Improving Tree-to-Tree Translation with Packed Forests

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

Abstract

Current tree-to-tree models suffer from parsing errors as they usually use only 1-best parses for rule extraction and decoding. We instead propose a forest-based tree-to-tree model that uses packed forests. The model is based on a probabilistic synchronous tree substitution grammar (STSG), which can be learned from aligned forest pairs automatically. The decoder finds ways of decomposing trees in the source forest into elementary trees using the source projection of STSG while building target forest in parallel. Comparable to the state-of-the-art phrase-based system Moses, using packed forests in tree-to-tree translation results in a significant absolute improvement of 3.6 BLEU points over using 1-best trees.

1 Introduction

Approaches to syntax-based statistical machine translation make use of parallel data with syntactic annotations, either in the form of phrase structure trees or dependency trees. They can be roughly divided into three categories: *string-to-tree* models (e.g., (Galley et al., 2006; Marcu et al., 2006; Shen et al., 2008)), *tree-to-string* models (e.g., (Liu et al., 2006; Huang et al., 2006)), and *tree-to-tree* models (e.g., (Eisner, 2003; Ding and Palmer, 2005; Cowan et al., 2006; Zhang et al., 2008)). By modeling the syntax of both source and target languages, tree-to-tree approaches have the potential benefit of providing rules linguistically better motivated. However, while string-to-tree and tree-to-string models demonstrate promising results in empirical evaluations, tree-to-tree models have still been underachieving.

We believe that tree-to-tree models face two major challenges. First, tree-to-tree models are more vulnerable to parsing errors. Obtaining syntactic annotations in quantity usually entails running automatic parsers on a parallel corpus. As the amount and domain of the data used to train parsers are relatively limited, parsers will inevitably output ill-formed trees when handling real-world text. Guided by such noisy syntactic information, syntax-based models that rely on 1-best parses are prone to learn noisy translation rules in training phase and produce degenerate translations in decoding phase (Quirk and Corston-Oliver, 2006). This situation aggravates for tree-to-tree models that use syntax on both sides.

Second, tree-to-tree rules provide poorer rule coverage. As a tree-to-tree rule requires that there must be trees on both sides, tree-to-tree models lose a larger amount of linguistically unmotivated mappings. Studies reveal that the absence of such non-syntactic mappings will impair translation quality dramatically (Marcu et al., 2006; Liu et al., 2007; DeNeefe et al., 2007; Zhang et al., 2008).

Compactly encoding exponentially many parses, *packed forests* prove to be an excellent fit for alleviating the above two problems (Mi et al., 2008; Mi and Huang, 2008). In this paper, we propose a forest-based tree-to-tree model. To learn STSG rules from aligned forest pairs, we introduce a series of notions for identifying minimal tree-to-tree rules. Our decoder first converts the source forest to a translation forest and then finds the best derivation that has the source yield of one source tree in the forest. Comparable to Moses, our forest-based tree-to-tree model achieves an absolute improvement of 3.6 BLEU points over conventional tree-based model.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 558–566,
Suntec, Singapore, 2-7 August 2009.

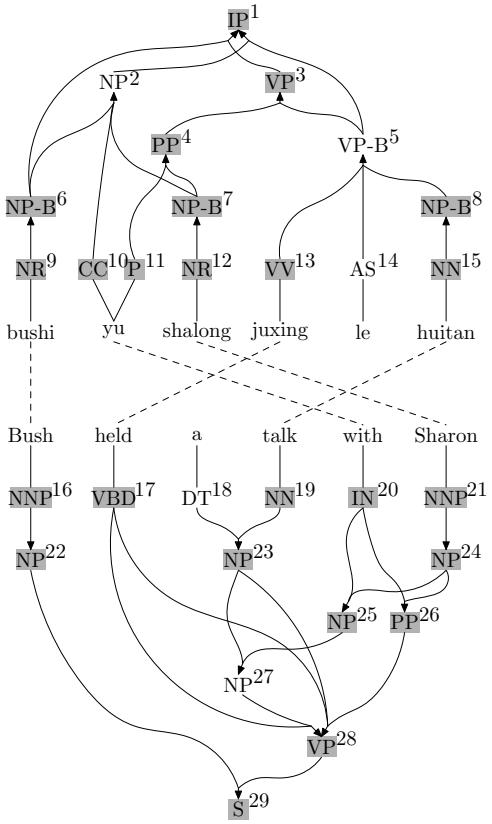

Figure 1: An aligned packed forest pair. Each node is assigned a unique identity for reference. The solid lines denote hyperedges and the dashed lines denote word alignments. Shaded nodes are frontier nodes.

2 Model

Figure 1 shows an aligned forest pair for a Chinese sentence and an English sentence. The solid lines denote *hyperedges* and the dashed lines denote word alignments between the two forests. Each node is assigned a unique identity for reference. Each hyperedge is associated with a probability, which we omit in Figure 1 for clarity. In a forest, a *node* usually has multiple incoming hyperedges. We use $IN(v)$ to denote the set of incoming hyperedges of node v. For example, the source node "IP[1]" has following two incoming hyperedges: [1]

$$e_1 = \langle (\text{NP-B}^6, \text{VP}^3), \text{IP}^1 \rangle$$
$$e_2 = \langle (\text{NP}^2, \text{VP-B}^5), \text{IP}^1 \rangle$$

[1]As there are both source and target forests, it might be confusing by just using a span to refer to a node. In addition, some nodes will often have the same labels and spans. Therefore, it is more convenient to use an identity for referring to a node. The notation "IP[1]" denotes the node that has a label of "IP" and has an identity of "1".

Formally, a packed parse forest is a compact representation of all the derivations (i.e., parse trees) for a given sentence under a context-free grammar. Huang and Chiang (2005) define a forest as a tuple $\langle V, E, \bar{v}, \mathbf{R} \rangle$, where V is a finite set of nodes, E is a finite set of hyperedges, $\bar{v} \in V$ is a distinguished node that denotes the goal item in parsing, and $\mathbf{R}$ is the set of weights. For a given sentence $w_{1:l} = w_1 \dots w_l$, each node $v \in V$ is in the form of $X_{i,j}$, which denotes the recognition of non-terminal X spanning the substring from positions i through j (that is, $w_{i+1} \dots w_j$). Each hyperedge $e \in E$ is a triple $e = \langle T(e), h(e), f(e) \rangle$, where $h(e) \in V$ is its head, $T(e) \in V^*$ is a vector of tail nodes, and $f(e)$ is a weight function from $\mathbf{R}^{|T(e)|}$ to $\mathbf{R}$.

Our forest-based tree-to-tree model is based on a probabilistic STSG (Eisner, 2003). Formally, an STSG can be defined as a quintuple $G = \langle \mathcal{F}_s, \mathcal{F}_t, \mathcal{S}_s, \mathcal{S}_t, P \rangle$, where

- $\mathcal{F}_s$ and $\mathcal{F}_t$ are the source and target alphabets, respectively,
- $\mathcal{S}_s$ and $\mathcal{S}_t$ are the source and target start symbols, and
- P is a set of production rules. A rule r is a triple $\langle t_s, t_t, \sim \rangle$ that describes the correspondence $\sim$ between a source tree t_s and a target tree t_t.

To integrate packed forests into tree-to-tree translation, we model the process of synchronous generation of a source forest F_s and a target forest F_t using a probabilistic STSG grammar:

$$\begin{aligned} Pr(F_s, F_t) &= \sum_{T_s \in F_s} \sum_{T_t \in F_t} Pr(T_s, T_t) \\ &= \sum_{T_s \in F_s} \sum_{T_t \in F_t} \sum_{d \in D} Pr(d) \\ &= \sum_{T_s \in F_s} \sum_{T_t \in F_t} \sum_{d \in D} \prod_{r \in d} p(r) \quad (1) \end{aligned}$$

where T_s is a source tree, T_t is a target tree, D is the set of all possible derivations that transform T_s into T_t, d is one such derivation, and r is a tree-to-tree rule.

Table 1 shows a derivation of the forest pair in Figure 1. A derivation is a sequence of tree-to-tree rules. Note that we use x to represent a nonterminal.

(1)	IP(x_1:NP-B, x_2:VP) → S(x_1:NP, x_2:VP)
(2)	NP-B(x_1:NR) → NP(x_1:NNP)
(3)	NR(bushi) → NNP(Bush)
(4)	VP(x_1:PP, VP-B(x_2:VV, AS(le), x_3:NP-B)) → VP(x_2:VBD, NP(DT(a), x_3:NP), x_1:PP)
(5)	PP(x_1:P, x_2:NP-B) → PP(x_1:IN, x_2:NP)
(6)	P(yu) → IN(with)
(7)	NP-B(x_1:NR) → NP(x_1:NP)
(8)	NR(shalong) → NNP(Sharon)
(9)	VV(juxing) → VBD(held)
(10)	NP-B(x_1:NN) → NP(x_1:NN)
(11)	NN(huitan) → NN(talk)

Table 1: A minimal derivation of the forest pair in Figure 1.

id	span	cspan	complement	consistent	frontier	counterparts
1	1-6	1-2, 4-6		1	1	29
2	1-3	1, 5-6	2, 4	0	0	
3	2-6	2, 4-6	1	1	1	28
4	2-3	5-6	1-2, 4	1	1	25, 26
5	4-6	2, 4	1, 5-6	1	0	
6	1-1	1	2, 4-6	1	1	16, 22
7	3-3	6	1-2, 4-5	1	1	21, 24
8	6-6	4	1-2, 5-6	1	1	19, 23
9	1-1	1	2, 4-6	1	1	16, 22
10	2-2	5	1-2, 4, 6	1	1	20
11	2-2	5	1-2, 4, 6	1	1	20
12	3-3	6	1-2, 4-5	1	1	21, 24
13	4-4	2	1, 4-6	1	1	17
14	5-5		1-2, 4-6	1	0	
15	6-6	4	1-2, 5-6	1	1	19, 23
16	1-1	1	2-4, 6	1	1	6, 9
17	2-2	4	1-3, 6	1	1	13
18	3-3		1-4, 6	1	0	
19	4-4	6	1-4	1	1	8, 15
20	5-5	2	1, 3-4, 6	1	1	10, 11
21	6-6	3	1-2, 4, 6	1	1	7, 12
22	1-1	1	2-4, 6	1	1	6, 9
23	3-4	6	1-4	1	1	8, 15
24	6-6	3	1-2, 4, 6	1	1	7, 12
25	5-6	2-3	1, 4, 6	1	1	4
26	5-6	2-3	1, 4, 6	1	1	4
27	3-6	2-3, 6	1, 4	0	0	
28	2-6	2-4, 6	1	1	1	3
29	1-6	1-4, 6		1	1	1

Table 2: Node attributes of the example forest pair.

3 Rule Extraction

Given an aligned forest pair as shown in Figure 1, how to extract all valid tree-to-tree rules that explain its synchronous generation process? By constructing a theory that gives formal semantics to word alignments, Galley et al. (2004) give principled answers to these questions for extracting tree-to-string rules. Their GHKM procedure draws connections among word alignments, derivations, and rules. They first identify the tree nodes that subsume tree-string pairs consistent with word alignments and then extract rules from these nodes. By this means, GHKM proves to be able to extract all valid tree-to-string rules from training instances. Although originally developed for the tree-to-string case, it is possible to extend GHKM to extract all valid tree-to-tree rules from aligned packed forests.

In this section, we introduce our tree-to-tree rule extraction method adapted from GHKM, which involves four steps: (1) identifying the correspondence between the nodes in forest pairs, (2) identifying minimum rules, (3) inferring composed rules, and (4) estimating rule probabilities.

3.1 Identifying Correspondence Between Nodes

To learn tree-to-tree rules, we need to find aligned tree pairs in the forest pairs. To do this, the starting point is to identify the correspondence between nodes. We propose a number of attributes for nodes, most of which derive from GHKM, to facilitate the identification.

Definition 1 Given a node v, its *span* $\sigma(v)$ is an index set of the words it covers.

For example, the span of the source node "VP-B^5" is $\{4, 5, 6\}$ as it covers three source words: "*juxing*", "*le*", and "*huitan*". For convenience, we use $\{4\text{-}6\}$ to denotes a contiguous span $\{4, 5, 6\}$.

Definition 2 Given a node v, its *corresponding span* $\gamma(v)$ is the index set of aligned words on another side.

For example, the corresponding span of the source node "VP-B^5" is $\{2, 4\}$, corresponding to the target words "*held*" and "*talk*".

Definition 3 Given a node v, its *complement span* $\delta(v)$ is the union of corresponding spans of nodes that are neither antecedents nor descendants of v.

For example, the complement span of the source node "VP-B^5" is $\{1, 5\text{-}6\}$, corresponding to target words "*Bush*", "*with*", and "*Sharon*".

Definition 4 A node v is said to be *consistent* with alignment if and only if $closure(\gamma(v)) \cap \delta(v) = \emptyset$.

For example, the closure of the corresponding span of the source node "VP-B^5" is $\{2\text{-}4\}$ and its complement span is $\{1, 5\text{-}6\}$. As the intersection of the closure and the complement span is an empty set, the source node "VP-B^5" is consistent with the alignment.

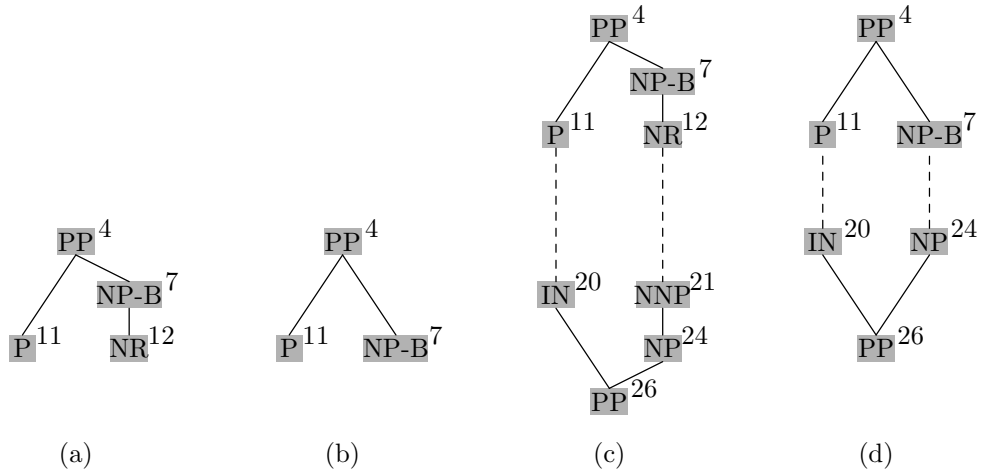

Figure 2: (a) A frontier tree; (b) a minimal frontier tree; (c) a frontier tree pair; (d) a minimal frontier tree pair. All trees are taken from the example forest pair in Figure 1. Shaded nodes are frontier nodes. Each node is assigned an identity for reference.

Definition 5 A node v is said to be a *frontier* node if and only if:

1. v is consistent;
2. There exists at least one consistent node v' on another side satisfying:
   - $closure(\gamma(v')) \subseteq \sigma(v)$;
   - $closure(\gamma(v)) \subseteq \sigma(v')$.

v' is said to be a *counterpart* of v. We use $\tau(v)$ to denote the set of counterparts of v.

A frontier node often has multiple counterparts on another side due to the usage of unary rules in parsers. For example, the source node "NP-B^6" has two counterparts on the target side: "NNP^{16}" and "NP^{22}". Conversely, the target node "NNP^{16}" also has two counterparts counterparts on the source side: "NR^9" and "NP-B^6".

The node attributes of the example forest pair are listed in Table 2. We use identities to refer to nodes. "cspan" denotes corresponding span and "complement" denotes complement span. In Figure 1, there are 12 frontier nodes (highlighted by shading) on the source side and 12 frontier nodes on the target side. Note that while a consistent node is equal to a frontier node in GHKM, this is not the case in our method because we have a tree on the target side. Frontier nodes play a critical role in forest-based rule extraction because they indicate where to cut the forest pairs to obtain tree-to-tree rules.

3.2 Identifying Minimum Rules

Given the frontier nodes, the next step is to identify aligned tree pairs, from which tree-to-tree rules derive. Following Galley et al. (2006), we distinguish between *minimal* and *composed* rules. As a composed rule can be decomposed as a sequence of minimal rules, we are particularly interested in how to extract minimal rules. Also, we introduce a number of notions to help identify minimal rules.

Definition 6 A *frontier tree* is a subtree in a forest satisfying:

1. Its root is a frontier node;
2. If the tree contains only one node, it must be a lexicalized frontier node;
3. If the tree contains more than one nodes, its leaves are either non-lexicalized frontier nodes or lexicalized non-frontier nodes.

For example, Figure 2(a) shows a frontier tree in which all nodes are frontier nodes.

Definition 7 A *minimal frontier tree* is a frontier tree such that all nodes other than the root and leaves are non-frontier nodes.

For example, Figure 2(b) shows a minimal frontier tree.

Definition 8 A *frontier tree pair* is a triple $\langle t_s, t_t, \sim \rangle$ satisfying:

1. t_s is a source frontier tree;

2. t_t is a target frontier tree;

3. The root of t_s is a counterpart of that of t_t;

4. There is a one-to-one correspondence $\sim$ between the frontier leaves of t_s and t_t.

For example, Figure 2(c) shows a frontier tree pair.

Definition 9 A frontier tree pair $\langle t_s, t_t, \sim \rangle$ is said to be a *subgraph* of another frontier tree pair $\langle {t_s}', {t_t}', \sim' \rangle$ if and only if:

1. $root(t_s) = root({t_s}')$;

2. $root(t_t) = root({t_t}')$;

3. t_s is a subgraph of ${t_s}'$;

4. t_t is a subgraph of ${t_t}'$.

For example, the frontier tree pair shown in Figure 2(d) is a subgraph of that in Figure 2(c).

Definition 10 A frontier tree pair is said to be *minimal* if and only if it is not a subgraph of any other frontier tree pair that shares with the same root.

For example, Figure 2(d) shows a minimal frontier tree pair.

Our goal is to find the minimal frontier tree pairs, which correspond to minimal tree-to-tree rules. For example, the tree pair shown in Figure 2(d) denotes a minimal rule as follows:

$$\text{PP}(x_1\text{:P}, x_2\text{:NP-B}) \rightarrow \text{PP}(x_1\text{:IN}, x_2\text{:NP})$$

Figure 3 shows the algorithm for identifying minimal frontier tree pairs. The input is a source forest F_s, a target forest F_t, and a source frontier node v (line 1). We use a set $\mathcal{P}$ to store collected minimal frontier tree pairs (line 2). We first call the procedure FINDTREES(F_s, v) to identify a set of frontier trees rooted at v in F_s (line 3). For example, for the source frontier node "PP^4" in Figure 1, we obtain two frontier trees:

$$(\text{PP}^4(\text{P}^{11})(\text{NP-B}^7))$$
$$(\text{PP}^4(\text{P}^{11})(\text{NP-B}^7(\text{NR}^{12})))$$

Then, we try to find the set of corresponding target frontier trees (i.e., $\mathcal{T}_t$). For each counterpart v' of v (line 5), we call the procedure FINDTREES(F_t, v') to identify a set of frontier trees rooted at v' in F_t (line 6). For example, the source

```
1: procedure FINDTREEPAIRS(F_s, F_t, v)
2:     P = ∅
3:     T_s ← FINDTREES(F_s, v)
4:     T_t ← ∅
5:     for v' ∈ τ(v) do
6:         T_t ← T_t ∪ FINDTREES(F_t, v')
7:     end for
8:     for ⟨t_s, t_t⟩ ∈ T_s × T_t do
9:         if t_s ∼ t_t then
10:            P ← P ∪ {⟨t_s, t_t, ∼⟩}
11:        end if
12:    end for
13:    for ⟨t_s, t_t, ∼⟩ ∈ P do
14:        if ∃⟨t_s', t_t', ∼'⟩ ∈ P : ⟨t_s', t_t', ∼'⟩ ⊆
       ⟨t_s, t_t, ∼⟩ then
15:            P ← P − {⟨t_s, t_t, ∼⟩}
16:        end if
17:    end for
18: end procedure
```

Figure 3: Algorithm for identifying minimal frontier tree pairs.

frontier node "PP^4" has two counterparts on the target side: "NP^{25}" and "PP^{26}". There are four target frontier trees rooted at the two nodes:

$$(\text{NP}^{25}(\text{IN}^{20})(\text{NP}^{24}))$$
$$(\text{NP}^{25}(\text{IN}^{20})(\text{NP}^{24}(\text{NNP}^{21})))$$
$$(\text{PP}^{26}(\text{IN}^{20})(\text{NP}^{24}))$$
$$(\text{PP}^{26}(\text{IN}^{20})(\text{NP}^{24}(\text{NNP}^{21})))$$

Therefore, there are $2 \times 4 = 8$ pairs of trees. We examine each tree pair $\langle t_s, t_t \rangle$ (line 8) to see whether it is a frontier tree pair (line 9) and then update $\mathcal{P}$ (line 10). In the above example, all the eight tree pairs are frontier tree pairs.

Finally, we keep only minimal frontier tree pairs in $\mathcal{P}$ (lines 13-15). As a result, we obtain the following two minimal frontier tree pairs for the source frontier node "PP^4":

$$(\text{PP}^4(\text{P}^{11})(\text{NP-B}^7)) \leftrightarrow (\text{NP}^{25}(\text{IN}^{20})(\text{NP}^{24}))$$
$$(\text{PP}^4(\text{P}^{11})(\text{NP-B}^7)) \leftrightarrow (\text{PP}^{26}(\text{IN}^{20})(\text{NP}^{24}))$$

To maintain a reasonable rule table size, we restrict that the number of nodes in a tree of an STSG rule is no greater than n, which we refer to as *maximal node count*.

It seems more efficient to let the procedure FINDTREES(F, v) to search for minimal frontier

trees rather than frontier trees. However, a minimal frontier tree pair is not necessarily a pair of minimal frontier trees. On our Chinese-English corpus, we find that 38% of minimal frontier tree pairs are not pairs of minimal frontier trees. As a result, we have to first collect all frontier tree pairs and then decide on the minimal ones.

Table 1 shows some minimal rules extracted from the forest pair shown in Figure 1.

3.3 Inferring Composed Rules

After minimal rules are learned, composed rules can be obtained by composing two or more minimal rules. For example, the composition of the second rule and the third rule in Table 1 produces a new rule:

$$\text{NP-B(NR(}\textit{shalong}\text{))} \rightarrow \text{NP(NNP(}\textit{Sharon}\text{))}$$

While minimal rules derive from minimal frontier tree pairs, composed rules correspond to non-minimal frontier tree pairs.

3.4 Estimating Rule Probabilities

We follow Mi and Huang (2008) to estimate the fractional count of a rule extracted from an aligned forest pair. Intuitively, the relative frequency of a subtree that occurs in a forest is the sum of all the trees that traverse the subtree divided by the sum of all trees in the forest. Instead of enumerating all trees explicitly and computing the sum of tree probabilities, we resort to inside and outside probabilities for efficient calculation:

$$c(r) = \frac{p(t_s) \times \alpha(root(t_s)) \times \prod_{v \in leaves(t_s)} \beta(v)}{\beta(\bar{v}_s)} \times \frac{p(t_t) \times \alpha(root(t_t)) \times \prod_{v \in leaves(t_t)} \beta(v)}{\beta(\bar{v}_t)}$$

where $c(r)$ is the fractional count of a rule, t_s is the source tree in r, t_t is the target tree in r, $root(\cdot)$ a function that gets tree root, $leaves(\cdot)$ is a function that gets tree leaves, and $\alpha(v)$ and $\beta(v)$ are outside and inside probabilities, respectively.

4 Decoding

Given a source packed forest F_s, our decoder finds the target yield of the single best derivation d that has source yield of $T_s(d) \in F_s$:

$$\hat{e} = e\left(\underset{d \ s.t. \ T_s(d) \in F_s}{\text{argmax}} \ p(d)\right) \quad (2)$$

We extend the model in Eq. 1 to a log-linear model (Och and Ney, 2002) that uses the following eight features: relative frequencies in two directions, lexical weights in two directions, number of rules used, language model score, number of target words produced, and the probability of matched source tree (Mi et al., 2008).

Given a source parse forest and an STSG grammar G, we first apply the conversion algorithm proposed by Mi et al. (2008) to produce a *translation forest*. The translation forest has a similar hypergraph structure. While the nodes are the same as those of the parse forest, each hyperedge is associated with an STSG rule. Then, the decoder runs on the translation forest. We use the cube pruning method (Chiang, 2007) to approximately intersect the translation forest with the language model. Traversing the translation forest in a bottom-up order, the decoder tries to build target parses at each node. After the first pass, we use lazy Algorithm 3 (Huang and Chiang, 2005) to generate k-best translations for minimum error rate training.

5 Experiments

5.1 Data Preparation

We evaluated our model on Chinese-to-English translation. The training corpus contains 840K Chinese words and 950K English words. A trigram language model was trained on the English sentences of the training corpus. We used the 2002 NIST MT Evaluation test set as our development set, and used the 2005 NIST MT Evaluation test set as our test set. We evaluated the translation quality using the BLEU metric, as calculated by mteval-v11b.pl with its default setting except that we used *case-insensitive* matching of n-grams.

To obtain packed forests, we used the Chinese parser (Xiong et al., 2005) modified by Haitao Mi and the English parser (Charniak and Johnson, 2005) modified by Liang Huang to produce entire parse forests. Then, we ran the Python scripts (Huang, 2008) provided by Liang Huang to output packed forests. To prune the packed forests, Huang (2008) uses inside and outside probabilities to compute the distance of the best derivation that traverses a hyperedge away from the globally best derivation. A hyperedge will be pruned away if the difference is greater than a threshold p. Nodes with all incoming hyperedges pruned are also pruned. The greater the threshold p is,

p	avg trees	# of rules	BLEU
0	1	73,614	0.2021 ± 0.0089
2	238.94	105,214	0.2165 ± 0.0081
5	5.78×10^6	347,526	0.2336 ± 0.0078
8	6.59×10^7	573,738	0.2373 ± 0.0082
10	1.05×10^8	743,211	0.2385 ± 0.0084

Table 3: Comparison of BLEU scores for tree-based and forest-based tree-to-tree models.

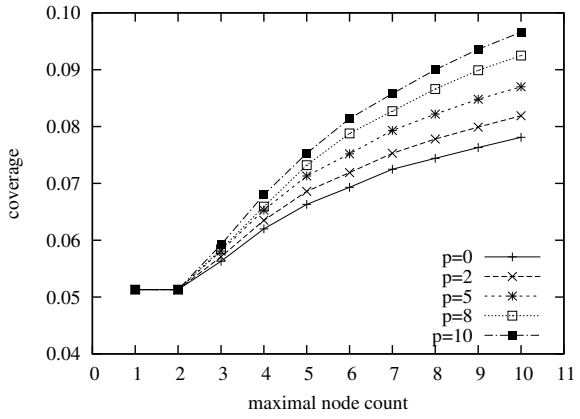

Figure 4: Coverage of lexicalized STSG rules on bilingual phrases.

the more parses are encoded in a packed forest.

We obtained word alignments of the training data by first running GIZA++ (Och and Ney, 2003) and then applying the refinement rule "grow-diag-final-and" (Koehn et al., 2003).

5.2 Forests Vs. 1-best Trees

Table 3 shows the BLEU scores of tree-based and forest-based tree-to-tree models achieved on the test set over different pruning thresholds. p is the threshold for pruning packed forests, "avg trees" is the average number of trees encoded in one forest on the test set, and "# of rules" is the number of STSG rules used on the test set. We restrict that both source and target trees in a tree-to-tree rule can contain at most 10 nodes (i.e., the maximal node count $n = 10$). The 95% confidence intervals were computed using Zhang 's significance tester (Zhang et al., 2004).

We chose five different pruning thresholds in our experiments: $p = 0, 2, 5, 8, 10$. The forests pruned by $p = 0$ contained only 1-best tree per sentence. With the increase of p, the average number of trees encoded in one forest rose dramatically. When p was set to 10, there were over 100M parses encoded in one forest on average.

p	extraction	decoding
0	1.26	6.76
2	2.35	8.52
5	6.34	14.87
8	8.51	19.78
10	10.21	25.81

Table 4: Comparison of rule extraction time (seconds/1000 sentence pairs) and decoding time (second/sentence)

Moreover, the more trees are encoded in packed forests, the more rules are made available to forest-based models. The number of rules when $p = 10$ was almost 10 times of $p = 0$. With the increase of the number of rules used, the BLEU score increased accordingly. This suggests that packed forests enable tree-to-tree model to learn more useful rules on the training data. However, when a pack forest encodes over 1M parses per sentence, the improvements are less significant, which echoes the results in (Mi et al., 2008).

The forest-based tree-to-tree model outperforms the original model that uses 1-best trees dramatically. The absolute improvement of 3.6 BLEU points (from 0.2021 to 0.2385) is statistically significant at $p < 0.01$ using the *sign-test* as described by Collins et al. (2005), with 700(+1), 360(-1), and 15(0). We also ran Moses (Koehn et al., 2007) with its default setting using the same data and obtained a BLEU score of 0.2366, slightly lower than our best result (i.e., 0.2385). But this difference is not statistically significant.

5.3 Effect on Rule Coverage

Figure 4 demonstrates the effect of pruning threshold and maximal node count on rule coverage. We extracted phrase pairs from the training data to investigate how many phrase pairs can be captured by lexicalized tree-to-tree rules that contain only terminals. We set the maximal length of phrase pairs to 10. For tree-based tree-to-tree model, the coverage was below 8% even the maximal node count was set to 10. This suggests that conventional tree-to-tree models lose over 92% linguistically unmotivated mappings due to hard syntactic constraints. The absence of such non-syntactic mappings prevents tree-based tree-to-tree models from achieving comparable results to phrase-based models. With more parses included

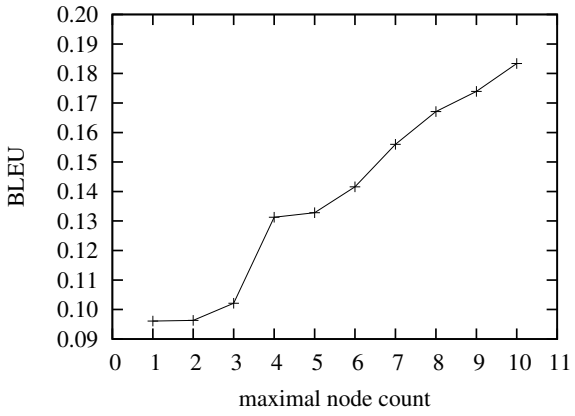

Figure 5: Effect of maximal node count on BLEU scores.

in packed forests, the rule coverage increased accordingly. When $p = 10$ and $n = 10$, the coverage was 9.7%, higher than that of $p = 0$. As a result, packed forests enable tree-to-tree models to capture more useful source-target mappings and therefore improve translation quality. [2]

5.4 Training and Decoding Time

Table 4 gives the rule extraction time (seconds/1000 sentence pairs) and decoding time (second/sentence) with varying pruning thresholds. We found that the extraction time grew faster than decoding time with the increase of p. One possible reason is that the number of frontier tree pairs (see Figure 3) rose dramatically when more parses were included in packed forests.

5.5 Effect of Maximal Node Count

Figure 5 shows the effect of maximal node count on BLEU scores. With the increase of maximal node count, the BLEU score increased dramatically. This implies that allowing tree-to-tree rules to capture larger contexts will strengthen the expressive power of tree-to-tree model.

5.6 Results on Larger Data

We also conducted an experiment on larger data to further examine the effectiveness of our approach. We concatenated the small corpus we used above and the FBIS corpus. After removing the sentences that we failed to obtain forests, the new training corpus contained about 260K sentence pairs with 7.39M Chinese words and 9.41M English words. We set the forest pruning threshold $p = 5$. Moses obtained a BLEU score of 0.3043 and our forest-based tree-to-tree system achieved a BLEU score of 0.3059. The difference is still not significant statistically.

[2] Note that even we used packed forests, the rule coverage was still very low. One reason is that we set the maximal phrase length to 10 words, while an STSG rule with 10 nodes in each tree usually cannot subsume 10 words.

6 Related Work

In machine translation, the concept of packed forest is first used by Huang and Chiang (2007) to characterize the search space of decoding with language models. The first direct use of packed forest is proposed by Mi et al. (2008). They replace 1-best trees with packed forests both in training and decoding and show superior translation quality over the state-of-the-art hierarchical phrase-based system. We follow the same direction and apply packed forests to tree-to-tree translation.

Zhang et al. (2008) present a tree-to-tree model that uses STSG. To capture non-syntactic phrases, they apply tree-sequence rules (Liu et al., 2007) to tree-to-tree models. Their extraction algorithm first identifies initial rules and then obtains abstract rules. While this method works for 1-best tree pairs, it cannot be applied to packed forest pairs because it is impractical to enumerate all tree pairs over a phrase pair.

While Galley (2004) describes extracting tree-to-string rules from 1-best trees, Mi and Huang et al. (2008) go further by proposing a method for extracting tree-to-string rules from aligned forest-string pairs. We follow their work and focus on identifying tree-tree pairs in a forest pair, which is more difficult than the tree-to-string case.

7 Conclusion

We have shown how to improve tree-to-tree translation with packed forests, which compactly encode exponentially many parses. To learn STSG rules from aligned forest pairs, we first identify minimal rules and then get composed rules. The decoder finds the best derivation that have the source yield of one source tree in the forest. Experiments show that using packed forests in tree-to-tree translation results in dramatic improvements over using 1-best trees. Our system also achieves comparable performance with the state-of-the-art phrase-based system Moses.

Acknowledgement

The authors were supported by National Natural Science Foundation of China, Contracts 60603095 and 60736014, and 863 State Key Project No. 2006AA010108. Part of this work was done while Yang Liu was visiting the SMT group led by Stephan Vogel at CMU. We thank the anonymous reviewers for their insightful comments. Many thanks go to Liang Huang, Haitao Mi, and Hao Xiong for their invaluable help in producing packed forests. We are also grateful to Andreas Zollmann, Vamshi Ambati, and Kevin Gimpel for their helpful feedback.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL 2005*.

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

Brooke Cowan, Ivona Kučerová, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proc. of EMNLP 2006*.

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proc. of EMNLP 2007*.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of ACL 2005*.

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

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. of NAACL/HLT 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 COLING/ACL 2006*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proc. of IWPT 2005*.

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

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

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL/HLT 2008*.

Phillip Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL 2003*.

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 *Proc. of ACL 2007 (demonstration session)*.

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

Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *Proc. of ACL 2007*.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. Spmt: Statistical machine translation with syntactified target language phrases. In *Proc. of EMNLP 2006*.

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

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. of ACL/HLT 2008*.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL 2002*.

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

Chris Quirk and Simon Corston-Oliver. 2006. The impact of parsing quality on syntactically-informed statistical machine translation. In *Proc. of EMNLP 2006*.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL/HLT 2008*.

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

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting bleu/nist scores how much improvement do we need to have a better system? In *Proc. of LREC 2004*.

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/HLT 2008*.