Adjoining Tree-to-String Translation

Yang Liu, Qun Liu, and **Yajuan Lü**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{yliu,liuqun,lvyajuan}@ict.ac.cn

Abstract

We introduce synchronous tree adjoining grammars (TAG) into tree-to-string translation, which converts a source tree to a target string. Without reconstructing TAG derivations explicitly, our rule extraction algorithm directly learns tree-to-string rules from aligned Treebank-style trees. As tree-to-string translation casts decoding as a tree parsing problem rather than parsing, the decoder still runs fast when adjoining is included. Less than 2 times slower, the adjoining tree-to-string system improves translation quality by +0.7 BLEU over the baseline system only allowing for tree substitution on NIST Chinese-English test sets.

1 Introduction

Syntax-based translation models, which exploit hierarchical structures of natural languages to guide machine translation, have become increasingly popular in recent years. So far, most of them have been based on synchronous context-free grammars (CFG) (Chiang, 2007), tree substitution grammars (TSG) (Eisner, 2003; Galley et al., 2006; Liu et al., 2006; Huang et al., 2006; Zhang et al., 2008), and inversion transduction grammars (ITG) (Wu, 1997; Xiong et al., 2006). Although these formalisms present simple and precise mechanisms for describing the basic recursive structure of sentences, they are not powerful enough to model some important features of natural language syntax. For example, Chiang (2006) points out that the translation of languages that can stack an unbounded number of clauses in an "inside-out" way (Wu, 1997) provably goes beyond the expressive power of synchronous CFG and TSG. Therefore, it is necessary to find ways to take advantage of more powerful synchronous grammars to improve machine translation.

Synchronous tree adjoining grammars (TAG) (Shieber and Schabes, 1990) are a good candidate. As a formal tree rewriting system, TAG (Joshi et al., 1975; Joshi, 1985) provides a larger domain of locality than CFG to state linguistic dependencies that are far apart since the formalism treats trees as basic building blocks. As a mildly context-sensitive grammar, TAG is conjectured to be powerful enough to model natural languages. Synchronous TAG generalizes TAG by allowing the construction of a pair of trees using the TAG operations of substitution and adjoining on tree pairs. The idea of using synchronous TAG in machine translation has been pursued by several researchers (Abeille et al., 1990; Prigent, 1994; Dras, 1999), but only recently in its probabilistic form (Nesson et al., 2006; DeNeefe and Knight, 2009). Shieber (2007) argues that probabilistic synchronous TAG possesses appealing properties such as expressivity and trainability for building a machine translation system.

However, one major challenge for applying synchronous TAG to machine translation is computational complexity. While TAG requires $O(n^6)$ time for monolingual parsing, synchronous TAG requires $O(n^{12})$ for bilingual parsing. One solution is to use tree insertion grammars (TIG) introduced by Schabes and Waters (1995). As a restricted form of TAG, TIG still allows for adjoining of unbounded trees but only requires $O(n^3)$ time for monolingual parsing. Nesson et al. (2006) firstly demonstrate

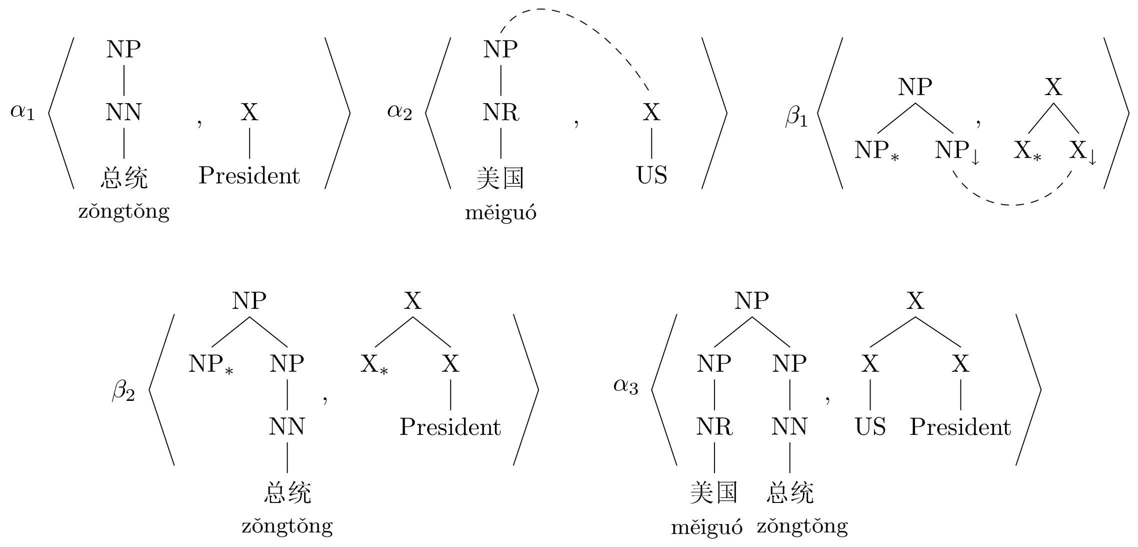


Figure 1: Initial and auxiliary tree pairs. The source side (Chinese) is a Treebank-style linguistic tree. The target side (English) is a purely structural tree using a single non-terminal (X). By convention, substitution and foot nodes are marked with a down arrow ($\downarrow$) and an asterisk ($*$), respectively. The dashed lines link substitution sites (e.g., NP$_\downarrow$ and X$_\downarrow$ in β_1) and adjoining sites (e.g., NP and X in α_2) in tree pairs. Substituting the initial tree pair α_1 at the NP$_\downarrow$-X$_\downarrow$ node pair in the auxiliary tree pair β_1 yields a derived tree pair β_2, which can be adjoined at NN-X in α_2 to generate α_3.

the use of synchronous TIG for machine translation and report promising results. DeNeefe and Knight (2009) prove that adjoining can improve translation quality significantly over a state-of-the-art string-to-tree system (Galley et al., 2006) that uses synchronous TSG with tractable computational complexity.

In this paper, we introduce synchronous TAG into tree-to-string translation (Liu et al., 2006; Huang et al., 2006), which is the simplest and fastest among syntax-based approaches (Section 2). We propose a new rule extraction algorithm based on GHKM (Galley et al., 2004) that directly induces a synchronous TAG from an aligned and parsed bilingual corpus without converting Treebank-style trees to TAG derivations explicitly (Section 3). As tree-to-string translation takes a source parse tree as input, the decoding can be cast as a tree parsing problem (Eisner, 2003): reconstructing TAG derivations from a derived tree using tree-to-string rules that allow for both substitution and adjoining. We describe how to convert TAG derivations to translation forest (Section 4). We evaluated the new tree-to-string system on NIST Chinese-English tests and obtained consistent improvements (+0.7 BLEU) over the STSG-based baseline system without significant loss in efficiency (1.6 times slower) (Section 5).

2 Model

A synchronous TAG consists of a set of linked elementary tree pairs: **initial** and **auxiliary**. An initial tree is a tree of which the interior nodes are all labeled with non-terminal symbols, and the nodes on the frontier are either words or non-terminal symbols marked with a down arrow ($\downarrow$). An auxiliary tree is defined as an initial tree, except that exactly one of its frontier nodes must be marked as foot node ($*$). The foot node must be labeled with a non-terminal symbol that is the same as the label of the root node.

Synchronous TAG defines two operations to build derived tree pairs from elementary tree pairs: **substitution** and **adjoining**. Nodes in initial and auxiliary tree pairs are linked to indicate the correspondence between substitution and adjoining sites. Figure 1 shows three initial tree pairs (i.e., α_1, α_2, and α_3) and two auxiliary tree pairs (i.e., β_1 and β_2). The dashed lines link substitution nodes (e.g., NP$_\downarrow$ and X$_\downarrow$ in β_1) and adjoining sites (e.g., NP and X in α_2) in tree pairs. Substituting the initial tree pair α_1 at

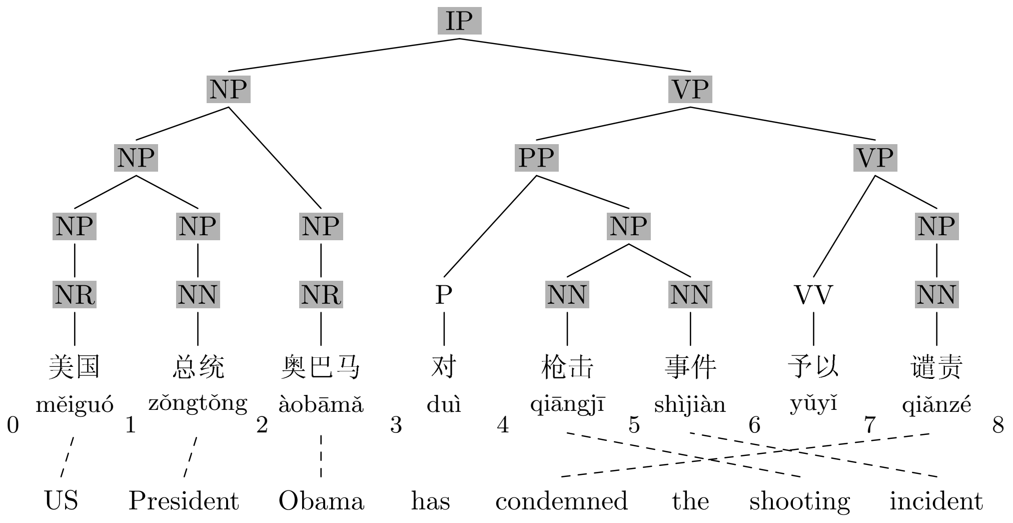


Figure 2: A training example. Tree-to-string rules can be extracted from shaded nodes.

node	minimal initial rule	minimal auxiliary rule
NR$_{0,1}$	[1] (NR měiguó) → US	
NP$_{0,1}$	[2] (NP (x_1:NR$_{\downarrow}$)) → x_1	
NN$_{1,2}$	[3] (NN zǒngtǒng) → President	
NP$_{1,2}$	[4] (NP (x_1:NN$_{\downarrow}$)) → x_1	
NP$_{0,2}$	[5] (NP (x_1:NP$_{\downarrow}$) (x_2:NP$_{\downarrow}$)) → x_1 x_2	
	[6] (NP$_{0:1}$ (x_1:NR$_{\downarrow}$)) → x_1	[7] (NP (x_1:NP$_{*}$) (x_2:NP$_{\downarrow}$)) → x_1 x_2
		[8] (NP$_{0:2}$ (x_1:NP$_{*}$) (x_2:NP$_{\downarrow}$)) → x_1 x_2
	[9] (NP$_{0:1}$ (x_1:NN$_{\downarrow}$)) → x_1	[10] (NP (x_1:NP$_{\downarrow}$) (x_2:NP$_{*}$)) → x_1 x_2
		[11] (NP$_{0:2}$ (x_1:NP$_{\downarrow}$) (x_2:NP$_{*}$)) → x_1 x_2
NR$_{2,3}$	[12] (NR àobāmǎ) → Obama	
NP$_{2,3}$	[13] (NP (x_1:NR$_{\downarrow}$)) → x_1	
NP$_{0,3}$	[14] (NP (x_1:NP$_{\downarrow}$) (x_2:NP$_{\downarrow}$)) → x_1 x_2	
	[15] (NP$_{0:2}$ (x_1:NP$_{\downarrow}$) (x_2:NP$_{\downarrow}$)) → x_1 x_2	[16] (NP (x_1:NP$_{*}$) (x_2:NP$_{\downarrow}$)) → x_1 x_2
	[17] (NP$_{0:1}$ (x_1:NR$_{\downarrow}$)) → x_1	[18] (NP (x_1:NP$_{\downarrow}$) (x_2:NP$_{*}$)) → x_1 x_2
	[19] (NP$_{0:1}$ (x_1:NN$_{\downarrow}$)) → x_1	
	[20] (NP$_{0:1}$ (x_1:NR$_{\downarrow}$)) → x_1	
NN$_{4,5}$	[21] (NN qiāngjī) → shooting	
NN$_{5,6}$	[22] (NN shìjiàn) → incident	
NP$_{4,6}$	[23] (NP (x_1:NN$_{\downarrow}$) (x_2:NN$_{\downarrow}$)) → x_1 x_2	
PP$_{3,6}$	[24] (PP (duì) (x_1:NP$_{\downarrow}$)) → x_1	
NN$_{7,8}$	[25] (NN qiǎnzé) → condemned	
NP$_{7,8}$	[26] (NP (x_1:NN$_{\downarrow}$)) → x_1	
VP$_{6,8}$	[27] (VP (VV yǔyǐ) (x_1:NP$_{\downarrow}$)) → x_1	
VP$_{3,8}$	[28] (VP (x_1:PP$_{\downarrow}$) (x_2:VP$_{\downarrow}$)) → x_2 the x_1	
	[29] (VP$_{0:1}$ (VV yǔyǐ) (x_1:NP$_{\downarrow}$)) → x_1	[30] (VP (x_1:PP$_{\downarrow}$) (x_2:VP$_{*}$)) → x_2 the x_1
IP$_{0,8}$	[31] (IP (x_1:NP$_{\downarrow}$) (x_2:VP$_{\downarrow}$)) → x_1 has x_2	

Table 1: Minimal initial and auxiliary rules extracted from Figure 2. Note that an adjoining site has a span as subscript. For example, NP$_{0:1}$ in rule 6 indicates that the node is an adjoining site linked to a target node dominating the target string spanning from position 0 to position 1 (i.e., x_1). The target tree is hidden because tree-to-string translation only considers the target surface string.

the NP$_\downarrow$-X$_\downarrow$ node pair in the auxiliary tree pair β_1 yields a derived tree pair β_2, which can be adjoined at NN-X in α_2 to generate α_3.

For simplicity, we represent α_2 as a tree-to-string rule:

$$\text{(NP}_{0:1}\text{ (NR měiguó))} \rightarrow \text{US}$$

where NP$_{0:1}$ indicates that the node is an adjoining site linked to a target node dominating the target string spanning from position 0 to position 1 (i.e., "US"). The target tree is hidden because tree-to-string translation only considers the target surface string. Similarly, β_1 can be written as

$$\text{(NP (} x_1\text{:NP}_* \text{) (} x_2\text{:NP}_\downarrow \text{))} \rightarrow x_1\ x_2$$

where x denotes a non-terminal and the subscripts indicate the correspondence between source and target non-terminals.

The parameters of a probabilistic synchronous TAG are

$$\sum_{\alpha} P_i(\alpha) = 1 \qquad (1)$$

$$\sum_{\alpha} P_s(\alpha|\eta) = 1 \qquad (2)$$

$$\sum_{\beta} P_a(\beta|\eta) + P_a(\text{NONE}|\eta) = 1 \qquad (3)$$

where α ranges over initial tree pairs, β over auxiliary tree pairs, and η over node pairs. $P_i(\alpha)$ is the probability of beginning a derivation with α; $P_s(\alpha|\eta)$ is the probability of substituting α at η; $P_a(\beta|\eta)$ is the probability of adjoining β at η; finally, $P_a(\text{NONE}|\eta)$ is the probability of nothing adjoining at η.

For tree-to-string translation, these parameters can be treated as feature functions of a discriminative framework (Och, 2003) combined with other conventional features such as relative frequency, lexical weight, rule count, language model, and word count (Liu et al., 2006).

3 Rule Extraction

Inducing a synchronous TAG from training data often begins with converting Treebank-style parse trees to TAG derivations (Xia, 1999; Chen and Vijay-Shanker, 2000; Chiang, 2003). DeNeefe and Knight (2009) propose an algorithm to extract synchronous TIG rules from an aligned and parsed bilingual corpus. They first classify tree nodes into heads, arguments, and adjuncts using heuristics (Collins, 2003), then transform a Treebank-style tree into a TIG derivation, and finally extract minimally-sized rules from the derivation tree and the string on the other side, constrained by the alignments. Probabilistic models can be estimated by collecting counts over the derivation trees.

However, one challenge is that there are many TAG derivations that can yield the same derived tree, even with respect to a single grammar. It is difficult to choose appropriate single derivations that enable the resulting grammar to translate unseen data well. DeNeefe and Knight (2009) indicate that the way to reconstruct TIG derivations has a direct effect on final translation quality. They suggest that one possible solution is to use derivation forest rather than a single derivation tree for rule extraction.

Alternatively, we extend the GHKM algorithm (Galley et al., 2004) to *directly* extract tree-to-string rules that allow for both substitution and adjoining from aligned and parsed data. There is no need for transforming a parse tree into a TAG derivation explicitly before rule extraction and all derivations can be easily reconstructed using extracted rules. [1] Our rule extraction algorithm involves two steps: (1) extracting minimal rules and (2) composition.

3.1 Extracting Minimal Rules

Figure 2 shows a training example, which consists of a Chinese parse tree, an English string, and the word alignment between them. By convention, shaded nodes are called **frontier** nodes from which tree-to-string rules can be extracted. Note that the source phrase dominated by a frontier node and its corresponding target phrase are consistent with the word alignment: all words in the source phrase are aligned to all words in the corresponding target phrase and vice versa.

We distinguish between three categories of tree-

[1]Note that our algorithm does not take heads, complements, and adjuncts into consideration and extracts all possible rules with respect to word alignment. Our hope is that this treatment would make our system more robust in the presence of noisy data. It is possible to use the linguistic preferences as features. We leave this for future work.

to-string rules:

1. **substitution rules**, in which the source tree is an initial tree without adjoining sites.

2. **adjoining rules**, in which the source tree is an initial tree with at least one adjoining site.

3. **auxiliary rules**, in which the source tree is an auxiliary tree.

For example, in Figure 1, α_1 is a substitution rule, α_2 is an adjoining rule, and β_1 is an auxiliary rule.

Minimal substitution rules are the same with those in STSG (Galley et al., 2004; Liu et al., 2006) and therefore can be extracted directly using GHKM. By minimal, we mean that the interior nodes are not frontier and cannot be decomposed. For example, in Table 2, rule 1 (for short r_1) is a minimal substitution rule extracted from $\text{NR}_{0,1}$.

Minimal adjoining rules are defined as minimal substitution rules, except that each root node must be an adjoining site. In Table 2, r_2 is a minimal substitution rule extracted from $\text{NP}_{0,1}$. As $\text{NP}_{0,1}$ is a descendant of $\text{NP}_{0,2}$ with the same label, $\text{NP}_{0,1}$ is a possible adjoining site. Therefore, r_6 can be derived from r_2 and licensed as a minimal adjoining rule extracted from $\text{NP}_{0,2}$. Similarly, four minimal adjoining rules are extracted from $\text{NP}_{0,3}$ because it has four frontier descendants labeled with NP.

Minimal auxiliary rules are derived from minimal substitution and adjoining rules. For example, in Table 2, r_7 and r_{10} are derived from the minimal substitution rule r_5 while r_8 and r_{11} are derived from r_{15}. Note that a minimal auxiliary rule can have adjoining sites (e.g., r_8).

Table 1 lists 17 minimal substitution rules, 7 minimal adjoining rules, and 7 minimal auxiliary rules extracted from Figure 2.

3.2 Composition

We can obtain composed rules that capture rich contexts by substituting and adjoining minimal initial and auxiliary rules. For example, the composition of r_{12}, r_{17}, r_{25}, r_{26}, r_{29}, and r_{31} yields an initial rule with two adjoining sites:

(IP ($\text{NP}_{0:1}$ (NR àobāmǎ)) ($\text{VP}_{2:3}$ (VV yǔyǐ) (NP (NN qiǎnzé)))) $\rightarrow$ Obama has condemned

Note that the source phrase "àobāmǎ . . . yǔyǐ qiǎnzé" is discontinuous. Our model allows both the source and target phrases of an initial rule with adjoining sites to be discontinuous, which goes beyond the expressive power of synchronous CFG and TSG.

Similarly, the composition of two auxiliary rules r_8 and r_{16} yields a new auxiliary rule:

(NP (NP (x_1:NP_*) (x_2:$\text{NP}_\downarrow$)) (x_3:$\text{NP}_\downarrow$)) $\rightarrow x_1 x_2 x_3$

We first compose initial rules and then compose auxiliary rules, both in a bottom-up way. To maintain a reasonable grammar size, we follow Liu (2006) to restrict that the tree height of a rule is no greater than 3 and the source surface string is no longer than 7.

To learn the probability models $P_i(\alpha)$, $P_s(\alpha|\eta)$, $P_a(\beta|\eta)$, and $P_a(\text{NONE}|\eta)$, we collect and normalize counts over these extracted rules following DeNeefe and Knight (2009).

4 Decoding

Given a synchronous TAG and a derived source tree π, a tree-to-string decoder finds the English yield of the best derivation of which the Chinese yield matches π:

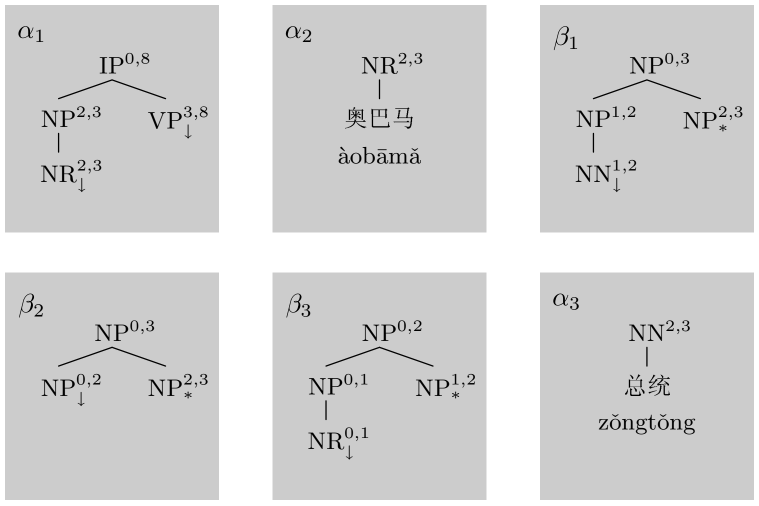
$$\hat{e} = e\left(\underset{D \text{ s.t. } f(D)=\pi}{\arg\max} P(D) \right) \tag{4}$$

This is called **tree parsing** (Eisner, 2003) as the decoder finds ways of decomposing π into elementary trees.

Tree-to-string decoding with STSG is usually treated as **forest rescoring** (Huang and Chiang, 2007) that involves two steps. The decoder first converts the input tree into a translation forest using a translation rule set by pattern matching. Huang et al. (2006) show that this step is a depth-first search with memorization in $O(n)$ time. Then, the decoder searches for the best derivation in the translation forest intersected with n-gram language models and outputs the target string. [2]

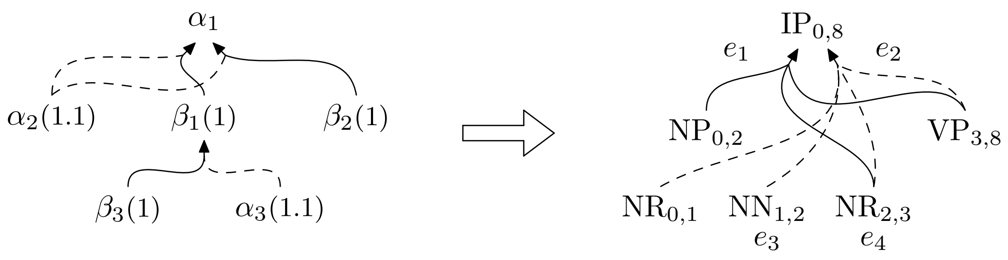
Decoding with STAG, however, poses one major challenge to forest rescoring. As translation forest only supports substitution, it is difficult to construct a translation forest for STAG derivations because of

[2]Mi et al. (2008) give a detailed description of the two-step decoding process. Huang and Mi (2010) systematically analyze the decoding complexity of tree-to-string translation.



elementary tree		translation rule
α_1	r_1	(IP (NP$_{0:1}$ (x_1:NR$_{\downarrow}$)) (x_2:VP$_{\downarrow}$)) $\rightarrow x_1\ x_2$
α_2	r_2	(NR àobāmǎ) $\rightarrow$ Obama
β_1	r_3	(NP (NP$_{0:1}$ (x_1:NN$_{\downarrow}$)) (x_2:NP$_{*}$)) $\rightarrow x_1\ x_2$
β_2	r_4	(NP (x_1:NP$_{\downarrow}$) (x_2:NP$_{*}$)) $\rightarrow x_1\ x_2$
β_3	r_5	(NP (NP (x_1:NR$_{\downarrow}$)) (x_2:NP$_{*}$)) $\rightarrow x_1\ x_2$
α_3	r_6	(NN zǒngtǒng) $\rightarrow$ President

Figure 3: Matched trees and corresponding rules. Each node in a matched tree is annotated with a span as superscript to facilitate identification. For example, IP^{0,8} in α_1 indicates that IP$_{0,8}$ in Figure 2 is matched. Note that its left child NP^{2,3} is not its direct descendant in Figure 2, suggesting that adjoining is required at this site.



hyperedge		translation rule
e_1	$r_1 + r_4$	(IP (NP (x_1:NP$_{\downarrow}$) (NP (x_2:NR$_{\downarrow}$))) (x_3:VP$_{\downarrow}$) $\rightarrow x_1\ x_2\ x_3$
e_2	$r_1 + r_3 + r_5$	(IP (NP (NP (x_1:NP$_{\downarrow}$) (x_2:NP$_{\downarrow}$)) (NP (x_3:NR$_{\downarrow}$))) (x_4:VP$_{\downarrow}$)) $\rightarrow x_1\ x_2\ x_3\ x_4$
e_3	r_6	(NN zǒngtǒng) $\rightarrow$ President
e_4	r_2	(NR àobāmǎ) $\rightarrow$ Obama

Figure 4: Converting a derivation forest to a translation forest. In a derivation forest, a node in a derivation forest is a matched elementary tree. A hyperedge corresponds to operations on related trees: substitution (dashed) or adjoining (solid). We use Gorn addresses as tree addresses. $\alpha_2(1.1)$ denotes that α_2 is substituted in the tree α_1 at the node NR$^{2,3}_{\downarrow}$ of address 1.1 (i.e., the first child of the first child of the root node). As translation forest only supports substitution, we combine trees with adjoining sites to form an equivalent tree without adjoining sites. Rules are composed accordingly (e.g., $r_1 + r_4$).

adjoining. Therefore, we divide forest rescoring for STAG into three steps:

1. **matching**, matching STAG rules against the input tree to obtain a TAG derivation forest;

2. **conversion**, converting the TAG derivation forest into a translation forest;

3. **intersection**, intersecting the translation forest with an n-gram language model.

Given a tree-to-string rule, rule matching is to find a subtree of the input tree that is identical to the source side of the rule. While matching STSG rules against a derived tree is straightforward, it is somewhat non-trivial for STAG rules that move beyond nodes of a local tree. We follow Liu et al. (2006) to enumerate all elementary subtrees and match STAG rules against these subtrees. This can be done by first enumerating all minimal initial and auxiliary trees and then combining them to obtain composed trees, assuming that every node in the input tree is frontier (see Section 3). We impose the same restrictions on the tree height and length as in rule extraction. Figure 3 shows some matched trees and corresponding rules. Each node in a matched tree is annotated with a span as superscript to facilitate identification. For example, $\text{IP}^{0,8}$ in α_1 means that $\text{IP}_{0,8}$ in Figure 2 is matched. Note that its left child $\text{NP}^{2,3}$ is not its direct descendant in Figure 2, suggesting that adjoining is required at this site.

A **TAG derivation tree** specifies uniquely how a derived tree is constructed using elementary trees (Joshi, 1985). A node in a derivation tree is an elementary tree and an edge corresponds to operations on related elementary trees: substitution or adjoining. We introduce **TAG derivation forest**, a compact representation of multiple TAG derivation trees, to encodes all matched TAG derivation trees of the input derived tree.

Figure 4 shows part of a TAG derivation forest. The six matched elementary trees are nodes in the derivation forest. Dashed and solid lines represent substitution and adjoining, respectively. We use Gorn addresses as tree addresses: 0 is the address of the root node, p is the address of the p^{th} child of the root node, and $p \cdot q$ is the address of the q^{th} child of the node at the address p. The derivation forest should be interpreted as follows: α_2 is substituted in the tree α_1 at the node $\text{NR}_{\downarrow}^{2,3}$ of address 1.1 (i.e., the first child of the first child of the root node) and β_1 is adjoined in the tree α_1 at the node $\text{NP}^{2,3}$ of address 1.

To take advantage of existing decoding techniques, it is necessary to convert a derivation forest to a **translation forest**. A hyperedge in a translation forest corresponds to a translation rule. Mi et al. (2008) describe how to convert a derived tree to a translation forest using tree-to-string rules only allowing for substitution. Unfortunately, it is not straightforward to convert a derivation forest including adjoining to a translation forest. To alleviate this problem, we combine initial rules with adjoining sites and associated auxiliary rules to form *equivalent* initial rules without adjoining sites on the fly during decoding.

Consider α_1 in Figure 3. It has an adjoining site $\text{NP}^{2,3}$. Adjoining β_2 in α_1 at the node $\text{NP}^{2,3}$ produces an equivalent initial tree with only substitution sites:

$$(\ \text{IP}^{0,8}\ (\ \text{NP}^{0,3}\ (\ \text{NP}_{\downarrow}^{0,2}\ )\ (\ \text{NP}^{2,3}\ (\ \text{NR}_{\downarrow}^{2,3}\ )\ )\ )\ (\ \text{VP}_{\downarrow}^{3,8}\ )\ )$$

The corresponding composed rule $r_1 + r_4$ has no adjoining sites and can be added to translation forest.

We define that the elementary trees needed to be composed (e.g., α_1 and β_2) form a **composition tree** in a derivation forest. A node in a composition tree is a matched elementary tree and an edge corresponds to adjoining operations. The root node must be an initial tree with at least one adjoining site. The descendants of the root node must all be auxiliary trees. For example, (α_1 (β_2)) and (α_1 (β_1 (β_3))) are two composition trees in Figure 4. The number of children of a node in a composition tree depends on the number of adjoining sites in the node. We use **composition forest** to encode all possible composition trees.

Often, a node in a composition tree may have multiple matched rules. As a large amount of composition trees and composed rules can be identified and constructed on the fly during forest conversion, we used *cube pruning* (Chiang, 2007; Huang and Chiang, 2007) to achieve a balance between translation quality and decoding efficiency.

category	description	number
VP	verb phrase	12.40
NP	noun phrase	7.69
IP	simple clause	7.26
QP	quantifier phrase	0.14
CP	clause headed by C	0.10
PP	preposition phrase	0.09
CLP	classifier phrase	0.02
ADJP	adjective phrase	0.02
LCP	phrase formed by “XP+LC”	0.02
DNP	phrase formed by “XP+DEG”	0.01

Table 2: Top-10 phrase categories of foot nodes and their average occurrences in training corpus.

5 Evaluation

We evaluated our adjoining tree-to-string translation system on Chinese-English translation. The bilingual corpus consists of 1.5M sentences with 42.1M Chinese words and 48.3M English words. The Chinese sentences in the bilingual corpus were parsed by an in-house parser. To maintain a reasonable grammar size, we follow Liu et al. (2006) to restrict that the height of a rule tree is no greater than 3 and the surface string's length is no greater than 7. After running GIZA++ (Och and Ney, 2003) to obtain word alignment, our rule extraction algorithm extracted 23.0M initial rules without adjoining sites, 6.6M initial rules with adjoining sites, and 5.3M auxiliary rules. We used the SRILM toolkit (Stolcke, 2002) to train a 4-gram language model on the Xinhua portion of the GIGAWORD corpus, which contains 238M English words. We used the 2002 NIST MT Chinese-English test set as the development set and the 2003-2005 NIST test sets as the test sets. We evaluated translation quality using the BLEU metric, as calculated by mteval-v11b.pl with *case-insensitive* matching of n-grams.

Table 2 shows top-10 phrase categories of foot nodes and their average occurrences in training corpus. We find that VP (verb phrase) is most likely to be the label of a foot node in an auxiliary rule. On average, there are 12.4 nodes labeled with VP are identical to one of its ancestors per tree. NP and IP are also found to be foot node labels frequently. Figure 4 shows the average occurrences of foot node labels VP, NP, and IP over various distances. A distance is the difference of levels between a foot node

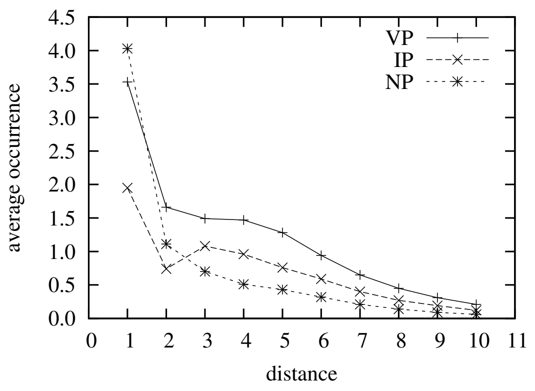


Figure 5: Average occurrences of foot node labels VP, NP, and IP over various distances.

system	grammar	MT03	MT04	MT05
Moses	-	33.10	33.96	32.17
hierarchical	SCFG	33.40	34.65	32.88
tree-to-string	STSG	33.13	34.55	31.94
	STAG	33.64	**35.28**	**32.71**

Table 3: BLEU scores on NIST Chinese-English test sets. Scores marked in bold are significantly better that those of STSG at $p \leq 0.01$ level.

and the root node. For example, in Figure 2, the distance between $\mathrm{NP}_{0,1}$ and $\mathrm{NP}_{0,3}$ is 2 and the distance between $\mathrm{VP}_{6,8}$ and $\mathrm{VP}_{3,8}$ is 1. As most foot nodes are usually very close to the root nodes, we restrict that a foot node must be the direct descendant of the root node in our experiments.

Table 3 shows the BLEU scores on the NIST Chinese-English test sets. Our baseline system is the tree-to-string system using STSG (Liu et al., 2006; Huang et al., 2006). The STAG system outperforms the STSG system significantly on the MT04 and MT05 test sets at $p \leq 0.01$ level. Table 3 also gives the results of Moses (Koehn et al., 2007) and an in-house hierarchical phrase-based system (Chiang, 2007). Our STAG system achieves comparable performance with the hierarchical system. The absolute improvement of +0.7 BLEU over STSG is close to the finding of DeNeefe and Knight (2009) on string-to-tree translation. We feel that one major obstacle for achieving further improvement is that composed rules generated on the fly during decoding (e.g., $r_1 + r_3 + r_5$ in Figure 4) usually have too many non-terminals, making cube pruning in the in-

	STSG	STAG
matching	0.086	0.109
conversion	0.000	0.562
intersection	0.946	1.064
other	0.012	0.028
total	1.044	1.763

Table 4: Comparison of average decoding time.

tersection phase suffering from severe search errors (only a tiny fraction of the search space can be explored). To produce the 1-best translations on the MT05 test set that contains 1,082 sentences, while the STSG system used 40,169 initial rules without adjoining sites, the STAG system used 28,046 initial rules without adjoining sites, 1,057 initial rules with adjoining sites, and 1,527 auxiliary rules.

Table 4 shows the average decoding time on the MT05 test set. While rule matching for STSG needs 0.086 second per sentence, the matching time for STAG only increases to 0.109 second. For STAG, the conversion of derivation forests to translation forests takes 0.562 second when we restrict that at most 200 rules can be generated on the fly for each node. As we use cube pruning, although the translation forest of STAG is bigger than that of STSG, the intersection time barely increases. In total, the STAG system runs in 1.763 seconds per sentence, only 1.6 times slower than the baseline system.

6 Conclusion

We have presented a new tree-to-string translation system based on synchronous TAG. With translation rules learned from Treebank-style trees, the adjoining tree-to-string system outperforms the baseline system using STSG without significant loss in efficiency. We plan to introduce left-to-right target generation (Huang and Mi, 2010) into the STAG tree-to-string system. Our work can also be extended to forest-based rule extraction and decoding (Mi et al., 2008; Mi and Huang, 2008). It is also interesting to introduce STAG into tree-to-tree translation (Zhang et al., 2008; Liu et al., 2009; Chiang, 2010).

Acknowledgements

The authors were supported by National Natural Science Foundation of China Contracts 60736014, 60873167, and 60903138. We thank the anonymous reviewers for their insightful comments.

References

Anne Abeille, Yves Schabes, and Aravind Joshi. 1990. Using lexicalized tags for machine translation. In *Proc. of COLING 1990*.

John Chen and K. Vijay-Shanker. 2000. Automated extraction of tags from the penn treebank. In *Proc. of IWPT 2000*.

David Chiang. 2003. Statistical parsing with an automatically extracted tree adjoining grammar. *Data-Oriented Parsing*.

David Chiang. 2006. An introduction to synchronous grammars. ACL Tutorial.

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

David Chiang. 2010. Learning to translate with source and target syntax. In *Proc. of ACL 2010*.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4).

Steve DeNeefe and Kevin Knight. 2009. Synchronous tree adjoining machine translation. In *Proc. of EMNLP 2009*.

Mark Dras. 1999. A meta-level grammar: Redefining synchronous tag for translation and paraphrase. In *Proc. of ACL 1999*.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. of ACL 2003*.

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

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

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL 2007*.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proc. of EMNLP 2010*.

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

Aravind Joshi, L. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1).

Aravind Joshi. 1985. How much contextsensitivity is necessary for characterizing structural descriptions—tree adjoining grammars. *Natural Language*

Processing—Theoretical, Computational, and Psychological Perspectives.

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 ACL 2007 (poster)*, pages 77–80, Prague, Czech Republic, June.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL 2006*.

Yang Liu, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proc. of ACL 2009*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP 2008*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL/HLT 2008*, pages 192–199, Columbus, Ohio, USA, June.

Rebecca Nesson, Stuart Shieber, and Alexander Rush. 2006. Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *Proc. of AMTA 2006*.

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

Franz Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL 2003*.

Gilles Prigent. 1994. Synchronous tags and machine translation. In *Proc. of TAG+3*.

Yves Schabes and Richard Waters. 1995. A cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4).

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proc. of COLING 1990*.

Stuart M. Shieber. 2007. Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In *Proc. of SSST 2007*.

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

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proc. of the Fifth Natural Language Processing Pacific Rim Symposium*.

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

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proc. of ACL 2008*.