Forest-based Translation Rule Extraction

Haitao Mi[1]
[1]Key Lab. of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
htmi@ict.ac.cn

Liang Huang[2,1]
[2]Dept. of Computer & Information Science
University of Pennsylvania
3330 Walnut St., Levine Hall
Philadelphia, PA 19104, USA
lhuang3@cis.upenn.edu

Abstract

Translation rule extraction is a fundamental problem in machine translation, especially for *linguistically syntax-based* systems that need parse trees from either or both sides of the bitext. The current dominant practice only uses 1-best trees, which adversely affects the rule set quality due to parsing errors. So we propose a novel approach which extracts rules from a *packed forest* that compactly encodes exponentially many parses. Experiments show that this method improves translation quality by over 1 BLEU point on a state-of-the-art tree-to-string system, and is 0.5 points better than (and twice as fast as) extracting on 30-best parses. When combined with our previous work on forest-based decoding, it achieves a 2.5 BLEU points improvement over the baseline, and even outperforms the hierarchical system of Hiero by 0.7 points.

1 Introduction

Automatic extraction of translation rules is a fundamental problem in statistical machine translation, especially for many syntax-based models where translation rules directly encode linguistic knowledge. Typically, these models extract rules using parse trees from *both* or *either* side(s) of the bitext. The former case, with trees on both sides, is often called *tree-to-tree* models; while the latter case, with trees on either source or target side, include both *tree-to-string* and *string-to-tree* models (see Table 1). Leveraging from structural and linguistic information from parse trees, these models are believed to be better than their phrase-based counterparts in handling non-local reorderings, and have achieved promising translation results.[1]

source target	*examples (partial)*
tree-to-tree	Ding and Palmer (2005)
tree-to-string	Liu et al. (2006); Huang et al. (2006)
string-to-tree	Galley et al. (2006)
string-to-string	Chiang (2005)

Table 1: A classification of syntax-based MT. The first three use *linguistic syntax*, while the last one only *formal syntax*. Our experiments cover the second type using a packed forest in place of the tree for rule-extraction.

However, these systems suffer from a major limitation, that the rule extractor only uses 1-best parse tree(s), which adversely affects the rule set quality due to parsing errors. To make things worse, modern statistical parsers are often trained on domains quite different from those used in MT. By contrast, *formally syntax-based* models (Chiang, 2005) do not rely on parse trees, yet usually perform better than these linguistically sophisticated counterparts.

To alleviate this problem, an obvious idea is to extract rules from k-best parses instead. However, a k-best list, with its limited scope, has too few variations and too many redundancies (Huang, 2008). This situation worsens with longer sentences as the number of possible parses grows exponentially with the sentence length and a k-best list will only capture a tiny fraction of the whole space. In addition, many subtrees are repeated across different parses, so it is

[1]For example, in recent NIST Evaluations, some of these models (Galley et al., 2006; Quirk et al., 2005; Liu et al., 2006) ranked among top 10. See http://www.nist.gov/speech/tests/mt/.

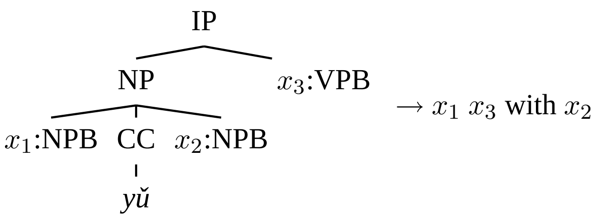


Figure 1: Example translation rule r_1. The Chinese conjunction *yǔ* "and" is translated into English prep. "with".

also inefficient to extract rules separately from each of these very similar trees (or from the cross-product of k^2 similar tree-pairs in tree-to-tree models).

We instead propose a novel approach that extracts rules from *packed forests* (Section 3), which compactly encodes many more alternatives than k-best lists. Experiments (Section 5) show that forest-based extraction improves BLEU score by over 1 point on a state-of-the-art tree-to-string system (Liu et al., 2006; Mi et al., 2008), which is also 0.5 points better than (and twice as fast as) extracting on 30-best parses. When combined with our previous orthogonal work on forest-based decoding (Mi et al., 2008), the forest-forest approach achieves a 2.5 BLEU points improvement over the baseline, and even outperforms the hierarchical system of Hiero, one of the best-performing systems to date.

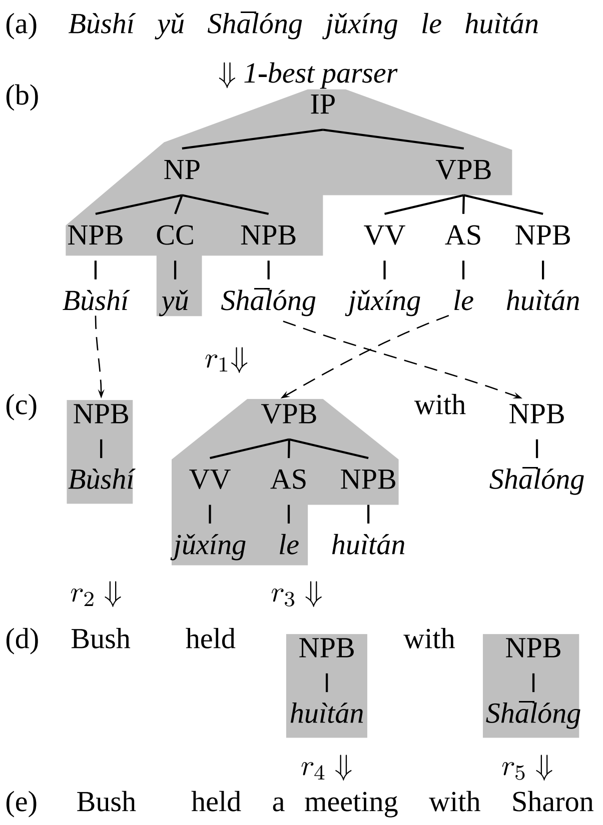
Besides tree-to-string systems, our method is also applicable to other paradigms such as the string-to-tree models (Galley et al., 2006) where the rules are in the reverse order, and easily generalizable to pairs of forests in tree-to-tree models.

2 Tree-based Translation

We review in this section the tree-based approach to machine translation (Liu et al., 2006; Huang et al., 2006), and its rule extraction algorithm (Galley et al., 2004; Galley et al., 2006).

2.1 Tree-to-String System

Current tree-based systems perform translation in two separate steps: parsing and decoding. The input string is first parsed by a parser into a 1-best tree, which will then be converted to a target language string by applying a set of tree-to-string transformation rules. For example, consider the following example translating from Chinese to English:



r_2	NPB(*Bùshí*) → Bush
r_3	VPB(VV(*jǔxíng*) AS(*le*) x_1:NPB) → held x_1
r_4	NPB(*Shālóng*) → Sharon
r_5	NPB(*huìtán*) → a meeting

Figure 2: Example derivation of tree-to-string translation, with rules used. Each shaded region denotes a tree fragment that is pattern-matched with the rule being applied.

(1)	*Bùshí*	*yǔ*	*Shālóng*	*jǔxíng*	*le*	*huìtán*
	Bush	and/with	Sharon$_1$	hold	*past.*	meeting$_2$

"Bush held a meeting$_2$ with Sharon$_1$"

Figure 2 shows how this process works. The Chinese sentence (a) is first parsed into a parse tree (b), which will be converted into an English string in 5 steps. First, at the root node, we apply rule r_1 shown in Figure 1, which translates the Chinese coordination construction ("... and ...") into an English prepositional phrase. Then, from step (c) we continue applying rules to untranslated Chinese subtrees, until we get the complete English translation in (e).[2]

[2]We swap the 1-best and 2-best parses of the example sentence from our earlier paper (Mi et al., 2008), since the current 1-best parse is easier to illustrate the rule extraction algorithm.

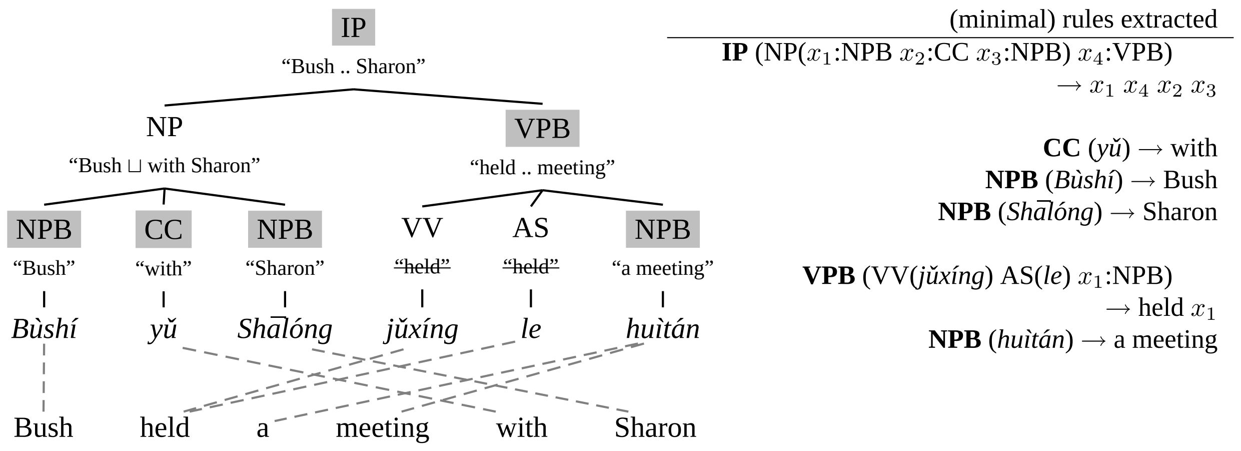


Figure 3: Tree-based rule extraction (Galley et al., 2004). Each non-leaf node in the tree is annotated with its target span (below the node), where ⊔ denotes a gap, and non-faithful spans are crossed out. Shadowed nodes are **admissible**, with contiguous and faithful spans. The first two rules can be "composed" to form rule r_1 in Figure 1.

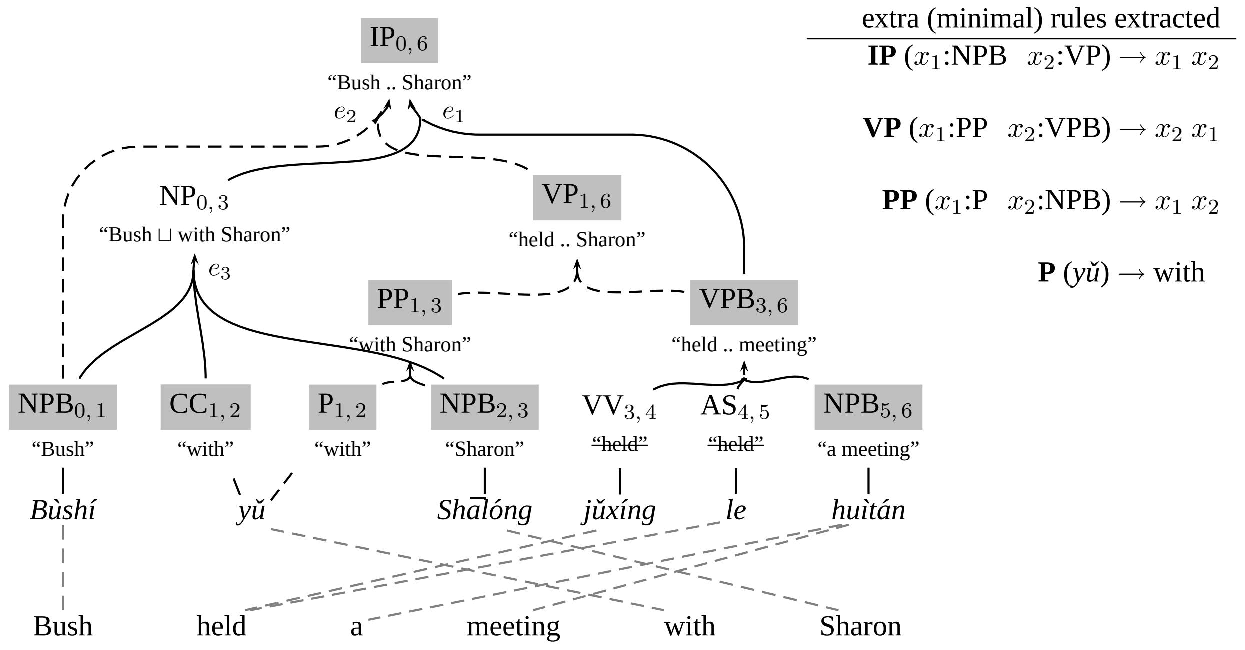


Figure 4: Forest-based rule extraction. Solid hyperedges correspond to the 1-best tree in Figure 3, while dashed hyperedges denote the alternative parse interpreting *yǔ* as a preposition in Figure 5.

More formally, a (tree-to-string) **translation rule** (Galley et al., 2004; Huang et al., 2006) 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 (like "*yǔ*") or variables from a set $\mathcal{X} = \{x_1, x_2, \ldots\}$; $rhs(r)$ is the target-side string expressed in target-language words (like "with") and variables; 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, for rule r_1 in Figure 1,

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

These rules are being used in the reverse direction of the string-to-tree transducers in Galley et al. (2004).

2.2 Tree-to-String Rule Extraction

We now briefly explain the algorithm of Galley et al. (2004) that can extract these translation rules from a word-aligned bitext with source-side parses.

Consider the example in Figure 3. The basic idea is to decompose the source (Chinese) parse into a series of tree fragments, each of which will form a rule with its corresponding English translation. However, not every fragmentation can be used for rule extraction, since it may or may not respect the alignment and reordering between the two languages. So we say a fragmentation is **well-formed** with respect to an alignment if the root node of every tree fragment corresponds to a **contiguous span** on the target side; the intuition is that there is a "translational equivalence" between the subtree rooted at the node and the corresponding target span. For example, in Figure 3, each node is annotated with its corresponding English span, where the NP node maps to a non-contiguous one "Bush $\sqcup$ with Sharon".

More formally, we need a precise formulation to handle the cases of one-to-many, many-to-one, and many-to-many alignment links. Given a source-target sentence pair (σ, τ) with alignment a, the (target) **span** of node v is the set of target words aligned to leaf nodes $yield(v)$ under node v:

$$span(v) \triangleq \{\tau_i \in \tau \mid \exists \sigma_j \in yield(v), (\sigma_j, \tau_i) \in a\}.$$

For example, in Figure 3, every node in the parse tree is annotated with its corresponding span below the node, where most nodes have contiguous spans except for the NP node which maps to a gapped phrase "Bush $\sqcup$ with Sharon". But contiguity alone is not enough to ensure well-formedness, since there might be words within the span aligned to source words uncovered by the node. So we also define a span s to be **faithful** to node v if every word in it is *only* aligned to nodes dominated by v, i.e.:

$$\forall \tau_i \in s, (\sigma_j, \tau_i) \in a \Rightarrow \sigma_j \in yield(v).$$

For example, sibling nodes VV and AS in the tree have non-faithful spans (crossed out in the Figure), because they both map to "held", thus *neither* of them can be translated to "held" alone. In this case, a larger tree fragment rooted at VPB has to be extracted. Nodes with non-empty, contiguous, *and* faithful spans form the **admissible set** (shaded nodes

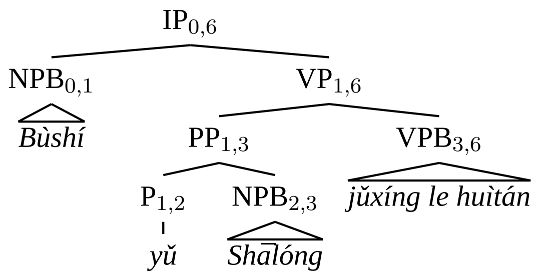


Figure 5: An alternative parse of the Chinese sentence, with *yǔ* as a preposition instead of a conjunction; common parts shared with 1-best parse in Fig. 3 are elided.

in the figure), which serve as potential cut-points for rule extraction.[3]

With the admissible set computed, rule extraction is as simple as a depth-first traversal from the root: we "cut" the tree at all admissible nodes to form tree fragments and extract a rule for each fragment, with variables matching the admissible descendant nodes. For example, the tree in Figure 3 is cut into 6 pieces, each of which corresponds to a rule on the right.

These extracted rules are called *minimal rules*, which can be glued together to form *composed rules* with larger tree fragments (e.g. r_1 in Fig. 1) (Galley et al., 2006). Our experiments use composed rules.

3 Forest-based Rule Extraction

We now extend tree-based extraction algorithm from the previous section to work with a packed forest representing exponentially many parse trees.

3.1 Packed Forest

Informally, a packed parse forest, or *forest* in short, is a compact representation of all the derivations (i.e., parse trees) for a given sentence under a context-free grammar (Earley, 1970; Billot and Lang, 1989). For example, consider again the Chinese sentence in Example (1) above, which has (at least) two readings depending on the part-of-speech of the word *yǔ*: it can be either a conjunction (CC "and") as shown in Figure 3, or a preposition (P "with") as shown in Figure 5, with only PP and VPB swapped from the English word order.

[3]Admissible set (Wang et al., 2007) is also known as "frontier set" (Galley et al., 2004). For simplicity of presentation, we assume every target word is aligned to at least one source word; see Galley et al. (2006) for handling unaligned target words.

These two parse trees can be represented as a single forest by sharing common subtrees such as NPB$_{0,1}$ and VPB$_{3,6}$, as shown in Figure 4. Such a forest has a structure of a *hypergraph* (Huang and Chiang, 2005), where items like NP$_{0,3}$ are called *nodes*, whose indices denote the source span, and combinations like

$$e_1 : \text{IP}_{0,6} \rightarrow \text{NPB}_{0,3}\ \text{VP}_{3,6}$$

we call *hyperedges*. We denote $head(e)$ and $tails(e)$ to be the consequent and antecedant items of hyperedge e, respectively. For example,

$$head(e_1) = \text{IP}_{0,6},\ \ tails(e_1) = \{\text{NPB}_{0,3}, \text{VP}_{3,6}\}.$$

We also denote $BS(v)$ to be the set of **incoming hyperedges** of node v, being different ways of deriving it. For example, in Figure 4, $BS(\text{IP}_{0,6}) = \{e_1, e_2\}$.

3.2 Forest-based Rule Extraction Algorithm

Like in tree-based extraction, we extract rules from a packed forest F in two steps:

(1) admissible set computation (where to cut), and

(2) fragmentation (how to cut).

It turns out that the exact formulation developed for admissible set in the tree-based case can be applied to a forest without any change. The fragmentation step, however, becomes much more involved since we now face a choice of multiple parse hyperedges at each node. In other words, it becomes *non-deterministic* how to "cut" a forest into tree fragments, which is analogous to the non-deterministic pattern-match in forest-based decoding (Mi et al., 2008). For example there are two parse hyperedges e_1 and e_2 at the root node in Figure 4. When we follow one of them to grow a fragment, there again will be multiple choices at each of its tail nodes. Like in tree-based case, a fragment is said to be **complete** if all its leaf nodes are admissible. Otherwise, an incomplete fragment can grow at any non-admissible frontier node v, where following each parse hyperedge at v will split off a new fragment. For example, following e_2 at the root node will immediately lead us to two admissible nodes, NPB$_{0,1}$ and VP$_{1,6}$ (we will highlight admissible nodes by gray shades

Algorithm 1 Forest-based Rule Extraction.

```
    Input: forest $F$, target sentence $\tau$, and alignment $a$
    Output: minimal rule set $\mathcal{R}$
 1: $admset \leftarrow$ ADMISSIBLE$(F, \tau, a)$        ▷ admissible set
 2: for each $v \in admset$ do
 3:     $open \leftarrow \varnothing$                   ▷ queue of active fragments
 4:     for each $e \in BS(v)$ do              ▷ incoming hyperedges
 5:         $front \leftarrow tails(e) \setminus admset$      ▷ initial frontier
 6:         $open$.append($\langle\{e\}, front\rangle$)
 7:     while $open \neq \varnothing$ do
 8:         $\langle frag, front\rangle \leftarrow open$.pop()    ▷ active fragment
 9:         if $front = \varnothing$ then
10:             generate a rule $r$ using fragment $frag$
11:             $\mathcal{R}$.append($r$)
12:         else                        ▷ incomplete: further expand
13:             $u \leftarrow front$.pop()                ▷ a frontier node
14:             for each $e \in BS(u)$ do
15:                 $front' \leftarrow front \cup (tails(e) \setminus admset)$
16:                 $open$.append($\langle frag \cup \{e\}, front'\rangle$)
```

in this section like in Figures 3 and 4). So this fragment, $frag_1 = \{e_2\}$, is now complete and we can extract a rule,

$$\textbf{IP}\ (x_1\text{:NPB}\ \ x_2\text{:VP}) \rightarrow x_1\ x_2.$$

However, following the other hyperedge e_1

$$\text{IP}_{0,6} \rightarrow \text{NP}_{0,3}\ \text{VPB}_{3,6}$$

will leave the new fragment $frag_2 = \{e_1\}$ *incomplete* with one non-admissible node NP$_{0,3}$. We then grow $frag_2$ at this node by choosing hyperedge e_3

$$\text{NP}_{0,3} \rightarrow \text{NPB}_{0,1}\ \text{CC}_{1,2}\ \text{NPB}_{2,3},$$

and spin off a new fragment $frag_3 = \{e_1, e_3\}$, which is now complete since all its four leaf nodes are admissible. We then extract a rule with four variables:

$$\textbf{IP}\ (\text{NP}(x_1\text{:NPB}\ x_2\text{:CC}\ x_3\text{:NPB})\ x_4\text{:VPB}) \rightarrow\ x_1\, x_4\, x_2\, x_3.$$

This procedure is formalized by a breadth-first search (BFS) in Pseudocode 1. The basic idea is to visit each frontier node v, and keep a queue $open$ of actively growing fragments rooted at v. We keep expanding incomplete fragments from $open$, and extract a rule if a complete fragment is found (line 10). Each fragment is associated with a *frontier* (variable *front* in the Pseudocode), being the subset of non-admissible leaf nodes (recall that expansion stops at admissible nodes). So each initial fragment along hyperedge e is associated with an initial frontier (line 5), $\mathit{front} = \mathit{tails}(e) \setminus \mathit{admset}$.

A fragment is complete if its frontier is empty (line 9), otherwise we pop one frontier node u to expand, spin off new fragments by following hyperedges of u, and update the frontier (lines 14-16), until all active fragments are complete and *open* queue is empty (line 7).

A single parse tree can also be viewed as a trivial forest, where each node has only one incoming hyperedge. So the Galley et al. (2004) algorithm for tree-based rule extraction (Sec. 2.2) can be considered a special case of our algorithm, where the queue *open* always contains one single active fragment.

3.3 Fractional Counts and Rule Probabilities

In tree-based extraction, for each sentence pair, each rule extracted naturally has a count of one, which will be used in maximum-likelihood estimation of rule probabilities. However, a forest is an implicit collection of many more trees, each of which, when enumerated, has its own probability accumulated from of the parse hyperedges involved. In other words, a forest can be viewed as a virtual weighted k-best list with a huge k. So a rule extracted from a non 1-best parse, i.e., using non 1-best hyperedges, should be penalized accordingly and should have a *fractional count* instead of a unit one, similar to the E-step in EM algorithms.

Inspired by the parsing literature on pruning (Charniak and Johnson, 2005; Huang, 2008) we penalize a rule r by the posterior probability of its tree fragment $\mathit{frag} = \mathit{lhs}(r)$. This posterior probability, notated $\alpha\beta(\mathit{frag})$, can be computed in an Inside-Outside fashion as the product of three parts: the outside probability of its root node, the probabilities of parse hyperedges involved in the fragment, and the inside probabilities of its leaf nodes,

$$\begin{aligned}\alpha\beta(\mathit{frag}) = &\alpha(\mathit{root}(\mathit{frag})) \\ &\cdot \prod_{e \in \mathit{frag}} \mathrm{P}(e) \\ &\cdot \prod_{v \in \mathit{yield}(\mathit{frag})} \beta(v)\end{aligned} \tag{2}$$

where $\alpha(\cdot)$ and $\beta(\cdot)$ denote the outside and inside probabilities of tree nodes, respectively. For example in Figure 4,

$$\begin{aligned}\alpha\beta(\{e_2, e_3\}) = \alpha(\mathrm{IP}_{0,6}) \;\cdot\; &\mathrm{P}(e_2) \cdot \mathrm{P}(e_3) \\ \cdot\, \beta(\mathrm{NPB}_{0,1})\beta(\mathrm{CC}_{1,2})&\beta(\mathrm{NPB}_{2,3})\beta(\mathrm{VPB}_{3,6}).\end{aligned}$$

Now the fractional count of rule r is simply

$$c(r) = \frac{\alpha\beta(\mathit{lhs}(r))}{\alpha\beta(\mathrm{TOP})} \tag{3}$$

where TOP denotes the root node of the forest.

Like in the M-step in EM algorithm, we now extend the maximum likelihood estimation to fractional counts for three conditional probabilities regarding a rule, which will be used in the experiments:

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

$$\mathrm{P}(r \mid \mathit{rhs}(r)) = \frac{c(r)}{\sum_{r':\mathit{rhs}(r')=\mathit{rhs}(r)} c(r')}, \tag{5}$$

$$\begin{aligned}&\mathrm{P}(r \mid \mathit{root}(\mathit{lhs}(r))) \\ &\quad = \frac{c(r)}{\sum_{r':\mathit{root}(\mathit{lhs}(r'))=\mathit{root}(\mathit{lhs}(r))} c(r')}.\end{aligned} \tag{6}$$

4 Related Work

The concept of *packed forest* has been previously used in translation rule extraction, for example in rule composition (Galley et al., 2006) and tree binarization (Wang et al., 2007). However, both of these efforts only use 1-best parses, with the second one packing different *binarizations* of the same tree in a forest. Nevertheless we suspect that their extraction algorithm is in principle similar to ours, although they do not provide details of forest-based fragmentation (Algorithm 1) which we think is non-trivial.

The forest concept is also used in machine translation decoding, for example to characterize the search space of decoding with integrated language models (Huang and Chiang, 2007). The first *direct* application of parse forest in translation is our previous work (Mi et al., 2008) which translates a packed forest from a parser; it is also the base system in our experiments (see below). This work, on the other hand, is in the orthogonal direction, where we utilize forests in rule extraction instead of decoding.

Our experiments will use both default 1-best decoding and forest-based decoding. As we will see in the next section, the best result comes when we combine the merits of both, i.e., using forests in both rule extraction and decoding.

There is also a parallel work on extracting rules from k-best parses and k-best alignments (Venugopal et al., 2008), but both their experiments and our own below confirm that extraction on k-best parses is neither efficient nor effective.

5 Experiments

5.1 System

Our experiments are on Chinese-to-English translation based on a tree-to-string system similar to (Huang et al., 2006; Liu et al., 2006). Given a 1-best tree T, the decoder searches for the best derivation d^* among the set of all possible derivations D:

$$\begin{aligned} d^* = \arg\max_{d \in D} \lambda_0 \log \mathrm{P}(d \mid T) + \lambda_1 \log \mathrm{P}_{\mathrm{lm}}(\tau(d)) \\ + \lambda_2 |d| + \lambda_3 |\tau(d)| \end{aligned} \tag{7}$$

where the first two terms are translation and language model probabilities, $\tau(d)$ is the target string (English sentence) for derivation d, and the last two terms are derivation and translation length penalties, respectively. The conditional probability $\mathrm{P}(d \mid T)$ decomposes into the product of rule probabilities:

$$\mathrm{P}(d \mid T) = \prod_{r \in d} \mathrm{P}(r). \tag{8}$$

Each $\mathrm{P}(r)$ is in turn a product of five probabilities:

$$\begin{aligned} \mathrm{P}(r) \; = & \mathrm{P}(r \mid \mathit{lhs}(r))^{\lambda_4} \cdot \mathrm{P}(r \mid \mathit{rhs}(r))^{\lambda_5} \\ & \cdot \mathrm{P}(r \mid \mathit{root}(\mathit{lhs}(r)))^{\lambda_6} \\ & \cdot \mathrm{P}_{\mathrm{lex}}(\mathit{lhs}(r) \mid \mathit{rhs}(r))^{\lambda_7} \\ & \cdot \mathrm{P}_{\mathrm{lex}}(\mathit{rhs}(r) \mid \mathit{lhs}(r))^{\lambda_8} \end{aligned} \tag{9}$$

where the first three are conditional probabilities based on fractional counts of rules defined in Section 3.3, and the last two are lexical probabilities. These parameters $\lambda_1 \ldots \lambda_8$ are tuned by minimum error rate training (Och, 2003) on the dev sets. We refer readers to Mi et al. (2008) for details of the decoding algorithm.

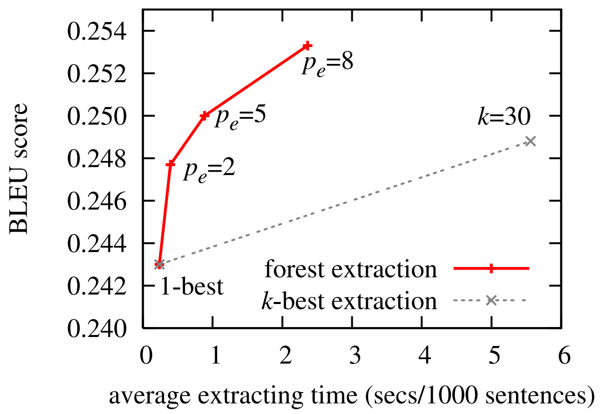


Figure 6: Comparison of extraction time and BLEU score: forest-based vs.1-best and 30-best.

rules from...	extraction	decoding	BLEU
1-best trees	0.24	1.74	0.2430
30-best trees	5.56	3.31	0.2488
forest: p_e=8	2.36	3.40	**0.2533**
Pharaoh	-	-	0.2297

Table 2: Results with different rule extraction methods. Extraction and decoding columns are running times in secs per 1000 sentences and per sentence, respectively.

We use the Chinese parser of Xiong et al. (2005) to parse the source side of the bitext. Following Huang (2008), we also modify this parser to output a packed forest for each sentence, which can be pruned by the marginal probability-based inside-outside algorithm (Charniak and Johnson, 2005; Huang, 2008). We will first report results trained on a small-scaled dataset with detailed analysis, and then scale to a larger one, where we also combine the technique of forest-based decoding (Mi et al., 2008).

5.2 Results and Analysis on Small Data

To test the effect of forest-based rule extraction, we parse the training set into parse forests and use three levels of pruning thresholds: $p_e = 2, 5, 8$.

Figure 6 plots the extraction speed and translation quality of forest-based extraction with various pruning thresholds, compared to 1-best and 30-best baselines. Using more than one parse tree apparently improves the BLEU score, but at the cost of much slower extraction, since each of the top-k trees has to be processed individually although they share many

rules from ...	total #	on dev	new rules used
1-best trees	440k	90k	-
30-best trees	1.2M	130k	8.71%
forest: p_e=8	3.3M	188k	16.3%

Table 3: Statistics of rules extracted from small data. The last column shows the ratio of *new* rules introduced by non 1-best parses being used in 1-best derivations.

extract. \ decoding	1-best tree	forest: p_d=10
1-best trees	0.2560	0.2674
30-best trees	0.2634	0.2767
forest: p_e=5	0.2679	**0.2816**
Hiero	0.2738	

Table 4: BLEU score results trained on large data.

common subtrees. Forest extraction, by contrast, is much faster thanks to packing and produces consistently better BLEU scores. With pruning threshold $p_e = 8$, forest-based extraction achieves a (case insensitive) BLEU score of 0.2533, which is an absolute improvement of 1.0% points over the 1-best baseline, and is statistically significant using the *sign-test* of Collins et al. (2005) ($p < 0.01$). This is also 0.5 points better than (and twice as fast as) extracting on 30-best parses. These BLEU score results are summarized in Table 2, which also shows that decoding with forest-extracted rules is less than twice as slow as with 1-best rules, and only fractionally slower than with 30-best rules.

We also investigate the question of how often rules extracted from non 1-best parses are used by the decoder. Table 3 shows the numbers of rules extracted from 1-best, 30-best and forest-based extractions, and the numbers that survive after filtering on the dev set. Basically in the forest-based case we can use about twice as many rules as in the 1-best case, or about 1.5 times of 30-best extraction. But the real question is, are these extra rules really useful in generating the final (1-best) translation? The last row shows that 16.3% of the rules used in 1-best derivations are indeed *only* extracted from non 1-best parses in the forests. Note that this is a stronger condition than changing the distribution of rules by considering more parses; here we introduce *new* rules never seen on any 1-best parses.

5.3 Final Results on Large Data

We also conduct experiments on a larger training dataset, FBIS, which contains 239K sentence pairs with about 6.9M/8.9M words in Chinese/English, respectively. We also use a bigger trigram model trained on the first 1/3 of the Xinhua portion of Gigaword corpus. To integrate with forest-based decoding, we use both 1-best trees and packed forests during both rule extraction and decoding phases. Since the data scale is larger than the small data, we are forced to use harsher pruning thresholds, with $p_e = 5$ for extraction and $p_d = 10$ for decoding.

The final BLEU score results are shown in Table 4. With both tree-based and forest-based decoding, rules extracted from forests significantly outperform those extracted from 1-best trees ($p < 0.01$). The final result with both forest-based extraction and forest-based decoding reaches a BLEU score of 0.2816, outperforming that of Hiero (Chiang, 2005), one of the best performing systems to date. These results confirm that our novel forest-based rule extraction approach is a promising direction for syntax-based machine translation.

6 Conclusion and Future Work

In this paper, we have presented a novel approach that extracts translation rules from a packed forest encoding exponentially many trees, rather than from 1-best or k-best parses. Experiments on a state-of-the-art tree-to-string system show that this method improves BLEU score significantly, with reasonable extraction speed. When combined with our previous work on forest-based decoding, the final result is even better than the hierarchical system Hiero. For future work we would like to apply this approach to other types of syntax-based translation systems, namely the string-to-tree systems (Galley et al., 2006) and tree-to-tree systems.

Acknowledgement

This work was funded by National Natural Science Foundation of China, Contracts 60736014 and 60573188, and 863 State Key Project No. 2006AA010108 (H. M.), and by NSF ITR EIA-0205456 (L. H.). We would also like to thank Qun Liu for supporting this work, and the three anonymous reviewers for improving the earlier version.

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 and Mark Johnson. 2005. Coarse-to-fine-grained n-best parsing and discriminative reranking. In *Proceedings of the 43rd ACL*, Ann Arbor, MI.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd ACL*, Ann Arbor, MI.

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

Yuan Ding and Martha Palmer. 2005. Machine translation using probablisitic synchronous dependency insertion grammars. In *Proceedings of the 43rd ACL*, Ann Arbor, MI.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

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

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*.

Liang Huang and David Chiang. 2005. Better k-best Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT-2005)*.

Liang Huang and David Chiang. 2007. Forest rescoring: Fast decoding with integrated language models. In *Proceedings of ACL*, Prague, Czech Rep., June.

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

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the ACL: HLT*, Columbus, OH, 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.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL: HLT*, Columbus, OH.

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

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of the 43rd ACL*, Ann Arbor, MI.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2008. Wider pipelines: N-best alignments and parses in mt training. In *Proceedings of AMTA*, Honolulu, Hawaii.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of EMNLP*, Prague, Czech Rep., July.

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.