Addressing Troublesome Words in Neural Machine Translation

Yang Zhao[1,2], **Jiajun Zhang**[1,2], **Zhongjun He**[4], **Chengqing Zong**[1,2,3], **and Hua Wu**[4]
[1]National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
[4]Baidu Inc., Beijing, China
{yang.zhao, jjzhang, cqzong}@nlpr.ia.ac.cn, {hezhongjun,wu_hua} @baidu.com

Abstract

One of the weaknesses of Neural Machine Translation (NMT) is in handling low-frequency and ambiguous words, which we refer as troublesome words. To address this problem, we propose a novel memory-enhanced NMT method. First, we investigate different strategies to define and detect the troublesome words. Then, a contextual memory is constructed to memorize which target words should be produced in what situations. Finally, we design a hybrid model to dynamically access the contextual memory so as to correctly translate the troublesome words. The extensive experiments on Chinese-to-English and English-to-German translation tasks demonstrate that our method significantly outperforms the strong baseline models in translation quality, especially in handling troublesome words.

1 Introduction

Neural machine translation (NMT) based on the encoder-decoder architecture becomes the new state-of-the-art due to distributed representation and end-to-end learning (Cho et al., 2014; Bahdanau et al., 2015; Junczys-Dowmunt et al., 2016; Gehring et al., 2017; Vaswani et al., 2017).

However, the current NMT is a global model that maximizes the performance on the overall data and has problems in handling low-frequency words and ambiguous words[1], we refer these words as troublesome words and define them in Section 3.1.

Some previous work attempt to tackle the translation problem of low-frequency words. Sennrich et al. (2016) propose to decompose the words into subwords which are used as translation units so

[1]In this work, we consider a source word is ambiguous if it has multiple translations with high entropy of probability distribution.

Source: 阿尔卡特 宣称 去年 第四 季 销售 成长 近 百分之三十
Pinyin: **aerkate** cheng qunian disi ji xiaoshou **chengzhang** jin baifenzhisanshi
Reference: **alcatel** says sales in fourth quarter last year **grew** nearly 30 %
NMT: **he** said sales **grew** nearly 30 percent in fourth quarter of last year
NMT+LexiconTable: **alcatel** said sales **growth** nearly 30 percent in fourth quarter of last year

Figure 1: The NMT model produces a wrong translation for the low-frequency word "aerkat". While introducing an external lexicon table without contextual information, the model incorrectly translates the ambiguous word "chengzhang" into "growth".

that the low-frequency words can be represented by frequent subword sequences. Arthur et al. (2016) and Feng et al. (2017) try to incorporate a translation lexicon into NMT in order to obtain the correct translation of low-frequency words. However, the former method still faces the low-frequency problem of subwords. And the latter one has a drawback that they use lexicons without considering specific contexts. Fig. 1 shows an example, in which "aerkate" is an infrequent word and the baseline NMT incorrectly translates it into a pronoun "he". Incorporation of bilingual lexicon rectifies the mistake but wrongly converts "chengzhang" into an incorrect target word "growth" since an entry "(chengzhang, growth)" in the bilingual lexicon is somewhat wrongly used without taking the contexts into account. Furthermore, these two kinds of methods mainly focus on low-frequency words that are just a part of the troublesome words.

In this paper, we categorize the words (including infrequent words and ambiguous words) which are difficult to translate as troublesome words and propose a novel memory-augmented framework to address them. Our method first investigates different strategies to define the troublesome words. Then, these words and their contexts in the training data are memorized with a contextual memory which is finally accessed dynamically during decoding to solve the translation problem of the troublesome words.

Specifically, we first decode all the source sentences of the bilingual training data with baseline NMT and define the troublesome source words according to the distance between the predicted words and the gold words. The troublesome words associated with their hidden contextual representations are stored in a memory which memorizes the correct translation and the corresponding contextual information. During decoding, we activate the contextual memory when we encounter the troublesome words and employ the contextual similarity between the test sentence and the memory to determine appropriate target words. We test our methods on Chinese-to-English and English-to-German translation tasks. The experimental results demonstrate that the translation performance can be significantly improved and a large portion of troublesome words can be correctly translated. The contributions are listed as follows:

1) We are the first to define and handle the troublesome words in neural machine translation.

2) We propose to memorize not only the bilingual lexicons but also their contexts with a contextual memory.

3) We design a dynamic approach to correctly translate the troublesome words by combining the contextual memory and the NMT model.

2 Neural Machine Translation

NMT contains an encoder and a decoder. The encoder transforms a source sentence $X = \{x_1, x_2, ..., x_{Tx}\}$ into a set of context vectors $C = (h_1^m, h_2^m, ..., h_{Tx}^m)$ by using m stacked Long Short Term Memory (LSTM) layers (Hochreiter and Schmidhuber, 1997) . h_j^m is the hidden state of the top layer in encoder. The bottom layer of encoder is a bi-direction LSTM layer to collect the context from the left side and right side.

The decoder generates one target word at a time by computing $p_i^N(y_i|y_{<i}, C)$ as follows:

$$p_i^N(y_i|y_{<i}, C) = softmax(W_{y_i}\widetilde{z}_i + b_s) \quad (1)$$

where $\widetilde{z}_i$ is the attention output:

$$\widetilde{z}_i = tanh(W_z[z_i^m; c_i]) \quad (2)$$

c_i can be calculated as follows:

$$c_i = \sum_{j=1}^{Tx} a_{ij}h_i^m \quad (3)$$

where $a_{i,j}$ is the attention weight:

$$a_{i,j} = \frac{h_j^m z_i^m}{\sum_j h_j^m z_i^m} \quad (4)$$

where z_i^m is the hidden state of the top layer in decoder. More detailed introduction can be found in (Luong et al., 2015).

Notation. In this paper, we denote the whole source vocabulary by $V_S = \{s_m\}_{m=1}^{|V_S|}$ and target vocabulary by $V_T = \{t_n\}_{n=1}^{|V_T|}$, where s_m is the source word and t_n is the target word. We denote a source sentence by X and a target sentence by Y. Each source word in X is denoted by x_j. Each target word in Y is denoted by y_i. Accordingly, a target word can be denoted not only by t_n, but also by y_i. This does not contradict. t_n means this target word is the n^{th} word in vocabulary V_T, and y_i means this target word is the i^{th} word in sentence Y. Similarly, we denote a source word by s_m and x_j.

3 Method Description

Our method contains three parts: 1) definition and detection of the troublesome words (Section 3.1); 2) contextual memory construction (Section 3.2); and 3) hybrid approach combining contextual memory and baseline NMT model (Section 3.3).

3.1 Troublesome Word Definition

Generally speaking, troublesome words are those that are difficult to translate for the baseline NMT system $BNMT$. Fig. 2 shows the main process to detect the troublesome words. Given each training sentence pair (X, Y), $BNMT$ decodes the source sentence X and outputs the predicted probability of each gold target word $p_i^N(y_i)$. We call y_i an **exception** if $p_i^N(y_i)$ satisfies the predefined **exception criteria** introduced below. The source word x_j is an exception (a candidate troublesome word) if (x_j, y_i) is an entry in the word alignment A[2]. Suppose x_j appears N times in the training data and there are M exceptions among all its aligned

[2]The word alignments A is extracted using the fast-align tool (Dyer et al., 2013) on the bilingual training data with both source-to-target and target-to-source directions.

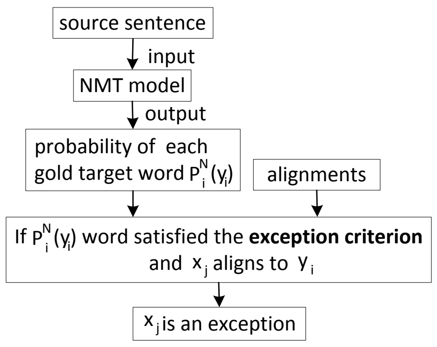
source sentence
input
NMT model
output
probability of each gold target word $P_i^N(y_i)$
alignments
If $P_i^N(y_i)$ word satisfied the **exception criterion** and x_j aligns to y_i
x_j is an exception

Figure 2: The main process to detect an exception.

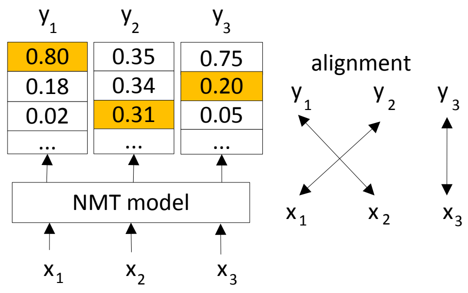

Figure 3: A toy example to show the process: if $p_i^N(y_i)$ (left) satisfies the predefined exception criteria and x_j aligns to y_i, then x_j is an exception.

gold target words. Then, the **exception rate** $r(x_j)$ will be M/N.

Definition: x_j is a **troublesome word** if $r(x_j) > \epsilon$ in which ϵ is a predefined threshold.

Exception Criteria. As discussed before, we need an exception criterion to measure whether a gold target word is an exception or not. In this paper, we investigate three exception criteria. Here, we introduce each of them through a toy example shown in Fig. 3, in which the source sentence is $X = \{x_1, x_2, x_3\}$ and the gold target sentence is $Y = \{y_1, y_2, y_3\}$. The left shows the probability distribution of all target vocabulary $p_i^N(V_T)$ at each decoding step i, where the probability of the gold target word is highlighted in yellow. The right shows the word alignments between X and Y.

1) Absolute Criterion. A gold target word y_i is an exception if its predicted probability $p_i^N(y_i)$ is lower than a predefined threshold, namely $p_i^N(y_i) < p_0$. In Fig. 3, $p_i^N(y_i)$ at each decoding step is respectively 0.8, 0.31 and 0.2. If we set $p_0 = 0.5$, $p_2^N(y_2)$ and $p_3^N(y_3)$ are lower than threshold p_0. x_1 and x_3 are both exceptions according to the alignments.

2) Gap Criterion. For this criterion, we utilize the predicted probability gap between the gold target word and the top one. Specifically, the gap can be calculated by:

$$g(y_i) = max(p_i^N(V_T)) - p_i^N(y_i) \qquad (5)$$

where $max(p_i^N(V_T))$ is the top one in the probability distribution at the i^{th} decoding step. y_i is an exception if $g(y_i) > g_0$. In Fig. 3, the largest predicted probabilities at each decoding step $max(p_i^N(V_T))$ are respectively 0.8, 0.35 and 0.75. Thus, the gap is 0.0, 0.04 and 0.55. If $g_0 = 0.1$, x_3 is an exception since $g(y_3) > g_0$ and x_3 aligns to y_3.

3) Ranking Criterion. This criterion is based on the ranking of $p_i^N(y_i)$ in $p_i^N(V_T)$ (denoted by $rank(y_i)$). If $rank(y_i) > rank_0$, then y_i is an exception. In Fig. 3, the ranking of each gold target word is 1, 3 and 2. If we set $rank_0 = 2$, then $rank(y_2) = 3 > rank_0$ and x_1 is an exception due to the alignment between x_1 and y_2.

Using the above exception criteria and the definition of troublesome words, we can detect all the source-side troublesome words in the bilingual training data.

3.2 Contextual Memory Construction

For a troublesome word, we now introduce how to build a contextual memory $\mathbb{M}$ to store its translation knowledge. Specifically, the contextual memory contains five elements:

$$\{s_m, t_n, c(s_m, t_n), p^L(s_m, t_n), r(s_m)\} \qquad (6)$$

each of them is described as follows:

- s_m is a troublesome source word.
- t_n is a gold target word for s_m.
- $c(s_m, t_n)$ is the context of lexicon pair (s_m, t_n). Here, we use the hidden states of encoder h_j to represent the context, since it contains the information from left ($\overrightarrow{h_j}$) and right ($\overleftarrow{h_j}$). Note that when we traverse the training data and memorize the contexts of all troublesome words, there must be many cases in which the same pair (s_m, t_n) appears in different contexts. In order to reduce the memory size and fuse different contexts of a same lexicon pair, we merge these memories by averaging the contexts. Assume there are K different contexts for (s_m, t_n), and they

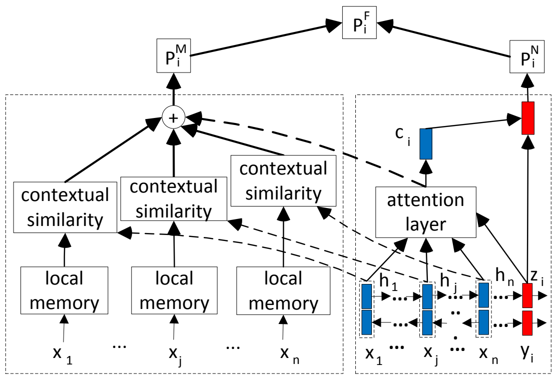

Figure 4: The architecture of contextual memory-augmented NMT.

are denoted by $h_k(s_m, t_n)$. The average context of (s_m, t_n) can be calculated by:

$$c(s_m, t_n) = \frac{\sum_{k=1}^{K} h_k(s_m, t_n)}{K} \qquad (7)$$

Note that the context here is defined on the source side.

- $p^L(s_m, t_n)$ is the lexicon translation probability. It is the average of source-to-target and target-to-source probabilities calculated through maximum likelihood estimation on word alignments.

- $r(s_m)$ is the exception rate of s_m introduced in Section 3.1 and it can indicate the translation difficulty of a source word. We will use $r(s_m)$ to determine the dynamic weights of contextual memories in Section 4.

Noise Reduction. As we know, the training data and word alignments are not perfect and may introduce noise to the contextual memory. To reduce the noise, we employ two strategies.

1) To improve the quality of the alignments A, we derive the alignment results from source-to-target and target-to-source, respectively. We only save the alignments which exist in both directions.

2) We eliminate the lexicon pairs whose translation probabilities are too small. For a lexicon pair (s_m, t_n), if its lexicon translation probability is smaller than 0.01, we treat this lexicon pair as a noisy sample and eliminate it from our memory.

3.3 Integrating Contextual Memory into NMT

In this section, we integrate the contextual memory into NMT to handle troublesome words. The overall framework is depicted in Fig. 4 and the integration process can be divided into four steps:

Step 1. Given a test sentence X, the first step is to find the troublesome words in X and collect corresponding **local memories** from the **global contextual memory** $\mathbb{M}$. For each source word x_j, we retrieve from $\mathbb{M}$ if it is a troublesome word and obtain the local memory as follows:

$$\{x_j, t_n, c(x_j, t_n), p^L(x_j, t_n), r(x_j)\} \qquad (8)$$

Step 2. The next step is to measure the contextual similarity between the context in the test sentence X and the context in $\mathbb{M}$. For the troublesome word $x_j \in X$, we still use the encoder hidden state h_j to represent the context in X. The corresponding context in $\mathbb{M}$ is $c(x_j, t_n)$ in Eq. (8). Here, we use a feed-forward network to measure this similarity[3]:

$$d_j(t_n) = \\ sigmoid(v_d^T * tanh(W_h * h_j + W_c * c(x_j, t_n))) \qquad (9)$$

where v_d, W_h and W_c are learnable parameters. The sigmoid function guarantees the similarity score is in the range $(0, 1)$. This similarity $d_j(t_n)$ will determine whether or not to adopt the target translation word t_n in $\mathbb{M}$.

Step 3. The next task is calculating the probability $p_i^M(t_n)$ of t_n at each decoding step i. $p_i^M(t_n)$ is the probability predicted by the contextual memory $\mathbb{M}$ and is calculated by:

$$p_i^M(t_n) = \sum_{j}^{Tx} a_{i,j} * d_j(t_n) * p^L(x_j, t_n) \qquad (10)$$

where $a_{i,j}$ is the attention weight, $d_j(t_n)$ is the context similarity in Eq. (9), and $p^L(x_j, t_n)$ is the lexicon translation probability.

Step 4. The final task is to combine the memory predicted probability (p_i^M in Eq. (10)) and the NMT predicted one (p_i^N in Eq. (1)). Here, we propose a dynamic strategy to balance these two probabilities:

$$p_i^F(t_n) = \lambda_i * p_i^M(t_n) + (1 - \lambda_i) * p_i^N(t_n) \quad (11)$$

where $p_i^F(t_n)$ is the final probability of the target word t_n, λ_i is the dynamic weight to adjust

[3]In our preliminary experiment, we also try cosine distance to measure this similarity, while the performance of cosine distance is lower than the current feed-forward network method.

the contribution from the memory and NMT. Here we explain the reason why we apply the dynamic manner. Recall that for each source troublesome word s_m, we calculate its exception rate (similar to error rate). If a troublesome word has a lower exception rate, indicating that this source word is easier to be translated for the neural model. In this case, p_i^N is more reliable. Thus we design the dynamic weight λ_i according to the exception rate $r(x_j)$:

$$\begin{aligned}\lambda_i &= sigmoid(\beta_\gamma * \gamma_i)\\ \gamma_i &= \sum_{j}^{Tx} a_{i,j} * r(x_j)\end{aligned} \tag{12}$$

where β_γ is a learnable parameter. From Eq. (12), the dynamic weight λ_i is determined by both of the attention weight $a_{i,j}$, and the exception rate $r(x_j)$.

Training the parameters. As discussed above, our method contains some parameters (v_d, W_h, W_c and β_γ) to be learned. We denote the parameters introduced by our method by θ^M and the parameters in NMT by θ^N. To make it efficient, given the aligned training data $D = \{X^{(d)}, Y^{(d)}\}_{d=1}^{|D|}$, we keep θ^N unchanged and optimize θ^M by maximizing the following objective function.

$$L(\theta^M) = \frac{1}{|D|}\sum_{d=1}^{|D|}\sum_{i}^{T_y} log\ p_i^F(y_i^{(d)}; \theta^M) \tag{13}$$

where p_i^F can be calculated by Eq. (11).

4 Experimental Settings

We test the proposed methods on Chinese-to-English (CH-EN) and English-to-German (EN-DE) translation. In CH-EN translation, we use LDC corpus which includes 2.1M sentence pairs for training. NIST 2003 dataset is used for validation. NIST04-06 and 08 datasets are used for testing. In EN-DE translation, we use WMT 2014 EN-DE dataset, which includes 4.5M sentence pairs for training. 2012-2013 datasets are used for validation and 2014 dataset is used for testing.

We use the Zoph_RNN toolkit[4] to implement all described methods. In all experiments, the encoder and decoder include two stacked LSTM layers. The word embedding dimension and the size of hidden layers are both set to 1,000. The minibatch size is set to 128. We discard the training sentence pairs whose length exceeds 100. We run a total of 20 iterations for all translation tasks. We test all methods based on two granularities: words and sub-words. For word granularity, we limit the vocabulary to 30K (CH-EN) and 50K (EN-DE) for both the source and target languages. For sub-word granularity, we use the BPE method (Sennrich et al., 2016) to merge 30K (CH-EN) and 32K (EN-DE) steps. The beam size is set to 12. We use case-insensitive 4-gram BLEU (Papineni et al., 2002) for translation quality evaluation.

[4]`https://github.com/isi-nlp/Zoph_RNN`. We extend this toolkit with global attention.

We compare our method with other relevant methods as follows:

1) **Baseline**: It is the baseline NMT system with global attention (Luong et al., 2015; Zoph and Knight, 2016; Jean et al., 2015).

2) **Arthur**: It is the state-of-the-art method which incorporates discrete translation lexicons into NMT (Arthur et al., 2016). We implement Arthur et al. (2016)'s method in two different ways. In the first way, we fix the **Baseline** unchanged, and utilize Arthur et al. (2016)'s method in the test phase. We denote this system by **Arthur(test)**. In second way, we allow **Baseline** to be retrained by Arthur et al. (2016)'s method, and denote the system by **Arthur(train+test)**. We replicate the Arthurs work using the bias method with the hyper parameter being set to 0.001 as reported in their paper.

3) **X+MEM**: It is our proposed memory augment method for any neural model **X**, in which we define the troublesome word by using the gap criterion with threshold $g_0 = 0.1$. We set threshold $\epsilon = 0.05$, which is fine-tuned in validation set. It means if the exception rate of a source word exceeds 0.05, we treat this word as a troublesome word.

5 Results on CH-EN Translation

5.1 Our methods vs. Baseline

Table 1 reports the main translation results of CH-EN translation. We first compare **Baseline+MEM** with **Baseline**. As shown in row 1 and row 5 in Table 1, **Baseline+MEM** can improve over **Baseline** on all test datasets, and the average improvement is 1.37 BLEU points. The results show that our method could significantly outperform the baseline model.

#	Model	03	04	05	06	08	Avg.	△
1	Baseline	41.01	42.94	40.31	40.57	30.96	39.16	-
2	Arthur(test)	41.34	43.31	40.79	40.84	31.11	39.48	-
3	Arthur(train+test)	41.88	43.75	41.16	41.63	31.47	39.98	-
4	Baseline(sub-word)	43.93	44.74	42.46	43.01	32.53	41.33	-
5	Baseline+MEM	42.74†	43.94†	42.15†	41.94†	31.86†	40.53	+1.37
6	Arthur(train+test)+MEM	43.04†	44.65*	42.19†	42.59†	32.05*	40.90	+0.92
7	Baseline(sub-word)+MEM	44.98†	45.51†	43.93†	43.95†	33.33†	42.34	+1.01

Table 1: The main results of CH-EN translation. △ shows the BLEU points improvement of system "X+MEM" than system X. "*" indicates that system "X+MEM" is statistically significant better ($p < 0.05$) than system X and "†" indicates $p < 0.01$.

Model	#Pairs	Size	Time	BLEU
Baseline	-	-	0.406	39.16
Arthur(test)	938K	58M	0.429	39.48
Tword	125K	7.4M	0.423	39.77
+Context	125K	893M	0.508	40.19
+Dynamic	125K	893M	0.511	40.53
All+Context	938K	6.4G	1.829	40.23

Table 2: The effects of lexicon pairs only containing troublesome words (Tword), context and dynamic weights. Column **#Pairs** shows the number of lexicon pairs. Column **Time** shows the average decoding time (seconds) of per sentence.

Source: 阿尔卡特 宣称 去年 第四 季 销售 成长 近 百分之三十

Pinyin: aerkate cheng qunian disi ji xiaoshou chengzhang jin baifenzhisanshi

Reference: alcatel says sales in fourth quarter last year grew nearly 30 %

Baseline: he said sales grew nearly 30 percent in fourth quarter of last year

Arthur: alcatel says sales growth nearly 30 percent in fourth quarter of last year

Baseline+MEM: alcatel says sales grew nearly 30 percent in fourth quarter of last year

Figure 5: An example to show that considering context could produce a better translation result. In the example, **Arthur(test)** translates *chengzhang* into a wrong target word *growth*, while **Baseline+MEM** could overcome this mistake with the help of the context modeling.

5.2 Results on Sub-words

We also test the proposed method when the translation unit is sub-word. The baseline and our method using sub-word as translation unit are respectively denoted by **Baseline(sub-word)** and **Baseline(sub-word)+MEM**. The results are shown in row 4 and row 7. From the results, **Baseline(sub-word)+MEM** outperforms **Baseline(sub-word)** by 1.01 BLEU points, indicating that adopting sub-words as translation units still faces the problem of troublesome tokens, and our method could alleviate this problem.

5.3 Our Method vs. Method Using Translation Lexicon

We also compare our method with Arthur et al. (2016)'s method which incorporates a translation lexicon into NMT. Here, the comparison is conducted in two ways based on whether the baseline neural model is fixed or retrained.

Fixed Baseline. Comparing **Arthur(test)** (row 2 in Table 1) and **Baseline+MEM** (row 5 in Table 1), we can see that our proposed method can surpass **Arthur(test)** with 1.05 BLEU points. As there are three differences between our methods and **Arthur(test)**, we take the following experiments to evaluate the effect of each difference.

The first difference is that our memory only stores the lexicon pairs for troublesome words, while **Arthur(test)** utilizes all the available lexicon pairs. We implement another system which is similar to **Arthur(test)**, except that we only utilize the troublesome lexicon pairs. We denote the system by **Tword**. The results are reported in Table 2. From the results, we can find that **Tword** obtains better translation results than **Arthur(test)** while using much fewer lexicon pairs (125K vs. 938K).

The second difference is that we take the context into consideration. When we add the context on the basis of **Tword** (denoted by **+Context**), it further improves the baseline system by 1.03 BLEU points, indicating the importance of the context. Fig.5 shows the mentioned example in Section 1, in which **Arthur(test)** translates *chengzhang* into a wrong target word *growth*, while **Baseline+MEM** could overcome this mistake with the help of the context modeling.

We also implement another system, in which we build the contextual memory for all source words.

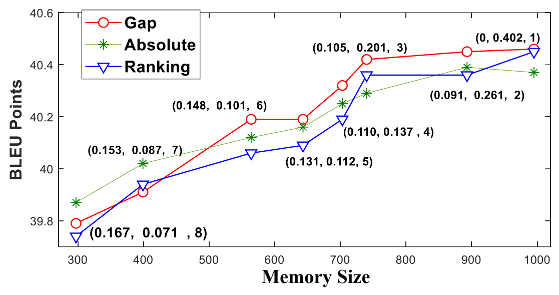

Figure 6: The comparison of different criteria. The gap criterion outperforms others with the increase of the memory size.

We denote the system by **All+Context** and the results are reported in Table 2. As shown in Table 2, **All+Context** surpasses **Arthur(test)** with 0.75 BLEU points while at the cost of 6.4G memory footprint and 1.829s time consuming. However, if we only build the contextual memory for the troublesome words, comparing to **All+Context**, there is only a slighter BLEU points decline (40.19 vs. 40.23) while sharply reduces memory size to 893M and decoding time to 0.511s, showing that our strategy of only building the contextual memory for troublesome words is effective.

The final difference is that we employ the dynamic strategy to balance between NMT and the contextual memory. When we employ this dynamic strategy (denoted by **+Dynamic**), the improvement can further reach 1.37 BLEU points.

Retrained Baseline. In the second comparison, we allow the baseline model to be retrained by Arthur's method (**Arthur(train+test)**). We then implement our method using **Arthur(train+test)** as baseline (denoted by **Arthur(train+test)+MEM**). Comparing the results of these two methods in Table 1 (line 3 and 6), our method is still effective on the retrained model. The average gains are 0.92 BLEU points.

5.4 Effects of Different Exception Criteria

In our method, we investigate three exception criteria to define the troublesome words. The following experiment is conducted to compare their performances. For fairness, the comparison of the three criteria is conducted under the same number of contextual memory, which can be achieved by adjusting the respective thresholds (p_0, g_0 and $rank_0$). The results are reported in Fig. 6, in which the x axis represents the size of contextual memory, the y axis denotes BLEU score, and the numbers in the bracket from left to right are the respective thresholds of **gap**, **absolute** and **ranking**. As shown in Fig. 6, all the three criteria can improve the translation quality. When the memory size is relatively small, absolute criterion performs best. With the size increases, the gap criterion achieves a higher performance than others.

Note that our current criteria only consider one single factor. The combination of different criteria may be more beneficial, and we leave this as our future work.

Type	Baseline	Baseline+MEM
Low+Amb	38.15	39.75
Low	38.56	40.03
Amb	39.86	40.67
Others	40.49	40.76

Table 3: The BLEU score on different kinds of sentences. **Low** denotes low frequency words, **Amb** denotes ambiguous words. **Low+Amb** denotes the low frequency and ambiguous words.

Type	Tword	Error	Rectify	Deterio
Total	374	153	70 (45.8%)	11
Low+Amb	77	30	16 (53.3%)	3
Low	144	59	30 (50.8%)	4
Amb	117	48	20 (41.7%)	4
Others	36	16	4 (25%)	0

Table 4: The manual analysis on the word level. Column **Tword** shows the number of troublesome words in sentence. Column **Error** shows the number of errors made by **Baseline** when translates the troublesome words. The number (ratio) of rectification caused by our method is reported in column **Rectify**. Column **Deterio** shows the number of deterioration (the original translation is correct, while our method produces the incorrect translation) caused by our method.

5.5 Results on Low-Frequency Words and Ambiguous Words

We further analyze our method on specific troublesome words, such as low-frequency words and ambiguous words. Here, we use the following definition in our analysis.

Low-frequency words: The words whose frequency is lower than 100.

Ambiguous words: Assume a word s_m contains K candidate translations with a probability p_k^L. If the entropy of probability distribution $-\sum_{k=1}^{K} p_k^L log p_k^L > E_0$ ($E_0 = 1.5$ in this paper), we treat this word as an ambiguous word.

Therefore, the sentences containing troublesome words can be divided into four different parts: 1) sentences which contain both low-frequency and ambiguous words (**Low+Amb**,

Model	Rectify	Deterio
Arthur(test)	51	17
Baseline+MEM	70	11

Table 5: The numbers of rectification (**Rectify**) and deterioration (**Deterio**) caused by different models.

986 sentences), 2) sentences which contain low-frequency words but no ambiguous words (**Low**, 1427 sentences). 3) sentences which contain ambiguous words while do not contain low-frequency words (**Amb**, 1301 sentences). 4) Other sentences (**Others**, 832 sentences). The results are reported in Table 3. From this table, we observe that our proposed method improves the translation quality on all kinds of sentences. **Low+Amb** performs best (**Low** the second), indicating that our method is most effective in dealing with low-frequency words. The improvement on **Amb** is 0.81 BLEU points, showing that our method can also well handle the ambiguous words.

We also conduct a manual analysis to figure out how many troublesome words could be rectified by our method. We randomly select 200 testing sentences, and count the following three numbers: 1) the number of troublesome words in the sentence (**Tword**), 2) the number of mistakes produced by **Baseline** (**Error**), 3) the number (ratio) of rectification using our method (**Rectify**). 4) The number of deterioration caused by our method (**Deterio**). The statistics are reported in Table 4. From the results, we can get similar conclusions that our method is most effective on low-frequency and ambiguous words with the rectification rate 50.8% and 41.7% respectively.

We can notice that the proposed method also produces 11 deterioration cases (**Deterio**) when rectifying the troublesome words. As a comparison, we also count the total rectification and deterioration numbers of **Arthur(test)**. The results are reported in Table 5. These results show that our method could rectify more words (51 vs. 70) with less deterioration (17 vs. 11) than **Arthur(test)**.

6 Results on EN-DE Translation

We also test our method on EN-DE translation and the results are reported in Table 6. We can see that our method is still effective on EN-DE translation. Specifically, when the translation unit is word, the proposed method improves the baseline by 1.13 BLEU points. The improvement is 0.76 BLEU points when the translation unit is sub-word.

Model	Unit	EN-DE	
		dev	test
Baseline	word	20.28	21.04
Baseline+MEM	word	21.34†	22.17†
Baseline	sub-word	22.10	22.85
Baseline+MEM	sub-word	23.05†	23.61*

Table 6: The results on EN-DE translation. "*" indicates that it is statistically significantly better ($p < 0.05$) than system X and "†" indicates $p < 0.01$.

7 Related Work

The related work can be divided into three categories and we describe each of them as follows:

Neural Turing Machine for NMT. Our idea is first inspired by the Neural Turing Machine (NTM) (Graves et al., 2014, 2016) and memory network (Weston et al., 2014). (Wang et al., 2017a) used special NTM memory to extend the decoder in the attention-based NMT. In their method, the memory is used to provide temporary information from source to assist the decoding process. In contrast, our work uses memory to store contextual knowledge in the training data.

Smaller translation granularity. Our work is also inspired by the other studies to deal with the low-frequency and ambiguous words (Vickrey et al., 2005; Zhai et al., 2013; Rios et al., 2017; Carpuat and Wu, 2007; Li et al., 2016). Among them, the most relevant is the work that decomposes the low-frequency words into smaller granularities, e.g, hybrid word-character model (Luong and Manning, 2016), sub-word model (Sennrich et al., 2016) or word piece model (Wu et al., 2016). These methods mainly focus on low-frequency words that are just a subset of the troublesome words. Furthermore, our experimental results show that even using a smaller translation unit, the NMT model still faces the problem of troublesome tokens and our method could alleviate this problem.

Combining SMT and NMT. Our ideas are also inspired by the work which combines SMT and NMT. Earlier studies were mostly based on the SMT framework, and have been deeply discussed by the review paper in Zhang and Zong (2015). Later, the researchers transfer to NMT framework, e.g. (Wang et al., 2017b; Zhang and Zong, 2016; Zhou et al., 2017; Tu et al., 2016; Mi et al., 2016; He et al., 2016; Dahlmann et al., 2017; Wang et al., 2017c,d; Gu et al., 2018; Zhao et al., 2018). The most relevant studies are Arthur et al. (2016) and

Feng et al. (2017). They incorporate the lexicon pairs into NMT to improve the translation quality. There are three differences between our method and theirs. First, we only utilize the lexicon pairs for the troublesome words, rather than using all lexicon pairs. Second, we take contextual information into consideration for memory construction. Third, we design a dynamic strategy to balance the memory and NMT. The experiments show the superiority of our proposed methods.

8 Conclusions

To address troublesome words in NMT, we have proposed a novel memory-enhanced framework. We first define and detect the troublesome words, then construct a contextual memory to store the translation knowledge and finally access the contextual memory dynamically to correctly translate the troublesome words. The extensive experiments on Chinese-to-English and English-to-German translation tasks demonstrate that our method significantly outperforms the strong baseline models in translation quality, especially in handling the troublesome words.

Acknowledgments

The research work described in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2016QY02D0303 and the Natural Science Foundation of China under Grant No. 61333018. The research work in this paper also has been supported by Beijing Advanced Innovation for Language Resources of Beijing Language and Culture University.

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. *In Proceedings of EMNLP 2016*, pages 1557–1567.

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

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *In proceedings of EMNLP 2007*, pages 61–72.

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. *In Proceedings of EMNLP 2014*, pages 1724–1734.

Leonard Dahlmann, Evgeny Matusov, Pavel Petrushkov, and Shahram Khadivi. 2017. Neural machine translation leveraging phrase-based models in a hybrid search. *In proceedings of EMNLP 2017*, pages 1422–1431.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *In proceedings of NAACL 2013*.

Yang Feng, Shiyue Zhang, Andi Zhang, Dong Wang, and Andrew Abel. 2017. Memory-augmented neural machine translation. *In proceedings of EMNLP 2017*, pages 1401–1410.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1601.03317*.

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

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. 2018. Search engine guided nonparametric neural machine translation. *In proceedings of AAAI 2018*.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with smt features. *In Proceedings of AAAI 2016*.

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

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. *In Proceedings of ACL 2015*, pages 124–129.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? a case study on 30 translation directions. *arXiv preprint arXiv:1610.01108*.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown word in neural machine translation. *In Proceedings of IJCAI 2016*, pages 2852–2858.

Minh Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *In proceedings of EMNLP 2016*, pages 1054–1063.

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

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding model for neural machine translation. *In Proceedings of EMNLP 2016*, pages 955–960.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *In Proceedings of ACL 2002*, pages 311–318.

Annette Rios, Laura Mascarell, and Rico Sennrich. 2017. Improving word sense disambiguation in neural machine translation with sense embeddings. In *In proceedings of WMT 2017*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *In Proceedings of ACL 2016*, pages 1715–1725.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. *In Proceedings of ACL 2016*, pages 76–85.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and ukasz Kaiser. 2017. Attention is all you need. *arXiv preprint arXiv:1601.03317*.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word-sense disambiguation for machine translation. In *In proceedings of ACL 2005*, pages 771–778.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2017a. Memory-enhanced decoder for neural machine translation. *In proceedings of EMNLP 2017*, pages 278–286.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017b. Neural machine translation advised by statistical machine translation. *In proceedings of AAAI 2017*.

Xing Wang, Zhaopeng Tu, Deyi Xiong, and Min Zhang. 2017c. Translating phrases in neural machine translation. *In proceedings of EMNLP 2017*, pages 1432–1442.

Yining Wang, Yang Zhao, Jiajun Zhang, Chengqing Zong, and Zhengshan Xue. 2017d. Towards neural machine translation with partially aligned corpora. *In proceedings of IJCNLP 2017*.

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

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Feifei Zhai, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2013. Handling ambiguities of bilingual predicate-argument structures for statistical machine translation. In *In proceedings of ACL 2013*, pages 1127–1136.

Jiajun Zhang and Chengqing Zong. 2015. Deep neural networks in machine translation: An overview. *IEEE Intelligent Systems*, 30(5):16–25.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. *In Proceedings of EMNLP 2016*, pages 1535–1545.

Yang Zhao, Yining Wang, Jiajun Zhang, and Chengqing Zong. 2018. Phrase table as recommendation memory for neural machine translation. *In proceedings of IJCAI 2018*, pages 4609–4615.

Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. 2017. Neural system combination for machine translation. *In Proceedings of ACL 2017*, pages 378–384.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *In Proceedings of NAACL 2016*, pages 30–34.