Binarized Forest to String Translation

Hao Zhang
Google Research
haozhang@google.com

Licheng Fang
Computer Science Department
University of Rochester
lfang@cs.rochester.edu

Peng Xu
Google Research
xp@google.com

Xiaoyun Wu
Google Research
xiaoyunwu@google.com

Abstract

Tree-to-string translation is syntax-aware and efficient but sensitive to parsing errors. Forest-to-string translation approaches mitigate the risk of propagating parser errors into translation errors by considering a forest of alternative trees, as generated by a source language parser. We propose an alternative approach to generating forests that is based on combining sub-trees within the first best parse through binarization. Provably, our binarization forest can cover any non-consitituent phrases in a sentence but maintains the desirable property that for each span there is at most one nonterminal so that the grammar constant for decoding is relatively small. For the purpose of reducing search errors, we apply the synchronous binarization technique to forest-to-string decoding. Combining the two techniques, we show that using a fast shift-reduce parser we can achieve significant quality gains in NIST 2008 English-to-Chinese track (1.3 BLEU points over a phrase-based system, 0.8 BLEU points over a hierarchical phrase-based system). Consistent and significant gains are also shown in WMT 2010 in the English to German, French, Spanish and Czech tracks.

1 Introduction

In recent years, researchers have explored a wide spectrum of approaches to incorporate syntax and structure into machine translation models. The unifying framework for these models is *synchronous grammars* (Chiang, 2005) or *tree transducers* (Graehl and Knight, 2004). Depending on whether or not monolingual parsing is carried out on the source side or the target side for inference, there are four general categories within the framework:

- *string-to-string* (Chiang, 2005; Zollmann and Venugopal, 2006)
- *string-to-tree* (Galley et al., 2006; Shen et al., 2008)
- *tree-to-string* (Lin, 2004; Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006; Mi et al., 2008)
- *tree-to-tree* (Eisner, 2003; Zhang et al., 2008)

In terms of search, the string-to-x models explore all possible source parses and map them to the target side, while the tree-to-x models search over the subspace of structures of the source side constrained by an input tree or trees. Hence, tree-to-x models are more constrained but more efficient. Models such as Huang et al. (2006) can match multi-level tree fragments on the source side which means larger contexts are taken into account for translation (Poutsma, 2000), which is a modeling advantage. To balance efficiency and accuracy, forest-to-string models (Mi et al., 2008; Mi and Huang, 2008) use a compact representation of exponentially many trees to improve tree-to-string models. Traditionally, such forests are obtained through hyper-edge pruning in the k-best search space of a monolingual parser (Huang, 2008). The pruning parameters that control the size of forests are normally hand-tuned. Such forests encode both syntactic variants and structural variants. By syntactic variants, we refer to the fact that a parser can parse a substring into either a noun phrase or verb phrase in certain cases.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 835–845,
Portland, Oregon, June 19-24, 2011. ©2011 Association for Computational Linguistics

We believe that structural variants which allow more source spans to be explored during translation are more important (DeNeefe et al., 2007), while syntactic variants might improve word sense disambiguation but also introduce more spurious ambiguities (Chiang, 2005) during decoding. To focus on structural variants, we propose a family of binarization algorithms to expand one single constituent tree into a packed forest of binary trees containing combinations of adjacent tree nodes. We control the freedom of tree node binary combination by restricting the distance to the lowest common ancestor of two tree nodes. We show that the best results are achieved when the distance is two, i.e., when combining tree nodes sharing a common grand-parent. In contrast to conventional parser-produced-forest-to-string models, in our model:

- Forests are not generated by a parser but by combining sub-structures using a tree binarizer.
- Instead of using arbitary pruning parameters, we control forest size by an integer number that defines the degree of tree structure violation.
- There is at most one nonterminal per span so that the grammar constant is small.

Since GHKM rules (Galley et al., 2004) can cover multi-level tree fragments, a synchronous grammar extracted using the GHKM algorithm can have synchronous translation rules with more than two nonterminals regardless of the branching factor of the source trees. For the first time, we show that similar to string-to-tree decoding, synchronous binarization significantly reduces search errors and improves translation quality for forest-to-string decoding.

To summarize, the whole pipeline is as follows. First, a parser produces the highest-scored tree for an input sentence. Second, the parse tree is restructured using our binarization algorithm, resulting in a binary packed forest. Third, we apply the forest-based variant of the GHKM algorithm (Mi and Huang, 2008) on the new forest for rule extraction. Fourth, on the translation forest generated by all applicable translation rules, which is not necessarily binary, we apply the synchronous binarization algorithm (Zhang et al., 2006) to generate a binary translation forest. Finally, we use a bottom-up decoding algorithm with intergrated LM intersection using the cube pruning technique (Chiang, 2005).

The rest of the paper is organized as follows. In Section 2, we give an overview of the forest-to-string models. In Section 2.1, we introduce a more efficient and flexible algorithm for extracting composed GHKM rules based on the same principle as cube pruning (Chiang, 2007). In Section 3, we introduce our source tree binarization algorithm for producing binarized forests. In Section 4, we explain how to do synchronous rule factorization in a forest-to-string decoder. Experimental results are in Section 5.

2 Forest-to-string Translation

Forest-to-string models can be described as

$$e = \mathcal{Y}(\underset{d \in D(T),\, T \in F(f)}{\arg\max} P(d|T)) \qquad (1)$$

where f stands for a source string, e stands for a target string, F stands for a forest, D stands for a set of synchronous derivations on a given tree T, and $\mathcal{Y}$ stands for the target side yield of a derivation. The search problem is finding the derivation with the highest probability in the space of all derivations for all parse trees for an input sentence. The log probability of a derivation is normally a linear combination of local features which enables dynamic programming to find the optimal combination efficiently. In this paper, we focus on the models based on the Synchronous Tree Substitution Grammars (STSG) defined by Galley et al. (2004). In contrast to a tree-to-string model, the introduction of F augments the search space systematically. When the first-best parse is wrong or no good translation rules are applicable to the first-best parse, the model can recover good translations from alternative parses.

In STSG, local features are defined on tree-to-string rules, which are synchronous grammar rules defining how a sequence of terminals and nonterminals on the source side translates to a sequence of target terminals and nonterminals. One-to-one mapping of nonterminals is assumed. But terminals do not necessarily need to be aligned. Figure 1 shows a typical English-Chinese tree-to-string rule with a reordering pattern consisting of two nonterminals and different numbers of terminals on the two sides.

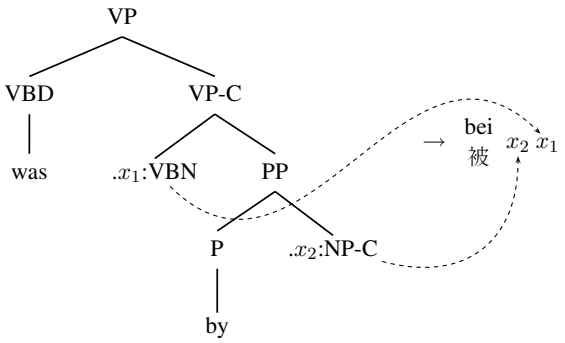


Figure 1: An example tree-to-string rule.

Forest-to-string translation has two stages. The first stage is rule extraction on word-aligned parallel texts with source forests. The second stage is rule enumeration and DP decoding on forests of input strings. In both stages, at each tree node, the task on the source side is to generate a list of tree fragments by composing the tree fragments of its children. We propose a cube-pruning style algorithm that is suitable for both rule extraction during training and rule enumeration during decoding.

At the highest level, our algorithm involves three steps. In the first step, we label each node in the input forest by a boolean variable indicating whether it is a site of interest for tree fragment generation. If it is marked true, it is an *admissible* node. In the case of rule extraction, a node is admissible if and only if it corresponds to a phrase pair according to the underlying word alignment. In the case of decoding, every node is admissible for the sake of completeness of search. An initial one-node tree fragment is placed at each admissible node for seeding the tree fragment generation process. In the second step, we do cube-pruning style bottom-up combinations to enumerate a pruned list of tree fragments at each tree node. In the third step, we extract or enumerate-and-match tree-to-string rules for the tree fragments at the admissible nodes.

2.1 A Cube-pruning-inspired Algorithm for Tree Fragment Composition

Galley et al. (2004) defined minimal tree-to-string rules. Galley et al. (2006) showed that tree-to-string rules made by composing smaller ones are important to translation. It can be understood by the analogy of going from word-based models to phrase-based models. We relate composed rule extraction to cube-pruning (Chiang, 2007). In cube-pruning, the process is to keep track of the k-best sorted language model states at each node and combine them bottom-up with the help of a priority queue. We can imagine substituting k-best LM states with k composed rules at each node and composing them bottom-up. We can also borrow the cube pruning trick to compose multiple lists of rules using a priority queue to lazily explore the space of combinations starting from the top-most element in the cube formed by the lists.

We need to define a ranking function for composed rules. To simulate the breadth-first expansion heuristics of Galley et al. (2006), we define the figure of merit of a tree-to-string rule as a tuple $m = (h, s, t)$, where h is the height of a tree fragment, s is the number of frontier nodes, i.e., bottom-level nodes including both terminals and non-terminals, and t is the number of terminals in the set of frontier nodes. We define an additive operator $+$:

$$m_1 + m_2 = (\ \max\{h_1, h_2\} + 1,\ s_1 + s_2,\ t_1 + t_2\ )$$

and a $\min$ operator based on the order $<$:

$$m_1 < m_2 \iff \begin{cases} h_1 < h_2 \vee \\ h_1 = h_2 \ \wedge\ s_1 < s_2 \vee \\ h_1 = h_2 \ \wedge\ s_1 = s_2 \ \wedge\ t_1 < t_2 \end{cases}$$

The $+$ operator corresponds to rule compositions. The $<$ operator corresponds to ranking rules by their sizes. A concrete example is shown in Figure 2, in which case the monotonicity property of $(+, <)$ holds: if $m_a < m_b$, $m_a + m_c < m_b + m_c$. However, this is not true in general for the operators in our definition, which implies that our algorithm is indeed like cube-pruning: an approximate k-shortest-path algorithm.

3 Source Tree Binarization

The motivation of tree binarization is to factorize large and rare structures into smaller but frequent ones to improve generalization. For example, Penn Treebank annotations are often flat at the phrase level. Translation rules involving flat phrases are unlikely to generalize. If long sequences are binarized,

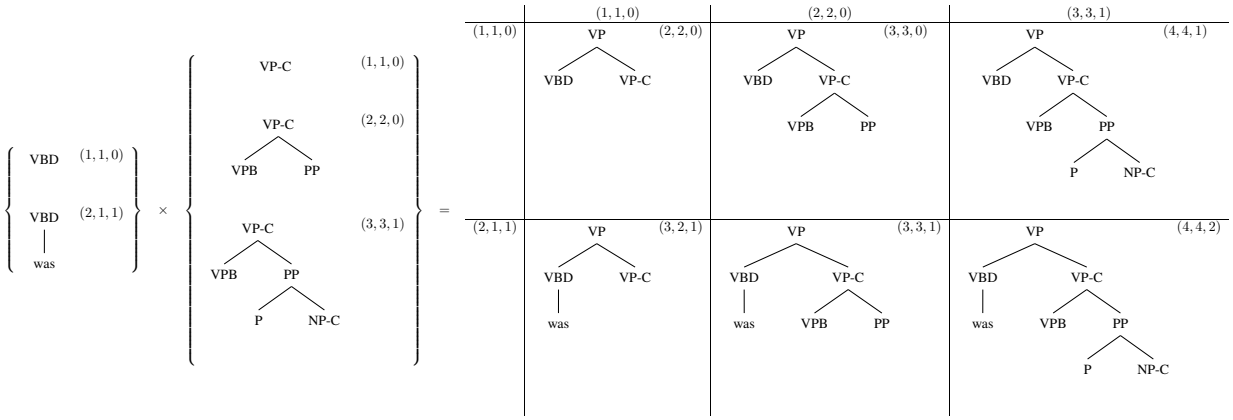


Figure 2: Tree-to-string rule composition as cube-pruning. The left shows two lists of composed rules sorted by their geometric measures $(height, \#\ frontiers, \#\ frontier\ terminals)$, under the gluing rule of VP $\rightarrow$ VBD VP$-$C. The right part shows a cube view of the combination space. We explore the space from the top-left corner to the neighbors.

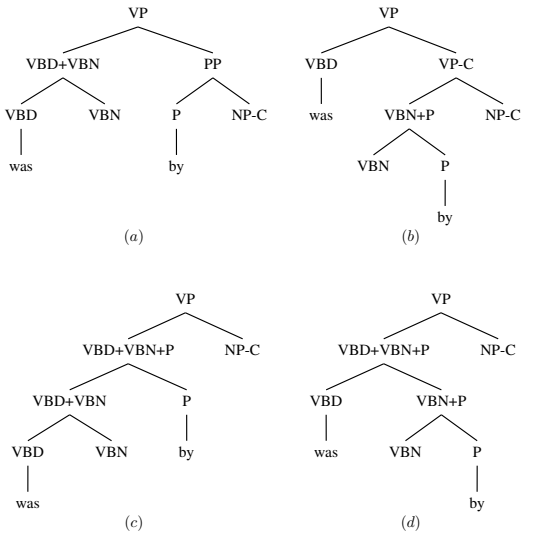
the commonality of subsequences can be discovered. For example, the simplest binarization methods *left-to-right*, *right-to-left*, and *head-out* explore sharing of prefixes or suffixes. Among exponentially many binarization choices, these algorithms pick a single bracketing structure for a sequence of sibling nodes. To explore all possible binarizations, we use a CYK algorithm to produce a packed forest of binary trees for a given sibling sequence.

With CYK binarization, we can explore any span that is nested within the original tree structure, but still miss all cross-bracket spans. For example, translating from English to Chinese, The phrase "There is" should often be translated into one verb in Chinese. In a correct English parse tree, however, the subject-verb boundary is between "There" and "is". As a result, tree-to-string translation based on constituent phrases misses the good translation rule.

The CYK-n binarization algorithm shown in Algorithm 1 is a parameterization of the basic CYK binarization algorithm we just outlined. The idea is that binarization can go beyond the scope of parent nodes to more distant ancestors. The CYK-n algorithm first annotates each node with its n nearest ancestors in the source tree, then generates a binarization forest that allows combining any two nodes with common ancestors. The ancestor chain labeled at each node licenses the node to only combine with nodes having common ancestors in the past n generations.

The algorithm creates new tree nodes on the fly. New tree nodes need to have their own states indicated by a node label representing what is covered internally by the node and an ancestor chain representing which nodes the node attaches to externally. Line 22 and Line 23 of Algorithm 1 update the label and ancestor annotations of new tree nodes. Using the parsing semiring notations (Goodman, 1999), the ancestor computation can be summarized by the $(\cap, \cup)$ pair. $\cap$ produces the ancestor chain of a hyper-edge. $\cup$ produces the ancestor chain of a hyper-node. The node label computation can be summarized by the $(\mathrm{concatenate}, \mathrm{min})$ pair. $\mathrm{concatenate}$ produces a concatenation of node labels. min yields the label with the shortest length. A *tree-sequence* (Liu et al., 2007) is a sequence of sub-trees covering adjacent spans. It can be proved that the final label of each new node in the forest corresponds to the tree sequence which has the minimum length among all sequences covered by the node span. The ancestor chain of a new node is the common ancestors of the nodes in its minimum tree sequence.

For clarity, we do full CYK loops over all $O(|w|^2)$ spans and $O(|w|^3)$ potential hyper-edges, where $|w|$ is the length of a source string. In reality, only descendants under a shared ancestor can combine. If we assume trees have a bounded branching factor b, the number of descendants after n generations is still bounded by a constant $c = b^n$. The algorithm is $O(c^3 \cdot |w|)$, which is still linear to the size of input sentence when the parameter n is a constant.



	1	2	3	4
0	VBD	*VBD+VBN*	*VBD+VBN+P*	VP
1		VBN	*VBN+P*	VP-C
2			P	PP
3				NP-C

Figure 3: Alternative binary parses created for the original tree fragment in Figure 1 through CYK-2 binarization (a and b) and CYK-3 binarization (c and d). In the chart representation at the bottom, cells with labels containing the concatenation symbol $+$ hold nodes created through binarization.

Figure 3 shows some examples of alternative trees generated by the CYK-n algorithm. In this example, standard CYK binarization will not create any new trees since the input is already binary. The CYK-2 and CYK-3 algorithms discover new trees with an increasing degree of freedom.

4 Synchronous Binarization for Forest-to-string Decoding

In this section, we deal with binarization of translation forests, also known as translation hypergraphs (Mi et al., 2008). A translation forest is a packed forest representation of all synchronous derivations composed of tree-to-string rules that match the source forest. Tree-to-string decoding algorithms work on a translation forest, rather than a source forest. A binary source forest does not necessarily always result in a binary translation forest. In the tree-to-string rule in Figure 4, the source tree is already

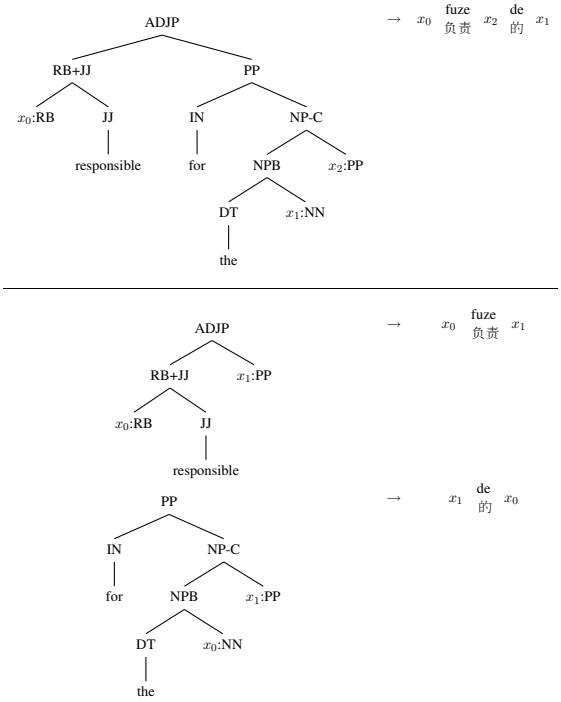


Figure 4: Synchronous binarization for a tree-to-string rule. The top rule can be binarized into two smaller rules.

binary with the help of source tree binarization, but the translation rule involves three variables in the set of frontier nodes. If we apply synchronous binarization (Zhang et al., 2006), we can factorize it into two smaller translation rules each having two variables. Obviously, the second rule, which is a common pattern, is likely to be shared by many translation rules in the derivation forest. When beams are fixed, search goes deeper in a factorized translation forest.

The challenge of synchronous binarization for a forest-to-string system is that we need to first match large tree fragments in the input forest as the first step of decoding. Our solution is to do the matching using the original rules and then run synchronous binarization to break matching rules down to factor rules which can be shared in the derivation forest. This is different from the offline binarization scheme described in (Zhang et al., 2006), although the core algorithm stays the same.

5 Experiments

We ran experiments on public data sets for English to Chinese, Czech, French, German, and Spanish

Algorithm 1 The CYK-n Binarization Algorithm

function CYKBINARIZER(T, n)
  for each tree $node \in T$ in bottom-up topological order **do**
    Make a copy of $node$ in the forest output F
    $Ancestors[node]$ = the nearest n ancestors of $node$
    $Label[node]$ = the label of $node$ in T
  $L \leftarrow$ the length of the yield of T
  for $k = 2...L$ **do**
    for $i = 0, ..., L - k$ **do**
      for $j = i + 1, ..., i + k - 1$ **do**
        $lnode \leftarrow Node[i, j]$; $rnode \leftarrow Node[j, i + k]$
        if $Ancestors[lnode] \cap Ancestors[rnode] \neq \emptyset$ **then**
          $pnode \leftarrow$ GETNODE($i, i + k$)
          ADDEDGE($pnode, lnode, rnode$)
  return F
function GETNODE($begin, end$)
  if $Node[begin, end] \notin F$ **then**
    Create a new $node$ for the span $(begin, end)$
    $Ancestors[node] = \emptyset$
    $Label[node]$ = the sequence of terminals in the span $(begin, end)$ in T
  return $Node[begin, end]$
function ADDEDGE($pnode, lnode, rnode$)
  Add a hyper-edge from $lnode$ and $rnode$ to $pnode$
  $Ancestors[pnode] = Ancestors[pnode] \cup (Ancestors[lnode] \cap Ancestors[rnode])$
  $Label[pnode] = \min\{Label[pnode],$ CONCATENATE($Label[lnode], Label[rnode]$)$\}$

translation to evaluate our methods.

5.1 Setup

For English-to-Chinese translation, we used all the allowed training sets in the NIST 2008 constrained track. For English to the European languages, we used the training data sets for WMT 2010 (Callison-Burch et al., 2010). For NIST, we filtered out sentences exceeding 80 words in the parallel texts. For WMT, the filtering limit is 60. There is no filtering on the test data set. Table 1 shows the corpus statistics of our bilingual training data sets.

	Source Words	Target Words
English-Chinese	287M	254M
English-Czech	66M	57M
English-French	857M	996M
English-German	45M	43M
English-Spanish	216M	238M

Table 1: The Sizes of Parallel Texts.

At the word alignment step, we did 6 iterations of IBM Model-1 and 6 iterations of HMM. For English-Chinese, we ran 2 iterations of IBM Model-4 in addition to Model-1 and HMM. The word alignments are symmetrized using the "union" heuristics. Then, the standard phrase extraction heuristics (Koehn et al., 2003) were applied to extract phrase pairs with a length limit of 6. We ran the hierarchical phrase extraction algorithm with the standard heuristics of Chiang (2005). The phrase-length limit is interpreted as the maximum number of symbols on either the source side or the target side of a given rule. On the same aligned data sets, we also ran the tree-to-string rule extraction algorithm described in Section 2.1 with a limit of 16 rules per tree node.

The default parser in the experiments is a shift-reduce dependency parser (Nivre and Scholz, 2004). It achieves 87.8% labelled attachment score and 88.8% unlabeled attachment score on the standard Penn Treebank test set. We convert dependency parses to constituent trees by propagating the part-of-speech tags of the head words to the corresponding phrase structures.

We compare three systems: a phrase-based system (Och and Ney, 2004), a hierarchical phrase-based system (Chiang, 2005), and our forest-to-string system with different binarization schemes. In the phrase-based decoder, jump width is set to 8. In the hierarchical decoder, only the glue rule is applied

to spans longer than 10. For the forest-to-string system, we do not have such length-based reordering constraints.

We trained two 5-gram language models with Kneser-Ney smoothing for each of the target languages. One is trained on the target side of the parallel text, the other is on a corpus provided by the evaluation: the Gigaword corpus for Chinese and news corpora for the others. Besides standard features (Och and Ney, 2004), the phrase-based decoder also uses a Maximum Entropy phrasal reordering model (Zens and Ney, 2006). Both the hierarchical decoder and the forest-to-string decoder only use the standard features. For feature weight tuning, we do Minimum Error Rate Training (Och, 2003). To explore a larger n-best list more efficiently in training, we adopt the hypergraph-based MERT (Kumar et al., 2009).

To evaluate the translation results, we use BLEU (Papineni et al., 2002).

5.2 Translation Results

Table 2 shows the scores of our system with the best binarization scheme compared to the phrase-based system and the hierarchical phrase-based system. Our system is consistently better than the other two systems in all data sets. On the English-Chinese data set, the improvement over the phrase-based system is 1.3 BLEU points, and 0.8 over the hierarchical phrase-based system. In the tasks of translating to European languages, the improvements over the phrase-based baseline are in the range of 0.5 to 1.0 BLEU points, and 0.3 to 0.5 over the hierarchical phrase-based system. All improvements except the *bf2s* and *hier* difference in English-Czech are significant with confidence level above 99% using the bootstrap method (Koehn, 2004). To demonstrate the strength of our systems including the two baseline systems, we also show the reported best results on these data sets from the 2010 WMT workshop. Our forest-to-string system (*bf2s*) outperforms or ties with the best ones in three out of four language pairs.

		BLEU	
		dev	test
English-Chinese	pb	29.7	39.4
	hier	31.7	38.9
	bf2s	31.9	40.7**
English-Czech	*wmt best*	-	*15.4*
	pb	14.3	15.5
	hier	14.7	16.0
	bf2s	14.8	16.3*
English-French	*wmt best*	-	*27.6*
	pb	24.1	26.1
	hier	23.9	26.1
	bf2s	24.5	26.6**
English-German	*wmt best*	-	*16.3*
	pb	14.5	15.5
	hier	14.9	15.9
	bf2s	15.2	16.3**
English-Spanish	*wmt best*	-	*28.4*
	pb	24.1	27.9
	hier	24.2	28.4
	bf2s	24.9	28.9**

Table 2: Translation results comparing *bf2s*, the binarized-forest-to-string system, *pb*, the phrase-based system, and *hier*, the hierarchical phrase-based system. For comparison, the best scores from WMT 2010 are also shown. ** indicates the result is significantly better than both *pb* and *hier*. * indicates the result is significantly better than *pb* only.

5.3 Different Binarization Methods

The translation results for the *bf2s* system in Table 2 are based on the *cyk* binarization algorithm with bracket violation degree 2. In this section, we vary the degree to generate forests that are incrementally augmented from a single tree. Table 3 shows the scores of different tree binarization methods for the English-Chinese task.

It is clear from reading the table that *cyk*-2 is the optimal binarization parameter. We have verified this is true for other language pairs on non-standard data sets. We can explain it from two angles. At degree 2, we allow phrases crossing at most one bracket in the original tree. If the parser is reasonably good, crossing just one bracket is likely to cover most interesting phrases that can be translation units. From another point of view, enlarging the forests entails more parameters in the resulting translation model, making over-fitting likely to happen.

5.4 Binarizer or Parser?

A natural question is how the binarizer-generated forests compare with parser-generated forests in translation. To answer this question, we need a

	rules	BLEU dev	BLEU test
no binarization	378M	28.0	36.3
head-out	408M	30.0	38.2
cyk-1	527M	31.6	40.5
cyk-2	803M	31.9	40.7
cyk-3	1053M	32.0	40.6
cyk-∞	1441M	32.0	40.3

Table 3: Comparing different source tree binarization schemes for English-Chinese translation, showing both BLEU scores and model sizes. The rule counts include normal phrases which are used at the leaf level during decoding.

parser that can generate a packed forest. Our fast deterministic dependency parser does not generate a packed forest. Instead, we use a CRF constituent parser (Finkel et al., 2008) with state-of-the-art accuracy. On the standard Penn Treebank test set, it achieves an F-score of 89.5%. It uses a CYK algorithm to do full dynamic programming inference, so is much slower. We modified the parser to do hyperedge pruning based on posterior probabilities. The parser preprocesses the Penn Treebank training data through binarization. So the packed forest it produces is also a binarized forest. We compare two systems: one is using the *cyk*-2 binarizer to generate forests; the other is using the CRF parser with pruning threshold e^{-p}, where $p = 2$ to generate forests.[1] Although the parser outputs binary trees, we found cross-bracket *cyk*-2 binarization is still helpful.

	BLEU dev	BLEU test
cyk-2	14.9	16.0
parser	14.7	15.7

Table 4: Binarized forests versus parser-generated forests for forest-to-string English-German translation.

Table 4 shows the comparison of binarization forest and parser forest on English-German translation. The results show that *cyk*-2 forest performs slightly better than the parser forest. We have not done full exploration of forest pruning parameters to fine-tune the parser-forest. The speed of the constituent parser is the efficiency bottleneck. This actually demonstrates the advantage of the binarizer plus forest-to-string scheme. It is flexible, and works with any parser that generates projective parses. It does not require hand-tuning of forest pruning parameters for training.

[1]All hyper-edges with negative log posterior probability larger than p are pruned. In Mi and Huang (2008), the threshold is $p = 10$. The difference is that they do the forest pruning on a forest generated by a k-best algorithm, while we do the forest-pruning on the full CYK chart. As a result, we need more aggressive pruning to control forest size.

5.5 Synchronous Binarization

In this section, we demonstrate the effect of synchronous binarization for both tree-to-string and forest-to-string translation. The experiments are on the English-Chinese data set. The baseline systems use k-way cube pruning, where k is the branching factor, i.e., the maximum number of nonterminals on the right-hand side of any synchronous translation rule in an input grammar. The competing system does online synchronous binarization as described in Section 4 to transform the grammar intersected with the input sentence to the minimum branching factor k' ($k' < k$), and then applies k'-way cube pruning. Typically, k' is 2.

		BLEU dev	BLEU test
head-out	cube pruning	29.2	37.0
	+ synch. binarization	30.0	38.2
cyk-2	cube pruning	31.7	40.5
	+ synch. binarization	31.9	40.7

Table 5: The effect of synchronous binarization for tree-to-string and forest-to-string systems, on the English-Chinese task.

Table 5 shows that synchronous binarization does help reduce search errors and find better translations consistently in all settings.

6 Related Work

The idea of concatenating adjacent syntactic categories has been explored in various syntax-based models. Zollmann and Venugopal (2006) augmented hierarchial phrase based systems with joint syntactic categories. Liu et al. (2007) proposed tree-sequence-to-string translation rules but did not provide a good solution to place joint subtrees into connection with the rest of the tree structure. Zhang et

al. (2009) is the closest to our work. But their goal was to augment a k-best forest. They did not binarize the tree sequences. They also did not put constraint on the tree-sequence nodes according to how many brackets are crossed.

Wang et al. (2007) used target tree binarization to improve rule extraction for their string-to-tree system. Their binarization forest is equivalent to our *cyk*-1 forest. In contrast to theirs, our binarization scheme affects decoding directly because we match tree-to-string rules on a binarized forest.

Different methods of translation rule binarization have been discussed in Huang (2007). Their argument is that for tree-to-string decoding target side binarization is simpler than synchronous binarization and works well because creating discontinous source spans does not explode the state space. The forest-to-string senario is more similar to string-to-tree decoding in which state-sharing is important. Our experiments show that synchronous binarization helps significantly in the forest-to-string case.

7 Conclusion

We have presented a new approach to tree-to-string translation. It involves a source tree binarization step and a standard forest-to-string translation step. The method renders it unnecessary to have a k-best parser to generate a packed forest. We have demonstrated state-of-the-art results using a fast parser and a simple tree binarizer that allows crossing at most one bracket in each binarized node. We have also shown that reducing search errors is important for forest-to-string translation. We adapted the synchronous binarization technqiue to improve search and have shown significant gains. In addition, we also presented a new cube-pruning-style algorithm for rule extraction. In the new algorithm, it is easy to adjust the figure-of-merit of rules for extraction. In the future, we plan to improve the learning of translation rules with binarized forests.

Acknowledgments

We would like to thank the members of the MT team at Google, especially Ashish Venugopal, Zhifei Li, John DeNero, and Franz Och, for their help and discussions. We would also like to thank Daniel Gildea for his suggestions on improving the paper.

References

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics(MATR)*, pages 17–53, Uppsala, Sweden, July. Association for Computational Linguistics. Revised August 2010.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, pages 263–270, Ann Arbor, MI.

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

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 755–763, Prague, Czech Republic, June. Association for Computational Linguistics.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics, companion volume*, pages 205–208, Sapporo, Japan.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL-08: HLT*, pages 959–967, Columbus, Ohio, June. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-04)*, 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 the International Conference on Computational Linguistics/Association for Computational Linguistics (COLING/ACL-06)*, pages 961–968, July.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-04)*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, MA.

Liang Huang. 2007. Binarization, synchronous binarization, and target-side binarization. In *Proceedings of the NAACL/AMTA Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 33–40, Rochester, NY.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the 46th Annual Conference of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT)*, Columbus, OH. ACL.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, Edmonton, Alberta.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, Barcelona, Spain, July.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 163–171, Suntec, Singapore, August. Association for Computational Linguistics.

Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, pages 625–630, Geneva, Switzerland.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the International Conference on Computational Linguistics/Association for Computational Linguistics (COLING/ACL-06)*, Sydney, Australia, July.

Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *Proceedings of the 45th Annual Conference of the Association for Computational Linguistics (ACL-07)*, Prague.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 206–214, Honolulu, Hawaii, October. Association for Computational Linguistics.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of the 46th Annual Conference of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT)*, pages 192–199.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of Coling 2004*, pages 64–70, Geneva, Switzerland, Aug 23–Aug 27. COLING.

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

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41th Annual Conference of the Association for Computational Linguistics (ACL-03)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*.

Arjen Poutsma. 2000. Data-oriented translation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, pages 271–279, Ann Arbor, Michigan.

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 the 46th Annual Conference of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT)*, Columbus, OH. ACL.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 746–754, Prague, Czech Republic, June. Association for Computational Linguistics.

Richard Zens and Hermann Ney. 2006. Discriminative reordering models for statistical machine translation. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 55–63, New York City, June. Association for Computational Linguistics.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the 2006 Meeting of the*

North American chapter of the Association for Computational Linguistics (NAACL-06), pages 256–263, New York, NY.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of ACL-08: HLT*, pages 559–567, Columbus, Ohio, June. Association for Computational Linguistics.

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 Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 172–180, Suntec, Singapore, August. Association for Computational Linguistics.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 138–141, New York City, June. Association for Computational Linguistics.