5th Workshop on Tree-Adjoining Grammars and Related Formalisms (TAG+5), Paris, 25–27 May 2000

A Faster Parsing Algorithm for Lexicalized Tree-Adjoining Grammars

Jason Eisner† and Giorgio Satta‡

†Dept. of Computer Science
University of Rochester
P.O. Box 270226
Rochester, NY 14627-0226 USA
jason@cs.rochester.edu

‡Dip. di Elettronica e Informatica
Università di Padova
via Gradenigo 6/A
I-35131 Padova, Italy
satta@dei.unipd.it

Abstract

This paper points out some computational inefficiencies of standard TAG parsing algorithms when applied to LTAGs. We propose a novel algorithm with an asymptotic improvement, from $\mathcal{O}(n^8 g^2 t)$ *to* $\mathcal{O}(n^6 \max(n, g) g t)$*, where* n *is the input length and* g, t *are grammar constants that are independent of vocabulary size.*

Introduction

Lexicalized Tree-Adjoining Grammars (LTAGs) were first introduced in (Schabes *et al.*, 1988) as a variant of Tree-Adjoining Grammars (TAGs) (Joshi, 1987). In LTAGs each elementary tree is specialized for some individual lexical item. Following the original proposal, LTAGs have been used in several state-of-the-art, real-world parsers; see for instance (Abeillé & Candito, 2000) and (Doran *et al.*, 2000).

Like link grammar (Sleator & Temperley, 1991) and lexicalized formalisms from the statistical parsing literature (Collins, 1997; Charniak, 1997; Alshawi, 1996; Eisner, 1996) LTAGs provide two main recognized advantages over more standard non-lexicalized formalisms:

- subcategorization can be specified separately for each word; and
- each word can restrict the anchors (head words) of its arguments and adjuncts, encoding lexical preferences as well as some effects of semantics and world knowledge.

To give a simple example, consider the verb *walk*, which is usually intransitive but can take an object in some restricted cases. An LTAG can easily specify the acceptability of sentence *Mary walks the dog* by associating *walk* with a transitive elementary tree that selects for an indirect object tree anchored at word *dog* (and some other words within a limited range).

LTAGs are large because they include separate trees for each word in the vocabulary. However, parsing need consider only those trees of the grammar that are associated with the lexical symbols in the input string. While this strategy reduces parsing time in all practical cases, since the input string length tends to be considerably smaller than the grammar size, it also introduces an additional factor in the runtime that depends on the input string length. This problem was recognized early in (Schabes *et al.*, 1988), but a precise computational analysis has not been provided in the literature, up to the authors' knowledge. See (Eisner, 1997; Eisner, 2000) for related discussion on different lexicalized formalisms.

In this work we reconsider LTAG parsing in the above perspective. Under standard assumptions, we show that existing LTAG parsing methods perform with $\mathcal{O}(tg^2\,|w|^8)$ worst case running time, where t and g are smallish constants depending on the grammar and w is the input string. As our main result we present an $\mathcal{O}(tg\,|w|^6\max\{g,|w|\})$ time algorithm, a considerable improvement over the standard LTAG parsing methods.

1. The problem

We assume the reader is familiar with TAGs, LTAGs and related notions (Joshi, 1987; Joshi & Schabes, 1992). Each node n in an elementary tree is associated with a selectional constraint $Adj(n)$ representing a (possibly empty) set of elementary trees that can be adjoined at n (we treat substitution as adjunction at a childless node). We define the size of n as $1+|Adj(n)|$. The size of an LTAG G, written $|G|$, is the sum of the sizes of all nodes in the elementary trees of G.

Standard parsing algorithms for TAGs have running time $\mathcal{O}(|G|\,|w|^6)$, where G is the input grammar and w is the input string. As already mentioned in the introduction, LTAGs allow more selective parsing strategies, resulting in $\mathcal{O}(f(G,w)\,|w|^6)$ running time, for some function $f(G,w)$ that is independent of the size of the vocabulary treated by G (hence typically much less than $|G|$). In order to give an estimate of the factor $f(G,w)$, let us define t as the maximum number of nodes in an elementary tree (of G), and g as the maximum number of elementary trees that are anchored in a common lexical item. We argue below that $f(G,w)$ is $\mathcal{O}(tg^2\,|w|^2)$.

We need to introduce some additional notation. We write $w_{i,j}$ to denote the substring of w from position i to position j, $0\le i\le j\le |w|$. (Position i is the boundary between the i-th and the $(i+1)$-th symbols of w.) We write w_i for $w_{i-1,i}$. In the grammar, assume some arbitrary ordering for the elementary trees with a given anchor and for the nodes of each elementary tree, with the root node always being the first. Then $\langle h,k\rangle$ denotes the k-th elementary tree anchored at w_h, $\langle h,k,1\rangle$ denotes its root node, and $\langle h,k,m\rangle$ denotes its m-th node (for $1\le h\le |w|$, $1\le k\le g$, $1\le m\le t$).

By "tree" we now mean an elementary or derived tree that may contain a foot-node. The most time-expensive step in TAG and LTAG tabular parsing is the recognition of adjunction at nodes dominating a foot-node. Say that we have constructed a subtree that is rooted at the node $\langle h,k,m\rangle$, which may be an *internal* node of some elementary tree, and covers substrings $w_{i,p}$ and $w_{q,j}$. Say also that we have constructed a complete tree β rooted at $\langle h',k',1\rangle$, covering substrings $w_{i',i}$ and $w_{j,j'}$. In a tabular method these two analyses can be represented, respectively, by the items $[\langle h,k,m\rangle,i,p,q,j]$ and $[\langle h',k',1\rangle_\top,i',i,j,j']$.[1] In items, the subscript $\top$ on a node indicates that no further adjunction is allowed to take place there (i.e., adjunction has already

[1]Top-down tabular algorithms, and those that enforce the valid-prefix property, might use more indices in item representations, in addition to those shown in our example. In some cases this may damage the asymptotic runtime.

occurred or has been explicitly declined). Adjunction of β at the node $\langle h,k,m\rangle$ is then carried out as illustrated by the following abstract inference rule (see for instance (Vijay-Shanker & Joshi, 1985; Vijay-Shanker & Weir, 1993)):

$$\frac{[\langle h,k,m\rangle,i,p,q,j] \quad [\langle h',k',1\rangle_\top,i',i,j,j']}{[\langle h,k,m\rangle_\top,i',p,q,j']}\ \langle h',k'\rangle \in \mathit{Adj}(\langle h,k,m\rangle) \tag{1}$$

Item $[\langle h,k,m\rangle_\top,i',p,q,j']$ represents a new partial analysis spanning $w_{i',p}$ and $w_{q,j'}$; no further adjunction is possible at node $\langle h,k,m\rangle$ in this analysis.

In order to bound $f(G,w)$, let us fix positions i',i,p,q,j and j'. Then step (1) can be executed a number of times bounded by $((i-i')+(j'-j))(p+|w|-q)tg^2$. This is because $w_{h'}$ can freely range within $w_{i',i}$ or $w_{j,j'}$, w_h can freely range within $w_{0,p}$ or $w_{q,|w|}$, since the anchor w_h of tree $\langle h,k\rangle$ might not be dominated by node $\langle h,k,m\rangle$; also, k, k' and m can assume any values within their respective ranges. We therefore conclude that $f(G,w)=\mathcal{O}(tg^2\,|w|^2)$.

Note that a better upper bound would be given by $\mathcal{O}(tg^2\min\{|V_\mathrm{T}|^2,|w|^2\})$, V_T the terminal alphabet (vocabulary) of G, since each anchor can assume no more than $|V_\mathrm{T}|$ different values. However, in practical applications we have $|w| \ll |V_\mathrm{T}|$, and therefore in this paper we will always use the former bound. We then conclude that standard LTAG parsing algorithms run with a worst case time of $\mathcal{O}(tg^2\,|w|^8)$.

2. A novel algorithm

This section improves upon the time upper bound reported in §1. The result is achieved by splitting step (1) into three substeps. (A similar method may be applied to speed up parsing of lexicalized context-free grammars (Eisner & Satta, 1999).)

We start by observing that at step (1) we simultaneously carry out two tests on the trees under analysis:

- we check that the tree $\langle h',k'\rangle$ is found in the selectional constraint $\mathit{Adj}(\langle h,k,m\rangle)$; and
- we check that the tree yield $w_{i',i}$, $w_{j,j'}$ "wraps around" the tree yield $w_{i,p}$, $w_{q,j}$, i.e., that the two copies of i match and likewise j.

To some extent, the two computations above can be carried out independently of each other. More precisely, the result of the check on the selectional constraint does not depend on the value of positions p and q. Furthermore, once the check has been carried out, we can do away with the anchor position h', since this information is not used by the wrapping test or mentioned in the result of step (1).

In order to implement the above idea, we define two new kinds of items, which we write as $[\![\langle h,k,m\rangle,i,j]\!]$ and $[\![\langle h,k,m\rangle_\top,i',i,j,j']\!]$. Item $[\![\langle h,k,m\rangle,i,j]\!]$ packages together all items of the form $[\langle h,k,m\rangle,i,u,v,j]$. Similarly, item $[\![\langle h,k,m\rangle_\top,i',i,j,j']\!]$ packages together all items of the form $[\langle h',k',1\rangle_\top,i',i,j,j']$ such that $\langle h',k'\rangle \in \mathit{Adj}(\langle h,k,m\rangle)$. We can then replace step (1) with the following three steps:

$$\frac{[\langle h,k,m\rangle,i,p,q,j]}{[\![\langle h,k,m\rangle,i,j]\!]} \tag{2}$$

$$\frac{[\![\langle h,k,m\rangle,i,j]\!] \quad [\langle h',k',1\rangle_\top,i',i,j,j']}{[\![\langle h,k,m\rangle_\top,i',i,j,j']\!]}\ \langle h',k'\rangle \in \mathit{Adj}(\langle h,k,m\rangle) \tag{3}$$

$$\frac{[\![\langle h,k,m\rangle_\top, i', i, j, j']\!] \quad [\langle h,k,m\rangle, i, p, q, j]}{[\langle h,k,m\rangle_\top, i', p, q, j']} \tag{4}$$

A computational analysis similar to the one carried out in §1 shows the following overall time costs: step (2) takes time $\mathcal{O}(tg\,|w|^5)$, step (3) takes time $\mathcal{O}(tg^2\,|w|^6)$ and step (4) takes time $\mathcal{O}(tg\,|w|^7)$. Thus the overall time cost for all the above steps is $\mathcal{O}(tg\,|w|^6\max\{g,|w|\})$.

All the remaining steps in standard LTAG tabular parsing algorithms that have not been considered here can easily be accommodated within the indicated upper bound. Thus, steps (2) to (4) can be integrated into a standard LTAG parser, providing a new parsing algorithm for LTAG with worst case running time $\mathcal{O}(tg\,|w|^6\max\{g,|w|\})$.

3. Discussion

We have discussed standard LTAG, in which every elementary tree has exactly one lexical anchor. Multiply anchored trees can be handled straightforwardly and without additional cost: for the analysis, simply consider one anchor to be primary when defining the grammar constant g and when naming the tree $\langle h,k\rangle$. The parse table should be seeded with all of a tree's anchor nodes if and only if all those anchor words appear in the input w in the correct order. (Recall that it was always possible to construct subtrees over substrings that do not include the primary anchor.)

Our inference rules enforce the traditional prohibition against multiple adjunctions at the same node (Vijay-Shanker & Joshi, 1985). This prohibition has been questioned on linguistic grounds (Schabes & Shieber, 1994), since for example a verb may need to select lexically for each of its multiple PP adjuncts. To relax the prohibition it is sufficient to drop the symbol $\top$ throughout the rules.

Our algorithm is an asymptotic improvement for any values of g, t, and $|w|$. However, we really have in mind grammars where g is a smallish constant, much smaller than the vocabulary size. In particular, we do not expect a word to anchor multiple elementary trees that have the same labeled internal structure as one another, differing only in their selectional constraints. Thus, the selectional constraints at each node in an elementary tree only depend on the tree's head and the internal structure of the tree itself. Grammars satisfying this requirement have been called node-dependent or SLG(2) in (Carroll & Weir, 1997), and bilexical in (Eisner, 1997; Eisner & Satta, 1999; Eisner, 2000). If we drop the above assumption, the grammar can capture lexical relations of arity larger than two. For instance, in an LTAG which is not bilexical, a verb V_1 could anchor many instances of the basic transitive-sentence elementary tree, in each of which the selectional constraint at the object node required a specific object tree (with a specific head). In this case, the selectional constraint at each V_1 tree's subject node would depend on both V_1 *and* its required object, thus establishing a relation between three lexical elements. Moreover, an upstairs verb V_0 could select for certain of these V_1 trees and thereby restrict both V_1 and the head of V_1's object, again establishing a relation between three lexical elements. This style of grammar can dramatically increase g as a function of the vocabulary size. To overcome this one would again have to substitute some factor that depends on the input string length.

Even in the bilexical grammars we expect, where g is unrelated to vocabulary size, g can still be somewhat large in broad-coverage grammars such as those cited in the introduction, which include large tree families for each word. The literature describes some further tricks for efficiency in this case. Similar trees in the same family may be made to share structure (Evans

& Weir, 1997; Carroll *et al.*, 1998). “Supertagging” techniques (Srinivas & Joshi, 1999; Chen *et al.*, 1999) use contextual probabilities to eliminate some elementary trees heuristically before parsing begins. Alternatively, under a stochastic LTAG (Resnik, 1992; Schabes, 1992), one may prune away unpromising items, such as those with low inside probability. It should be possible to combine any of these tricks with our technique.

References

ABEILLÉ A. & CANDITO M.-H. (2000). FTAG: A lexicalized tree-adjoining grammar for french. In A. ABEILLÉ & O. RAMBOW, Eds., *Tree-Adjoining Grammar*. Stanford, CA: CSLI Lecture Notes. To appear.

ALSHAWI H. (1996). Head automata and bilingual tiling: Translation with minimal representations. In *Proc. of the 34th ACL*, p. 167–176, Santa Cruz, CA.

CARROLL J., NICOLOV N., SHAUMYAN O., SMETS M. & WEIR D. (1998). Grammar compaction and computation sharing in automaton-based parsing. In *Proceedings of the 1st Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, p. 16–25, Paris, France.

CARROLL J. & WEIR D. (1997). Encoding frequency information in lexicalized grammars. In *Proceedings of the 5th Int. Workshop on Parsing Technologies*, MIT, Cambridge, MA.

CHARNIAK E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proc. of AAAI-97*, Menlo Park, CA.

CHEN J., SRINIVAS B. & VIJAY-SHANKER K. (1999). New models for improving supertag disambiguation. In *Proc. of the 9th EACL*, p. 188–195, Bergen, Norway.

COLLINS M. (1997). Three generative, lexicalised models for statistical parsing. In *Proc. of the 35th ACL*, Madrid, Spain.

DORAN C., HOCKEY B., SARKAR A., SRINIVAS B. & XIA F. (2000). Evolution of the XTAG system. In A. ABEILLÉ & O. RAMBOW, Eds., *Tree-Adjoining Grammar*. Stanford, CA: CSLI Lecture Notes. To appear.

EISNER J. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proc. of the 16th COLING*, p. 340–345, Copenhagen, Denmark.

EISNER J. (1997). Bilexical grammars and a cubic-time probabilistic parser. In *Proceedings of the 5th Int. Workshop on Parsing Technologies*, MIT, Cambridge, MA.

EISNER J. (2000). Bilexical grammars and a cubic-time probabilistic parser. In H. C. BUNT & A. NIJHOLT, Eds., *New Developments in Natural Language Parsing*. Kluwer.

EISNER J. & SATTA G. (1999). Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. of the 37th ACL*, p. 457–464, College Park, Maryland.

EVANS R. & WEIR D. (1997). Automata-based parsing for lexicalized grammars. In *Proceedings of the 5th Int. Workshop on Parsing Technologies*, MIT, Cambridge, MA.

JOSHI A. K. (1987). An introduction to tree adjoining grammars. In A. MANASTER-RAMER, Ed., *Mathematics of Language*. Amsterdam: John Benjamins.

JOSHI A. K. & SCHABES Y. (1992). Tree adjoining grammars and lexicalized grammars. In M. NIVAT & A. PODELSKY, Eds., *Tree Automata and Languages*. Amsterdam, The Netherlands: Elsevier.

RESNIK P. (1992). Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proc. of the 14th COLING*, p. 418–424, Nantes, France.

SCHABES Y. (1992). Stochastic lexicalized tree-adjoining grammars. In *Proc. of the 14th COLING*, p. 426–432, Nantes, France.

SCHABES Y., ABEILLÉ A. & JOSHI A. K. (1988). Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proc. of the 12th COLING*, p. 578–583, Budapest, Hungary.

SCHABES Y. & SHIEBER S. M. (1994). An alternative conception of tree-adjoining derivation. *Computational Linguistics*, **20** (1), p. 91–118.

SLEATOR D. & TEMPERLEY D. (1991). *Parsing English with a Link Grammar*. Computer Science Technical Report CMU-CS-91-196, Carnegie Mellon University.

SRINIVAS B. & JOSHI A. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, **20** (3), p. 331–378.

VIJAY-SHANKER K. & JOSHI A. K. (1985). Some computational properties of tree adjoining grammars. In 23rd *Meeting of the Association for Computational Linguistics*, p. 82–93, Chicago, Illinois.

VIJAY-SHANKER K. & WEIR D. J. (1993). Parsing some constrained grammar formalisms. *Computational Linguistics*, **19** (4), p. 591–636.