Binarization, Synchronous Binarization, and Target-side Binarization*

Liang Huang
University of Pennsylvania
3330 Walnut Street, Levine Hall
Philadelphia, PA 19104
lhuang3@cis.upenn.edu

Abstract

Binarization is essential for achieving polynomial time complexities in parsing and syntax-based machine translation. This paper presents a new binarization scheme, *target-side binarization*, and compares it with source-side and synchronous binarizations on both string-based and tree-based systems using synchronous grammars. In particular, we demonstrate the effectiveness of target-side binarization on a large-scale tree-to-string translation system.

1 Introduction

Several recent syntax-based models for machine translation (Chiang, 2005; Galley et al., 2006) can be seen as instances of the general framework of synchronous grammars and tree transducers. In this framework, decoding can be thought of as parsing problems, whose complexity is in general exponential in the number of nonterminals on the right hand side of a grammar rule. To alleviate this problem, one can borrow from parsing the technique of binarizing context-free grammars (into Chomsky Normal Form) to reduce the complexity. With synchronous context-free grammars (SCFG), however, this problem becomes more complicated with the additional dimension of target-side permutation.

The simplest method of binarizing an SCFG is to binarize (left-to-right) on the source-side as if treating it as a monolingual CFG for the source-langauge. However, this approach does not guarantee contiguous spans on the target-side, due to the arbitrary re-ordering of nonterminals between the two languages. As a result, decoding with an integrated language model still has an exponential complexity.

Synchronous binarization (Zhang et al., 2006) solves this problem by simultaneously binarizing both source and target-sides of a synchronous rule, making sure of contiguous spans on both sides whenever possible. Neglecting the small amount of non-binarizable rules, the decoding complexity with an integrated language model becomes polynomial and translation quality is significantly improved thanks to the better search. However, this method is more sophisticated to implement than the previous method and binarizability ratio decreases on freer word-order languages (Wellington et al., 2006).

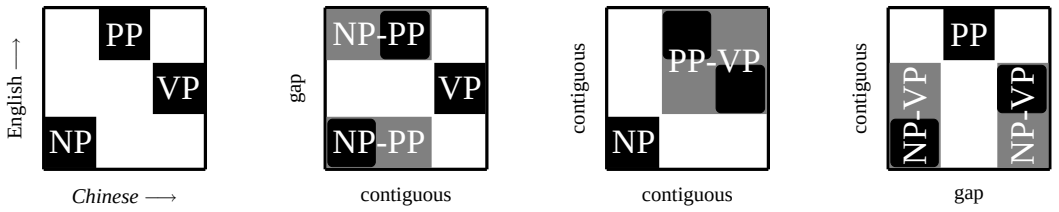
This paper presents a third alternative, *target-side binarization*, which is the symmetric version of the simple source-side variant mentioned above. We compare it with the other two schemes in two popular instantiations of MT systems based on SCFGs: the *string-based systems* (Chiang, 2005; Galley et al., 2006) where the input is a string to be parsed using the source-side of the SCFG; and the *tree-based systems* (Liu et al., 2006; Huang et al., 2006) where the input is a parse tree and is recursively converted into a target string using the SCFG as a tree-transducer. While synchronous binarization is the best strategy for string-based systems, we show that target-side binarization can achieve the same performance of synchronous binarization for tree-based systems, with much simpler implementation and 100% binarizability.

2 Synchronous Grammars and Binarization Schemes

In this section, we define synchronous context-free grammars and present the three binarization

*This work is partially supported by NSF ITR grants IIS-0428020 (while I was visiting USC/ISI) and EIA-0205456. I also wish to thank Jonathan Graehl, Giorgio Satta, Hao Zhang, and the three anonymous reviewers for helpful comments.

Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation, pages 33–40,
Rochester, New York, April 2007.



(a) example rule (b) source-side (c) synchronous (d) target-side

Figure 1: Illustration of the three binarization schemes, with virtual nonterminals in gray.

schemes through a motivational example.

A **synchronous CFG** (SCFG) is a context-free rewriting system for generating string pairs. Each rule (*synchronous production*) rewrites a nonterminal in two dimensions subject to the constraint that the sequence of nonterminal children on one side is a permutation of the nonterminal sequence on the other side. Each co-indexed child nonterminal pair will be further rewritten as a unit. The **rank** of a rule is defined as the number of its synchronous nonterminals. We also define the source and target projections of an SCFG to be the CFGs for the source and target languages, respectively.

For example, the following SCFG[1]

$$(1)\quad \begin{array}{rcll} \text{S} & \rightarrow & \text{NP}\boxed{1}\ \text{PP}\boxed{2}\ \text{VP}\boxed{3}, & \text{NP}\boxed{1}\ \text{VP}\boxed{3}\ \text{PP}\boxed{2} \\ \text{NP} & \rightarrow & \textit{Baoweier}, & \text{Powell} \\ \text{VP} & \rightarrow & \textit{juxing le huitan}, & \text{held a meeting} \\ \text{PP} & \rightarrow & \textit{yu Shalong}, & \text{with Sharon} \end{array}$$

captures the re-ordering of PP and VP between Chinese (source) and English (target). The source-projection of the first rule, for example, is

$$\text{S} \rightarrow \text{NP PP VP}.$$

Decoding with an SCFG (e.g., translating from Chinese to English using the above grammar) can be cast as a parsing problem (see Section 3 for details), in which case we need to binarize a synchronous rule with more than two nonterminals to achieve polynomial time algorithms (Zhang et al., 2006). We will next present the three different binarization schemes using Example 1.

[1]An alternative notation, used by Satta and Peserico (2005), allows co-indexed nonterminals to take different symbols across languages, which is convenient in describing syntactic divergences (see Figure 2).

2.1 Source-side Binarization

The first and simplest scheme, *source-side binarization*, works left-to-right on the source projection of the SCFG without respecting the re-orderings on the target-side. So it will binarize the first rule as:

$$(2)\quad \begin{array}{rcl} \text{S} & \rightarrow & \text{NP-PP VP} \\ \text{NP-PP} & \rightarrow & \text{NP PP} \end{array}$$

which corresponds to Figure 1 (b). Notice that the *virtual nonterminal* NP-PP representing the intermediate symbol is *discontinuous* with two spans on the target (English) side, because this binarization scheme completely ignores the reorderings of nonterminals. As a result, the binarized grammar, with a gap on the target-side, is no longer an SCFG, but can be represented in the more general formalism of Multi-Text Grammars (MTG) (Melamed, 2003):

$$(3)\quad \begin{pmatrix} \text{S} \\ \text{S} \end{pmatrix} \rightarrow\!\!\bowtie \begin{matrix} [1,2] \\ [1,2,1] \end{matrix} \begin{pmatrix} \text{NP-PP} & \text{VP} \\ \text{NP-PP}\ (2) & \text{VP} \end{pmatrix}$$

here $[1,2,1]$ denotes that on that target-side, the first nonterminal NP-PP has two discontinuous spans, with the second nonterminal VP in the gap.

Intuitively speaking, the gaps on the target-side will lead to exponential complexity in decoding with integrated language models (see Section 3), as well as synchronous parsing (Zhang et al., 2006).

2.2 Synchronous Binarization

A more principled method is *synchronous binarization*, which simultaneously binarizes both source and target sides, with the constraint that virtual nonterminals always have contiguous spans on both sides. The resulting grammar is thus another SCFG, the binary branching equivalent of the original grammar, which can be thought of as an extension of the

[*jinyibu*]$_1$ [*jiu zhongdong weiji*]$_2$ [*juxing*]$_3$ [*huitan*]$_4$
further on Mideast crisis hold talk

'[hold]$_3$ [further]$_1$ [talks]$_4$ [on the Mideast crisis]$_2$'

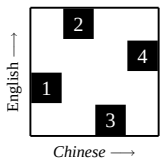


Figure 2: An example of non-binarizable rule from the hand-aligned Chinese-English data in Liu et al. (2005). The SCFG rule is VP $\rightarrow$ ADVP[1] PP[2] VB[3] NN[4], VP $\rightarrow$ VB[3] JJ[1] NNS[4] PP[2] in the notatoin of Satta and Peserico (2005).

Chomsky Normal Form in synchronous grammars. The example rule is now binarized into:

$$\begin{array}{rcll} \text{S} & \rightarrow & \text{NP}^{\boxed{1}}\ \text{PP-VP}^{\boxed{2}}, & \text{NP}^{\boxed{1}}\ \text{PP-VP}^{\boxed{2}} \\ \text{PP-VP} & \rightarrow & \text{PP}^{\boxed{1}}\ \text{VP}^{\boxed{2}}, & \text{VP}^{\boxed{2}}\ \text{PP}^{\boxed{1}} \end{array} \quad (4)$$

which corresponds to Figure 1 (c). This representation, being contiguous on both sides, successfully reduces the decoding complexity to a low polynomial and significantly improved the search quality (Zhang et al., 2006).

However, this scheme has the following drawbacks. First, synchronous binarization is *not* always possible with an arbitrary SCFG. Some reorderings, for example, the permutation $(2, 4, 1, 3)$, is non-binarizable. Although according to Zhang et al. (2006), the vast majority (99.7%) of rules in their Chinese-English dataset are binarizable, there do exist some interesting cases that are not (see Figure 2 for a real-data example). More importantly, the ratio of binarizability, as expected, decreases on freer word-order languages (Wellington et al., 2006). Second, synchronous binarization is significantly more complicated to implement than the straightforward source-side binarization.

2.3 Target-side Binarization

We now introduce a novel scheme, target-side binarization, which is the symmetric version of the source-side variant. Under this method, the target-side is always contiguous, while leaving some gaps on the source-side. The example rule is binarized into the following MTG form:

$$\begin{pmatrix} \text{S} \\ \text{S} \end{pmatrix} \rightarrow\bowtie \begin{matrix} [1,2,1] \\ [1,2] \end{matrix} \begin{pmatrix} \text{NP-VP}\ (2) & \text{PP} \\ \text{NP-VP} & \text{PP} \end{pmatrix} \quad (5)$$

which corresponds to Figure 1 (d).

scheme	$s(\mathbf{b})$	$t(\mathbf{b})$
source-side	1	$\leq n/2$
synchronous	1	1
target-side	$\leq n/2$	1

Table 1: Source and target arities of the three binarization schemes of an SCFG rule of rank n.

Although the discontinuity on the source-side in this new scheme causes exponential complexity in string-based systems (Section 3.1), the continuous spans on the target-side will ensure polynomial complexity in tree-based systems (Section 3.2).

Before we move on to study the effects of various binarization schemes in decoding, we need some formal machineries of discontinuities.

We define the **source** and **target arities** of a virtual nonterminal V, denoted $s(V)$ and $t(V)$, to be the number of (consecutive) spans of V on the source and target sides, respectively. This definition extends to a binarization $\mathbf{b}$ of an SCFG rule of rank n, where arities $s(\mathbf{b})$ and $t(\mathbf{b})$ are defined as the maximum source and target arities over all virtual nonterminals in $\mathbf{b}$, respectively. For example, the source and target arities of the three binarizations in Figure 1 are 1 and 2 for (b), 1 and 1 for (c), and 2 and 1 for (d). In general, the arities for the three binarization schemes are summarized in Table 1.

3 Theoretical Analysis

We now compare the algorithmic complexities of the three binarization schemes in a central problem of machine translation: decoding with an integrated n-gram language model. Depending on the input being a string or a parse-tree, we divide MT systems based on synchronous grammars into two broad categories: string-based and tree-based.

3.1 String-based Approaches

String-based approaches include both string-to-string (Chiang, 2005) and string-to-tree systems (Galley et al., 2006).[2] To simplify the presentation we will just focus on the former but the analysis also applies to the latter. We will first discuss decoding with a pure SCFG as the translation model (henceforth *−LM decoding*), and then extend it to include an n-gram model (+LM decoding).

3.1.1 Translation as Parsing

The −LM decoder can be cast as a (monolingual) parser on the source language: it takes the source-language string as input and parses it using the source-projection of the SCFG while building the corresponding target-language sub-translations in parallel. For source-side and synchronous binarizations, since the resulting grammar has contiguous source spans, we can apply the CKY algorithm which guarantees cubic time complexity.

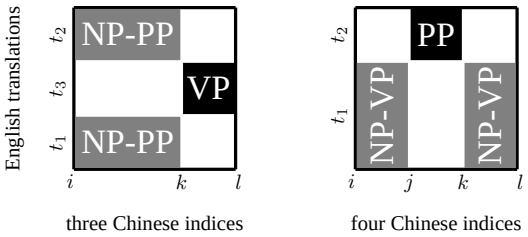
For example, a deduction along the virtual rule in the synchronously binarized grammar (4) is notated

$$\frac{(\text{PP}_{j,k}) : (w_1, t_1) \qquad (\text{VP}_{k,l}) : (w_2, t_2)}{(\text{PP-VP}_{j,l}) : (w_1 + w_2, t_2 t_1)} \tag{6}$$

where i, j, k are free indices in the source string, w_1, w_2 are the scores of the two antecedent items, and t_1, t_2 are the corresponding sub-translations.[3] The resulting translation $t_2 t_1$ is the inverted concatenation as specified by the target-side of the SCFG rule.

The case for a source-side binarized grammar (3) is slightly more complicated than the above, because we have to keep track of gaps on the target side. For example, we first combine NP with PP

$$\frac{(\text{NP}_{i,j}) : (w_1, t_1) \qquad (\text{PP}_{j,k}) : (w_2, t_2)}{(\text{NP-PP}_{i,k}) : (w_1 + w_2, t_1 \sqcup t_2)} \tag{7}$$



(a): Deduction (8) (b): Deduction (10)

Figure 3: Illustrations of two deductions with gaps.

leaving a gap (⊔) on the target-side resulting item, because NP and PP are not contiguous in the English ordering. This gap is later filled in by the sub-translation t_3 of VP (see also Figure 3 (a)):

$$\frac{(\text{NP-PP}_{i,k}) : (w_1, t_1 \sqcup t_2) \qquad (\text{VP}_{k,l}) : (w_2, t_3)}{(\text{S}_{i,l}) : (w_1 + w_2, t_1 t_3 t_2)} \tag{8}$$

In both cases, there are still only three free indices on the source-side, so the complexity remains cubic. The gaps on the target-side do not require any extra computation in the current −LM setting, but as we shall see shortly below, will lead to exponential complexity when integrating a language model.

For a target-side binarized grammar as in (5), however, the source-side spans are discontinuous where CKY can not apply, and we have to enumerate more free indices on the source side. For example, the first deduction

$$\frac{(\text{NP}_{i,j}) : (w_1, t_1) \qquad (\text{VP}_{k,l}) : (w_2, t_2)}{(\text{NP-VP}_{i,j \sqcup k,l}) : (w_1 + w_2, t_1 t_2)} \tag{9}$$

leaves a gap in the source-side span of the resulting item, which is later filled in when the item is combined with a PP (see also Figure 3 (b)):

$$\frac{(\text{NP-VP}_{i,j \sqcup k,l}) : (w_1, t_1) \qquad (\text{PP}_{j,k}) : (w_2, t_2)}{(\text{S}_{i,l}) : (w_1 + w_2, t_1 t_2)} \tag{10}$$

Both of the above deductions have four free indices, and thus of complexity $\mathcal{O}(|w|^4)$ instead of cubic in the length of the input string w.

More generally, the complexity of a binarization scheme depends on its source arity. In the worst-case, a binarized grammar with a source arity of s will require at most $(2s+1)$ free indices in a deduction, because otherwise if one rule needs $(2s+2)$

[2] Our notation of *X-to-Y systems* is defined as follows: X denotes the input, either a string or a tree; while Y represents the RHS structure of an individual rule: Y is *string* if the RHS is a flat one-level tree (as in SCFGs), and Y is *tree* if the RHS is multi-level as in (Galley et al., 2006). This convention also applies to tree-based approaches.

[3] The actual system does not need to store the translations since they can be recovered from backpointers and they are *not* considered part of the state. We keep them here only for presentation reasons.

indices, then there are $s+1$ spans, which contradicts the definition of arity (Huang et al., 2005).[4]

These deductive systems represent the search space of decoding without a language model. When one is instantiated for a particular input string, it defines a set of derivations, called a *forest*, represented in a compact structure that has a structure of a hypergraph. Accordingly we call items like $(\text{PP}_{1,3})$ *nodes* in the forest, and an instantiated deduction like

$$(\text{PP-VP}_{1,6}) \rightarrow (\text{PP}_{1,3})(\text{VP}_{3,6})$$

we call a *hyperedge* that connects one or more antecedent nodes to a consequent node. In this representation, the time complexity of $-$LM decoding, which we refer to as *source-side complexity*, is proportional to the size of the forest F, i.e., the number of hyperedges (instantiated deductions) in F. To summarize, the source-side complexity for a binarized grammar of source arity s is

$$|F| = \mathcal{O}(|w|^{2s+1}).$$

3.1.2 Adding a Language Model

To integrate with a bigram language model, we can use the dynamic-programming algorithm of Wu (1996), which we may think of as proceeding in two passes. The first pass is as above, and the second pass traverses the first-pass forest, assigning to each node v a set of augmented items, which we call *+LM items*, of the form $(v^{a \star b})$, where a and b are target words and $\star$ is a placeholder symbol for an elided part of a target-language string. This item indicates that a possible translation of the part of the input spanned by v is a target string that starts with a and ends with b.

Here is an example deduction in the synchronously binarized grammar (4), for a $+$LM item for the node $(\text{PP-VP}_{1,6})$ based on the $-$LM Deduction (6):

$$\frac{(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}}) : (w_1, t_1) \quad (\text{VP}_{3,6}^{\text{held} \star \text{talk}}) : (w_2, t_2)}{(\text{PP-VP}_{1,6}^{\text{held} \star \text{Sharon}}) : (w', t_2 t_1)} \tag{11}$$

where $w' = w_1 + w_2 - \log P_{lm}(\text{with} \mid \text{talk})$ is the score of the resulting $+$LM item: the sum of the scores of the antecedent items, plus a *combination cost* which is the negative log probability of the bigrams formed in combining adjacent boundary words of antecedents.

Now that we keep track of target-side boundary words, an additional complexity, called *target-side complexity*, is introduced. In Deduction (11), four target words are enumerated, and each $+$LM item stores two boundary words; this is also true in general for synchronous and target-side binarized grammars where we always combine two consecutive target strings in a deduction. More generally, this scheme can be easily extended to work with an m-gram model (Chiang, 2007) where m is usually ≥ 3 (trigram or higher) in practice. The target-side complexity for this case is thus

$$\mathcal{O}(|V|^{4(m-1)})$$

where V is the target language vocabulary. This is because each constituent must store its initial and final $(m-1)$-grams, which yields four $(m-1)$-grams in a binary combination. In practice, it is often assumed that there are only a constant number of translations for each input word, which reduces this complexity into $\mathcal{O}(|w|^{4(m-1)})$.

However, for source-side binarization which leaves gaps on the target-side, the situation becomes more complicated. Consider Deduction (8), where the sub-translation for the virtual node NP-PP is gapped ($t_1 \sqcup t_2$). Now if we integrate a bigram model based on that deduction, we have to maintain the boundary words of both t_1 and t_2 in the $+$LM node of NP-PP. Together with the boundary words in node VP, there are a total of six target words to enumerate for this $+$LM deduction:

$$\frac{(\text{NP-PP}_{i,k}^{a \star b \sqcup e \star f}) : (w_1, t_1 \sqcup t_2) \quad (\text{VP}_{k,l}^{c \star d}) : (w_2, t_3)}{(\text{S}_{i,l}^{a \star f}) : (w', t_1 t_3 t_2)} \tag{12}$$

where $w' = w_1 + w_2 - \log P_{lm}(\text{c} \mid \text{b}) P_{lm}(\text{e} \mid \text{d})$.

With an analysis similar to that of the source-side, we state that, for a binarized grammar with target arity t, the target-side complexity, denoted $\mathcal{T}$, is

$$\mathcal{T} = \mathcal{O}(|w|^{2(t+1)(m-1)})$$

[4] Actually this is true only if in any binarization scheme, a non-contiguous item is always combined with a contiguous item. We define both source and target binarizations to be *incremental* (i.e., left-to-right or right-to-left), so this assumption trivially holds. More general binarization schemes are possible to have even higher complexities, but also possible to achieve better complexities. Full discussion is left for a separate paper.

scheme	string-based	tree-based
source-side	$\|w\|^{3+2(t+1)(m-1)}$	$\|w\|^{1+2(t+1)(m-1)}$
synchronous	$\|w\|^{3+4(m-1)}$	$\|w\|^{1+4(m-1)}$
target-side	$\|w\|^{(2s+1)+4(m-1)}$	$\|w\|^{1+4(m-1)}$

Table 2: Worst-case decoding complexities of the three binarization schemes in the two approaches (excluding the $O(|w|^3)$ time for source-side parsing in tree-based approaches).

because in the worst-case, there are $t+1$ spans involved in a $+$LM deduction (t of them from one virtual antecedent and the other one non-virtual), and for each span, there are $m-1$ target words to enumerate at both left and right boundaries, giving a total of $2(t+1)(m-1)$ words in this deduction. We now conclude that, in a string-based system, the combined complexities for a binarized grammar with source arity s and target arity t is

$$\mathcal{O}(|F|\mathcal{T}) = \mathcal{O}(|w|^{(2s+1)+2(t+1)(m-1)}).$$

The results for the three specific binarization schemes are summarized in Table 2. Although both source-side and target-side binarizations lead to exponential complexities, it is likely that language model combinations (target-side complexity) dominate the computation, since m is larger than 2 in practice. In this sense, target-side binarization is still preferable to source-side binarization.

It is also worth noting that with the hook trick of Huang et al. (2005), the target-side complexity can be reduced to $\mathcal{O}(|w|^{(2t+1)(m-1)})$, making it more analogous to its source-side counterpart: if we consider the decoding problem as intersecting the SCFG with a source-side DFA which has $|S| = |w|+1$ states, and a target-side DFA which has $|T| = O(|w|^{m-1})$ states, then the intersected grammar has a parsing complexity of $\mathcal{O}(|S|^{2s+1}|T|^{2t+1})$, which is symmetric from both sides.

3.2 Tree-based Approaches

The *tree-based approaches* include the tree-to-string (also called *syntax-directed*) systems (Liu et al., 2006; Huang et al., 2006). This approach takes a source-language parse tree, instead of the plain string, as input, and tries to find the best derivation that recursively rewrites the input tree into a target

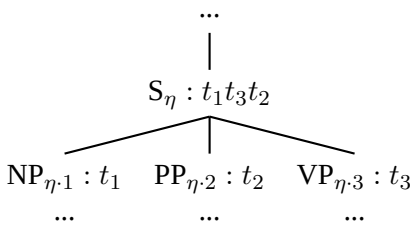


Figure 4: Illustration of tree-to-string deduction.

string, using the SCFG as a tree-transducer. In this setting, the $-$LM decoding phase is a *tree-parsing* problem (Eisner, 2003) which aims to cover the entire tree by a set of rules. For example, a deduction of the first rule in Example 1 would be:

$$\frac{(\text{NP}_{\eta\cdot 1}) : (w_1, t_1) \quad (\text{PP}_{\eta\cdot 2}) : (w_2, t_2) \quad (\text{VP}_{\eta\cdot 3}) : (w_3, t_3)}{(\text{S}_\eta) : (w_1 + w_2 + w_3, t_1 t_3 t_2)} \tag{13}$$

where η and $\eta \cdot i (i = 1, 2, 3)$ are tree addresses (Shieber et al., 1995), with $\eta \cdot i$ being the i^{th} child of η (the address of the root node is ϵ). The nonterminal labels at these tree nodes must match those in the SCFG rule, e.g., the input tree must have a PP at node $\eta \cdot 2$.

The semantics of this deduction is the following: if the label of the current node in the input tree is S, and its three children are labeled NP, PP, and VP, with corresponding sub-translations t_1, t_2, and t_3, then a possible translation for the current node S is $t_1 t_3 t_2$ (see Figure 4). An alternative, top-down version of this bottom-up deductive system is, at each node, try all SCFG rules that *pattern-match* the current subtree, and recursively solve sub-problems indicated by the variables, i.e., synchronous nonterminals, of the matching rule (Huang et al., 2006).

With the input tree completely given, this setting has some fundamental differences from its string-based counterpart. First, we do *not* need to binarize the SCFG grammar before $-$LM decoding. In fact, it will be much harder to do the tree-parsing (pattern-matching) with a binarized grammar. Second, regardless of the number of nonterminals in a rule, building the $-$LM forest always costs time linear in the size of the input tree (times a grammar constant, see (Huang et al., 2006, Sec. 5.1) for details), which is in turn linear in the length of the input string. So we have:

$$\mathcal{O}(|F|) = \mathcal{O}(|w|).$$

This fast −LM decoding is a major advantage of tree-based approaches.

Now in +LM decoding, we still need binarization of the hyperedges, as opposed to rules, in the forest, but the analysis is almost identical to that of string-based approach. For example, the tree-based version of Deduction (12) for source-side binarization is now notated

$$\frac{(\text{NP}_{\eta\cdot 1}\text{-PP}_{\eta\cdot 2}{}^{a\star b\sqcup e\star f}) : (w_1, t_1 \sqcup t_2) \quad (\text{VP}_{\eta\cdot 3}{}^{c\star d}) : (w_2, t_3)}{(\text{S}_{\eta}{}^{a\star f}) : (w', t_1 t_3 t_2)} \tag{14}$$

In general, the target-side complexity of a binarized grammar with target arity t is still $\mathcal{T} = \mathcal{O}(|w|^{2(t+1)(m-1)})$ and the combined decoding complexity of the tree-based approach is

$$\mathcal{O}(|F|\mathcal{T}) = \mathcal{O}(|w|^{1+2(t+1)(m-1)}).$$

Table 2 shows that in this tree-based setting, target-side binarization has exactly the same performance with synchronous binarization while being much simpler to implement and does not have the problem of non-binarizability. The fact that simple binarization works (at least) equally well, which is not possible in string-based systems, is another advantage of the tree-based approaches.

4 Experiments

Section 3 shows that target-side binarization achieves the same polynomial decoding complexity as the more sophisticated synchronous binarization in the tree-based systems. We now empirically compare target-side binarization with an even simpler variant, *on-the-fly generation*, where the only difference is that the latter does target-side left-to-right binarization during +LM decoding on a hyperedge-per-hyperedge basis, without sharing common virtual nonterminals across hyperedges, while the former binarizes the whole −LM forest before the +LM decoding.

Our experiments are on English-to-Chinese translation in the tree-to-string system of Huang et al. (2006), which takes a source-language parse tree as input and tries to recursively convert it to a target-language string according to transfer rules in a synchronous grammar (Galley et al., 2006). For instance, the following rule

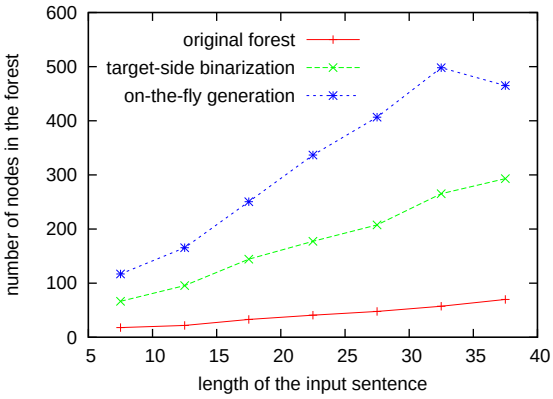
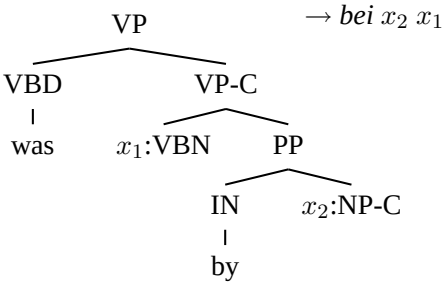


Figure 5: Number of nodes in the forests. Input sentences are grouped into bins according to their lengths (5-9, 10-14, 15-20, etc.).



translates an English passive construction into Chinese. Although the rules are actually in a synchronous tree-substitution grammar (STSG) instead of an SCFG, its derivation structure is still a hypergraph and all the analysis in Section 3.2 still applies. This system performs slightly better than the state-of-the-art phrase-based system Pharaoh (Koehn, 2004) on English to Chinese translation. A very similar system for the reverse direction is described in (Liu et al., 2006).

Our data preparation follows (Huang et al., 2006): the training data is a parallel corpus of 28.3M words on the English side, from which we extracted 24.7M tree-to-string rules using the algorithm of (Galley et al., 2006), and trained a Chinese trigram model on the Chinese side. We test our methods on the same test-set as in (Huang et al., 2006) which is a 140 sentence subset of NIST 2003 MT evaluation with 9–36 words on the English side. The weights for the log-linear model is tuned on a separate development set.

Figure 5 compares the number of nodes in the binarized forests against the original forest. On-the-fly generation essentially works on a larger forest with

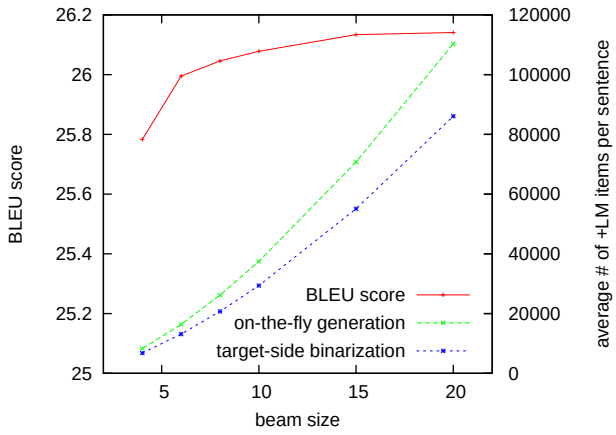


Figure 6: Decoding speed and BLEU scores under beam search.

duplicate nodes due to the lack of sharing, which is on average 1.85 times bigger than the target-side binarized forest. This difference is also reflected in the decoding speed, which is illustrated in Figure 6 under various beam settings and where the amount of computation is measured by the number of +LM items generated. At each individual beam setting, the two methods produce exactly the same set of translations (i.e., there is no relative search error), but the target-side binarization is consistently 1.3 times faster thanks to the sharing. In terms of translation quality, the final BLEU score at the largest beam setting is 0.2614, significantly higher than Pharaoh's 0.2354 as reported in (Huang et al., 2006).

5 Conclusion

This paper introduces a simple binarization scheme, *target-side binarization*, and presents a systematic study of the theoretical properties of the three binarization schemes in both string-based and tree-based systems using syncrhonous grammars. In particular, we show that target-side binarization achieves the same polynomial complexity as synchronous binarization while being much simpler to implement and universally applicable to arbitrary SCFGs. We also demonstrate the empirical effectiveness of this new scheme on a large-scale tree-to-string system.

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

David Chiang. 2007. Hierarchical phrase-based translation. In *Computational Linguistics*, volume 33. To appear.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of ACL (poster)*, pages 205–208.

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, Hao Zhang, and Daniel Gildea. 2005. Machine translation as lexicalized parsing with hooks. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT-2005)*.

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

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

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of ACL*.

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

I. Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of NAACL*.

Giorgio Satta and Enoch Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Proc. of HLT-EMNLP 2005*.

Stuart Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of COLING-ACL*.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of ACL*.

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