Dependency Tree Abstraction for Long-Distance Reordering in Statistical Machine Translation

Chenchen Ding
Department of Computer Science
University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki , Japan
tei@mibel.cs.tsukuba.ac.jp

Yuki Arase
Microsoft Research
No. 5 Danling St., Haidian Dist.
Beijing, P.R. China
yukiar@microsoft.com

Abstract

Word reordering is a crucial technique in statistical machine translation in which syntactic information plays an important role. Synchronous context-free grammar has typically been used for this purpose with various modifications for adding flexibilities to its synchronized tree generation. We permit further flexibilities in the synchronous context-free grammar in order to translate between languages with drastically different word order. Our method pre-processes a parallel corpus by abstracting source-side dependency trees, and performs long-distance reordering on top of an off-the-shelf phrase-based system. Experimental results show that our method significantly outperforms previous phrase-based and syntax-based models for translation between English and Japanese.

1 Introduction

Since the inception of statistical machine translation (SMT), long-distance word reordering has been a notable challenge, particularly when translating between languages with drastically different word orders, such as subject-verb-object (SVO) and subject-object-verb (SOV) languages like English and Japanese, respectively. Phrase-based models (Koehn et al., 2003; Och and Ney, 2004; Xiong et al., 2006) have been strong in local translation and reordering. However, phrase-based models cannot effectively conduct long-distance reordering because they are based purely on statistics of syntax-independent phrases. As a complementary approach to phrase-based models, some researchers have incorporated syntactic information into an SMT framework (Wu, 1997; Yamada and Knight, 2001; Liu et al., 2006) using *synchronous context-free grammar* (SCFG) (Aho and Ullman, 1972). The original SCFG assumes that the syntactic trees of the source and target languages can be derived synchronously. However, this assumption is too strict for handling parallel sentences that are often comparable rather than parallel. For alleviating this assumption, some researchers have added flexibilities in synchronized tree generation (Wu, 1997; Burkett et al., 2010). In addition, in the SMT framework, there is an approach that alleviates the assumption by only generating the source-side syntactic tree and projecting it to the target-side sentence (Yamada and Knight, 2001; Liu et al., 2006).

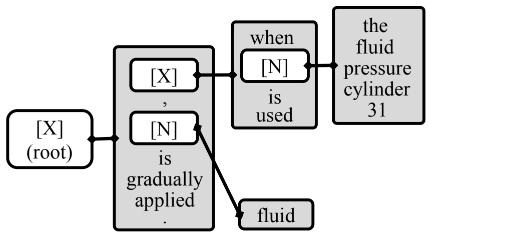
[X] (root)
[X] ,
[N] is gradually applied .
when [N] is used
the fluid pressure cylinder 31
fluid
when the fluid pressure cylinder 31 is used, fluid is gradually applied.

Figure 1: English abstraction tree example

In practice, these existing methods are not flexible enough to handle parallel sentence pairs, especially those of SVO and SOV languages. Therefore, we permit further flexibility in SCFG aiming to effectively conduct long-distance reordering. We design our method as a pre-processing procedure so that we can use a well-developed phrase-based system without adding heavy computational complexity to the system. Specifically, we propose an *abstraction tree* that is a shallow and nested representation, *i.e.*, abstraction of the dependency tree as Fig. 1 depicts. Our method pre-processes a parallel corpus by generating source-side abstraction trees and projecting the trees onto the target-side sentences. It then decomposes the corpus by collecting corresponding node pairs as a new corpus, and finally trains the phrase-based model. In this manner, the source-side grammar is determined on the fly for each sentence based on a dependency parse of the source sentence. The target side of each production in the grammar is determined by running the phrase-based decoder.

We empirically show effectiveness of our method for English-to-Japanese and Japanese-to-English translations by comparing it to phrase-based and syntax-based models. Experimental results show that our method significantly outperforms the previous methods with respect to the BLEU (Papineni et al., 2002) metric.

2 Related Work

For adding flexibilities to SCFG under an SMT scenario, previous studies generate only a source-side syntactic tree and project it to the target-side sentence regardless of the true target-side syntactic structure. Liu et al. (2006) propose a tree-to-string model using a source-side constituency tree to extract correspondences between the source-side tree and the target-side sentence. Quirk et al. (2005) and Xie et al. (2011) use a dependency tree for the same purpose. Since these methods project a fine-grained source-side syntax tree, an accurate projection is possible only when the target-side sentence has a syntactic structure that is similar to the source-side. Zhu and Xiao (2011) and Huang and Pendus (2013) generalize rules obtained by the tree-to-string model to increase the chance of rule matching at decoding. Despite their merits, none of these methods resolves the problem of tree projection to the target-side.

The hierarchical phrase-based model (HIERO) proposed by Chiang (2007) is independent of any syntactic information and generates SCFG rules only from parallel sentence pairs. Li et al. (2012) and Feng et al. (2012) incorporate syntactic information into HIERO as soft constraints. Since these methods are bound by the original HIERO rules that are independent of syntactic information, their rules cannot represent the global syntactic structure of a sentence.

There are also pre-reordering methods for long-distance reordering in SVO-to-SOV translations using heuristics designed based on source-side syntactic structures (Xu et al., 2009; Isozaki et al., 2010; Isozaki et al., 2012). They are fine-tuned to handle only specific reordering problems in a predetermined language pair. Another approach is to statistically learn pre-reordering rules from a corpus; however, this requires a highly parallel training corpus consisting of literal translations to learn effective rules (Neubig et al., 2012; Navratil et al., 2012). Such a training dataset is not widely available in many languages.

Algorithm 1 CKY-style decoding

```
Input: Input sentence $u$ and its dependency tree $r_u$, translation model $TM$, block-LM $bLM$, sentence-LM $sLM$, size of $m$-best $m$
 1: $\tau_u \leftarrow$ generate abstraction tree of $u$ using $r_u$
 2: $NodeTrans[][] \leftarrow \emptyset$
 3: for all $node$ in $\tau_u$ do
 4:     $m\text{-}best \leftarrow Decode(node, TM, bLM, m)$
 5:     $(start, end) \leftarrow$ start and end indices of $node$ in $u$
 6:     $NodeTrans[start][end] \leftarrow m\text{-}best$
 7: end for
 8: for $start := 1$ to $|u|$ do
 9:     for $span := 0$ to $|u| - 1$ do
10:         $end \leftarrow start + span$
11:         $ChildTrans[] \leftarrow \emptyset$
12:         for all $(i, j)$ such that $start \leq i \leq j \leq end$ do
13:             if $NodeTrans[i][j] \neq \emptyset$ then
14:                 add $NodeTrans[i][j]$ to $ChildTrans$
15:             end if
16:         end for
17:         $CubePruning(NodeTrans[start][end],$
                $ChildTrans, sLM, m)$
18:     end for
19: end for
```

3 Overview of the Proposed Method

Our method pre-processes sentences in a parallel corpus based on source-side abstraction trees. It first generates an abstraction tree τ_s of a source-side sentence s by abstracting its dependency tree r_s: $(s, r_s) \rightarrow \tau_s$. It then projects the tree to the target-side sentence t for generating a target-side abstraction tree τ_t that has exactly the same structure to τ_s, *i.e.*, $(\tau_s, t) \rightarrow \tau_t$. The abstraction-tree generation process can be adapted to translating languages by specifying source-side part-of-speeches (POSs) as input. Abstraction tree structures also depend on the dependency grammar that a parser uses. In this study, we assume commonly used Stanford typed dependency (de Marneffe et al., 2006) for English and the chunk-based dependency with ipadic (Asahara and Matsumoto, 2003) for Japanese. Investigation of effects of different dependency grammars is our future work.

We decompose the sentence pair into node pairs according to correspondences between the source and target abstraction trees, and generate a new corpus referred to as a *block-corpus* (Fig. 6). Using the block-corpus and the original corpus, we train a phrase-based model. Its translation model is trained with the block-corpus, and two target-side language models (LMs) are trained with the block-corpus (referred to as *block-LM*) and the

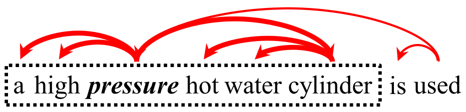

Figure 2: [N] node detection example

original corpus (referred to as *sentence-LM*), respectively. Thus the sentence-LM can be trained using a larger-scale monolingual corpus. Compared to previous methods that also decompose sentence pairs (Xu et al., 2005; Sudoh et al., 2010), our method is more syntax-oriented.

In decoding, we adopt the parsing algorithm with cube-pruning (Huang and Chiang, 2005) into a phrase-based decoder to translate the abstraction tree of an input sentence efficiently. As Algorithm 1 shows, our decoder first generates the abstraction tree of the input sentence (line 1), and independently translates each node using the block-LM that models ordering among non-terminals and lexical words (line 3–7). It then combines the m-best translation hypotheses of each node to construct a sentence-level translation (line 8–19). Specifically, we insert sets of the m-best translation hypotheses of child nodes into the m-best hypotheses of their parent node by replacing the corresponding non-terminals using the cube-pruning (line 17). The ordering of these child nodes has been determined in their parent node by the phrase-based model that regards non-terminals only as single words. By doing so, long-distance reordering is solved considering the global syntactic structure and contexts (lexical strings) preserved in the node. In cube-pruning, we use the sentence-LM to compose fluent sentence-level translation. The block-LM and sentence-LM scores are treated as independent features.

The computational complexity of our decoder (line 3–19) is $\mathcal{O}(|\mathcal{N}|C)$, where $|\mathcal{N}|$ is the number of nodes in the abstraction-tree and C is a constant representing the complexity of phrase-based decoder and cube-pruning. Since combinations of hypotheses in cube-pruning are determined by the abstraction-tree in our method, the computational cost is significantly smaller than HIERO's case.

4 Abstraction Tree Generation

In this section, we provide the formal definition of an abstraction tree and the generation method.

4.1 Definition of Abstraction Tree

We define an abstraction tree as $\tau = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N}$ is a set of nodes and $\mathcal{E}$ is a set of edges. For conducting abstraction based on syntactic structures, we merge a span governed by a dependency head as a node and represent it by a non-terminal in a parent node. As a result, the i-th node $\mathcal{N}_i$ consists of a sequence of lexical words w and non-terminals L that replace spans governed by heads in the corresponding child nodes:

$$\mathcal{N}_i = \{\Psi|\psi_1, \ldots, \psi_{|\mathcal{N}_i|}\}, \psi_k \in \{w, L\}.$$

The edge $\mathcal{E}_{ij}$ between a parent node $\mathcal{N}_i$ and its child node $\mathcal{N}_j$ corresponds to a governor-dependent relationship from the head in $\mathcal{N}_i$ to its dependent w_x in $\mathcal{N}_j$. w_x is another head and governs other words in $\mathcal{N}_j$. The span covered by $\mathcal{N}_j$ is replaced by a non-terminal in $\mathcal{N}_i$.

We use three kinds of labels to represent L for explicitly using syntactic information that is useful for long-distance reordering; [N], [P], and [X] according to the head in the corresponding node. We label a child node [N] when its head word is a noun and forms a base noun phrase, [P] when its head word is an adposition[1], and [X] for others like verb phrases, conjunctive phrases, and relative phrases. These nodes play different roles in a sentence. An [N] node, *i.e.*, a base noun phrase, adds context to a sentence. A [P] node depends on other phrases and generally appears relatively freely in a sentence. Thus, we assume that the [P] node requires special reordering.

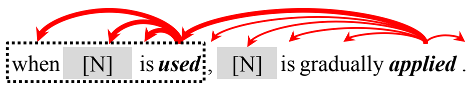

Figure 3: [X] and [P] node detection. Since the word "used" is a head, it and its governing span are detached from the root "applied" as a child node.

The abstraction tree depicted in Fig. 1 has a parent [X] node "when [N] is used" and its child [N] node "the fluid pressure cylinder 31." The word "used" governs "cylinder" in the [N] node, and the [N] node folds the context in the [X] node.

4.2 Tree Construction

We begin with detecting [N] nodes, then proceed to [P] and [X] nodes. These processes require to specify source-side POSs as input for adapting to translating languages. We finally flatten fragmented nodes.

For detecting [N] nodes, we take a POS of noun as input and identify each noun and its governing span, *i.e.*, a string of all governed words, using the source-side dependency tree as Fig. 2 shows. We

[1] A preposition in English and a postposition in Japanese.

Algorithm 2 [P] and [X] node detection

```
Input: Source-side sentence $s$ and its dependency tree $r_s$,
    POS list of adpositions $PPos$, [N] node list $N\text{-}Nodes$
Output: [P] and [X] nodes $P\text{-}Nodes$, $X\text{-}Nodes$
 1: $P\text{-}Nodes[] \leftarrow \emptyset$, $X\text{-}Nodes[] \leftarrow \emptyset$
 2: $HeadList[] \leftarrow$ root of $r_s$
 3: repeat
 4:   $head \leftarrow$ pop node from $HeadList$
 5:   $ChildList[] \leftarrow$ all dependents of $head$
 6:   for all $child$ in $ChildList$ do
 7:     if $child \notin N\text{-}Nodes$ and $child$ has dependents then
 8:       add $child$ to $HeadList$
 9:       remove $child$ from $ChildList$
10:     end if
11:   end for
12:   $start \leftarrow$ smallest start index of nodes in $ChildList$
13:   $end \leftarrow$ largest end index of nodes in $ChildList$
14:   if POS of $head \in PPos$ then
15:     add span $[start, end]$ of $s$ to $P\text{-}Nodes$
16:   else
17:     add span $[start, end]$ of $s$ to $X\text{-}Nodes$
18:   end if
19:   remove $head$ from $HeadList$
20: until $HeadList = \emptyset$
```

regard descendant dependents as being governed by the noun for detecting a noun phrase of a complete form. We extract the span as a node and replace it by an [N] label in the sentence.

Next, we identify [P] and [X] nodes given a list of source-side POSs of adpositions as input. As Algorithm 2 shows, after [N] node detection, we trace the dependency tree from its root to leaves (line 3–20). We find all the dependents to the root, then check if each dependent is a head. If a dependent of the root is a head and governs other words, we detach the dependent to process later (line 6–11). We then find the smallest start index and largest end index of dependent words and set the corresponding span as a node (if a dependent is in an [N] node, we use the start and end indices of the [N] node). Each node is labeled according to the POS of its head as [P] or [X] (line 14–18). We then take the detached dependent as a new root and repeat the process until no more detachment is possible. The computational complexity is $\mathcal{O}(|s|^2)$. Through this process, a span with direct dependencies is extracted as a node, and other spans with descendant dependencies become descendant nodes, replaced by non-terminals in their parent node as shown in Fig. 3.

4.3 Handling Complex Noun Phrase

As described in Sec. 4.2, we detect a noun phrase as an [N] node. However, an [N] node becomes more complex than a base noun phrase when the head governs a clause, such as a relative clause. Such a complex node may require long-distance reordering of an inside clause when translating. Therefore, we separate the noun phrase and clause. We take a POS list whose word can be a head of [P] and [X] nodes (preposition, verb, *to*, *Wh*-determiner/pronoun/adverb, and coordinating conjunction for English) as input. If the POS of a noun's dependent is in the list, we detach the dependency arc, and then re-attach the dependency arc to the head of the noun. As a result, the base noun phrase becomes an [N] node and its clause becomes a [P] or [X] node that is transformed to a sibling of the [N] node.

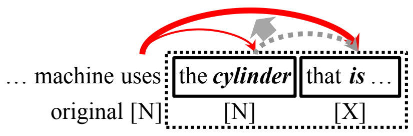

Figure 4: Handling a complex noun phrase

In Fig. 4, the word "cylinder" in the original [N] node has a relative clause and governs "is." We detach the dependency arc and re-attach it to "uses" (the head of "cylinder"), so that the noun phrase and the clause become sibling [N] and [X] nodes.

4.4 Flattening Fragmented Nodes

The above processes are independent of the size of each node, meaning they produce fragmented nodes of only a few words. Such fragmented nodes make the tree projection to the target-side difficult. To solve this problem, we flatten the abstraction tree as shown in Algorithm 3. We process an internal node in τ_s from bottom to top. If the covering span of an internal node is less than a threshold $\gamma \in \mathbb{N}$, its child nodes are merged (line 3 and 4, Algorithm 3). Specifically, we reinsert the child nodes by replacing the corresponding non-terminals with lexical strings that the child nodes have been covered by. The computational cost is $\mathcal{O}(|\mathcal{N}|)$. We investigate the effect of γ in the following evaluation section (Table 2).

5 Abstraction Tree Projection

In this section, we describe a method for projecting the obtained source-side abstraction tree onto the target-side sentence.

5.1 Tree Structure Projection

We use word alignment results for tree structure projection. However, accurate word alignment is challenging when handling language pairs in which long-distance reordering is needed, and the alignment noise propagates to the tree projection.

Algorithm 3 Tree flattening

```
Input: Abstraction tree $\tau_s$, threshold $\gamma$
Output: Flattened tree $\tau'_s$
 1: for all internal $node$ in $\tau_s$, from bottom to top do
 2:     $(start, end) \leftarrow$ start and end indices of $node$
 3:     if $end - start + 1 < \gamma$ then
 4:         $\tau'_s \leftarrow MergeChildNodes(node, \tau_s)$
 5:     end if
 6: end for
```

To avoid this problem, we first omit alignment links of function words whose alignment quality tends to be lower than that of the content words. We then complement the quality of word alignment by adapting the *syntactic cohesion assumption* (Yamada and Knight, 2001) that assumes a word string covered by a sub-tree of the source-side syntactic tree corresponds to a string of *contiguous* words in the target-side sentence. Following the assumption, we project the k-th node of the source-side abstraction tree $\mathcal{N}_k^{(s)}$ to a string of contiguous words in the target-side:

$$\mathcal{N}_k^{(s)} \mapsto t_i, \ldots, t_j, \text{ s.t. } 1 \le i \le j \le |t|,$$

where t_i is the i-th word in the target-side sentence and $|t|$ is the number of words in the sentence.

For each node of the source-side abstraction tree, we first obtain its covering span. We then define a vector $\boldsymbol{c} \in \{0,1\}^n$ whose elements represent word alignment links in a binary manner. If and only if the i-th target word is aligned to a word in the span, the i-th element of $\boldsymbol{c}$ becomes 1, otherwise it is 0. Since the original word alignment represented by the vector $\boldsymbol{c}$ may be noisy, we find a vector $\boldsymbol{c}^* \in \{0,1\}^n$ that maximizes the syntactic cohesion assumption. In $\boldsymbol{c}^*$, *only* consecutive elements between two indices i and j are 1, and others are 0. We derive such $\boldsymbol{c}^*$ as follows:

$$\begin{aligned} C_{min}(\boldsymbol{c}) &= \{\boldsymbol{c}' | \operatorname*{argmin}_{\boldsymbol{c}'} \|\boldsymbol{c}' - \boldsymbol{c}\|\}, \qquad (1)\\ & \quad \text{s.t. } \exists\, i,\ j,\ 1 \le i \le j \le n \text{ and} \\ & \quad c'_k = \begin{cases} 1 & i \le k \le j, \\ 0 & \text{otherwise,} \end{cases} \\ \boldsymbol{c}^* &= \operatorname*{argmax}_{\boldsymbol{c}' \in C_{min}(\boldsymbol{c})} \|\boldsymbol{c}'\|. \qquad (2) \end{aligned}$$

The operator $\|\cdot\|$ computes the Euclidean norm of a vector and c'_k is the k-th element of a vector $\boldsymbol{c}'$. Finally, $\boldsymbol{c}^*$ represents the best possible word links that maximize the syntactic cohesion assumption, *i.e.*, the longest contiguous word string in the target-side, and that are closest to the original word alignment. Specifically, Eq. (1) determines vectors that have the smallest distance to the original vector $\boldsymbol{c}$ while satisfying the hard constraint, and Eq. (2) selects the one whose norm is largest, *i.e.*, a vector that has longest contiguous word links to the target-side. For computational efficiency, we use the greedy-search so that the computational cost is $\mathcal{O}(|t|)$. When Eq. (2) has multiple solutions, word links in these solutions are equally likely, and thus we merge them into a unique solution. Specifically, we take the union of the solutions and find the smallest index i_l and largest index i_r whose elements are 1. We then set all elements between i_l and i_r to 1.

Algorithm 4 Tree projection

```
Input: Source-side abstraction tree $\tau_s$, target-side sentence
       $t$, word alignment $\boldsymbol{A_w}$ between $s$ and $t$
Output: Target-side abstraction tree $\tau_t$
 1: $\tau_t[] \leftarrow \emptyset$
 2: remove links of function words in $\boldsymbol{A_w}$
 3: for $span := |s| - 1$ to 0 do
 4:     for $start := 1$ to $|s|$ do
 5:         $end \leftarrow start + span$
 6:         if span $[start, end] \in \tau_s$ then
 7:             $\boldsymbol{c} \leftarrow GenerateVector([start, end], \boldsymbol{A_w})$
 8:             $\boldsymbol{c}^* \leftarrow Solve(\boldsymbol{c})$ ◁ Eq. (1) and Eq. (2)
 9:             $(i, j) \leftarrow$ start and end indices of $\boldsymbol{c}^*$
10:             add span $[i, j]$ of $t$ as a node into $\tau_t$
11:             $\boldsymbol{A_w} \leftarrow UpdateWordAlignment(\boldsymbol{c}, \boldsymbol{c}^*)$
12:         end if
13:     end for
14: end for
```

As Algorithm 4 shows, we conduct this process in a top-down manner throughout the abstraction tree (line 3–14). When processing each node, word alignment links are updated by overwriting links in $\boldsymbol{c}$ with the ones in $\boldsymbol{c}^*$ (line 11). The computational cost is $\mathcal{O}(|\mathcal{N}^{(s)}||t|)$, where $|\mathcal{N}^{(s)}|$ is the number of nodes in τ_s. Figure 5 shows a projection example of a node. A node of "when [N] is used" covers a span of "when the fluid pressure cylinder 31 is used." The words in the span are aligned to the 1st, 2nd, and 5th target words (chunks)[2]; however, the link to the 5th target word (chunk) is a mis-alignment. With the alignment vector $\boldsymbol{c}$ of $[1, 1, 0, 0, \mathbf{1}, 0, 0]$, we can remove this misalignment and derive $\boldsymbol{c}^*$ of $[1, 1, 0, 0, \mathbf{0}, 0, 0]$.

5.2 Fixed-Expression Recovery

The abstraction tree generation and projection are based on a dependency tree, and thus may over-segment a *fixed-expression*, such as idioms and multi-word expressions. Since a fixed-expression composes a complete meaning using a contiguous word string, splitting it into different nodes results

[2]In Japanese, a unit of dependency is a chunk in general, and thus we conduct chunking before projection.

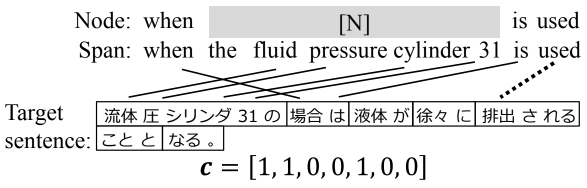

Figure 5: Abstraction tree projection

in poor translation. To avoid this issue, we generate a list of fixed-expressions using conventional methods (Evert, 2008) and force them to remain in one node. On both the source and target abstraction trees, we recursively reinsert nodes to their parent node when such a fixed-expression is over-segmented and spread over multiple nodes.

5.3 Block-Corpus Construction

After tree structure projection, we extract corresponding node pairs as a block-corpus. Each node pair has a form $\langle \Psi_s, \Psi_t, \boldsymbol{A_L} \rangle$, where $\Psi_s \in \{\Omega_s, L\}^n$ represents the source-side node of length n. It consists of a sequence of lexical words in the source-side vocabulary Ω_s and non-terminals L. $\Psi_t \in \{\Omega_t, L\}^m$ similarly represents the target-side node of length m. $\boldsymbol{A_L}$ preserves correspondences between the non-terminals in the source and target nodes.

Specifically, we extract a pair of leaf nodes as a pair of lexical strings. As for internal nodes, we use the same non-terminal labels appearing in the source-side node at the target-side node. Namely, the span covered by child nodes are replaced by corresponding non-terminal labels in the source-side node. At the same time, we record the correspondence between the non-terminals. Figure 6 shows an example of the block-corpus, in which the boxed indices indicate correspondences of non-terminals in the source and target nodes.

6 Evaluation

We evaluate our method in English-to-Japanese (EJ) and Japanese-to-English (JE) translation tasks, since long-distance reordering is a serious problem in this language pair.

6.1 Experiment Corpus

We use NTCIR-7 PATMT (Fujii et al., 2008), a publicly available standard evaluation dataset, for EJ and JE machine translation. The dataset is constructed using English and Japanese patents and consists of 1.8 million parallel sentence pairs for training, 915 sentence pairs for development, and 1,381 sentence pairs for testing. The development

Input parallel corpus	
When the fluid pressure cylinder 31 is used , fluid is gradually applied .	流体 圧 シリンダ 31 の 場合 は 流体 が 徐々 に 排出 さ れる こと と なる 。
Block-Corpus	
[X]$\boxed{0}$, [N]$\boxed{0}$is gradually applied .	[X]$\boxed{0}$ [N]$\boxed{0}$ が 徐々 に 排出 さ れる こと と なる 。
when [N]$\boxed{1}$ is used	[N]$\boxed{1}$ の 場合 は
the fluid pressure cylinder 31	流体 圧 シリンダ 31
fluid	流体

Figure 6: Block-corpus example. Boxed indices link non-terminals in the source and target exemplars.

and test sets have one reference per sentence. This dataset is bidirectional and can be used for both EJ and JE translation evaluation.

6.2 Implementation of Proposed Method

We implement our method for EJ and JE translation tasks. In both cases, we use an in-house implementation of English POS tagger (Collins, 2002) and a Japanese morphological analyzer (Kudo et al., 2004) for tokenization and POS tagging. As for EJ translation, we use the Stanford parser (de Marneffe et al., 2006) to obtain English abstraction trees. We also use an in-house implementation of a Japanese chunker (Kudo and Matsumoto, 2002) to obtain chunks in Japanese sentences. We apply the chunker just before tree projection for using a chunk as a projection unit, since a chunk is the basic unit in Japanese. As for JE translation, we use a popular Japanese dependency parser (Kudo and Matsumoto, 2002) to obtain Japanese abstraction trees. We convert Japanese chunk-level dependency tree to a word-level using a simple heuristic. We use GIZA++ (Och and Ney, 2003) with the grow-diag-final-and heuristic for word alignment.

We use an in-house implementation of the bracketing transduction grammar (BTG) model (Xiong et al., 2006) as the phrase-based model that our method relies on for translation. Non-terminals in our block-corpus are regarded as a single word, and their alignments $\boldsymbol{A_L}$ determined in the block-corpus are exclusively used to align them. We set the maximum phrase length to 5 when training the translation model, since we find that the performance is stable even setting larger values as in (Koehn et al., 2003). We then train the sentence-LM and block-LM using the original corpus and the obtained block-corpus, respectively. We ignore a sentence-end tag (</s>) in the block-LM. With each corpus, we train a 5-gram LM using the SRI toolkit (Stolcke, 2002).

6.3 Comparison Method

Since our method pre-processes the parallel corpus based on SCFG with increased flexibility and trains a BTG model using the processed corpus, we compare our method to another BTG model trained only with the original corpus (simply referred to as the BTG model). We also compare to the tree-to-string model and HIERO using state-of-the-art implementations available in the Moses system (Koehn et al., 2007), since they are based on SCFG. The tree-to-string model requires source-side constituency trees. For EJ translation, we use a state-of-the-art English constituency parser (Miyao and Tsujii, 2005; Miyao and Tsujii, 2008). For JE translation, we transform a Japanese dependency tree into a constituency tree using a simple heuristic because there is no publicly available constituency parser. During the translation model training, we use the same setting as our method. In addition, we set the maximum span of rule extraction to infinity for the tree-to-string model and 10 for HIERO following Moses' default. We use the sentence-LM in these models as they assume.

In addition, we compare our method to Head-Finalization (Isozaki et al., 2010; Isozaki et al., 2012) because it has achieved the best BLEU score in EJ translation by handling long-distance reordering. It is a specialized method to EJ translation, where a syntactic head in an English sentence is reordered behind its constituents for complying with the head-final nature of the Japanese language. We pre-process the parallel corpus using the Head-Finalization and train a BTG model using the same setting with our method to observe the effect of different pre-processing methods.

During decoding, we set the translation table size to 10 for each source string, and the stack and beam sizes in the cube pruning to 100 for our method (*i.e.*, m-best $= 100$) and all other models. The maximum reordering span in the tree-to-string model and HIERO is the same as the rule extraction setting (infinity and 10, respectively). We set the word reordering limit to infinity for our method and the BTG model, while we set it to 3 for Head-Finalization as their papers report.

We tune feature weights by the minimum error rate training (Och, 2003) to maximize the BLEU score using the development set. As an evaluation metric, we compute the BLEU score using the test set, and all the scores discussed in Sec. 6.4 are the test-set BLEU scores. Significance tests are conducted using bootstrap sampling (Koehn, 2004).

Method	EJ	JE
Proposed method ($\gamma = 10$)	$\mathbf{31.78}$	$\mathbf{28.55}$
BTG	28.82^{**}	26.98^{**}
HIERO	29.27^{**}	27.96^{*}
Tree-to-string	30.97^{**}	26.28^{**}
Head-Finalization	29.52^{**}	NA

Table 1: Test-set BLEU scores. The symbol ** represents a significant difference at the $p < .01$ level and * indicates a significant difference at the $p < .05$ level against our method.

6.4 Result and Discussion

In this section, we present experimental results and discuss them in detail.

Overall Performance Table 1 shows the BLEU scores, in which our method significantly outperforms all other models for both EJ and JE translation tasks. These results indicate that our method effectively incorporates syntactic information into the phrase-based model and improves the translation quality.

For EJ translation, our method outperforms the BTG model by 2.96, the HIERO by 2.51, the tree-to-string model by 0.81, and the Head-Finalization[3] by 2.26 in terms of BLEU score. When we compare our method to the Head-Finalization, both of them improve the BTG model by pre-processing the parallel corpus. Moreover, our method outperforms the Head-Finalization using richer syntactic information.

For JE translation, our method outperforms the BTG model by 1.57, the HIERO by 0.59, and the tree-to-string model by 2.27 in terms of BLEU score. Our method and the tree-to-string model, which depend on syntactic information, largely outperform the BTG model and HIERO in EJ translation. While the BTG model and HIERO, which are independent of syntactic information, outperform the tree-to-string model in JE translation. One reason for this phenomenon is that English is a strongly configurational language that has rigid word order while Japanese is an agglutinative language that has relatively free word order. A rigid syntactic structure provides solid clues for word reordering when translated into a flexible language, while a flexible structure provides weak clues for fitting it to a rigid structure.

[3]The BLEU score reported in this experiment differs from their papers. This may be because they use a phrase-based model in the Moses system, while we use the BTG model.

γ	EJ		JE	
	BLEU	height	BLEU	height
0	31.15	4.1 (1.5)	28.41	4.2 (1.4)
3	30.88	3.8 (1.7)	28.34	3.9 (1.6)
5	31.21	3.7 (1.5)	28.39	3.8 (1.5)
8	31.61	3.4 (1.4)	28.52	3.4 (1.4)
10	**31.78**	3.1 (1.3)	**28.55**	3.2 (1.3)
12	31.76	2.9 (1.3)	28.54	3.0 (1.3)
15	31.25	2.6 (1.2)	28.21	2.7 (1.2)
∞	28.82	1.0 (–)	26.98	1.0 (–)

Table 2: Effect of threshold γ

Effect of Flattening Threshold Table 2 shows BLEU scores when changing the flattening threshold γ in our method, and averages and standard deviations of the abstraction tree heights ($\gamma = \infty$ is equal to the BTG model). The performance improves as we increase the threshold, *i.e.*, increasing the level of abstraction. Our method achieves the best BLEU score when $\gamma = 10$ for both EJ and JE translation, with the performance degrading as we further increase the threshold.

This trend shows the trade-off between phrase-based and syntax-based approaches. When the threshold is too small, an abstraction tree becomes closer to the dependency tree and the tree-projection becomes difficult. In addition, context information becomes unavailable when conducting long-distance reordering with a deep tree. On the other hand, when setting the threshold too large, the abstraction tree becomes too abstracted and syntactic structures useful for long-distance word reordering are lost. We need to balance these effects by setting an appropriate threshold.

Effect of Non-Terminals and Fixed-Expressions We change the kinds of non-terminal labels in an abstraction tree to investigate their effect on the translation quality. When we merge the [P] label to the [X] label, *i.e.*, use only [N] and [X] labels, the BLEU score drops 0.40 in EJ translation while the score is unaffected in JE translation. This is because flexible Japanese syntax does not differentiate postpositional phrases with others, while English syntax prohibits such a flexibility.

When we merge all labels and only use the [X] label, the BLEU score drops 0.57 in EJ translation and 0.43 in JE translation. This result supports our design of the abstraction tree that distinguishes non-terminals according to their different functionalities in a sentence.

We also evaluate the effect of fixed-expressions as described in Sec. 5.2. Results show a significant change when over-splitting fixed-expressions; the BLEU score drops 1.13 for EJ and 0.36 for JE translation without reinserting fixed-expressions.

Method	acceptable ↑	global ↓	local ↓
Proposed	**52**	**30**	**4**
BTG	34	38	7
Tree-to-string	47	32	7

Table 3: Error distribution in 100 samples of EJ translation

Error Analysis We randomly sample 100 translation outputs per our method ($\gamma = 10$), BTG, and tree-to-string models for each EJ and JE translation tasks, and manually categorize errors based on (Vilar et al., 2006). We focus primarily on reordering errors and exclusively categorize the samples into acceptable translations, translations with only global or local reordering errors, as well as others that are complicated combinations of various errors. An acceptable translation correctly conveys the information in a source sentence even if it contains minor grammatical errors.

Table 3 shows the distribution of acceptable translations and those with global/local reordering errors in the EJ task (results of JE task are omitted due to the severe space limitation, but their trend is similar). It confirms that our method reduces reordering errors, not only for long-distance but for local reordering, and increases the ratio of acceptable translations compared to the BTG and tree-to-string models. We also find that long-distance reordering was attempted in 85, 66, and 70 sentences by our method, BTG, and tree-to-string, respectively, among these translations. The results show that our method performs long-distance reordering more frequently than others.

When we compare translations performed by our method to those performed by the tree-to-string model, we observe that their effectiveness depends on a range of reordering. Our method is effective in long-distance reordering like those of clauses, while the tree-to-string model performs middle-range reordering well. This is due to the trade-off regarding the level of abstraction as discussed in the flattening threshold experiment.

7 Conclusion and Future Work

We have proposed an abstraction tree for effectively conducting long-distance reordering using an off-the-shelf phrase-based model. Evaluation results show that our method outperforms conventional phrase-based and syntax-based models.

We plan to investigate the effect of translating language pairs and dependency grammars in abstraction tree generation. In addition, we will apply a structure-aware word aligner (Neubig et al., 2011) to improve the tree projection.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The theory of parsing, translation, and compiling*. Prentice-Hall Inc.

Masayuki Asahara and Yuji Matsumoto. 2003. ipadic version 2.7.0 user's manual. `http://sourceforge.jp/projects/ipadic/docs/ipadic-2.7.0-manual-en.pdf`.

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT 2010)*, pages 127–135.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of International Conference on Language Resources and Evaluation (LREC 2006)*, pages 449–454.

Stefan Evert. 2008. Corpora and collocations. In Anke Lüdeling and Merja Kytö, editors, *Corpus Linguistics. An International Handbook*, volume 2, chapter 58. Mouton de Gruyter.

Yang Feng, Dongdong Zhang, Mu Li, Ming Zhou, and Qun Liu. 2012. Hierarchical chunk-to-string translation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 950–958.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2008. Overview of the patent translation task at the NTCIR-7 workshop. In *Proceedings of NTCIR-7 Workshop Meeting (NTCIR)*, pages 389–400.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of International Workshop on Parsing Technology (IWPT 2005)*, pages 53–64.

Fei Huang and Cezar Pendus. 2013. Generalized reordering rules for improved SMT. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 387–392.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010. Head Finalization: A simple reordering rule for SOV languages. In *Proceedings of Joint Workshop on Statistical Machine Translation and Metrics MATR (WMT-MetricsMATR 2010)*, pages 244–251.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2012. HPSG-based preprocessing for English-to-Japanese translation. *ACM Transactions on Asian Language Information Processing (TALIP)*, 11(3):8:1–8:16.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT 2003)*, pages 48–54.

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. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, pages 177–180.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 388–395.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of Conference on Natural Language Learning (CoNLL 2002)*, pages 1–7.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 230–237.

Junhui Li, Zhaopeng Tu, Guodong Zhou, and Josef van Genabith. 2012. Head-driven hierarchical phrase-based translation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 33–37.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 609–616.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of Annual Meeting on Association for Computational Linguistics (ACL 2005)*, pages 83–90.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Jiri Navratil, Karthik Visweswariah, and Ananthakrishnan Ramanathan. 2012. A comparison of syntactic reordering methods for English-German machine translation. In *Proceedings of International Conference on Computational Linguistics (COLING 2012)*, pages 2043–2058.

Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. 2011. An unsupervised model for joint phrase alignment and extraction. In *Proceedings of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011)*, pages 632–641.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*, pages 843–853.

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

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of Annual Meeting on Association for Computational Linguistics (ACL 2003)*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of Annual Meeting on Association for Computational Linguistics (ACL 2002)*, pages 311–318.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of Annual Meeting on Association for Computational Linguistics (ACL 2005)*, pages 271–279.

Andreas Stolcke. 2002. SRILM - An extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing (ICSLP 2002)*, pages 901–904.

Katsuhito Sudoh, Kevin Duh, Hajime Tsukada, Tsutomu Hirao, and Masaaki Nagata. 2010. Divide and translate: Improving long distance reordering in statistical machine translation. In *Proceedings of Joint Workshop on Statistical Machine Translation and Metrics MATR (WMT-MetricsMATR 2010)*, pages 418–427.

David Vilar, Jia Xu, Luis Fernando d'Haro, and Hermann Ney. 2006. Error analysis of statistical machine translation output. In *Proceedings of International Conference on Language Resources and Evaluation (LREC 2006)*, pages 697–702.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 216–226.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of International Conference on Computational Linguistics and Annual Meeting on Association for Computational Linguistics (COLING-ACL 2006)*, pages 521–528.

Jia Xu, Richard Zens, and Hermann Ney. 2005. Sentence segmentation using IBM word alignment model 1. In *Proceedings of Annual Conference of the European Association for Machine Translation (EAMT 2005)*, pages 280–287.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz J. Och. 2009. Using a dependency parser to improve SMT for subject-object-verb languages. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT 2009)*, pages 245–253.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of Annual Meeting on Association for Computational Linguistics (ACL 2001)*, pages 523–530.

Jingbo Zhu and Tong Xiao. 2011. Improving decoding generalization for tree-to-string translation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2001)*, pages 418–423.