A preordering method using head-restructured CFG parse tree for SMT

Zhongyuan Zhu, Masanori Taniguchi, Chenchen Ding and Mikio Yamamoto
University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki, Japan
{raphael, m.taniguchi, tei}@mibel.cs.tsukuba.ac.jp, myama@cs.tsukuba.ac.jp

1. INTRODUCTION

Statistical Machine Translation (SMT) has been proved to be practical in the translations of many major language pairs. However, for language pairs with a sharp contrast in word orders such as English-Japanese, SMT runs into obstacles. The translation correspondences between these languages are difficult to capture, which finally affects the translation quality.

In previous researches, reordering input sentences to the expected word order of the target language before training a translation model improved the translation quality. The work of (Yamada & Knight, 2001) is considered as a forerunner in reordering syntactic parse trees. Basically, automatically generated syntactic information or human-made rules are utilized in the reordering process. As an example, the state-of-the-art reordering method for English-to-Japanese, Head-finalization (Isozaki et al., 2010) utilizes head-driven phrase structure grammar (HPSG) parse trees.

For reordering methods based on word alignments, the work of (Lerner & Petrov, 2013) reorders a dependency parse tree by a multi-class classifier with a set of human-made features, which is trained on reordered a training corpus according to the expected word order of the target language. In (Navrátil et al., 2012), subtrees are reordered on the basis of their reordering probabilities. In their model, parent and sibling nodes are considered to be the context. In (Bisazza & Federico, 2012), language models are utilized for pruning an n-best list of reordered chunk representations. Further, they combine language models trained for different representations by log-linear combinations.

In this paper, we present a novel reordering model based on head-restructured context-free grammar (CFG) parse tree. This parse tree is built by restructuring all S nodes (non-terminal nodes with the tag "S", which indicate sentences) in a CFG parse tree with head-dependent relationships, and re-tagging some non-terminal nodes with words. We call this new parse tree "head-restructured CFG parse tree" in this paper. However, for the reordering model, we reorder these parse trees by applying language models to estimate the probabilities of all possible orders and find the best order. From the reordered parse tree, we can export a string of words that are optimized in the expected word order of the target language.

2. HEAD-RESTRUCTURED CFG PARSE TREE

2.1 Restructuring S nodes

The word order of many languages such as English and Japanese is considered to be closely related to the grammatical roles of words (e.g., subjects and objects), which can be retrieved from dependency parse trees. Therefore, the use of dependency parse trees for reordering can help the reordering model to capture these correspondences. While we want to maintain reasonable syntactic structures for most parts of the entire parse tree, we extract head-dependent relationships from a dependency parse tree and restructure S nodes in the corresponding CFG parse tree.

To construct head-restructured CFG parse trees, we need both CFG and dependency parse trees as the input. Here, a dependency parse tree is required to be consistent with the CFG parse tree of the same sentence. The construction process is divided into the following four steps:

1. Find a node $node_s$ with the syntactic tag "S" in the CFG parse tree, and mark the words covered by $node_s$ with $span(node_s)$.
2. In the dependency parse tree, find the head word $dephead_s$, which also exactly covers $span(node_s)$. (As our generated dependency parse tree is consistent with the CFG parse tree, $dephead_s$ shall exist).
3. For each word $depchild_i \in \{depchild_1, ..., depchild_n\}$ that modifies the head word $dephead_s$ in the dependency parse tree, we create a new non-terminal node. Its syntactic tag is set to be the type of grammatical relation between $depchild_i$ and $dephead_s$. Then, we append this newly created non-terminal node to be a child node of $node_s$ in the CFG parse tree. In the case of head word $dephead_s$, we put the corresponding terminal node in the CFG parse tree underneath $node_s$ directly.
4. For each word $depchild_i$ in Step 3, we find a non-terminal node in the CFG parse tree that covers the same span as $depchild_i$ covers in the dependency parse tree. Then, we move this non-terminal node to be a child node of the non-terminal node that we created in Step 3 corresponding to the word $depchild_i$.

We execute these four steps recursively until all the S nodes are restructured. In the left side of Figure 1, we show an example of CFG parse trees. The dotted lines show the dependency relations of each word, and the type of its grammatical relation is mentioned to the right of the arrow. The right side of Figure 1 shows the restructured parse tree by applying these four steps. As there is only one S node (the root node), we only perform the reconstruction for this node. Hence, $span(node_s)$ includes all words in this case. In Step 2, the verb "throw" is determined as $dephead_s$, which has three dependent words: "I", "ball", and "rapidly". Then, in Step 3, their grammatical relations are applied as the tag of the three newly created non-terminal nodes under $node_s$ ("nsubj", "dobj", and "advmod"). For the head word "throw", we put (VBP throw) underneath $node_s$ directly. In Step 4, the spans that the three dependent words cover are "I", "the ball", and "rapidly", respectively, which correspond to the non-terminal node "PRP", "NP", and "ADVP" in the CFG parse tree. Therefore, we moved these three non-terminal nodes beneath the corresponding nodes created in Step 3, respectively.

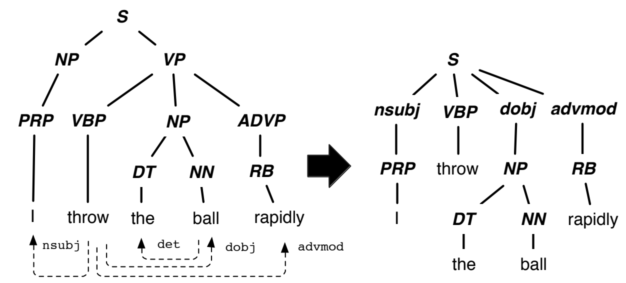

Figure 1: An example of a restructured tree

The restructured parse tree obtained after applying the abovementioned four steps is similar to the output of Charniak-Johnson Re-ranking Parser (Charniak & Johnson, 2005) to some extent. For the example shown in the left of Figure 1, their parser gives the following result in S-expression:

```
(S (NP (PRP I))
   (VP (VBP throw) (NP (DT the) (NN ball))
       (ADVP (RB rapidly)) ) )
```

In the parsing result of their parser given above, all the dependent portions are placed beneath the node "VP" together with the head word.

2.2 Re-tagging non-terminal nodes

To capture fine reordering correspondences, both the tags and the words should be considered in the reordering model. Hence, we use a word to represent a non-terminal node in CFG parse trees if possible.

For the following two patterns, we re-tag all the matching non-terminal nodes in a CFG parse tree by words.

1. There is only one child node of the given non-terminal node, which is the parent of a terminal node. We apply the word of this terminal node as the tag
2. There are several child nodes of the given non-terminal node, in which the left-most child node is the parent of a terminal node, and one of the remaining child nodes is the parent of the non-terminal node. We apply the word corresponding to the terminal node beneath the first child node as the tag.

We illustrate these two patterns of re-tagging in Figure 2.

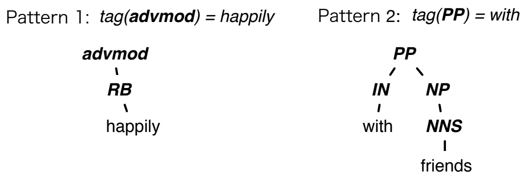

Figure 2: An illustration of two patterns of re-tagging

However, we apply a practical restriction that forces the tags representing nouns ("NP", "nsubj", and others) to remain unchanged. This restriction prevents the reordering model from becoming very sparse.

3. PROPOSED REORDERING MODEL

3.1 Model

While considering the head-restructured CFG parse trees as inputs, the reordering model focuses on obtaining the best order for the child nodes of each non-terminal node.

Given a node with the tag t_p and its child nodes with tags $\mathbf{T} = \{t_1, ..., t_n\}$, we define an order $\mathbf{o}$ as a sequence, where $o_i \in \{1, ..., n\}$ should be unique in $\mathbf{o}$. In our reordering model, the probability of an order $\mathbf{o}$ in the case of this tag set is calculated as follows:

$$\begin{aligned} p(t_p, \mathbf{T}, \mathbf{o}) &= p(t_p, t_{o_1}, t_{o_2}, ..., t_{o_n}) & (1)\\ &= p(t_p)p(t_{o_1}|t_p)...p(t_{o_n}|t_p, t_{o_1}, ..., t_{o_{n-1}}) & (2)\\ &= p(t_p)\prod_{i=1}^{n} p(t_{o_i}|t_p, t_{o_1}, ..., t_{o_{i-1}}). & (3)\end{aligned}$$

The best order $\hat{\mathbf{o}}$ can be obtained as follows:

$$\hat{\mathbf{o}} = \underset{\mathbf{o}'}{\operatorname{argmax}}\, p(t_p, \mathbf{T}, \mathbf{o}').$$

As expressed by equation (3), the purpose of our reordering model is to utilize language models to obtain the probability of the tag sequence $(t_p, t_{o_1}, t_{o_2}, ..., t_{o_n})$. For example, the original order of the S node in Figure 1 corresponds to the tag sequence "**S nsubj throw dobj rapidly**". We add the tag of the parent node to the beginning as context information.

3.2 Model training

In the model training phase, we generate a large number of reordered tag sequences on the basis of the automatically retrieved word alignments and then, train an n-gram language model using them as our reordering model.

Here, we show how to obtain a tag sequence with the optimized order for a node in the parse tree. First, for each child node, we get the aligned words of target language corresponding to the words it cover. Then we calculate the average position of those aligned words in the target sentence.

As an example, the bilingual pair corresponding to the restructured tree in the right of Figure 1 is shown in Figure 3. The dotted lines represent the automatically retrieved word alignments. Therefore, for all child nodes of the S node in this case, we show the details of each step to get the average position in Table 1.

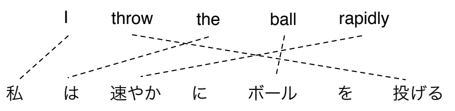

Figure 3: An example of a bilingual pair with word alignments

Table 1: The details of each step to get the average position for the S node in the right of Figure 1

nodes	nsubj	VBP	dobj	advmod
tags	nsubj	throw	dobj	rapidly
words	{I}	{throw}	{the, ball}	{rapidly}
aligned words	{私}	{投げる}	{は, ボール}	{速やか}
positions	{1}	{7}	{2,5}	{3}
average positions	1	7	3.5	3

To obtain a tag sequence in the expected word order of the target language, we merely sort the tag sequence by the value of average position for each node. For the example in Table 1, the optimized tag sequence is "**nsubj rapidly dobj throw**". We append the tag of the parent node before outputting the abovementioned tag sequence as a piece of training data of the language models, which will be "**S nsubj rapidly dobj throw**".

3.3 Reordering

In the reordering phase, given a node with its child nodes, we look up all possible orders of its child nodes. Then, we use language models to evaluate the probability of the corresponding tag sequences. The order with the highest probability is selected as the best order. We reorder these child nodes according to the best order. Given a head-restructured CFG parse tree, we reorder all the child nodes of each node to form a new tree and export all the words of the terminal nodes to form a string output.

4. EXPERIMENTS

4.1 Experimental settings

Our experiments were carried out on the NTCIR-7 corpus in the English-to-Japanese translation; this corpus contains 1.8M bilingual sentences for use as training data and 915 bilingual sentences for tuning the translation models. We use the data provided for the formal run of NTCIR-7, which contains 1,381 sentences, as test data. In the experiments, we clean the training data by limiting each sentence to no more than 40 words. Our reordering model and translation model are both trained on these cleaned data.

To generate head-restructured CFG parse trees, we parse all sentences on the English side into CFG and dependency parse trees. We selected the Berkeley parser for CFG parsing. Then, we use CFG parse trees to generate standard Stanford dependencies using the Stanford parser. Finally, we obtain head-restructured CFG parse trees using the method described in Section 2.

We first use GIZA++ to train word alignments from the original training data. Then, on the basis of the word alignments and head-restructured CFG parse trees, we generate a list of tag sequences using the method described in Section 3.2, which is optimized in the expected word order of Japanese. We utilize SRILM to train a 5-gram language model with Kneser-Ney smoothing on the optimized tag sequences.

As our experiments are focused on English-to-Japanese translation, we select the state-of-the-art reordering method Head-finalization as our baseline. Similar to Head-finalization, we append seed words "va_nsubjpass", "va_nsubj", and "va_dobj" to the end of words covered by non-terminal nodes with the tags "nsubjpass", "nsubj", and "dobj", respectively. The first two seed words are considered to correspond to "は" or "が", while the remaining one is considered to correspond to "を" in Japanese. Further, we removed "a", "an", and "the" from all the sentences on the English side of the training corpus.

4.2 Evaluations using Kendall's Tau

To evaluate how close the orders are between our reordered English sentences and the corresponding Japanese sentences, we estimate Kendall's τ in the same manner as that of Head-finalization for the cleaned training data. To compare, we created a replication of Head-finalization processing the results of Enju HPSG parser. The distribution of the estimated Kendall's τ values after adding seed words is shown in Figure 4. Because there is no recorded τ value lower than -0.7, we omit this part in the figure. The second column of Table 2 shows a comparison of the average Kendall's τ value for each method.

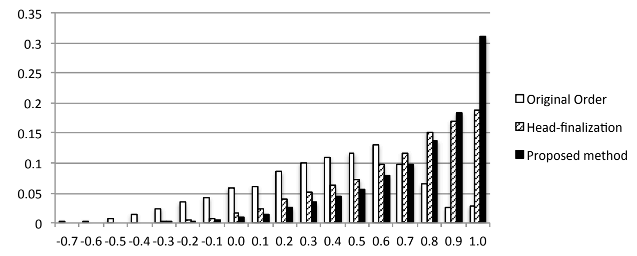

Figure 4: A comparison of the distributions of Kendall's τ on the cleaned training data

4.3 Evaluations of translation quality

We carried out experiments for testing the translation quality using the phrase-based model (Koehn et al., 2003) implemented in Moses. For the training data in the original order, we gained the best automatic scores when the distortion limit (dl) is set to 18. The best automatic scores are obtained when dl = 6 for both the proposed method and Head-finalization. A comparison of the BLEU (Papineni, 2002) and RIBES (Isozaki, 2010) scores for each method is shown in Table 2.

4.4 Evaluations of reordering speed

Table 2: Automatic scores for each method

Method	Average τ	BLEU(%)	RIBES
Original Order (dl = 18)	0.419	31.21	0.702
Head-finalization (dl = 6)	0.727	32.01	0.749
Proposed method (dl = 6)	0.797	**34.56***	0.769

* means that the result is significantly different at the 5% level as compared to that of other methods (only for BLEU evaluation criterion).

To get a comparison for reordering speed, we estimated the average processing time (user time) for reordering one sentence by Head-finalization and our proposed method, which is shown in table 3. Where the column of "Stanford" for proposed method means the time consumed for generating Standard Stanford Dependencies from the parsing result of Berkeley parser.

Table 3: A comparison of average processing time(ms) for one sentence

Head-finalization	Enju		Reordering	Total
	446.58		7.12	453.7
Proposed method	Berkeley	Stanford	Reordering	Total
	220.24	31.53	1.65	**253.42**

4.5 Reordering samples

In this section, we show some samples of the reordered sentences. Basically, for sentences with simple grammatical structures, the proposed method produces similar results as Head-finalization. For sentence (1), the proposed method generates the reordered result (2), while Head-finalization returns (3). Here, the corresponding Japanese translation is "期間T 2 において電流が停止している." Sentence (2) can be translated to Japanese monotonically.

Because Japanese is an SOV language, often, the subject words appear before the verbs. However, the position of the other parts corresponding to the prepositional clauses (e.g., "in ..." and "for ...") is considered to be related more with the actual words. By re-tagging non-terminal nodes with words, the proposed model is able to capture these patterns.

```
(1) the current is stopped in the interval t2

(2) interval t2 in current va_nsubjpass stopped is

(3) current _va1 interval t2 in stopped is
```

For exemplifying what happens in the case of complex sentences, we show sample sentence (4), which translates to "目皿板が回転する装置も多く提案されている" in Japanese. The proposed method produces (5) as the result of reordering, while (6) is the result of Head-finalization. In this sample, the proposed method benefits from the correct parsing result that recognizes "proposed" as a head. The head-restructured CFG parse tree of (4) has the following form: "**(S (S (there (NP (EX there))) (have (VBP have)) (been (VBN been)) (VBN proposed) (ccomp ...)) (. .))**".

```
(4) there have been proposed many apparatuses wherein
    a perforated disk is made rotatable.

(5) wherein perforated disk va_nsubjpass
    made is rotatable many apparatuses there
    proposed been have.

(6) there _va1 perforated disk _va1 rotatable
    made is wherein proposed many apparatuses
    _va2 been _va2 have.
```

5. CONCLUSION AND FUTURE WORKS

In this paper, we proposed a reordering model based on head-restructured CFG parse trees built from CFG and dependency parse trees, and used language models to reorder them. Because the standard Stanford Dependencies can be generated from any CFG parse tree very rapidly, the performance of the proposed method actually depends only on a CFG parser. Owing to the trade-offs between the parsing speed and accuracy, the proposed method satisfied many different needs by applying different CFG parsers. When compared to this, the parsers for other linguistic grammar such as HPSG are considerably limited.

The proposed reordering model tries to capture correspondences at the word level to perform more accurate reordering. In our experiments, both BLEU and RIBES scores improved on using the proposed reordering model as compared to the state-of-the-art reordering method Head-finalization. On the other hand, involving words in the reordering model resulted in sparseness problem. In some simple cases, the proposed method failed to reorder a tag sequence like "**S nsubj comprises dobj**" correctly because "comprises" appeared only a few times in the training data.

6. REFERENCES

[1] Arianna Bisazza and Marcello Federico. Modified distortion matrices for phrase-based statistical machine translation. In *Proceedings of the 50th Annual Meeting of the ACL: Long Papers-Volume 1*, pages 478–487. ACL, 2012.

[2] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on ACL*, pages 173–180. ACL, 2005.

[3] Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on EMNLP*, pages 944–952. ACL, 2010.

[4] Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. Head finalization: A simple reordering rule for sov languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 244–251. ACL, 2010.

[5] Ramanathan Jirí Navrátil Karthik Visweswariah Ananthakrishnan. A comparison of syntactic reordering methods for english-german machine translation. 2012.

[6] 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 ACL on Human Language Technology-Volume 1*, pages 48–54. ACL, 2003.

[7] Uri Lerner and Slav Petrov. Source-side classifier preordering for machine translation. In *Proceedings of EMNLP*, 2013.

[8] 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 ACL*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. ACL.

[9] Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on ACL*, pages 523–530. ACL, 2001.