Learning to Refine Source Representations for Neural Machine Translation

Xinwei Geng∗ • Longyue Wang∗ • Xing Wang♥ • Bing Qin♥ • Ting Liu♥ • Zhaopeng Tu♥
♥Harbin Institute of Technology  ♥{xwgeng, qinb, tliu}@ir.hit.edu.cn
♠Tencent AI Lab  ♠{vinnylywang, brightxwang, zptu}@tencent.com

Abstract
Neural machine translation (NMT) models generally adopt an encoder-decoder architecture for modeling the entire translation process. The encoder summarizes the representation of input sentence from scratch, which is potentially a problem if the sentence is ambiguous. When translating a text, humans often create an initial understanding of the source sentence and then incrementally refine it along the translation on the target side. Starting from this intuition, we propose a novel encoder-refiner-decoder framework, which dynamically refines the source representations based on the generated target-side information at each decoding step. Since the refining operations are time-consuming, we propose a strategy, leveraging the power of reinforcement learning models, to decide when to refine at specific decoding steps. Experimental results on both Chinese–English and English–German translation tasks show that the proposed approach significantly and consistently improves translation performance over the standard encoder-decoder framework. Furthermore, when refining strategy is applied, results still show reasonable improvement over the baseline without much decrease in decoding speed.

1 Introduction
The encoder-decoder framework has achieved promising progress in sequence generation tasks including dialog system [Vinyals and Le, 2015; Li et al., 2017], question answering [Xiong et al., 2016; Chen et al., 2017] as well as machine translation [Sutskever et al., 2014; Bahdanau et al., 2015]. In neural machine translation (NMT), the encoder summarizes the source sentence into a vector representation, and the decoder generates the target sentence word-by-word from the vector representation. Besides, the attention mechanism [Bahdanau et al., 2015] dynamically select parts of the source representations according to its relevance to the next target word.

∗Work done when interning at Tencent AI Lab.

Figure 1: Overview of the encoder-refiner-decoder NMT framework. The newly introduced refiner dynamically updates the source representations based on the generated target-side information at each decoding step.

However, the encoder summarizes the representation of input sentence from scratch, which is potentially a problem if the sentence is ambiguous. Like a human translator, the encoding process is analogous to reading a sentence in the source language and summarizing its meanings (i.e., source representations) for generating the equivalents in the target language. When humans translate some complex sentences, they generally create an initial understanding on the source sentence (which may be wrong), and then incrementally refine the understanding based on the partial translation they have generated [Hayes and Flower, 1986]. As seen, it is difficult even for humans to translate text based on unchanged understanding in a single pass.

Inspired by human translation process, we propose a novel translation model, namely encoder-refiner-decoder, as illustrated in Figure 1. As seen, we introduce an additional refiner to dynamically refine the source representations by considering the target-side context information at each decoding time step. Specifically, the refiner consists of a gate that reads the target-side hidden state, the output of which is fed to a separate re-encoder to refine the source representations. Since greedily refining source representations at every decoding step is time-consuming, we propose a conditional refining strategy, which adopts an auxiliary policy network trained by
reinforcement learning to decide whether to perform the refine operation at each decoding step.

Experiments on Chinese–English and English–German corpora show that the proposed approach significantly improves translation performance by refining source representations for NMT. Furthermore, when refining strategy is applied, it alleviates the decoding speed problem by cutting down unnecessary refining operations. Results on Chinese–English translation task show improvement over NMT with unnecessary refining operations. Experiments for English–German translation task show significant improvement of +1.23 BLEU points, demonstrating the potential universality of the proposed approach across language pairs.

Contributions Our main contributions can be summarized as follows:

1. We proposed a novel encoder-refiner-decoder framework to produce target-aware source representations for improving NMT models;
2. We introduce a policy network to reduce refining computations, making the approach highly practical, for example for translation in industry applications;
3. We find our approach performs especially better on long sentences, which are generally complex and thus hard to be translated by NMT model. This finding confirms our claim that the introduced refiner can produce a better understanding of complex sentences.

2 Background

Suppose that \( x = x_1, \ldots, x_J \) represents a source sentence and \( y = y_1, \ldots, y_I \) a target sentence. NMT directly models the probability of translation from the source sentence to the target sentence word by word:

\[
P(y|x) = \prod_{i=1}^{I} P(y_i|y_{<i}, x; \theta) \tag{1}
\]

where \( \theta \) is a set of model parameters and \( y_{<i} = y_1, \ldots, y_{i-1} \) denotes a partial translation before the position \( i \).

The encoder-decoder architecture is now widely employed, where the encoder summarizes the source sentence \( x \) into a sequence of hidden states \( h = h_1, \ldots, h_J \) where \( h_j \) is the hidden state of the \( j \)-th source word \( x_j \), as in:

\[
h = \text{encoder}(x_1, \ldots, x_J) \tag{2}
\]

in which \( \text{encoder} \) is an encoding function to generate the \( J \) sequence of hidden states given all the related inputs. It can be either a Recurrent Neural Network (RNN) [Hochreiter and Schmidhuber, 1997], Cho et al., 2014] or Convolutional Neural Network (CNN) [Gehring et al., 2017] or Self-Attention Network (SAN) [Vaswani et al., 2017]. Next, the decoder generates each target word \( y_i \) based on source context \( c_i \), target context \( s' \) and previously generated word(s) \( y' \), as follows:

\[
s_i = \text{decoder}(y', s', c_i) \tag{3}
\]

where \( \text{decoder} \) is a decoding function to dynamically generate the decoder state \( s_i \). The target word \( y_i \) is generated given all the related inputs with non-linear activation function. Similar to \( \text{encoder} \), it can be either a RNN, CNN or SAN. Besides, \( s' = s_{i-1} \) and \( y' = y_{i-1} \) in RNN search models, otherwise \( s' = s_{<i} \) and \( y' = y_{<i} \). \( c_i \) is calculated by attention mechanism based on the source representations.

Specifically, the source representations stand for source context, which embed the information (e.g., syntax, semantics etc.) from the source sentence. However, in the traditional framework, the source representations always remain fixed during the whole decoding time steps regardless of target context. This actually adopts a greedy way to summarize excessive information (including relevant and irrelevant information) for generating each target word. Consequently, NMT needs to spend a substantial amount of its capacity in disambiguating source and target words based on the source context [Choi et al., 2017].

Therefore, it is necessary to take the target context into consideration to dynamically generate the target-aware source representations. Ideally, we expect the refined representations contain the information relevant to current target word, filtering irrelevant one.

3 Approach

3.1 Refiner

As shown in Figure 2, the presented encoder-refiner-decoder framework literally consists of three major components: the standard \( \text{encoder}, \text{decoder} \) as well as the additional \( \text{refiner} \). The basic idea of our approach is to dynamically refine source representation by using target context, and then use the decoder-sensitive representations as new source context for generating each target word. With the auxiliary context, we aim to encourage the source-side latent representations to embed dynamic target-side information, and thus generate better translation with enhanced representations.

Shallow Re-Understanding Originally, the attention mechanism is used to selectively summarize certain parts of source-side information. However, the source representations are fixed and thus treated universally for each target word. Accordingly, there may exist duplicated/useless information, which is irrelevant to the target word at current decoding step. From this observation, we introduce a context gate [Tu et al., 2017a] to dynamically control the amount of source representations used for generating the next target word at each time step.

At time step \( i \), the refiner reads the previous hidden state of decoder (e.g., \( s_{i-1} \)) and a sequence of hidden states from standard encoder (i.e., \( h \)), and then refined hidden states (i.e., \( \tilde{h}^i \)), which store decoder-sensitive information such as translated and untranslated contents [Zheng et al., 2018]. We introduce the context gate that consists of a sigmoid neural network layer and an element-wise multiplication operation. Each gate unit takes \( h_j \) and \( s_{i-1} \) to compute the gate vector \( z_j^i \), and assigns an element-wise weight to \( h_j \), computed by

\[
z_j^i = \sigma(W^z h_j + U^z s_{i-1} + b_z) \tag{4}
\]

\[
\tilde{h}^i_j = z_j^i \odot h_j \tag{5}
\]
where $\sigma(\cdot)$ is a sigmoid activation function and $\odot$ is element-wise multiplication. $W_z$ and $U_z$ are the weight matrices, and $b_z$ is the bias vector. These parameters are trained to learn how to refine source representation to maximize the overall translation performance. As results, we obtain the refined source representations $\hat{h}^i$, which are associated with the decoder time step $i$.

**Deep Re-understanding** After adopt the target-side context into source representations via context gate, we then deeply re-understand them by an additional encoder. We expect this process can enhance the ability of distinguishing different translation predictions. Given the tailored representations $\hat{H}^i = \hat{h}_1^i, \ldots, \hat{h}_j^i, \ldots, \hat{h}_j^i$, we re-encode them to generate the refined representations $\hat{h}_j^i$, as follows:

$$\hat{h}_j^i = encoder_{re}(\hat{h}_1^i, \ldots, \hat{h}_j^i, \ldots, \hat{h}_j^i)$$ (6)

where $encoder_{re}$ is an encoding function, which is similar to $encoder$ in Equation (3) with different parameters. Furthermore, $\hat{h}^i$ are used as a better source context for the decoder to generate the target word $y_i$, using the attention model. In experiment section, we verify the idea.

### 3.2 Conditional Refining Strategy

**Definition** The direct strategy is to process greedy refining at each decoder time step (i.e., $1, \ldots, l$). However, it is very time-consuming due to additional gating and re-encoding operations. One observation is that humans dynamically refresh the understanding of source sentence only after translating a complete semantic unit (e.g., notional word, phrase, clause et al.). It is unnecessary to refine source representations at each decoder time step and some refining operations may be redundancy. Therefore, instead of greedy refining, we propose a conditional refining strategy to learn when to process refining.

Formally, at the decoder time step $i$, we parameterize the possible actions $a_i \in \{\text{REFINE, REUSE}\}$ with an auxiliary policy network, where REFINE indicates processing gating and re-encoding to refine representations (i.e., $\hat{h}^{i+1}$) while REUSE represents skipping refining by reusing the refined source representations at previous time step (i.e., $\hat{h}^i$). We employ two-layer feed forward network to calculate the policy as follows:

$$\pi(a_i|m_i) = \text{softmax}(W_pm_i + b_p)$$ (7)

where $W_p$ is a weight matrix and $b_p$ is a bias vector. $m_i$ is a representation of current policy state, which is computed as:

$$m_i = \tanh (W_p[s_i; E_{y_{i-1}}; c_i] + b_p)$$ (8)

in which $s_i$ is the current decoder state, $E_{y_{i-1}}$ is the embedding of previous target word, and $c_i$ is the context vector. Note that our policy network makes the decision for the next decoding step rather than the current.

**Reinforcement Learning** Towards utilizing the discrete variables into the network, we employ the Gumbel-Softmax [Jang et al., 2016; Maddison et al., 2016] to approximate the one-hot vectors sampled from the categorical distribution with continuous representations. Using the reparameterization trick, the standard backpropagation can be utilized to compute the policy gradients of model parameters for reinforcement learning. As a result, the sample $\hat{\alpha}_i = (\hat{\alpha}_1^i, \ldots, \hat{\alpha}_n^i) \in \mathbb{R}^K$ can be approximated using Gumbel-Softmax as follows:

$$\hat{\alpha}_i^k = \frac{\exp((\hat{o}_i^k + g_i^k)/\tau)}{\sum_{k' = 1}^K \exp((\hat{o}_i^{k'} + g_i^{k'}/\tau))}$$ (9)

$$g_i^k = -\log(-\log(u_i^k))$$ (10)

$$u_i^k \sim \text{Uniform}(0, 1)$$ (11)

where $o_i$ is the unnormalized probability of softmax, $g_i^k$ is Gumbel noise [Gumbel, 1954], and $\tau \in (0, \infty)$ is a temperature parameter. The softmax function approaches argmax function as $\tau \to 0$, whereas it becomes uniform as $\tau \to \infty$.

In order to discretize the continuous probability $\hat{\alpha}_i$, we apply the straight-through version of Gumbel-Softmax, named Straight-Through (ST) Gumbel-Softmax [Jang et al., 2016]. During the forward phase, we use the Gumbel-Max trick, while computing the gradient with the continuous $\hat{\alpha}_i$. Given the continuous probability $\hat{\alpha}_i$ sampled from the Gumbel-Softmax, the discrete one-hot vector $\hat{a}_i = (\hat{a}_1^i, \ldots, \hat{a}_n^i)$ are calculated as follows:

$$\hat{a}_i^k = \begin{cases} 1 & k = \arg \max_{k'} \hat{a}_i^{k'} \\ 0 & \text{otherwise} \end{cases}$$ (12)
Using the above Gumbel-Softmax trick, at the decoder time step \(i + 1\), the choice on using the previous context-aware representations \(\hat{h}^i\) or the refined output \(\hat{h}^{i+1}\) can be formalized as follows:

\[
\hat{h}^{i+1} = \tilde{a}^i_{\text{REUSE}} \hat{h}^i + \tilde{a}^i_{\text{REFINE}} \hat{h}^{i+1}
\]

where \(\hat{h}^{i+1}\) is the final encoding representation for attention mechanism, \(\tilde{a}^i_{\text{REUSE}}\) and \(\tilde{a}^i_{\text{REFINE}}\) is the corresponding elements for the choices REUSE and REFINE in the discrete one-hot vector \(\hat{a}^i\).

Furthermore, we impose a constraint on the ratio of REFINe operations to the total decoding length, encouraging the model to re-use the previous context-specific representations as much as possible. As a remedy, we add a small penalty \(r(\hat{a})\) the model pay for choosing the REFINe operation as follows:

\[
r(\hat{a}) = \alpha \left( \sum_{i=1}^{I} \tilde{a}^i_{\text{REFINE}} \right) / I
\]

where \(\alpha\) is hyper-parameter which controls the strength of penalty.

4 Experiments

4.1 Data

We carried out experiments on Chinese–English translation task. We used the corpus consisting of 1.25M bilingual sentence pairs extracted from LDC corpora.\[^1\] We used the NIST 2002 (MT02) as the tuning set for hyper-parameter optimization and model selection, and NIST 2003 (MT03), 2004 (MT04), 2005 (MT05), 2006 (MT06) and 2008 (MT08) as test sets. As seen, it totally contains 27.9M Chinese words and 34.5M English words. As most sentences in the corpus is in newswire domain, the average length is relatively longer than other informal domains (e.g., 5.63 and 7.71 in subtitle corpus), which makes translation difficult.

We used case-insensitive 4-gram NIST BLEU metrics \[^2\] as calculated by the multi-bleu.perl script for evaluation, and sign-test \[^3\] script to test for statistical significance.

4.2 Model

The baseline is our re-implemented attention-based NMT system, which incorporates dropout \[^4\] on the output layer and improves the attention model by feeding the most recently generated word. For training the translation models, we limited the source and target vocabularies to the most frequent 30K words in Chinese and English, covering approximately 97.2% and 99.3% of the words in the two languages, respectively. Our models were trained on sentences of length up to a maximum of 50 words with early stopping. Mini-batches were shuffled during processing with a mini-batch size of 80. The word-embedding dimension was 620 and the hidden layer size was 1,000. We set learning rate as \(5 \times 10^{-4}\), gradient norm as 1.0 and dropout rate as 0.3. We applied Rmsprop \[^5\] to train models for 10 epochs and selected the model parameters that yielded best performances on the tuning set. The beam size is set as 10. The proposed model was implemented on top of the baseline model with the same settings where applicable. The hidden layer size in the refiner was 1,000.

4.3 Results and Discussion

We evaluated the presented approaches in terms of translation quality, speed and robustness.

Translation Quality Table 1 shows translation performances for Chinese–English. The two baseline NMT models, one being trained with the standard NMT system (i.e., “Baseline”), while the other was trained with multi-layer encoders (i.e., “+ Multi-layer”). Benefiting from the deeper layers, the stronger baseline system is able to improve performance over the standard baseline system by + 0.29 BLEU point.

Clearly the proposed models significantly improve the translation quality in all cases, although there are still considerable differences among different variants. Introducing context gate for refining source representations (i.e., “Shallow Refiner”) improves translation performance over “Baseline” by + 1.02 BLEU points. It demonstrates the effectiveness of our proposed refiner model over the baseline model. Furthermore, adding re-encoding (i.e., “Deep Refiner”) together achieves the best performance overall, which is + 2.34 BLEU points better than the baseline model. This confirms our assumption that re-encoder applied to the shallow-refined source representations indeed help to re-understand the deeper semantics of source sentence. The “Conditional Strategy” shows the the policy network can skip 41% unnecessary refining operations (as illustrated in Table 2) but still keep reasonable translation performances (i.e., around +1.83 BLEU points than “Baseline”). Note that we can control the percent of refining operations depending on the requirement via the hyper-parameter \(\alpha\) and threshold value of choosing the refining operation. In this work, we choose appropriate \(\alpha\) and threshold value to report the corresponding result.

Parameters In terms of additional parameters introduced by the refining models, both shallow and deep refiners introduce a large number of parameters. Beginning with the baseline model’s 86.69M parameters, the “+ Shallow Refiner” adds +6.00M new parameters, while the “+ Deep Refiner” adds a further +18.01M new parameters with an additional encoder layer. For fair comparison, our “+ Multi-layer” also adds +18.01M new parameters by adding same encoder layer over “Baseline” as a stronger baseline. Besides, “+ Conditional” needs further +2.25M new parameters to learn the decision of skipping refining.

Speed More parameters may capture more information, at the cost of posing difficulties to training. Although gains are made in terms of translation quality by introducing refining, we need to consider the potential trade-off with respect to a possible increase in time consumption, due to the large number of newly introduced parameters resulting from the incorporation of context gate and additional encoder into the NMT model. As shown in Table 2 when running on a single GPU.

\[^1\] The training corpus includes LDC2002E18, LDC2003E07, LDC2003E14, part of LDC2004T07, LDC2004T08 and LDC2005T06.

\[^2\] https://github.com/moses-smt/mosesdecoder
device Tesla P40, the decoding speed of “Baseline” is 558 target words per second, and this reduces to 499 words per second with a slight decrease when context gate is added. With the introduction of additional encoder, the decoding speed has a drastic decrease to 132 target words per second. In terms of decoding time trade-off, our conditional refining model increases decoding speed by 33.33% over the “+ Deep Refiner”. We attribute this to the fact that no re-encoding for each step, which avoids high computation consumption. Taking the time consumption into consideration, our “+ Shallow Refiner” can be utilized in online decoding scenario while “+ Deep Refiner” is more appropriate for offline translation. Further, we can leverage the conditional strategy to balance the decoding speed and translation performance.

**English–German Translation Task** To validate the robustness of our approach on other language pairs, we conducted experiments on WMT2014 English-German corpus, which contains 4.5M bilingual sentence pairs with 116M English words and 110M German words. We use news-test2013 as tuning set and the news-test2014 as test set. Particularly, we segmented words via byte pair encoding (BPE) [Sennrich et al., 2016]. We consider a joint source and target byte-pair encoding with 32K types. We set beam size as 4 in our work. We used the same settings as used in Chinese–English experiments. As shown in Table 3 our proposed “+ Shallow Refiner” and “+ Deep Refiner” also significantly improves translation performance on the English–German task, demonstrating the efficiency and universality of the proposed approach.

| Model                 | MT02  | MT03  | MT04  | MT05  | MT06  | MT08  | Ave.  | △    |
|-----------------------|-------|-------|-------|-------|-------|-------|-------|------|
| Baseline              | 40.16 | 37.26 | 40.50 | 36.67 | 37.10 | 28.54 | 36.01 | –    |
| + Multi-layer         | 40.18 | 37.30 | 40.75 | 37.43 | 37.29 | 28.72 | 36.30 | +0.29|
| + Shallow Refiner     | 41.08 | 38.12 | 41.35 | 38.51 | 37.85 | 29.32 | 37.03 | +1.02|
| + Deep Refiner        | 41.58 | 39.23 | 42.72 | 39.90 | 39.24 | 30.68 | 38.35 | +2.34|
| + Conditional         | 41.02 | 38.73 | 42.34 | 39.23 | 38.92 | 29.97 | 37.84 | +1.83|

Table 1: Evaluation of translation performance for Chinese–English. “Baseline” is trained with standard NMT model, while “+ Multi-layer” is trained with an additional encoder layer. “+ Shallow Refiner” indicates adding context gate for refining source representations, and “+ Deep Refiner” denotes processing re-encoding on refined representations. “+ Conditional” indicates the conditional refining strategy to decide when to refine the source representations. “△” column denotes performance improvements over “Baseline”. “†” indicate statistically significant difference (p < 0.01) from baseline.

| Model     | # Params | Speed | P % |
|-----------|----------|-------|-----|
| Baseline  | 86.69M   | 558   | –   |
| + Multi-layer | 104.70M | 516   | –   |
| + Shallow Refiner | 92.69M  | 499   | 100 |
| + Deep Refiner    | 110.70M | 132   | 100 |
| + Conditional  | 112.95M | 176   | 59  |

Table 2: Decoding speed on Chinese–English translation task. “Params” denotes the number of parameters. “Speed” is measured in words/second. “P” indicates the percentage of refining operations over the translated words.

| Model     | news-2013 | news-2014 | △    |
|-----------|------------|------------|------|
| Baseline  | 22.35      | 22.33      | –    |
| + Multi-layer | 22.32     | 22.39      | +0.06|
| + Shallow Refiner | 22.95 | 23.02†      | +0.69|
| + Deep Refiner     | 23.20     | 23.56†     | +1.23|
| + Conditional  | 23.06     | 23.27†     | +0.94|

Table 3: Translation performance on English–German.

**4.4 Analysis**

**Length Analysis** We follow Tu et al. [2016] to group sentences with similar lengths together. As shown in Figure 3 our “+ Shallow Refiner” and “+ Deep Refiner” substantially outperform the “Baseline” on each length span. Moreover, “+ Deep Refiner” also makes the remarkable improvement over “+ Shallow Refiner” on the entire length segments. More importantly, we discover that the increment percent of “+ Shallow Refiner” and “+ Deep Refiner” grows drastically, as the length of source sentences rises. Specifically, the “+ Shallow Refiner” on the length span (e.g., ≥ 45) increases 3.64% BLEU over the “Baseline”. As a comparison, the “+ Deep Refiner” start with 4.79% increment (e.g., ≥ 45) increases 4.4% consistency between linguistic categories.

**Effects on Linguistic Insights** Towards investigating the distributions of the learned refining policy, we evaluate the consistency between linguistic categories (i.e., chunk) and chunk tags.

---

*We utilized OpenNLP toolkit (https://opennlp.apache.org/) to automatically annotated the translation output with chunk tags.*
the \textsc{Refine} operation. We measured the consistency by calculating the percentage of \textsc{Refine} operations in one corresponding type, and the results are shown in Table 4.

As seen, the \textsc{Refine} operation at the beginning position \textit{(i.e., “B”)} of chunks happens more times than that at other positions \textit{(i.e., “I”).} Taking NP for example, 67.67\% refining happens at the beginning of chunks while 52.60\% at the following positions. This confirms our assumption that there is no need to refine the units inside a semantic component such as phrase and clause. Beside, we found a contrary phenomenon in VPs, where “I\%” is bigger than “B\%”. The reason may be that fictional words that have little lexical meaning often occurs at begging positions such as “has made” and “to severely punish”, and it is unnecessary to process refining on these kinds of words.

**Effect of Gated Refiner** Some researchers may argue the gated refiner computes a scalar as output, rather than the vector as ours. We force the gate refiner to compute the scalar weight as output \textit{(i.e., “Hard Mask”).} As a result, the corresponding “+ Shallow Hard Mask” and “+ Deep Hard Mask” achieves comparable performances with a slight improvement about +0.11 and +0.21 BLEU over “Baseline”, but significant decreases about -0.91 and -2.13 BLEU over our proposed “+ Shallow Refiner” and “+ Deep Refiner”. We conjecture that the above results attribute to the more expressive capability of the vector that scalar output. In our work, the target of context gate assigns the weight to source representations. We assume that the each element in the source representation is taken as a feature, and the output of context gate is the corresponding weight, indicating the importance. With the vector as the weight, we assign large scores for some important features but small for others. Therefore, from this viewpoint, it’s reasonable that our context gate achieves significant improvement over the scalar counterparts.

**Case Study** Towards investigating the refining over source tokens based target partial translation, we visualize the corresponding context gate for each decoding time step. Motivated by visualizing neural model strategy in \cite{Li2015}, the contributions of gating vector to final output can be approximated by first derivatives. At each decoding time step, we compute first derivatives for gating vector with back-propagation to measure saliency score. As illustrated in Figure 4, our model can select relevant important fragments in the source sentence, taking the current decoding context into consideration. For example, when generating the target words “seeks”, the relevant fragments “zhengxun” and “yijian” in source sentence attach great importance. Although the saliency scores of source tokens “dianxinju” and “ip” are extremely low during the generation of the target tokens “ofta” and “ip”, the corresponding gating mechanism prefers to paying no attention over the source sentence. That is, when generating the target prepositions, the information of source sentence is useless for current decoding. Hence, the generation of target prepositions only relies on preceding target context. These results verify the proposal that generation of a content word should rely more on the source context and generation of a functional word should rely more on the target context \cite{Tu2017a}.

### Table 4: Relations between refining policy and chunking categories.

| Type | Overall \% | B \% | I \% |
|------|------------|------|------|
| NP   | 42.28      | 67.67| 52.60|
| PP   | 18.56      | 63.18| 58.82|
| VP   | 15.24      | 62.91| 72.62|
| ADVP | 2.13       | 72.21| 62.96|
| ADJP | 0.90       | 70.29| 61.29|
| Others | 20.89   | 80.80| 66.67|

### Table 5: Evaluation of translation performance when Hard Mask is used as context gate.

| Model                  | BLEU | \(\Delta\) |
|------------------------|------|------------|
| Baseline               | 36.01| -          |
| + Shallow Hard Mask    | 36.12| +0.11      |
| + Deep Hard Mask       | 36.22| +0.21      |
| + Shallow Refiner      | 37.03| +1.02      |
| + Deep Refiner         | 38.35| +2.34      |

Figure 3: Performance of the generated translations with respect to the lengths of the source sentences.

5 Related Work

**Conditional Sequence Processing** More relevant to our work, \cite{Ke2018} present a focused hierarchical RNNs architecture for sequence modelling tasks, which allows them to attend to key parts of the input as needed. Similarly, a discrete gating mechanism is adopted to make a discrete decision on whether or not the token is relevant to current context. Subsequently, the selected tokens are fed into the high RNN layer. Essentially the gating mechanism is hard mask, trained using reinforcement learning. In comparison with their work, our context gate computes a gate vector to mask the corresponding source vector. More importantly, it’s proven that the introduction of the hard mask into our work has no effect.
showing a substantial margin with our context gate in our experimental results. In addition, our model can be optimized using the standard methods instead of reinforcement learning.

**GRU-Gated Attention** Recently, Zhang et al. [2017] proposed a gru-gated attention to consider the decoding context into calculating the context vector with attention mechanism. Similar to our adopted context gate module, [Zhang et al. [2017] also introduces a gating layer to refine the source representations at each decoding time step, based on the current decoding state. Instead of using conventional gating mechanism, a GRU cell is chose to deal with complex interactions between source sentence and partial translation. Furthermore, another important difference from their work is that our work subsequently adopts the additional encoder to further encode the tailored decoding context-aware source representations again, which has been proven its excellent effectiveness in corresponding experimental results.

**Context-Dependent Word Embedding** More than one meaning of a word can be encoded through measuring the multiple dimensions of similarity. Towards explicitly disambiguating source and target words, Choi et al. [2017] propose to contextualize the word embedding vectors using a nonlinear bag-of-words representation of the source sentence. Similarly, we propose to learn to refine the continuous representations generated by standard encoder rather than word embedding to produce the target-side context-aware representations for each decoding time.

**Novel Model Architectures** Tu et al. [2017b] introduced a novel encoder-decoder-reconstructor architecture, which reconstructs decoder states back to the original input sentence. Wang et al. [2018a, 2018b] moved one step further by simultaneously reconstructing encoder states back to the input sentence. Xia et al. [2017] and Zhang et al. [2018] independently introduced a second-pass decoder to polish the raw translation generated by the first-pass decoder. In contrast, we introduced a refiner to polish the encoder states, and thus can be regarded as a second-pass encoder.

### 6 Conclusion and Future Work

This paper is an early attempt to use target context to refine source representations for improving translation performance. As a result, the generated source representations concentrate on most relevant semantic to current target-side context. Furthermore, as a trade-off between efficiency and effectiveness, we further propose to learn when to refine the source representations at each decoding time step. We train the policy network for learning to refine with reinforcement learning. The experimental results on Chinese–English and English–German translation tasks demonstrate the remarkable effectiveness over the standard encoder-decoder architecture. In addition, it’s proven that the excellent performance of our proposed architecture on the long sentences. In future work we plan to adopt a diversity of context information except for target context into our proposed architecture to improve the translation.

### References

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

[Chen et al., 2017] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051, 2017.

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

[Cho et al., 2017] Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. Context-dependent word representation for neural machine translation. Computer Speech & Language, 45:149–160, 2017.

[Collins et al., 2005] Michael Collins, Philipp Koehn, and Ivona Kucerova. Clause restructuring for statistical machine translation. In ACL 2005, pages 531–540, Ann Arbor, Michigan, 2005.

[Gehring et al., 2017] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In ICML, 2017.

[Graves, 2013] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[Gumbel, 1954] Emil Julius Gumbel. Statistical theory of extreme values and some practical applications: a series of lectures. Number 33. US Govt. Print. Office, 1954.
