Learning to Remember Translation History with a Continuous Cache

Zhaopeng Tu
Tencent AI Lab
zptu@tencent.com

Yang Liu
Tsinghua University
liuyang2011@tsinghua.edu.cn

Shuming Shi
Tencent AI Lab
shumingshi@tencent.com

Tong Zhang
Tencent AI Lab
bradymzhang@tencent.com

Abstract

Existing neural machine translation (NMT) models generally translate sentences in isolation, missing the opportunity to take advantage of document-level information. In this work, we propose to augment NMT models with a very light-weight cache-like memory network, which stores recent hidden representations as translation history. The probability distribution over generated words is updated online depending on the translation history retrieved from the memory, endowing NMT models with the capability to dynamically adapt over time. Experiments on multiple domains with different topics and styles show the effectiveness of the proposed approach with negligible impact on the computational cost.

1 Introduction

Neural machine translation (NMT) has advanced the state of the art in recent years (Kalchbrenner et al., 2014; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). However, existing models generally treat documents as a list of independent sentence pairs and ignore cross-sentence information, which leads to translation inconsistency and ambiguity arising from a single source sentence.

There have been few recent attempts to model cross-sentence context for NMT: Wang et al. (2017a) use a hierarchical RNN to summarize the previous K source sentences, while Jean et al. (2017) use an additional set of an encoder and attention model to dynamically select part of the previous source sentence. While these approaches have proven their ability to represent cross-sentence context, they generate the context from discrete lexicons, thus would cause errors propagated from generated translations. Accordingly, they only take into account source sentences but fail to make use of target-side information.[1] Another potential limitation is that they are computationally expensive, which limits the scale of cross-sentence context.

In this work, we propose a very light-weight alternative that can both cover large-scale cross-sentence context as well as exploit bilingual translation history. Our work is inspired by recent successes of memory-augmented neural networks on multiple NLP tasks (Weston et al., 2015; Sukhbaatar et al., 2015; Miller et al., 2016; Gu et al., 2018), especially the efficient cache-like memory networks for language modeling (Grave et al., 2017; Daniluk et al., 2017). Specifically, the proposed approach augments NMT models with *a continuous cache* (CACHE), which stores recent hidden representations as history context. By minimizing the computation burden of the cache-like memory, we are able to use larger memory and scale to longer translation history. Since we leverage internal representations instead of output words, our approach is more robust to the error propagation problem, and thus can incorporate useful target-side context.

Experimental results show that the proposed approach significantly and consistently improves translation performance over a strong NMT baseline on multiple domains with different topics and styles. We found the introduced cache is able to remember translation patterns at different levels of matching and granularity, ranging from exactly matched lexi-

[1] Wang et al. (2017a) indicate that "considering target-side history inversely harms translation performance, since it suffers from serious error propagation problems."

Transactions of the Association for Computational Linguistics, vol. 6, pp. 407–420, 2018. Action Editor: Philipp Koehn.
Submission batch: 9/2017; Revision batch: 11/2017; Published 7/2018.

cal patterns to fuzzily matched patterns, from word-level patterns to phrase-level patterns.

2 Neural Machine Translation

Suppose that $\mathbf{x} = x_1, \ldots x_j, \ldots x_J$ represents a source sentence and $\mathbf{y} = y_1, \ldots y_t, \ldots y_T$ a target sentence. NMT directly models the probability of translation from the source sentence to the target sentence word by word:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} P(y_t|y_{<t}, \mathbf{x}). \tag{1}$$

As shown in Figure 2 (a), the probability of generating the t-th word y_t is computed by:

$$P(y_t|y_{<t}, \mathbf{x}) = g(y_{t-1}, \mathbf{s}_t, \mathbf{c}_t) \tag{2}$$

where $g(\cdot)$ first linearly transforms its input and then applies a softmax function, y_{t-1} is the previously generated word, $\mathbf{s}_t$ is the t-th decoding hidden state, and $\mathbf{c}_t$ is the t-th source representation. The decoder state $\mathbf{s}_t$ is computed as follows:

$$\mathbf{s}_t = f(y_{t-1}, \mathbf{s}_{t-1}, \mathbf{c}_t) \tag{3}$$

where $f(\cdot)$ is an activation function, which is implemented as GRU (Cho et al., 2014) in this work. $\mathbf{c}_t$ is a dynamic vector that selectively summarizes certain parts of the source sentence at each decoding step:

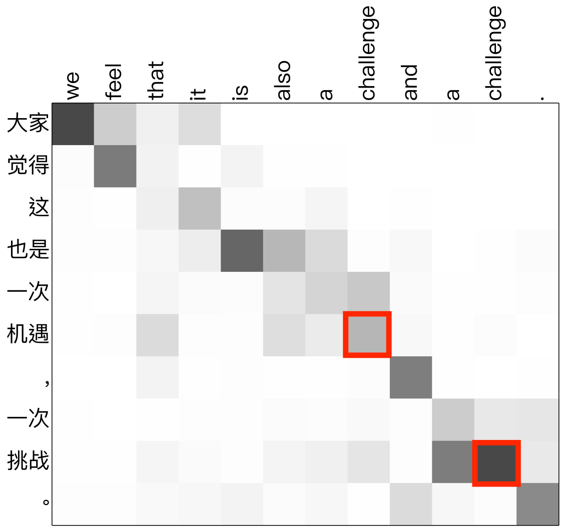
$$\mathbf{c}_t = \sum_{j=1}^{J} \alpha_{t,j} \mathbf{h}_j \tag{4}$$

where $\alpha_{t,j}$ is the alignment probability calculated by an attention model (Bahdanau et al., 2015; Luong et al., 2015a), and $\mathbf{h}_j$ is the encoder hidden state of the j-th source word x_j.

Since the continuous representation of a symbol (*e.g.,* $\mathbf{h}_j$ and $\mathbf{s}_t$) encodes multiple meanings of a word, NMT models need to spend a substantial amount of their capacity in disambiguating source and target words based on the context defined by a source sentence (Choi et al., 2016). Consistency is another critical issue in document-level translation, where a repeated term should keep the same translation throughout the whole document (Xiao et al., 2011). Nevertheless, current NMT models still process a document by translating each sentence alone, suffering from inconsistency and ambiguity arising from a single source sentence, as shown in Table 1. These problems can be alleviated by the proposed approach via modeling translation history, as described below.

Src	…开始 都 **觉得** …大家 **觉得** 这 也是 一次 **机遇** ， 一次 挑战 。
Ref	…initially they all **felt** that …everyone **felt** that this was also an **opportunity** and a challenge .
NMT	…**felt** that …we **feel** that it is also a **challenge** and a challenge .

(a) The translation of “机遇” (“*opportunity*”) suffers from the ambiguity problem, while the translation of “觉得” (“*feel*”) suffers from a tense inconsistency problem. The former problem is ***not*** caused by attending to wrong source words, as shown below.



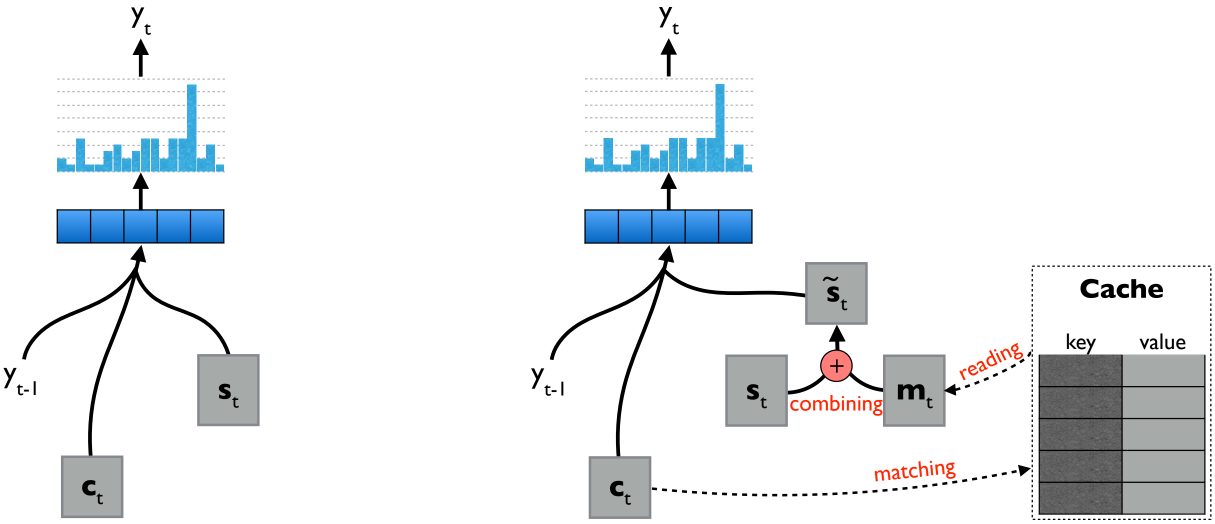
(b) Attention matrix.

Figure 1: An example translation.

3 Approach

3.1 Architecture

The proposed approach augments neural machine translation models with a cache-like memory, which has proven useful for capturing a longer history for the language modeling task (Grave et al., 2017; Daniluk et al., 2017). The cache-like memory is essentially a key-value memory (Miller et al., 2016), which is an array of slots in the form of (*key*,



(a) Standard NMT (b) NMT augmented with a continuous cache

Figure 2: Architectures of (a) standard NMT, and (b) NMT augmented with an external cache to exploit translation history. At each decoding step, the current attention context c_t that represents source-side content serves as a query to retrieve the cache (*key matching*) and an output vector $\mathbf{m}_t$ that represents target-side information in the past translations is returned (*value reading*), which is combined with the current decoder state $\mathbf{s}_t$ (*representation combining*) to subsequently produce the target word y_t.

value) pairs. The matching stage is based on the key records while the reading stage uses the value records. From here on, we use *cache* to denote the cache-like memory.

Since modern NMT models generate translation in a word-by-word manner, translation information is generally stored at word level, including source-side context that embeds content being translated and target-side context that corresponds to the generated word. With the goal of remembering translation history in mind, the key should be designed with features to help match it to the source-side context, while the value should be designed with features to help match it to the target-side context. To this end, we define the cache slots as pairs of vectors $\{(\mathbf{c}_1, \mathbf{s}_1), \ldots, (\mathbf{c}_i, \mathbf{s}_i), \ldots, (\mathbf{c}_I, \mathbf{s}_I)\}$ where $\mathbf{c}_i$ and $\mathbf{s}_i$ are the attention context vector and its corresponding decoder state at time step i from the previous translations. The two types of representation vectors correspond well to the source- and target-side contexts (Tu et al., 2017a).

Figure 2(b) illustrates the model architecture. At each decoding step t, the current attention context $\mathbf{c}_t$ serves as a query, which is used to *match* and *read* from the cache looking for relevant information to generate the target word. The retrieved vector $\mathbf{m}_t$, which embeds target-side contexts of generating similar words in the translation history, is *combined* with the current decoder state $\mathbf{s}_t$ to subsequently produce the target word y_t (Section 3.2). When the full translation is generated, the decoding contexts are stored in the cache as a history for future translations (Section 3.3).

3.2 Reading from Cache

Cache reading involves the following three steps:

Key Matching The goal of key matching is to retrieve similar records in the cache. To this end, we exploit the attention context representations $\mathbf{c}_t$ to define a probability distribution over the records in the cache. Using context representations as keys in the cache, the cache lookup operator can be implemented with simple dot products between the stored representations and the current one:

$$P_m(\mathbf{c}_i \,|\mathbf{c}_t) = \frac{\exp(\mathbf{c}_t^\top \mathbf{c}_i)}{\sum_{i'=1}^{I} \exp(\mathbf{c}_t^\top \mathbf{c}'_i)} \tag{5}$$

where $\mathbf{c}_t$ is the attention context representation at the current step t, $\mathbf{c}_i$ is the stored representation at the i-th slot of the cache, and I is the number of slots in the cache. In contrast to existing memory-augmented neural networks, the proposed cache avoids the need to learn the memory matching parameters, such as those related to parametric attention models (Sukhbaatar et al., 2015; Daniluk et al., 2017), transformations between the query and keys (Miller et al., 2016; Gu et al., 2018), or human-defined scalars to control the flatness of the distribution (Grave et al., 2017).[2]

Value Reading The values of the cache is read by taking a sum over the stored values $\mathbf{s}_i$, weighted by the matching probabilities from the keys, and the retrieved vector $\mathbf{m}_t$ is returned:

$$\mathbf{m}_t = \sum_{(\mathbf{c}_i, \mathbf{s}_i) \in cache} P_m(\mathbf{c}_i \,|\mathbf{c}_t)\, \mathbf{s}_i.$$

From the view of memory-augmented neural networks, the matching probability $P_m(\mathbf{c}_i \,|\mathbf{c}_t)$ can be interpreted as the probability to retrieve similar target-side information $\mathbf{m}_t$ from the cache given the source-side context $\mathbf{c}_t$, where the desired answer is the contexts related to similar target words generated in past translations.

Representation Combining The final decoder state that is used to generate the next-word distribution is computed from a linear combination of the original decoder state $\mathbf{s}_t$ and the output vector $\mathbf{m}_t$ retrieved from the cache:[3]

$$\tilde{\mathbf{s}}_t = (\mathbf{1} - \boldsymbol{\lambda}_t) \otimes \mathbf{s}_t + \boldsymbol{\lambda}_t \otimes \mathbf{m}_t \tag{6}$$

$$P(y_t|y_{<t}, \mathbf{x}) = g(y_{t-1}, \mathbf{c}_t, \tilde{\mathbf{s}}_t) \tag{7}$$

where $\otimes$ is an element-wise multiplication, and $\boldsymbol{\lambda}_t \in \mathbb{R}^d$ is a dynamic weight vector calculated at each decoding step. This strategy is inspired by the concept of *update gate* from GRU (Cho et al., 2014), which takes a linear sum between the previous hidden state and the candidate new hidden state. The starting point for this strategy is an observation: generating target words at different steps has the different needs of the translation history. For example, translation history representation is more useful if a similar slot is retrieved in the cache, while less by other cases. To this end, we calculate the dynamic weight vector by:

$$\boldsymbol{\lambda}_t = \sigma(\mathbf{U}\mathbf{s}_t + \mathbf{V}\mathbf{c}_t + \mathbf{W}\mathbf{m}_t) \tag{8}$$

Here $\sigma(\cdot)$ is a logistic sigmoid function, and $\{\mathbf{U} \in \mathbb{R}^{d \times d}, \mathbf{V} \in \mathbb{R}^{d \times l}, \mathbf{W} \in \mathbb{R}^{d \times d}\}$ are the newly introduced parameter matrices with d and l being the number of units of decoder state and attention context vector, respectively. Note that $\boldsymbol{\lambda}_t$ has the same dimensionality as $\mathbf{s}_t$ and $\mathbf{m}_t$, and thus each element in the two vectors has a distinct interpolation weight. In this way, we offer a more precise control to combine the representations, since different elements retain different information.

The addition of the continuous cache to a NMT model inherits the advantages of cache-like memories: the probability distribution over generated words is updated online depending on the translation history, and consistent translations can be generated when they have been seen in the history. The neural cache also inherits the ability of the decoder hidden states to model longer-term cross-sentence contexts than intra-sentence context, and thus allows for a finer modeling of the document-level context.

3.3 Writing to Cache

The cache component is an external key-value memory structure which stores I elements of recent histories, where the key at position $i \in [1, M]$ is $\mathbf{k}_i$ and its value is $\mathbf{v}_i$. For each key-value pair, we also store the corresponding target word y_t as an indicator for the following updating operator. [4]

In this work, we focus on learning to remember and exploit cross-sentence translation history. Accordingly, different from (Grave et al., 2017; Kawakami et al., 2017) where the cache is updated after each generation of target word, we write to the cache after a translation sentence is fully generated. Given a generated translation sentence $\mathbf{y} = \{y_1, \dots, y_t, \dots, y_T\}$, its corresponding attention vector sequence is $\{\mathbf{c}_1, \dots, \mathbf{c}_t, \dots, \mathbf{c}_T\}$ and

[2] We tried these matching implementations in our preliminary experiments, but found no improvements for this task.

[3] We tried the strategy of "Gating Auxiliary Context" used in (Wang et al., 2017a) in our preliminary experiments, and found similar performance.

[4] In the writing phrase, the cache component works like a standard cache, in which the target word y_t serves as the "key" to address the "value" ($\mathbf{k}_t$, $\mathbf{v}_t$) for updating the cache.

the decoder state sequence is $\{\mathbf{s}_1, \ldots, \mathbf{s}_t, \ldots, \mathbf{s}_T\}$. Each triple $\langle \mathbf{c}_t, \mathbf{s}_t, y_t \rangle$ is written to the cache as follows:

1. *If y_t does not exist in the cache*, an empty slot is chosen or the least recently used slot is overwritten, where the key slot is c_t, the value slot is $\mathbf{s}_t$ and the indicator is y_t.
2. *If y_t already exists in the cache at some position i*, the key and value are updated: $\mathbf{k}_i = (\mathbf{k}_i + \mathbf{c}_t)/2$ and $\mathbf{v}_i = (\mathbf{v}_i + \mathbf{s}_t)/2$.

From the perspective of "general cache policy", it can be regarded as a sort of exponential decay, since at each update the previous keys and values are halved. From the perspective of continuous cache, on the other hand, the intuition behind it is to model temporal order for the same word – the more recent histories serve as more important roles.

Some researchers may worry that the key $\mathbf{k}_i$ and the attention vector $\mathbf{c}_t$ could be fully unrelated, since they "align" the same word y_t to the source words of different source sentences. We believe that such cases would rarely happen. When $\mathbf{c}_t$ is aligned to a target word y_t (we assume that the aligns are always correct and align error problem is beyond the focus of this work), we expect that a certain portion of $\mathbf{c}_t$ and the embedding of y_t are semantically equivalent (that is how the information of the source side is transformed to the target side). Therefore, there should be always a certain relation among attention vectors, which are aligned to the same target word. Averaging the attention vectors in different source sentences is expected to highlight the shared portion (*i.e.,* corresponds to y_t) and dilute the unshared parts (*i.e.,* correspond to the contexts of different source sentences).

3.4 Training and Inference

Training Two pass strategies have proven useful to ease training difficulty when the model is relatively complicated (Shen et al., 2016; Wang et al., 2017b; Tu et al., 2017b). Inspired by this, we add the cache to a pre-trained NMT model with fine training of only the new parameters related to the cache.

First, we pre-train a standard NMT model which is able to generate reasonable representations (*i.e.,* $\mathbf{c}_t$ and $\mathbf{s}_t$) to interact with the cache. Formally, the parameters $\boldsymbol{\theta}$ of the standard NMT model are trained to maximize the *likelihood* of a set of training examples $\{[\mathbf{x}^n, \mathbf{y}^n]\}_{n=1}^N$:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log P(\mathbf{y}^n|\mathbf{x}^n; \boldsymbol{\theta}) \qquad (9)$$

where the probabilities of generating target words are computed by Equation 2.

Second, we fix the trained parameters $\hat{\boldsymbol{\theta}}$ and only fine train the new parameters $\boldsymbol{\gamma} = \{\mathbf{U}, \mathbf{V}, \mathbf{W}\}$ related to the cache (*i.e.,* Equation 8):

$$\hat{\boldsymbol{\gamma}} = \arg\max_{\boldsymbol{\gamma}} \sum_{n=1}^{N} \log P(\mathbf{y}^n|\mathbf{x}^n; \hat{\boldsymbol{\theta}}, \boldsymbol{\gamma}) \qquad (10)$$

where the probabilities of generating target words are computed by Equation 7, and $\hat{\boldsymbol{\theta}}$ are trained parameters via Equation 9. During training, the representations $\mathbf{c}_t$ and $\mathbf{s}_t$ remain the same for a given sentence pair with the fixed NMT parameters, thus the cache can be explicitly trained to learn when to exploit translation history to maximize the overall translation performance.

Inference Once a model is trained, we use a beam search to find a translation that approximately maximizes the likelihood, which is the same as standard NMT models. After the beam search procedure is finished, we write to the cache the representations that correspond to the 1-best output. The reason why we do not use k-best outputs or all hypotheses in the beam search is two-fold: (1) we want to improve the translation consistency for the final outputs; and (2) continuous representations suffer less from data sparsity problem, in the scenario of which k-best outputs generally works better. Our premiliary experiments validate our assumption, in which k-best outputs or hypotheses does not show improvement over their 1-best counterpart.

4 Experiment

4.1 Setup

Data We carried out Chinese-English translation experiments on multiple domains, each of which differs from others in topic, genre, style, level of formality, etc.

Domain	Training			Tuning			Test		
	$\vert S\vert$	$\vert W\vert$		$\vert S\vert$	$\vert W\vert$		$\vert S\vert$	$\vert W\vert$	
		Zh	En		Zh	En		Zh	En
News	1.25M	27.9M	34.5M	878	22.6K	23.7K	6.8K	174.1K	186.9K
Subtitle	2.15M	12.1M	16.6M	1.1K	6.7K	9.2K	1.2K	6.7K	9.5K
TED	0.21M	4.1M	4.4M	887	21.3K	17.5K	5.5K	104.1K	92.2K

Table 1: Statistics of sentences ($|S|$) and words ($|W|$). K stands for thousands and M for millions.

- **News**: The News domain is extracted from LDC corpora.[5] Most sentences in this corpora are formal articles with syntactic structures such as complicated conjuncted phrases, which make textual translation very difficult. We choose the NIST 2002 (MT02) dataset as tuning set, and the NIST 2003-2008 (MT03-08) datasets as test sets.

- **Subtitle**: The subtitles are extracted from TV episodes, which are usually simple and short (Wang et al., 2018). Most of the translations of subtitles do not preserve the syntactic structures of their original sentences at all. We randomly select two episodes as the tuning set, and two other episodes as the test set.[6]

- **TED**: The corpora are from the MT track on TED Talks of IWSLT2015 (Cettolo et al., 2012).[7] Koehn and Knowles (2017) point out that NMT systems have a steeper learning curve with respect to the amount of training data, resulting in worse quality in low-resource settings. The TED talks are difficult to translate for its variety of topics while small-scale training data. We choose the "dev2010" dataset as the tuning set, and the combination of "tst2010-2013" datasets as the test set.

The statistics of the corpora are listed in Table 1. As seen, the averaged lengths of the source sentences in News, Subtitle, and TED domains are 22.3, 5.6, and 19.5 words, respectively. We use the case-insensitive 4-gram NIST BLEU score (Papineni et al., 2002) as evaluation metric, and *sign-test* (Collins et al., 2005) for statistical significance test.

Models The baseline is a re-implemented attention-based NMT system RNNSEARCH, which incorporates dropout (Hinton et al., 2012) on the output layer and improves the attention model by feeding the lastly generated word. For training RNNSEARCH, we limited the source and target vocabularies to the most frequent 30K words in Chinese and English, and employ an unknown replacement post-processing technique (Jean et al., 2015; Luong et al., 2015b). We trained each model with the sentences of length up to 80 words in the training data. We shuffled mini-batches as we proceed and the mini-batch size is 80. The word embedding dimension is 620 and the hidden layer dimension is 1000. We trained for 15 epochs using Adadelta (Zeiler, 2012), and selected the model that yields best performances on the validation set.

For our model, we used the same setting as RNNSEARCH if applicable. The parameters of our model that are related to the standard encoder and decoder were initialized by the baseline RNNSEARCH model and were fixed in the following step. We further trained the new parameters related to the cache for another 5 epochs. Again, the model that performs best on the tuning set was selected as the final model.

4.2 Effect of Cache Size

Inspired by the recent success of the continuous cache on language modeling (Grave et al., 2017), we thought it likely that a large cache would benefit from the long-range context, and thus outperforms a small one. This turned out to be false. Table 2 lists translation performances of different cache sizes on the tuning set. As seen, small caches (*e.g.,* size=25) generally achieve similar performances with larger

[5] LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, LDC2004T08 and LDC2005T06.

[6] The corpora are available at `https://github.com/longyuewangdcu/tvsub`.

[7] `https://wit3.fbk.eu/mt.php?release=2015-01`

Cache	News	Subtitle	TED	Ave
0	38.36	27.54	8.45	24.78
25	39.34	28.36	**9.24**	**25.65**
50	39.36	28.32	9.18	25.62
100	39.48	28.15	9.23	25.62
200	39.39	**28.39**	8.98	25.59
500	**39.56**	28.10	9.07	25.58
1000	39.37	27.90	8.89	25.39

Table 2: Translation performances of different cache sizes on the tuning sets.

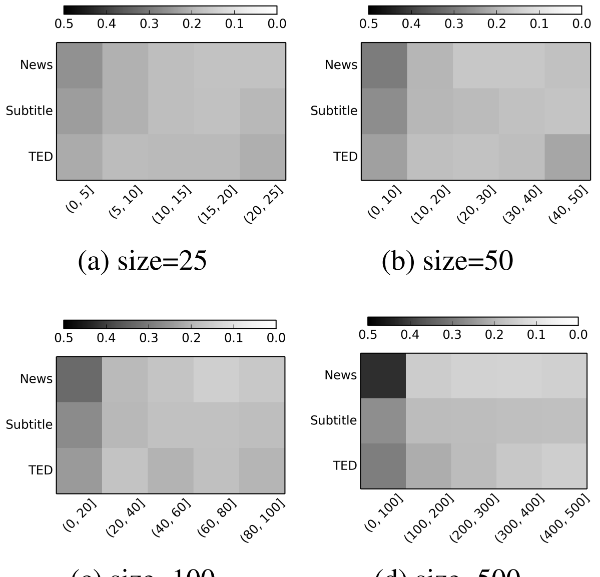
Overwrite	News	Subtitle	TED	Ave
×	39.42	28.26	9.12	25.60
✓	39.34	28.36	9.24	25.65

Table 3: Effect of the cache overwrite mechanism for slots that correspond to the same target word.

caches (*e.g.,* size=500). At the very start, we attributed this to the strength of the cache overwrite mechanism for slots that correspond to the same target word, which implicitly models long-range contexts by combining different context representations of the same target word in the translation history. As shown in Table 3, the overwrite mechanism contributes little to the good performance of smaller cache.

There are several more possible reasons. First, a larger cache is able to remember a longer translation history (*i.e.,* cache capacity), while poses difficulty to matching related records in the cache (*i.e.,* matching accuracy). Second, the current caching mechanism fails to model long-range context well, which suggests a better modeling of long-term dependency for future work. Finally, neighboring sentences are more correlated than long-distance sentences, and thus modeling short-range context properly works well (Daniluk et al., 2017). In the following experiment, we try to validate the last hypothesis by visualizing which positions in the cache are attended most by the proposed model.

Cache Matching Probability Distribution Following Daniluk et al. (2017), we plot in Figure 3 the average matching probability that the proposed model pays to specific positions in the history. As seen, the proposed approach indeed pays more attention to the most recent history (*e.g.,* the leftmost positions) in all domains. Specifically, the larger the cache, the more attention the model pays to the most recent history.



(a) size=25

(b) size=50

(c) size=100

(d) size=500

Figure 3: Average cache matching probability distribution on the tune sets,where the leftmost positions represent the most recent history.

Notably, there are still considerable differences among the different domains. For example, the proposed model attends over records further in the past more often in the Subtitle and TED domains than in the News domain. This may be because a talk in the TED testset contains more words than an article in the News testset (1.9K vs. 0.6K words). Though a scene in the Subtitle testset contains the least words (*i.e.,* 0.3K words), repetitive words and phrases are observed in neighboring scenes of the same episode, which is generally related to a specific topic. Given that larger caches do not lead to any performance improvement, it seems to be notoriously hard to judge whether long-range contexts are not modeled well, or they are less useful than the short-range contexts. We leave the validation for future work.

For the following experiments, the cache size is set to 25 unless otherwise stated.

Model	News		Subtitle		TED		Ave	
	BLEU	△	BLEU	△	BLEU	△	BLEU	△
Base	35.39	–	32.92	–	11.69	–	26.70	–
(Wang et al., 2017a)	**36.52***	**+3.2%**	33.34	+1.0%	12.43*	+6.3%	27.43	+2.7%
(Jean et al., 2017)	36.11*	+2.0%	33.00	0%	12.46*	+6.6%	27.19	+1.8%
Ours	**36.48***	**+3.1%**	**34.30***	**+3.9%**	**12.68***	**+8.5%**	**27.82**	**+4.2%**

Table 4: Translation qualities on multiple domains. "*" indicates statistically significant difference ($p < 0.01$) from"Base" , and "△" denotes relative improvement over "Base".

Model	# Para.	Speed	
		Train	*Test*
Base	84.2M	1469.1	21.1
(Wang et al., 2017a)	103.0M	300.2	20.8
(Jean et al., 2017)	104.2M	933.8	19.4
Ours	88.2M	1163.9	21.1

Table 5: Model complexity. "Speed" is measured in words/second for both training and testing. We employ a beam search with beam being 10 for testing.

4.3 Main Results

Table 4 shows the translation performances on multiple domains with different textual styles. As seen, the proposed approach significantly outperforms the baseline system (*i.e.,* Base) in all cases, demonstrating the effectiveness and universality of our model. We reimplemented the models in Wang et al. (2017a) and Jean et al. (2017) on top of the baseline system, which also exploit cross-sentence context in terms of source-side sentences. Both approaches achieve significant improvements in the News and TED domains, while achieve marginal or no improvement in the Subtitle domain. Comparing with these two approaches, the proposed model consistently outperforms the baseline system in all domains, which confirms the robustness of our approach. We attribute the superior translation quality of our approach in the Subtitle domain to the exploitation of target-side information, since most of the translations of dialogues in this domain do not preserve the syntactic structure of their original sentences at all. They are completely paraphrased in the target language and seem very hard to be improved with only source-side cross-sentence contexts.

Table 5 shows the model complexity. The cache model only introduces 4M additional parameters (*i.e.,* related to Equation 8), which is small compared to the numbers of the existing models, such as Wang et al. (2017a) (*i.e.,* 18.8M) and Jean et al. (2017) (*i.e.,* 20M). Our model is more efficient in training, which benefit from training cache-related parameters only. To minimize the waste of computation, the other models sort 20 mini-batches by their lengths before parameter updating (Bahdanau et al., 2015), while our model cannot enjoy the benefit since it depends on the hidden states of preceding sentences.[8] Concerning decoding with additional attention models, our approach does not slow down the decoding speed, while Jean et al. (2017) decreases decoding speed by 8.1%. We attribute this to the efficient strategies for cache key matching without any additional parameters.

4.4 Deep Fusion vs. Shallow Fusion

Some researchers would expect that storing the words may be a better way to encourage lexical consistency, as done in Grave et al. (2017). Following Gu et al. (2018), we call this a *Shallow Fusion* at shallow word level, in contrast to deep fusion at a deep representation level (*i.e.,* our approach). We follow Grave et al. (2017) to calculate the probability of generating y_t in shallow fusion as

$$\begin{aligned} P(y_t) &= (1-\lambda_t)P_{vocab}(y_t) + \lambda_t P_{cache}(y_t) \\ P_{cache}(y_t) &= \mathbb{1}_{\{y_t=y_i\}} P_m(\mathbf{c}_i|\mathbf{c}_t) \end{aligned}$$

in which $P_{vocab}(y_t)$ is the probability of the NMT model (Equation 2) and $P_m(\mathbf{c}_i|\mathbf{c}_t)$ is the cache prob-

[8] To make a fair comparison, which means our model is required to train all the parameters and the other models cannot use mini-batch sorting; the training speeds for the models listed in Table 5 are 728.8, 159.8, 572.4, and 627.3, respectively.

Model	Tune	Test
BASE	38.36	35.39
Shallow Fusion	38.34	35.18
Deep Fusion	39.34	36.48

Table 6: Comparison of shallow fusion (*i.e.,* words as cache values) and deep fusion (*i.e.,* continuous vectors as cache values) in the News domain.

ability (Equation 5). We compute the interpolation weight λ_t in the same way as Equation 8, except λ_t is a scalar instead of a vector.

Table 6 lists the results by comparing shallow fusion and deep fusion on the widely evaluated News domain (Tu et al., 2016; Li et al., 2017; Zhou et al., 2017; Wang et al., 2017c). As seen, deep fusion significantly outperforms its shallow counterpart, which is consistent with the results in Gu et al. (2018). Different from Gu et al. (2018), the shallow fusion does not achieve improvement over the baseline system. One possible reason is the generated words are less repetitive in the translation history than in similar sentences retrieved from the training corpus. Accordingly, storing words in the cache encourages lexical consistency at the cost of introducing noises, while storing continuous vectors improves this problem by doing fusion in a soft way. In addition, the continuous vectors can store useful information beyond a single word, which we will show later.

4.5 Translation Patterns Stored in the Cache

In this experiment, we present analysis to gain insight about what kinds of translation patterns are captured by the cache to potentially improve translation performance, as shown in Figure 4.

Tense Consistency Consistency is a critical issue in document-level translation, where a repeated term should keep the same translation throughout the whole document (Xiao et al., 2011; Carpuat and Simard, 2012). Among all consistency cases, we are interested in the verb tense consistency. We found our model works well on improving tense consistency. For example, the baseline model translated the word “觉得” into present tense “*feel*” in present tense (Figure 4(a)), while from the translation history (Table 1) we can learn it should be translated into “felt” in past tense. The cache model can improve tense consistency by exploring document-level context. As shown in the left panel of Figure 4(b), the proposed model generates the correct word “*felt*” by attending to the desired slot in the cache. It should be emphasized that our approach is still likely to generate the correct word even without the cache slot “felt”, since the previously generated word “*everyone*” already attended to a slot “*should*”, which also contains information of the past tense. The improvement of tense consistency may not lead to a significant increase of BLEU score, but is very important for user experience.

Fuzzily Matched Patterns Besides exactly matched lexical patterns (*e.g.,* the slot “felt”), we found that the cache also stores useful “fuzzy match” patterns, which can improve translation performance by acting as some kind of “indicator” context. Take the generation of “*opportunity*” in the left panel of Figure 4(b) as an example, although the attended slots “*courses*”, “*training*”, “*pressure*”, and “*tasks*” are not matched with “*opportunity*” at lexical level, they are still helpful for generating the correct word “*opportunity*” when working together with the attended source vector centering at “机遇”.

Patterns Beyond Word Level By visualizing the cache during the translation process, we found that the proposed cache is able to remember not only word-level translation patterns, but also phrase-level translation patterns, as shown in the right panel of Figure 4(b). The latter is especially encouraging to us, since phrases play an important role in machine translation while it is difficult to integrate them into current NMT models (Zhou et al., 2017; Wang et al., 2017c; Huang et al., 2018). We attribute this to the fact that decoder states, which serve as cache values, stores phrasal information due to the strength of decoder RNN on memorizing short-term history (*e.g.,* previous few words).

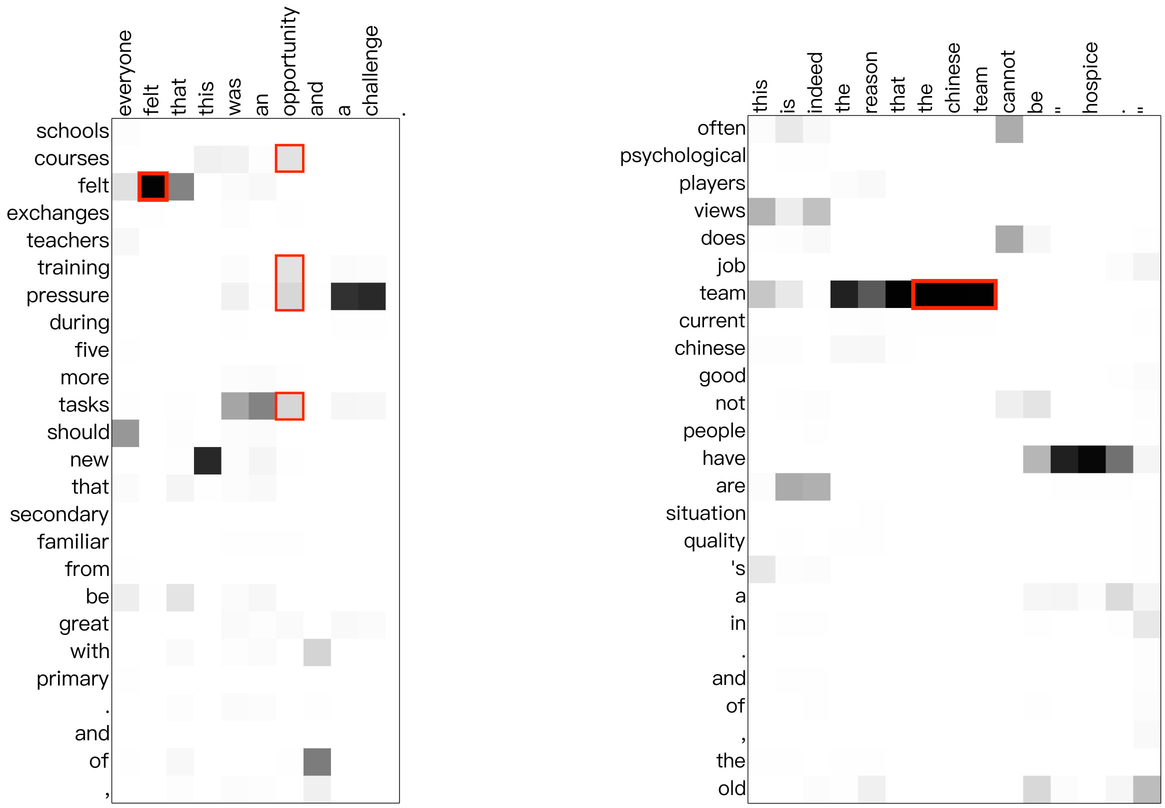
5 Related Work

Our research builds on previous work in the field of *memory-augmented neural networks*, *exploitation of cross-sentence contexts* and *cache in NLP*.

Memory-Augmented Neural Networks Neural Turing Machines (Graves et al., 2014) and Mem-

Input	大家 **觉得** 这 也是 一次 **机遇** ， 一次 挑战 。
Reference	everyone **felt** that this was also an **opportunity** and a challenge .
BASE	we *feel* that it is also a *challenge* and a challenge .
OURS	everyone **felt** that this was an **opportunity** and a challenge .
Input	这 确实 是 **中国队** 不能 “ 善终 ” 的 一 个 原因 。
Reference	this is indeed a reason why **the chinese team** could not have a “ good ending . ”
BASE	this is indeed the reason why *china* can not be “ hospice . ”
OURS	this is indeed a reason why **the chinese team** cannot be “ hospice . ”

(a) We italicize some *mis-translated* errors and highlight the **correct** ones in bold. Our approach is able to correct the errors with the target-side context retrieved from the cache, as shown below.



(b) Visualization of the cache matching matrix, in which the x-axis is the generated words and the y-axis is the cache slots indicated by the corresponding word in translation history. Our approach improves performance by retrieving useful information (*e.g.,* verb tense for “felt” and phrasal-level patterns for “the chinese team”, in boxes with red frames) from the cache.

… **觉得** 新 课程 … … **felt** that new courses …	… ， 然后 **中国队** 将 比分 … … , **the chinese team** won the …
… 在 听完 了 所有 **培训** 课程 后 … … after listening to all **training** courses …	… **中国队** 经常 是 在 形势 大好 … … **the chinese team** often does not have a …

(c) Bilingual snippets in translation history that correspond to the slots in boxes with red frames.

Figure 4: Translation examples in which the proposed approach shows its ability to remember translation patterns at different levels of granularity and matching.

ory Networks (Weston et al., 2015; Sukhbaatar et al., 2015) are early models that augment neural networks with a possibly large external memory. Our work is based on the Memory Networks, which have proven useful for question answering and document reading tasks (Weston et al., 2016; Hill et al., 2016). Specifically, we use the Key-Value Memory Network (Miller et al., 2016), which is a simplified version of Memory Networks with better interpretability and has yielded encouraging results in document reading (Miller et al., 2016), question answering (Pritzel et al., 2017) and language modeling (Tran et al., 2016; Grave et al., 2017; Daniluk et al., 2017). We use the memory to store information specific to the translation history so that this information is available to influence future translations. Closely related to our approach, Grave et al. (2017) use a continuous cache to improve language modeling by capturing longer history. We generalize from the original model and adapt it to machine translation: we use the cache to store bilingual information rather than monolingual information, and release the hand-tuned parameters for cache matching.

In the context of neural machine translation, Kaiser et al. (2017) use an external key-value memory to remember rare training events in test time, and Gu et al. (2018) use a memory to store a set of sentence pairs retrieved from the training corpus given the source sentence. This is similar to our approach in exploiting more information than a current source sentence with a key-value memory. Unlike their approaches, ours aims to *learning to remember translation history* rather than incorporating arbitrary meta-data, which results in different sources of the auxiliary information (*e.g.,* previous translations vs. similar training examples). Accordingly, due to the different availability of target symbols in the two scenarios, different strategies of incorporating the retrieved values from the key-value memory are adopted: hidden state interpolation (Gulcehre et al., 2016) performs better on our task while word probability interpolation (Gu et al., 2016) works better in Gu et al. (2018).

Exploitation of Cross-Sentence Context Cross-sentence context, which is generally encoded into a continuous space using a neural network, has a noticeable effect in various deep learning based NLP tasks, such as language modeling (Ji et al., 2015; Wang and Cho, 2016), query suggestion (Sordoni et al., 2015), dialogue modeling (Vinyals and Le, 2015; Serban et al., 2016), and machine translation (Wang et al., 2017a; Jean et al., 2017).

In statistical machine translation, cross-sentence context has proven useful for alleviating inconsistency and ambiguity arising from a single source sentence. Wide-range context is firstly exploited to improve statistical machine translation models (Gong et al., 2011; Xiao et al., 2012; Hardmeier et al., 2012; Hasler et al., 2014). Closely related to our approach, Gong et al. (2011) deploy a *discrete* cache to store bilingual phrases from the best translation hypotheses of previous sentences. In contrast, we use a *continuous* cache to store bilingual representations, which are more suitable for neural machine translation models.

Concerning neural machine translation, Wang et al. (2017a) and Jean et al. (2017) are two early attempts to model cross-sentence context. Wang et al. (2017a) use a hierarchical RNN to summarize the previous K (*e.g.,* $K = 3$) source sentences, while Jean et al. (2017) use an additional set of encoder and attention model to encode and select part of the previous source sentence for generating each target word. While their approaches only exploit source-side cross-sentence contexts, the proposed approach is able to take advantage of bilingual contexts by directly leveraging continuous vectors to represent translation history. As shown in Tables 4 and 5, the proposed approach is more robust in improving translation performances across different domains, and is more efficient in both training and testing.

Cache in NLP In the NLP community, the concept of "cache" is firstly introduced by Kuhn and Mori (1990), which augments a statistical language model with a cache component and assigns relatively high probabilities to words that occur elsewhere in a given text. The success of the cache language model in improving word prediction rests on capturing the "burstiness" of word usage in a local context. It has been shown that caching is by far the most useful technique for perplexity reduction over the standard n-gram approach (Goodman, 2001), and becomes a standard component in most LM toolkits, such as IRSTLM (Federico et al., 2008). Inspired by the great success of caching on language modeling, Nepveu et al. (2004) propose to use a cache model to adapt language and translation models for SMT systems, and Tiedemann (2010) apply an exponentially decaying cache for the domain adaptation task. In this work, we have generalized and adapted from the original discrete cache model, and integrate a "continuous" cache-like memory into NMT models.

6 Conclusion

We propose to augment NMT models with a cache-like memory network, which stores translation history in terms of bilingual hidden representations at decoding steps of previous sentences. The cache component is an external key-value memory structure with the keys being attention vectors and values being decoder states collected from translation history. At each decoding step, the probability distribution over generated words is updated online depending on the history information retrieved from the cache with a query of the current attention vector. Using simply a dot-product for key matching, this history information is quite cheap to store and can be accessed efficiently.

In our future work, we expect several developments that will shed more light on utilizing long-range contexts, e.g., designing novel architectures, and employing discourse relations instead of directly using decoder states as cache values.

Acknowledgments

Yang Liu is supported by the National Key R&D Program of China (No. 2017YFB0202204) and National Natural Science Foundation of China (No. 61432013, No. 61522204). We thank action editor Philipp Koehn and three anonymous reviewers for their insightful comments.

References

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

[Carpuat and Simard2012] Marine Carpuat and Michel Simard. 2012. The trouble with SMT consistency. In *The Workshop on Statistical Machine Translation*, pages 442–449.

[Cettolo et al.2012] Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: web inventory of transcribed and translated talks. In *EAMT 2012*, pages 261–268.

[Cho et al.2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP 2014*, pages 1724–1734.

[Choi et al.2016] Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. 2016. Context-dependent word representation for neural machine translation. *arXiv:1607.00578*.

[Collins et al.2005] M. Collins, P. Koehn, and I. Kučerová. 2005. Clause restructuring for statistical machine translation. In *ACL 2005*, pages 531–540.

[Daniluk et al.2017] Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. In *ICLR 2017*.

[Federico et al.2008] Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. IRSTLM: an open source toolkit for handling large scale language models. In *Interspeech*, pages 1618–1621.

[Gong et al.2011] Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level statistical machine translation. In *EMNLP 2011*, pages 909–919.

[Goodman2001] Joshua T. Goodman. 2001. A bit of progress in language modeling. *Journal of Computer Speech and Language*, 15(4):403–434, October.

[Grave et al.2017] Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In *ICLR 2017*.

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

[Gu et al.2016] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL 2016*, pages 1631–1640.

[Gu et al.2018] Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K Li. 2018. Search engine guided non-parametric neural machine translation. In *AAAI 2018*, pages 5133–5140.

[Gulcehre et al.2016] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *ACL 2016*, pages 140–149.

[Hardmeier et al.2012] Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-wide decoding for phrase-based statistical machine translation. In *EMNLP 2012*, pages 1179–1190.

[Hasler et al.2014] Eva Hasler, Phil Blunsom, Philipp Koehn, and Barry Haddow. 2014. Dynamic topic adaptation for phrase-based MT. In *EACL 2014*, pages 328–337.

[Hill et al.2016] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR 2016*.

[Hinton et al.2012] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*.

[Huang et al.2018] Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. In *ICLR 2018*.

[Jean et al.2015] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL 2015*, pages 1–10.

[Jean et al.2017] Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? *arXiv:1704.05135*.

[Ji et al.2015] Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2015. Document context language models. In *ICLR 2015*.

[Kaiser et al.2017] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events. In *ICLR 2017*.

[Kalchbrenner et al.2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL 2014*, pages 655–665.

[Kawakami et al.2017] Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2017. Learning to create and reuse words in open-vocabulary neural language modeling. In *ACL 2017*, pages 1492–1502.

[Koehn and Knowles2017] Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the 1st Workshop on Neural Machine Translation*, pages 28–39.

[Kuhn and Mori1990] Roland Kuhn and Renato De Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583.

[Li et al.2017] Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *ACL 2017*, pages 688–697.

[Luong et al.2015a] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015a. Effective approaches to attention-based neural machine translation. In *EMNLP 2015*, pages 1412–1421.

[Luong et al.2015b] Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the Rare Word Problem in Neural Machine Translation. In *ACL 2015*, pages 11–19.

[Miller et al.2016] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *EMNLP 2016*, pages 1400–1409.

[Nepveu et al.2004] Laurent Nepveu, Guy Lapalme, Philippe Langlais, and George Foster. 2004. Adaptive language and translation models for interactive machine translation. In *EMNLP 2004*.

[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 *ACL 2002*, pages 311–318.

[Pritzel et al.2017] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. 2017. Neural episodic control. *arXiv:1703.01988*.

[Serban et al.2016] Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI 2016*, pages 3776–3783.

[Shen et al.2016] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum Risk Training for Neural Machine Translation. In *ACL 2016*, pages 1683–1692.

[Sordoni et al.2015] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM 2015*, pages 553–562.

[Sukhbaatar et al.2015] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS 2015*, pages 2440–2448.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 2014*, pages 3104–3112.

[Tiedemann2010] Jörg Tiedemann. 2010. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 8–15.

[Tran et al.2016] Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *NAACL 2016*, pages 76–85.

[Tu et al.2016] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. In *ACL 2016*, pages 76–85.

[Tu et al.2017a] Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017a. Context gates for neural machine translation. In *TACL 2017*, pages 87–99.

[Tu et al.2017b] Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017b. Neural machine translation with reconstruction. In *AAAI 2017*, pages 3097–3103.

[Vinyals and Le2015] Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the International Conference on Machine Learning, Deep Learning Workshop*, pages 1–8.

[Wang and Cho2016] Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *ACL 2016*, pages 1319–1329.

[Wang et al.2017a] Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017a. Exploiting cross-sentence context for neural machine translation. In *EMNLP 2017*, pages 2826–2831.

[Wang et al.2017b] Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017b. Neural machine translation advised by statistical machine translation. In *AAAI 2017*, pages 3330–3336.

[Wang et al.2017c] Xing Wang, Zhaopeng Tu, Deyi Xiong, and Min Zhang. 2017c. Translating phrases in neural machine translation. In *EMNLP 2017*, pages 1421–1431.

[Wang et al.2018] Longyue Wang, Zhaopeng Tu, Shuming Shi, Tong Zhang, Yvette Graham, and Qun Liu. 2018. Translating pro-drop languages with reconstruction models. In *AAAI 2018*, pages 4937–4945.

[Weston et al.2015] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *ICLR 2015*.

[Weston et al.2016] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards AI-complete question answering: a set of prerequisite toy tasks. In *ICLR 2016*.

[Xiao et al.2011] Tong Xiao, Jingbo Zhu, Shujie Yao, and Hao Zhang. 2011. Document-level consistency verification in machine translation. In *Machine Translation Summit*, pages 131–138.

[Xiao et al.2012] Xinyan Xiao, Deyi Xiong, Min Zhang, Qun Liu, and Shouxun Lin. 2012. A Topic Similarity Model for Hierarchical Phrase-based Translation. In *ACL 2012*, pages 750–758.

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

[Zhou et al.2017] Hao Zhou, Zhaopeng Tu, Shujian Huang, Xiaohua Liu, Hang Li, and Jiajun Chen. 2017. Chunk-based bi-scale decoder for neural machine translation. In *ACL 2017*, pages 580–586.