Forest-to-String Statistical Translation Rules

Yang Liu , Yun Huang , Qun Liu and **Shouxun Lin**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100080, China
{yliu,huangyun,liuqun,sxlin}@ict.ac.cn

Abstract

In this paper, we propose forest-to-string rules to enhance the expressive power of tree-to-string translation models. A forest-to-string rule is capable of capturing non-syntactic phrase pairs by describing the correspondence between multiple parse trees and one string. To integrate these rules into tree-to-string translation models, auxiliary rules are introduced to provide a generalization level. Experimental results show that, on the NIST 2005 Chinese-English test set, the tree-to-string model augmented with forest-to-string rules achieves a relative improvement of 4.3% in terms of BLEU score over the original model which allows tree-to-string rules only.

1 Introduction

The past two years have witnessed the rapid development of linguistically syntax-based translation models (Quirk et al., 2005; Galley et al., 2006; Marcu et al., 2006; Liu et al., 2006), which induce tree-to-string translation rules from parallel texts with linguistic annotations. They demonstrated very promising results when compared with the state of the art phrase-based system (Och and Ney, 2004) in the NIST 2006 machine translation evaluation [1]. While Galley et al. (2006) and Marcu et al. (2006) put emphasis on target language analysis, Quirk et al. (2005) and Liu et al. (2006) show benefits from modeling the syntax of source language.

[1] See http://www.nist.gov/speech/tests/mt/

One major problem with linguistically syntax-based models, however, is that tree-to-string rules fail to syntactify non-syntactic phrase pairs because they require a syntax tree fragment over the phrase to be syntactified. Here, we distinguish between *syntactic* and *non-syntactic* phrase pairs. By "syntactic" we mean that the phrase pair is subsumed by some syntax tree fragment. The phrase pairs without trees over them are non-syntactic. Marcu et al. (2006) report that approximately 28% of bilingual phrases are non-syntactic on their English-Chinese corpus.

We believe that it is important to make available to syntax-based models all the bilingual phrases that are typically available to phrase-based models. On one hand, phrases have been proven to be a simple and powerful mechanism for machine translation. They excel at capturing translations of short idioms, providing local re-ordering decisions, and incorporating context information straightforwardly. Chiang (2005) shows significant improvement by keeping the strengths of phrases while incorporating syntax into statistical translation. On the other hand, the performance of linguistically syntax-based models can be hindered by making use of only syntactic phrase pairs. Studies reveal that linguistically syntax-based models are sensitive to syntactic analysis (Quirk and Corston-Oliver, 2006), which is still not reliable enough to handle real-world texts due to limited size and domain of training data.

Various solutions are proposed to tackle the problem. Galley et al. (2004) handle non-constituent phrasal translation by traversing the tree upwards until reaches a node that subsumes the phrase. Marcu et al. (2006) argue that this choice is inappropriate because large applicability contexts are required.

For a non-syntactic phrase pair, Marcu et al. (2006) create a xRS rule headed by a pseudo, non-syntactic nonterminal symbol that subsumes the phrase and corresponding multi-headed syntactic structure; and one sibling xRS rule that explains how the non-syntactic nonterminal symbol can be combined with other genuine nonterminals so as to obtain genuine parse trees. The name of the pseudo nonterminal is designed to reflect how the corresponding rule can be fully realized. However, they neglect alignment consistency when creating sibling rules. In addition, it is hard for the naming mechanism to deal with more complex phenomena.

Liu et al. (2006) treat bilingual phrases as lexicalized TATs (Tree-to-string Alignment Template). A bilingual phrase can be used in decoding if the source phrase is subsumed by the input parse tree. Although this solution does help, only syntactic bilingual phrases are available to the TAT-based model. Moreover, it is problematic to combine the translation probabilities of bilingual phrases and TATs, which are estimated independently.

In this paper, we propose forest-to-string rules which describe the correspondence between multiple parse trees and a string. They can not only capture non-syntactic phrase pairs but also have the capability of generalization. To integrate these rules into tree-to-string translation models, auxiliary rules are introduced to provide a generalization level. As there is no pseudo node or naming mechanism, the integration of forest-to-string rules is flexible, relying only on their root nodes. The forest-to-string and auxiliary rules enable tree-to-string models to derive in a more general way, while the strengths of conventional tree-to-string rules still remain.

2 Forest-to-String Translation Rules

We define a tree-to-string rule r as a triple $\langle \tilde{T}, \tilde{S}, \tilde{A} \rangle$, which describes the alignment $\tilde{A}$ between a source parse tree $\tilde{T} = T(f_1^{J'})$ and a target string $\tilde{S} = e_1^{I'}$. A source string $f_1^{J'}$, which is the sequence of leaf nodes of $T(f_1^{J'})$, consists of both terminals (source words) and nonterminals (phrasal categories). A target string $e_1^{I'}$ is also composed of both terminals (target words) and nonterminals (placeholders). An alignment $\tilde{A}$ is defined as a subset of the Cartesian product of source and target symbol positions:

$$\tilde{A} \subseteq \{(j, i) : j = 1, \ldots, J'; i = 1, \ldots, I'\}$$

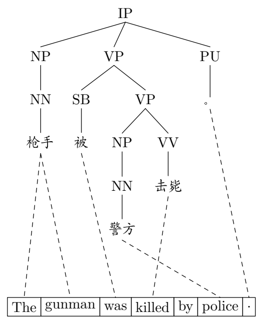

Figure 1: An English sentence aligned with a Chinese parse tree.

A derivation $\theta = r_1 \circ r_2 \circ \ldots \circ r_n$ is a left-most composition of translation rules that explains how a source parse tree $T = T(f_1^J)$, a target sentence $S = e_1^I$, and the word alignment A are synchronously generated. For example, Table 1 demonstrates a derivation composed of only tree-to-string rules for the $\langle T, S, A \rangle$ tuple in Figure 1[2].

As we mentioned before, tree-to-string rules can not syntactify phrase pairs that are not subsumed by any syntax tree fragments. For example, for the phrase pair ⟨"枪手 被", "*The gunman was*"⟩ in Figure 1, it is impossible to extract an equivalent tree-to-string rule that subsumes the same phrase pair because valid tree-to-string rules can not be multi-headed.

To address this problem, we propose *forest-to-string rules*[3] to subsume the non-syntactic phrase pairs. A forest-to-string rule r[4] is a triple $\langle \tilde{F}, \tilde{S}, \tilde{A} \rangle$, which describes the alignment $\tilde{A}$ between K source parse trees $\tilde{F} = \tilde{T}_1^K$ and a target string $\tilde{S}$. The source string $f_1^{J'}$ is therefore the sequence of leaf nodes of $\tilde{F}$.

Auxiliary rules are introduced to integrate forest-to-string rules into tree-to-string translation models. An auxiliary rule is a special unlexicalized tree-to-string rule that allows multiple source nonterminals

[2]We use "X" to denote a nonterminal in the target string. If there are more than one nonterminals, they are indexed.

[3]The term "forest" refers to an ordered and finite set of trees.

[4]We still use "r" to represent a forest-to-string rule to reduce notational overhead.

No.	Rule		
(1)	(IP (NP) (VP) (PU))	X_1 X_2 X_3	1:1 2:2 3:3
(2)	(NP (NN 枪手))	The gunman	1:1 1:2
(3)	(VP (SB 被) (VP (NP (NN)) (VV 击毙)))	was killed by X	1:1 2:4 3:2
(4)	(NN 警方)	police	1:1
(5)	(PU 。)	.	1:1

Table 1: A derivation composed of only tree-to-string rules for Figure 1.

No.	Rule		
(1)	(IP (NP) (VP (SB) (VP)) (PU))	X_1 X_2	1:1 2:1 3:2 4:2
(2)	(NP (NN 枪手)) (SB 被)	The gunman was	1:1 1:2 2:3
(3)	(VP (NP) (VV 击毙)) (PU 。)	killed by X .	1:3 2:1 3:4
(4)	(NP (NN 警方))	police	1:1

Table 2: A derivation composed of tree-to-string, forest-to-string, and auxiliary rules for Figure 1.

to correspond to one target nonterminal, suggesting that the forest-to-string rules that are rooted at such source nonterminals can be integrated.

For example, Table 2 shows a derivation composed of tree-to-string, forest-to-string, and auxiliary rules for the $\langle T, S, A \rangle$ tuple in Figure 1. r_1 is an auxiliary rule, r_2 and r_3 are forest-to-string rules, and r_4 is a conventional tree-to-string rule.

Following Marcu et al. (2006), we define the probability of a tuple $\langle T, S, A \rangle$ as the sum over all derivations $\theta_i \in \Theta$ that are consistent with the tuple, $c(\Theta) = \langle T, S, A \rangle$. The probability of each derivation θ_i is given by the product of the probabilities of all the rules $p(r_j)$ in the derivation.

$$Pr(T, S, A) = \sum_{\theta_i \in \Theta, c(\Theta) = \langle T, S, A \rangle} \prod_{r_j \in \theta_i} p(r_j) \quad (1)$$

3 Training

We obtain tree-to-string and forest-to-string rules from word-aligned, source side parsed bilingual corpus. The extraction algorithm is shown in Figure 2. Note that T' denotes either a tree or a forest.

For each span, the ⟨tree/forest, string, alignment⟩ triples are identified first. If a triple is consistent with the alignment, the *skeleton* of the triple is computed then. A skeleton s is a rule satisfying the following:

1. $s \in \mathcal{R}(t)$, s is induced from t.

2. $node(T(s)) \geq 2$, the tree/forest of s contains two or more nodes.

3. $\forall r \in \mathcal{R}(t) \wedge node(T(r)) \geq 2, T(s) \subseteq T(r)$, the tree/forest of s is the subgraph of that of any r containing two or more nodes.

```
1:  Input: a source tree $T = T(f_1^J)$, a target string
    $S = e_1^I$, and word alignment $A$ between them
2:  $\mathcal{R} := \emptyset$
3:  for $u := 0$ to $J - 1$ do
4:    for $v := 1$ to $J - u$ do
5:      identify the triple set $\mathcal{T}$ corresponding to
        span $(v, v + u)$
6:      for each triple $t = \langle T', S', A' \rangle \in \mathcal{T}$ do
7:        if $\langle T', S' \rangle$ is not consistent with $A$ then
8:          continue
9:        end if
10:       if $u = 0 \wedge node(T') = 1$ then
11:         add $t$ to $\mathcal{R}$
12:         add $\langle root(T'), “X”, 1{:}1 \rangle$ to $\mathcal{R}$
13:       else
14:         compute the skeleton $s$ of the triple $t$
15:         register rules that are built on $s$ using rules
            extracted from the sub-triples of $t$:
            $\mathcal{R} := \mathcal{R} \cup build(s, \mathcal{R})$
16:       end if
17:     end for
18:   end for
19: end for
20: Output: rule set $\mathcal{R}$
```

Figure 2: Rule extraction algorithm.

Given the skeleton and rules extracted from the sub-triples, the rules for the triple can be acquired.

For example, the algorithm identifies the following triple for span $(1, 2)$ in Figure 1:

⟨(NP (NN 枪手)) (SB 被),“The gunman was”, 1:1 1:2 2:3⟩

The skeleton of the triple is:

⟨(NP) (SB),“X_1 X_2”, 1:1 2:2⟩

As the algorithm proceeds bottom-up, five rules have already been extracted from the sub-triples, rooted at “NP” and “SB” respectively:

⟨(NP),“X”, 1:1⟩
⟨(NP (NN)),“X”, 1:1⟩
⟨(NP (NN 枪手)),“The gunman”, 1:1 1:2⟩

$\langle$(SB),“X”, 1:1$\rangle$
$\langle$(SB 被),“was”, 1:1$\rangle$

Hence, we can obtain new rules by replacing the source and target symbols of the skeleton with corresponding rules and also by modifying the alignment information. For the above triple, the combination of the five rules produces $2 \times 3 = 6$ new rules:

$\langle$(NP) (SB),“X_1 X_2”, 1:1 2:2$\rangle$
$\langle$(NP) (SB 被),“X was”, 1:1 2:2$\rangle$
$\langle$(NP (NN)) (SB),“X_1 X_2”, 1:1 2:2$\rangle$
$\langle$(NP (NN)) (SB 被),“X was”, 1:1 2:2$\rangle$
$\langle$(NP (NN 枪手)) (SB),“The gunman X”, 1:1 1:2$\rangle$
$\langle$(NP (NN 枪手)) (SB 被),“The gunman was”, 1:1 1:2 2:3$\rangle$

Since we need only to check the alignment consistency, in principle all phrase pairs can be captured by tree-to-string and forest-to-string rules. To lower the complexity for both training and decoding, we impose four restrictions:

1. Both the first and the last symbols in the target string must be aligned to some source symbols.
2. The height of a tree or forest is no greater than h.
3. The number of direct descendants of a node is no greater than c.
4. The number of leaf nodes is no greater than l.

Although possible, it is infeasible to learn auxiliary rules from training data. To extract an auxiliary rule which integrates at least one forest-to-string rule, one need traverse the parse tree upwards until one reaches a node that subsumes the entire forest without violating the alignment consistency. This usually results in very complex auxiliary rules, especially on real-world training data, making both training and decoding very slow. As a result, we construct auxiliary rules in decoding instead.

4 Decoding

Given a source parse tree $T(f_1^J)$, our decoder finds the target yield of the single best derivation that has source yield of $T(f_1^J)$:

$$\begin{aligned}\hat{S} &= \operatorname*{argmax}_{S,A} Pr(T,S,A) \\ &= \operatorname*{argmax}_{S,A} \sum_{\theta_i \in \Theta, c(\Theta)=\langle T,S,A\rangle} \prod_{r_j \in \theta_i} p(r_j) \\ &\approx \operatorname*{argmax}_{S,A,\theta} \prod_{r_j \in \theta, c(\theta)=\langle T,S,A\rangle} p(r_j) \qquad (2)\end{aligned}$$

```
1:  Input: a source parse tree $T = T(f_1^J)$
2:  for $u := 0$ to $J-1$ do
3:    for $v := 1$ to $J-u$ do
4:      for each $T'$ spanning from $v$ to $v+u$ do
5:        if $T'$ is a tree then
6:          for each usable tree-to-string rule $r$ do
7:            for each derivation $\theta$ inferred from $r$
              and derivations in matrix do
8:              add $\theta$ to matrix$[v, v+u, root(T')]$
9:            end for
10:         end for
11:         search subcell divisions $\mathcal{D}[v, v+u]$
12:         for each subcell division $d \in \mathcal{D}[v, v+u]$ do
13:           if $d$ contains at least one forest cell then
14:             construct auxiliary rule $r_a$
15:             for each derivation $\theta$ inferred from $r_a$
                and derivations in matrix do
16:               add $\theta$ to matrix$[v, v+u, root(T')]$
17:             end for
18:           end if
19:         end for
20:       else
21:         for each usable forest-to-string rule $r$ do
22:           for each derivation $\theta$ inferred from $r$
              and derivations in matrix do
23:             add $\theta$ to matrix$[v, v+u, “”]$
24:           end for
25:         end for
26:         search subcell divisions $\mathcal{D}[v, v+u]$
27:       end if
28:     end for
29:   end for
30: end for
31: find the best derivation $\hat{\theta}$ in matrix$[1, J, root(T)]$ and
    get the best translation $\hat{S} = e(\hat{\theta})$
32: Output: a target string $\hat{S}$
```

Figure 3: Decoding algorithm.

Figure 3 demonstrates the decoding algorithm. It organizes the derivations into an array *matrix* whose cells *matrix*$[j_1, j_2, X]$ are sets of derivations. $[j_1, j_2, X]$ represents a tree/forest rooted at X spanning from j_1 to j_2. We use the empty string “” to denote the pseudo root of a forest.

Next, we will explain how to infer derivations for a tree/forest provided a usable rule. If $T(r) = T'$, there is only one derivation which contains only the rule r. This usually happens for leaf nodes. If $T(r) \subset T'$, the rule r resorts to derivations from subcells to infer new derivations. Suppose that the decoder is to translate the source tree in Figure 1 and finds a usable rule for $[1, 5, \text{“IP”}]$:

$\langle$(IP (NP) (VP) (PU)),“X_1 X_2 X_3”, 1:1 2:2 3:3$\rangle$

Subcell Division	**Auxiliary Rule**		
$[1,1][2,2][3,5]$	(IP (NP) (VP (SB) (VP)) (PU))	$X_1\ X_2\ X_3$	1:1 2:2 3:3 4:3
$[1,2][3,4][5,5]$	(IP (NP) (VP (SB) (VP)) (PU))	$X_1\ X_2\ X_3$	1:1 2:1 3:2 4:3
$[1,3][4,5]$	(IP (NP) (VP (SB) (VP (NP) (VV))) (PU))	$X_1\ X_2$	1:1 2:1 3:1 4:2 5:2
$[1,1][2,5]$	(IP (NP) (VP) (PU))	$X_1\ X_2$	1:1 2:2 3:2

Table 3: Subcell divisions and corresponding auxiliary rules for the source tree in Figure 1

Since the decoding algorithm proceeds in a bottom-up fashion, the uncovered portions have already been translated.

For $[1, 1, $"NP"$]$, suppose that we can find a derivation in *matrix*:

$\langle$(NP (NN 枪手)),"The gunman", 1:1 1:2$\rangle$

For $[2, 4, $"VP"$]$, we find a derivation in *matrix*:

$\langle$(VP (SB 被) (VP (NP (NN)) (VV 击毙))),
"was killed by X", 1:1 2:4 3:2$\rangle$
$\langle$(NN 警察),"police", 1:1$\rangle$

For $[5, 5, $"PU"$]$, we find a derivation in *matrix*:

$\langle$(PU 。),".", 1:1$\rangle$

Henceforth, we get a derivation for $[1, 5, $"IP"$]$, shown in Table 1.

A translation rule r is said to be *usable* to an input tree/forest T' if and only if:

1. $T(r) \subseteq T'$, the tree/forest of r is the subgraph of T'.

2. $root(T(r)) = root(T')$, the root sequence of $T(r)$ is identical to that of T'.

For example, the following rules are usable to the tree "(NP (NR 中国) (NN 经济))":

$\langle$(NP (NR) (NN)),"$X_1\ X_2$", 1:2 2:1$\rangle$
$\langle$(NP (NR 中国) (NN)),"China X", 1:1 2:2$\rangle$
$\langle$(NP (NR 中国) (NN 经济)),"China economy", 1:1 2:2$\rangle$

Similarly, the forest-to-string rule

$\langle$((NP (NR) (NN)) (VP)),"$X_1\ X_2\ X_3$", 1:2 2:1 3:3$\rangle$

is usable to the forest

(NP (NR 布什) (NN 总统)) (VP (VV 发表)(NN 演讲))

As we mentioned before, auxiliary rules are special unlexicalized tree-to-string rules that are built in decoding rather than learnt from real-world data. To get an auxiliary rule for a cell, we need first identify its *subcell division*.

A cell sequence $c_1, c_2, \ldots, c_n$ is referred to as a subcell division of a cell c if and only if:

1. $c_1.begin = c.begin$

```
Input: a cell $[j_1, j_2]$, the derivation array matrix,
  the subcell division array $\mathcal{D}$
if $j_1 = j_2$ then
    $\hat{p} := 0$
    for each derivation $\theta$ in matrix$[j_1, j_2, \cdot]$ do
        $\hat{p} := max(p(\theta), \hat{p})$
    end for
    add $\{[j_1, j_2]\} : \hat{p}$ to $\mathcal{D}[j_1, j_2]$
else
    if $[j_1, j_2]$ is a forest cell then
        $\hat{p} := 0$
        for each derivation $\theta$ in matrix$[j_1, j_2, \cdot]$ do
            $\hat{p} := max(p(\theta), \hat{p})$
        end for
        add $\{[j_1, j_2]\} : \hat{p}$ to $\mathcal{D}[j_1, j_2]$
    end if
    for $j := j_1$ to $j_2 - 1$ do
        for each division $d_1 \in \mathcal{D}[j_1, j]$ do
            for each division $d_2 \in \mathcal{D}[j + 1, j_2]$ do
                create a new division: $d := d_1 \oplus d_2$
                add $d$ to $\mathcal{D}[j_1, j_2]$
            end for
        end for
    end for
end if
Output: subcell divisions $\mathcal{D}[j_1, j_2]$
```

Figure 4: Subcell division search algorithm.

2. $c_n.end = c.end$

3. $c_j.end + 1 = c_{j+1}.begin, 1 \leq j < n$

Given a subcell division, it is easy to construct the auxiliary rule for a cell. For each subcell, one need transverse the parse tree upwards until one reaches nodes that subsume it. All descendants of these nodes are dropped. The target string consists of only nonterminals, the number of which is identical to that of subcells. To limit the search space, we assume that the alignment between the source tree and the target string is monotone.

Table 3 shows some subcell divisions and corresponding auxiliary rules constructed for the source tree in Figure 1. For simplicity, we ignore the root node label.

There are 2^{n-1} subcell divisions for a cell which has a length of n. We need only consider the subcell divisions which contain at least one forest cell because tree-to-string rules have already explored those contain only tree cells.

The actual search algorithm for subcell divisions is shown in Figure 4. We use *matrix*$[j_1, j_2, \cdot]$ to denote all trees or forests spanning from j_1 to j_2. The subcell divisions and their associated probabilities are stored in an array $\mathcal{D}$. We define an operator $\oplus$ between two divisions: their cell sequences are concatenated and the probabilities are accumulated.

As sometimes there are no usable rules available, we introduce *default* rules to ensure that we can always get a translation for any input parse tree. A default rule is a tree-to-string rule[5], built in two ways:

1. If the input tree contains only one node, the target string of the default rule is equal to the source string.

2. If the height of the input tree is greater than one, the tree of the default rule contains only the root node and its direct descendants of the input tree, the string contains only nonterminals, and the alignment is monotone.

To speed up the decoder, we limit the search space by reducing the number of rules used for each cell. There are two ways to limit the rule table size: by a fixed limit a of how many rules are retrieved for each cell, and by a probability threshold α that specify that the rule probability has to be above some value. Also, instead of keeping the full list of derivations for a cell, we store a top-scoring subset of the derivations. This can also be done by a fixed limit b or a threshold β. The subcell division array $\mathcal{D}$, in which divisions containing forest cells have priority over those composed of only tree cells, is pruned by keeping only a-best divisions.

Following Och and Ney (2002), we base our model on log-linear framework and adopt the seven feature functions described in (Liu et al., 2006). It is very important to balance the preference between conventional tree-to-string rules and the newly-introduced forest-to-string and auxiliary rules. As the probabilities of auxiliary rules are not learnt from training data, we add a feature that sums up the node count of auxiliary rules of a derivation to penalize the use of forest-to-string and auxiliary rules.

[5]There are no default rules for forests because only tree-to-string rules are essential to tree-to-string translation models.

5 Experiments

In this section, we report on experiments with Chinese-to-English translation. The training corpus consists of $31,149$ sentence pairs with $843,256$ Chinese words and $949,583$ English words. For the language model, we used SRI Language Modeling Toolkit (Stolcke, 2002) to train a trigram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998) on the $31,149$ English sentences. We selected 571 short sentences from the 2002 NIST MT Evaluation test set as our development corpus, and used the 2005 NIST MT Evaluation test set as our test corpus. Our evaluation metric is BLEU-4 (Papineni et al., 2002), as calculated by the script mteval-v11b.pl with its default setting except that we used case-sensitive matching of n-grams. To perform minimum error rate training (Och, 2003) to tune the feature weights to maximize the system's BLEU score on development set, we used the script optimizeV5IBMBLEU.m (Venugopal and Vogel, 2005).

We ran GIZA++ (Och and Ney, 2000) on the training corpus in both directions using its default setting, and then applied the refinement rule "diag-and" described in (Koehn et al., 2003) to obtain a single many-to-many word alignment for each sentence pair. Next, we employed a Chinese parser written by Deyi Xiong (Xiong et al., 2005) to parse all the $31,149$ Chinese sentences. The parser was trained on articles 1-270 of Penn Chinese Treebank version 1.0 and achieved 79.4% in terms of F1 measure.

Given the word-aligned, source side parsed bilingual corpus, we obtained bilingual phrases using the training toolkits publicly released by Philipp Koehn with its default setting. Then, we applied extraction algorithm described in Figure 2 to extract both tree-to-string and forest-to-string rules by restricting $h = 3$, $c = 5$, and $l = 7$. All the rules, including bilingual phrases, tree-to-string rules, and forest-to-string rules, are filtered for the development and test sets.

According to different levels of lexicalization, we divide translation rules into three categories:

Rule	L	P	U	Total
BP	$251,173$	0	0	$251,173$
TR	$56,983$	$41,027$	$3,529$	$101,539$
FR	$16,609$	$254,346$	$25,051$	$296,006$

Table 4: Number of rules used in experiments (BP: bilingual phrase, TR: tree-to-string rule, FR: forest-to-string rule; L: lexicalized, P: partial lexicalized, U: unlexicalized).

System	Rule Set	BLEU4
Pharaoh	BP	0.2182 ± 0.0089
Lynx	BP	0.2059 ± 0.0083
	TR	0.2302 ± 0.0089
	TR + BP	0.2346 ± 0.0088
	TR + FR + AR	0.2402 ± 0.0087

Table 5: Comparison of Pharaoh and Lynx with different rule sets.

1. *lexicalized*: all symbols in both the source and target strings are terminals
2. *unlexicalized*: all symbols in both the source and target strings are nonterminals
3. *partial lexicalized*: otherwise

Table 4 shows the statistics of rules used in our experiments. We find that even though forest-to-string rules are introduced the total number (i.e. $73,592$) of lexicalized tree-to-string and forest-to-string rules is still far less than that (i.e. $251,173$) of bilingual phrases. This difference results from the restriction we impose in training that both the first and last symbols in the target string must be aligned to some source symbols. For the forest-to-string rules, partial lexicalized ones are in the majority.

We compared our system Lynx against a freely available phrase-based decoder Pharaoh (Koehn et al., 2003). For Pharaoh, we set $a = 20$, $\alpha = 0$, $b = 100$, $\beta = 10^{-5}$, and distortion limit $dl = 4$. For Lynx, we set $a = 20$, $\alpha = 0$, $b = 100$, and $\beta = 0$. Two postprocessing procedures ran to improve the outputs of both systems: OOVs removal and recapitalization.

Table 5 shows results on test set using Pharaoh and Lynx with different rule sets. Note that Lynx is capable of using only bilingual phrases plus default rules to perform monotone search. The 95% confidence intervals were computed using Zhang's significance tester (Zhang et al., 2004). We modified it to conform to NIST's current definition of the BLEU brevity penalty. We find that Lynx outperforms Pharaoh significantly. The integration of forest-to-string rules achieves an absolute improvement of 1.0% (4.3% relative) over using tree-to-string rules only. This difference is statistically significant ($p < 0.01$). It also achieves better result than treating bilingual phrases as lexicalized tree-to-string rules. To produce the best result of 0.2402, Lynx made use of $26,082$ tree-to-string rules, $9,219$ default rules, $5,432$ forest-to-string rules, and $2,919$ auxiliary rules. This suggests that tree-to-string rules still play a central role, although the integration of forest-to-string and auxiliary rules is really beneficial.

Forest-to-String Rule Set	BLEU4
None	0.2225 ± 0.0085
L	0.2297 ± 0.0081
P	0.2279 ± 0.0083
U	0.2270 ± 0.0087
L + P + U	0.2312 ± 0.0082

Table 6: Effect of lexicalized, partial lexicalized, and unlexicalized forest-to-string rules.

Table 6 demonstrates the effect of forest-to-string rules with different lexicalization levels. We set $a = 3$, $\alpha = 0$, $b = 10$, and $\beta = 0$. The second row "None" shows the result of using only tree-to-string rules. "L" denotes using tree-to-string rules and lexicalized forest-to-string rules. Similarly, "L+P+U" denotes using tree-to-string rules and all forest-to-string rules. We find that lexicalized forest-to-string rules are more useful.

6 Conclusion

In this paper, we introduce forest-to-string rules to capture non-syntactic phrase pairs that are usually unaccessible to traditional tree-to-string translation models. With the help of auxiliary rules, forest-to-string rules can be integrated into tree-to-string models to offer more general derivations. Experiment results show that the tree-to-string model augmented with forest-to-string rules significantly outperforms the original model which allows tree-to-string rules only.

Our current rule extraction algorithm attaches the unaligned target words to the nearest ascendants that subsume them. This constraint hampers the expressive power of our model. We will try a more general way as suggested in (Galley et al., 2006), making no a priori assumption about assignment and using EM training to learn the probability distribution. We will also conduct experiments on large scale training data to further examine our design philosophy.

Acknowledgement

This work was supported by National Natural Science Foundation of China, Contract No. 60603095 and 60573188.

References

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Harvard University Center for Research in Computing Technology.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL 2005*, pages 263–270, Ann Arbor, Michigan, June.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL 2004*, pages 273–280, Boston, Massachusetts, USA, May.

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 *Proceedings of COLING/ACL 2006*, pages 961–968, Sydney, Australia, July.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL 2003*, pages 127–133, Edmonton, Canada, May.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING/ACL 2006*, pages 609–616, Sydney, Australia, July.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. Spmt: Statistical machine translation with syntactified target language phrases. In *Proceedings of EMNLP 2006*, pages 44–52, Sydney, Australia, July.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL 2000*, pages 440–447.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL 2002*, pages 295–302.

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

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

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

Chris Quirk and Simon Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *Proceedings of EMNLP 2006*, pages 62–69, Sydney, Australia, July.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of ACL 2005*, pages 271–279, Ann Arbor, Michigan, June.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, volume 30, pages 901–904.

Ashish Venugopal and Stephan Vogel. 2005. Considerations in maximum mutual information and minimum classification error training for statistical machine translation. In *Proceedings of the Tenth Conference of the European Association for Machine Translation*, pages 271–279.

Deyi Xiong, Shuanglong Li, Qun Liu, and Shouxun Lin. 2005. Parsing the penn chinese treebank with semantic knowledge. In *Proceedings of IJCNLP 2005*, pages 70–81.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting bleu/nist scores how much improvement do we need to have a better system? In *Proceedings of Fourth International Conference on Language Resources and Evaluation*, pages 2051–2054.