Constituency to Dependency Translation with Forests

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

Abstract

Tree-to-string systems (and their forest-based extensions) have gained steady popularity thanks to their simplicity and efficiency, but there is a major limitation: they are unable to guarantee the grammaticality of the output, which is explicitly modeled in string-to-tree systems via target-side syntax. We thus propose to combine the advantages of both, and present a novel constituency-to-dependency translation model, which uses constituency forests on the source side to direct the translation, and dependency trees on the target side (as a language model) to ensure grammaticality. Medium-scale experiments show an absolute and statistically significant improvement of +0.7 BLEU points over a state-of-the-art forest-based tree-to-string system even with fewer rules. This is also the first time that a tree-to-tree model can surpass tree-to-string counterparts.

1 Introduction

Linguistically syntax-based statistical machine translation models have made promising progress in recent years. By incorporating the syntactic annotations of parse trees from *both* or *either* side(s) of the bitext, they are believed better than phrase-based counterparts in reorderings. Depending on the type of input, these models can be broadly divided into two categories (see Table 1): the *string-based* systems whose input is a string to be simultaneously parsed and translated by a synchronous grammar, and the *tree-based* systems whose input is already a parse tree to be directly converted into a target tree or string. When we also take into account the type of output (tree or string), the *tree-based* systems can be divided into *tree-to-string* and *tree-to-tree* efforts.

tree on	*examples (partial)*	*fast*	*gram.*	BLEU
source	Liu06, Huang06	+	-	+
target	Galley06, Shen08	-	+	+
both	Ding05, Liu09	+	+	-
both	**our work**	+	+	+

Table 1: A classification and comparison of linguistically syntax-based SMT systems, where *gram.* denotes grammaticality of the output.

On one hand, tree-to-string systems (Liu et al., 2006; Huang et al., 2006) have gained significant popularity, especially after incorporating packed forests (Mi et al., 2008; Mi and Huang, 2008; Liu et al., 2009; Zhang et al., 2009). Compared with their string-based counterparts, tree-based systems are much faster in decoding (linear time vs. cubic time, see (Huang et al., 2006)), do not require a binary-branching grammar as in string-based models (Zhang et al., 2006; Huang et al., 2009), and can have separate grammars for parsing and translation (Huang et al., 2006). However, they have a major limitation that they do not have a principled mechanism to guarantee grammaticality on the target side, since there is no linguistic tree structure of the output.

On the other hand, string-to-tree systems explicitly model the grammaticality of the output by using target syntactic trees. Both string-to-constituency system (e.g., (Galley et al., 2006; Marcu et al., 2006)) and string-to-dependency model (Shen et al., 2008) have achieved significant improvements over the state-of-the-art formally syntax-based system Hiero (Chiang, 2007). However, those systems also have some limitations that they run slowly (in cubic time) (Huang et al., 2006), and do not utilize the useful syntactic information on the source side.

We thus combine the advantages of both tree-to-string and string-to-tree approaches, and propose a novel constituency-to-dependency model, which uses constituency forests on the source side to direct translation, and dependency trees on the target side to guarantee grammaticality of the output. In contrast to conventional tree-to-tree approaches (Ding and Palmer, 2005; Quirk et al., 2005; Xiong et al., 2007; Zhang et al., 2007; Liu et al., 2009), which only make use of a single type of trees, our model is able to combine two types of trees, outperforming both phrase-based and tree-to-string systems. Current tree-to-tree models (Xiong et al., 2007; Zhang et al., 2007; Liu et al., 2009) still have not outperformed the phrase-based system Moses (Koehn et al., 2007) significantly even with the help of forests.[1]

Our new constituency-to-dependency model (Section 2) extracts rules from word-aligned pairs of source constituency forests and target dependency trees (Section 3), and translates source constituency forests into target dependency trees with a set of features (Section 4). Medium data experiments (Section 5) show a statistically significant improvement of +0.7 BLEU points over a state-of-the-art forest-based tree-to-string system even with less translation rules, this is also the first time that a tree-to-tree model can surpass tree-to-string counterparts.

2 Model

Figure 1 shows a word-aligned source constituency forest F_c and target dependency tree D_e, our constituency to dependency translation model can be formalized as:

$$\begin{aligned} \mathrm{P}(F_c, D_e) &= \sum_{C_c \in F_c} \mathrm{P}(C_c, D_e) \\ &= \sum_{C_c \in F_c} \sum_{o \in O} \mathrm{P}(O) \\ &= \sum_{C_c \in F_c} \sum_{o \in O} \prod_{r \in o} \mathrm{P}(r), \end{aligned} \quad (1)$$

where C_c is a constituency tree in F_c, o is a derivation that translates C_c to D_e, O is the set of derivation, r is a constituency to dependency translation rule.

[1]According to the reports of Liu et al. (2009), their forest-based constituency-to-constituency system achieves a comparable performance against Moses (Koehn et al., 2007), but a significant improvement of +3.6 BLEU points over the 1-best tree-based constituency-to-constituency system.

2.1 Constituency Forests on the Source Side

A constituency forest (in Figure 1 left) is a compact representation of all the derivations (i.e., parse trees) for a given sentence under a context-free grammar (Billot and Lang, 1989).

More formally, following Huang (2008), such a constituency forest is a pair $F_c = G^f = \langle V^f, H^f \rangle$, where V^f is the set of **nodes**, and H^f the set of **hyperedges**. For a given source sentence $c_{1:m} = c_1 \ldots c_m$, each node $v^f \in V^f$ is in the form of $X_{i,j}$, which denotes the recognition of nonterminal X spanning the substring from positions i through j (that is, $c_{i+1} \ldots c_j$). Each hyperedge $h^f \in H^f$ is a pair $\langle tails(h^f), head(h^f) \rangle$, where $head(h^f) \in V^f$ is the **consequent node** in the deductive step, and $tails(h^f) \in (V^f)^*$ is the list of **antecedent nodes**. For example, the hyperedge h_0^f in Figure 1 for deduction (*)

$$\frac{\mathrm{NPB}_{0,1} \quad \mathrm{CC}_{1,2} \quad \mathrm{NPB}_{2,3}}{\mathrm{NP}_{0,3}}\ , \qquad (*)$$

is notated:

$$\langle (\mathrm{NPB}_{0,1},\ \mathrm{CC}_{1,2},\ \mathrm{NPB}_{2,3}),\ \ \mathrm{NP}_{0,3} \rangle.$$

where

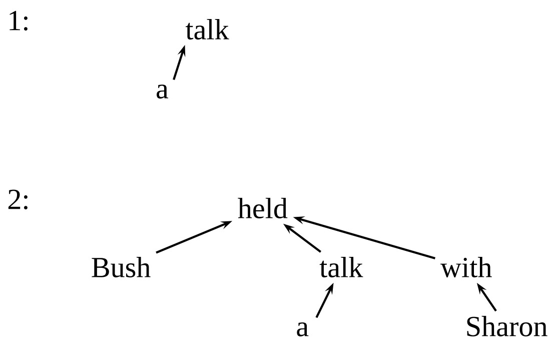
$$head(h_0^f) = \{\mathrm{NP}_{0,3}\},$$
and
$$tails(h_0^f) = \{\mathrm{NPB}_{0,1}, \mathrm{CC}_{1,2}, \mathrm{NPB}_{2,3}\}.$$

The solid line in Figure 1 shows the best parse tree, while the dashed one shows the second best tree. Note that common sub-derivations like those for the verb $\mathrm{VPB}_{3,5}$ are shared, which allows the forest to represent exponentially many parses in a compact structure.

We also denote $IN(v^f)$ to be the set of **incoming hyperedges** of node v^f, which represents the different ways of deriving v^f. Take node $\mathrm{IP}_{0,5}$ in Figure 1 for example, $IN(\mathrm{IP}_{0,5}) = \{h_1^f, h_2^f\}$. There is also a distinguished **root node** TOP in each forest, denoting the goal item in parsing, which is simply $\mathrm{S}_{0,m}$ where S is the start symbol and m is the sentence length.

2.2 Dependency Trees on the Target Side

A dependency tree for a sentence represents each word and its syntactic dependents through directed arcs, as shown in the following examples. The main advantage of a dependency tree is that it can explore the long distance dependency.



We use the lexicon dependency grammar (Hellwig, 2006) to express a projective dependency tree. Take the dependency trees above for example, they will be expressed:

1: (a) talk

2: (Bush) held ((a) talk) (with (Sharon))

where the lexicons in brackets represent the dependencies, while the lexicon out the brackets is the head.

More formally, a dependency tree is also a pair $D_e = G^d = \langle V^d, H^d \rangle$. For a given target sentence $e_{1:n} = e_1 \dots e_n$, each node $v^d \in V^d$ is a word e_i ($1 \leqslant i \leqslant n$), each hyperedge $h^d \in H^d$ is a directed arc $\langle v_i^d, v_j^d \rangle$ from node v_i^d to its head node v_j^d. Following the formalization of the constituency forest scenario, we denote a pair $\langle tails(h^d), head(h^d) \rangle$ to be a hyperedge h^d, where $head(h^d)$ is the head node, $tails(h^d)$ is the node where h^d leaves from.

We also denote $L_l(v^d)$ and $L_r(v^d)$ to be the left and right children sequence of node v^d from the nearest to the farthest respectively. Take the node $v_2^d =$ "held" for example:

$$L_l(v_2^d) = \{\text{Bush}\},$$
$$L_r(v_2^d) = \{\text{talk, with}\}.$$

2.3 Hypergraph

Actually, both the constituency forest and the dependency tree can be formalized as a **hypergraph** G, a pair $\langle V, H \rangle$. We use G^f and G^d to distinguish them. For simplicity, we also use F_c and D_e to denote a constituency forest and a dependency tree respectively. Specifically, the size of $tails(h^d)$ of a hyperedge h^d in a dependency tree is a constant one.

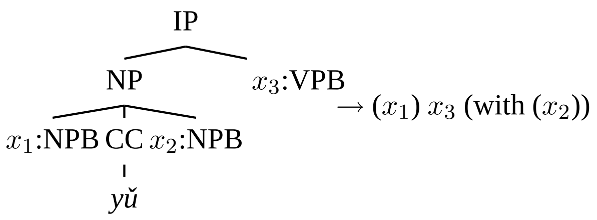


Figure 2: Example of the rule r_1. The Chinese conjunction *yǔ* "and" is translated into English preposition "with".

3 Rule Extraction

We extract constituency to dependency rules from word-aligned source constituency forest and target dependency tree pairs (Figure 1). We mainly extend the tree-to-string rule extraction algorithm of Mi and Huang (2008) to our scenario. In this section, we first formalize the constituency to string translation rule (Section 3.1). Then we present the restrictions for dependency structures as well formed fragments (Section 3.2). Finally, we describe our rule extraction algorithm (Section 3.3), fractional counts computation and probabilities estimation (Section 3.4).

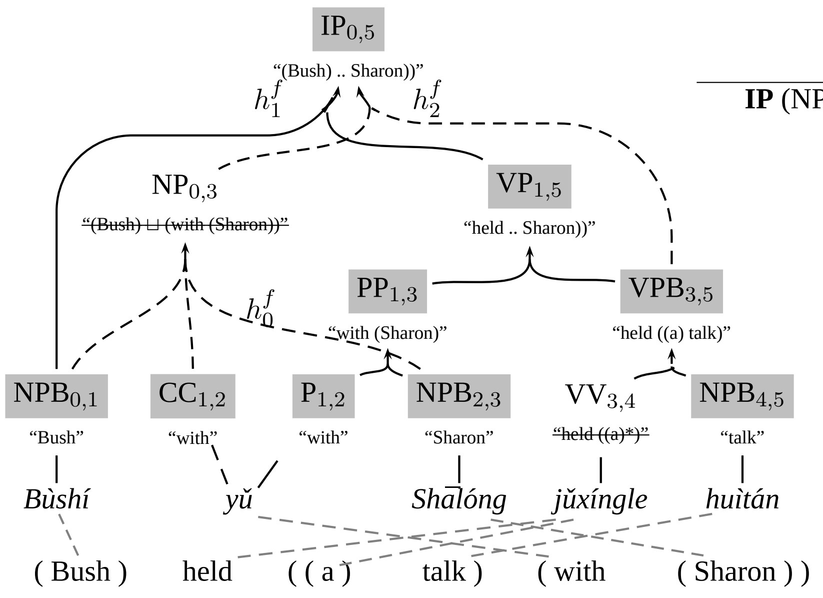
3.1 Constituency to Dependency Rule

More formally, a **constituency to dependency translation rule** r is a tuple $\langle lhs(r), rhs(r), \phi(r) \rangle$, where $lhs(r)$ is the source side tree fragment, whose internal nodes are labeled by nonterminal symbols (like NP and VP), and whose frontier nodes are labeled by source language words c_i (like "*yǔ*") or variables from a set $\mathcal{X} = \{x_1, x_2, \dots\}$; $rhs(r)$ is expressed in the target language dependency structure with words e_j (like "with") and variables from the set $\mathcal{X}$; and $\phi(r)$ is a mapping from $\mathcal{X}$ to nonterminals. Each variable $x_i \in \mathcal{X}$ occurs *exactly once* in $lhs(r)$ and *exactly once* in $rhs(r)$. For example, the rule r_1 in Figure 2,

$$lhs(r_1) = \text{IP(NP(}x_1\ \text{CC(}yǔ\text{)}\ x_2\text{)}\ x_3\text{)},$$
$$rhs(r_1) = (x_1)\ x_3\ \text{(with}\ (x_2)\text{)},$$
$$\phi(r_1) = \{x_1 \mapsto \text{NPB}, x_2 \mapsto \text{NPB}, x_3 \mapsto \text{VPB}\}.$$

3.2 Well Formed Dependency Fragment

Following Shen et al. (2008), we also restrict $rhs(r)$ to be **well formed** dependency fragment. The main difference between us is that we use more flexible restrictions. Given a dependency



Minimal rules extracted

IP (NP(x_1:NPB x_2:CC x_3:NPB) x_4:VPB) → (x_1) x_4 (x_2 (x_3))
IP (x_1:NPB x_2:VP) → (x_1) x_2
VP (x_1:PP x_2:VPB) → x_2 (x_1)
PP (x_1:P x_2:NPB) → x_1 (x_2)
VPB (VV(*jǔxíngle*)) x_1:NPB) → held ((a) x_1)
NPB (*Bùshí*) → Bush
NPB (*huìtán*) → talk
CC (*yǔ*) → with
P (*yǔ*) → with
NPB (*Shālóng*) → Sharon

Figure 1: Forest-based constituency to dependency rule extraction.

fragment $d_{i:j}$ composed by the words from i to j, two kinds of well formed structures are defined as follows:

Fixed on one node v^d_{one}, **fixed** for short, if it meets the following conditions:

- the head of v^d_{one} is out of $[i, j]$, i.e.: $\forall h^d$, if $tails(h^d) = v^d_{one} \Rightarrow head(h^d) \notin e_{i:j}$.
- the heads of other nodes except v^d_{one} are in $[i, j]$, i.e.: $\forall k \in [i, j]$ and $v^d_k \neq v^d_{one}, \forall h^d$ if $tails(h^d) = v^d_k \Rightarrow head(h^d) \in e_{i:j}$.

Floating with multi nodes M, **floating** for short, if it meets the following conditions:

- all nodes in M have a same head node, i.e.: $\exists x \notin [i, j], \forall h^d$ if $tails(h^d) \in M \Rightarrow head(h^d) = v^h_x$.
- the heads of other nodes not in M are in $[i, j]$, i.e.: $\forall k \in [i, j]$ and $v^d_k \notin M, \forall h^d$ if $tails(h^d) = v^d_k \Rightarrow head(h^d) \in e_{i:j}$.

Take the " (Bush) held ((a) talk))(with (Sharon)) " for example: partial fixed examples are " (Bush) held " and " held ((a) talk)"; while the partial floating examples are " (talk) (with (Sharon)) " and " ((a) talk) (with (Sharon)) ". Please note that the floating structure " (talk) (with (Sharon)) " can not be allowed in Shen et al. (2008)'s model.

The dependency structure " held ((a))" is not a well formed structure, since the head of word "a" is out of scope of this structure.

3.3 Rule Extraction Algorithm

The algorithm shown in this Section is mainly extended from the forest-based tree-to-string extraction algorithm (Mi and Huang, 2008). We extract rules from word-aligned source constituency forest and target dependency tree pairs (see Figure 1) in three steps:

(1) frontier set computation,

(2) fragmentation,

(3) composition.

The **frontier set** (Galley et al., 2004) is the potential points to "cut" the forest and dependency tree pair into fragments, each of which will form a **minimal rule** (Galley et al., 2006).

However, not every fragment can be used for rule extraction, since it may or may not respect to the restrictions, such as word alignments and well formed dependency structures. So we say a fragment is **extractable** if it respects to all restrictions. The root node of every extractable tree fragment corresponds to a **faithful structure** on the target side, in which case there is a "translational equivalence" between the subtree rooted at the node and the corresponding target structure. For example, in Figure 1, every node in the forest is annotated with its corresponding English structure. The NP$_{0,3}$ node maps to a non-contiguous structure "(Bush) ⊔ (with (Sharon))", the VV$_{3,4}$ node maps to a contiguous but non-faithful structure "held ((a) *)".

Algorithm 1 Forest-based constituency to dependency rule extraction.

```
Input: Source constituency forest $F_c$, target dependency tree $D_e$, and alignment $a$
Output: Minimal rule set $\mathcal{R}$
1: $fs \leftarrow$ FRONTIER$(F_c, D_e, a)$                    ▷ compute frontier set
2: for each $v^f \in fs$ do
3:     $open \leftarrow \{\langle \varnothing, \{v^f\} \rangle\}$          ▷ initial queue of growing fragments
4:     while $open \neq \varnothing$ do
5:         $\langle hs, exps \rangle \leftarrow open.pop()$              ▷ extract a fragment
6:         if $exps = \varnothing$ then                       ▷ nothing to expand?
7:             generate a rule $r$ using fragment $hs$         ▷ generate a rule
8:             $\mathcal{R}$.append($r$)
9:         else                                      ▷ incomplete: further expand
10:            $v' \leftarrow exps$.pop()                       ▷ a non-frontier node
11:            for each $h^f \in IN(v')$ do
12:                $newexps \leftarrow exps \cup (tails(h^f) \setminus fs)$     ▷ expand
13:                $open$.append($\langle hs \cup \{h^f\}, newexps \rangle$)
```

Following Mi and Huang (2008), given a source target sentence pair $\langle c_{1:m}, e_{1:n} \rangle$ with an alignment a, the **span** of node v^f on source forest is the set of target words aligned to leaf nodes under v^f:

$$span(v^f) \triangleq \{e_i \in e_{1:n} \mid \exists c_j \in yield(v^f), (c_j, e_i) \in a\}.$$

where the $yield(v^f)$ is all the leaf nodes under v^f. For each $span(v^f)$, we also denote $dep(v^f)$ to be its corresponding dependency structure, which represents the dependency structure of all the words in $span(v^f)$. Take the $span(\text{PP}_{1,3})$ $=${with, Sharon} for example, the corresponding $dep(\text{PP}_{1,3})$ is "with (Sharon)". A $dep(v^f)$ is **faithful structure** to node v^f if it meets the following restrictions:

- all words in $span(v^f)$ form a continuous substring $e_{i:j}$,
- every word in $span(v^f)$ is *only* aligned to leaf nodes of v^f, i.e.: $\forall e_i \in span(v^f), (c_j, e_i) \in a \Rightarrow c_j \in yield(v^f)$,
- $dep(v^f)$ is a well formed dependency structure.

For example, node $\text{VV}_{3,4}$ has a non-faithful structure (crossed out in Figure 1), since its $dep(\text{VV}_{3,4} =$ " held ((a) *)" is not a well formed structure, where the head of word "a" lies in the outside of its words covered. Nodes with faithful structure form the **frontier set** (shaded nodes in Figure 1) which serve as potential cut points for rule extraction.

Given the frontier set, fragmentation step is to "cut" the forest at all frontier nodes and form tree fragments, each of which forms a rule with variables matching the frontier descendant nodes. For example, the forest in Figure 1 is cut into 10 pieces, each of which corresponds to a minimal rule listed on the right.

Our rule extraction algorithm is formalized in Algorithm 1. After we compute the frontier set *fs* (line 1). We visit each frontier node $v^f \in fs$ on the source constituency forest F_c, and keep a queue $open$ of growing fragments rooted at v^f. We keep expanding incomplete fragments from $open$, and extract a rule if a complete fragment is found (line 7). Each fragment *hs* in $open$ is associated with a list of *expansion sites* ($exps$ in line 5) being the subset of leaf nodes of the current fragment that are *not* in the frontier set. So each fragment along hyperedge h is associated with

$$exps = tails(h^f) \setminus fs.$$

A fragment is complete if its expansion sites is empty (line 6), otherwise we pop one expansion node v' to grow and spin-off new fragments by following hyperedges of v', adding new expansion sites (lines 11-13), until all active fragments are complete and $open$ queue is empty (line 4).

After we get all the minimal rules, we glue them together to form **composed rule**s following Galley et al. (2006). For example, the composed rule r_1 in Figure 2 is glued by the following two minimal rules:

IP (NP(x_1:NPB x_2:CC x_3:NPB) x_4:VPB) $\rightarrow$ (x_1) x_4 (x_2 (x_3)) $\quad r_2$

CC (*yǔ*) $\rightarrow$ with $\quad r_3$

where x_2:CC in r_2 is replaced with r_3 accordingly.

3.4 Fractional Counts and Rule Probabilities

Following Mi and Huang (2008), we penalize a rule r by the posterior probability of the corresponding constituent tree fragment $lhs(r)$, which can be computed in an Inside-Outside fashion, being the product of the outside probability of its root node, the inside probabilities of its leaf nodes, and the probabilities of hyperedges involved in the fragment.

$$\begin{aligned}\alpha\beta(lhs(r)) =& \alpha(root(r)) \\ & \cdot \prod_{h^f \in lhs(r)} \mathrm{P}(h^f) \\ & \cdot \prod_{v^f \in leaves(lhs(r))} \beta(v^f)\end{aligned} \tag{2}$$

where $root$(r) is the root of the rule r, $\alpha(v)$ and $\beta(v)$ are the outside and inside probabilities of node v, and $leaves(lhs(r))$ returns the leaf nodes of a tree fragment lhs(r).

We use fractional counts to compute three conditional probabilities for each rule, which will be used in the next section:

$$\mathrm{P}(r \mid lhs(r)) = \frac{c(r)}{\sum_{r':lhs(r')=lhs(r)} c(r')}, \tag{3}$$

$$\mathrm{P}(r \mid rhs(r)) = \frac{c(r)}{\sum_{r':rhs(r')=rhs(r)} c(r')}, \tag{4}$$

$$\mathrm{P}(r \mid root(r)) = \frac{c(r)}{\sum_{r':root(r')=root(r)} c(r')}. \tag{5}$$

4 Decoding

Given a source forest F_c, the decoder searches for the best derivation o^* among the set of all possible derivations O, each of which forms a source side constituent tree $T_c(o)$, a target side string $e(o)$, and a target side dependency tree $D_e(o)$:

$$\begin{aligned} o^* = \arg\max_{T_c \in F_c, o \in O} & \lambda_1 \log \mathrm{P}(o \mid T_c) \\ & +\lambda_2 \log \mathrm{P}_{\mathrm{lm}}(e(o)) \\ & +\lambda_3 \log \mathrm{P}_{\mathrm{DLM_w}}(D_e(o)) \\ & +\lambda_4 \log \mathrm{P}_{\mathrm{DLM_p}}(D_e(o)) \\ & +\lambda_5 \log \mathrm{P}(T_c(o)) \\ & +\lambda_6 ill(o) + \lambda_7 |o| + \lambda_8 |e(o)|, \end{aligned} \tag{6}$$

where the first two terms are translation and language model probabilities, $e(o)$ is the target string (English sentence) for derivation o, the third and forth items are the dependency language model probabilities on the target side computed with words and POS tags separately, $D_e(o)$ is the target dependency tree of o, the fifth one is the parsing probability of the source side tree $T_c(o) \in F_c$, the $ill(o)$ is the penalty for the number of ill-formed dependency structures in o, and the last two terms are derivation and translation length penalties, respectively. The conditional probability $\mathrm{P}(o \mid T_c)$ is decomposes into the product of rule probabilities:

$$\mathrm{P}(o \mid T_c) = \prod_{r \in o} \mathrm{P}(r), \tag{7}$$

where each $\mathrm{P}(r)$ is the product of five probabilities:

$$\begin{aligned}\mathrm{P}(r) =& \mathrm{P}(r \mid lhs(r))^{\lambda_9} \cdot \mathrm{P}(r \mid rhs(r))^{\lambda_{10}} \\ & \cdot \mathrm{P}(r \mid root(lhs(r)))^{\lambda_{11}} \\ & \cdot \mathrm{P}_{\mathrm{lex}}(lhs(r) \mid rhs(r))^{\lambda_{12}} \\ & \cdot \mathrm{P}_{\mathrm{lex}}(rhs(r) \mid lhs(r))^{\lambda_{13}},\end{aligned} \tag{8}$$

where the first three are conditional probabilities based on fractional counts of rules defined in Section 3.4, and the last two are lexical probabilities. When computing the lexical translation probabilities described in (Koehn et al., 2003), we only take into accout the terminals in a rule. If there is no terminal, we set the lexical probability to 1.

The decoding algorithm works in a bottom-up search fashion by traversing each node in forest F_c. We first use pattern-matching algorithm of Mi et al. (2008) to convert F_c into a **translation forest**, each hyperedge of which is associated with a constituency to dependency translation rule. However, pattern-matching failure[2] at a node v^f will

[2]Pattern-matching failure at a node v^f means there is no translation rule can be matched at v^f or no translation hyperedge can be constructed at v^f.

cut the derivation path and lead to translation failure. To tackle this problem, we construct a **pseudo translation rule** for each parse hyperedge $h^f \in IN(v^f)$ by mapping the CFG rule into a target dependency tree using the head rules of Magerman (1995). Take the hyperedge h_0^f in Figure1 for example, the corresponding pseudo translation rule is:

$$\textbf{NP}(x_1\text{:NPB } x_2\text{:CC } x_3\text{:NPB}) \rightarrow (x_1)\ (x_2)\ x_3,$$

since the x_3:NPB is the head word of the CFG rule: NP → NPB CC NPB.

After the translation forest is constructed, we traverse each node in translation forest also in bottom-up fashion. For each node, we use the **cube pruning** technique (Chiang, 2007; Huang and Chiang, 2007) to produce partial hypotheses and compute all the feature scores including the dependency language model score (Section 4.1). If all the nodes are visited, we trace back along the 1-best derivation at goal item $\mathrm{S}_{0,m}$ and build a target side dependency tree. For k-best search after getting 1-best derivation, we use the lazy Algorithm 3 of Huang and Chiang (2005) that works backwards from the root node, incrementally computing the second, third, through the kth best alternatives.

4.1 Dependency Language Model Computing

We compute the score of a dependency language model for a dependency tree D_e in the same way proposed by Shen et al. (2008). For each nonterminal node $v_h^d = e_h$ in D_e and its children sequences $L_l = e_{l_1}, e_{l_2}...e_{l_i}$ and $L_r = e_{r_1}, e_{r_2}...e_{r_j}$, the probability of a trigram is computed as follows:

$$\mathrm{P}(L_l, L_r \mid e_h\S) = \mathrm{P}(L_l \mid e_h\S) \cdot \mathrm{P}(L_r \mid e_h\S), \quad (9)$$

where the $\mathrm{P}(L_l \mid e_h\S)$ is decomposed to be:

$$\begin{aligned} \mathrm{P}(L_l \mid e_h\S) =& \mathrm{P}(e_{l_l} \mid e_h\S) \\ & \cdot \mathrm{P}(e_{l_2} \mid e_{l_1}, e_h\S) \\ & \dots \\ & \cdot \mathrm{P}(e_{l_n} \mid e_{l_{n-1}}, e_{l_{n-2}}). \end{aligned} \quad (10)$$

We use the suffix "§" to distinguish the head word and child words in the dependency language model.

In order to alleviate the problem of data sparse, we also compute a dependency language model for POS tages over a dependency tree. We store the POS tag information on the target side for each constituency-to-dependency rule. So we will also generate a POS taged dependency tree simultaneously at the decoding time. We calculate this dependency language model by simply replacing each e_i in equation 9 with its tag $t(e_i)$.

5 Experiments

5.1 Data Preparation

Our training corpus consists of 239K sentence pairs with about 6.9M/8.9M words in Chinese/English, respectively. We first word-align them by GIZA++ (Och and Ney, 2000) with refinement option "grow-diag-and" (Koehn et al., 2003), and then parse the Chinese sentences using the parser of Xiong et al. (2005) into parse forests, which are pruned into relatively small forests with a pruning threshold 3. We also parse the English sentences using the parser of Charniak (2000) into 1-best constituency trees, which will be converted into dependency trees using Magerman (1995)'s head rules. We also store the POS tag information for each word in dependency trees, and compute two different dependency language models for words and POS tags in dependency tree separately. Finally, we apply translation rule extraction algorithm described in Section 3. We use SRI Language Modeling Toolkit (Stolcke, 2002) to train a 4-gram language model with Kneser-Ney smoothing on the first 1/3 of the Xinhua portion of Gigaword corpus. At the decoding step, we again parse the input sentences into forests and prune them with a threshold 10, which will direct the translation (Section 4).

We use the 2002 NIST MT Evaluation test set as our development set and the 2005 NIST MT Evaluation test set as our test set. We evaluate the translation quality using the BLEU-4 metric (Papineni et al., 2002), which is calculated by the script mteval-v11b.pl with its default setting which is case-insensitive matching of n-grams. We use the standard minimum error-rate training (Och, 2003) to tune the feature weights to maximize the system's BLEU score on development set.

5.2 Results

Table 2 shows the results on the test set. Our baseline system is a state-of-the-art forest-based constituency-to-string model (Mi et al., 2008), or *forest c2s* for short, which translates a source forest into a target string by pattern-matching the constituency-to-string (*c2s*) rules and the bilingual phrases (*s2s*). The baseline system extracts 31.9M *c2s* rules, 77.9M *s2s* rules respectively and achieves a BLEU score of 34.17 on the test set[3].

At first, we investigate the influence of different rule sets on the performance of baseline system. We first restrict the target side of translation rules to be well-formed structures, and we extract 13.8M constituency-to-dependency (*c2d*) rules, which is 43% of *c2s* rules. We also extract 9.0M string-to-dependency (*s2d*) rules, which is only 11.6% of *s2s* rules. Then we convert *c2d* and *s2d* rules to *c2s* and *s2s* rules separately by removing the target-dependency structures and feed them into the baseline system. As shown in the third line in the column of BLEU score, the performance drops 1.7 BLEU points over baseline system due to the poorer rule coverage. However, when we further use all *s2s* rules instead of *s2d* rules in our next experiment, it achieves a BLEU score of 34.03, which is very similar to the baseline system. Those results suggest that restrictions on *c2s* rules won't hurt the performance, but restrictions on *s2s* will hurt the translation quality badly. So we should utilize all the *s2s* rules in order to preserve a good coverage of translation rule set.

The last two lines in Table 2 show the results of our new forest-based constituency-to-dependency model (*forest c2d* for short). When we only use *c2d* and *s2d* rules, our system achieves a BLEU score of 33.25, which is lower than the baseline system in the first line. But, with the same rule set, our model still outperform the result in the second line. This suggests that using dependency language model really improves the translation quality by less than 1 BLEU point.

In order to utilize all the *s2s* rules and increase the rule coverage, we parse the target strings of the *s2s* rules into dependency fragments, and construct the *pseudo s2d* rules (*s2s-dep*). Then we use *c2d* and *s2s-dep* rules to direct the translation. With the help of the dependency language model, our new model achieves a significant improvement of +0.7 BLEU points over the *forest c2s* baseline system ($p < 0.05$, using the *sign-test* suggested by

System	Rule Set		BLEU
	Type	#	
forest c2s	c2s s2s	31.9M 77.9M	34.17
	c2d s2d	13.8M 9.0M	32.48(↓1.7)
	c2d s2s	13.8M 77.9M	34.03(↓0.1)
forest c2d	c2d s2d	13.8M 9.0M	33.25(↓0.9)
	c2d s2s-dep	13.8M 77.9M	34.88(↑0.7)

Table 2: Statistics of different types of rules extracted on training corpus and the BLEU scores on the test set.

Collins et al. (2005)). For the first time, a tree-to-tree model can surpass tree-to-string counterparts significantly even with fewer rules.

6 Related Work

The concept of packed forest has been used in machine translation for several years. For example, Huang and Chiang (2007) use forest to characterize the search space of decoding with integrated language models. Mi et al. (2008) and Mi and Huang (2008) use forest to direct translation and extract rules rather than 1-best tree in order to weaken the influence of parsing errors, this is also the first time to use forest directly in machine translation. Following this direction, Liu et al. (2009) and Zhang et al. (2009) apply forest into tree-to-tree (Zhang et al., 2007) and tree-sequence-to-string models(Liu et al., 2007) respectively. Different from Liu et al. (2009), we apply forest into a new constituency tree to dependency tree translation model rather than constituency tree-to-tree model.

Shen et al. (2008) present a string-to-dependency model. They define the well-formed dependency structures to reduce the size of translation rule set, and integrate a dependency language model in decoding step to exploit long distance word relations. This model shows a significant improvement over the state-of-the-art hierarchical phrase-based system (Chiang, 2005). Compared with this work, we put fewer restrictions on the definition of well-formed dependency structures in order to extract more rules; the

[3]According to the reports of Liu et al. (2009), with a more larger training corpus (FBIS plus 30K) but *no* name entity translations (+1 BLEU points if it is used), their forest-based constituency-to-constituency model achieves a BLEU score of 30.6, which is similar to Moses (Koehn et al., 2007). So our baseline system is much better than the BLEU score (30.6+1) of the constituency-to-constituency system and Moses.

other difference is that we can also extract more expressive constituency to dependency rules, since the source side of our rule can encode multi-level reordering and contain more variables being larger than two; furthermore, our rules can be pattern-matched at high level, which is more reasonable than using glue rules in Shen et al. (2008)'s scenario; finally, the most important one is that our model runs very faster.

Liu et al. (2009) propose a forest-based constituency-to-constituency model, they put more emphasize on how to utilize parse forest to increase the tree-to-tree rule coverage. By contrast, we only use 1-best dependency trees on the target side to explore long distance relations and extract translation rules. Theoretically, we can extract more rules since dependency tree has the best inter-lingual phrasal cohesion properties (Fox, 2002).

7 Conclusion and Future Work

In this paper, we presented a novel forest-based constituency-to-dependency translation model, which combines the advantages of both tree-to-string and string-to-tree systems, runs fast and guarantees grammaticality of the output. To learn the constituency-to-dependency translation rules, we first identify the frontier set for all the nodes in the constituency forest on the source side. Then we fragment them and extract minimal rules. Finally, we glue them together to be composed rules. At the decoding step, we first parse the input sentence into a constituency forest. Then we convert it into a translation forest by patter-matching the constituency to string rules. Finally, we traverse the translation forest in a bottom-up fashion and translate it into a target dependency tree by incorporating string-based and dependency-based language models. Using all constituency-to-dependency translation rules and bilingual phrases, our model achieves +0.7 points improvement in BLEU score significantly over a state-of-the-art forest-based tree-to-string system. This is also the first time that a tree-to-tree model can surpass tree-to-string counterparts.

In the future, we will do more experiments on rule coverage to compare the constituency-to-constituency model with our model. Furthermore, we will replace 1-best dependency trees on the target side with dependency forests to further increase the rule coverage.

Acknowledgement

The authors were supported by National Natural Science Foundation of China, Contracts 60736014 and 90920004, and 863 State Key Project No. 2006AA010108. We thank the anonymous reviewers for their insightful comments. We are also grateful to Liang Huang for his valuable suggestions.

References

Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of ACL '89*, pages 143–151.

Eugene Charniak. 2000. A maximum-entropy inspired parser. In *Proceedings of NAACL*, pages 132–139.

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

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

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

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL*, pages 541–548, June.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *In Proceedings of EMNLP-02*.

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

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*, pages 961–968, July.

Peter Hellwig. 2006. *Parsing with Dependency Grammars*, volume II. An International Handbook of Contemporary Research.

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

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151, June.

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

Liang Huang, Hao Zhang, Daniel Gildea, , and Kevin Knight. 2009. Binarization of synchronous context-free grammars. *Comput. Linguist.*

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

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

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*, pages 177–180, June.

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

Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *Proceedings of ACL*, pages 704–711, June.

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

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of ACL*, pages 276–283, June.

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

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP 2008*, pages 206–214, Honolulu, Hawaii, October.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08:HLT*, pages 192–199, Columbus, Ohio, June.

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

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

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

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

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, June.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.

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.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2007. A dependency treelet string correspondence model for statistical machine translation. In *Proceedings of SMT*, pages 40–47.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proc. of HLT-NAACL*.

Min Zhang, Hongfei Jiang, Aiti Aw, Jun Sun, Sheng Li, and Chew Lim Tan. 2007. A tree-to-tree alignment-based model for statistical machine translation. In *Proceedings of MT-Summit*.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw, and Chew Lim Tan. 2009. Forest-based tree sequence to string translation model. In *Proceedings of the ACL/IJCNLP 2009*.