NAIST-IS-DD1561037

Doctoral Dissertation

Hybrid Neural-Symbolic Machine Translation

Jingyi Zhang

February 14, 2018

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Jingyi Zhang

Thesis Committee:

Professor Satoshi Nakamura	(Supervisor)
Professor Yuji Matsumoto	(Co-supervisor)
Associate Professor Katsuhito Sudoh	(Co-supervisor)
Assistant Professor Graham Neubig	(Carnegie Mellon University)
Dr. Masao Utiyama	(NICT)

Hybrid Neural-Symbolic Machine Translation*

Jingyi Zhang

Abstract

Machine translation (MT) investigates the use of software to translate a source sentence into the corresponding target sentence. The state-of-the-art MT approach is statistical MT, which learns statistical models from a parallel bilingual corpus to model the translation process.

When this thesis started (2015), symbolic MT, which is based on symbolic translation rules, such as phrase-based MT [32], hierarchical phrase-based MT [8], and syntax-based MT [36, 43], had the best translation performance. Because symbolic MT is based on context-free and ambiguous symbolic translation rules, a log-linear framework and statistical models learned from parallel or monolingual corpora, such as language models [32], reordering models [63, 14], rule selection models [35, 10, 23], are exploited to select the most possible translation during symbolic MT decoding.

Now (2018), neural MT (NMT) [4, 55, 68], which is based on a single large neural network and does not use any explicit translation rules, has outperformed symbolic MT on various translation tasks. Compared to symbolic MT, NMT uses distributed word representations and generally produces more fluent translations, but often sacrifices adequacy [31], such as NMT is more likely to generate completely unrelated translations, under and over translations.

This thesis focuses on hybrid neural-symbolic MT methods, which aim to produce translations that are both fluent and adequate by combining the advantages of both symbolic and neural MT.

The first part of our contribution is that we develop various neural models for the log-linear framework of symbolic MT, because neural models learn distributed representations for words and sentences, which can generalize better

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1561037, February 14, 2018.

[66, 54, 17, 73]. In particular, we propose: a neural word reordering model which outperforms previous word reordering models [63] by adopting a feed-forward neural network and selecting useful reordering information (i.e. short reorderings); a binarized neural network joint model (NNJM) which outperforms the original NNJM [11] by converting the multiclass classification problem of the NNJM into a binary classification problem and adopting an effective noise sampling method based on translation probabilities for model training; a neural rule selection model which performs rule selection based on minimal translation rules [16] for reducing data sparsity and outperforms previous rule selection models [35] by learning distributed representations for both translation rules and context words.

The second part of our contribution is that we exploit knowledge from symbolic MT to improve the adequacy of NMT. We propose to perform phrase-based forced decoding for NMT outputs and rerank NMT outputs with phrase-based decoding scores. Because the search space of phrase-based MT is limited by translation rules and the standard phrase-based forced decoding may fail for some NMT outputs, we propose a phrase-based soft forced decoding algorithm, which can successfully find a phrase-based decoding path for any NMT outputs.

Keywords:

hybrid neural-symbolic MT, neural reordering model, binarized neural translation model, neural rule selection model, forced decoding for NMT

Contents

List of Figures

List of Tables

1. Introduction

Machine translation (MT) uses software to translate a source language sentence F into the corresponding target language sentence E, enabling people to understand and communicate with each other even if they do not speak the same language. The state-of-the-art MT approach is statistical MT, which learns statistical models from parallel bilingual corpora to model the translation process.

When this thesis started (2015), symbolic MT, such as phrase-based MT [32], hierarchical phrase-based MT [8], and syntax-based MT [36, 43], had the best translation performance. Symbolic MT is based on context-free and ambiguous symbolic translation rules. During decoding, a large number of translation rules can be applied on one input sentence to generate different translations. To select the most possible translation, symbolic MT exploits a log-linear framework and various statistical features learned from bilingual or monolingual corpora, such as rule selection models [35, 10, 23], reordering models [63, 14] and language models [32].

Now (2018), neural MT (NMT) [61, 4, 55, 68] has outperformed symbolic MT on various translation tasks. NMT is based on a single large neural network, which takes the source sentence as input and outputs the target sentence without using any explicit translation rules. Compared to symbolic MT, NMT learns distributed representations for both source and target words, which generally generates more fluent translations, but often sacrifices adequacy, such as, NMT is more likely to generate completely unrelated translations, over and under translations [31].

This thesis focuses on hybrid neural-symbolic MT methods, which aim to produce translations that are both fluent and adequate by combining the advantages of both symbolic and neural MT. Figure 1 is an overview of our contribution.

The first part of our contribution is that we develop neural models for the log-linear framework of symbolic MT. Traditional features of the log-linear framework in symbolic MT are also based on symbolic representations, such as n-gram language models [32] or maximum entropy based reordering models [63] or rule selection models [35]. Neural models, such as neural language models [66] and translation models [54, 17, 73], which learn distributed representations for words and phrases, have been proved to achieve better performance than previous statistical models that use symbolic representations. In this thesis, we propose three

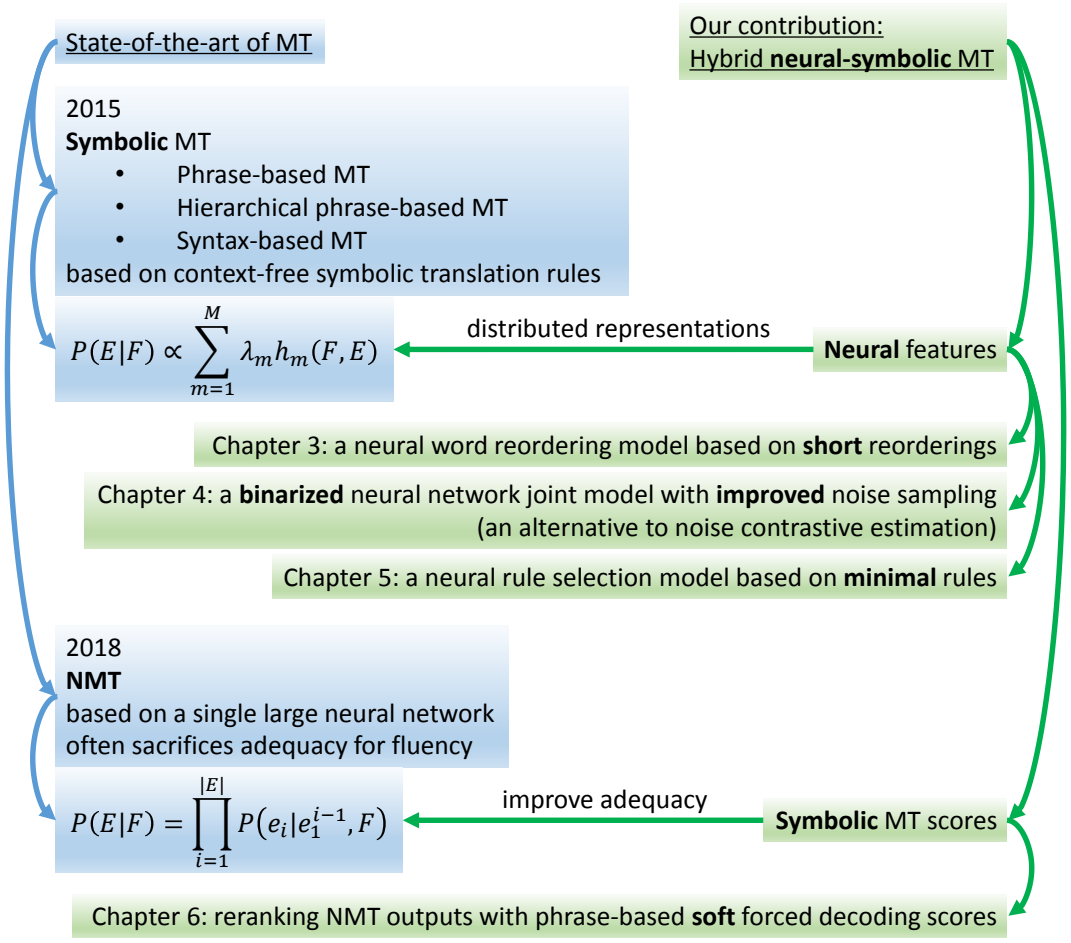

Figure 1. An overview of the state-of-the-art MT methods and our contribution.

novel neural models for symbolic MT as following,

- We propose a neural word reordering model, which is learned by a feed-forward neural network and learns reorderings only for short distance reorderings, because we find long distance reordering information cannot significantly improve the translation performance in our experiments. Compared with previous reordering models [63, 14], our neural reordering model can more effectively and efficiently exploit helpful word reordering information.

- We propose a binarized neural network joint model (NNJM), which converts the multiclass classification problem of the original NNJM [11] into a binary classification problem. The binarized NNJM is an alternative to noise contrastive estimation (NCE) [66] for solving the high normalization cost problem of the NNJM. We also propose an effective sampling method based on translation probabilities for both training the binarized NNJM and training the NNJM with NCE. We show that the binarized NNJM can outperform the original NNJM trained by NCE with the proposed noise sampling method.

- We propose a neural rule selection model for syntax-based MT [36, 43] to perform context-dependent rule selection. Our neural rule selection model learns distributed representations for both translation rules and context words, which outperforms previous maximum entropy based rule selection models [35] that use symbolic representations. In addition, we propose a method to train the rule selection models only on minimal rules [16] to reduce data sparsity, because minimal rules are more frequent and have richer training data compared to non-minimal rules.

The second part of our contribution is that we exploit knowledge from symbolic MT to improve the adequacy of NMT. There are existing methods that incorporate knowledge from symbolic MT into NMT, such as lexical translation probabilities [3, 22], phrase memory [62], and n-gram posterior probabilities based on syntactic translation lattices [59]. These can improve the adequacy of NMT outputs, but do not impose hard alignment constraints like symbolic MT and therefore cannot effectively solve all over-translation or under-translation problems. There are also methods that use both NMT and symbolic MT decoding scores for selecting the best translation [46, 60]. However, the search space of symbolic MT is limited by the translation rule table, compared to NMT which has unlimited search space. Therefore, using both symbolic MT and NMT decoding scores for translation selection also limits the translation search space. We propose a phrase-based soft forced decoding algorithm to solve the limited search space problem of phrase-based MT.

The rest of this thesis is organized as following: Section 2 reviews the background of statistical MT; Section 3, 4 and 5 describe the proposed neural word

reordering model, binarized NNJM and neural rule selection model, respectively; Section 6 describes the proposed phrased-based soft forced decoding method for reranking NMT outputs; We conclude in Section 7 and discuss future work in Section 8.

2. Statistical MT Background

Statistical MT learns statistical translation models from parallel corpora (i.e. sentence-level aligned bilingual corpus). This section reviews different statistical MT approaches.

2.1 Word Alignment

IBM models [6] are widely used to learn word-level alignments for parallel corpus. Figure 2 shows an example of word alignments for an English-to-Chinese sentence pair.

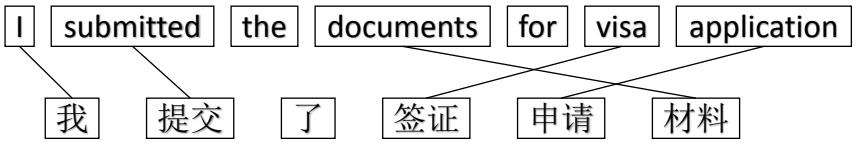

Figure 2. An example of word alignments.

IBM models learn alignments a_1^J for a source sentence f_1^I and a target sentence e_1^J. That is e_j is aligned to f_{a_j}. IBM models consist of five models.

Model 1 Model 1 only uses lexical translation probabilities. The joint likelihood of a target sentence and an alignment given a source sentence is

$$\Pr\left(e_1^J, a_1^J | f_1^I\right) = \frac{\varepsilon}{(I+1)^J} \prod_{j=1}^{J} t\left(e_j | f_{a_j}\right) \tag{1}$$

where $t\left(e|f\right)$ is lexical translation probabilities that satisfy,

$$\sum_e t\left(e|f\right) = 1 \tag{2}$$

Therefore,

$$\Pr\left(e_1^J | f_1^I\right) = \frac{\varepsilon}{(I+1)^J} \sum_{a_1=0}^{I} ... \sum_{a_J=0}^{I} \prod_{j=1}^{J} t\left(e_j | f_{a_j}\right) \tag{3}$$

$a_j = 0$ means e_j is not aligned to any source word. The parameters $t\left(e|f\right)$ are estimated by maximizing $\Pr\left(e_1^J | f_1^I\right)$ on the training data.

Model 2 Model 2 introduces reordering probabilities into the joint likelihood in Equation 1,

$$\Pr\left(e_1^J, a_1^J | f_1^I\right) = \varepsilon \prod_{j=1}^{J} t\left(e_j | f_{a_j}\right) a\left(a_j | j, J, I\right) \tag{4}$$

where $a\left(i|j, J, I\right)$ satisfy,

$$\sum_{i=0}^{I} a\left(i|j, J, I\right) = 1 \tag{5}$$

Model 3-5 introduce more complicated features.

IBM models align each target word to one source word even if one target word should be aligned to multiple source words. To solve this problem, a widely used method is to run bi-directional (source-to-target and target-to-source) IBM models and then use the *grow-diag-final-and* heuristic [32] to combine the bi-directional alignments together for symmetric word alignments.

2.2 Phrase-based MT

Phrase-based MT [32] extracts phrase-based translation rules from the word-aligned training corpus and then uses them to translate new sentences.

A phrase-based translation rule r includes a source phrase, a target phrase and several feature scores, such as direct and inverse lexical and phrase translation probabilities. Table 1 shows some examples of phrase-based translation rules extracted from the word-aligned sentence pair in Figure 2.

documents→材料(documents)
visa→签证(visa)
application→申请(application)
visa application → 签证(visa) 申请(application)
documents for visa application →签证(visa) 申请(application) 材料(documents)

Table 1. Examples of phrase-based rules.

The translation process of phrase-based MT applies phrase-based translation rules to source phrases in the input sentence and then reorders target phrases into the target language word order.

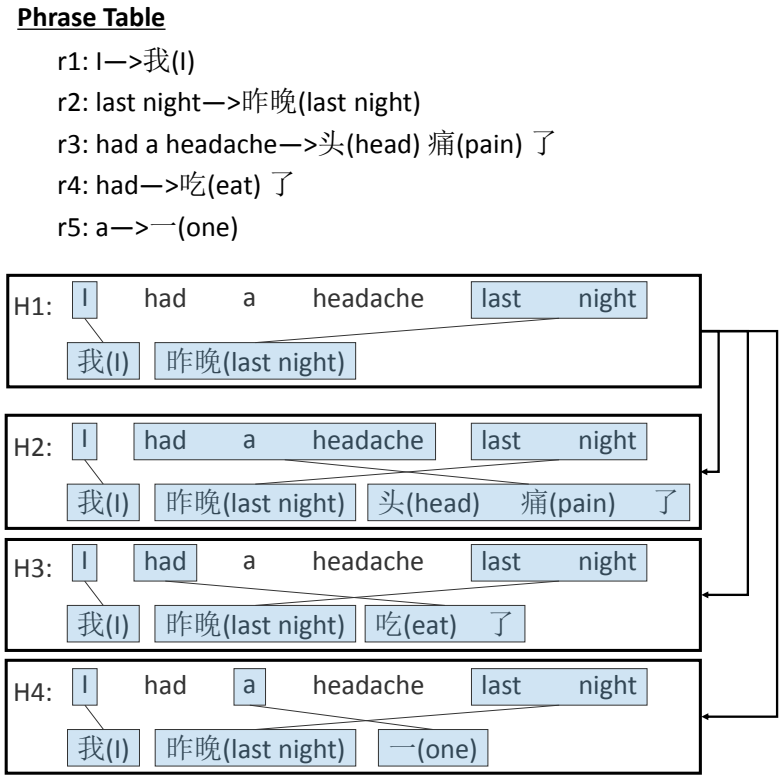

Figure 3. An example of phrase-based decoding.

Because each source phrase can have different translations and target phrases can be reordered arbitrarily, one input sentence has a huge number of possible translations. The phrase-based decoding algorithm aims to find the best translation E with the highest translation probability for a given input sentence F.

$$\hat{E} = \arg\max_{E} \Pr(E|F). \tag{6}$$

The probability of E given F is computed using a log linear model as shown in Equation 7. h_m can be features calculated by various statistical models, such as translation models, reordering models and language models. λ_m are feature

weights.

$$\Pr(E|F) \approx \frac{\exp\left(\sum_{m=1}^{M} \lambda_m h_m(E,F)\right)}{\sum_{E'} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(E',F)\right)}. \quad (7)$$

The standard phrase-based decoding algorithm generates target phrases from left to right by applying a list of translation rules to the input sentence. A basic concept in phrase-based decoding is hypotheses. As shown in Figure 3, the hypothesis H_1 consists of two rules r_1 and r_2. An existing hypothesis can be expanded into a new hypothesis by applying a new rule. As shown in Figure 3, H_1 can be expanded into H_2, H_3 and H_4. H_2 cannot be further expanded, because it covers all source words, while H_3 and H_4 can (and must) be further expanded. The decoder starts with an initial empty hypothesis H_0 and selects the hypothesis with the highest score from all completed hypotheses. The score of a hypothesis $S(H)$ can be computed as the log sum of weighted feature scores.

During decoding, hypotheses are stored in stacks. For a source sentence with I words, the decoder builds I stacks. The hypotheses that cover i source words are stored in stack s_i. The decoder expands hypotheses in $s_1, s_2, ..., s_I$ in turn as shown in Algorithm 1. Here, EXPAND(H) is expanding H to get new hypotheses and putting the new hypotheses into corresponding stacks. For each stack, a beam of the best n hypotheses is kept to speed up the decoding process.

Algorithm 1 Standard phrase-based decoding.

Require: Source sentence F with length I
Ensure: Translation E and decoding path D
  initialize H_0 and $s_1, s_2, ..., s_I$
  EXPAND(H_0)
  for $i = 1$ to $I - 1$ **do**
    for each hypothesis H_{ik} in s_i **do**
      EXPAND(H_{ik})
    end for
  end for
  select best hypothesis in s_I

2.3 Hierarchical Phrase-based MT

Hierarchical phrase-based MT [8] introduces non-terminal X and hierarchical structures into phrase-based MT.

A hierarchical phrase-based translation rule r is defined as:

$$X \rightarrow \left\langle \tilde{f}, \tilde{e}, \sim \right\rangle,$$

where X is a nonterminal, $\tilde{f}$ and $\tilde{e}$ are respectively source and target strings of terminals and nonterminals, and $\sim$ is the alignment between nonterminals and terminals in $\tilde{f}$ and $\tilde{e}$.

Table 2 shows some hierarchical phrase-based rules extracted from the word-aligned sentence pair in Figure 2.

$X \rightarrow$ documents, 材料(documents), 0-0
$X \rightarrow$ application, 申请(application), 0-0
$X \rightarrow$ visa application, 签证(visa) 申请(application), 0-0 1-1
$X \rightarrow$ documents for visa X, 签证(visa) X 材料(documents), 0-2 2-0 3-1
$X \rightarrow$ documents for X, X 材料(documents), 0-1 2-0

Table 2. Examples of hierarchical phrase-based rules.

In hierarchical phrase-based MT, the translation of a new source sentence can be done by applying a series of hierarchical phrase-based rules, which perform translating and reordering at the same time via non-terminal X in contrast to phrase-based MT which does translating and reordering separately.

Like phrase-based MT, hierarchical phrase-based MT can also generate plenty of translations for one input sentence by applying different rules. The standard decoding algorithm used to find the most possible translation in hierarchical phrase-based MT is the bottom-up CKY parsing algorithm with beam search. To compute probabilities for different translation, hierarchical phrase-based MT uses the same log-linear model as phrase-based MT with different statistical features.

2.4 Tree-to-String MT

Tree-to-string MT [36] uses a statistical parser to obtain the parse tree for source sentences and then extract/apply translation rules whose source sides are syntac-

tic subtrees. Figure 4 shows two translation rule examples extracted in tree-to-string MT. These two rules have the same source subtree $\tilde{t}$ but different target translations $\tilde{e}$, which demonstrates the ambiguous rule selection problem in tree-to-string MT.

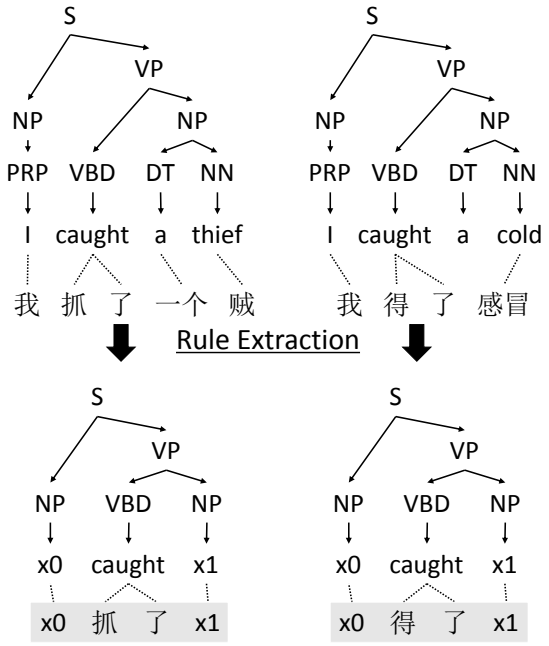

Figure 4. Tree-to-string translation rule examples.

Tree-to-string MT decoding exploits the same CKY parsing algorithm and the log-linear framework as hierarchical pharse-based MT. Because the parsing errors can hurt the performance of tree-to-string MT, Mi et al. proposed a forest-to-string MT model [43], which uses n-best parse trees instead of the 1-best parse tree to reduce the influence of parsing errors.

2.5 Attentional NMT

Attentional NMT [4] uses a single large neural network to model the entire translation process, which includes an encoder, a decoder and an attention model.

Given a source sentence $F = \{f_1, ..., f_I\}$, the encoder learns an annotation $h_i = \left[\vec{h}_i; \overleftarrow{h}_i\right]$ for f_i using a bi-directional recurrent neural network.

The decoder generates the target translation from left to right. The probability of generating next word e_j is,[1]

$$P\left(e_j|e_1^{j-1}, F\right) = softmax\left(g\left(e_{j-1}, t_j, s_j\right)\right) \tag{8}$$

where t_j is a decoding state for time step j, computed by,

$$t_j = f\left(t_{j-1}, e_{j-1}, s_j\right) \tag{9}$$

s_j is a source representation for time j, calculated as,

$$s_j = \sum_{i=1}^{I} \alpha_{j,i} \cdot h_i \tag{10}$$

where $\alpha_{j,i}$ is the attention model that scores how well the inputs around position i and the output at position j match, computed as,

$$\alpha_{j,i} = \frac{\exp\left(a\left(t_{j-1}, h_i\right)\right)}{\sum_{k=1}^{I} \exp\left(a\left(t_{j-1}, h_k\right)\right)} \tag{11}$$

As we can see, NMT only learns an attention distribution for each target word over all source words and does not apply exact mutually-exclusive word or phrase level alignments. As a result, it is known that attentional NMT systems make mistakes in over- or under-translation [9, 44]. Besides, without the limitation of translation rules in symbolic MT, NMT can generate any translation for one input sentence and is more likely to produce completely unrelated translation.

The NMT training can be done by backpropagation with different optimization methods, such as Adam [27]. The decoding process of NMT is to find the best translation with the highest probability as computed in Equation 12.

$$P\left(E|F\right) = \prod_{j=1}^{J} P\left(e_j|e_1^{j-1}, F\right) \tag{12}$$

The standard decoding algorithm for NMT is beam search, that is to keep n-best hypotheses at each time step j.

[1] g, f and a in Equation 13, 14 and 11 are nonlinear, potentially multi-layered, functions.

3. A Neural Word Reordering Model for Hierarchical Phrase-based MT

3.1 Introduction

Hierarchical phrase-based MT [8] applies a series of hierarchical phrase-based translation rules, which can perform both lexical translating and reordering, to translate a source sentence. However, selecting proper translation rules during decoding is a major challenge, as a large number of hierarchical rules can be applied to any source sentence.

Which translation rules should be used on one particular input sentence depends on the sentence context. He et al. [23] and Liu et al. [35] used maximum entropy approaches to integrate rich contextual information for target side rule selection. Cui et al. [10] proposed a joint model to select hierarchical rules for both source and target sides. Wang et al. [67] proposed to estimate the semantic similarity between nonterminals and their phrasal substitutions during decoding to award translation rules with high similarities.

In addition, word or phrase reordering models have also been integrated into hierarchical phrase-based MT for better rule selection [21, 24, 47, 7]. Among these, Hayashi et al. [21] demonstrated the effectiveness of using word reordering within hierarchical phrase-based MT by integrating Tromble and Eisner's word reordering model [63] into the hierarchical translation model.

The word reordering model helps score the reordering of words during translation, reducing the number of reordering errors caused by selecting the wrong translation rules. Figure 5 shows a Chinese-to-English hierarchical phrase-based MT example that demonstrates how the word reordering model can help hierarchical phrase-based MT. In this translation, the rule "X1 X2 男生→ X1 *guy* X2" is applied to the Chinese input sentence. Alternatively, the rule "X1 X2 男生→ X1 X2 *guy*" can also be applied to the input sentence but will cause an incorrect translation in this particular case. If the word pair "眼镜(*glasses*) 男生(*guy*)" can be predicted to be reversed by the reordering model, then the translation system will prefer the translation rule adopted in Figure 5 that reverses this word pair.

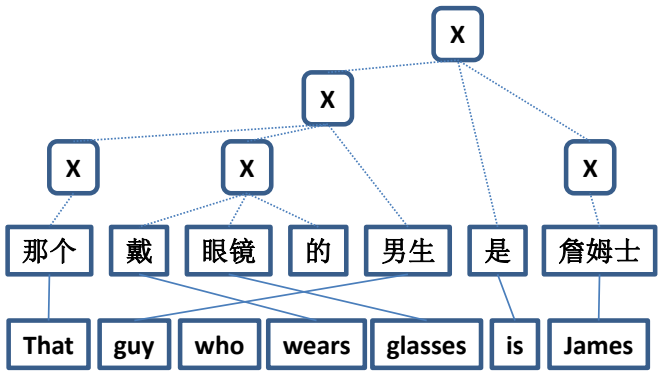

Figure 5. A example of hierarchical phrase-based MT.

Hayashi et al.'s method [21] demonstrated that reordering models can help resolve this problem, but one deficiency of the method is that the computation cost is quite expensive, as the model predicts reorderings for all word pairs in the input sentence. That is, if the input sentence length is n, this model needs to calculate reorderings for $O(n^2)$ word pairs.

In contrast, Feng et al. [14] proposed a word reordering model that only estimates reorderings for two contiguous source words, and applied their model to phrase-based MT. Because this limitation of the model results in a complexity of $O(n)$, it reduces computation cost significantly compared with Tromble and Eisner's model [63], and it still achieves significant reordering improvements over the baseline system.

We strike a balance between these two approaches. Specifically, we incorporate word reordering information into hierarchical phrase-based MT by training a series of separate reordering sub-models for word pairs with different distances. In Chinese-to-English and Japanese-to-English translation experiments, the translation performance achieved consistent improvements as more sub-models for longer distance reorderings were integrated, but the improvement levelled off quickly. In other words, sub-models for reordering distance longer than a given threshold did not improve translation quality significantly. In the experiments section, we also give detailed analyses of why reordering sub-models for longer distances were not as useful for translation quality.

By predicting local reorderings shorter than a given threshold, our model

exploits more reordering information than Feng et al. [14], while preventing the quadratic explosion in computation time of Hayashi et al.'s method [21]. In addition, our reordering model learned by feed-forward neural network (FNN) achieves better performance than the previous linear model that uses symbolic features.

3.2 Related Work

Reordering modeling has been extensively studied for phrase-based MT [32]. Because it is bilingual phrase pairs that are used as the translation unit for phrase-based MT, most reordering models used in phrase-based MT learn reordering of phrase pairs and implicitly make an assumption that word reorderings within phrase pairs are correct [29, 72, 48, 33].

These existing reordering models are not suitable for hierarchical phrase-based MT, which does not use phrase pairs as translation units. Huck et al. [24] introduced a reordering model for hierarchical phrase-based translation, which determines and estimates the orientations of nonterminals in translation rules. Nguyen and Vogel [47] proposed to integrate phrase-based reordering features into hierarchical phrase-based MT by mapping a hierarchical phrase-based derivation into a discontinuous phrase-based translation path, which enhanced the hierarchical phrase-based model significantly. However, there are some forms of hierarchical phrase-based rules which cannot be mapped into a reasonable sequence of phrase pairs and non-terminals. Cao et al. [7] proposed a lexicalized reordering model which is built directly on hierarchical phrase-based rules and compatible with any kind of hierarchical phrase-based rules.

Compared to the proposed word reordering model, these phrase-based reordering models limited to learning the reordering of contiguous phrases. When phrase length is short, in extreme cases, when phrase length is one, their models only learn reordering for contiguous word pairs, while our model releases such a constraint and can be applied to two source words with longer distances. And our experiments showed that reordering prediction for word pairs with distance 2,3... can improve translation qualities significantly as well.

Bisazza and Federico [5] modelled reordering as the problem of deciding whether a given input word should be translated after another. However, two

source words that are aligned to contiguous target words may have long distance, which makes this classification task harder than determining local reorderings as in our model.

There is also some work that exploits syntactic information to help reorderings in hierarchical phrase-based translation [18, 26, 39]. However, high quality parsers are not always available and parsing errors can influence the performance of these methods significantly.

3.3 Modeling

Let $e_1^J = e_1, \ldots, e_J$ be a target translation of $f_1^I = f_1, \ldots, f_I$ and A be word alignment links between e_1^J and f_1^I. Our model estimates the reordering probability of the source sentence as follows:

$$\Pr\left(f_1^I, e_1^J, A\right) \approx \prod_{n=1}^{N} \prod_{i,i':1\leq i<i'\leq I, i'-i=n} \Pr\left(f_1^I, e_1^J, A, i, i'\right) \tag{13}$$

where $\Pr\left(f_1^I, e_1^J, A, i, i'\right)$ is the reordering probability of the word pair $\langle f_i, f_{i'}\rangle$ during translation and N is the maximum distance considered by the reordering model, which is empirically determined by supposing that estimating reorderings longer than N does not improve translation performance significantly.

Previous word reordering models [63, 14] consider the reordering of a source word pair to be reversed or not. When a source word is aligned to several discontinuous target words, it can be hard to determine if a word pair is reversed or not as shown in Figure 6. They solved this problem by only using one alignment from multiple alignment links and ignoring the others. For example, in Figure 6 the alignment between "放弃(*give up*)" and "*up*" is ignored. In contrast, our model handles all alignment links to cover more word reordering patterns.

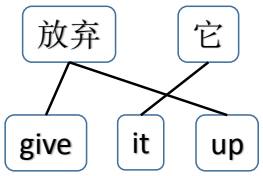

Figure 6. Multiple alignment links.

Suppose that f_i is aligned to π_i ($\pi_i \geq 0$) target words. When $\pi_i > 0$, we use

$$\{a_{ik} | 1 \leq k \leq \pi_i\},$$

to represent the positions of target words aligned to f_i. If $\pi_i = 0$ or $\pi_{i'} = 0$, $\Pr\left(f_1^I, e_1^J, A, i, i'\right) = 1$,[2] otherwise,

$$\Pr\left(f_1^I, e_1^J, A, i, i'\right) = \prod_{u=1}^{\pi_i} \prod_{v=1}^{\pi_{i'}} \Pr\left(o_{ii'uv} | f_{i-3}^{i'+3}, e_{a_{iu}}, e_{a_{i'v}}\right) \tag{14}$$

where

$$o_{ii'uv} = \begin{cases} 0 & (a_{iu} \leq a_{i'v}) \\ 1 & (a_{iu} > a_{i'v}) \end{cases}. \tag{15}$$

Here, $o_{ii'uv}$ indicates whether the translation $e_{a_{iu}}$ of f_i and the translation $e_{a_{i'v}}$ of $f_{i'}$ should be reversed or not in the target side.

Next, we need a model to estimate the probability of each $o_{ii'uv}$. As mentioned in the introduction, we train a series of sub-models,

$$M_1, M_2, \ldots, M_N$$

to learn reorderings for word pairs with different distances. In other words, for the word pair $\langle f_i, f_{i'} \rangle$ with distance $i' - i = n$, its reordering probability $\Pr\left(o_{ii'uv} | f_{i-3}^{i'+3}, e_{a_{iu}}, e_{a_{i'v}}\right)$ is estimated by M_n. Different sub-models are trained and integrated into the translation system separately.

Each sub-model M_n is implemented by an FNN, which has the same structure with Vaswani et al.'s neural language model [66]. The input to M_n is a sequence of $n + 9$ words: $f_{i-3}^{i'+3}, e_{a_{iu}}, e_{a_{i'v}}$. In Hayashi's model, only source-side contextual features were used. Because multiple correct translations may exist for an input sentence and different translations need different reorderings of the source sentence as shown in Figure 7, our model also integrates target-side features $e_{a_{iu}}, e_{a_{i'v}}$ for reordering.

The input layer projects each word into an embedding vector using a matrix of input word embeddings. These embeddings are followed by two hidden layers that combine all of the input embeddings. Thus unlike when using the averaged

[2]In translation experiments, we also tried adding a new penalty feature (how many source words in the input sentence are unaligned) to penalize unaligned words. However, this feature did not influence translation performance significantly.

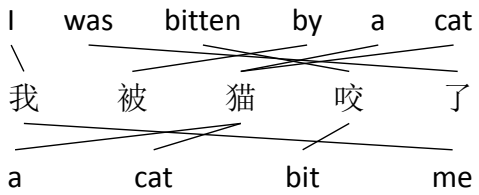

Figure 7. Different correct translations for one source sentence.

perceptron algorithm of Hayashi et al. [21], we do not need to manually design features to achieve high accuracy. The output layer has two neurons that calculate $\Pr(o_{ii'uv} = 1)$ and $\Pr(o_{ii'uv} = 0)$.

The backpropagation algorithm [53] is used to train the parameters for each reordering sub-model. The training instances for each sub-model are extracted from the word-aligned parallel corpus according to Algorithm 2. For example, the word pair "戴(*wears*) 男生(*guy*)" in Figure 5 will be extracted as a positive instance for M_3. The input of this instance is as follows: "<*s*> <*s*> 那个 戴 眼镜 的 男生 是 詹姆士 </*s*> *wears guy*", where "<*s*>" and "</*s*>" represent the beginning and ending of a sentence. If a word never occurs or only occurs once in the training corpus, we replace it with a special symbol "<*unk*>".

3.4 Decoding

To integrate our model into the hierarchical phrase-based translation system, a new feature $score_n(r)$ is added to each rule r for each M_n.[3]

Suppose that r is applied to the input sentence f_1^I, where

- r covers the source span $[f_\varphi, f_\vartheta]$
- $\tilde{f}$ contains nonterminals $\{X_k | 1 \leq k \leq K\}$
- X_k covers the span $[f_{\varphi_k}, f_{\vartheta_k}]$

[3]Note that these scores are correspondingly calculated for different sub-models M_n and the sub-model weights are tuned separately.

Algorithm 2 Extract training instances.

Require: A pair of parallel sentence f_1^I and e_1^J with word alignment links.
Ensure: Training examples for $M_1, M_2, \ldots, M_N$.

```
for i = 1 to I − 1 do
  for i' = i + 1 to I do
    if i' − i ≤ N then
      for u = 1 to π_i do
        for v = 1 to π_{i'} do
          if a_{iu} ≤ a_{i'v} then
            (f_{i−3}^{i'+3}, e_{a_{iu}}, e_{a_{i'v}}, 0) is a negative instance for M_{i'−i}
          else
            (f_{i−3}^{i'+3}, e_{a_{iu}}, e_{a_{i'v}}, 1) is a positive instance for M_{i'−i}
          end if
        end for
      end for
    end if
  end for
end for
```

Then,

$$score_n(r) = \sum_{\langle i,i' \rangle \in S - \bigcup_{k=1}^{K} S_k \wedge i'-i=n} \log\left(\Pr\left(f_1^I, e_1^J, A, i, i'\right)\right) \tag{16}$$

where

$$S : \{\langle i, i' \rangle \,|\varphi \leq i < i' \leq \vartheta\}$$
$$S_k : \{\langle i, i' \rangle \,|\varphi_k \leq i < i' \leq \vartheta_k\}$$

For example, in Figure 8, if a rule "X1 X2 男生→ X1 *guy* X2" is applied to the source phrase with shade, then

$$[f_\varphi, f_\vartheta] = [1,5]\,; [f_{\varphi_1}, f_{\vartheta_1}] = [1,1]\,; [f_{\varphi_2}, f_{\vartheta_2}] = [2,4]$$
$$S - \bigcup_{k=1}^{K} S_k = \left\{ \begin{array}{l} \langle 1,2\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 1,5\rangle, \\ \langle 2,5\rangle, \langle 3,5\rangle, \langle 4,5\rangle \end{array} \right\}$$

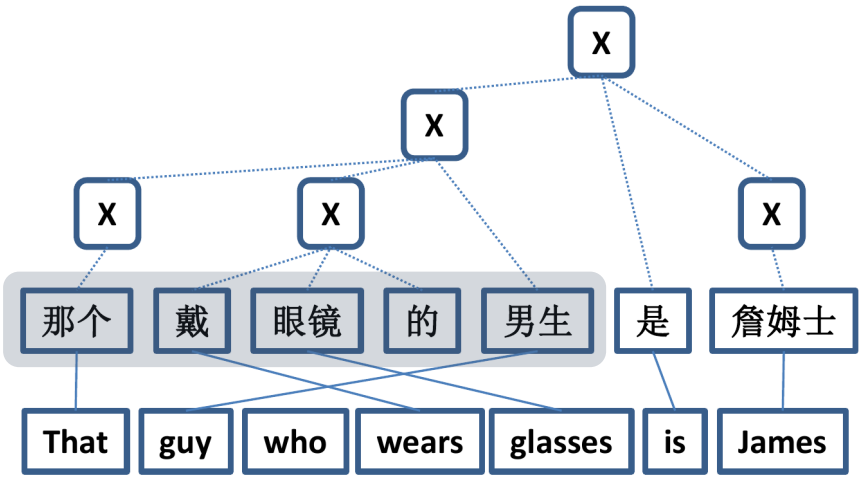

Figure 8. A example of hierarchical phrase-based MT decoding.

We can see from these reordering features that $score_n(r)$ cannot be calculated before decoding, because the information about $\{X_k|1 \leq k \leq K\}$ is needed. And to calculate neural network probabilities, we also need source-side context information from the input sentence, which means feature scores can be different for one translation rule when it is applied to different source sentences. Thus, this new feature must be calculated separately for each input source sentence. However, since our model does not use target n-gram information, therefore, we do not need to consider future costs for these reordering features as we do when using n-gram language models in hierarchical phrase-based system.

One concern in using target features is the computational efficiency, because reordering probabilities have to be calculated during decoding. However, we can cache probabilities to reduce the expensive neural network computation using hash tables. That is, for each sequence $\left(f_{i-3}^{i'+3}, e_{a_{iu}}, e_{a_{i'v}}, o_{ii'uv}\right)$, our model only needs to calculate the reordering probability $\Pr\left(o_{ii'uv}|f_{i-3}^{i'+3}, e_{a_{iu}}, e_{a_{i'v}}\right)$ once. Note that we clear the cache for each input sentence. This is because the reordering probabilities calculated for one sentence are seldom used for other sentences, and the lookup will slow down somewhat as the hash table size grows.[4]

[4]As we are using a cache, memory usage is a concern, but the size of the cache for each sentence is negligible compared to the size of the translation and language models, and thus the memory footprint is not increased significantly.

3.5 Experiments

3.5.1 Setting

We evaluated the proposed approach for Chinese-to-English (zh-en) and Japanese-to-English (ja-en) translation tasks. The official datasets for the patent machine translation task at NTCIR-9 [19] were used in our experiments. The detailed statistics for training, development and test sets are given in Table 3.

			SOURCE	TARGET
zh-en	TRAINING	#Sents	954K	
		#Words	37.2M	40.4M
		#Vocab	288K	504K
	DEV	#Sents	2K	
		#Words	75.4K	77.5K
	TEST	#Sents	2K	
		#Words	55.5K	58.1K
ja-en	TRAINING	#Sents	3.14M	
		#Words	118M	104M
		#Vocab	150K	273K
	DEV	#Sents	2K	
		#Words	74.6K	66.5K
	TEST	#Sents	2K	
		#Words	77.8K	69.5K

Table 3. Data sets.

In NTCIR-9, the development and test sets were both provided for the zh-en task while only the test set was provided for the ja-en task. Therefore, we used the sentences from the NTCIR-8 ja-en and en-ja test set as the development set for our ja-en experiments. The word segmentation was done by BaseSeg [75] for Chinese and Mecab[5] for Japanese.

To learn neural reordering models, the training and development sets were combined to obtain symmetric word alignments using GIZA++ [51] and the *grow-diag-final-and* heuristic [32]. The reordering instances extracted from the aligned

[5]http://sourceforge.net/projects/mecab/files/

training and development sets were used as the training and validation data for learning neural reordering models. We trained our model on the training data iteratively and stopped the training process when validation perplexity stopped decreasing. The validation data was randomly split into two parts. One part was used to stop the training process and the other part was used to calculate accuracies of reordering models. Neural reordering models were trained by the toolkit NPLM [66]. For the zh-en task, training instances extracted from all the 954K sentence pairs were used to train neural reordering models and the numbers of training instances are 40.0M, 38.4M, 37.1M, 36.0M for M_1, M_2, M_3, M_4, respectively. For the ja-en task, training instances are from 1M sentence pairs that were randomly selected from all the 3.14M sentence pairs and the numbers of instances are 38.8M, 36.6M, 35.5M, 34.3M for M_1, M_2, M_3, M_4, respectively.

We implemented Hayashi et al.'s model [21] to compare with our approach. The training instances for their model were extracted from the same sentence pairs as ours and instance sizes are 771.6M and 725.6M for zh-en and ja-en, respectively. We also implemented Cao et al.'s phrase reordering model [7] for comparison. The whole rule table extracted from the aligned training set was used as training data for their model.

For each translation task, a recent version of the Moses hierarchical phrase-based decoder [30] with the training scripts was used as the baseline system. We used the default parameters for Moses. A 5-gram language model was trained on the target side of the training corpus by IRST LM Toolkit[6] with improved Kneser-Ney smoothing. Since both zh-en and ja-en language pairs have quite different word orders, we set the distortion limit (max chart span) to be 20. The test set (2K sentences) contains 1.31K and 1.73K sentences with length longer than 20 for the zh-en and ja-en tasks, respectively.

We integrated our reordering models into Base. Each sub-model weight was tuned by MERT [49] together with other feature weights (language model, word penalty, etc.) under the log-linear framework [50].

		BASE	Hayashi	Cao	M_1^1	M_1^2	M_1^3	M_1^4
zh-en	Average	33.14	34.36	34.70	34.66	35.82	35.85	**35.93**
	Deviation	0.19	0.14	0.13	0.09	0.26	0.18	0.20
ja-en	Average	30.03	30.92	31.69	31.53	32.11	32.50	**32.58**
	Deviation	0.18	0.21	0.16	0.19	0.18	0.17	0.13

Table 4. Translation results (BLEU).

zh-en	BASE	M_1^1	M_1^2	M_1^3
M_1^1	$\gg$			
M_1^2	$\gg$	$\gg$		
M_1^3	$\gg$	$\gg$	$-$	
M_1^4	$\gg$	$\gg$	$-$	$-$

ja-en	BASE	M_1^1	M_1^2	M_1^3
M_1^1	$\gg$			
M_1^2	$\gg$	$\gg$		
M_1^3	$\gg$	$\gg$	$\gg$	
M_1^4	$\gg$	$\gg$	$\gg$	$-$

Table 5. Significance test results using bootstrap resampling w.r.t. BLEU scores.

3.5.2 Result and Analysis

Table 4 gives detailed translation results and Table 5 shows significance test results using bootstrap resampling[7] [28]. "Hayashi" represents the method of Hayashi et al. [21], "Cao" represents the method of Cao et al. [7] and "M_1^j $(j = 1, 2, 3, 4)$" means that BASE was augmented with the reordering scores calcuated from a series of sub-models M_1 to M_j. For example, M_1^3 means M_1, M_2 and M_3 are integrated; M_1^4 means M_1, M_2, M_3 and M_4 are integrated. We ran MERT 4 times for each experiment and show the average BLEU score with the standard deviation.

We can see that, with 4 sub-models integrated, our method outperformed both the Hayashi and Cao models significantly. Note that integrating only M_1, which predicts reordering for two contiguous source words, has already given a BLEU improvement of 1.8% and 1.2% over BASE on zh-en and ja-en, respectively. As more sub-models for longer distance reordering are integrated, the translation performance improved consistently, although the improvement leveled off quickly. For the zh-en and ja-en tasks, M_n with $n \geq 3$ and $n \geq 4$, respectively, did not

[6]http://hlt.fbk.eu/en/irstlm

[7]The symbol $\gg$ represents a significant difference at the $p < 0.01$ level; $-$ means not significantly different at $p = 0.05$.

give a further performance improvement at a significant level. We also did extra experiments with much longer reordering sub-models, in which we trained a model M_{15} for word pairs with distance 15, then integrated both M_{15} and M_1^4 into BASE. However, the translation results had no significant improvement compared to BASE augmented with M_1^4.

Why did the improvement level off quickly? In other words, why do long distance reordering models have much less leverage over translation performance than short ones?

First, the prediction accuracy decreases as the reordering distance increases. Table 6 gives prediction accuracies on the validation data for each sub-model. One reason for accuracy decreasing is that the input size of the sub-model grows as the reordering distance increases. Namely, there is more context information between words that are farther apart, which is harder to capture with limited training data and simple models that do not explicitly consider information about the syntactic structure of the sentence.

Sub-model	M_1	M_2	M_3	M_4
zh-en	93.9	92.8	92.2	91.2
ja-en	92.9	91.3	90.1	89.3

Table 6. Classification accuracy of our model (%)

Second, we attribute the decrease in influence of the longer reordering models to the redundancy of the predictions among the reordering sub-models. That is, a long distance word reordering can often be determined by a series of shorter word reordering pairs. For example, in Figure 5, if word pairs "男生(*guy*) 是(*is*)" and "是(*is*) 詹姆士(*James*)" are both predicted to be not reversed, the reordering for "男生(*guy*) 詹姆士(*James*)" can be logically determined to be not reversed without prediction. As a result, sometimes predictions for longer reorderings will not be useful for the translation process. In fact, although the longest distance of source words in Figure 5 is 6, the longest distance of word pairs whose reorderings need to be predicted in order to accurately determine the ordering of all the words is 4.

But still, some predictions for longer reorderings are useful. For example, the reordering of "戴(*wears*) 男生(*guy*)" cannot be determined when "戴(*wears*) 眼

S	该(the) 照 明(lighting) 设 备(device) 1 还(further) 具 有(has) 耦合(coupled) 到 固态(solid-state) 光源(light source) 4 的 光纤(optical fiber) 5 。
R	the lighting device 1 further has an optical fiber 5 which is coupled to the solid-state light source 4 .
B	the illumination apparatus 1 also has coupled to the optical fiber of the solid-state light source 4 5 .
M_1^1	the illumination apparatus 1 also has a fiber coupled to the solid-state light source 4 5 .
M_1^2	the illumination apparatus 1 also has a fiber coupled to the solid-state light source 4 5 .
M_1^3	the illumination apparatus 1 also has a fiber 5 coupled to the solid-state light source 4 .
M_1^4	the illumination apparatus 1 also has a fiber 5 coupled to the solid-state light source 4 .

Table 7. Translation examples. S: input sentence, R: reference sentence, B: translation result of BASE, M_1^j ($j = 1, 2, 3, 4$): translation result with M_1^j being integrated.

镜(*glasses*)" is predicted to be not reversed and "眼镜(*glasses*) 男生(*guy*)" is reversed. This is the reason why the translation performance improves as more sub-models are integrated.

Table 7 gives a translation example to demonstrate how our model improves the reordering during translating.[8] As shown, the distance between source words "4" and "5" is 3, and after M_3 being integrated into BASE, this word pair can be correctly reordered.

Note that if we only integrate M_4 into BASE, the translation quality of BASE was improved in preliminary experiments. However, M_4 cannot predict reorderings for word pairs with distance less than 4. So M_1^3 will be still needed for predicting reorderings of word pairs with distance 1,2,3. But after M_1^3 being in-

[8]Note that "4" and "5" in source and target sentences are original source and target words. This sentence pair is from a patent translation corpus and there is a figure in the article, where the light source is labeled as 4 and the optical fiber is labeled as 5.

Reordering Distance	1	2	3	4
zh-en	90.1	88.3	87.0	85.6
ja-en	85.3	81.9	80.6	78.8

Table 8. Classification accuracy of Hayashi model (%).

Reordering Distance		1	2	3	4
Hayashi	zh-en	82.4	76.5	73.6	72.6
	ja-en	67.8	60.9	57.0	55.8
Our model	zh-en	95.3	93.8	92.7	91.6
	ja-en	93.9	91.6	90.3	89.1

Table 9. Classification accuracy for one-to-one alignment links (%).

tegrated, M_4 did not provide a large improvement due to the redundancy of the predictions among different reordering sub-models.

Now we analyze the reasons that our model outperformed the Hayashi and Cao models, respectively, as shown in Table 4.

Table 8 shows the reordering prediction accuracies of Hayashi model for word pairs with different distances. Note that Hayashi's model predicts reorderings for all word pairs, but only prediction accuracies for word pairs with distance 4 or less are shown. The definitions of classification accuracy for our method and Hayashi's model are slightly different. In Hayashi's model, one word pair is counted as one reordering instance. In contrast, one word pair with multiple alignment links may contain several reordering instances for our model, and if one source word is not aligned to any target word, we do not consider the reordering about this source word. For a direct comparison, Table 9 shows the reordering classification accuracy for two words that were both aligned to exactly one target word. As shown in Table 9, our approach learned reorderings much better than the Hayashi model. This is easy to understand, since our model was trained by feed-forward neural networks with distributed representations and incorporated rich context information, while Hayashi's model used the averaged perceptron algorithm and manually crafted symbolic features.

Our model also outperformed Cao's model, which already had a strong improvement compared to Base. Since their model needs to be trained on the

whole rule table and the hierarchical translation rule table is quite large, the training process will be very time-consuming. Thus they only used simply relative frequency and the add 0.5 smoothing technique to estimate the reordering probability. In other words, it is hard to use other features in their model due to efficiency issues. Besides, their model only estimates reorderings for contiguous phrase pairs.

Efficiency We performed a group of experiments to show how much the caching strategy can bring about efficiency improvements. We used a computer with Xeon E5-4650 CPU and CentOS 6.3 to translate all input sentences in the zh-en test set. Table 10 gives the hit rate (HR) of caching and the average translation time for one sentence with and without caching. The average translation time for BASE was 3.92 seconds.

$$HR = \frac{T_{cache}}{T_{cache} + T_{calculate}}$$

Here, T_{cache} was the number of times that we could find the reordering probability in the cache; $T_{calculate}$ was the number of times that the reordering probability could not be found in the cache and then had to be calculated by the neural reordering model.

Sub-models	Caching (sec)	No Caching (sec)	Hit Rate (%)
M_1^1	4.60	102.65	99.85
M_1^2	6.56	212.95	99.84
M_1^3	8.50	330.27	99.86
M_1^4	10.11	442.39	99.88

Table 10. Translation time and hit rate.

According to the results in Table 10, we can see that the hit rates were quite high. Using a cache in decoding, in most cases we just need to perform look up in hash tables to get the reordering probabilities. This results in high efficiency as hash table lookup is much faster than calculating neural networks.

One Model vs. Mutiple Sub-models Different from using one model to learn reordering for all word pairs, our model learns reordering with several separate sub-models. Different sub-models can be trained entirely separately, and we can take advantage of this easy parallelism to train models in a more reasonable time.

However, theoretically, one unified model will have better performance since separate sub-models do not share training instances. Suppose that the training corpus contains these two sentences "*I like sunny days*" and "*I like sunny and warm days*". The word pair "*I days*" occurs twice in the training corpus for the unified model and once for two separate sub-models, respectively. This indicates that the unified model suffers less from data sparsity. To test these effects, we did some extra experiments and let one neural network learn for word pairs with distance 4 or less. This neural network has the same structure as M_4 with 13 inputs. For word pairs with distance 1,2,3,4, the inputs are

$$f_{i-3}, \ldots, f_i, f'_i, \ldots, f_{i'+3}, e_{a_{iu}}, e_{a_{i'v}}, null, null, null$$
$$f_{i-3}, \ldots, f_i, f'_i, \ldots, f_{i'+3}, e_{a_{iu}}, e_{a_{i'v}}, f_{i+1}, null, null$$
$$f_{i-3}, \ldots, f_i, f'_i, \ldots, f_{i'+3}, e_{a_{iu}}, e_{a_{i'v}}, f_{i+1}, f_{i+2}, null$$
$$f_{i-3}, \ldots, f_i, f'_i, \ldots, f_{i'+3}, e_{a_{iu}}, e_{a_{i'v}}, f_{i+1}, f_{i+2}, f_{i+3}$$

Here, *null* is a specific symbol that represents a default position.

Table 11 shows the reordering prediction accuracy of this model for word pairs with different distances. Table 12 gives the translation result after integrating this model into BASE to predict reordering for word pairs with different distances. The corresponding original results using multiple sub-models are also shown for a direct comparison.

Reordering Distance		1	2	3	4
One	zh-en	93.9	93.0	92.2	91.3
	ja-en	92.8	91.6	90.3	89.2
Multiple	zh-en	93.9	92.8	92.2	91.2
	ja-en	92.9	91.3	90.1	89.3

Table 11. Classification accuracy of using one unified model (%).

Reordering	Distance	1	1,2	1,2,3	1,2,3,4
One	zh-en	34.50	35.70	35.50	35.74
	ja-en	31.42	32.00	32.47	32.54
Multiple	zh-en	34.66	35.82	35.85	35.93
	ja-en	31.53	32.11	32.50	32.58

Table 12. Translation performance of using one unified model (BLEU).

As can be seen, using one model or using multiple sub-models to learn reordering have nearly the same classification and translation performance. This shows that using separate models does not hurt performance while keeping the merit of training efficiency. This means that although one unified model is theoretically more robust to sparse data, when the training corpus becomes large, separate sub-models can also learn the reorderings well, and the performance difference between one unified model and separate sub-models is negligible.

Comparison of Machine Learning Methods To analyze the influence of machine learning method choice for our model, we also tried the averaged perceptron (AP) algorithm to learn each sub-model, which was used by Hayashi et al. [21] for their model training. The features are given in Table 13.

$f_k, f_k e_{a_{iu}}, f_k e_{a_{i'v}} \, (i-3 \leq k \leq i'+3)$
$f_i f_{i'}, f_i f_{i'} e_{a_{iu}}, f_i f_{i'} e_{a_{i'v}}, f_i f_{i'} e_{a_{iu}} e_{a_{i'v}}$
$f_i f_k, f_{i'} f_k, f_i f_k e_{a_{iu}}, f_{i'} f_k e_{a_{i'v}},$ $f_i f_{i'} f_k, f_i f_{i'} f_k e_{a_{iu}}, f_i f_{i'} f_k e_{a_{i'v}}, f_i f_{i'} f_k e_{a_{iu}} e_{a_{i'v}}$ $(i-3 \leq k \leq i'+3, k \neq i, k \neq i')$

Table 13. Features.

Table 14 and 15 show prediction accuracies and translation performance using AP algorithm. The corresponding original results using FNN are also shown for a direct comparison.

As can be seen, classifiers learned by feedforward neural networks perform better than the averaged perceptron algorithm and the translation performance

Sub-model		M_1	M_2	M_3	M_4
AP	zh-en	93.4	92.2	90.7	89.7
	ja-en	92.3	90.6	89.1	87.7
FNN	zh-en	93.9	92.8	92.2	91.2
	ja-en	92.9	91.3	90.1	89.3

Table 14. Classification accuracy of using AP for model training (%).

Sub-models		M_1^1	M_1^2	M_1^3	M_1^4
AP	zh-en	34.27	34.54	34.47	34.43
	ja-en	30.74	31.79	31.60	31.92
FNN	zh-en	34.66	35.82	35.85	35.93
	ja-en	31.53	32.11	32.50	32.58

Table 15. Translation performance of using AP for model training (BLEU).

with the neural reordering model outperformed that with the reordering model learned by the AP algorithm, which means the difference of training methods is an important factor explaining why our model outperformed Hayashi et al.'s model [21] in Table 4. However, FNNs are not suitable for use in the Hayashi model since the training and decoding time for FNN is already quite long. Using FNN for Hayashi et al.'s model will cost nearly one minute to translate one sentence according to our experiments, while our most complex model took about 10 seconds as shown in Table 10.[9]

3.6 Conclusion

We adopted a series of separate sub-models to reorder source word pairs with different distances and integrate this model into hierarchical phrase-based MT. Experiments and analyses have shown that only reordering predictions for word pairs with distances less than a specific threshold improved translation performance clearly, and longer distance reordering sub-models were not as helpful for translation quality. With only sub-models for short distance reorderings being used, training and decoding for our model are much more efficient compared to

[9]Cache was used in all experiments.

previous models, while keeping the majority of helpful word reordering information. Besides, our reordering model is learned by feed-forward neural networks and incorporates rich context information for better performance. On both Chinese-to-English and Japanese-to-English translation tasks, the proposed model outperformed the previous models significantly.

4. A Binarized Neural Network Joint Model for Hierarchical Phrase-based MT

4.1 Introduction

The neural network joint model (NNJM), which augments the n-gram neural language model (NLM) with an m-word source context window as shown in Figure 9a, can be used to guide symbolic MT and had achieved large gains in translation accuracy [11].

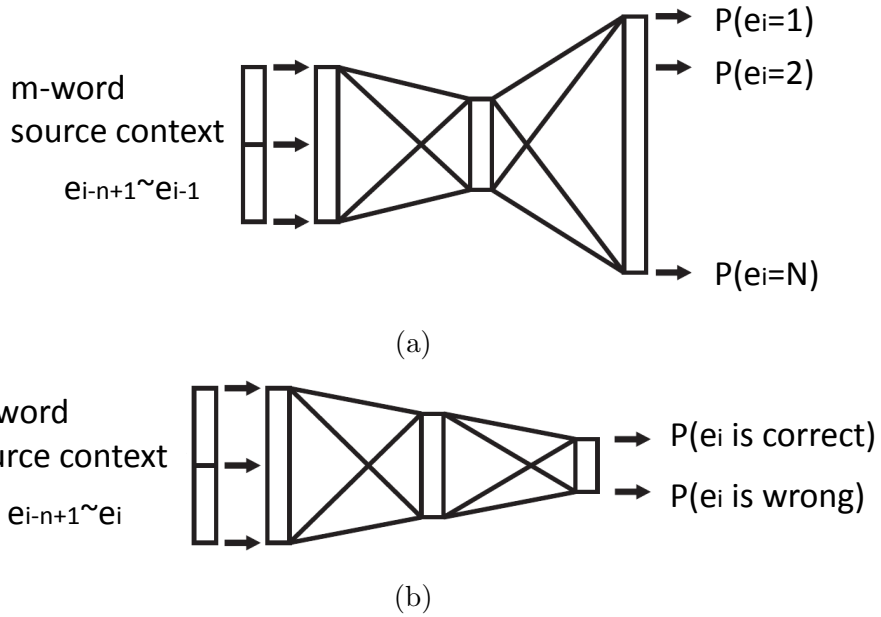

Figure 9. (a) the original NNJM and (b) the proposed BNNJM

While this model is effective, the computation cost of using it in a large-vocabulary MT task is quite expensive, as probabilities need to be normalized over the entire vocabulary. To solve this problem, Devlin et al. [11] presented a technique to train the NNJM to be self-normalized and avoided the expensive normalization cost during decoding. However, they also note that this self-normalization technique sacrifices neural network accuracy, and the training process for the self-normalized neural network is very slow, as with standard maximum likelihood estimation (MLE).

To remedy the problem of long training times in the context of NLMs, Vaswani et al. [66] used a method called noise contrastive estimation (NCE). Compared with MLE, NCE does not require repeated summations over the whole vocabulary and performs nonlinear logistic regression to discriminate between the observed data and artificially generated noise.

We proposed an alternative framework of binarized NNJMs (BNNJM), which are similar to the NNJM, but use the current target word not as the output, but as the input of the neural network, estimating whether the target word under examination is correct or not, as shown in Figure 9b. Because the BNNJM uses the current target word as input, the information about the current target word can be combined with the context word information and processed in the hidden layers.

The BNNJM learns a simple binary classifier, given the context and target words, therefore it can be trained by MLE very efficiently. "Incorrect" target words for the BNNJM can be generated in the same way as NCE generates noise for the NNJM. We proposed a novel noise distribution based on translation probabilities to train the NNJM and the BNNJM effectively and efficiently.

4.2 Related Work

Xu et al. [71] proposed a method to use binary classifiers to learn NNLMs. But they also used the current target word in the output, similarly to NCE. The BNNJM uses the current target word as input, so the information about the current target word can be combined with the context word information and processed in hidden layers.

Mauser et al. [40] presented discriminative lexicon models to predict target words. They train a separate classifier for each target word, as these lexicon models use symbolic representations of words and different classifiers do not share features. In contrast, the BNNJM uses real-valued vector representations of words and shares training data, so we train one classifier and can use the similarity information between words.

4.3 Neural Network Joint Model

Let $E = e_1^J$ be a translation of $F = f_1^I$. The NNJM [11] defines the following probability,

$$P\left(E|F\right) = \prod_{j=1}^{J} P\left(e_j | f_{a_j-(m-1)/2}^{a_j+(m-1)/2}, e_{j-n+1}^{j-1}\right) \tag{17}$$

where target word e_j is affiliated with source word f_{a_j}. Affiliation a_j is derived from the word alignments using heuristics. If e_j aligns to exactly one source word, a_j is the index of this source word; If e_j aligns to multiple source words, a_j is the index of the aligned word in the middle; If e_j is unaligned, they inherit its affiliation from the closest aligned word.

To estimate these probabilities, the NNJM uses m source context words and $n-1$ target history words as input to a neural network and performs estimation of unnormalized probabilities $p\left(e_j|C\right)$ before normalizing over all words in the target vocabulary V,

$$\begin{aligned} P\left(e_j|C\right) &= \tfrac{p(e_j|C)}{Z(C)} \\ Z\left(C\right) &= \sum_{e_j{}' \in V} p\left(e_j{}'|C\right) \end{aligned} \tag{18}$$

where C stands for source and target context words as in Equation 17.

The NNJM can be trained on a word-aligned parallel corpus using standard MLE, but the cost of normalizing over the entire vocabulary to calculate the denominator in Equation 18 is quite large. Devlin et al. [11]'s self-normalization technique can avoid normalization cost during decoding, but not during training.

NCE can be used to train NNLM-style models [66] to reduce training times. NCE creates a noise distribution $q\left(e_j\right)$, selects K noise samples $e_{j1}, ..., e_{jK}$ for each e_j and introduces a random variable v which is 1 for training examples and 0 for noise samples,

$$\begin{aligned} P\left(v=1, e_j|C\right) &= \tfrac{1}{1+K} \cdot \tfrac{p(e_j|C)}{Z(C)} \\ P\left(v=0, e_j|C\right) &= \tfrac{K}{1+K} \cdot q\left(e_j\right). \end{aligned} \tag{19}$$

NCE trains the model to distinguish training data from noise by maximize the conditional likelihood,

$$L = \log P\left(v=1|C, e_j\right) + \sum_{k=1}^{K} \log P\left(v=0|C, e_{jk}\right). \tag{20}$$

The normalization cost can be avoided by using $p\left(e_j|C\right)$ as an approximation of $P\left(e_j|C\right)$.[10]

4.4 Binarized Neural Network Joint Model

We proposed a new framework of the binarized NNJM (BNNJM), which is similar to the NNJM but learns not to predict the next word given the context, but solves a binary classification problem by adding a variable $v \in \{0, 1\}$ that stands for whether the current target word e_j is correctly/wrongly produced in terms of source context words $f_{a_j-(m-1)/2}^{a_j+(m-1)/2}$ and target history words e_{j-n+1}^{j-1} ,

$$P\left(v|f_{a_j-(m-1)/2}^{a_j+(m-1)/2}, e_{j-n+1}^{j-1}, e_j\right). \tag{21}$$

The BNNJM is learned by a feed-forward neural network with $m + n$ inputs $\left\{f_{a_j-(m-1)/2}^{a_j+(m-1)/2}, e_{j-n+1}^{j-1}, e_j\right\}$ and two outputs for $v = 1/0$.

Because the BNNJM uses the current target word as input, the information about the current target word can be combined with the context word information and processed in the hidden layers. Thus, the hidden layers can be used to learn the difference between correct target words and noise in the BNNJM, while in the NNJM the hidden layers just contain information about context words and only the output layer can be used to discriminate between the training data and noise, giving the BNNJM more power to learn this classification problem.

We can use the BNNJM probability in translation as an approximation for the NNJM as below,

$$P\left(e_j|f_{a_j-(m-1)/2}^{a_j+(m-1)/2}, e_{j-n+1}^{j-1}\right) \approx P\left(v = 1|f_{a_j-(m-1)/2}^{a_j+(m-1)/2}, e_{j-n+1}^{j-1}, e_j\right). \tag{22}$$

As a binary classifier, the gradient for a single example in the BNNJM can be calculated efficiently by MLE without it being necessary to calculate the softmax over the full vocabulary. On the other hand, we need to create "positive" and "negative" examples for the classifier. Positive examples can be extracted directly from the word-aligned parallel corpus as $\left\langle f_{a_j-(m-1)/2}^{a_j+(m-1)/2}, e_{j-n+1}^{j-1}, e_j \right\rangle$; Negative examples can be generated for each positive example in the same way that NCE generates noise data as $\left\langle f_{a_j-(m-1)/2}^{a_j+(m-1)/2}, e_{j-n+1}^{j-1}, e_j{}' \right\rangle$, where $e_j{}' \in V \setminus \{e_j\}$.

[10]The theoretical properties of self-normalization techniques, including NCE and Devlin et al. [11]'s method, are investigated by Andreas and Klein [2].

4.5 Noise Sampling

Unigram Noise Vaswani et al. [66] adopted the unigram probability distribution (UPD) to sample noise for training NLMs with NCE,

$$q\left(e_j{}'\right) = \frac{occur\left(e_j{}'\right)}{\sum\limits_{e_j{}'' \in V} occur\left(e_j{}''\right)} \tag{23}$$

where $occur\left(e_j{}'\right)$ stands for how many times $e_j{}'$ occurs in the training corpus.

Translation Model Noise We proposed a noise distribution specialized for translation models, such as the NNJM or BNNJM.

Figure 10 gives a Chinese-to-English parallel sentence pair with word alignments to demonstrate the intuition behind our method.

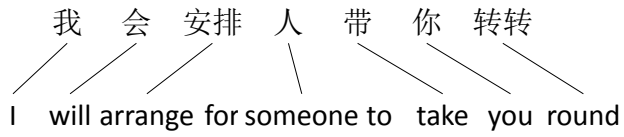

Figure 10. A parallel sentence pair.

Focusing on f_{a_j}="安排", this is translated into e_j ="*arrange*". For this positive example, UPD is allowed to sample any arbitrary noise, such as $e_j{}'$ = "*banana*". However, in this case, noise $e_j{}'$ = "*banana*" is not useful for model training, as constraints on possible translations given by the phrase table ensure that "安排" will never be translated into "*banana*". On the other hand, noise $e_j{}'$ = "*arranges*" and "*arrangement*" are both possible translations of "安排" and therefore useful training data, that we would like our model to penalize.

Based on this intuition, we propose the use of another noise distribution that only uses $e_j{}'$ that are possible translations of f_{a_j}, i.e., $e_j{}' \in U\left(f_{a_j}\right) \setminus \{e_j\}$, where $U\left(f_{a_j}\right)$ contains all target words aligned to f_{a_j} in the parallel corpus.

Because $U\left(f_{a_j}\right)$ may be quite large and contain many wrong translations caused by wrong alignments, "*banana*" may actually be included in U("安排"). To mitigate the effect of uncommon examples, we use a translation probability

distribution (TPD) to sample noise e_j' from $U\left(f_{a_j}\right) \setminus \{e_j\}$ as follows,

$$q\left(e_j'|f_{a_j}\right) = \frac{align\left(f_{a_j}, e_j'\right)}{\sum_{e_j'' \in U\left(f_{a_j}\right)} align\left(f_{a_j}, e_j''\right)} \tag{24}$$

where $align\left(f_{a_j}, e_j'\right)$ is how many times e_j' is aligned to f_{a_j} in the parallel corpus.

Note that e_j could be unaligned, in which case we assume that it is aligned to a special *null* word. Noise for unaligned words is sampled according to the TPD of the *null* word. If several target/source words are aligned to one source/target word, we choose to combine these target/source words as a new target/source word. The processing for multiple alignments helps sample more useful negative examples for TPD, and had little effect on the translation performance when UPD was used as the noise distribution for the NNJM and the BNNJM in our preliminary experiments.

4.6 Experiments

4.6.1 Setting

We evaluated the effectiveness of the proposed approach for Chinese-to-English (zh-en), Japanese-to-English (ja-en) and French-to-English (fr-en) translation tasks. The datasets officially provided for the patent machine translation task at NTCIR-9 [19] were used for the zh-en and ja-en tasks. The development and test sets were both provided for the zh-en task while only the test set was provided for the ja-en task. Therefore, we used the sentences from the NTCIR-8 ja-en and en-ja test set as the development set. Word segmentation was done by BaseSeg [75] for Chinese and Mecab[11] for Japanese. For the fr-en language pair, we used standard data for the WMT 2014 translation task. The training sets for zh-en, ja-en and fr-en tasks contain 1M, 3M and 2M sentence pairs, respectively.

For each translation task, a recent version of Moses hierarchical phrase-based decoder [30] with the training scripts was used as the baseline (Base). We used the default parameters for Moses, and a 5-gram language model was trained on the target side of the training corpus using the IRSTLM Toolkit[12] with improved

[11] http://sourceforge.net/projects/mecab/files/

[12] http://hlt.fbk.eu/en/irstlm

		zh-en		ja-en		fr-en	
		E	T	E	T	E	T
NNJM	UPD	20	22	19	49	20	28
	TPD	4		6		4	
BNNJM	UPD	14	16	12	34	11	22
	TPD	11		9		9	

Table 16. Epochs (E) and time (T) in minutes per epoch for each task.

Kneser-Ney smoothing. Feature weights were tuned by MERT [49].

The word-aligned training set was used to learn the NNJM and the BNNJM.[13] For both NNJM and BNNJM, we set $m = 7$ and $n = 5$. The NNJM was trained by NCE using UPD and TPD as noise distributions. The BNNJM was trained by standard MLE using UPD and TPD to generate negative examples.

The number of noise samples for NCE was set to be 100. For the BNNJM, we used only one negative example for each positive example in each training epoch, as the BNNJM needs to calculate the whole neural network (not just the output layer like the NNJM) for each noise sample and thus noise computation is more expensive. However, for different epochs, we resampled the negative example for each positive example, so the BNNJM can make use of different negative examples.

4.6.2 Result and Analysis

Table 16 shows how many epochs these two models needed and the training time for each epoch on a 10-core 3.47GHz Xeon X5690 machine.[14] Translation results are shown in Table 17[15].

[13]Both the NNJM and the BNNJM had one hidden layer, 100 hidden nodes, input embedding dimension 50, output embedding dimension 50. A small set of training data was used as validation data. The training process was stopped when validation likelihood stopped increasing.

[14]The decoding time for the NNJM and the BNNJM were similar, since the NNJM trained by NCE uses $p(e_j|C)$ as an approximation of $P(e_j|C)$ without normalization and the BNNJM only needs to be normalized over two output neurons.

[15]The symbol + and * represent significant differences at the $p < 0.01$ level against Base and NNJM+UPD, respectively. Significance tests were conducted using bootstrap resampling [28].

		zh-en	ja-en	fr-en
Base		32.95	30.13	24.56
NNJM	UPD	34.36+	**31.30+**	24.68
	TPD	34.60+	**31.50+**	24.80
BNNJM	UPD	32.89	30.04	24.50
	TPD	**35.05+***	**31.42+**	**25.84+***

Table 17. Translation results.

We can see that using TPD instead of UPD as a noise distribution for the NNJM trained by NCE can speed up the training process significantly, with a small improvement in performance. But for the BNNJM, using different noise distributions affects translation performance significantly. The BNNJM with UPD does not improve over the baseline system, likely due to the small number of noise samples used in training the BNNJM, while the BNNJM with TPD achieves good performance, even better than the NNJM with TPD on the Chinese-to-English and French-to-English translation tasks.

From Table 17, the NNJM does not improve translation performance significantly on the fr-en task. Note that the baseline BLEU for the fr-en task is lower than zh-en and ja-en tasks, indicating that learning is harder for the fr-en task than zh-en and ja-en tasks. The validation perplexities of the NNJM with UPD for zh-en, ja-en and fr-en tasks are 4.03, 3.49 and 8.37. Despite these difficult learning circumstances and lack of large gains for the NNJM, the BNNJM improves translations significantly for the fr-en task, suggesting that the BNNJM is more robust to difficult translation tasks that are hard for the NNJM.

Table 18 gives Chinese-to-English translation examples to demonstrate how the BNNJM (with TPD) helps to improve translations over the NNJM (with TPD). In this case, the BNNJM helps to translate the phrase "该 移动 持续 到" better. Table 19 gives translation scores for these two translations calculated by the NNJM and the BNNJM. Context words are used for predictions but not shown in the table.

As can be seen, the BNNJM prefers T_2 while the NNJM prefers T_1. Among these predictions, the NNJM and the BNNJM predict the translation for "到" most differently. The NNJM clearly predicts that in this case "到" should be

S: 该(this) 移动(movement) 持续(continued) 到(until) 寄生虫(parasite) 由(by) 两(two) 个 舌(tongues) 部 21 彼此(each other) 接触(contact) 时(where) 的 点(point) 接触(touched) 。
R: *this movement is continued until the parasite is touched by the point where the two tongues 21 contact each other .*
T_1**:** *the mobile continues to the parasite from the two tongue 21 contacts the points of contact with each other .*
T_2**:** *this movement is continued until the parasite by two tongue 21 contact points of contact with each other .*

Table 18. Translation examples. Here, S: source; R: reference; T_1 uses NNJM; T_2 uses BNNJM.

	NNJM	BNNJM
该– >the	1.681	-0.126
移动– >mobile	-4.506	-3.758
持续– >continues	-1.550	-0.130
到– >to	2.510	-0.220
SUM	-1.865	-4.236
该– >this	-2.414	-0.649
移动– >movement	-1.527	-0.200
null– >is	0.006	-0.055
持续– >continued	-0.292	-0.249
到– >until	-6.846	-0.186
SUM	-11.075	-1.341

Table 19. Scores for different translations.

translated into "to" more than "until", likely because this example rarely occurs in the training corpus. However, the BNNJM prefers "until" more than "to", which demonstrates the BNNJM's robustness to less frequent examples.

Analysis for ja-en Translation Results Finally, we examine the translation results to explore why the BNNJM with TPD did not outperform the NNJM with TPD for the ja-en translation task, as it did for the other translation tasks. We

found that using the BNNJM instead of the NNJM on the ja-en task did improve translation quality significantly for infrequent words, but not for frequent words.

First, we describe how we estimate translation quality for infrequent words. Suppose we have a test set S, a reference set R and a translation set T with I sentences,

$$S_i \left(1 \leq i \leq I\right), R_i \left(1 \leq i \leq I\right), T_i \left(1 \leq i \leq I\right)$$

T_i contains J individual words,

$$W_{ij} \in Words\left(T_i\right)$$

$T_o\left(W_{ij}\right)$ is how many times W_{ij} occurs in T_i and $R_o\left(W_{ij}\right)$ is how many times W_{ij} occurs in R_i.

The general 1-gram translation accuracy [52] is calculated as,

$$P_g = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} \min(T_o(W_{ij}), R_o(W_{ij}))}{\sum_{i=1}^{I}\sum_{j=1}^{J} T_o(W_{ij})}$$

This general 1-gram translation accuracy does not distinguish word frequency.

We use a modified 1-gram translation accuracy that weights infrequent words more heavily,

$$P_c = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} \min(T_o(W_{ij}), R_o(W_{ij})) \cdot \frac{1}{Occur\left(W_{ij}\right)}}{\sum_{i=1}^{I}\sum_{j=1}^{J} T_o(W_{ij})}$$

where $Occur\left(W_{ij}\right)$ is how many times W_{ij} occurs in the whole reference set. Note P_c will not be 1 even in the case of completely accurate translations, but it can approximately reflect infrequent word translation accuracy, since correct frequent word translations contribute less to P_c.

Table 20 shows P_g and P_c for different translation tasks. It can be seen that the BNNJM improves infrequent word translation quality similarly for all translation tasks, but improves general translation quality less for the ja-en task than the other translation tasks. We conjecture that the reason why the BNNJM is less useful for frequent word translations on the ja-en task is the fact that the ja-en parallel corpus has less accurate function word alignments than other

	zh-en		ja-en		fr-en	
	P_g	P_c	P_g	P_c	P_g	P_c
NNJM	70.3	5.79	68.2	4.15	61.2	6.70
BNNJM	70.9	5.97	68.4	4.30	61.7	6.86
Imp. (%)	0.85	3.1	0.29	3.6	0.81	2.4

Table 20. 1-gram precisions and improvements.

language pairs, as the grammatical features of Japanese and English are quite different.[16] Wrong function word alignments will make noise sampling less effective and therefore lower the BNNJM performance for function word translations. Although wrong word alignments will also make noise sampling less effective for the NNJM, the BNNJM only uses one noise sample for each positive example, so wrong word alignments affect the BNNJM more than the NNJM.

4.7 Conclusion

We proposed an alternative to the NNJM, the BNNJM, which learns a binary classifier that takes both the context and target words as input and combines all useful information in the hidden layers. We also proposed a novel noise distribution based on translation probabilities to train the BNNJM effectively. With the improved noise sampling method, the BNNJM achieved comparable performance with the NNJM and even improved the translation results over the NNJM on Chinese-to-English and French-to-English translations.

[16]Infrequent words are usually content words and frequent words are usually function words.

5. A Neural Rule Selection Model for Tree-to-String MT

5.1 Introduction

In tree-to-string [36, 20] and forest-to-string [43] MT, a source tree or forest is used as input and translated into a target sentence by a series of tree-based translation rules, which can perform reordering and translation jointly. However, like other symbolic MT approaches, tree-to-string MT also has this ambiguity problem existing in translation rules and selecting appropriate rules during decoding is still a major challenge for tree-to-string MT.

Liu et al. [35] proposed a maximum entropy based rule selection (MERS) model for syntax-based MT, which used contextual information for rule selection, including both lexical and syntactic features.

We proposed a neural rule selection (NRS) model, which is learned by a feed-forward neural network and replaces the symbolic features used in the MERS model with distributed representations for better generalization.

In addition, we proposed a new method, applicable to both the MERS and NRS models, to train rule selection models only on minimal rules. These minimal rules are more frequent and have richer training data compared to non-minimal rules, making it possible to further relieve the data sparsity problem.

5.2 Related Work

The rule selection problem for tree-based MT (hierarchical phrase-based and syntax-based MT) has received much attention. He et al. [23] proposed a lexicalized rule selection model to perform context-sensitive rule selection for hierarchical phrase-base translation. Cui et al. [10] introduced a joint rule selection model for hierarchical phrase-based translation, which also approximated the rule selection problem by a binary classification problem like our approach. However, these two models adopted linear classifiers similar to those used in the MERS model [35], which suffers more from the data sparsity problem compared to the NRS model.

There are also existing works that exploited neural networks to learn transla-

tion probabilities for translation rules used in the phrase-based translation model. Namely, these methods estimated translation probabilities for phrase pairs extracted from the parallel corpus. Schwenk [54] proposed a continuous space translation model, which calculated the translation probability for each word in the target phrase and then multiplied the probabilities together as the translation probability of the phrase pair. Gao et al. [17] and Zhang et al. [73] proposed methods to learn distributed phrase representations and use the similarity between the source and target phrases as translation probabilities for phrase pairs. All these three methods can only be used for the phrase-based translation model, not for tree-based translation models.

There are also works that used minimal rules for modeling. Vaswani et al. [65] proposed a rule Markov model using minimal rules for both training and decoding to achieve a slimmer model, a faster decoder and comparable performance with using non-minimal rules. Durrani et al. [12] proposed a method to model with minimal translation units and decode with phrases for phrase-based MT to improve translation performances. Both of these two methods do not use distributed representations as used in our model for better generalization.

In addition, NMT has shown promising results recently [61, 4, 37, 25, 38]. And there are also other neural translation models [11, 42, 57]. All these models are trained on plain source and target sentences without considering any syntactic information while our neural model learns rule selection for tree-based translation rules and makes use of the tree structure of natural language for better translation. There is also a new syntactic NMT model [13], which extends the original sequence-to-sequence NMT model with the source-side phrase structure. Although this model takes source-side syntax into consideration, it still produces target words one by one as a sequence. In contrast, the tree-based translation rules used in our model can take advantage of the hierarchical structures of both source and target languages.

5.3 Maximum Entropy Based Rule Selection

Liu et al. [35] proposed the MERS model for syntax-based MT to perform context-dependent rule selection. They built a maximum entropy classifier for each ambiguous source subtree $\tilde{t}$, which introduced contextual information C and

estimated the conditional probability using a log-linear model as shown below,

$$\Pr\left(\tilde{e}|\tilde{t},C\right)=\frac{\exp\left(\sum_{k=1}^{K}\lambda_k h_k\left(\tilde{e},C\right)\right)}{\sum_{\tilde{e}'}\exp\left(\sum_{k=1}^{K}\lambda_k h_k\left(\tilde{e}',C\right)\right)}. \tag{25}$$

The target strings $\tilde{e}$ are treated as different classes for the classifier.

Supposing that,

- r covers source span $[f_\varphi, f_\vartheta]$ and target span $[e_\gamma, e_\sigma]$,
- $\tilde{t}$ contains K nonterminals $\{X_k|0 \leq k \leq K-1\}$,
- X_k covers source span $[f_{\varphi_k}, f_{\vartheta_k}]$ and target span $[e_{\gamma_k}, e_{\sigma_k}]$,

the MERS model used 5 kinds of source-side features as follows,

1. Lexical features: words around a rule (e.g. $f_{\varphi-1}$) and words covered by nonterminals in a rule (e.g. f_{φ_0}).
2. Part-of-speech features: part-of-speech (POS) of context words that are used as lexical features.
3. Span features: span lengths of source phrases covered by nonterminals in r.
4. Parent features: the parent node of $\tilde{t}$ in the parse tree of the source sentence.
5. Sibling features: the siblings of the root of $\tilde{t}$.

Note that the MERS model does not use features of the source subtree $\tilde{t}$, because the source subtree $\tilde{t}$ is fixed for each classifier.

The MERS model was integrated into the translation system as two additional features in Equation 26. Supposing that the derivation R contains M rules $r_1, ..., r_M$ with ambiguous source subtrees, then these two MERS features are as follows,

$$\begin{aligned} h_1\left(E,R,F\right) &= \sum_{m=1}^{M}\log\Pr\left(\tilde{e}_m|\tilde{t}_m,C_m\right) \\ h_2\left(E,R,F\right) &= M, \end{aligned} \tag{26}$$

where $\tilde{t}_m$ and $\tilde{e}_m$ are the source subtree and the target string contained in r_m, and C_m is the context of r_m. h_1 is the MERS probability feature, and, h_2 is a penalty feature counting the number of predictions made by the MERS model.

5.4 Neural Rule Selection

The proposed NRS model differs from the MERS model in three ways.

1. Instead of learning a single classifier for each source subtree $\tilde{t}$, it learns a single classifier for all rules.

2. Instead of hand-crafted features, it uses a feed-forward neural network to induce features from context words.

3. Instead of one-hot representations, it uses distributed representations to exploit similarities between words.

First, with regard to training, our NRS model follows the binarized NNJM (Section 4) in approximating the posterior probability by a binary classifier as follows,

$$\Pr\left(\tilde{e}|\tilde{t}, C\right) \approx \Pr\left(v = 1|\tilde{e}, \tilde{t}, C\right), \tag{27}$$

where $v \in \{0, 1\}$ is an indicator of whether $\tilde{t}$ is translated into $\tilde{e}$. This is in contrast to the MERS model, which treated the rule selection problem as a multi-class classification task. If instead we attempted to estimate output probabilities for all different $\tilde{e}$, the cost of estimating the normalization coefficient would be prohibitive, as the number of unique output-side word strings $\tilde{e}$ is large. There are a number of remedies to this, including noise contrastive estimation [66], but the binary approximation method has been reported to have better performance (see Section 4).

To learn this model, we use a feed-forward neural network with structure similar to neural network language models [66]. The input of the neural rule selection model is a vector representation for $\tilde{t}$, another vector representation for $\tilde{e}$, and a set of ξ vector representations for both source-side and target-side context words of r:

$$C(r) = w_1, ..., w_\xi \tag{28}$$

In our model, $C\left(r\right)$ is calculated differently depending on the number of non-terminals included in the rule. Specifically, Equation 29 defines $C_{out}\left(r, n\right)$ to be context words (n-grams) around r and $C_{in}\left(r, n, X_k\right)$ to be boundary words

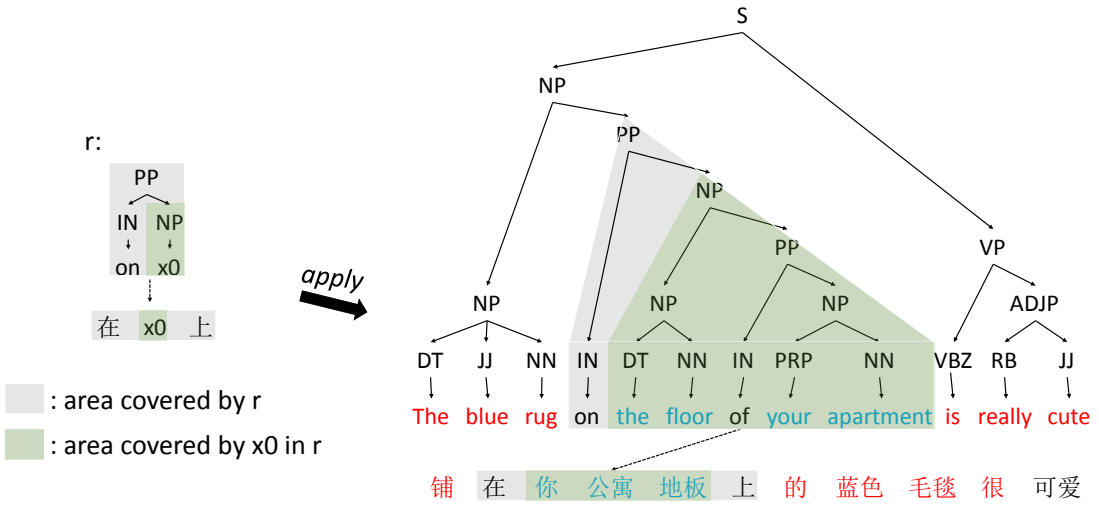

Figure 11. Context word examples. The red words are contained in $C_{out}(r,4)$ and the blue words are contained in $C_{in}(r,2,X_0)$.

(n-grams) covered by nonterminal X_k in r.[17]

$$\begin{aligned} &C_{out}(r,n) \\ &= f_{\varphi-n}^{\varphi-1}, f_{\theta+1}^{\theta+n}, e_{\gamma-n}^{\gamma-1}, e_{\sigma+1}^{\sigma+n} \\ &C_{in}(r,n,X_k) \\ &= f_{\varphi_k}^{\varphi_k+n-1}, f_{\theta_k-(n-1)}^{\theta_k}, e_{\gamma_k}^{\gamma_k+n-1}, e_{\sigma_k-(n-1)}^{\sigma_k} \end{aligned} \tag{29}$$

The context words used for a translation rule r with K nonterminals are shown as below.

K	$C(r)$
$=0$	$C_{out}(r,6)$
$=1$	$C_{out}(r,4), C_{in}(r,2,X_0)$
>1	$C_{out}(r,2), C_{in}(r,2,X_0), C_{in}(r,2,X_1)$

We can see that rules with different numbers of nonterminals K use different

[17]Note that when extracting C_{out}, we use "$\langle s\rangle$" and "$\langle/s\rangle$" for context words that exceed the length of the sentence; When extracting C_{in}, we use "$\langle non\rangle$" for context words that exceed the length of the nonterminal. Words that occur less than twice in the training data are replaced by "$\langle unk\rangle$".

context words.[18] For example, if r does not contain nonterminals, then C_{in} is not used. Besides, we use more context words surrounding the rule ($C_{out}(r, 6)$) for rules with $K = 0$ than rules that contain nonterminals ($C_{out}(r, 4)$ for $K = 1$ and $C_{out}(r, 2)$ for $K > 1$). This is based on the intuition that rules with $K = 0$ can only use the context words surrounding the rule as information for rule selection, hence this information is more important than for other rules. Figure 11 gives an example of context words when applying the rule r to the example sentence.

Note that we use target-side context because source-side context is not enough for selecting correct rules. Since it is not uncommon for one source sentence to have different correct translations, a translation rule used in one correct derivation may be incorrect for other derivations. In these cases, target-side context is useful for selecting appropriate translation rules. [19]

The vector representations for $\tilde{t}$, $\tilde{e}$ and C are obtained by using a projection matrix to project each one-hot input into a real-valued embedding vector. This projection is another key advantage over the MERS model. Because the NRS model learns one unified model for all rules and can share all training data to learn better vector representations of words and rules, and the similarities between vectors can be used for better generalization.

After calculating the projections, two hidden layers are used to combine all inputs. Finally, the neural network has two outputs $\Pr\left(v = 1|\tilde{e}, \tilde{t}, C\right)$ and $\Pr\left(v = 0|\tilde{e}, \tilde{t}, C\right)$.

To train the NRS model, we need both positive and negative training examples. Positive examples, $\left\langle \tilde{e}, \tilde{t}, C, 1 \right\rangle$, can be extracted directly from the parallel corpus. For each positive example, we generate one negative example, $\left\langle \tilde{e}', \tilde{t}, C, 0 \right\rangle$. Here, $\tilde{e}'$ is randomly generated according to the translation distribution proposed

[18] In most cases, restrictions on extracted rules will ensure that rules will only contain two nonterminals. However, when using minimal rules as described in the next section, more than two nonterminals are possible, and in these cases, only contextual information covered by the first two nonterminals is used in the input. These cases are sufficiently rare, however, that we chose to consider only the first two.

[19] It is also possible to consider target-side context in a framework like the MERS model, but we show in experiments that a linear model using the same features as the NRS model did not improve accuracy.

in Section 4,

$$\Pr\left(\tilde{e}|\tilde{t}\right) = \frac{Count\left(\tilde{e},\tilde{t}\right)}{\sum_{\tilde{e}'} Count\left(\tilde{e}',\tilde{t}\right)}, \tag{30}$$

where, $Count\left(\tilde{e},\tilde{t}\right)$ is how many times $\tilde{t}$ is translated into $\tilde{e}$ in the parallel corpus.

During translating, following the MERS model, the NRS model only calculates probabilities for rules with ambiguous source subtrees. These predictions are converted into two NRS features for the translation system similar to the two MERS features in Equation 26: one is the product of probabilities calculated by the NRS model and the other one is a penalty feature that stands for how many rules with ambiguous source subtrees are contained in one translation.

5.5 Usage of Minimal Rules

Despite the fact that the NRS model can share information among instances using distributed word representations, it still poses an extremely sparse learning problem. Specifically, the numbers of unique subtrees $\tilde{t}$ and strings $\tilde{e}$ are extremely large, and many may only appear a few times in the corpus. To reduce these problems of sparsity, we proposed another improvement to the model, specifically through the use of minimal rules.

Minimal rules [16] are translation rules that cannot be split into two smaller rules. For example, in Figure 12, Rule2 is not a minimal rule, since Rule2 can be split into Rule1 and Rule3. In the same way, Rule4 and Rule6 are not minimal while Rule1, Rule3 and Rule5 are minimal.

Minimal rules are more frequent than non-minimal rules and have richer training data. Hence, we can expect that a rule selection model trained on minimal rules will suffer less from data sparsity problems. Besides, without non-minimal rules, the rule selection model will need less memory and can be trained faster.

To take advantage of this fact, we train another version of the NRS model (NRS-MINI) over only minimal rules. The probability of a non-minimal rule is then calculated using the product of the probability of minimal rules contained therein.

Note that for both the standard NRS and NRS-MINI models, we use the same baseline translation system which can use non-minimal translation rules.

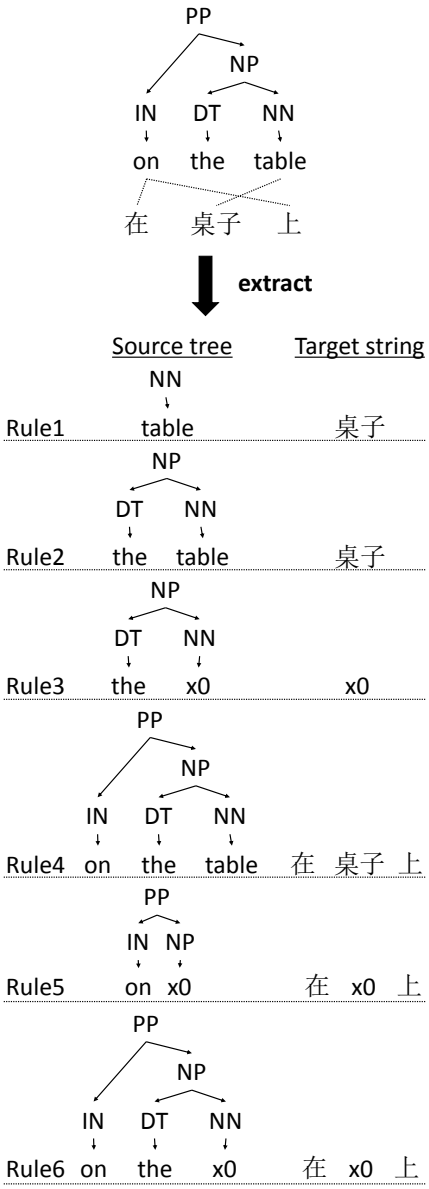

Figure 12. Examples of tree-to-string MT rules.

The NRS-MINI model will break translation rules used in translations down into minimal rules and multiply all probabilities to calculate the necessary features.

5.6 Experiments

5.6.1 Setting

We evaluated the proposed approach for English-to-German (en-de), English-to-French (en-fr), English-to-Chinese (en-zh) and English-to-Japanese (en-ja) translation tasks. For the en-de and en-fr tasks, the translation systems are trained on Europarl v7 parallel corpus and tested on the WMT 2015 translation task.[20] The test sets for the WMT 2014 translation task were used as development sets in our experiments. For the en-zh and en-ja tasks, we used datasets provided for the patent machine translation task at NTCIR-9 [19].[21] The detailed statistics for training, development and test sets are given in Table 21. The word segmentation was done by BaseSeg [75] for Chinese and Mecab[22] for Japanese.

For each translation task, we used Travatar [45] to train a forest-to-string translation system. GIZA++ [51] was used for word alignment. A 5-gram language model was trained on the target side of the training corpus using the IRST-LM Toolkit[23] with modified Kneser-Ney smoothing. Rule extraction was performed using the GHKM algorithm [15] and the maximum numbers of non-terminals and terminals contained in one rule were set to 2 and 10 respectively. Note that when extracting minimal rules, we release this limit. The decoding algorithm is the bottom-up forest-to-string decoding algorithm of Mi et al. [43]. For English parsing, we used Egret[24], which is able to output packed forests for decoding.

We trained the NRS models (NRS and NRS-MINI) on translation rules ex-

[20]The WMT tasks provided other training corpora. We used only the Europarl corpus, because training a large-scale system on the whole data set requires large amounts of time and computational resources.

[21]Note that NTCIR-9 only contained a Chinese-to-English translation task. Because we want to test the proposed approach with a similarly accurate parsing model across our tasks, we used English as the source language in our experiments. In NTCIR-9, the development and test sets were both provided for the zh-en task while only the test set was provided for the en-ja task. Therefore, we used the sentences from the NTCIR-8 en-ja and ja-en test sets as the development set in our experiments.

[22]http://sourceforge.net/projects/mecab/files/

[23]http://hlt.fbk.eu/en/irstlm

[24]https://code.google.com/archive/p/egret-parser

			SOURCE	TARGET
en-de	TRAIN	#Sents	1.90M	
		#Words	52.2M	49.7M
		#Vocab	113K	376K
	DEV	#Sents	3,003	
		#Words	67.6K	63.0K
	TEST	#Sents	2,169	
		#Words	46.8K	44.0K
en-fr	TRAIN	#Sents	1.99M	
		#Words	54.4M	60.4M
		#Vocab	114K	137K
	DEV	#Sents	3,003	
		#Words	71.1K	81.1K
	TEST	#Sents	1.5K	
		#Words	27.1K	29.8K
en-zh	TRAIN	#Sents	954K	
		#Words	40.4M	37.2M
		#Vocab	504K	288K
	DEV	#Sents	2K	
		#Words	77.5K	75.4K
	TEST	#Sents	2K	
		#Words	58.1K	55.5K
en-ja	TRAIN	#Sents	3.14M	
		#Words	104M	118M
		#Vocab	273K	150K
	DEV	#Sents	2K	
		#Words	66.5K	74.6K
	TEST	#Sents	2K	
		#Words	70.6K	78.5K

Table 21. Data sets.

tracted from the training set. Translation rules extracted from the development set were used as validation data for model training to avoid over-fitting. For different training epochs, we resample negative examples for each positive example to make use of different negative examples. The embedding dimension was set

	en-de	en-fr	en-zh	en-ja
Base	15.00	26.76	29.42	37.10
MERS	15.62	27.33	29.75	37.76
NRS	16.15	28.05	30.12	37.83
MERS-MINI	15.77	28.13	30.53	38.14
NRS-MINI	**16.49**	**28.30**	**31.63**	**38.32**

Table 22. Translation results. The bold numbers stand for the best systems.

	en-de	en-fr	en-zh	en-ja
NRS vs. MERS	>>	>>	>	–
NRS-MINI vs. MERS-MINI	>>	–	>>	–
MERS-MINI vs. MERS	–	>>	>>	>>
NRS-MINI vs. NRS	>	–	>>	>>

Table 23. Significance test results.

to be 50 and the number of hidden nodes was 100. The initial learning rate was set to be 0.1. The learning rate was halved each time the validation likelihood decreased. The number of epoches was set to be 20. A model was saved after each epoch and the model with highest validation likelihood was used in the translation system.

We implemented Liu et al. [35]'s MERS model to compare with our approach. The training instances for their model were extracted from the training set. Following their work, the iteration number was set to be 100 and the Gaussian prior was set to be 1. We also compared the original MERS model and the MERS model trained only on minimal rules (MERS-MINI) to test the benefit of using minimal rules for model training.

The MERS and NRS models were both used to calculate features used to rerank unique 1,000-best outputs of the baseline system. Tuning is performed to maximize BLEU score using minimum error rate training [49].

Source: typical dynamic response rate of an **optical gap sensor** as described above is approximately 2 khz , or 0.5 milliseconds .
Reference:上述(described above) **光学(optical) 间隙(gap) 传感器(sensor)** 的 典型(typical) 动态(dynamic) 响应(response) 率(rate) 约(approximately) 为(is) 2KHz 或(or) 为 0.5 毫秒(milliseconds) 。
T_{Base}**:** 典型(typical) 的 动态(dynamic) 响应(response) 速率(rate) **间隙(gap) 传感器(sensor)** 的 **光学(optical)** 如上(above) 描述(described) 的 是(is) 约(approximately) 2 千赫(khz) 兹 , 或(or) 0.5 毫秒(milliseconds) 。
T_{MERS}**:** 典型(typical) 的 动态(dynamic) 响应(response) 率(rate) **光学(optical) 传感器(sensor)** , 如(as) 以上(above) 所述(described) **间隙(gap)** 的 约(approximately) 2 千赫(khz) , 或(or) 0.5 毫秒(milliseconds) 。
T_{NRS}**:** **光学(optical) 传感器(sensor)** , 如(as) 以上(above) 所述(described) **间隙(gap)** 的 典型(typical) 的 动态(dynamic) 响应(response) 速率(rate) 为(is) 约(approximately) 2 千赫(khz) 兹 , 或(or) 0.5 毫秒(milliseconds) 。
$T_{MERS-MINI}$**:** 典型(typical) 的 动态(dynamic) 响应(response) 率(rate) **间隙(gap) 传感器(sensor)** 的 **光学(optical)** 如上(above) 描述(described) 的 是(is) 约(approximately) 2 千赫(khz) 兹 , 或(or) 0.5 毫秒(milliseconds) 。
$T_{NRS-MINI}$**:** 如上(above) 描述(described) 的 **光学(optical) 间隙(gap) 传感器(sensor)** 典型(typical) 的 动态(dynamic) 响应(response) 速率(rate) 为(is) 约(approximately) 2 千赫(khz) 兹 , 或(or) 0.5 毫秒(milliseconds) 。

Table 24. Translation examples.

5.6.2 Result and Analysis

Table 22 shows the translation results and Table 23 shows significance test results using bootstrap resampling[25] [28]: "Base" stands for the baseline system without any; "MERS", "NRS", "MERS-MINI" and "NRS-MINI" means the outputs of the baseline system were reranked using features from the MERS, NRS, MERS-MINI and NRS-MINI models respectively. Generally, the NRS model outperformed the MERS model and the NRS-MINI model outperformed the MERS-MINI model on different translation tasks. In addition, using minimal rules for model training benefitted both the MERS and NRS models.

[25]The symbol >> (>) represents a significant difference at the $p < 0.01$ ($p < 0.05$) level and the symbol - represents no significant difference at the $p < 0.05$ level.

R_1: T_{MERS}&T_{NRS}	PP (IN (“of”) NP (NP (DT (“an”) NP’ (JJ (“optical”) x0:NN)) x1:NP’)) → “光学(optical)” x1 x0 “的”
R_2: $T_{MERS-MINI}$	PP (IN (“of”) NP (NP (DT (“an”) NP’ (JJ (“optical”) x0:NP’)) x1:SBAR)) → x0 “的” “光学(optical)” x1 “的”
R_3: $T_{NRS-MINI}$	NP’ (JJ (“optical”) x0:NP’) → “光学(optical)” x0

Table 25. Rules used to translate the source word “optical” in different translations. Shadows (R_3) stand for ambiguous rules.

Table 24 shows translation examples in the en-zh task to demonstrate the reason why our approach improved accuracy. Among all translations, $T_{NRS-MINI}$ is basically the same as the reference with only a few paraphrases that do not alter the meaning of the sentence. In contrast, T_{Base}, T_{MERS}, T_{NRS} and $T_{MERS-MINI}$ all contain apparent mistakes. For example, the source phrase “optical gap sensor” (covered by gray shadows in Table 24) is wrongly translated in T_{Base}, T_{MERS}, T_{NRS} and $T_{MERS-MINI}$ due to incorrect reorderings.

Table 25 shows rules used to translate the source word “optical” in different translations: R_1 is used in T_{MERS} and T_{NRS}; R_2 is used in $T_{MERS-MINI}$; R_3 is used in $T_{NRS-MINI}$. Although the source word “optical” is translated to the correct translation “光学(optical)” in all translations, R_1, R_2 and R_3 cause different reorderings for the source phrase “optical gap sensor”. R_3 reorders this source phrase correctly while R_1 and R_2 cause wrong reorderings for this source phrase.

We can see that R_1 is unambiguous, so the MERS and NRS models will give probability 1 to R_1, which could make the MERS and NRS models prefer T_{MERS} and T_{NRS}. This is a typical translation error caused by sparse rules since the source subtree in R_1 does not have other translations in the training corpus.

To compare the MERS-MINI and NRS-MINI models, Table 26 shows minimal rules (R_{2a}, R_{2b}, R_{3a} and R_{3b}) contained in R_2 and R_3. Table 27 shows probabilities of these minimal rules calculated by the MERS-MINI and NRS-MINI models respectively. We can see that the NRS-MINI model gave higher scores for the correct translation rules R_{3a} and R_{3b} than the MERS-MINI model, while the MERS-MINI model gave a higher score to the incorrect rule R_{2b} than the NRS-MINI model.

R_{2a}	PP (IN (“of”) NP (NP (DT (“an”) NP’ (x0:JJ x1:NP’)) x2:SBAR)) → x1 “的” x0 x2 “的”
R_{2b}	JJ (“optical”) → “光学(optical)”
R_{3a}	NP’ (x0:JJ x1:NP’) → x0 x1
R_{3b}	JJ (“optical”) → “光学(optical)”

Table 26. Minimal rules contained in R_2 and R_3. Shadows (R_{2b}, R_{3a} and R_{3b}) stand for ambiguous rules.

	MERS-MINI	NRS-MINI
R_{2a}	1	1
R_{2b}	0.5441	0.09632
R_{3a}	0.9943	0.9987
R_{3b}	0.5441	0.7317

Table 27. Scores of minimal rules.

Note that R_{2b} and R_{3b} are the same rule, but the target-side context in $T_{MERS-MINI}$ and $T_{NRS-MINI}$ is different. The NRS-MINI model will give R_{2b} and R_{3b} different scores because the NRS-MINI model used target-side context. However, the MERS-MINI model only used source-side features and gave R_{2b} and R_{3b} the same score. The fact that the NRS-MINI model gave a higher score for R_{3b} than R_{2b} means that the NRS-MINI model predicted the target string in R_{2b} and R_{3b} is a good translation in the context of $T_{NRS-MINI}$ but not so good in the context of $T_{MERS-MINI}$. As we can see, the target phrase “如上(above) 描述(described) 的(of) 光学(optical) 间隙(gap) 传感器(sensor)” around “光学(optical)” in $T_{NRS-MINI}$ is a reasonable Chinese phrase while the target phrase “间隙(gap) 传感器(sensor) 的(of) 光学(optical) 如上(above) 描述(described) 的(of)” around “光学(optical)” in $T_{MERS-MINI}$ does not make sense. Namely, the NRS model trained with target-side context can perform rule selection considering target sentence fluency, which is the reason why target-side context can help in the rule selection task.

To analyze the influence of different features, we trained the MERS model using source-side and target-side n-gram lexical features similar to the NRS model.

When using this feature set, the performance of the MERS model dropped significantly. This indicates that the syntactic, POS and span features used in the original MERS model are important for their model, since these features can generalize better. Purely lexical features are less effective due to sparsity problems when training one maximum entropy based classifier for each ambiguous source subtree and training data for each classifier is quite limited. In contrast, the NRS model is trained with distributed representations and does not split training data, which relieves the sparsity problem of lexical features. As a result, the NRS model achieved better performance using only lexical features compared to the MERS model. We also tried to use pre-trained word embedding features for the MERS model, but it did not improve the performance of the MERS model, which indicates that the log-linear model is not able to benefit from distributed representations as well as the non-linear neural network model.

We also tried reranking with both the NRS and MERS models added as features, but it did not achieve further improvement compared to only using the NRS model. This indicates that although these two models use different type of features, the information contained in these features are similar. For example, the POS features used in the MERS model and the distributed representations used in the NRS model are both used for better generalization.

In addition, using both the NRS and NRS-MINI models did not improve over using only the NRS-MINI model in our experiments. There are two main differences between the NRS and NRS-MINI models. First, minimal rules are more frequent and have more training data than non-minimal rules, which is why the NRS-MINI model is more robust than the NRS model. Second, non-minimal rules contain more information than minimal rules. For example, in Figure 33, Rule4 contains more information than Rule1, which could be an advantage for rule selection. However, the information contained in Rule4 will be considered as context features for Rule1. Therefore, this is no longer an advantage for the NRS model as long as we use rich enough context features, which could be the reason why using both the NRS and NRS-MINI models cannot further improve the translation quality compared to using only the NRS-MINI model.

5.7 Conclusion

We proposed an NRS model for syntax-based MT, which is learned by a feed-forward neural network with distributed representations and hence can generalize better compared to the previous MERS model that uses symbolic features. In addition, we proposed to use only minimal rules for rule selection to further relieve the data sparsity problem, since minimal rules are more frequent and have richer training data. In our experiments, the NRS model outperformed the previous MERS model and the usage of minimal rules benefitted both NRS and MERS models on different translation tasks.

6. Improving NMT through Phrase-based Forced Decoding

6.1 Introduction

Neural machine translation (NMT) [4], which uses a single large neural network to model the entire translation process, has recently been shown to outperform symbolic MT such as phrase-based MT [32] on several translation tasks [55, 68]. Compared to symbolic MT, NMT generally produces more fluent translations, but often sacrifices adequacy, such as translating source words into completely unrelated target words, over-translation or under-translation [31].

There are a number of methods that combine the two paradigms to address their respective weaknesses. For example, it is possible to incorporate NMT as features into symbolic MT to disambiguate hypotheses [46, 60]. However, the search space of symbolic MT is usually limited by translation rule tables, reducing the ability of symbolic MT to generate hypotheses on the same level of fluency as NMT, even after reranking.

There are also methods that incorporate knowledge from symbolic MT into NMT, such as lexical translation probabilities [3, 22], phrase memory [62], and n-gram posterior probabilities based on symbolic MT translation lattices [59]. These improve the adequacy of NMT output, but do not impose hard alignment constraints like symbolic MT systems and therefore cannot effectively solve all over-translation or under-translation problems.

We proposed a method that exploits an existing phrase-based translation model to compute the phrase-based decoding cost for a given NMT translation.[26] That is, we force a phrase-based translation system to take in the source sentence and generate an NMT translation. Then we use the cost of this phrase-based forced decoding to rerank the NMT outputs. The phrase-based decoding cost will heavily punish completely unrelated translations, over-translations, and under-translations, as they will not be able to be found in the translation phrase table.

[26] In fact, our method can take in the output of *any* up-stream system, but we experiment exclusively with using it to rerank NMT output.

One challenge in implementing this method is that the NMT output may not be in the search space of the phrase-based translation model, which is limited by the phrase-based translation rule table. To solve this problem, we proposed a soft forced decoding algorithm, which is based on the standard phrase-based decoding algorithm and integrates new types of translation rules (deleting a source word or inserting a target word). The proposed forced decoding algorithm can always successfully find a decoding path and compute a phrase-based decoding cost for any NMT output. Another challenge is that we need a diverse NMT n-best list for reranking. Because beam search for NMT often lacks diversity in the beam – candidates only have slight differences, with most of the words overlapping – we used a random sampling method to obtain a more diverse n-best list.

We tested the proposed method on English-to-Chinese, English-to-Japanese, English-to-German and English-to-French translation tasks, obtaining large improvements over a strong NMT baseline that already incorporates symbolic lexicon features.

6.2 Related Work

Wuebker et al. [70, 69] applied forced decoding on the training set to improve the training process of phrase-based MT and prune the phrase-based rule table. They also used word insertions and deletions for forced decoding, but they used a high penalty for all insertions and deletions. In contrast, our forced decoding algorithm for NMT outputs uses a small penalty for function words and a high penalty for content words, because function words are usually translated very flexibly and more likely to be inserted or deleted. The different penalties come from that our method is used for reranking NMT outputs while their method was used for improving the training process of phrase-based MT.

A major difference of symbolic MT and NMT is that the alignment model in symbolic MT provides exact word or phrase level alignments between the source and target sentences while the attention model in NMT only computes an alignment probability distribution for each target word over all source words, which is the main reason why NMT is more likely to produce completely unrelated translations, over-translation or under-translation compared to symbolic MT. To relieve these problems of NMT, there are methods that modify the NMT neural network

structure [64, 41, 1] while we rerank NMT outputs by exploiting knowledge from symbolic MT.

There are also existing methods that rerank NMT outputs by using target-bidirectional NMT models [34]. Their reranking method aims to overcome the issue of unbalanced accuracy in NMT outputs while our reranking method aims to solve the inadequacy problem of NMT.

6.3 Phrase-based Forced Decoding for NMT

As stated before, our goal is not to generate new hypotheses with phrase-based MT, but instead use the phrase-based model to calculate scores for NMT output. In order to do so, we can perform *forced decoding*, which is very similar to the standard phrase-based decoding algorithm but discards all partial hypotheses that do not match the NMT output. However, the NMT output is not limited by the phrase-based rule table, so there may be no decoding path that completely matches the NMT output when using only the phrase-based rules.

To remedy this problem, inspired by previous work in forced decoding for training phrase-based MT systems [70, 69] we proposed a soft forced decoding algorithm that can always successfully find a decoding path for a source sentence F and an NMT translation E.

First, we introduce two new types of rules R_1 and R_2.

$\mathbf{R}_1$ A source word f can be translated into a special word `null`. This corresponds to deleting f during translation. The score of deleting f is calculated as,

$$s\left(f \rightarrow null\right) = \frac{unalign\left(f\right)}{|\mathcal{T}|} \tag{31}$$

where $unalign\left(f\right)$ is how many times f is unaligned in the word-aligned training set $\mathcal{T}$ and $|\mathcal{T}|$ is the number of sentence pairs in $\mathcal{T}$.

$\mathbf{R}_2$ A target word e can be translated from a special word `null`, which corresponds to inserting e during translation. The score of inserting e is calculated as,

$$s\left(null \rightarrow e\right) = \frac{unalign\left(e\right)}{|\mathcal{T}|} \tag{32}$$

where $unalign\,(e)$ is how many times e is unaligned in $\mathcal{T}$.

One motivation for Equations 31 and 32 is that function words usually have high frequencies, but do not have as clear a correspondence with a word in the other language as content words. As a result, function words are more often unaligned than content words in the training set. As an example, Table 28 shows the occurring and unaligned times for different words counted on the word-aligned training set of English-to-Chinese task in our experiments, which contains 953K sentence pairs. Based on the equations above, the scores of deleting or inserting "of" and "a" will be higher.

Words	of	a	temperature	water
Occur	1,352K	1,019K	35K	29K
Unaligned	513K	412K	4K	3K

Table 28. Number of times words occur in the training corpus and number of times unaligned.

In our forced decoding, we choose to model the score of each translation rule that exists in the phrase table as the product of direct and inverse phrase translation probabilities.[27] To make sure that the scale of the scores for R_1 and R_2 match the other phrase (which are the product of two probabilities), we use the square of the score in Equation 31/32 as the rule score for R_1/R_2.

Algorithm 3 shows the forced decoding algorithm that integrates the new rules. Because the translation E is given for the forced decoding algorithm, the proposed forced decoding algorithm keeps J stacks, where J is the length of E. In other words, the stack size is corresponding to the target word size during forced decoding while the stack size is corresponding to the source word size during standard phrase-based decoding. The stack s'_j in Algorithm 3 contains all hypotheses in which the first j target words have been generated. We expand hypotheses in $s'_1, s'_2, ..., s'_J$ in turn. When expanding a hypothesis H_{old} in s'_j, besides expanding it using the original rule table $\textsc{Expand}(H_{old})$, we also expand

[27] In standard phrase-based decoding it is common to integrate reordering probabilities in the decoding cost. However, because NMT generally produces more properly ordered sentences than symbolic MT, in this work we do not consider reordering probabilities in our forced decoding algorithm.

H_{old} by inserting the next target word e_{j+1} at the end of H_{old} to get an additional hypothesis H_{new} and put H_{new} into s'_{j+1}. For a final hypothesis in stack s'_J, it may not cover all source words. We update its score by translating uncovered words into `null`.

Algorithm 3 Forced phrase-based decoding.

Require: Source sentence F with length I and translation E with length J
Ensure: Decoding path D
  initialize H_0 and $s'_1, s'_2, ..., s'_J$
  EXPAND(H_0)
  expand H_0 with rule `null`$\rightarrow e_1$
  for $j = 1$ to $J - 1$ **do**
    for each hypothesis H_{jk} in s'_j **do**
      EXPAND(H_{jk})
      expand H_{jk} with rule `null`$\rightarrow e_{j+1}$
    end for
  end for
  for each hypothesis H_{Jk} in s'_J **do**
    updata $S(H_{Jk})$ for uncovered source words
  end for
  select best hypothesis in s'_J

Because different decoding paths can generate the same final translation, there can be different decoding paths that fit the NMT translation E. We use the score of the single decoding path with the highest decoding score as the forced decoding score for E.

6.4 Reranking NMT Outputs

We reranked the n-best NMT outputs using Equation 33.

$$\log P(E|F) = w_1 \cdot \log P_n(E|F) + w_2 \cdot \log S_d(E|F) \tag{33}$$

where $P_n(E|F)$ is the original NMT translation probability, $S_d(E|F)$ is the forced decoding probability, w_1 and w_2 are weights that can be tuned on the n-best list of the development set.

The easiest way to get an n-best list for NMT is by using the n-best translations from beam search, which is the standard decoding algorithm for NMT. While beam search is likely to find the highest-scoring hypothesis, it often lacks diversity in the beam: candidates only have slight differences, with most of the words overlapping. In order to obtain a more diverse list of hypotheses for reranking, we additionally augment the 1-best hypothesis discovered by beam search with translations sampled from the NMT conditional probability distribution.

The standard method for sampling hypotheses in NMT is ancestral sampling, where we randomly select a word from the vocabulary according to the NMT probability distribution $P\left(e_j|e_1^{j-1},F\right)$ [58]. This will make a diverse list of hypotheses, but may reduce the probability of selecting a highly scoring hypothesis, and the whole n-best list may not contain any translation that is better than the standard beam search output.

Instead, we take an alternative approach that proved empirically better in our experiments: at each time step j, we use sampling to randomly select the next word from e' and e'' according to Equation 34. Here, e' and e'' are the two target words with the highest probability according to Equation 8.

$$\begin{aligned} P_{rdm}\left(e'\right) &= \frac{P\left(e'|e_1^{j-1},F\right)}{P\left(e'|e_1^{j-1},F\right)+P\left(e''|e_1^{j-1},F\right)} \\ P_{rdm}\left(e''\right) &= \frac{P\left(e''|e_1^{j-1},F\right)}{P\left(e'|e_1^{j-1},F\right)+P\left(e''|e_1^{j-1},F\right)} \end{aligned} \tag{34}$$

The sampling process ends when $\langle/s\rangle$ is selected as the next word.

We repeat the decoding process $1,000$ times to sample $1,000$ outputs for each source sentence. We additionally add the 1-best output of standard beam search, making the size of the list used for reranking is $1,001$.

6.5 Experiments

6.5.1 Setting

We evaluated the proposed approach for English-to-Chinese (en-zh), English-to-Japanese (en-ja), English-to-German (en-de) and English-to-French (en-fr) translation tasks. For the en-zh and en-ja tasks, we used datasets provided for the

			SOURCE	TARGET
en-de	TRAIN	#Sents	1.90M	
		#Words	52.2M	49.7M
		#Vocab	113K	376K
	DEV	#Sents	3,003	
		#Words	67.6K	63.0K
	TEST	#Sents	2,169	
		#Words	46.8K	44.0K
en-fr	TRAIN	#Sents	1.99M	
		#Words	54.4M	60.4M
		#Vocab	114K	137K
	DEV	#Sents	3,003	
		#Words	71.1K	81.1K
	TEST	#Sents	1.5K	
		#Words	27.1K	29.8K
en-zh	TRAIN	#Sents	954K	
		#Words	40.4M	37.2M
		#Vocab	504K	288K
	DEV	#Sents	2K	
		#Words	77.5K	75.4K
	TEST	#Sents	2K	
		#Words	58.1K	55.5K
en-ja	TRAIN	#Sents	3.14M	
		#Words	104M	118M
		#Vocab	273K	150K
	DEV	#Sents	2K	
		#Words	66.5K	74.6K
	TEST	#Sents	2K	
		#Words	70.6K	78.5K

Table 29. Data sets.

patent machine translation task at NTCIR-9 [19].[28] For the en-de and en-fr tasks,

[28]Note that NTCIR-9 only contained a Chinese-to-English translation task, we used English as the source language in our experiments. In NTCIR-9, the development and test sets were both provided for the zh-en task while only the test set was provided for the en-ja task. We

	en-zh		en-ja		en-de		en-fr	
	dev	test	dev	test	dev	test	dev	test
PBMT	30.73	27.72	35.67	33.46	12.37	13.95	25.96	27.50
NMT	34.60	32.71	41.67	39.00	12.52	14.05	23.63	23.99
NMT+lex	36.06	34.80	44.47	41.09	13.36	15.60	24.00	24.91
Our(P_n)	34.38	33.23	38.92	34.18	12.34	13.59	23.13	23.61
Our(S_d)	36.17	34.09	42.91	40.16	13.08	15.29	24.28	25.71
Our(P_n+S_d)	37.94	**35.59**	45.34	41.75	**14.56**	**16.61**	**25.96**	**27.12**
Our(P_n+WP)	37.44	34.93	45.81	41.90	13.75	15.46	24.47	25.09
Our(S_d+WP)	36.44	33.73	43.52	40.49	13.39	15.71	24.74	26.25
Our(P_n+S_d+WP)	**38.69**	**35.75**	**46.92**	**43.17**	**14.61**	**16.65**	**25.98**	**27.15**

Table 30. Translation results (BLEU). NMT+lex: [3]; Our: NMT+lex+rerank, i.e. we rerank the n-best outputs of [3] using different features (P_n, S_d and WP).

we used version 7 of the Europarl corpus as training data, WMT 2014 test sets as our development sets and WMT 2015 test sets as our test sets. The detailed statistics for training, development and test sets are given in Table 29. The word segmentation was done by BaseSeg [75] for Chinese and Mecab[29] for Japanese.

We built attentional NMT systems with Lamtram[30]. Word embedding size and hidden layer size are both 512. We used Byte-pair encoding (BPE) [56] and set the vocabulary size to be 50K. We used the Adam algorithm for optimization.

To obtain a phrase-based translation rule table for our forced decoding algorithm, we used GIZA++ [51] and *grow-diag-final-and* heuristic [32] to obtain symmetric word alignments for the training set. Then we extracted the rule table using Moses [30].

6.5.2 Result and Analysis

Table 30 shows results of the phrase-based MT system[31], the baseline NMT system, the lexicon integration method [3] and the proposed reranking method. We

used the sentences from the NTCIR-8 en-ja and ja-en test sets as the development set in our experiments.

[29]http://sourceforge.net/projects/mecab/files/

[30]https://github.com/neubig/lamtram

[31]We used the default Moses settings for phrase-based MT.

	en-zh		en-ja		en-de		en-fr	
	dev	test	dev	test	dev	test	dev	test
PBMT	1.008	1.018	1.005	0.998	1.077	1.069	0.986	1.004
NMT	0.953	0.954	0.960	0.961	1.059	1.038	0.985	0.977
NMT+lex	0.936	0.966	0.955	0.963	1.054	1.019	1.030	0.977
Our(P_n)	0.875	0.898	0.814	0.775	0.874	0.854	0.904	0.900
Our(S_d)	0.973	0.989	0.985	0.981	1.062	1.060	1.030	1.031
Our(P_n+S_d)	0.949	0.965	0.945	0.936	1.000	0.992	0.999	0.992
Our(P_n+WP)	0.996	1.019	0.999	0.983	1.000	0.975	0.998	1.001
Our(S_d+WP)	1.000	1.024	1.001	1.001	1.011	1.007	0.999	0.989
Our(P_n+S_d+WP)	0.990	1.014	1.000	0.986	1.000	0.989	1.000	0.992

Table 31. Ratio of translation length to reference length for different system outputs in Table 30.

tested three features for reranking: the NMT score P_n, the forced decoding score S_d and a word penalty (WP) feature, which is the length of the translation. The best NMT system and the systems that have no significant difference from the best NMT system at the $p < 0.05$ level using bootstrap resampling [28] are shown in bold font.

As we can see, integrating lexical translation probabilities improved the baseline NMT system and reranking with both three features achieved further improvements for all four language pairs. Even on English-to-Chinese and English-to-Japanese tasks, where the NMT system outperformed the phrase-based MT system 7-8 BLEU scores, using the forced decoding score for reranking NMT outputs can still achieve significant improvements. With or without the word penalty feature, using both P_n and S_d for reranking gave better results than only using P_n or S_d alone.

The Word Penalty Feature The word penalty feature generally improved the reranking results, especially when only the NMT score P_n was used for reranking. As we can see, using only P_n for reranking decreased the translation quality compared to the standard beam search result of NMT. This is because the search spaces of beam search and random sampling are quite different, the best beam search output does not necessarily have the highest NMT score compared to

Source for hypophysectomized (hypop hy sec to mized) rats , the drinking water additionally contains 5 % glucose .
Reference 对于(for) 去(remove) 垂体(hypophysis) 大(big) 鼠(rat) ， 饮用水(drinking water) 中(in) 另外(also) 含有(contain) 5 % 葡萄糖(glucose) 。
PBMT 用 于(for) 大(big) 鼠(rat) 垂体(hypophysis) HySecto，(Hy Sec to ，) 饮 用 水(drinking water) 另外(also) 含有(contain) 5 % 葡萄糖(glucose) 。
NMT 对于(for) 过(pass) 盲肠(cecum) 的(of) 大(big) 鼠(rat) ， 饮用水(drinking water) 另外(also) 含有(contain) 5 % 葡萄糖(glucose) 。
NMT+lex/NMT+lex+P_n/NMT+lex+P_n+WP 对于(for) 低(low) 酪(cheese) 蛋白(protein) 切除(remove) 的(of) 大(big) 鼠(rat) ，饮用水(drinking water) 另外(also) 含有(contain) 5 % 葡萄糖(glucose) 。
NMT+lex+S_d/NMT+lex+S_d+WP 对 于(for) 垂体(hypophysis) 在(is) 切除(remove) 大(big) 鼠(rat) 中(in) ， 饮 用 水(drinking water) 另外(also) 含有(contain) 5 % 葡萄糖(glucose) 。
NMT+lex+P_n+S_d/NMT+lex+P_n+S_d+WP 对于(for) 垂体(hypophysis) 在(is) 切除(remove) 的(of) 大(big) 鼠(rat) 中(in) ， 饮用水(drinking water) 另外(also) 含有(contain) 5 % 葡萄糖(glucose) 。

Table 32. An example of improving inaccurate rare word translation by using S_d for reranking.

random sampling outputs. Therefore, even the P_n reranking results do have higher NMT scores, but have lower BLEU scores according to Table 30. To explain why this happened, we show the ratio of translation length to reference length in Table 31. As we can see, the P_n reranking outputs are much shorter. This is because NMT generally prefers shorter translations, because Equation 12 multiplies all target word probabilities together. So the word penalty feature can improve the P_n reranking results a lot, by preferring longer sentences. Because the forced decoding score S_d does not obviously prefer shorter or longer sentences, so when S_d was used for reranking, the word penalty feature became less helpful. When both P_n and S_d were used for reranking, the word penalty feature only

achieved further significant improvement on the English-to-Japanese task.

Table 32 gives translation examples of our reranking method from the English-to-Chinese task. The source English word "hypophysectomized" is an unknown word which does not occur in the training set. By doing BPE, this word is split into "hypop", "hy", "sec", "to" and "mized". The correct translation for "hypophysectomized" is "去(remove) 垂体(hypophysis)" as shown in the reference sentence. The original attentional NMT translated it into incorrect translation "过(pass) 盲肠(cecum)". After integrating lexicons, the NMT system translated it into "低(low) 酪(cheese) 蛋白(protein) 切除(remove)". The last word "切除(remove)" is correct, but the rest of the translation is still wrong. Only by using the forced decoding score S_d for reranking, we get the more accurate translation "垂体(hypophysis) 在(is) 切除(remove)".

To further demonstrate how the reranking method works, Table 33 shows translation rules and their log-scores contained in the forced decoding paths found for T_1, the NMT translation without reranking and T_2, the NMT translation using both P_n and S_d for reranking. As we can see, the four rules r_a, r_b, r_c and r_d used for T_1 have low scores. r_a is an unlikely translation. In r_b, r_c and r_d, "酪(cheese)", "蛋白(protein)" and "hypop" are content words, which are unlikely to be deleted or inserted during translation. Table 33 also shows that the translation of function words is very flexible. The score of inserting a function word "的(of)" is very high. The translation rule "the →在(is)" used for T_2 is incorrect, but its score is relatively high, because function words are often incorrectly aligned in the training set. The reason why function words are more likely to be incorrectly aligned to each other is that they usually have high frequencies and do not have clear correspondences between different languages.

In T_1, "hypophysectomized (hypop hy sec to mized)" is incorrectly translated into "低(low) 酪(cheese) 蛋白(protein) 切除(remove)". However, from Table 33, we can see that the forced decoding algorithm learns it as unlikely translation (hy→低(low)), over-translation (`null`→酪(cheese), `null`→蛋白(protein)) and under-translation (hypop→`null`, sec→`null`), because there is no translation rule between "hypop" "sec" and "酪(cheese)" "蛋白(protein)". Because content words are unlikely to be deleted or inserted during translation, they have low forced decoding scores. So using the forced decoding score for reranking NMT

T_1 (NMT+lex):	
for →对于(for)	-3.04
r_a: hy →低(low)	-12.19
r_b: `null`→酪(cheese)	-21.99
r_c: `null`→蛋白(protein)	-13.83
to mized →切除(remove)	-6.22
`null`→的(of)	-1.53
rats →大(big) 鼠(rat)	-1.52
, the drinking water →， 饮用水(drinking water)	-1.38
additionally contains →另外(also) 含有(contain)	-3.68
5 % →5 %	-0.51
glucose . →葡萄糖(glucose) 。	-0.60
r_d: hypop→`null`	-25.33
sec→`null`	-20.66
T_2 (NMT+lex+P_n+S_d):	
for →对于(for)	-3.04
hypop hy →垂体(hypophysis)	-5.09
the →在(is)	-5.32
to mized →切除(remove)	-6.22
`null`→的(of)	-1.53
rats →大(big) 鼠(rat)	-1.52
, →中(in) ,	-4.11
drinking water →饮用水(drinking water)	-1.03
additionally contains →另外(also) 含有(contain)	-3.68
5 % →5 %	-0.51
glucose . →葡萄糖(glucose) 。	-0.60
sec→`null`	-20.66

Table 33. Forced decoding paths for T_1 and T_2: used rules and log scores. The translation rules with shade are used only for T_1 or T_2.

outputs can naturally improve over-translation or under-translation as shown in Table 34. As we can see, without using S_d for reranking, NMT under-translated "temperature" and over-translated "ph" twice, which will be assigned low scores by forced decoding. By using S_d for reranking, the correct translation was se-

lected.

Source: such changes in reaction conditions include , but are not limited to , **an increase in temperature or change in ph** .
Reference: 所(such) 述(said) 反应(reaction) 条件(condition) 的(of) 改变(change) 包括(include) 但(but) 不(not) 限于(limit) **温度(temperature) 的(of) 增加(increase) 或(or) pH 值(value) 的(of) 改变(change)** 。
PBMT: 中(in) 的(of) 这种(such) 变化(change) 的(of) 反应(reaction) 条件(condition) 包括(include) , 但(but) 不(not) 限于(limit) , **增加(increase) 的(of) 温度(temperature) 或(or) pH 变化(change)** 。
NMT: 这种(such) 反应(reaction) 条件(condition) 的(of) 变化(change) 包括(include) 但(but) 不(not) 限于(limit) **pH 或(or) pH 的(of) 变化(change)** 。
NMT+lex/NMT+lex+P_n: 这种(such) 反应(reaction) 条件(condition) 的(of) 变化(change) 包括(include) , 但(but) 不(not) 限于(limit) , **pH 的(of) 升高(increase) 或(or) pH 变化(change)** 。
NMT+lex+S_d: 这种(such) 反应(reaction) 条件(condition) 的(of) 变化(change) 包括(include) 但(but) 不(not) 限于(limit) , **温度(temperature) 的(of) 升高(increase) 或(or) 改变(change) pH 值(value)** 。
NMT+lex+P_n+S_d: 这种(such) 反应(reaction) 条件(condition) 的(of) 变化(change) 包括(include) , 但(but) 不(not) 限于(limit) , **温度(temperature) 的(of) 升高(increase) 或(or) 改变(change) pH 值(value)** 。
NMT+lex+P_n+WP: 这种(such) 反应(reaction) 条件(condition) 的(of) 变化(change) 包括(include) , 但(but) 不(not) 限于(limit) , **pH 的(of) 升高(increase) 或(or) 改变(change) pH 值(value)** 。
NMT+lex+S_d+WP/ NMT+lex+P_n+S_d+WP: 这种(such) 反应(reaction) 条件(condition) 的(of) 变化(change) 包括(include) , 但(but) 不(not) 限于(limit) , **温度(temperature) 的(of) 升高(increase) 或(or) 改变(change) pH 值(value)** 。

Table 34. An example of improving under-translation and over-translation by using S_d for reranking.

Reranking PBMT outputs with NMT We also did experiments that use the NMT score as an additional feature to rerank PBMT outputs (unique $1,000$-best list). The results are shown in Table 35. We also copy results of baseline PBMT and NMT from Table 30 for direct comparison. As we can see, using

NMT to rerank PBMT outputs achieved improvements over the baseline PBMT system. However, when the baseline NMT system is significantly better than the baseline PBMT system (en-zh, en-ja), even using NMT to rerank PBMT outputs still achieved lower translation quality compared to the baseline NMT system.

		en-zh	en-ja	en-de	en-fr
PBMT+rerank		32.77	37.68	**14.23**	**28.86**
PBMT	dev	30.73	35.67	12.37	25.96
NMT		**34.60**	**41.67**	12.52	23.63
PBMT+rerank		30.04	35.14	**15.89**	**29.77**
PBMT	test	27.72	33.46	13.95	27.50
NMT		**32.71**	**39.00**	14.05	23.99

Table 35. Results of using NMT for reranking PBMT outputs.

6.6 Conclusion

We proposed to exploit an existing phrase-based MT model to compute the phrase-based decoding cost for NMT outputs and then use the phrase-based decoding cost to rerank the n-best NMT outputs, so that we can combine the advantages of both PBMT and NMT. Because an NMT output may not be in the search space of standard phrase-based MT, we propose a soft forced decoding algorithm, which can always successfully find a decoding path for any NMT output by deleting source words and inserting target words. Results showed that using the forced decoding cost to rerank NMT outputs improved translation quality on four different language pairs.

7. Conclusions

This thesis started in 2015 when symbolic MT had the state-of-the-art translation performance and continues until now when NMT outperforms symbolic MT on various translation tasks [55, 68]. Compared to symbolic MT, NMT uses distributed word representations and generally generates more fluent translations, but often sacrifices adequacy, such as NMT is more likely to generate over, under or completely unrelated translations [31].

Both fluency and adequacy are important for the translation quality: fluency makes the translations understandable for human and adequacy makes sure the translations contain the same information as the original source sentences. The goal of machine translation is to replace human translation and generate good translations that are both fluent and adequate. Because NMT and symbolic MT have their own respective advantages, i.e. NMT is better at fluency while symbolic MT is better at adequacy, this thesis focuses on hybrid neural-symbolic MT to combine the advantages of both symbolic and neural MT for generating translations that are both fluent and adequate. The contribution of this thesis mainly consists of two parts as below.

First, we developed various neural models for the log-linear framework of symbolic MT to improve the fluency of symbolic MT. In particular, we proposed: a neural word reordering model which learned word reorderings more effectively and efficiently compared to previous word reordering models by exploiting a feed-forward neural network and selecting useful reordering information, i.e. short reorderings; a binarized NNJM which converted the multiclass classification problem of the original NNJM into a binary classification problem and outperformed the NNJM by adopting an effective noise sampling method based on translation probabilities for model training; a neural rule selection model which performed rule selection based on minimal translation rules for reducing data sparsity and outperformed previous rule selection models by learning distributed representations for both translation rules and context words. All these three proposed neural models were proved to be useful and improved the translation quality, specially the fluency of symbolic MT significantly.

Second, we exploited knowledge from symbolic MT to improve the adequacy of NMT. We proposed to perform phrase-based forced decoding for NMT out-

puts and rerank NMT outputs with phrase-based decoding scores. Because the search space of phrase-based MT is limited by translation rules and the standard phrase-based forced decoding may fail for some NMT outputs, we proposed a phrase-based soft forced decoding algorithm, which can successfully find a phrase-based decoding path for any NMT outputs by introducing two new types of rules, source word deletion and target word insertion rules. We assigned high probabilities to function word deletion/insertion and low probabilities to content word deletion/insertion, because unlike content words, function words do not have clear correspondence between different languages. Therefore, similar with the standard phrase-based decoding score, the phrase-based soft forced decoding score can also heavily punish over, under and completely unrelated translations to improve the adequacy of NMT. In our experiments, the adequacy of NMT achieved significant improvement after being reranked by our proposed phrase-based soft forced decoding scores.

8. Future Work

Our hybrid neural-symbolic MT methods, i.e. developing neural features for the log-linear framework of symbolic MT; reranking NMT outputs with the proposed phrase-based soft forced decoding scores, which aim to combine the advantages of both NMT and symbolic MT for generating translations that are both fluent and adequate, are proved to be useful, but still have some limitations. First, incorporating neural features into symbolic MT leads to a limited translation search space, because the search space of symbolic MT is limited by translation rules. The proposed phrase-based soft forced decoding algorithm can solve the limited search space problem of phrase-based MT, however, it was used for reranking n-best NMT outputs, which are only a small part of all the possible translations. Using a larger n-best list can improve the reranking result, but then the time cost will increase for both NMT decoding and phrase-based soft forced decoding.

In the future, our work will focus on overcoming the limitations of the current hybrid neural-symbolic MT methods and developing more effective and efficient joint neural-symbolic decoding algorithms to further improve both the fluency and adequacy of MT.

One way to improve our current hybrid neural-symbolic MT methods is to incorporate the proposed phrase-based soft forced decoding scores into the NMT decoding process as neural-symbolic joint decoding instead of reranking to further improve the translation quality. Because the proposed phrase-based soft forced decoding algorithm also computes decoding scores for incomplete hypotheses from left to right similar with NMT decoding as shown in Figure 13, it is straightforward to incorporate the phrase-based decoding scores into NMT decoding.

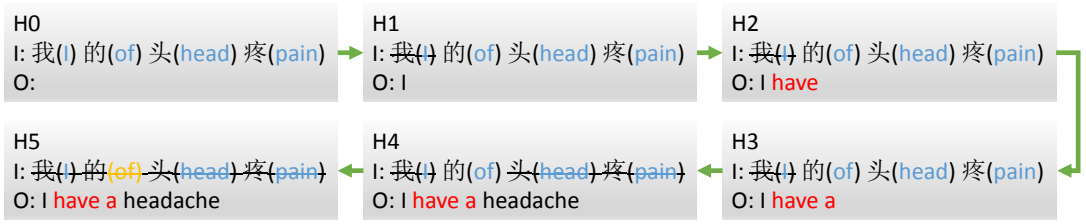

Figure 13. An example of the proposed phrase-based soft forced decoding path.

A major challenge in incorporating phrase-based soft forced decoding into NMT decoding is that the source word deletion rules in our phrase-based soft forced decoding algorithm are only applied after the target sentence generation is completed. Therefore, our method can only punish under translations for complete hypotheses, not for incomplete hypotheses during decoding. Because NMT usually keeps a small beam size during decoding due to computational and memory costs, it is very likely that all complete translations have under translation problems. To relief this problem, we can award more diverse hypotheses during decoding and use a larger beam size as the computer power grows.

Acknowledgements

Here I thank many people who have helped me complete this thesis. My supervisors, Professor Satoshi Nakamura and Assistant Professor Graham Neubig, are excellent teachers. They have given me both great advice and a lot of freedom for my research. They encouraged my enthusiasm for research. They have taught me a lot about how to be a researcher, how to do research and how to write good scientific papers. I am very grateful for everything they taught me. I also want to thank Associate Professor Katsuhito Sudoh, Associate Professor Sakriani Sakti, Assistant Professor Koichiro Yoshino, Associate Professor Yu Suzuki, Assistant Professor Hiroki Tanaka and all members in our AHC lab for discussing and advising my research.

I also give a lot of thanks to my supervisors in our NICT lab, Dr. Masao Utiyama and Dr. Eiichro Sumita. They welcomed me to Japan in the first place and encouraged me to pursue a doctoral degree in NAIST. They have given me great advice and many help for my research in the last four years. I also want to thank Dr. Andrew Finch, Dr. Chenchen Ding, Dr. Rui Wang, Dr. Atsushi Fujita, Dr. Benjamin Marie, Mr. Shoji Shiomi and all members in our NICT lab for all the help and advice they gave me.

I would also like to thank my thesis committee, Professor Satoshi Nakamura, Professor Yuji Matsumoto, Associate Professor Katsuhito Sudoh, Assistant Professor Graham Neubig, Dr. Masao Utiyama, for reviewing and helping me to improve my thesis.

I also thank my master degree supervisor, Professor Hai Zhao. He is my first advisor in research and taught me a lot about doing research and writing scientific papers. My research had a good start because of him.

I also want to thank our NAIST lab assistant Ms. Manami Matsuda, our NICT lab assistants Ms. Mari Oku and Ms. Chikako Tamasaki for helping my life in Japan. They have made my study and work in Japan much easier.

Finally, I want to thank my parents. They always support me.

References

[1] Tamer Alkhouli, Gabriel Bretschner, Jan-Thorsten Peter, Mohammed Hethnawi, Andreas Guta, and Hermann Ney. Alignment-based neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 54–65, August 2016.

[2] Jacob Andreas and Dan Klein. When and why are log-linear models self-normalizing? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 244–249, 2015.

[3] Philip Arthur, Graham Neubig, and Satoshi Nakamura. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, November 2016.

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

[5] Arianna Bisazza and Marcello Federico. Dynamically shaping the reordering search space of phrase-based statistical machine translation. *Transactions of the Association for Computational Linguistics*, 1:327–340, 2013.

[6] Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

[7] Hailong Cao, Dongdong Zhang, Mu Li, Ming Zhou, and Tiejun Zhao. A lexicalized reordering model for hierarchical phrase-based translation. In *The 25th International Conference on Computational Linguistics: Technical Papers*, pages 1144–1153, Dublin, Ireland, 2014.

[8] David Chiang. A hierarchical phrase-based model for statistical machine translation. In *The 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, 2005.

[9] Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 876–885, 2016.

[10] Lei Cui, Dongdong Zhang, Mu Li, Ming Zhou, and Tiejun Zhao. A joint rule selection model for hierarchical phrase-based translation. In *Proc. ACL*, pages 6–11, 2010.

[11] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proc. ACL*, pages 1370–1380, 2014.

[12] Nadir Durrani, Alexander Fraser, and Helmut Schmid. Model with minimal translation units, but decode with phrases. In *Proc. HLT-NAACL*, pages 1–11, 2013.

[13] Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. Tree-to-sequence attentional neural machine translation. *arXiv preprint arXiv:1603.06075*, 2016.

[14] Minwei Feng, Jan-Thorsten Peter, and Hermann Ney. Advancements in reordering models for statistical machine translation. In *The 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 322–332, Sofia, Bulgaria, 2013.

[15] Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inference and training of context-rich syntactic translation models. In *Proc. COLING-ACL*, pages 961–968, 2006.

[16] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *Proc. HLT-NAACL*, pages 273–280, 2004.

[17] Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. Learning continuous phrase representations for translation modeling. In *Proc. ACL*, pages 699–709, 2014.

[18] Yang Gao, Philipp Koehn, and Alexandra Birch. Soft dependency constraints for reordering in hierarchical phrase-based translation. In *The 2011 Conference on Empirical Methods in Natural Language Processing*, pages 857–868, Edinburgh, Scotland, UK., 2011.

[19] Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K Tsou. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proc. NTCIR-9*, pages 559–578, 2011.

[20] Jonathan Graehl and Kevin Knight. Training tree transducers. In *Proc. HLT-NAACL*, pages 105–112, 2004.

[21] Katsuhiko Hayashi, Hajime Tsukada, Katsuhito Sudoh, Kevin Duh, and Seiichi Yamamoto. Hierarchical phrase-based machine translation with word-based reordering model. In *The 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 439–446, Beijing, China, 2010.

[22] Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. Improved neural machine translation with smt features. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[23] Zhongjun He, Qun Liu, and Shouxun Lin. Improving statistical machine translation using lexicalized rule selection. In *Proc. Coling*, pages 321–328, 2008.

[24] Matthias Huck, Joern Wuebker, Felix Rietig, and Hermann Ney. A phrase orientation model for hierarchical machine translation. In *The Eighth Workshop on Statistical Machine Translation*, pages 452–463, Sofia, Bulgaria, 2013.

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

[26] Arefeh Kazemi, Antonio Toral, Andy Way, Amirhassan Monadjemi, and Mohammadali Nematbakhsh. Dependency-based reordering model for constituent pairs in hierarchical smt. In *The 18th Annual Conference of the Eu-*

ropean Association for Machine Translation, pages 43–50, Antalya, Turkey, 2015.

[27] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*, pages 388–395, 2004.

[29] Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, David Talbot, and Michael White. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *The International Workshop on Spoken Language Translation*, pages 68–75, Pittsburgh,USA, 2005.

[30] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, 2007.

[31] Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*, 2017.

[32] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54, 2003.

[33] Peng Li, Yang Liu, Maosong Sun, Tatsuya Izuha, and Dakun Zhang. A neural reordering model for phrase-based translation. In *The 25th International Conference on Computational Linguistics: Technical Papers*, pages 1897–1907, Dublin, Ireland, 2014.

[34] Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. Agreement on target-bidirectional neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 411–416, San Diego, California, June 2016. Association for Computational Linguistics.

[35] Qun Liu, Zhongjun He, Yang Liu, and Shouxun Lin. Maximum entropy based rule selection model for syntax-based statistical machine translation. In *Proc. EMNLP*, pages 89–97, 2008.

[36] Yang Liu, Qun Liu, and Shouxun Lin. Tree-to-string alignment template for statistical machine translation. In *Proc. COLING-ACL*, pages 609–616, 2006.

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

[38] Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proc. ACL-IJCNLP*, pages 11–19, 2015.

[39] Yuval Marton and Philip Resnik. Soft syntactic constraints for hierarchical phrased-based translation. In *The 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1003–1011, Columbus, Ohio, 2008.

[40] Arne Mauser, Saša Hasan, and Hermann Ney. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 210–218, 2009.

[41] Fandong Meng, Zhengdong Lu, Hang Li, and Qun Liu. Interactive attention for neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2174–2185, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[42] Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. Encoding source language with convolutional neural network for machine translation. In *Proc. ACL-IJCNLP*, pages 20–30, 2015.

[43] Haitao Mi, Liang Huang, and Qun Liu. Forest-based translation. In *Proc. HLT-ACL*, pages 192–199, 2008.

[44] Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, 2016.

[45] Graham Neubig. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proc. ACL*, pages 91–96, 2013.

[46] Graham Neubig, Makoto Morishita, and Satoshi Nakamura. Neural reranking improves subjective quality of machine translation: NAIST at WAT2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, Kyoto, Japan, October 2015.

[47] ThuyLinh Nguyen and Stephan Vogel. Integrating phrase-based reordering features into a chart-based decoder for machine translation. In *The 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1587–1596, Sofia, Bulgaria, 2013.

[48] Yizhao Ni, Craig Saunders, Sandor Szedmak, and Mahesan Niranjan. Handling phrase reorderings for machine translation. In *The ACL-IJCNLP 2009 Conference Short Papers*, pages 241–244, Suntec, Singapore, 2009.

[49] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pages 160–167, 2003.

[50] Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *The 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, Pennsylvania, USA, 2002.

[51] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51, 2003.

[52] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[53] David Rumelhart, Geoffrey Hinton, and Ronald Williams. Learning representations by back-propagating errors. *Nature 323(6088)*, pages 533–536, 1986.

[54] Holger Schwenk. Continuous space translation models for phrase-based statistical machine translation. In *Proc. COLING*, pages 1071–1080, 2012.

[55] Rico Sennrich, Barry Haddow, and Alexandra Birch. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, August 2016.

[56] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, August 2016.

[57] Hendra Setiawan, Zhongqiang Huang, Jacob Devlin, Thomas Lamar, Rabih Zbib, Richard Schwartz, and John Makhoul. Statistical machine translation features with multitask tensor networks. In *Proc. ACL-IJCNLP*, pages 31–41, 2015.

[58] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, August 2016.

[59] Felix Stahlberg, Adrià de Gispert, Eva Hasler, and Bill Byrne. Neural machine translation by minimising the bayes-risk with respect to syntactic translation lattices. In *Proceedings of the 15th Conference of the European*

Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 362–368, April 2017.

[60] Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 299–305, August 2016.

[61] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[62] Yaohua Tang, Fandong Meng, Zhengdong Lu, Hang Li, and Philip LH Yu. Neural machine translation with external phrase memory. *arXiv preprint arXiv:1606.01792*, 2016.

[63] Roy Tromble and Jason Eisner. Learning linear ordering problems for better translation. In *The 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1007–1016, Singapore, 2009.

[64] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, August 2016.

[65] Ashish Vaswani, Haitao Mi, Liang Huang, and David Chiang. Rule markov models for fast tree-to-string translation. In *Proc. HLT-ACL*, pages 856–864, 2011.

[66] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. Decoding with large-scale neural language models improves translation. In *Proc. EMNLP*, pages 1387–1392, 2013.

[67] Xing Wang, Deyi Xiong, and Min Zhang. Learning semantic representations for nonterminals in hierarchical phrase-based translation. In *The 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1391–1400, Lisbon, Portugal, 2015.

[68] Yuguang Wang, Shanbo Cheng, Liyang Jiang, Jiajun Yang, Wei Chen, Muze Li, Lin Shi, Yanfeng Wang, and Hongtao Yang. Sogou neural machine translation systems for wmt17. In *Proceedings of the Second Conference on Machine Translation*, pages 410–415, 2017.

[69] Joern Wuebker, Mei-Yuh Hwang, and Chris Quirk. Leave-one-out phrase model training for large-scale deployment. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 460–467, June 2012.

[70] Joern Wuebker, Arne Mauser, and Hermann Ney. Training phrase translation models with leaving-one-out. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 475–484, July 2010.

[71] Puyang Xu, Asela Gunawardana, and Sanjeev Khudanpur. Efficient subsampling for training complex language models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1128–1136, 2011.

[72] Richard Zens and Hermann Ney. Discriminative reordering models for statistical machine translation. In *The Workshop on Statistical Machine Translation*, pages 55–63, New York City, 2006.

[73] Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. Bilingually-constrained phrase embeddings for machine translation. In *Proc. ACL*, pages 111–121, 2014.

[74] Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. A binarized neural network joint model for machine translation. In *Proc. EMNLP*, pages 2094–2099, 2015.

[75] Hai Zhao, Chang-Ning Huang, and Mu Li. An improved chinese word segmentation system with conditional random field. In *Proc. SIGHAN*, pages 162–165, 2006.

Publication List

Peer review journal paper

1. Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Hai Zhao, Graham Neubig, Satoshi Nakamura. Learning Local Word Reorderings for Hierarchical Phrase-based Statistical Machine Translation. Machine Translation. Volume 30. Pages 1-18. June 2016. Related to Chapter 3.

2. Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, Satoshi Nakamura. Improving Neural Machine Translation through Phrase-based Soft Forced Decoding. Machine Translation. Submitted. Related to Chapter 6.

Peer review international conference

1. Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, Satoshi Nakamura. A Binarized Neural Network Joint Model for Machine Translation. The 2015 Conference on Empirical Methods in Natural Language Processing. Pages 2094-2099. September 2015. Related to Chapter 4.

2. Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, Satoshi Nakamura. A Continuous Space Rule Selection Model for Syntax-based Statistical Machine Translation. The 54th Annual Meeting of the Association for Computational Linguistics. Volume 1. Pages 1372-1381. August 2016. Related to Chapter 5.

3. Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, Satoshi Nakamura. Improving Neural Machine Translation through Phrase-based Forced Decoding. The 8th International Joint Conference on Natural Language Processing. Volume 1. Pages 152-162. November 2017. Related to Chapter 6.