Asynchronous Binarization for Synchronous Grammars

John DeNero, Adam Pauls, and **Dan Klein**
Computer Science Division
University of California, Berkeley
{denero, adpauls, klein}@cs.berkeley.edu

Abstract

Binarization of n-ary rules is critical for the efficiency of syntactic machine translation decoding. Because the target side of a rule will generally reorder the source side, it is complex (and sometimes impossible) to find synchronous rule binarizations. However, we show that synchronous binarizations are not necessary in a two-stage decoder. Instead, the grammar can be binarized one way for the parsing stage, then rebinarized in a different way for the reranking stage. Each individual binarization considers only one monolingual projection of the grammar, entirely avoiding the constraints of synchronous binarization and allowing binarizations that are separately optimized for each stage. Compared to n-ary forest reranking, even simple target-side binarization schemes improve overall decoding accuracy.

1 Introduction

Syntactic machine translation decoders search over a space of synchronous derivations, scoring them according to both a weighted synchronous grammar and an n-gram language model. The rewrites of the synchronous translation grammar are typically flat, n-ary rules. Past work has *synchronously binarized* such rules for efficiency (Zhang et al., 2006; Huang et al., 2008). Unfortunately, because source and target orders differ, synchronous binarizations can be highly constrained and sometimes impossible to find.

Recent work has explored *two-stage* decoding, which explicitly decouples decoding into a source parsing stage and a target language model integration stage (Huang and Chiang, 2007). Because translation grammars continue to increase in size and complexity, both decoding stages require efficient approaches (DeNero et al., 2009). In this paper, we show how two-stage decoding enables independent binarizations for each stage. The source-side binarization guarantees cubic-time construction of a derivation forest, while an entirely different target-side binarization leads to efficient forest reranking with a language model.

Binarizing a synchronous grammar twice independently has two principal advantages over synchronous binarization. First, each binarization can be fully tailored to its decoding stage, optimizing the efficiency of both parsing and language model reranking. Second, the ITG constraint on non-terminal reordering patterns is circumvented, allowing the efficient application of synchronous rules that do not have a synchronous binarization. The primary contribution of this paper is to establish that binarization of synchronous grammars need not be constrained by cross-lingual reordering patterns. We also demonstrate that even simple target-side binarization schemes improve the search accuracy of forest reranking with a language model, relative to n-ary forest reranking.

2 Asynchronous Binarization

Two-stage decoding consists of parsing and language model integration. The parsing stage builds a pruned forest of derivations scored by the translation grammar only. In the second stage, this forest is reranked by an n-gram language model. We rerank derivations with *cube growing*, a lazy beam search algorithm (Huang and Chiang, 2007).

In this paper, we focus on syntactic translation with tree-transducer rules (Galley et al., 2006). These synchronous rules allow multiple adjacent non-terminals and place no restrictions on rule size or lexicalization. Two example unlexicalized rules appear in Figure 1, along with aligned and parsed training sentences that would have licensed them.

2.1 Constructing Translation Forests

The parsing stage builds a forest of derivations by parsing with the source-side projection of the synchronous grammar. Each forest node $\mathbf{P}_{ij}$ compactly encodes all parse derivations rooted by grammar symbol P and spanning the source sentence from positions i to j. Each derivation of $\mathbf{P}_{ij}$ is rooted by a rule with non-terminals that each

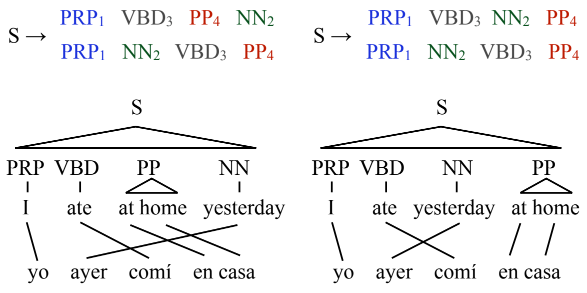

Figure 1: Two unlexicalized transducer rules (top) and aligned, parsed training sentences from which they could be extracted (bottom). The internal structure of English parses has been omitted, as it is irrelevant to our decoding problem.

anchor to some child node $\mathbf{C}_{k\ell}^{(t)}$, where the symbol $C^{(t)}$ is the tth child in the source side of the rule, and $i \leq k < \ell \leq j$.

We build this forest with a CKY-style algorithm. For each span (i, j) from small to large, and each symbol P, we iterate over all ways of building a node $\mathbf{P}_{ij}$, first considering all grammar rules with parent symbol P and then, for each rule, considering all ways of anchoring its non-terminals to existing forest nodes. Because we do not incorporate a language model in this stage, we need only operate over the source-side projection of the grammar.

Of course, the number of possible anchorings for a rule is exponential in the number of non-terminals it contains. The purpose of binarization during the parsing pass is to make this exponential algorithm polynomial by reducing rule branching to at most two non-terminals. Binarization reduces algorithmic complexity by eliminating redundant work: the shared substructures of n-ary rules are scored only once, cached, and reused. Caching is also commonplace in Early-style parsers that implicitly binarize when applying n-ary rules.

While any binarization of the source side will give a cubic-time algorithm, the particulars of a grammar transformation can affect parsing speed substantially. For instance, DeNero et al. (2009) describe normal forms particularly suited to transducer grammars, demonstrating that well-chosen binarizations admit cubic-time parsing algorithms while introducing very few intermediate grammar symbols. Binarization choice can also improve monolingual parsing efficiency (Song et al., 2008).

The parsing stage of our decoder proceeds by first converting the source-side projection of the translation grammar into lexical normal form (DeNero et al., 2009), which allows each rule to be applied to any span in linear time, then building a binary-branching translation forest, as shown in Figure 2(a). The intermediate nodes introduced during this transformation do not have a target-side projection or interpretation. They only exist for the sake of source-side parsing efficiency.

2.2 Collapsing Binarization

To facilitate a change in binarization, we transform the translation forest into n-ary form. In the n-ary forest, each hyperedge corresponds to an original grammar rule, and all nodes correspond to *original grammar symbols*, rather than those introduced during binarizaiton. Transforming the entire forest to n-ary form is intractable, however, because the number of hyperedges would be exponential in n. Instead, we include only the top k n-ary backtraces for each forest node. These backtraces can be enumerated efficiently from the binary forest. Figure 2(b) illustrates the result.

For efficiency, we follow DeNero et al. (2009) in pruning low-scoring nodes in the n-ary forest under the weighted translation grammar. We use a max-marginal threshold to prune unlikely nodes, which can be computed through a max-sum semiring variant of inside-outside (Goodman, 1996; Petrov and Klein, 2007).

Forest reranking with a language model can be performed over this n-ary forest using the cube growing algorithm of Huang and Chiang (2007). Cube growing lazily builds k-best lists of derivations at each node in the forest by filling a node-specific priority queue upon request from the parent. N-ary forest reranking serves as our baseline.

2.3 Reranking with Target-Side Binarization

Zhang et al. (2006) demonstrate that reranking over binarized derivations improves search accuracy by better exploring the space of translations within the strict confines of beam search. Binarizing the forest during reranking permits pairs of adjacent non-terminals in the target-side projection of rules to be rescored at intermediate forest nodes. This target-side binarization can be performed on-the-fly: when a node $\mathbf{P}_{ij}$ is queried for its k-best list, we binarize its n-ary backtraces.

Suppose $\mathbf{P}_{ij}$ can be constructed from a rule r with target-side projection

$$P \to \ell_0\ C_1\ \ell_1\ C_2\ \ell_2\ \ldots C_n\ \ell_n$$

where $C_1, \ldots, C_n$ are non-terminal symbols that are each anchored to a node $\mathbf{C}_{kl}^{(i)}$ in the forest, and ℓ_i are (possibly empty) sequences of lexical items.

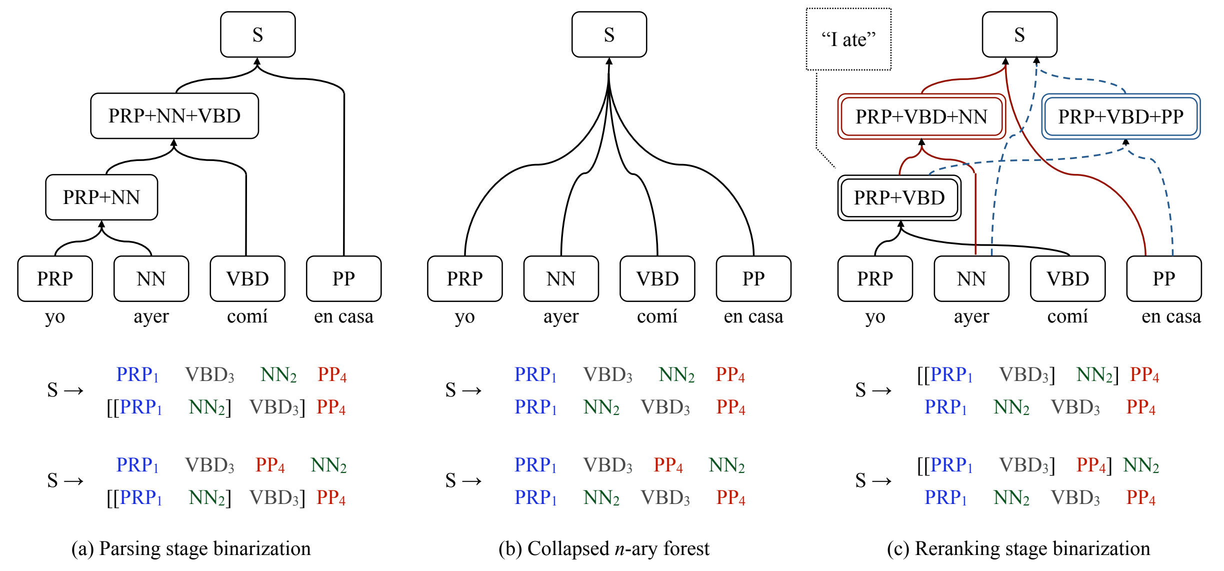
S
PRP+NN+VBD
PRP+NN
PRP NN VBD PP
yo ayer comí en casa
S → PRP1 VBD3 NN2 PP4 / [[PRP1 NN2] VBD3] PP4
S → PRP1 VBD3 PP4 NN2 / [[PRP1 NN2] VBD3] PP4
(a) Parsing stage binarization
S
PRP NN VBD PP
yo ayer comí en casa
S → PRP1 VBD3 NN2 PP4 / PRP1 NN2 VBD3 PP4
S → PRP1 VBD3 PP4 NN2 / PRP1 NN2 VBD3 PP4
(b) Collapsed n-ary forest
"I ate"
S
PRP+VBD+NN
PRP+VBD+PP
PRP+VBD
PRP NN VBD PP
yo ayer comí en casa
S → [[PRP1 VBD3] NN2] PP4 / PRP1 NN2 VBD3 PP4
S → [[PRP1 VBD3] PP4] NN2 / PRP1 NN2 VBD3 PP4
(c) Reranking stage binarization

Figure 2: A translation forest as it evolves during two-stage decoding, along with two n-ary rules in the forest that are rebinarized. (a) A source-binarized forest constructed while parsing the source sentence with the translation grammar. (b) A flat n-ary forest constructed by collapsing out the source-side binarization. (c) A target-binarized forest containing two derivations of the root symbol—the second is dashed for clarity. Both derivations share the node PRP+VBD, which will contain a single k-best list of translations during language model reranking. One such translation of PRP+VBD is shown: "I ate".

We apply a simple left-branching binarization to r, though in principle any binarization is possible. We construct a new symbol B and two new rules:

$$r_1 : \; B \rightarrow \ell_0 \; C_1 \; \ell_1 \; C_2 \; \ell_2$$
$$r_2 : \; P \rightarrow B \; C_3 \; \ell_3 \ldots C_n \; \ell_n$$

These rules are also anchored to forest nodes. Any C_i remains anchored to the same node as it was in the n-ary forest. For the new symbol B, we introduce a new forest node $\mathbf{B}$ that does not correspond to any particular span of the source sentence. We likewise transform the resulting r_2 until all rules have at most two non-terminal items. The original rule r from the n-ary forest is replaced by binary rules. Figure 2(c) illustrates the rebinarized forest.

Language model reranking treats the newly introduced forest node $\mathbf{B}$ as any other node: building a k-best derivation list by combining derivations from $\mathbf{C}^{(1)}$ and $\mathbf{C}^{(2)}$ using rule r_1. These derivations are made available to the parent of $\mathbf{B}$, which may be another introduced node (if more binarization were required) or the original root $\mathbf{P}_{ij}$.

Crucially, the ordering of non-terminals in the source-side projection of r does not play a role in this binarization process. The intermediate nodes $\mathbf{B}$ may comprise translations of discontiguous parts of the source sentence, as long as those parts are contained within the span (i, j).

2.4 Reusing Intermediate Nodes

The binarization we describe transforms the forest on a rule-by-rule basis. We must consider individual rules because they may contain different lexical items and non-terminal orderings. However, two different rules that can build a node often share some substructures. For instance, the two rules in Figure 2 both begin with PRP followed by VBD. In addition, these symbols are anchored to the same source-side spans. Thus, binarizing both rules yields the same intermediate forest node $\mathbf{B}$.

In the case where two intermediate nodes share the same intermediate rule anchored to the same forest nodes, they can be shared. That is, we need only generate one k-best list of derivations, then use it in derivations rooted by both rules. Sharing derivation lists in this way provides an additional advantage of binarization over n-ary forest reranking. Not only do we assess language model penalties over smaller partial derivations, but repeated language model evaluations are cached and reused across rules with common substructure.

3 Experiments

The utility of binarization for parsing is well known, and plays an important role in the efficiency of the parsing stage of decoding (DeNero et al., 2009). The benefit of binarization for language

Forest Reranked	**BLEU**	**Model Score**
N-ary baseline	58.2	41,543
Left-branching binary	**58.5**	**41,556**

Table 1: Reranking a binarized forest improves BLEU by 0.3 and model score by 13 relative to an n-ary forest baseline by reducing search errors during forest rescoring.

model reranking has also been established, both for synchronous binarization (Zhang et al., 2006) and for target-only binarization (Huang, 2007). In our experiment, we evaluate the benefit of target-side forest re-binarization in the two-stage decoder of DeNero et al. (2009), relative to reranking n-ary forests directly.

We translated 300 NIST 2005 Arabic sentences to English with a large grammar learned from a 220 million word bitext, using rules with up to 6 non-terminals. We used a trigram language model trained on the English side of this bitext. Model parameters were tuned with MERT. Beam size was limited to 200 derivations per forest node.

Table 1 shows a modest increase in model and BLEU score from left-branching binarization during language model reranking. We used the same pruned n-ary forest from an identical parsing stage in both conditions. Binarization did increase reranking time by 25% because more k-best lists are constructed. However, reusing intermediate edges during reranking binarization reduced binarized reranking time by 37%. We found that on average, intermediate nodes introduced in the forest are used in 4.5 different rules, which accounts for the speed increase.

4 Discussion

Asynchronous binarization in two-stage decoding allows us to select an appropriate grammar transformation for each language. The source transformation can optimize specifically for the parsing stage of translation, while the target-side binarization can optimize for the reranking stage.

Synchronous binarization is of course a way to get the benefits of binarizing both grammar projections; it is a special case of asynchronous binarization. However, synchronous binarization is constrained by the non-terminal reordering, limiting the possible binarization options. For instance, none of the binarization choices used in Figure 2 on either side would be possible in a synchronous binarization. There are rules, though rare, that cannot be binarized synchronously at all (Wu, 1997), but can be incorporated in two-stage decoding with asynchronous binarization.

On the source side, these limited binarization options may, for example, prevent a binarization that minimizes intermediate symbols (DeNero et al., 2009). On the target side, the speed of forest reranking depends upon the degree of reuse of intermediate k-best lists, which in turn depends upon the manner in which the target-side grammar projection is binarized. Limiting options may prevent a binarization that allows intermediate nodes to be maximally reused. In future work, we look forward to evaluating the wide array of forest binarization strategies that are enabled by our asynchronous approach.

References

John DeNero, Mohit Bansal, Adam Pauls, and Dan Klein. 2009. Efficient parsing for transducer grammars. In *Proceedings of the Annual Conference of the North American Association for Computational Linguistics*.

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 Annual Conference of the Association for Computational Linguistics*.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

Liang Huang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2008. Binarization of synchronous context-free grammars. *Computational Linguistics*.

Liang Huang. 2007. Binarization, synchronous binarization, and target-side binarization. In *Proceedings of the HLT-NAACL Workshop on Syntax and Structure in Statistical Translation (SSST)*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Xinying Song, Shilin Ding, and Chin-Yew Lin. 2008. Better binarization for the CKY parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

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

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.