Fine-grained Tree-to-String Translation Rule Extraction

Xianchao Wu† Takuya Matsuzaki† Jun’ichi Tsujii†‡∗
†Department of Computer Science, The University of Tokyo 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
‡School of Computer Science, University of Manchester
∗National Centre for Text Mining (NaCTeM)
Manchester Interdisciplinary Biocentre, 131 Princess Street, Manchester M1 7DN, UK

{wxc, matuzaki, tsujii}@is.s.u-tokyo.ac.jp

Abstract

Tree-to-string translation rules are widely used in linguistically syntax-based statistical machine translation systems. In this paper, we propose to use deep syntactic information for obtaining fine-grained translation rules. A head-driven phrase structure grammar (HPSG) parser is used to obtain the deep syntactic information, which includes a fine-grained description of the syntactic property and a semantic representation of a sentence. We extract fine-grained rules from aligned HPSG tree/forest-string pairs and use them in our tree-to-string and string-to-tree systems. Extensive experiments on large-scale bidirectional Japanese-English translations testified the effectiveness of our approach.

1 Introduction

Tree-to-string translation rules are generic and applicable to numerous linguistically syntax-based Statistical Machine Translation (SMT) systems, such as string-to-tree translation (Galley et al., 2004; Galley et al., 2006; Chiang et al., 2009), tree-to-string translation (Liu et al., 2006; Huang et al., 2006), and forest-to-string translation (Mi et al., 2008; Mi and Huang, 2008). The algorithms proposed by Galley et al. (2004; 2006) are frequently used for extracting minimal and composed rules from aligned 1-best tree-string pairs. Dealing with the parse error problem and rule sparseness problem, Mi and Huang (2008) replaced the 1-best parse tree with a packed forest which compactly encodes exponentially many parses for tree-to-string rule extraction.

However, current tree-to-string rules only make use of Probabilistic Context-Free Grammar tree fragments, in which part-of-speech (POS) or phrasal tags are used as the tree node labels. As will be testified by our experiments, we argue that the simple POS/phrasal tags are too coarse to reflect the accurate translation probabilities of the translation rules.

For example, as shown in Table 1, suppose a simple tree fragment “VBN(killed)” appears 6 times with “koroshita”, which is a Japanese translation of an active form of “killed”, and 4 times with “korosareta”, which is a Japanese translation of a passive form of “killed”. Then, without larger tree fragments, we will more frequently translate “VBN(killed)” into “koroshita” (with a probability of 0.6). But, “VBN(killed)” is indeed separable into two fine-grained tree fragments of “VBN(killed:active)” and “VBN(killed:passive)”. Consequently, “VBN(killed:active)” appears 5 times with “koroshita” and 1 time with “korosareta”; and “VBN(killed:passive)” appears 1 time with “koroshita” and 3 times with “korosareta”. Now, by attaching the voice information to “killed”, we are gaining a rule set that is more appropriate to reflect the real translation situations.

This motivates our proposal of using deep syntactic information to obtain a fine-grained translation rule set. We name the information such as the voice of a verb in a tree fragment as deep syntactic information. We use a head-driven phrase structure grammar (HPSG) parser to obtain the

<table>
<thead>
<tr>
<th>VBN(killed)</th>
<th>koroshita (active)</th>
<th>korosareta (passive)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 (6/10,6/6)</td>
<td>4 (4/10,4/4)</td>
<td></td>
</tr>
<tr>
<td>VBN(killed:active)</td>
<td>5 (5/6,5/6)</td>
<td>1 (1/6,1/4)</td>
</tr>
<tr>
<td>VBN(killed:passive)</td>
<td>1 (1/4,1/6)</td>
<td>3 (3/4,3/4)</td>
</tr>
</tbody>
</table>

Table 1: Bidirectional translation probabilities of rules, denoted in the brackets, change when voice is attached to “killed”.

1For example, “John has killed Mary.” versus “John was killed by Mary.”
deep syntactic information of an English sentence, which includes a fine-grained description of the syntactic property and a semantic representation of the sentence. We extract fine-grained translation rules from aligned HPSG tree/forest-string pairs. We localize an HPSG tree/forest to make it segmentable at any nodes to fit the extraction algorithms described in (Galley et al., 2006; Mi and Huang, 2008). We also propose a linear-time algorithm for extracting composed rules guided by predicate-argument structures. The effectiveness of the rules are testified in our tree-to-string and string-to-tree systems, taking bidirectional Japanese-English translations as our test cases.

This paper is organized as follows. In Section 2, we briefly review the tree-to-string and string-to-tree translation frameworks, tree-to-string rule extraction algorithms, and rich syntactic information previously used for SMT. The HPSG grammar and our proposal of fine-grained rule extraction algorithms are described in Section 3. Section 4 gives the experiments for applying fine-grained translation rules to large-scale Japanese-English translation tasks. Finally, we conclude in Section 5.

2 Related Work

2.1 Tree-to-string and string-to-tree translations

Tree-to-string translation (Liu et al., 2006; Huang et al., 2006) first uses a parser to parse a source sentence into a 1-best tree and then searches for the best derivation that segments and converts the tree into a target string. In contrast, string-to-tree translation (Galley et al., 2004; Galley et al., 2006; Chiang et al., 2009) is like bilingual parsing. That is, giving a (bilingual) translation grammar and a source sentence, we are trying to construct a parse forest in the target language. Consequently, the translation results can be collected from the leaves of the parse forest.

Figure 1 illustrates the training and decoding processes of bidirectional Japanese-English translations. The English sentence is “John killed Mary” and the Japanese sentence is “jyon ha mari wo koroshita”, in which the function words “ha” and “wo” are not aligned with any English word.

2.2 Tree/forest-based rule extraction

Galley et al. (2004) proposed the GHKM algorithm for extracting (minimal) tree-to-string translation rules from a tuple of \((F, E_t, A)\), where \(F = f^J_i\) is a sentence of a foreign language other than English, \(E_t\) is a 1-best parse tree of an English sentence \(E = e^t_j\), and \(A = \{(j, i)\}\) is an alignment between the words in \(F\) and \(E\).

The basic idea of GHKM algorithm is to decompose \(E_t\) into a series of tree fragments, each of which will form a rule with its corresponding translation in the foreign language. \(A\) is used as a constraint to guide the segmentation procedure, so that the root node of every tree fragment of \(E_t\) exactly corresponds to a contiguous span on the foreign language side. Based on this consideration, a frontier set \((fs)\) is defined to be a set of nodes \(n\) in \(E_t\) that satisfies the following constraint:

\[
fs = \{n | \text{span}(n) \cap \text{comp}_\text{span}(n) = \phi\}. \quad (1)
\]

Here, \(\text{span}(n)\) is defined by the indices of the first and last word in \(F\) that are reachable from a node \(n\), and \(\text{comp}_\text{span}(n)\) is defined to be the complement set of \(\text{span}(n)\), i.e., the union of the spans of all nodes \(n'\) in \(E_t\) that are neither descendants nor ancestors of \(n\). \(\text{span}(n)\) and \(\text{comp}_\text{span}(n)\) of each \(n\) can be computed by first a bottom-up exploration and then a top-down traversal of \(E_t\).

By restricting each fragment so that it only takes
the nodes in $fs$ as the root and leaf nodes, a well-formed fragmentation of $E_t$ is generated. With $fs$ computed, rules are extracted through a depth-first traversal of $E_t$: we cut $E_t$ at all nodes in $fs$ to form tree fragments and extract a rule for each fragment. These extracted rules are called minimal rules (Galley et al., 2004). For example, the 1-best tree (with gray nodes) in Figure 2 is cut into 7 pieces, each of which corresponds to the tree fragment in a rule (bottom-left corner of the figure).

In order to include richer context information and account for multiple interpretations of un-aligned words of foreign language, minimal rules which share adjacent tree fragments are connected together to form composed rules (Galley et al., 2006). For each aligned tree-string pair, Galley et al. (2006) constructed a derivation-forest, in which composed rules were generated, un-aligned words of foreign language were consistently attached, and the translation probabilities of rules were estimated by using Expectation-Maximization (EM) (Dempster et al., 1977) training. For example, by combining the minimal rules of 1, 4, and 5, we obtain a composed rule, as shown in the bottom-right corner of Figure 2.

Considering the parse error problem in the 1-best or $k$-best parse trees, Mi and Huang (2008) extracted tree-to-string translation rules from aligned packed forest-string pairs. A forest compactly encodes exponentially many trees

Figure 2: Illustration of an aligned HPSG forest-string pair. The forest includes two parse trees by taking “Mary” as a modifier ($t_3$, $t_4$) or an argument ($t_1$, $t_2$) of “killed”. Arrows with broken lines denote the PAS dependencies from the terminal node $t_1$ to its argument nodes ($c_1$ and $c_5$). The scores of the hyperedges are attached to the forest as well.
rather than the 1-best tree used by Galley et al. (2004; 2006). Two problems were managed to be tackled during extracting rules from an aligned forest-string pair: where to cut and how to cut. Equation 1 was used again to compute a frontier node set to determine where to cut the packed forest into a number of tree-fragments. The difference with tree-based rule extraction is that the nodes in a packed forest (which is a hypergraph) now are hypernodes, which can take a set of incoming hyperedges. Then, by limiting each fragment to be a tree and whose root/leaf hypernodes all appearing in the frontier set, the packed forest can be segmented properly into a set of tree fragments, each of which can be used to generate a tree-to-string translation rule.

2.3 Rich syntactic information for SMT

Before describing our approaches of applying deep syntactic information yielded by an HPSG parser for fine-grained rule extraction, we would like to briefly review what kinds of deep syntactic information have been employed for SMT.

Two kinds of supertags, from Lexicalized Tree-Adjoining Grammar and Combinatory Categorial Grammar (CCG), have been used as lexical syntactic descriptions (Hassan et al., 2007) for phrase-based SMT (Koehn et al., 2007). By introducing supertags into the target language side, i.e., the target language model and the target side of the phrase table, significant improvement was achieved for Arabic-to-English translation. Birch et al. (2007) also reported a significant improvement for Dutch-English translation by applying CCG supertags at a word level to a factorized SMT system (Koehn et al., 2007).

In this paper, we also make use of supertags on the English language side. In an HPSG parse tree, these lexical syntactic descriptions are included in the LEXENTRY feature (refer to Table 2) of a lexical node (Matsuzaki et al., 2007). For example, the LEXENTRY feature of “t1: killed” takes the value of [NP.nom<V.bse>NP.acc]_lxm-past _verb_rule in Figure 2. In which, [NP.nom<V.bse>NP.acc] is an HPSG style supertag, which tells us that the base form of “killed” needs a nominative NP in the left hand side and an accessorional NP in the right hand side. The major differences are that, we use a larger feature set (Table 2) including the supertags for fine-grained tree-to-string rule extraction, rather than string-to-string translation (Hassan et al., 2007; Birch et al., 2007).

The Logon project (Oepen et al., 2007) for Norwegian-English translation integrates in-depth grammatical analysis of Norwegian (using lexical functional grammar, similar to (Riezler and Maxwell, 2006)) with semantic representations in the minimal recursion semantics framework, and fully grammar-based generation for English using HPSG. A hybrid (of rule-based and data-driven) architecture with a semantic transfer backbone is taken as the vantage point of this project. In contrast, the fine-grained tree-to-string translation rule extraction approaches in this paper are totally data-driven, and easily applicable to numerous language pairs by taking English as the source or target language.

3 Fine-grained rule extraction

We now introduce the deep syntactic information generated by an HPSG parser and then describe our approaches for fine-grained tree-to-string rule extraction. Especially, we localize an HPSG tree/forest to fit the extraction algorithms described in (Galley et al., 2006; Mi and Huang, 2008). Also, we propose a linear-time composed rule extraction algorithm by making use of predicate-argument structures.

3.1 Deep syntactic information by HPSG parsing

Head-driven phrase structure grammar (HPSG) 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. Phrases and words represented by signs are composed into larger phrases by applications of schemata. The semantic representation of the new phrase is calculated at the same time. As such, an HPSG parse tree/forest can be considered as a tree/forest of signs (c.f. the HPSG forest in Figure 2).

An HPSG parse tree/forest has two attractive properties as a representation of an English sentence in syntax-based SMT. First, we can carefully control the condition of the application of a translation rule by exploiting the fine-grained syntactic
A sign in the HPSG tree/forest is represented by a typed feature structure (TFS) (Carpenter, 1992). A TFS is a directed-acyclic graph (DAG) wherein the edges are labeled with feature names and the nodes (feature values) are typed. In the original HPSG formalism, the types are defined in a hierarchy and the DAG can have arbitrary shape (e.g., it can be of any depth). However, we use a simplified form of TFS, for simplicity of the algorithms. In the simplified form, a TFS is converted to a (flat) set of pairs of feature names and their values. Table 2 lists the features used in this paper, which are a subset of those in the original output from an HPSG parser, Enju. The HPSG forest shown in Figure 2 is in this simplified format. An important detail is that we allow a feature value to be a pointer to another (simplified) TFS. Such pointer-valued features are necessary for denoting the semantics, as explained shortly.

In the Enju English HPSG grammar (Miyao et al., 2003) used in this paper, the semantic content of a sentence/phrase is represented by a predicate-argument structure (PAS). Figure 3 shows the PAS of the example sentence in Figure 2, “John killed Mary”, and a more complex PAS for another sentence, “She ignored the fact that I wanted to dispute”, which is adopted from (Miyao et al., 2003). In an HPSG tree/forest, each leaf node generally introduces a predicate, which is represented by the pair of LEXENTRY (lexical entry) feature and PRED (predicate type) feature. The arguments of a predicate are designated by the pointers from the ARG(x) features in a leaf node to non-terminal nodes.

### Table 2: Syntactic/semantic features extracted from HPSG signs that are included in the output of Enju. Features in phrasal nodes (top) and lexical nodes (bottom) are listed separately.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAT</td>
<td>phrasal category</td>
</tr>
<tr>
<td>XCAT</td>
<td>fine-grained phrasal category</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>name of the schema applied in the node</td>
</tr>
<tr>
<td>HEAD</td>
<td>pointer to the head daughter</td>
</tr>
<tr>
<td>SEM,HEAD</td>
<td>pointer to the semantic head daughter</td>
</tr>
<tr>
<td>CAT</td>
<td>syntactic category</td>
</tr>
<tr>
<td>POS</td>
<td>Penn Treebank-style part-of-speech tag</td>
</tr>
<tr>
<td>BASE</td>
<td>base form</td>
</tr>
<tr>
<td>TENSE</td>
<td>tense of a verb (past, present, untensed)</td>
</tr>
<tr>
<td>ASPECT</td>
<td>aspect of a verb (none, perfect, progressive, perfect-progressive)</td>
</tr>
<tr>
<td>VOICE</td>
<td>voice of a verb (passive, active)</td>
</tr>
<tr>
<td>AUX</td>
<td>auxiliary verb or not (minus, modal, have, be, do, to, copular)</td>
</tr>
<tr>
<td>LEXENTRY</td>
<td>lexical entry, with supertags embedded</td>
</tr>
<tr>
<td>PRED</td>
<td>type of a predicate</td>
</tr>
<tr>
<td>ARG(x)</td>
<td>pointer to semantic arguments, x = 1..4</td>
</tr>
</tbody>
</table>

3http://www-tsujii.is.s.u-tokyo.ac.jp/enju/index.html

Figure 3: Predicate argument structures for the sentences of “John killed Mary” and “She ignored the fact that I wanted to dispute”.

### 3.2 Localize HPSG forest

Our fine-grained translation rule extraction algorithm is sketched in Algorithm 1. Considering that a parse tree is a trivial packed forest, we only use the term forest to expand our discussion, hereafter. Recall that there are pointer-valued features in the TFSs (Table 2) which prevent arbitrary segmentation of a packed forest. Hence, we have to localize an HPSG forest.

For example, there are ARG pointers from t₁ to c₁ and c₅ in the HPSG forest of Figure 2. However, the three nodes are not included in one (minimal) translation rule. This problem is caused by not considering the predicate argument dependency among t₁, c₁, and c₅ while performing the GHKM algorithm. We can combine several minimal rules (Galley et al., 2006) together to address this dependency. Yet we have a faster way to tackle PASs, as will be described in the next subsection.

Even if we omit ARG, there are still two kinds of pointer-valued features in TFSs, HEAD and SEM,HEAD. Localizing these pointer-valued features is straightforward, since during parsing, the HEAD and SEM,HEAD of a node are automatically transferred to its mother node. That is, the syntactic and semantic head of a node only take
the identifier of the daughter node as the values. For example, HEAD and SEM\_HEAD of node $c_0$ take the identical value to be $c_0$ in Figure 2.

To extract tree-to-string rules from the tree structures of an HPSG forest, our solution is to pre-process an HPSG forest in the following way:

- for a phrasal hypernode, replace its HEAD and SEM\_HEAD value with $L$, $R$, or $S$, which respectively represent left daughter, right daughter, or single daughter (line 2 and 7); and,

- for a lexical node, ARG($x$) and PRED features are ignored (line 4).

A pure syntactic-based HPSG forest without any pointer-valued features can be yielded through this pre-processing for the consequent execution of the extraction algorithms (Galley et al., 2006; Mi and Huang, 2008).

### 3.3 Predicate-argument structures

In order to extract translation rules from PASs, we want to localize a predicate word and its arguments into one tree fragment. For example, in Figure 2, we can use a tree fragment which takes $c_0$ as its root node and $c_1$, $t_1$, and $c_3$ on its yield (= leaf nodes of a tree fragment) to cover “killed" and its subject and direct object arguments. We define this kind of tree fragment to be a minimum covering tree. For example, the minimum covering tree of $\{t_1, c_1, c_3\}$ is shown in the bottom-right corner of Figure 2. The definition supplies us a linear-time algorithm to directly find the tree fragment that covers a PAS during both rule extracting and rule matching when decoding an HPSG tree.

### 3.4 Experiments

#### 3.4.1 Translation models

We use a tree-to-string model and a string-to-tree model for bidirectional Japanese-English translations. Both models use a phrase translation table (PTT), an HPSG tree-based rule set (TRS), and a PAS-based rule set (PRS). Since the three rule sets are independently extracted and estimated, we
use Minimum Error Rate Training (MERT) (Och, 2003) to tune the weights of the features from the three rule sets on the development set.

Given a 1-best (localized) HPSG tree $E_t$, the tree-to-string decoder searches for the optimal derivation $d^*$ that transforms $E_t$ into a Japanese string among the set of all possible derivations $D$:

$$
d^* = \arg \max_{d \in D} \{ \lambda_1 \log p_{LM}(\tau(d)) + \lambda_2 |\tau(d)| + \log s(d|E_t) \}. \tag{2}$$

Here, the first item is the language model (LM) probability where $\tau(d)$ is the target string of derivation $d$; the second item is the translation length penalty; and the third item is the translation score, which is decomposed into a product of feature values of rules:

$$s(d|E_t) = \prod_{r \in d} f(r|PTT) f(r|TRS) f(r|PRS).$$

This equation reflects that the translation rules in one $d$ come from three sets. Inspired by (Liu et al., 2009b), it is appealing to combine these rule sets together in one decoder because PTT provides excellent rule coverages while TRS and PRS offer linguistically motivated phrase selections and non-local reorderings. Each $f(r)$ is in turn a product of five features:

$$f(r) = p(s|t)^{\lambda_3} \cdot p(t|s)^{\lambda_4} \cdot l(s|t)^{\lambda_5} \cdot l(t|s)^{\lambda_6} \cdot e^{\lambda_7}.$$ 

Here, $s/t$ represent the source/target part of a rule in PTT, TRS, or PRS; $p(\cdot|\cdot)$ and $l(\cdot|\cdot)$ are translation probabilities and lexical weights of rules from PTT, TRS, and PRS. The derivation length penalty is controlled by $\lambda_7$.

In our string-to-tree model, for efficient decoding with integrated n-gram LM, we follow (Zhang et al., 2006) and inversely binarize all translation rules into Chomsky Normal Forms that contain at most two variables and can be incrementally scored by LM. In order to make use of the binarized rules in the CKY decoding, we add two kinds of glue rules:

$$S \rightarrow X_m^{(1)}, X_m^{(1)}; \quad S \rightarrow S^{(1)} X_m^{(2)}, S^{(1)} X_m^{(2)}.$$ 

Here $X_m$ ranges over the nonterminals appearing in a binarized rule set. These glue rules can be seen as an extension from $X$ to $\{X_m\}$ of the two glue rules described in (Chiang, 2007).

The string-to-tree decoder searches for the optimal derivation $d^*$ that parses a Japanese string $F$ into a packed forest of the set of all possible derivations $D$:

$$d^* = \arg \max_{d \in D} \{ \lambda_1 \log p_{LM}(\tau(d)) + \lambda_2 |\tau(d)| + \lambda_3 g(d) + \log s(d|F) \}. \tag{3}$$

This formula differs from Equation 2 by replacing $E_t$ with $F$ in $s(d|\cdot)$ and adding $g(d)$, which is the number of glue rules used in $d$. Further definitions of $s(d|F)$ and $f(r)$ are identical with those used in Equation 2.

### 4.2 Decoding algorithms

In our translation models, we have made use of three kinds of translation rule sets which are trained separately. We perform derivation-level combination as described in (Liu et al., 2009b) for mixing different types of translation rules within one derivation.

For tree-to-string translation, we use a bottom-up beam search algorithm (Liu et al., 2006) for decoding an HPSG tree $E_t$. We keep at most 10 best derivations with distinct $\tau(d)$s at each node.

Recall the definition of minimum covering tree, which supports a faster way to retrieve available rules from PRS without generating all the sub-trees. That is, when node $n$ fortunately to be the root of some minimum covering tree(s), we use the tree(s) to seek available PAS-based rules in PRS. We keep a hash-table with the key to be the node identifier of $n$ and the value to be a priority queue of available PAS-based rules. The hash-table is easy to be filled by one-time traversal of the terminal nodes in $E_t$. At each terminal node, we seek its minimum covering tree, retrieve PRS, and update the hash-table. For example, suppose we are decoding an HPSG tree (with gray nodes) shown in Figure 2. At $t_1$, we can extract its minimum covering tree with the root node to be $c_0$, then take this tree fragment as the key to retrieve PRS, and consequently put $c_0$ and the available rules in the hash-table. When decoding at $c_0$, we can directly access the hash-table looking for available PAS-based rules.

In contrast, we use a CKY-style algorithm with beam-pruning and cube-pruning (Chiang, 2007) to decode Japanese sentences. For each Japanese sentence $F$, the output of the chart-parsing algorithm is expressed as a hypergraph representing a set of derivations. Given such a hypergraph, we
use the Algorithm 3 described in (Huang and Chiang, 2005) to extract its $k$-best ($k = 500$ in our experiments) derivations. Since different derivations may lead to the same target language string, we further adopt Algorithm 3’s modification, i.e., keep a hash-table to maintain the unique target sentences (Huang et al., 2006), to efficiently generate the unique $k$-best translations.

### 4.3 Setups

The JST Japanese-English paper abstract corpus\footnote{\url{http://www.jst.go.jp}. The corpus can be conditionally obtained from NTCIR-7 patent translation workshop homepage: \url{http://research.nii.ac.jp/ntcir/permission/ntcir-7/permiten-PATMT.html}.}, which consists of one million parallel sentences, was used for training and testing. This corpus was constructed from a Japanese-English paper abstract corpus by using the method of Utiyama and Isahara (2007). Table 3 shows the statistics of this corpus. Making use of Enju 2.3.1, we successfully parsed 987,401 English sentences in the training set, with a parse rate of 99.3%. We modified this parser to output a packed forest for each English sentence.

We executed GIZA++ (Och and Ney, 2003) and grow-diag-final-and balancing strategy (Koehn et al., 2007) on the training set to obtain a phrase-aligned parallel corpus, from which bidirectional phrase translation tables were estimated. SRI Language Modeling Toolkit (Stolcke, 2002) was employed to train 5-gram English and Japanese LMs on the training set. We evaluated the translation quality using the case-insensitive BLEU-4 metric (Papineni et al., 2002). The MERT toolkit we used is Z-mert\footnote{\url{http://www.cs.jhu.edu/ozaidan/zmert/}} (Zaidan, 2009).

The baseline system for comparison is Joshua (Li et al., 2009), a freely available decoder for hierarchical phrase-based SMT (Chiang, 2005). We respectively extracted 4.5M and 5.3M translation rules from the training set for the 4K English and Japanese sentences in the development and test sets. We used the default configuration of Joshua, except setting the maximum number of items/rules and the $k$ of $k$-best outputs to be the identical

<table>
<thead>
<tr>
<th>Tree nodes</th>
<th>TFS</th>
<th>POS</th>
<th>TFS</th>
<th>POS</th>
<th>TFS</th>
</tr>
</thead>
<tbody>
<tr>
<td># rules</td>
<td>0.9</td>
<td>62.1</td>
<td>83.9</td>
<td>92.5</td>
<td>103.7</td>
</tr>
<tr>
<td># tree types</td>
<td>0.4</td>
<td>23.5</td>
<td>34.7</td>
<td>40.6</td>
<td>45.2</td>
</tr>
<tr>
<td>extract time</td>
<td>3.5</td>
<td>-</td>
<td>98.6</td>
<td>-</td>
<td>121.2</td>
</tr>
</tbody>
</table>

Table 4: Statistics of several kinds of tree-to-string rules. Here, the number is in million level and the time is in hour.


We used four dual core Xeon machines (4×3.0GHz×2CPU, 4×64GB memory) to run all the experiments.

### 4.4 Results

Table 4 illustrates the statistics of several translation rule sets, which are classified by:

- using TFSs or simple POS/phrasal tags (annotated by a superscript $S$) to represent tree nodes;
- composed rules (PRS) extracted from the PAS of 1-best HPSG trees;
- composed rules ($C_3$), extracted from the tree structures of 1-best HPSG trees, and $3$ is the maximum number of internal nodes in the tree fragments; and
- forest-based rules ($F$), where the packed forests are pre-pruned by the marginal probability-based inside-outside algorithm used in (Mi and Huang, 2008).

Table 5 reports the BLEU-4 scores achieved by decoding the test set making use of Joshua and our systems (t2s = tree-to-string and s2t = string-to-tree) under numerous rule sets. We analyze this table in terms of several aspects to prove the effectiveness of deep syntactic information for SMT.

Let’s first look at the performance of TFSs. We take $C_3^S$ and $F^S$ as approximations of CFG-based translation rules. Comparing the BLEU-4 scores of PTT+$C_3^S$ and PTT+$C_3$, we gained 0.56 (t2s) and 0.57 (s2t) BLEU-4 points which are significant improvements ($p < 0.05$). Furthermore, we gained 0.50 (t2s) and 0.62 (s2t) BLEU-4 points from PTT+$F^S$ to PTT+$F$, which are also significant improvements ($p < 0.05$). The rich features included in TFSs contribute to these improvements.
<table>
<thead>
<tr>
<th>Systems</th>
<th>BLEU-t2s</th>
<th>Decoding</th>
<th>BLEU-s2t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joshua</td>
<td>21.79</td>
<td>0.486</td>
<td>19.73</td>
</tr>
<tr>
<td>PTT</td>
<td>18.40</td>
<td>0.013</td>
<td>17.21</td>
</tr>
<tr>
<td>PTT+PRS</td>
<td>22.12</td>
<td>0.031</td>
<td>19.33</td>
</tr>
<tr>
<td>PTT+CS</td>
<td>23.56</td>
<td>2.686</td>
<td>20.59</td>
</tr>
<tr>
<td>PTT+C3+PRS</td>
<td>24.12</td>
<td>2.753</td>
<td>21.16</td>
</tr>
<tr>
<td>PTT+F3</td>
<td>24.25</td>
<td>3.241</td>
<td>22.05</td>
</tr>
<tr>
<td>PTT+F</td>
<td>24.75</td>
<td>3.470</td>
<td>21.67</td>
</tr>
</tbody>
</table>

Table 5: BLEU-4 scores (%) achieved by Joshua and our systems under numerous rule configurations. The decoding time (seconds per sentence) of tree-to-string translation is listed as well.

Also, BLEU-4 scores were inspiringly increased 3.72 (t2s) and 2.12 (s2t) points by appending PRS to PTT, comparing PTT with PTT+PRS. Furthermore, in Table 5, the decoding time (seconds per sentence) of tree-to-string translation by using PTT+PRS is more than 86 times faster than using the other tree-to-string rule sets. This suggests that the direct generation of minimum covering trees for rule matching is extremely faster than generating all subtrees of a tree node. Note that PTT performed extremely bad compared with all other systems or tree-based rule sets. The major reason is that we did not perform any reordering or distorting during decoding with PTT.

However, in both t2s and s2t systems, the BLEU-4 score benefits of PRS were covered by the composed rules: both PTT+CS and PTT+C3 performed significant better ($p < 0.01$) than PTT+PRS, and there are no significant differences when appending PRS to PTT+C3. The reason is obvious: PRS is only a small subset of the composed rules, and the probabilities of rules in PRS were estimated by maximum likelihood, which is fast but biased compared with EM based estimation (Galley et al., 2006).

Finally, by using PTT+F, our systems achieved the best BLEU-4 scores of 24.75% (t2s) and 22.67% (s2t), both are significantly better ($p < 0.01$) than that achieved by Joshua.

5 Conclusion

We have proposed approaches of using deep syntactic information for extracting fine-grained tree-to-string translation rules from aligned HPSG forest-string pairs. The main contributions are the applications of GHKM-related algorithms (Galley et al., 2006; Mi and Huang, 2008) to HPSG forests and a linear-time algorithm for extracting composed rules from predicate-argument structures. We applied our fine-grained translation rules to a tree-to-string system and an Hiero-style string-to-tree system. Extensive experiments on large-scale bidirectional Japanese-English translations testified the significant improvements on BLEU score.

We argue the fine-grained translation rules are generic and applicable to many syntax-based SMT frameworks such as the forest-to-string model (Mi et al., 2008). Furthermore, it will be interesting to extract fine-grained tree-to-tree translation rules by integrating deep syntactic information in the source and/or target language side(s). These tree-to-tree rules are applicable for forest-to-tree translation models (Liu et al., 2009a).

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan) and Japanese/Chinese Machine Translation Project in Special Coordination Funds for Promoting Science and Technology (MEXT, Japan), and Microsoft Research Asia Machine Translation Theme. The first author thanks Naoaki Okazaki and Yusuke Miyao for their help and the anonymous reviewers for improving the earlier version.

References


Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In Proceedings of 7th AMTA.


