Two monolingual parses are better than one (synchronous parse)*

Chris Dyer
UMIACS Laboratory for Computational Linguistics and Information Processing
Department of Linguistics
University of Maryland, College Park, MD 20742, USA
redpony AT umd.edu

Abstract

We describe a synchronous parsing algorithm that is based on two successive monolingual parses of an input sentence pair. Although the worst-case complexity of this algorithm is and must be $O(n^6)$ for binary SCFGs, its average-case run-time is far better. We demonstrate that for a number of common synchronous parsing problems, the two-parse algorithm substantially outperforms alternative synchronous parsing strategies, making it efficient enough to be utilized without resorting to a pruned search.

1 Introduction

Synchronous context free grammars (SCFGs) generalize monolingual context-free grammars to generate strings concurrently in pairs of languages (Lewis and Stearns, 1968) in much the same way that finite state transducers (FSTs) generalize finite state automata (FSAs).¹ Synchronous parsing is the problem of finding the best derivation, or forest of derivations, of a source and target sentence pair $(f, e)$ under an SCFG, $G$.² Solving this problem is necessary for several applications, for example, optimizing how well an SCFG translation model fits parallel training data. Wu (1997) describes a bottom-up $O(n^6)$ synchronous parsing algorithm for ITGs, a binary SCFG with a restricted form. For general grammars, the situation is even worse: the problem has been shown to be NP-hard (Satta and Peserico, 2005). Even if we restrict ourselves to binary ITGs, the $O(n^6)$ run-time makes large-scale learning applications infeasible. The usual solution is to use a heuristic search that avoids exploring edges that are likely (but not guaranteed) to be low probability (Zhang et al., 2008; Haghighi et al., 2009). In this paper, we derive an alternative synchronous parsing algorithm starting from a conception of parsing with SCFGs as a composition of binary relations. This enables us to factor the synchronous parsing problem into two successive monolingual parses. Our algorithm runs more efficiently than $O(n^6)$ with many grammars (including those that required using heuristic search with other parsers), making it possible to take advantage of synchronous parsing without developing search heuristics; and the SCFGs are not required to be in a normal form, making it possible to easily parse with more complex SCFG types.

2 Synchronous parsing

Before presenting our algorithm, we review the $O(n^6)$ synchronous parser for binary ITGs.³

2.1 ITG synchronous parsing algorithm

Wu (1997) describes a bottom-up synchronous parsing algorithm that can be understood as a generalization of the CKY algorithm. CKY defines a table consisting of $n^2$ cells, with each cell corresponding to a span $[i, j]$ in the input sentence; and the synchronous variant defines a table in 4 dimensions, with cells corresponding to a source span $[s, t]$ and a target span $[u, v]$. The bottom of the chart is initialized first, and pairs of items are combined from bottom to top. Since combining items from the $n^4$ cells involves considering two split points (one source, one target), it is not hard to see that this algorithm runs in time $O(n^6)$.

---

¹This work was supported in part by the GALE program of DARPA, Contract No. HR0011-06-2-001. The author wishes to thank Philip Rensik, Adam Lopez, Phil Blunsom, and Jason Eisner for helpful discussions.

²SCFGs have enjoyed a resurgence in popularity as the formal basis for a number of statistical translation systems, e.g. Chiang (2007). However, translation requires only the manipulation of SCFGs using monolingual parsing algorithms.

³Generalizing the algorithm to higher rank grammars is possible (Wu, 1997), as is converting a grammar to a weakly equivalent binary form in some cases (Huang et al., 2009).
2.2 Parsing, intersection, and composition

We motivate an alternative conception of the synchronous parsing problem as follows. It has long been appreciated that monolingual parsing computes the intersection of an FSA and a CFG (Bar-Hillel et al., 1961; van Noord, 1995). That is, if \( S \) is an FSA encoding some sentence \( s \), intersection of \( S \) with a CFG, \( G \), results in a parse forest which contains all and only derivations of \( s \), that is \( L(S) \cap L(G) \in \{\{s\}, \emptyset\} \).\(^4\) Crucially for our purposes, the resulting parse forest is also itself a CFG.\(^5\) Figure 1 illustrates, giving two equivalent representations of the forest \( S \cap G \), once as a directed hypergraph and once as a CFG. While \( S \cap G \) appears similar to \( G \), the non-terminals (NTs) of the resulting CFG are a cross product of pairs of states from \( S \) and NTs from \( G \).\(^6\)

\[
G
\]

\[
S \cap G
\]

Figure 1: A CFG, \( G \), an FSA, \( S \), encoding a sentence, and two equivalent representations of the parse forest \( S \cap G \), (a) as a directed hypergraph and (b) as a CFG.

When dealing with SCFGs, rather than intersect-

\(^4\)\(L(x)\) denotes the set of strings generated by the grammatical automaton \( x \). In future mentions of intersection and composition operations, this will be implicit.

\(^5\)The forest grammar derives only \( s \), but using possibly many derivations.

\(^6\)Each pair of states from the FSA corresponds to a span \([i, j]\) in a CKY table.

tion, parsing computes a related operation, composition.\(^7\) The standard MT decoding-by-parsing task can be understood as computing the composition of an FST,\(^8\) \( F \), which encodes the source sentence \( f \) with the SCFG, \( G \), representing the translation model. The result is the translation forest, \( F \circ G \), which encodes all translations of \( f \) licensed by the translation model. While \( G \) can generate a potentially infinite set of strings in the source and target languages, \( F \circ G \) generates only \( f \) in the source language (albeit with possibly infinitely many derivations), but any number of different strings in the target language. It is not hard to see that a second composition operation of an FST, \( E \), encoding the target string \( e \) with the \( e \)-side of \( F \circ G \) (again using a monolingual parsing algorithm), will result in a parse forest that exactly derives \((f, e)\), which is the goal of synchronous composition. Figure 2 shows an example. In \( F \circ G \circ E \) the NTs (nodes) are the cross product of pairs of states from \( E \), the NTs from \( G \), and pairs of states in \( F \).

Thus, synchronous parsing is the task of computing \( F \circ G \circ E \). Since composition is associative, we can compute this quantity either as \((F \circ G) \circ E\) or \(F \circ (G \circ E)\). Alternatively, we can use an algorithm that performs 3-way composition directly.

2.3 The two-parse algorithm\(^9\)

The two-parse algorithm refers to performing a synchronous parse by computing either \((F \circ G) \circ E\) or \(F \circ (G \circ E)\). Each composition operation is carried out using a standard monolingual parsing algorithm, such as Earley’s or CKY. In the experiments below, since we use \( e \)-free grammars, we use a variant of CKY for unrestricted CFGs (Chiang, 2007).

Once the first composition is done, the resulting parse forest must be converted into a CFG representation that the second parser can utilize. This is straightforward to do: each node becomes a unique non-terminal symbol, with its incoming edges corresponding to different ways of rewriting it. Tails of edges are non-terminal variables in the RHS of these rewrites. A single bottom-up traversal of the forest is sufficient to perform the conversion. Since

\(^7\)Intersection is a special case of composition where the input and output labels on the transducers are identical (Mohri, 2009).

\(^8\)FSTs used to represent the source and target sentences have identical input and output labels on every transition.

\(^9\)Satta (submitted) has independently derived this algorithm.
The translation task is a straightforward ma-
a resurgence in popularity as the formal basis for
finite state transducers (FSTs) generalize finite state
Synchronous context free grammars (SCFGs) gener-

1 Introduction

rithm a natural fit when those training regimes
in the original algorithm become feasible, and
ing strategies that would be difficult to realize
chronous parsing problems, the two-parse al-
ably can not) improve the worst-case run-
is based on two successive

Figure 1: A CFG, 

put and output labels on the transducers are identical (Mohri,
SCFG,

Figure 2 plots the average runtime of the algorithm
compute sufficient statistics for a variety of machine
required synchronous parses to discriminatively train

3 Experiments

We now describe two different synchronous parsing
applications, with different classes of SCFGs, and
compare the performance of the two-parse algorithm
with that of previously used algorithms.

Phrasal ITGs. Here we compare performance of
the two-parse algorithm and the $O(n^6)$ ITG parsing
algorithm on an Arabic-English phrasal ITG align-
task. We used a variant of the phrasal ITG de-
scribed by Zhang et al. (2008).12 Figure 3 plots the
average run-time of the two algorithms as a function
of the Arabic sentence length. The two-parse ap-
proach is far more efficient. In total, aligning the 80k
sentence pairs in the corpus completed in less than
4 hours with the two-parse algorithm but required
more than 1 week with the baseline algorithm.13

“Hiero” grammars. An alternative approach to
computing a synchronous parse forest is based on
cube pruning (Huang and Chiang, 2007). While
more commonly used to integrate a target $m$-gram
LM during decoding, Blunsom et al. (2008), who
required synchronous parses to discriminatively train

10How tight these bounds are depends on the ambiguity in
the grammar w.r.t. the input: to generate $n^3$ edges, every item
in every cell must be derivable by every combination of its sub-
spans. Most grammars are substantially less ambiguous.

11Since many widely used SCFGs meet these criteria, in-
cluding hierarchical phrase-based translation grammars (Chi-
ang, 2007), SAMT grammars (Zollmann and Venugopal, 2006),
and phrasal ITGs (Zhang et al., 2008), a detailed analysis of e-
containing and higher rank grammars is left to future work.

12The restriction that phrases contain exactly a single align-
ment point was relaxed, resulting in much larger and more
ambiguous grammars than those used in the original work.

13A note on implementation: our ITG aligner was minimal; it
only computed the probability of the sentence pair using the in-
side algorithm. With the two-parse aligner, we stored the com-
plete forest during both the first and second parses.
Figure 3: Average synchronous parser run-time (in seconds) as a function of Arabic sentence length (in words).

an SCFG translation model, repurposed this algorithm to discard partial derivations during translation if the derivation yielded a target m-gram not found in e (p.c.). We replicated their BTEC Chinese-English baseline system and compared the speed of their ‘cube-parsing’ technique and our two-parse algorithm. The SCFG used here was extracted from a word-aligned corpus, as described in Chiang (2007). The following table compares the average per sentence synchronous parse time.

| Algorithm                  | avg. run-time (sec) |
|----------------------------|----------------------|
| Blunsom et al. (2008)      | 7.31                 |
| this work                  | 0.20                 |

4 Discussion

Thinking of synchronous parsing as two composition operations has both conceptual and practical benefits. The two-parse strategy can outperform both the ITG parsing algorithm (Wu, 1997), as well as the ‘cube-parsing’ technique (Blunsom et al., 2008). The latter result points to a connection with recent work showing that determinization of edges before LM integration leads to fewer search errors during decoding (Iglesias et al., 2009).

Our results are somewhat surprising in light of work showing that 3-way composition algorithms for FSTs operate far more efficiently than performing successive pairwise compositions (Allauzen and Mohri, 2009). This is certainly because the 3-way algorithm used here (the ITG algorithm) does an exhaustive search over all $n^4$ span pairs without awareness of any top-down constraints. This suggests that faster composition algorithms that incorporate top-down filtering may still be discovered.

References

C. Allauzen and M. Mohri. 2009. N-way composition of weighted finite-state transducers. *International Journal of Foundations of Comp. Sci.*, 20(4):613–627.
Y. Bar-Hillel, M. Perles, and E. Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172.
P. Blunsom, T. Cohn, and M. Osborne. 2008. Probabilistic inference for machine translation. In *EMNLP*.
D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.
S. L. Graham, W. L. Ruzzo, and M. Harrison. 1980. An improved context-free recognizer. *ACM Trans. Program. Lang. Syst.*, 2(3):415–462.
A. Haghighi, J. Blitzer, J. DeNero, and D. Klein. 2009. Better word alignments with supervised ITG models. In *Proc. of ACL/ICCNLP*, pages 923–931.
L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *ACL*.
L. Huang, H. Zhang, D. Gildea, and K. Knight. 2009. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4).
G. Iglesias, A. de Gispert, E. R. Banga, and W. Byrne. 2009. Hierarchical phrase-based translation with weighted finite state transducers. In *Proc. of ACL/ICCNLP*, pages 923–931.
P. M. Lewis, II and R. E. Stearns. 1968. Syntax-directed transduction. *J. ACM*, 15(3):465–488.
M. Mohri. 2009. Weighted automata algorithms. In M. Drost, W. Kuich, and H. Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science, pages 213–254. Springer.
G. Satta and E. Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Proceedings of NAACL*.
G. Satta. submitted. Translation algorithms by means of language intersection.
G. van Noord. 1995. The intersection of finite state automata and definite clause grammars. In *Proc. of ACL*.
D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.
H. Zhang, C. Quirk, R. C. Moore, and D. Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proceedings of ACL*.
A. Zollmann and A. Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. of the Workshop on SMT*.