEC, on top of the gated RNN, explicitly adds matrix-form memory equipped with content-based addressing to the system, hence greatly improving the power of the decoder RNN in representing the information important for the translation task.

MEMDEC is obviously related to the recent effort on attaching an external memory to neural networks, with two most salient examples being Neural Turing Machine (NTM) (Graves et al., 2014) and Memory Network (Weston et al., 2014). In fact MEMDEC can be viewed as a special case of NTM, with specifically designed reading (from two different types of memory) and writing mechanism for the translation task. Quite remarkably MEMDEC is among the rare instances of NTM which significantly improves upon state-of-the-arts on a real-world NLP task with large training corpus.

Our work is also related to the recent work on machine reading (Cheng et al., 2016), in which the machine reader is equipped with a memory tape, enabling the model to directly read all the previous hidden state with an attention mechanism. Different from their work, we use an external bounded memory and make an abstraction of previous information. In (Meng et al., 2015), Meng et. al. also proposed a deep architecture for sequence-to-sequence learning with stacked layers of memory to store the intermediate representations, while our external memory was applied within a sequence.

6 Conclusion

We propose to enhance the RNN decoder in a neural machine translator (NMT) with external memory. Our empirical study on Chinese-English translation shows that it can significantly improve the performance of NMT.

References

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Cheng et al.2016] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

[Cho et al.2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Graves et al.2014] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

[Hinton and Salakhutdinov2006] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[Hinton et al.2012] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hu et al.2015] Baotian Hu, Zhaopeng Tu, Zhengdong Lu, and Hang Li. 2015. Context-dependent translation selection using convolutional neural network.

[Luong et al.2015] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[Meng et al.2015] Fandong Meng, Zhengdong Lu, Zhaopeng Tu, Hang Li, and Qun Liu. 2015. A deep memory-based architecture for sequence-to-sequence learning. *arXiv preprint arXiv:1506.06442*.

[Papineni et al.2002] 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. Association for Computational Linguistics.

[Pascanu et al.2013] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

[Tu et al.2016] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *ArXiv eprints, January*.

[Weston et al.2014] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

[Xie et al.2011] Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation.

[Zeiler2012] Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.