Issues Concerning Decoding with Synchronous Context-free Grammar

Tagyoung Chung, **Licheng Fang** and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

We discuss some of the practical issues that arise from decoding with general synchronous context-free grammars. We examine problems caused by unary rules and we also examine how virtual nonterminals resulting from binarization can best be handled. We also investigate adding more flexibility to synchronous context-free grammars by adding glue rules and phrases.

1 Introduction

Synchronous context-free grammar (SCFG) is widely used for machine translation. There are many different ways to extract SCFGs from data. Hiero (Chiang, 2005) represents a more restricted form of SCFG, while GHKM (Galley et al., 2004) uses a general form of SCFG.

In this paper, we discuss some of the practical issues that arise from decoding general SCFGs that are seldom discussed in the literature. We focus on parsing grammars extracted using the method put forth by Galley et al. (2004), but the solutions to these issues are applicable to other general forms of SCFG with many nonterminals.

The GHKM grammar extraction method produces a large number of unary rules. Unary rules are the rules that have exactly one nonterminal and no terminals on the source side. They may be problematic for decoders since they may create cycles, which are unary production chains that contain duplicated dynamic programming states. In later sections, we discuss why unary rules are problematic and investigate two possible solutions.

GHKM grammars often have rules with many right-hand-side nonterminals and require binarization to ensure $O(n^3)$ time parsing. However, binarization creates a large number of virtual nonterminals. We discuss the challenges of, and possible solutions to, issues arising from having a large number of virtual nonterminals. We also compare binarizing the grammar with filtering rules according to *scope*, a concept introduced by Hopkins and Langmead (2010). By explicitly considering the effect of anchoring terminals on input sentences, scope-3 rules encompass a much larger set of rules than Chomsky normal form but they can still be parsed in $O(n^3)$ time.

Unlike phrase-based machine translation, GHKM grammars are less flexible in how they can segment sentence pairs into phrases because they are restricted not only by alignments between words in sentence pairs, but also by target-side parse trees. In general, GHKM grammars suffer more from data sparsity than phrasal rules. To alleviate this issue, we discuss adding glue rules and phrases extracted using methods commonly used in phrase-based machine translation.

2 Handling unary rules

Unary rules are common in GHKM grammars. We observed that as many as 10% of the rules extracted from a Chinese-English parallel corpus are unary.

Some unary rules are the result of alignment errors, but other ones might be useful. For example, Chinese lacks determiners, and English determiners usually remain unaligned to any Chinese words. Extracted grammars include rules that reflect this fact:

NP → NP, the NP
NP → NP, a NP

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:shortpapers, pages 413–417,
Portland, Oregon, June 19-24, 2011.

However, unary rules can be problematic:

- Unary production cycles corrupt the translation hypergraph generated by the decoder. A hypergraph containing a unary cycle cannot be topologically sorted. Many algorithms for parameter tuning and coarse-to-fine decoding, such as the inside-outside algorithm and cube-pruning, cannot be run in the presence of unary cycles.

- The existence of many unary rules of the form "NP → NP, the NP" quickly fills a pruning bin with guesses of English words to insert without any source-side lexical evidence.

The most obvious way of eliminating problematic unary rules would be converting grammars into Chomsky normal form. However, this may result in bloated grammars. In this section, we present two different ways to handle unary rules. The first involves modifying the grammar extraction method, and the second involves modifying the decoder.

2.1 Modifying grammar extraction

We can modify the grammar extraction method such that it does not extract any unary rules. Galley et al. (2004) extracts rules by segmenting the target-side parse parse tree based on *frontier nodes*. We modify the definition of a frontier node in the following way. We label frontier nodes in the English parse tree, and examine the Chinese span each frontier node covers. If a frontier node covers the same span as the frontier node that immediately dominates it, then the dominated node is no longer considered a frontier. This modification prevents unary rules from being extracted.

Figure 1 shows an example of an English-Chinese sentence pair with the English side automatically parsed. Frontier nodes in the tree in the original GHKM rule extraction method are marked with a box. With the modification, only the top boldfaced **NP** would be considered a frontier node. The GHKM rule extraction results in the following rules:

NPB → 白鹭 鸶, the snowy egret
NP → NPB, NPB
PP → NP, with NP
NP → PP, romance PP

With the change, only the following rule is extracted:

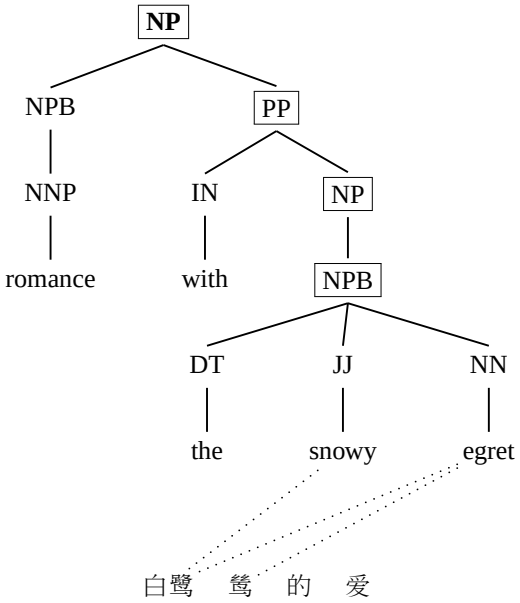

Figure 1: A sentence fragment pair with erroneous alignment and tokenization

NP → 白鹭 鸶, romance with the snowy egret

We examine the effect of this modification has on translation performance in Section 5.

2.2 Modifying the decoder

Modifying how grammars are extracted has an obvious down side, i.e., the loss of generality. In the previous example, the modification results in a bad rule, which is the result of bad alignments. Before the modification, the rule set includes a good rule:

NPB → 白鹭 鸶, the snowy egret

which can be applied at test time. Because of this, one may still want to decode with all available unary rules. We handle unary rules inside the decoder in the following ways:

- Unary cycle detection

  The naïve way to detect unary cycles is backtracking on a unary chain to see if a newly generated item has been generated before. The running time of this is constrained only by the number of possible items in a chart span. In practice, however, this is often not a problem: if all unary derivations have positive costs and a priority queue is used to expand unary derivations,

only the best K unary items will be generated, where K is the pruning constant.

- Ban negative cost unary rules

  When tuning feature weights, an optimizer may try feature weights that may give negative costs to unary productions. This causes unary derivations to go on forever. The solution is to set a maximum length for unary chains, or to ban negative unary productions outright.

3 Issues with binarization

3.1 Filtering and binarization

Synchronous binarization (Zhang et al., 2006) is an effective method to reduce SCFG parsing complexity and allow early language model integration. However, it creates virtual nonterminals which require special attention at parsing time. Alternatively, we can filter rules that have more than scope-3 to parse in $O(n^3)$ time with unbinarized rules. This requires Earley (Earley, 1970) style parsing, which does implicit binarization at decoding time. Scope-filtering may filter out unnecessarily long rules that may never be applied, but it may also throw out rules with useful contextual information. In addition, scope-filtering does not accommodate early language model state integration. We compare the two with an experiment. For the rest of the section, we discuss issues created by virtual nonterminals.

3.2 Handling virtual nonterminals

One aspect of grammar binarization that is rarely mentioned is how to assign probabilities to binarized grammar rules. The naïve solution is to assign probability one to any rule whose left-hand side is a virtual nonterminal. This maintains the original model. However, it is generally not fair to put chart items of virtual nonterminals and those of regular nonterminals in the same bin, because virtual items have artificially low costs. One possible solution is adding a heuristic to push up the cost of virtual items for fair comparison.

For our experiments, we use an outside estimate as a heuristic for a virtual item. Consider the following rule binarization (only the source side shown):

$$\text{A} \rightarrow \text{BCD} : -\log(p) \quad \Rightarrow \quad \begin{array}{l} \text{V} \rightarrow \text{BC} : 0 \\ \text{A} \rightarrow \text{VD} : -\log(p) \end{array}$$

$\text{A} \rightarrow \text{BCD}$ is the orginal rule and $-\log(p)$ is the cost of the rule. In decoding time, when a chart item is generated from the binarized rule $\text{V} \rightarrow \text{BC}$, we add $-\log(p)$ to its total cost as an optimistic estimate of the cost to build the original unbinarized rule. The heuristic is used only for pruning purposes, and it does not change the real cost. The idea is similar to A* parsing (Klein and Manning, 2003). One complication is that a binarized rule can arise from multiple different unbinarized rules. In this case, we pick the lowest cost among the unbinarized rules as the heuristic.

Another approach for handling virtual nonterminals would be giving virtual items separate bins and avoiding pruning them at all. This is usually not practical for GHKM grammars, because of the large number of nonterminals.

4 Adding flexibility

4.1 Glue rules

Because of data sparsity, an SCFG extracted from data may fail to parse sentences at test time. For example, consider the following rules:

$\text{NP} \rightarrow \text{JJ NN}, \text{JJ NN}$
$\text{JJ} \rightarrow \text{c}_1, \text{e}_1$
$\text{JJ} \rightarrow \text{c}_2, \text{e}_2$
$\text{NN} \rightarrow \text{c}_3, \text{e}_3$

This set of rules is able to parse the word sequence $\text{c}_1\ \text{c}_3$ and $\text{c}_2\ \text{c}_3$ but not $\text{c}_1\ \text{c}_2\ \text{c}_3$, if we have not seen "$\text{NP} \rightarrow \text{JJ JJ NN}$" at training time. Because SCFGs neither model adjunction, nor are they markovized, with a small amount of data, such problems can occur. Therefore, we may opt to add glue rules as used in Hiero (Chiang, 2005):

$\text{S} \rightarrow \text{C}, \text{C}$
$\text{S} \rightarrow \text{S C}, \text{S C}$

where S is the goal state and C is the glue nonterminal that can produce any nonterminals. We refer to these glue rules as the monotonic glue rules. We rely on GHKM rules for reordering when we use the monotonic glue rules. However, we can also allow glue rules to reorder constituents. Wu (1997) presents a better-constrained grammar designed to only produce tail-recursive parses. See Table 1 for the complete set of rules. We refer to these rules as ABC glue rules. These rules always generate left-

S → A	A → [A B]	B → ⟨ B A ⟩
S → B	A → [B B]	B → ⟨ A A ⟩
S → C	A → [C B]	B → ⟨ C A ⟩
	A → [A C]	B → ⟨ B C ⟩
	A → [B C]	B → ⟨ A C ⟩
	A → [C C]	B → ⟨ C C ⟩

Table 1: The ABC Grammar. We follow the convention of Wu (1997) that square brackets stand for straight rules and angle brackets stand for inverted rules.

heavy derivations, weeding out ambiguity and making search more efficient. We learn probabilities of ABC glue rules by using expectation maximization (Dempster et al., 1977) to train a word-level Inversion Transduction Grammar from data.

In our experiments, depending on the configuration, the decoder failed to parse about 5% of sentences without glue rules, which illustrates their necessity. Although it is reasonable to believe that reordering should always have evidence in data, as with GHKM rules, we may wish to reorder based on evidence from the language model. In our experiments, we compare the ABC glue rules with the monotonic glue rules.

4.2 Adding phrases

GHKM grammars are more restricted than the phrase extraction methods used in phrase-based models, since, in GHKM grammar extraction, phrase segmentation is constrained by parse trees. This may be a good thing, but it suffers from loss of flexibility, and it also cannot use non-constituent phrases. We use the method of Koehn et al. (2003) to extract phrases, and, for each phrase, we add a rule with the glue nonterminal as the left-hand side and the phrase pair as the right-hand side. We experiment to see whether adding phrases is beneficial.

There have been other efforts to extend GHKM grammar to allow more flexible rule extraction. Galley et al. (2006) introduce composed rules where minimal GHKM rules are fused to form larger rules. Zollmann and Venugopal (2006) introduce a model that allows more generalized rules to be extracted.

	BLEU
Baseline + monotonic glue rules	20.99
No-unary + monotonic glue rules	23.83
No-unary + ABC glue rules	23.94
No-unary (scope-filtered) + monotonic	23.99
No-unary (scope-filtered) + ABC glue rules	24.09
No-unary + ABC glue rules + phrases	23.43

Table 2: BLEU score results for Chinese-English with different settings

5 Experiments

5.1 Setup

We extracted a GHKM grammar from a Chinese-English parallel corpus with the English side parsed. The corpus consists of 250K sentence pairs, which is 6.3M words on the English side. Terminal-aware synchronous binarization (Fang et al., 2011) was applied to all GHKM grammars that are not scope-filtered. MERT (Och, 2003) was used to tune parameters. We used a 392-sentence development set with four references for parameter tuning, and a 428-sentence test set with four references for testing. Our in-house decoder was used for experiments with a trigram language model. The decoder is capable of both CNF parsing and Earley-style parsing with cube-pruning (Chiang, 2007).

For the experiment that incorporated phrases, the phrase pairs were extracted from the same corpus with the same set of alignments. We have limited the maximum size of phrases to be four.

5.2 Results

Our result is summarized in Table 2. The baseline GHKM grammar with monotonic glue rules yielded a worse result than the no-unary grammar with the same glue rules. The difference is statistically significant at $p < 0.05$ based on 1000 iterations of paired bootstrap resampling (Koehn, 2004).

Compared to using monotonic glue rules, using ABC glue rules brought slight improvements for both the no-unary setting and the scope-filtered setting, but the differences are not statistically significant. In terms of decoding speed and memory usage, using ABC glues and monotonic glue rules were virtually identical. The fact that glue rules are seldom used at decoding time may account for why there is

little difference in using monotonic glue rules and using ABC glue rules. Out of all the rules that were applied to decoding our test set, less than one percent were glue rules, and among the glue rules, straight glue rules outnumbered inverted ones by three to one.

Compared with binarized no-unary rules, scope-3 filtered no-unary rules retained 87% of the rules but still managed to have slightly better BLEU score. However, the score difference is not statistically significant. Because the size of the grammar is smaller, compared to using no-unary grammar, it used less memory at decoding time. However, decoding speed was somewhat slower. This is because the decoder employs Early-style dotted rules to handle unbinarized rules, and in order to decode with scope-3 rules, the decoder needs to build dotted items, which are not pruned until a rule is completely matched, thus leading to slower decoding.

Adding phrases made the translation result slightly worse. The difference is not statistically significant. There are two possible explanations for this. Since there were more features to tune, MERT may have not done a good job. We believe the more important reason is that once a phrase is used, only glue rules can be used to continue the derivation, thereby losing the richer information offered by GHKM grammar.

6 Conclusion

In this paper, we discussed several issues concerning decoding with synchronous context-free grammars, focusing on grammars resulting from the GHKM extraction method. We discussed different ways to handle cycles. We presented a modified grammar extraction scheme that eliminates unary rules. We also presented a way to decode with unary rules in the grammar, and examined several different issues resulting from binarizing SCFGs. We finally discussed adding flexibility to SCFGs by adding glue rules and phrases.

Acknowledgments We would like to thank the anonymous reviewers for their helpful comments. This work was supported by NSF grants IIS-0546554 and IIS-0910611.

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL-05*, pages 263–270, Ann Arbor, MI.

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

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society*, 39(1):1–21.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 6(8):451–455.

Licheng Fang, Tagyoung Chung, and Daniel Gildea. 2011. Terminal-aware synchronous binarization. In *Proceedings of the ACL 2011 Conference Short Papers*, Portland, Oregon, June. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of 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 COLING/ACL-06*, pages 961–968, July.

Mark Hopkins and Greg Langmead. 2010. SCFG decoding without binarization. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 646–655, Cambridge, MA, October. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of NAACL-03*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL-03*, Edmonton, Alberta.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395, Barcelona, Spain, July.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of ACL-03*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of NAACL-06*, pages 256–263, New York, NY.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. Workshop on Statistical Machine Translation*, pages 138–141.