Extracting Pre-ordering Rules from Predicate-Argument Structures

Xianchao Wu, Katsuhito Sudoh, Kevin Duh, Hajime Tsukada, Masaaki Nagata
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai Seika-cho, Soraku-gun Kyoto 619-0237 Japan
{wu.xianchao,sudoh.katsuhito,kevin.duh,tsukada.hajime,nagata.masaaki}@lab.ntt.co.jp

Abstract

Word ordering remains as an essential problem for translating between languages with substantial structural differences, such as SOV and SVO languages. In this paper, we propose to automatically extract pre-ordering rules from predicate-argument structures. A pre-ordering rule records the relative position mapping of *a predicate word* and its *argument phrases* from the source language side to the target language side. We propose 1) a linear-time algorithm to extract the pre-ordering rules from word-aligned HPSG-tree-to-string pairs and 2) a bottom-up algorithm to apply the extracted rules to HPSG trees to yield target language style source sentences. Experimental results are reported for large-scale English-to-Japanese translation, showing significant improvements of BLEU score compared with the baseline SMT systems.

1 Introduction

Statistical machine translation (SMT) suffers from an essential problem for translating between languages with substantial structural differences, such as between English which is a subject-verb-object (SVO) language and Japanese which is a typical subject-object-verb (SOV) language.

Numerous approaches have been consequently proposed to tackle this word-order problem, such as lexicalized reordering methods, syntax-based models, and pre-ordering ways. First, in order to overcome the shortages of traditional distance based distortion models (Brown et al., 1993; Koehn et al., 2007), phrase dependent lexicalized reordering models were proposed by several researchers (Tillman, 2004; Kumar and Byrne, 2005). Lexicalized reordering models learn local orientations (monotone or reordering) with probabilities for each bilingual phrase from the training data. For example, by taking lexical information as features, a maximum entropy phrase reordering model was proposed by Xiong et al. (2006).

Second, syntax-based models attempt to solve the word ordering problem by employing syntactic structures. For example, linguistically syntax-based approaches (Galley et al., 2004; Liu et al., 2006) first parse source and/or target sentences and then learn reordering templates from the subtree fragments of the parse trees. In contrast, hierarchical phrase based translation (Chiang, 2005) is a formally syntax-based approach which can automatically extract hierarchical ordering rules from aligned string-string pairs without using additional parsers. These approaches have been proved to be both algorithmically appealing and empirically successful.

However, most of current syntax-based SMT systems use IBM models (Brown et al., 1993) and hidden Markov model (HMM) (Vogel et al., 1996) to generate word alignments. These models have a penalty parameter associated with long distance jumps, and tend to misalign words which move far from the window sizes of their expected positions (Xu et al., 2009; Genzel, 2010).

The third type tackles the word-order problem in pre-ordering ways. Through the usage of a sequence of pre-ordering rules, the word order of an original source sentence is (approximately) changed into the word order of the target sentence. Here, the pre-ordering rules can be manually or automatically extracted. For manual extraction of pre-ordering rules, linguistic background and expertise are required for predetermined language pairs, such as for German-English (Collins et al., 2005), Chinese-to-English (Wang et al., 2007), Japanese-to-English (Katz-Brown and Collins, 2007), and English-to-SOV languages (Xu et al., 2009).

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 29–37,
Chiang Mai, Thailand, November 8 – 13, 2011.

Specially, for English-to-Japanese translation, Isozaki et al. (2010b) proposed to move syntactic or semantic heads to the end of corresponding phrases or clauses so that to yield head finalized English (HFE) sentences which follow the word order of Japanese. The head information of an English sentence is detected by a head-driven phrase structure grammar (HPSG) parser, Enju[1] (Miyao and Tsujii, 2008). In addition, transformation rules were manually written for appending particle seed words, refining POS tags to be used before parsing, and deleting English determiners. Due to the usage of the same parser, we take this HFE approach as one of our baseline systems.

The goal in this paper, however, is to learn pre-ordering rules from parallel data in an *automatic* way. Under this motivation, pre-ordering rules can be extracted in a language-independent manner. A number of researches follow this automatic way. For example, in (Xia and McCord, 2004), a variety of heuristic rules were applied to bilingual parse trees to extract pre-ordering rules for French-English translation. Rottmann and Vogen (2007) learned reordering rules based on sequences of part-of-speech (POS) tags, instead of parse trees. Dependency trees were used by Genzel (2010) to extract source-side reordering rules for translating languages from SVO to SOV, etc..

The novel idea expressed in this paper is that, predicate-argument structures (PASs) are introduced to extract fine-grained pre-ordering rules. PASs have the following merits for describing reordering phenomena:

- predicate words and argument phrases respectively record reordering phenomena in a *lexicalized level* and an *abstract level*;
- PASs provide a *fine-grained classification* of the reordering phenomena since they include factored representations of syntactic features of the predicate words and their argument phrases.

The idea of using PASs for pre-ordering follows (Komachi et al., 2006). Several reordering operations were manually designed by Komachi et al. (2006) to pre-ordering Japanese sentences into SVO-style English sentences. For comparison, our proposal 1) makes use of not only PASs but also the source syntactic tree structures for pre-ordering rule matching, 2) extracts pre-ordering rules in an automatic way, and 3) use factored representations of syntactic features to refine the pre-ordering rules.

Following (Wu et al., 2010a; Isozaki et al., 2010b), we use the HPSG parser Enju to generate the PASs of English sentences. HPSG (Pollard and Sag, 1994) is a lexicalist grammar framework. In HPSG, linguistic entities such as words and phrases are represented by a data structure called a *sign*. A sign gives a factored representation of the syntactic features of a word/phrase, as well as a representation of their *semantic content* which corresponds to PASs.

In order to record the relative positions among a predicate word and its argument phrases, we propose a linear-time algorithm to extract pre-ordering rules from word-aligned HPSG-tree-to-string pairs[2]. The syntactic features included in signs and the types of PASs enable us to extract fine-grained pre-ordering rules and thus make it easier to select appropriate rules for given source HPSG trees. We further propose a bottom-up algorithm to apply the extracted rules to HPSG trees to pre-order source sentences. Using the pre-ordered source sentences, we retrain word alignments again.

The remaining of this paper is organized as follows. In the next section, we describe the algorithms guided by using a real example for extracting and applying PAS-based pre-ordering rules. Then, we design experiments on large-scale English-to-Japanese translation to testify our proposal. Employing Moses (Koehn et al., 2007), we show that our proposal can significantly improve BLEU scores of 2.47∼3.15 points compared with using the original English sentences. We finally conclude this paper by summarizing our proposal and the experiment results.

2 Pre-ordering Rule Extraction and Application

2.1 An example

Figure 1 shows a word-aligned HPSG-tree-to-string pair for English-to-Japanese translation. PASs among lexical nodes and their argument nodes in this HPSG tree are described by arrows in thick-lines. For simplicity, we only draw the identifiers for the signs of the nodes in the HPSG tree. Note that the identifiers that start with 'c'

[1] http://www-tsujii.is.s.u-tokyo.ac.jp/enju/index.html

[2] These word alignments are gained by running GIZA++ (Och and Ney, 2003) on the original parallel sentences.

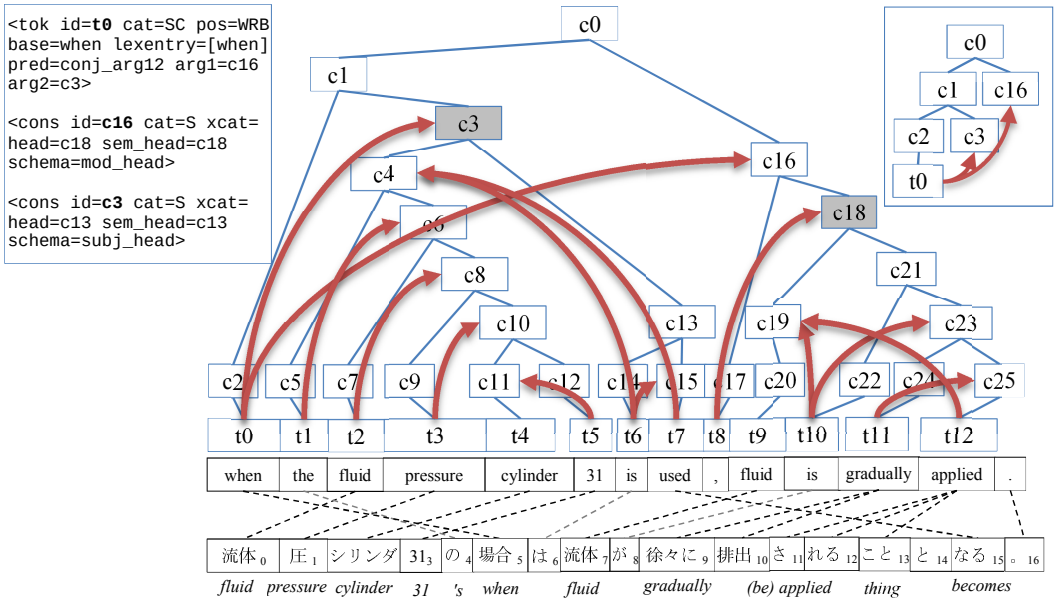

Figure 1: Illustration of a word-aligned HPSG-tree-to-string pair for English-to-Japanese translation.

denote non-terminal nodes (e.g., c0, c1), and the identifiers that start with 't' denote terminal nodes (e.g., t0, t2). In a complete HPSG tree (Wu et al., 2010b), factored syntactic features listed in Table 1 are included in the terminal and non-terminal signs. These features are used by us to sub-categorize pre-ordering rules. As an example of the XML output of Enju, the signs of "when" (t0) and its arguments c16, c3 are shown in the top-left corner of Figure 1.

2.2 Data structures

We define the following data structures for both extracting and applying pre-ordering rules. First, a PAS-based pre-ordering rule is defined to be a four-tuple <*pw*, *args*, srcOrder, trgOrder>. Here, *pw* is the predicate word, *args* are the argument nodes of *pw*, and srcOrder and trgOrder respectively record the relative positions among *pw* and *args* in the source and target language sides.

Then, we suppose an HPSG tree/subtree object contains the following methods:

- localize(): localize syntactic/semantic heads;
- computeSrcSpans(): topologically compute the source span of each node;
- computeSpans(A): topologically compute the source and target spans of each node (Galley et al., 2004). A is the word alignment;
- getArgs(*pw*): return the argument nodes of *pw*;

Name	Description	Examples
WORD	surface word form	"when"
BASE	base word form	"when"
POS	part-of-speech	WRB ("when")
LE	lexical entry	[when] ("when")
PRED	type of predicate argument structure	conj_arg12 ("when")
CAT	syntactic category	SC ("when")
TENSE	tense of a verb (past, present, untensed)	present ("used")
ASPECT	aspect of a verb (none, prefect, progressive, prefect-progressive)	none ("used")
VOICE	voice of a verb (passive, active)	passive ("used")
AUX	auxiliary verb or not (minus, modal, have, be, do, to, copular)	minus ("used")
CAT	syntactic category	S (c16), S (c3)
XCAT	extended category	
HEAD	syntactic head	R (c16), R(c3)
SEM_HEAD	semantic head	R (c16), R (c3)
SCHEMA	schema rule	mod_head (c16)

Table 1: Templates of atomic features included in the predicate node (top size) and its argument nodes (bottom side).

- MCT(*pw*, *args*): return the minimum cover tree (Wu et al., 2010a) of *pw* and *args*.

To implement the localize() method, we use the approach described in (Wu et al., 2010a). That is, we replace the pointer values of HEAD and SEM_HEAD features in non-terminal nodes with three labels: "S" for single daughter, "L" for the left-hand-side daughter, and "R" for the right-

hand-side daughter. For example, for node c16 in Figure 1, its HEAD and SEM_HEAD will change from c18 to "R".

We use the concept of *minimum covering trees* (MCT) defined in (Wu et al., 2010b) to guide the pre-ordering process. A MCT is a subtree of the original HPSG tree that takes a predicate node and its argument nodes as (new) leaf nodes. For example, as shown in the top-right corner of Figure 1, the MCT of "when" (t0) and its argument nodes c3, c16 is "c0(c1(c2(t0)c3)c16)".

Finally, the attributes in the nodes of an HPSG tree include: 1) pred: the PAS of a leaf node, 2) srcSpan: the index set of the source words that current node covers, 3) trgSpan: the index set of the target words that srcSpan aligned to, and 4) srcPhrase that stores the pre-ordered source phrase covered by current node.

2.3 Rule extraction algorithm

We express the idea for extracting PAS-based pre-ordering rules by using the first word "when" of the English sentence in Figure 1. Given the PAS information of "when" (t0) in the English side, we need to determine the target-side-order among t0 and its two arguments c16, c3. To achieve this, we compute the target spans of these three nodes by using current word alignment and then sort their target spans. Through referring to the word alignment shown in Figure 1, we can collect the target spans which are {5}, {4,0,1,2,3,6,15}, and {7,8,9,10,11,12,13} respectively for t0, c3, and c16. However, we cannot sort these three spans since there are overlapping between the first two spans[3]. In order to solve this problem, we sort the spans in a heuristic way. Note that in c3's target span, five indices are smaller than 5 yet only two indices are larger than 5. Thus, we take {4,0,1,2,3,6,15} to be *dominantly smaller* than {5}. Now, we can determine the pre-order rule guided by the PAS of t0 to be "t0 c3 c16 → c3 t0 c16" and formally to be "$t0_0$ $c3_1$ $c16_2$ → 1 0 2". Generally, we use the following heuristic rules to sort two spans, named span A and span B:

- if more than half of numbers in A is bigger than the maximum number in B, or if more than half of numbers in B is smaller than the minimum number in A, then B < A;
- if more than half of numbers in B is bigger than the maximum number in A, or if more than half of numbers in A is smaller than the minimum number in B, then A < B.

In case of a tie (e.g., A={3,4,7,8}, B={5,6}), we keep the original order of A and B in the source-side sentence without any reordering.

Algorithm 1 Pre-ordering Rule Extraction

Input: HPSG tree T_E of an English sentence E, word alignment A
Output: a pre-ordering rule set $\mathcal{R}$

```
1: T_E.localize()
2: T_E.computeSpans(A)
3: for each leaf node t of T_E do
4:     if t.pred is opened and t.trgSpan != NULL then
5:         Node[] args ← T_E.getArgs(t)
6:         if all nodes in args are aligned then
7:             int[] srcOrder ← SORTSPANS(t.srcSpan, srcSpans of args)
8:             int[] trgOrder ← SORTSPANS(t.trgSpan, trgSpans of args)
9:             R.add(< t, args, srcOrder, trgOrder>)
10:        end if
11:    end if
12: end for
```

Algorithm 1 sketches the pre-ordering rule extraction algorithm guided by PASs. The algorithm collect pre-ordering rules through a traversal of the leaf nodes in an HPSG tree. A non-terminal node will not be accessed unless it is an argument of some predicate node(s). Thus, this algorithm runs in a time that is approximately *linear* to the number of leaf nodes in the tree, i.e., the number of words in the source sentence.

We define that a terminal node's PAS is *opened* if at least one of its arguments is neither empty nor unknown. We will not extract a pre-ordering rule if the terminal node is unaligned or any of its argument node is unaligned. These constraints are reflected by Line 4 and 6 in Algorithm 1. After heuristically sorting the source/target spans of a predicate node and its argument nodes, we finally extract a pre-ordering rule.

Table 2 summarizes the PAS-based pre-ordering rules extracted from the example shown in Figure 1. Application of these pre-ordering rules to the original English sentence yields the following Japanese style sentence:

- `the fluid pressure cylinder 31 is used when, fluid is gradually applied.`

2.4 Applying pre-ordering rules

[3]In this example, the overlapping is caused by the wrong/ambiguous alignments between "used" and "$naru_{15}$", and between "is" and "ha_6".

Word	PRED	Pre-ordering Rule
when	conj_arg12	when c3 c16 → c3 when c16
the	det_arg1	the c6 → the c6
fluid	adj_arg1	fluid c8 → fluid c8
pressure	noun_arg1	pressure c10 → pressure c10
cylinder	noun_arg0	-
31	adj_arg1	c11 31 → c11 31
is	aux_arg12	c4 is c15 → c4 is c15
used	verb_arg12	c4 used → c4 used
,	punct_arg1	, c18 → , c18
fluid	noun_arg0	-
is	aux_arg12	c19 is c23 → c19 is c23
gradually	adj_arg1	gradually c25 → gradually c25
applied	verb_arg12	c19 applied → c19 applied

Table 2: PAS-based pre-ordering rules extracted from the example shown in Figure 1. We use real words instead of predicate nodes here for intuitive understanding.

Algorithm 2 Pre-ordering Rule Application

Input: HPSG tree T_E of an English sentence $E[]$, rule set $\mathcal{R}$
Output: srcPhrase in the root node of T_E

```
1: T_E.localize()
2: T_E.computeSrcSpans()
3: mct_rule ← {}
4: for each leaf node t of T_E do
5:     Node[] args ← T_E.getArgs(t)
6:     int[] srcOrder ← SORTSPANS(t.srcSpan, srcSpans of args)
7:     Rule r ←RULEMATCH(R, < t, args, srcOrder>)
8:     if r != NULL then
9:         mct ← T_E.MCT(t, args)
10:        mct_rule.add(<mct, r >)
11:    end if
12: end for
13: for each mct in mct_rule in a bottom-up order do
14:    Rule r ← mct_rule.get(mct)
15:    mct.root().srcPhrase ← '' ▷ root() returns root node
16:    for i from 0 to r.trgOrder.length-1 do
17:        mct.root().srcPhrase += ' ' + mct.leaves()[r.trgOrder[i]].srcPhrase
18:    end for
19: end for
20: for each node n in T_E in a topological order do
21:    if n is a terminal node then
22:        n.srcPhrase ← E[n.srcSpan[0]]
23:    else if n.srcPhrase = NULL then
24:        n.srcPhrase ← CONNECT(n.children().srcPhrase)
25:    end if
26: end for
```

Algorithm 2 sketches the algorithm for applying pre-ordering rules to a given HPSG tree T_E. The algorithm contains three parts: rule matching (Lines 4-12), bottom-up rule applying (Lines 13-19), and sentence collecting (Lines 20-26). We first retrieve available pre-ordering rules from rule set $\mathcal{R}$ by a left-to-right traversal of the leaf nodes of T_E. For each leaf node, we select one pre-ordering rule with the highest frequency. Our experiments testified that this greedy rule selection strategy worked quite well. We selected 93% of the top frequent rule without facing a tie.

The terminal node t, the argument nodes of t, and their source-side ordering are taken as the key for rule matching. Available rules will be assigned to the MCT of t. Then, we apply the available rules to the root nodes of each MCT through a bottom-up traversal of T_E. A competitive problem is that, a non-terminal node can be shared by several MCTs. For example, node c3 and c18 (gray color) in Figure 1 are respectively shared by two MCTs (t6 and t7, t10 and t12). In order to avoid duplicated reordering of these nodes, we first pick the pre-ordering rule in which there are no "gaps" among the predicate words and argument phrases. For example, there is a gap (t6) between t7 and its argument node c4. We then pick a rule by frequency if there are still more than one rule available. Finally, after applying all available rules, we collect the pre-ordered source sentence from the root node of the HPSG tree.

3 Experiments

3.1 Setup

We test our proposal by translating from English to Japanese. We use the NTCIR-9 English-Japanese patent corpus[4] as our experiment set. Since the reference set of the official test set has not been released yet, we instead split the original development set averagely into two parts, named dev.a and dev.b. In our experiments, we first take dev.a as our development set for minimum-error rate tuning (Och, 2003) and then report the final translation accuracies on dev.b. For direct comparison with other systems in the future, we use the configuration of the official baseline system[5]:

- Moses[6] (Koehn et al., 2007): revision = "3717" as the baseline decoder. Note that we also train Moses using HFE sentences (Isozaki et al., 2010b) and the English sentences pre-ordered by PASs;
- GIZA++: giza-pp-v1.0.3[7] (Och and Ney, 2003) for first training word alignment using the original English sentences for pre-ordering rule extraction, and then for retrain-

[4] http://ntcir.nii.ac.jp/PatentMT/
[5] http://ntcir.nii.ac.jp/PatentMT/baselineSystems
[6] http://www.statmt.org/moses/
[7] http://giza-pp.googlecode.com/files/giza-pp-v1.0.3.tar.gz

	Train	Dev.a	Dev.b
# of sent.	2,032,679	1,000	1,000
# of En words	48,322,058	31,890	31,935
Enju suc. rate	99.3%	98.9%	98.7%
parse time (sec./sent.)	0.30	0.38	0.48
# of Jp words	53,865,629	37,066	35,921

Table 3: Statistics of the experiment sets.

ing word alignments using the pre-ordered English sentences;

- SRILM[8] (Stolcke, 2002): version 1.5.12 for training a 5-gram language model using the target sentences in the total training set;
- Additional scripts[9]: for preprocessing English sentences and cleaning up too long (# of words > 40) parallel sentences;
- Japanese word segmentation: Mecab v0.98[10] with the dictionary of mecab-ipadic-2.7.0-20070801.tar.gz[11].

The statistics of the filtered training set, dev.a, and dev.b are shown in Table 3. The success parsing rate ranges from 98.7% to 99.3% by using Enju2.3.1. The averaged parsing time for each English sentence ranges from 0.30 to 0.48 seconds.

3.2 Statistics of PASs and PAS-based pre-ordering rules

Figure 2 shows the number (natural log) of the 40 types of the PASs that appeared in the HPSG trees of the three experiment sets. Top five types of opened PASs include adj_arg1, det_arg1, prep_arg12, noun_arg1, and verb_arg12. By comparing the distributions of the number of PASs in the three sets, we can see that the distributions approximately share the same tendency. Thus, the pre-ordering rules learned from the PASs in the training set can be expected to be properly applied in dev.a and dev.b.

Besides, the statistics of the number of arguments for the predicate words is shown in Table 4. From this table, we find that the ratio of the number of arguments in the three sets are approximately similar. In particular, nearly half of the

# of args	Train	Dev.a	Dev.b
0	22.9%	22.4%	22.3%
1	47.0%	47.0%	47.5%
2	29.5%	29.8%	29.4%
3	0.6%	0.8%	0.8%
4	0.0%	0.0%	0.0%

Table 4: Statistics of the number of arguments of the predicate words in the experiment sets.

	Number	Ratio
Parse success	45,617,387	94.4%
Opened	35,004,893	76.7%
Aligned	33,966,923	97.0%
Contiguous	30,256,858	89.1%

Table 5: Statistics of predicate words in the training set for rule extraction.

predicate words have one argument. The number of predicate words that contain two arguments occurs around 30.0% of all the predicate words. Also, we can not extract pre-ordering rules from around 23.0% of the predicate words since they do not contain any arguments. Finally, less than 1% of predicate words contain three arguments and we only find one four-argument example of verb_arg1234 in the training set.

Now, in Table 5, we show the statistics of predicate words in the training set for pre-ordering rule extraction. Of the 48.3 million English words in the training set, there are 45.6 million words (94.4%) that are included in the HPSG trees that were successfully generated. Then, in the PASs of these 45.6 million words, there are 35.0 million words whose PASs are opened. We also list the number (34.0 million) of aligned predicate words, since we only extract pre-ordering rules from predicate words that are aligned to some target word(s) in Algorithm 1. Finally, there are 89.1% of aligned predicate words that are aligned to contiguous target words.

In order to investigate the sub-categorization effectiveness of the syntactic features included in the pre-ordering rules, we pick four subsets of the total feature set (Table 1). These feature subsets, named from PAS-a to PAS-d, are listed in Table 6. Through the comparison of these four feature subsets, we also attempt to investigate the data-sparseness problem of available pre-ordering rules cased by the factored features.

PAS-a includes all the syntactic features listed in Table 1. In PAS-b, we only keep three features for the predicate word and one feature for the argu-

[8] http://www.speech.sri.com/projects/srilm/

[9] http://homepages.inf.ed.ac.uk/jschroe1/how-to/scripts.tgz

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

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

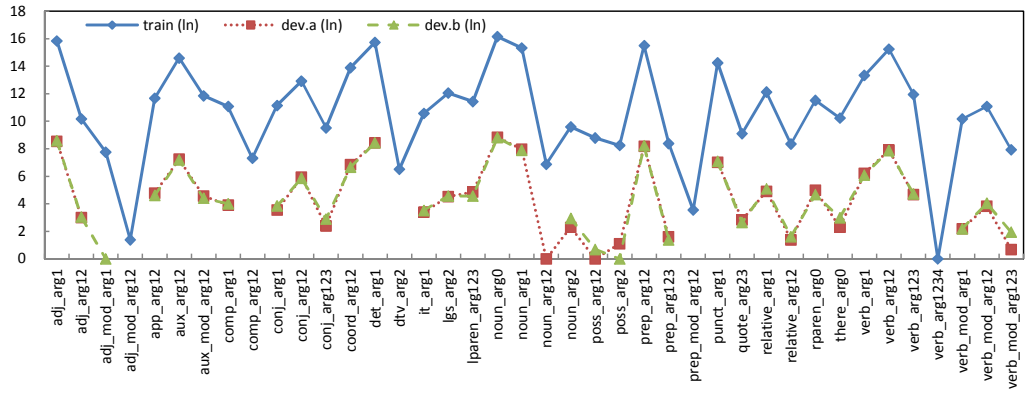

Figure 2: Number (natural log) of the types of the PASs that appeared in the experiment sets.

Feature	PAS-a	PAS-b	PAS-c	PAS-d
WORD	√	√	√	√
BASE	√			
POS	√			
LE	√			
PRED	√	√	√	√
CAT	√	√		
TENSE	√			
ASPECT	√			
VOICE	√			
AUX	√			
CAT	√	√	√	
XCAT	√			
HEAD	√			
SEM_HEAD	√			
SCHEMA	√			
# rules	469,014	203,184	200,968	148,047
# reorder	179,062	63,378	62,694	37,104
reorder ratio	38.2%	31.2%	31.2%	25.1%
avg. # train	12.0	12.1	12.1	12.1
avg. # dev.a	16.2	16.4	16.4	16.5
avg. # dev.b	16.2	16.4	16.4	16.5

Table 6: Feature subsets used in pre-ordering rules and statistics of the extraction and application of the pre-ordering rules under these feature subsets.

ment nodes. We further remove one feature (CAT) of the predicate word in PAS-c. In the fourth subset PAS-d, we only use two features WORD and PRED in the predicate word for sub-categorizing pre-ordering rules. Thus, PAS-d is only related to PASs (which can be generated by any kinds of parser) since it does not include additional features generated by the typical HPSG parser.

As the number of syntactic features decreases, more rules can be unified together. Thus, the number of pre-ordering rules and reordering rules, as shown in Table 6, also decreases. The number of reordering rules occurs from 25.1% (PAS-d) to 38.2% (PAS-a) in the pre-ordering rules. For each English sentence in the training set, there are averagely 12 *reordering* rules (instead of monotonic

Source sent.	BLEU	RIBES
Original sentences	0.2773	0.6619
PAS-a reordered	**0.3088**	**0.7406**
PAS-b reordered	0.3054	0.7334
PAS-c reordered	0.3063	0.7336
PAS-d reordered	0.3020	0.7265

Table 7: Translation accuracies by using the original English sentences or the pre-ordered English sentences under four types of pre-ordering rules.

pre-ordering rules) available under either of the four feature subsets. For each English sentence in dev.a and dev.b, the number of available *reordering* rules is averagely 16. Around 99.1%, 99.0%, and 98.6% English sentences were respectively reordered in the training set, dev.a set, and dev.b set.

3.3 Results

Table 7 shows the final translation accuracies under BLEU score (Papineni et al., 2002) and RIBES[12], i.e., the software implementation of Normalized Kendall's τ as proposed by (Isozaki et al., 2010a) to automatically evaluate the translation between distant language pairs based on rank correlation coefficients and significantly penalizes word order mistakes. Making use of our pre-ordered English sentences significantly ($p < 0.01$) improved BLEU scores from 2.47 (PAS-d) to 3.15 (PAS-a) points. The effectiveness of our proposal for tackling word-ordering problem can also be proved by comparing the scores of RIBES.

In addition, the accuracies change slightly among using the four types of pre-ordering rules. Among PAS-a, PAS-b, and PAS-c, we did significant test and could not differ them under $p < 0.01$ or $p < 0.05$. The only significant difference

[12]Code available at http://www.kecl.ntt.co.jp/icl/lirg/ribes

Source sent.	BLEU	RIBES	Same	PAS
HFE	0.3134	0.7370	-	-
HFE+PAS-a	0.3278	0.7379	11.0%	34.7%
HFE+PAS-b	**0.3302**	**0.7397**	12.3%	32.8%
HFE+PAS-c	0.3300	0.7380	10.8%	35.0%
HFE+PAS-d	0.3256	0.7337	11.5%	32.8%

Table 8: Translation accuracies by combining HFE and PAS based pre-ordering approach.

($p < 0.05$) appeared between PAS-a and PAS-d. Thus, we argue that the factored syntactic features such as WORD, PRED, and CAT are more essential for sub-categorizing pre-ordering rules than the remaining syntactic features.

As former mentioned, we also take the language-dependent HFE approach (Isozaki et al., 2010b) as another baseline. Note that word alignment was retrained using head-finalized English sentences and Japanese sentences in this HFE approach. Through comparing the HFE results listed in Table 8, we observe that the results are comparable between PAS-a and HFE: HFE is slightly better under BLEU score and PAS-a is slightly better under RIBES score.

Since similar HPSG parser (Enju) yet different linguistic information (syntactic head information vs. PASs) are used in HFE approach and our proposal. A straightforward question is whether we can combine these approaches together. Under this motivation, we select a better pre-ordered English sentence generated by the HFE method and our PAS-based method. Following (Genzel, 2010), we use *crossing score* as the metric for sentence selection. Crossing score is the number of crossing alignment links for a given aligned sentence pair. For monotonic alignments without reordering, crossing score is zero. During selection, we found that nearly 10% of the pre-ordered English sentences yielded by head-finalization and PAS-based methods were similar. In addition, among the different sentences, around 30% of PAS-based pre-ordering sentences were selected. Since we can not compute crossing score in the development/test sets, we instead take both kinds of pre-ordered English sentences as inputs and pick one output with a higher translation score.

The translation result based on this reselection approach is shown in Table 8. Compared with HFE approach, the reselection approach significantly ($p < 0.01$) improved BLEU scores of from 1.22 (PAS-d) to 1.68 (PAS-b) points. These interesting results reflect that syntactic head information and PASs describe the linguistic information of an English sentence in different aspects. Furthermore, compared with the single head-finalization rule, the automatically extracted pre-ordering rules kept the variety of word-ordering by dynamically inferring the word order of target sentences and thus enlarged the reordering space.

Source sent.	Averaged τ	$\tau \geq 0.8$
English	0.407	0.106
HFE	0.708	0.487
PAS-a	0.571	0.291
HFE+PAS-a	**0.809**	**0.643**

Table 9: Comparison of Kendall's τ.

3.4 Alignment comparison

In order to investigate how closely the pre-ordered English sentences follow target language word order, we measured Kendall's τ (Kendall, 1948), a rank correlation coefficient, as shown in Table 9. We exactly follow Isozaki et al. (2010b) to compute Kendall's τ. From Table 9, we can see that the quality of word alignments approximately reflects the final BLEU scores listed in Table 7 and 8.

4 Conclusion

We have proposed a pre-ordering approach by making use of predicate argument structures. The pre-ordering rules record the relative source-target position mapping among predicate words and their argument phrases. We first proposed an algorithm for automatically extracting these lexical pre-ordering rules from aligned HPSG-tree-to-string pairs. Then, we apply these pre-ordering rules to HPSG trees to yield pre-ordered source sentences that follow the word order of target sentences. Finally, we do word alignment again by using the pre-ordered source sentences together with the original target sentences.

Employing Moses (Koehn et al., 2007), our proposal significantly improved 2.47∼3.15 BLEU points compared with using the original English sentences. Combining with the HFE approach (Isozaki et al., 2010b), our approach significantly and impressively improved 5.29 points of BLEU score from 0.2773 to 0.3302. We finally argue that our proposal is not difficult to be implemented and can be easily applied to translate English into other languages.

References

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

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc.of ACL*, pages 263–270.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proc.of ACL*, pages 531–540.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc.of HLT-NAACL*.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proc.of COLING*, pages 376–384.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010a. Automatic evaluation of translation quality for distant language pairs. In *Proc.of EMNLP*, pages 944–952.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010b. Head finalization: A simple reordering rule for sov languages. In *Proc.of WMT-MetricsMATR*, pages 244–251.

Jason Katz-Brown and Michael Collins. 2007. Syntactic reordering in preprocessing for japanese-english translation: Mit system description for ntcir-7 patent translation task. In *Proc.of NTCIR-7 Workshop Meeting*, pages 409–414.

Maurice Kendall. 1948. *Rank Correlation Methods*. Charles Griffin.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *the ACL 2007 Demo-Poster*, pages 177–180.

Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2006. Phrase reordering for statistical machine translation based on predicate-argument structure. In *Proc.of IWSLT*, pages 77–82.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proc.of HLT-EMNLP*, pages 161–168.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment templates for statistical machine transaltion. In *Proc.of COLING-ACL*, pages 609–616, Sydney, Australia.

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

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

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

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

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Kay Rottmann and Stephan Vogel. 2007. Word reordering in statistical machine translation with a pos-based distortion model. In *Proc.of TMI*, pages 171–180, Skovde, Sweden.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proc.of ICSLP*, pages 901–904.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004*, pages 101–104.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *COLING*, pages 836–841.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proc.of EMNLP-CoNLL*, pages 737–745.

Xianchao Wu, Takuya Matsuzaki, and Jun'ichi Tsujii. 2010a. Fine-grained tree-to-string translation rule extraction. In *Proc.of the 48th ACL*, pages 325–334.

Xianchao Wu, Takuya Matsuzaki, and Jun'ichi Tsujii. 2010b. Improve syntax-based translation using deep syntactic structures. *Machine Translation (Special Issue : Pushing the frontiers of SMT)*, 24(2):141–157.

Fei Xia and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proc.of COLING*, pages 508–514.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proc.of COLING-ACL*, pages 521–528.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve smt for subject-object-verb languages. In *Proc.of HLT-NAACL*, pages 245–253.