Forest-Based Translation

Haitao Mi[†] **Liang Huang**[‡] **Qun Liu**[†]

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

[‡]Department of Computer & Information Science
University of Pennsylvania
Levine Hall, 3330 Walnut Street
Philadelphia, PA 19104, USA
lhuang3@cis.upenn.edu

Abstract

Among syntax-based translation models, the *tree-based* approach, which takes as input a parse tree of the source sentence, is a promising direction being faster and simpler than its string-based counterpart. However, current tree-based systems suffer from a major drawback: they only use the 1-best parse to direct the translation, which potentially introduces translation mistakes due to parsing errors. We propose a *forest-based* approach that translates a packed forest of exponentially many parses, which encodes many more alternatives than standard n-best lists. Large-scale experiments show an absolute improvement of 1.7 BLEU points over the 1-best baseline. This result is also 0.8 points higher than decoding with 30-best parses, and takes even less time.

1 Introduction

Syntax-based machine translation has witnessed promising improvements in recent years. Depending on the type of input, these efforts can be divided into two broad categories: the *string-based* systems whose input is a string to be simultaneously parsed and translated by a synchronous grammar (Wu, 1997; Chiang, 2005; Galley et al., 2006), and the *tree-based* systems whose input is already a parse tree to be directly converted into a target tree or string (Lin, 2004; Ding and Palmer, 2005; Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006). Compared with their string-based counterparts, tree-based systems offer some attractive features: they 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), and can have separate grammars for parsing and translation, say, a context-free grammar for the former and a tree substitution grammar for the latter (Huang et al., 2006). However, despite these advantages, current tree-based systems suffer from a major drawback: they only use the 1-best parse tree to direct the translation, which potentially introduces translation mistakes due to parsing errors (Quirk and Corston-Oliver, 2006). This situation becomes worse with resource-poor source languages without enough Treebank data to train a high-accuracy parser.

One obvious solution to this problem is to take as input k-best parses, instead of a single tree. This k-best list postpones some disambiguation to the decoder, which may recover from parsing errors by getting a better translation from a non 1-best parse. However, a k-best list, with its limited scope, often has too few variations and too many redundancies; for example, a 50-best list typically encodes a combination of 5 or 6 binary ambiguities (since $2^5 < 50 < 2^6$), and many subtrees are repeated across different parses (Huang, 2008). It is thus inefficient either to decode separately with each of these very similar trees. Longer sentences will also aggravate this situation as the number of parses grows exponentially with the sentence length.

We instead propose a new approach, *forest-based translation* (Section 3), where the decoder translates a *packed forest* of exponentially many parses,[1]

[1]There has been some confusion in the MT literature regarding the term *forest*: the word "forest" in "forest-to-string rules"

Proceedings of ACL-08: HLT, pages 192–199,
Columbus, Ohio, USA, June 2008.

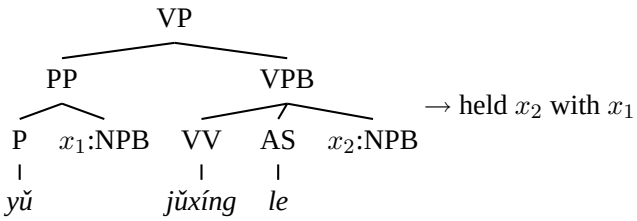


Figure 1: An example translation rule (r_3 in Fig. 2).

which compactly encodes many more alternatives than k-best parses. This scheme can be seen as a compromise between the string-based and tree-based methods, while combining the advantages of both: decoding is still fast, yet does not commit to a single parse. Large-scale experiments (Section 4) show an improvement of 1.7 BLEU points over the 1-best baseline, which is also 0.8 points higher than decoding with 30-best trees, and takes even less time thanks to the sharing of common subtrees.

2 Tree-based systems

Current *tree-based* systems perform translation in two separate steps: parsing and decoding. A parser first parses the source language input into a 1-best tree T, and the decoder then searches for the best *derivation* (a sequence of translation steps) d^* that converts source tree T into a target-language string among all possible derivations D:

$$d^* = \arg\max_{d \in D} \mathrm{P}(d|T). \quad (1)$$

We will now proceed with a running example translating from Chinese to English:

(2) 布什 与 沙龙 举行 了 会谈
Bùshí yǔ Shālóng jǔxíng le huìtán
Bush with/and Sharon$_1$ hold *pass.* talk$_2$
"Bush held a talk$_2$ with Sharon$_1$"

Figure 2 shows how this process works. The Chinese sentence (a) is first parsed into tree (b), which will be converted into an English string in 5 steps. First, at the root node, we apply rule r_1 preserving top-level word-order between English and Chinese,

(r_1) IP(x_1:NPB x_2:VP) → x_1 x_2

(Liu et al., 2007) was a misnomer which actually refers to a set of several unrelated subtrees over disjoint spans, and should not be confused with the standard concept of *packed forest*.

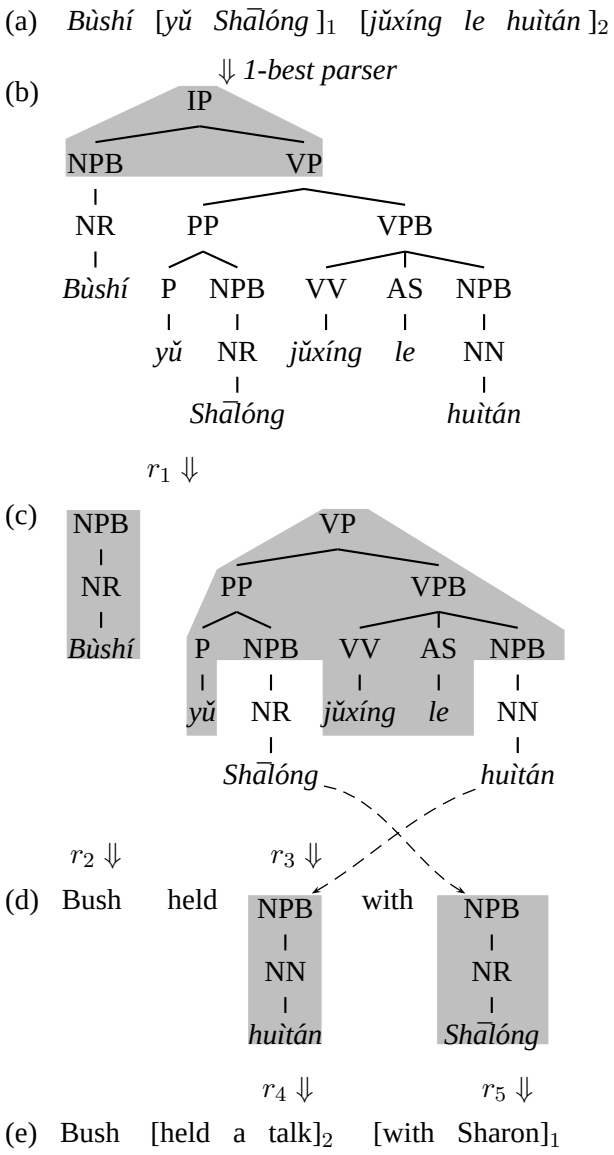


Figure 2: An example derivation of tree-to-string translation. Shaded regions denote parts of the tree that is pattern-matched with the rule being applied.

which results in two unfinished subtrees in (c). Then rule r_2 grabs the *Bùshí* subtree and transliterate it

(r_2) NPB(NR(*Bùshí*)) → Bush.

Similarly, rule r_3 shown in Figure 1 is applied to the VP subtree, which swaps the two NPBs, yielding the situation in (d). This rule is particularly interesting since it has multiple levels on the source side, which has more expressive power than synchronous context-free grammars where rules are flat.

More formally, a (tree-to-string) **translation rule** (Huang et al., 2006) is a tuple $\langle t, s, \phi \rangle$, where t is the source-side tree, whose internal nodes are labeled by nonterminal symbols in N, and whose frontier nodes are labeled by source-side terminals in Σ or variables from a set $\mathcal{X} = \{x_1, x_2, \ldots\}$; $s \in (\mathcal{X} \cup \Delta)^*$ is the target-side string where Δ is the target language terminal set; and ϕ is a mapping from $\mathcal{X}$ to nonterminals in N. Each variable $x_i \in \mathcal{X}$ occurs *exactly once* in t and *exactly once* in s. We denote $\mathcal{R}$ to be the translation rule set. A similar formalism appears in another form in (Liu et al., 2006). These rules are in the reverse direction of the original string-to-tree transducer rules defined by Galley et al. (2004).

Finally, from step (d) we apply rules r_4 and r_5

(r_4) NPB(NN(*huìtán*)) $\rightarrow$ a talk

(r_5) NPB(NR(*Shālóng*)) $\rightarrow$ Sharon

which perform phrasal translations for the two remaining subtrees, respectively, and get the Chinese translation in (e).

3 Forest-based translation

We now extend the tree-based idea from the previous section to the case of forest-based translation. Again, there are two steps, parsing and decoding. In the former, a (modified) parser will parse the input sentence and output a packed forest (Section 3.1) rather than just the 1-best tree. Such a forest is usually huge in size, so we use the *forest pruning algorithm* (Section 3.4) to reduce it to a reasonable size. The pruned parse forest will then be used to direct the translation.

In the decoding step, we first convert the parse forest into a *translation forest* using the translation rule set, by similar techniques of pattern-matching from tree-based decoding (Section 3.2). Then the decoder searches for the best derivation on the translation forest and outputs the target string (Section 3.3).

3.1 Parse 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 (Billot and Lang, 1989). For example, consider the Chinese sentence in Example (2) above, which has (at least) two readings depending on the part-of-speech of the word *yǔ*, which can be either a preposition (P "with") or a conjunction (CC "and"). The parse tree for the preposition case is shown in Figure 2(b) as the 1-best parse, while for the conjunction case, the two proper nouns (*Bùshí* and *Shālóng*) are combined to form a coordinated NP

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

which functions as the subject of the sentence. In this case the Chinese sentence is translated into

(3) " [Bush and Sharon] held a talk".

Shown in Figure 3(a), these two parse trees can be represented as a single forest by sharing common subtrees such as $\text{NPB}_{0,1}$ and $\text{VPB}_{3,6}$. Such a forest has a structure of a *hypergraph* (Klein and Manning, 2001; Huang and Chiang, 2005), where items like $\text{NP}_{0,3}$ are called *nodes*, and deductive steps like (*) correspond to *hyperedges*.

More formally, a **forest** is a pair $\langle V, E \rangle$, where V is the set of **nodes**, and E the set of **hyperedges**. For a given sentence $w_{1:l} = w_1 \ldots w_l$, each node $v \in V$ 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, $w_{i+1} \ldots w_j$). Each hyperedge $e \in E$ is a pair $\langle tails(e), head(e) \rangle$, where $head(e) \in V$ is the *consequent node* in the deductive step, and $tails(e) \in V^*$ is the list of *antecedent nodes*. For example, the hyperedge for deduction (*) is notated:

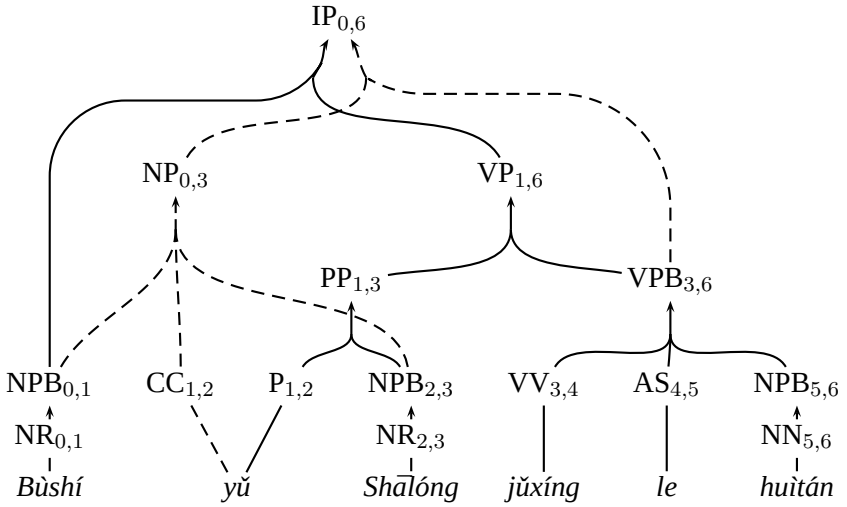
$$\langle (\text{NPB}_{0,1},\ \text{CC}_{1,2},\ \text{NPB}_{2,3}),\ \ \text{NP}_{0,3} \rangle.$$

There is also a distinguished **root node** TOP in each forest, denoting the goal item in parsing, which is simply $\text{S}_{0,l}$ where S is the start symbol and l is the sentence length.

3.2 Translation Forest

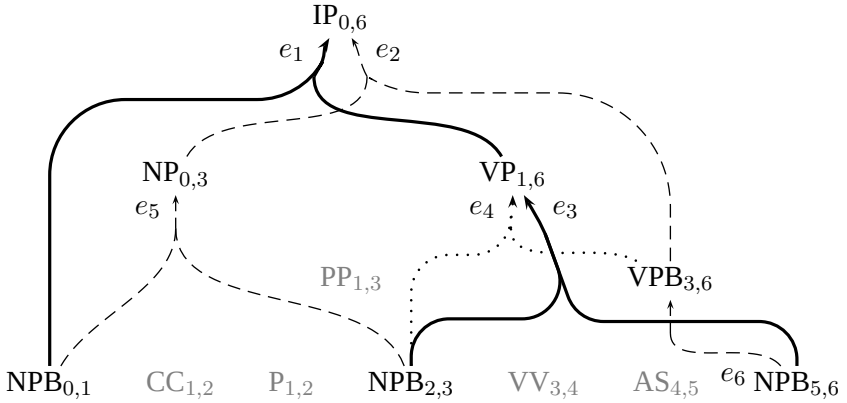
Given a parse forest and a translation rule set $\mathcal{R}$, we can generate a *translation forest* which has a similar hypergraph structure. Basically, just as the depth-first traversal procedure in tree-based decoding (Figure 2), we visit in top-down order each node v in the

(a)



⇓ translation rule set $\mathcal{R}$

(b)



(c)

translation hyperedge		translation rule
e_1	r_1	IP(x_1:NPB x_2:VP) → x_1 x_2
e_2	r_6	IP(x_1:NP x_2:VPB) → x_1 x_2
e_3	r_3	VP(PP(P(*yǔ*) x_1:NPB) VPB(VV(*jǔxíng*) AS(*le*) x_2:NPB)) → held x_2 with x_1
e_4	r_7	VP(PP(P(*yǔ*) x_1:NPB) x_2:VPB) → x_2 with x_1
e_5	r_8	NP(x_1:NPB CC(*yǔ*) x_2:NPB) → x_1 and x_2
e_6	r_9	VPB(VV(*jǔxíng*) AS(*le*) x_1:NPB) → held x_1

Figure 3: (a) the parse forest of the example sentence; solid hyperedges denote the 1-best parse in Figure 2(b) while dashed hyperedges denote the alternative parse due to Deduction (*). (b) the corresponding translation forest after applying the translation rules (lexical rules not shown); the derivation shown in bold solid lines (e_1 and e_3) corresponds to the derivation in Figure 2; the one shown in dashed lines (e_2, e_5, and e_6) uses the alternative parse and corresponds to the translation in Example (3). (c) the correspondence between translation hyperedges and translation rules.

parse forest, and try to pattern-match each translation rule r against the local sub-forest under node v. For example, in Figure 3(a), at node $\text{VP}_{1,6}$, two rules r_3 and r_7 both matches the local subforest, and will thus generate two *translation hyperedges* e_3 and e_4 (see Figure 3(b-c)).

More formally, we define a function $match(r, v)$ which attempts to pattern-match rule r at node v in the parse forest, and in case of success, returns a list of descendent nodes of v that are matched to the variables in r, or returns an empty list if the match fails. Note that this procedure is recursive and may

Pseudocode 1 The conversion algorithm.

```
1: Input: parse forest H_p and rule set R
2: Output: translation forest H_t
3: for each node v ∈ V_p in top-down order do
4:     for each translation rule r ∈ R do
5:         vars ← match(r, v)                ▷ variables
6:         if vars is not empty then
7:             e ← ⟨vars, v, s(r)⟩
8:             add translation hyperedge e to H_t
```

involve multiple parse hyperedges. For example,

$$match(r_3, \text{VP}_{1,6}) = (\text{NPB}_{2,3},\ \text{NPB}_{5,6}),$$

which covers three parse hyperedges, while nodes in gray do not pattern-match any rule (although they are involved in the matching of other nodes, where they match *interior nodes* of the source-side tree fragments in a rule). We can thus construct a translation hyperedge from $match(r, v)$ to v for each node v and rule r. In addition, we also need to keep track of the *target string* $s(r)$ specified by rule r, which includes target-language terminals and variables. For example, $s(r_3) =$ "held x_2 with x_1". The subtranslations of the matched variable nodes will be substituted for the variables in $s(r)$ to get a complete translation for node v. So a translation hyperedge e is a triple $\langle tails(e), head(e), s\rangle$ where s is the target string from the rule, for example,

$$e_3 = \langle(\text{NPB}_{2,3},\ \text{NPB}_{5,6}), \text{VP}_{1,6}, \text{"held } x_2 \text{ with } x_1\text{"}\rangle.$$

This procedure is summarized in Pseudocode 1.

3.3 Decoding Algorithms

The decoder performs two tasks on the translation forest: 1-best search with integrated language model (LM), and k-best search with LM to be used in minimum error rate training. Both tasks can be done efficiently by forest-based algorithms based on k-best parsing (Huang and Chiang, 2005).

For 1-best search, we use the *cube pruning* technique (Chiang, 2007; Huang and Chiang, 2007) which approximately intersects the translation forest with the LM. Basically, cube pruning works bottom up in a forest, keeping at most k +LM items at each node, and uses the best-first expansion idea from the Algorithm 2 of Huang and Chiang (2005) to speed up the computation. An +LM item of node v has the form $(v^{a \star b})$, where a and b are the target-language *boundary words*. For example, $(\text{VP}_{1,6}^{\text{held} \star \text{Sharon}})$ is an +LM item with its translation starting with "held" and ending with "Sharon". This scheme can be easily extended to work with a general n-gram by storing $n-1$ words at both ends (Chiang, 2007).

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. However, this time we work on a finer-grained forest, called *translation+LM* forest, resulting from the intersection of the translation forest and the LM, with its nodes being the +LM items during cube pruning. Although this new forest is prohibitively large, Algorithm 3 is very efficient with minimal overhead on top of 1-best.

3.4 Forest Pruning Algorithm

We use the pruning algorithm of (Jonathan Graehl, p.c.; Huang, 2008) that is very similar to the method based on marginal probability (Charniak and Johnson, 2005), except that it prunes hyperedges as well as nodes. Basically, we use an Inside-Outside algorithm to compute the Viterbi inside cost $\beta(v)$ and the Viterbi outside cost $\alpha(v)$ for each node v, and then compute the **merit** $\alpha\beta(e)$ for each hyperedge:

$$\alpha\beta(e) = \alpha(head(e)) + \sum_{u_i \in tails(e)} \beta(u_i) \quad (4)$$

Intuitively, this merit is the cost of the best derivation that traverses e, and the difference $\delta(e) = \alpha\beta(e) - \beta(\text{TOP})$ can be seen as the distance away from the globally best derivation. We prune away a hyperedge e if $\delta(e) > p$ for a threshold p. Nodes with all incoming hyperedges pruned are also pruned.

4 Experiments

We can extend the simple model in Equation 1 to a log-linear one (Liu et al., 2006; Huang et al., 2006):

$$d^* = \arg\max_{d \in D} \text{P}(d \mid T)^{\lambda_0} \cdot e^{\lambda_1 |d|} \cdot \text{P}_{\text{lm}}(s)^{\lambda_2} \cdot e^{\lambda_3 |s|} \quad (5)$$

where T is the 1-best parse, $e^{\lambda_1 |d|}$ is the penalty term on the number of rules in a derivation, $\text{P}_{\text{lm}}(s)$ is the language model and $e^{\lambda_3 |s|}$ is the length penalty term

on target translation. The derivation probability conditioned on 1-best tree, $\mathrm{P}(d \mid T)$, should now be replaced by $\mathrm{P}(d \mid H_p)$ where H_p is the parse forest, which decomposes into the product of probabilities of translation rules $r \in d$:

$$\mathrm{P}(d \mid H_p) = \prod_{r \in d} \mathrm{P}(r) \quad (6)$$

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

$$\mathrm{P}(r) = \mathrm{P}(t \mid s)^{\lambda_4} \cdot \mathrm{P}_{\mathrm{lex}}(t \mid s)^{\lambda_5} \cdot \mathrm{P}(s \mid t)^{\lambda_6} \cdot \mathrm{P}_{\mathrm{lex}}(s \mid t)^{\lambda_7} \cdot \mathrm{P}(t \mid H_p)^{\lambda_8}. \quad (7)$$

Here t and s are the source-side tree and target-side string of rule r, respectively, $\mathrm{P}(t \mid s)$ and $\mathrm{P}(s \mid t)$ are the two translation probabilities, and $\mathrm{P}_{\mathrm{lex}}(\cdot)$ are the lexical probabilities. The only extra term in forest-based decoding is $\mathrm{P}(t \mid H_p)$ denoting the source side parsing probability of the current translation rule r in the parse forest, which is the product of probabilities of each parse hyperedge e_p covered in the pattern-match of t against H_p (which can be recorded at conversion time):

$$\mathrm{P}(t \mid H_p) = \prod_{e_p \in H_p,\ e_p \text{ covered by } t} \mathrm{P}(e_p). \quad (8)$$

4.1 Data preparation

Our experiments are on Chinese-to-English translation, and we use the Chinese parser of Xiong et al. (2005) to parse the source side of the bitext. Following Huang (2008), we modify the parser to output a packed forest for each sentence.

Our training corpus consists of 31,011 sentence pairs with 0.8M Chinese words and 0.9M English words. We first word-align them by GIZA++ refined by "diagand" from Koehn et al. (2003), and apply the tree-to-string rule extraction algorithm (Galley et al., 2006; Liu et al., 2006), which resulted in 346K translation rules. Note that our rule extraction is still done on 1-best parses, while decoding is on k-best parses or packed forests. We also use the SRI Language Modeling Toolkit (Stolcke, 2002) to train a trigram language model with Kneser-Ney smoothing on the English side of the bitext.

We use the 2002 NIST MT Evaluation test set as our development set (878 sentences) and the 2005 NIST MT Evaluation test set as our test set (1082 sentences), with on average 28.28 and 26.31 words per sentence, respectively. We evaluate the translation quality using the *case-sensitive* BLEU-4 metric (Papineni et al., 2002). We use the standard minimum error-rate training (Och, 2003) to tune the feature weights to maximize the system's BLEU score on the dev set. On dev and test sets, we prune the Chinese parse forests by the forest pruning algorithm in Section 3.4 with a threshold of $p = 12$, and then convert them into translation forests using the algorithm in Section 3.2. To increase the coverage of the rule set, we also introduce a *default translation hyperedge* for each parse hyperedge by monotonically translating each tail node, so that we can always at least get a complete translation in the end.

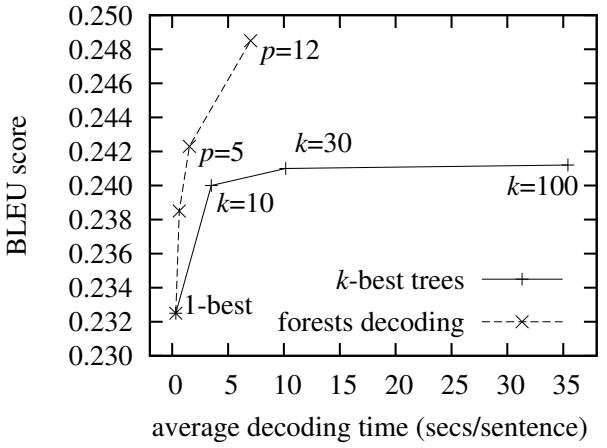


Figure 4: Comparison of decoding on forests with decoding on k-best trees.

4.2 Results

The BLEU score of the baseline 1-best decoding is 0.2325, which is consistent with the result of 0.2302 in (Liu et al., 2007) on the same training, development and test sets, and with the same rule extraction procedure. The corresponding BLEU score of Pharaoh (Koehn, 2004) is 0.2182 on this dataset.

Figure 4 compares forest decoding with decoding on k-best trees in terms of speed and quality. Using more than one parse tree apparently improves the BLEU score, but at the cost of much slower decoding, since each of the top-k trees has to be decoded individually although they share many common subtrees. Forest decoding, by contrast, is much faster

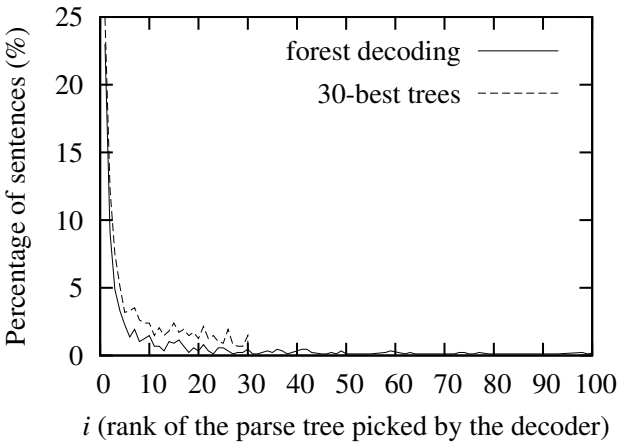


Figure 5: Percentage of the i-th best parse tree being picked in decoding. 32% of the distribution for forest decoding is beyond top-100 and is not shown on this plot.

and produces consistently better BLEU scores. With pruning threshold $p = 12$, it achieved a BLEU score of 0.2485, which is an absolute improvement of 1.6% points over the 1-best baseline, and is statistically significant using the *sign-test* of Collins et al. (2005) ($p < 0.01$).

We also investigate the question of how often the ith-best parse tree is picked to direct the translation ($i = 1, 2, \ldots$), in both k-best and forest decoding schemes. A packed forest can be roughly viewed as a (virtual) ∞-best list, and we can thus ask how often is a parse beyond top-k used by a forest, which relates to the fundamental limitation of k-best lists. Figure 5 shows that, the 1-best parse is still preferred 25% of the time among 30-best trees, and 23% of the time by the forest decoder. These ratios decrease dramatically as i increases, but the forest curve has a much longer tail in large i. Indeed, 40% of the trees preferred by a forest is beyond top-30, 32% is beyond top-100, and even 20% beyond top-1000. This confirms the fact that we need exponentially large k-best lists with the explosion of alternatives, whereas a forest can encode these information compactly.

4.3 Scaling to large data

We also conduct experiments on a larger dataset, which contains 2.2M training sentence pairs. Besides the trigram language model trained on the English side of these bitext, we also use another trigram model trained on the first 1/3 of the Xinhua portion of Gigaword corpus. The two LMs have distinct weights tuned by minimum error rate training. The dev and test sets remain the same as above.

approach \ ruleset	TR	TR+BP
1-best tree	0.2666	0.2939
30-best trees	0.2755	0.3084
forest ($p = 12$)	**0.2839**	**0.3149**

Table 1: BLEU score results from training on large data.

Furthermore, we also make use of bilingual phrases to improve the coverage of the ruleset. Following Liu et al. (2006), we prepare a phrase-table from a phrase-extractor, e.g. Pharaoh, and at decoding time, for each node, we construct on-the-fly flat translation rules from phrases that match the source-side span of the node. These phrases are called *syntactic phrases* which are consistent with syntactic constituents (Chiang, 2005), and have been shown to be helpful in tree-based systems (Galley et al., 2006; Liu et al., 2006).

The final results are shown in Table 1, where TR denotes translation rule only, and TR+BP denotes the inclusion of bilingual phrases. The BLEU score of forest decoder with TR is 0.2839, which is a 1.7% points improvement over the 1-best baseline, and this difference is statistically significant ($p < 0.01$). Using bilingual phrases further improves the BLEU score by 3.1% points, which is 2.1% points higher than the respective 1-best baseline. We suspect this larger improvement is due to the alternative constituents in the forest, which activates many syntactic phrases suppressed by the 1-best parse.

5 Conclusion and future work

We have presented a novel forest-based translation approach which uses a packed forest rather than the 1-best parse tree (or k-best parse trees) to direct the translation. Forest provides a compact data-structure for efficient handling of exponentially many tree structures, and is shown to be a promising direction with state-of-the-art translation results and reasonable decoding speed. This work can thus be viewed as a compromise between string-based and tree-based paradigms, with a good trade-off between speed and accuarcy. For future work, we would like to use packed forests not only in decoding, but also for translation rule extraction during training.

Acknowledgement

Part of this work was done while L. H. was visiting CAS/ICT. The authors were supported by National Natural Science Foundation of China, Contracts 60736014 and 60573188, and 863 State Key Project No. 2006AA010108 (H. M and Q. L.), and by NSF ITR EIA-0205456 (L. H.). We would also like to thank Chris Quirk for inspirations, Yang Liu for help with rule extraction, Mark Johnson for posing the question of virtual ∞-best list, and the anonymous reviewers for 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 and Mark Johnson. 2005. Coarse-to-fine-grained n-best parsing and discriminative reranking. In *Proceedings of the 43rd ACL*.

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, Ann Arbor, Michigan, June.

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

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

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, Sydney, Australia, July.

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

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

Dan Klein and Christopher D. Manning. 2001. Parsing and Hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001), 17-19 October 2001, Beijing, China*.

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

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*, pages 115–124.

Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of the 20th COLING*, Barcelona, Spain.

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, Prague, Czech Republic, June.

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 and Simon Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *Proceedings of EMNLP*.

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

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

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

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, Jeju Island, South Korea.

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