Efficiency of Top-Down Parsing of Recursive Adjunction for Tree Adjoining Grammar

Jing Ji
Stony Brook University
jing.ji@mail.mcgill.ca

Abstract

CKY-type parser and Earley-type parser are two widely-used parsing algorithms for Tree Adjoining Grammar (TAG). In contrast, a standard top-down parser is not efficient since the looping problem occurs during both the left and right recursion of standard TAG derivation. (Roark, 2001) combines the top-down parser for CFG with a beam search, showing that the probabilistic top-down parser yields a perplexity improvement over previous results. In this paper, we define the stochastic tree adjoining grammar and apply the probabilistic top-down parser for CFG to TAG. Comparing the parsing efficiency of the standard and alternative TAG derivation of the recursive adjunction, we find that the alternative derivation is more efficient since it avoids the looping problem of the right recursion, increasing the parsing efficiency of our top-down parser.

1 Introduction

Tree Adjoining Grammar (TAG), introduced by Joshi et al. (1975) (Joshi et al., 1975), falls in the class of mildly context-sensitive languages (Joshi, 1985), which contain context-free languages and can also describe cross-serial dependency as is in Dutch and Swiss German. Two well-known chart parsing algorithms for TAG are the $O(n^6)$-time CKY-type parser (Vijay-Shankar and Joshi, 1985) and Earley-type parser extended from standard algorithms for Context-Free Grammar (CFG) (Joshi and Schabes, 1997; Nederhof, 1999). Recently, neural network architectures have also been utilized to build a shift-reduce parsing model for TAG (Kasai et al., 2017).

A standard top-down parser, in contrast, is not an efficient algorithm for CFG as it suffers from looping when dealing with left recursion. For TAG, the looping problem of the top-down parser even occurs with the right recursion when adjoining to the root node of elementary trees. Thus top-down parsers are always combined with a beam search strategy. Empirical results show that probabilistic top-down parsing for CFG improves upon previous work in test corpus perplexity (Roark, 2001). Moreover, Top-down parsers are incremental in the sense that each word is attached to a fully connected derivation, a property that left-corner and bottom-up parsers do not have (Stabler, 2013).

In this paper, we apply the probabilistic top-down parser with a beam search for CFG to TAG based on the definition of stochastic tree adjoining grammar and compare parsing efficiency of the standard and alternative TAG derivation of recursive adjunction.

2 Stochastic Tree Adjoining Grammar

Tree Adjoining Grammar (TAG) is a tree-rewriting system that consists of a finite set of elementary trees. We provide the definition of TAG following (Vijay-Shanker and Weir, 1994; Kallmeyer, 2010) to set the stage.

Let N_+ denotes the set of positive integers. D is tree domain if it is a nonempty finite subset of N_+^* such that if $d \in D$ and $d = d_1 d_2$ then $d_1 \in D$ and if $di \in D$ where $i \in N$ then $dj \in D$ for all $1 \leq j \leq i$. Note that ϵ is the address of the root node of the tree. A tree γ is denoted by a function $\gamma : D_\gamma \rightarrow V_N \cup V_T$ where D_γ is the domain of γ, V_N and V_T are the set of nonterminal and terminal node labels, respectively.

Definition 1 (Auxiliary Trees)**.** Auxiliary trees is a set of trees $\beta : D_\beta \rightarrow V_T \cup V_N$ where

- the root and internal nodes of β are labelled by some $V \in V_N$;
- all leaf nodes of β except one are labelled by some $v \in V_T$. The remaining leaf node is the foot nodemarked by an asterisk (*).

Proceedings of the Society for Computation in Linguistics (SCiL) 2021, pages 374-379.
Held on-line February 14-19, 2021

Definition 2 (Initial Trees)**.** Initial trees is a set of trees $\alpha : D_\alpha \rightarrow V_T \cup V_N$ and the following hold.

- The root and internal nodes of α are labelled by some $V \in V_N$.
- Leaf nodes of α are labelled by some $v \in V_T$ or some $V \in V_N$ which is a substitution node marked by a down arrow ($\downarrow$).

Elementary trees are the union of auxiliary and initial trees. In lexicalized TAG (LTAG), every elementary tree γ has at least one non-empty lexical item, its lexical anchor. LTAG is weakly equivalent to TAG (defining the same language) (Kuhlmann and Satta, 2012).

Definition 3 (Tree Adjoining Grammar)**.** A TAG is a tuple $G = \langle V_N, V_T, V_L, I, A, S, f_{OA}, f_{SA}, - \rangle$ where

V_L is a finite set of tree labels,

I is a finite subset of initial trees,

A is a finite subset of auxiliary trees,

$S \in V_N$ is a start symbol,

f_{OA} and f_{SA} are functions representing adjunction constraints:

$f_{OA} : \langle \gamma, d \rangle \rightarrow \{true, false\}$ where $d \in D_\gamma, \gamma \in I \cup A, \gamma(d) \in V_N$,

$f_{SA} : \langle \gamma, d \rangle \rightarrow P(A)$ where $d \in D_\gamma, \gamma \in I \cup A, \gamma(d) \in V_N$,

$- : I \cup A \rightarrow V_L$ is the elementary tree labelling function.

Adjunction applies to nonterminal internal nodes of elementary trees. For $\gamma \in I \cup A, d \in D_\gamma, \gamma(d) \in V_N$, if $f_{SA}(\gamma, d) = \emptyset$, adjunction is forbidden for $\gamma(d)$; if $f_{OA}(\gamma, d) = true$, adjunction is obligatory; otherwise adjunction is optional.

Definition 4 (Adjunction and Substitution)**.** Given a TAG $G = \langle V_N, V_T, V_L, I, A, S, f_{OA}, f_{SA}, - \rangle$, for $\gamma \in I \cup A$, $\alpha \in I$, $\beta \in A$, $d \in D_\gamma$, $d' \in N_+^*$, $\gamma' = \gamma[d, \alpha]$ and $\gamma'' = \gamma[d, \beta]$ are the result of substituting α and adjoining β into γ, which are defined below.

$$\gamma'(d') = \begin{cases} \gamma(d') & \text{if } d = d' \\ \alpha(d'') & \text{if } d' = dd'' \text{ and } d'' \in D_\alpha \end{cases}$$

$$\gamma''(d') = \begin{cases} \gamma(d') & \text{if } d = d' \\ \beta(d'') & \text{if } d' = dd'' \text{ and } d'' \in D_\beta \\ \gamma(dd'') & \text{if } d' = dd_f d'' \text{ and } d_f \text{ is the} \\ & \text{address of the foot node of } \beta \end{cases}$$

Definition 5 (Derived Trees and Derivation Trees)**.** Let $G = \langle V_N, V_T, V_L, I, A, S, f_{OA}, f_{SA}, - \rangle$ be a TAG. A derivation tree in G can be taken as a directed graph, which is a pair $\langle V, E \rangle$ where V is a finite set of vertices and $E \subseteq V \times V$ is a set of edges.

- Every $\gamma \in I \cup A$ is a derived tree in G. The corresponding derivation tree is $\langle \{-(\gamma)\}, \emptyset \rangle$.
- Let γ be a derived tree and $\langle V, E \rangle$ be its derivation tree. If $\gamma' = \gamma[d, \eta]$ for $d \in D_\gamma$ and $\eta \in I \cup A$, then γ' is a derived tree and the derivation tree of γ' is $\langle V', E' \rangle$ such that
  - $V' = V \cup \{-(\eta)\}$,
  - $E' = E \cup \{\langle -(\gamma), -(\eta) \rangle\}$,
  - $g(\langle -(\gamma), -(\eta) \rangle) = d$.

A derived tree γ with no substitution nodes and no internal node $\gamma(d)$ such that $f_{OA}(\gamma, d) = true$ is a saturated derived tree. A derivation tree is complete if its corresponding derived tree is saturated. Let ω_i^j denote the string $\omega_{i+1} \ldots \omega_j$. Let $T_{\omega_0^n}$ be the set of all complete derivation trees with ω_0^n as leaves of their corresponding derived trees. A Stochastic TAG (STAG) is a TAG with a probability assigned to each rule. A STAG defines a probability distribution over strings of words in the following way.

$$P(\omega_0^n) = \sum_{t \in T_{\omega_0^n}} P(t) \tag{1}$$

3 A Probabilistic Top-Down Parser with Beam Search for TAG

The definition of probabilistic top-down parser stems from the top-down parser (left-to-right, depth-first) described in (Roark, 2001) for PCFG. We apply it to STAG parsing without left-factorization. The parser takes an input string ω_0^n, a STAG, and a priority queue of candidate analyses. A candidate analysis $C = (E, S, P_E, F, \omega_i^n)$ consists of a set of edges E of a derivation tree, a stack S, a derivation probability P_E, a figure of merit F, and a string ω_i^n to be parsed. The first word in the string remaining to be parsed, ω_{i+1}, is the look-ahead word. The stack S contains a sequence of node labels superscripted with the tree-labels and an end-of-stack marker \$ at the bottom. The probability P_E is the product of the probabilities of all the edges in E. F is the product of P_E and a look-ahead probability $LAP(S, \omega_{i+1})$.

A derives relation between two candidate analyses, denoted as $\mapsto$, is defined based on the following conditions.

Condition 1: the first symbol on the stack is not a foot node, and no substitution or adjunction performs. $(E, S, P_E, F, \omega_i^n) \mapsto (E', S', P_{E'}, F', \omega_j^n)$ where

- $E' = E$
- $S = V^{\gamma\pi}\chi\$$ $(\gamma \in V_L, \pi \in V_L^*)$
- if $V^\gamma \rightarrow X_1^\gamma \dots X_k^\gamma$, then $S' = X_1^{\gamma\pi} \dots X_k^{\gamma\pi}\chi\$$, and j = i; if $V^\gamma \rightarrow \omega_{i+1}$ or $V^\gamma = \omega_{i+1}$, then $j = i+1$, and $S' = \chi\$$;
- if $V^\gamma \rightarrow X_1^\gamma \dots X_k^\gamma$, then $P_{E'} = P_E P(V^\gamma \rightarrow X_1^\gamma \dots X_k^\gamma)$, otherwise $P_{E'} = P_E$
- $F' = P_{E'} LAP(S', \omega_{j+1})$

Condition 2: V^γ is the foot node of γ adjoining to the tree labeled η. $(E, S, P_E, F, \omega_i^n) \mapsto (E', S', P_{E'}, F', \omega_j^n)$ where

- $E' = E$
- $S = V^{\gamma\eta\pi}\chi\$$
- if $V^\eta \rightarrow X_1^\eta \dots X_k^\eta$, then $S' = X_1^{\eta\pi} \dots X_k^{\eta\pi}\chi\$$, and j = i; if $V^\eta \rightarrow \omega_{i+1}$, then $j = i+1$, and $S' = \chi\$$
- $P_{E'} = P_E$
- $F' = P_{E'} LAP(S', \omega_{j+1})$

Condition 3: the node V^γ is substituted or adjoined by a tree labeled η, denoted as $\gamma \overset{V}{\Longrightarrow} \eta$. $(E, S, P_E, F, \omega_i^n) \mapsto (E', S', P_{E'}, F', \omega_j^n)$ where

- $E' = E \cup \{\langle \gamma, \eta \rangle\}$
- $S = V^{\gamma\pi}\chi\$$
- $S' = V^{\eta\gamma\pi}\chi\$$
- $P_{E'} = P_E P(\gamma \overset{V}{\Longrightarrow} \eta)$
- $F' = P_{E'} LAP(S', \omega_{j+1})$

The parse begins with a single candidate analysis on the priority queue: $(\emptyset, S^{\overline{\alpha}}\$, 1, 1, \omega_0^n)$ where $\alpha \in I$ and $\alpha(\epsilon) = S$. If $S = \$$ and $\omega_{i+1} = \langle / s \rangle$, the end symbol of the string, the analysis is complete. The symbols on the stack are the contracted representation of the items.

Example 1. Suppose $G_{raising} = \langle \{S, NP, VP, V\}, \{John, seems, to\_sleep\}, \{\overline{\alpha_s}, \overline{\alpha_n}, \overline{\beta}\}, \{\alpha_s, \alpha_n\}, \{\beta\}, S, f_{OA}, f_{SA}, - \rangle$ is a TAG. The sentence to be generated is *John seems to_sleep*.The elementary trees and derivation tree are shown in Figure 1 and 2. The parsing trace is presented in Table 1.

The LAP is the probability of a particular terminal being derived as the first non-empty leaf from a set of nonterminal nodes. For a stochastic LTAG (SLTAG), a stack $S = V_1^{\gamma_1\pi_1} \dots V_k^{\gamma_k\pi_k}\$$ and a look-ahead terminal item ω_{i+1}, the look-ahead probability defined in Equation 2 computes the probability of the concatenation of yield of the subtrees rooted at $V_1^{\gamma_1\pi_1} \dots V_k^{\gamma_k\pi_k}$ of the derived

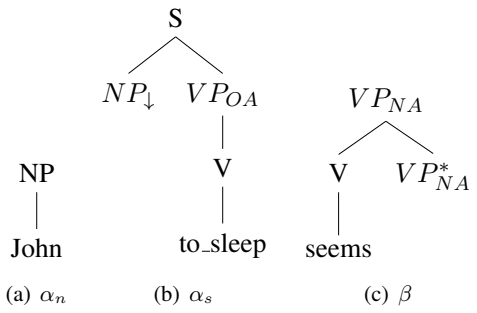


Figure 1: The elementary trees of $G_{raising}$.

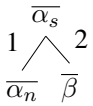


Figure 2: A derivation tree in $G_{raising}$.

tree starting with ω_{i+1}. Let $P(V^{\gamma\pi} \overset{\triangle}{\rightarrow} \lambda)$ denote the probability of the yield of the subtree of the derived tree rooted at $V^{\gamma\pi}$ being empty.

$$LAP(S, \omega_{i+1}) = P(V_1^{\gamma_1\pi_1} \dots V_k^{\gamma_1\pi_k} \overset{\star}{\rightarrow} \omega_{i+1}) \tag{2}$$

$$P(V_j^{\gamma_j\pi_j} \dots V_k^{\gamma_k\pi_k} \overset{\star}{\rightarrow} \omega_i) = P(V_j^{\gamma_j\pi_j} \overset{\star}{\rightarrow} \omega_i) + P(V_j^{\gamma_j\pi_j} \overset{\triangle}{\rightarrow} \epsilon) P(V_{j+1}^{\gamma_{j+1}\pi_{j+1}} \dots V_k^{\gamma_k\pi_k} \overset{\star}{\rightarrow} \omega_i) \tag{3}$$

where $P(V^{\gamma\pi} \overset{\star}{\rightarrow} \omega_i)$ and $P(V^{\gamma\pi} \overset{\triangle}{\rightarrow} \lambda)$ are defined recursively.

$$P(V^{\gamma\pi} \overset{\star}{\rightarrow} \omega_i) = P(V^\gamma \rightarrow X_1^\gamma \dots X_n^\gamma) P(X_1^\gamma \dots X_n^\gamma \overset{\star}{\rightarrow} \omega_i) + \sum_{\eta \in V_L} P(\gamma \overset{V}{\Longrightarrow} \eta) P(V^\eta \overset{\star}{\rightarrow} \omega_i) \tag{4}$$

$$P(V^{\gamma\pi} \overset{\triangle}{\rightarrow} \lambda) = P(V^\gamma \rightarrow X_1^\gamma \dots X_n^\gamma) P(X_1^\gamma \overset{\triangle}{\rightarrow} \lambda) \dots P(X_n^\gamma \overset{\triangle}{\rightarrow} \lambda) \tag{5}$$

The beam search works in the same way as (Roark, 2001). For each word position i, we have a separate priority queue H_i of analyses with look-ahead ω_{i+1}. When there are enough analyses on priority queue H_{i+1}, all candidate analyses remaining on H_i are discarded. The parse on H_{i+1} with the highest probability is returned for evaluation. The beam threshold at word ω_i is the same as that of the CFG. If $\tilde{p}$ is the probability of the highest-ranked analysis on H_{i+1}, then another analysis is discarded if its probability falls below $\tilde{p} f(t, |H_{i+1}|)$, where t is an initial parameter.

Stack	ω_{i+1}	Derivation Edges
$S^{\overline{\alpha_s}}\$$	John	$\emptyset$
$NP^{\overline{\alpha_s}}VP^{\overline{\alpha_s}}\$$	John	$\emptyset$
$NP^{\overline{\alpha_n\alpha_s}}VP^{\overline{\alpha_s}}\$$	John	$\{\langle\overline{\alpha_s},\overline{\alpha_n}\rangle\}$
$VP^{\overline{\alpha_s}}\$$	seems	$\{\langle\overline{\alpha_s},\overline{\alpha_n}\rangle\}$
$VP^{\overline{\beta\alpha_s}}\$$	seems	$\{\langle\overline{\alpha_s},\overline{\alpha_n}\rangle,\langle\overline{\alpha_s},\overline{\beta}\rangle\}$
$V^{\overline{\beta\alpha_s}}VP^{\overline{\beta\alpha_s}}\$$	seems	$\{\langle\overline{\alpha_s},\overline{\alpha_n}\rangle,\langle\overline{\alpha_s},\overline{\beta}\rangle\}$
$VP^{\overline{\beta\alpha_s}}\$$	to_sleep	$\{\langle\overline{\alpha_s},\overline{\alpha_n}\rangle,\langle\overline{\alpha_s},\overline{\beta}\rangle\}$
$V^{\overline{\alpha_s}}\$$	to_sleep	$\{\langle\overline{\alpha_s},\overline{\alpha_n}\rangle,\langle\overline{\alpha_s},\overline{\beta}\rangle\}$
$\$$	$\langle/s\rangle$	$\{\langle\overline{\alpha_s},\overline{\alpha_n}\rangle,\langle\overline{\alpha_s},\overline{\beta}\rangle\}$

Table 1: Stack trace for top-down parsing of *John seems to sleep* $\langle/s\rangle$ (ω_{i+1}: look-ahead word).

4 Parsing Efficiency with Standard and Alternative Derivations

For a top-down CFG parser, left recursion can force it to enter an infinite loop of top-down predictions. It is the same for top-down parsing of TAG. However, when the root node is the adjunction node, both the left and right-branching structure can result in an infinite loop of top-down prediction. A top-down parser with a beam threshold can avoid the infinite loop, but it cannot make a distinction between left and right recursion concerning memory load. From the psychological point of view, only left recursion leads to process difficulty. This section tries to tackle this issue with the alternative conception of adjunction instead of the standard one and compare the parsing efficiency with the probabilistic top-down parser.

The standard definition of derivation, attributable to (Vijayashanker and Joshi, 1988), requires that auxiliary trees be adjoined at distinct nodes in elementary trees. However, considering the difference between modification and predication, (Schabes and Shieber, 1994) proposed a redefinition of TAG derivation, whereby multiple auxiliary tree modification can be adjoined at a single node. For example, given an input string S_1 = *Mary has a nice big new red table* $\langle/s\rangle$ with elementary trees shown in A(a)-(h), its standard and alternative derivation of the sentence are depicted in Figure A(i)-(j).

The definition of top-down parser in section 3 is based on the standard conception of derivation. In order to allow multiple adjunction at the same node, the analysis of Condition 2 can be modified as follows. $(E, S, P_E, F, \omega_i^n) \mapsto (E', S', P_{E'}, F', \omega_j^n)$ where

- $E' = E$
- $S = V^{\gamma\eta\pi}\chi\$$
- $S' = V^{\eta\pi}\chi\$$
- $P'_E = P_E$
- $F' = P_{E'}LAP(S', \omega_{j+1})$

For the top-down parser with a beam search, the following analysis shows that the alternative adjunction is more efficient than the standard adjunction.

Assume that there is no empty leaf in the trees. Both ways of parsing result in the same analyses from the start to $C_0 = (\{\langle\overline{\alpha_s},\overline{\alpha_{n1}}\rangle,\langle\overline{\alpha_s},\overline{\alpha_{n2}}\rangle\}, NP^{\overline{\alpha_{n2}\alpha_s}}\$, P_0, F_0,$ *a nice big new red table* $\langle/s\rangle)$. The remaining stack are shown in Table 2.

It can be seen that in terms of the number of steps, standard parsing (SP) is less than alternative parsing (AP). In terms of the probability of each step, the analyses with continuous adjunction in SP can result in tiny probabilities. For example, in order to make prediction of the word *a*, SP needs five successive adjunction from C_0 to $C_{SP} = (E_{SP}, NP^{\overline{\beta_d\beta_{a1}\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$, P_{SP}, F_{SP},$ *a nice big new red table* $\langle/s\rangle)$, while AP requires only one adjunction to $C_{AP} = (E_{AP}, NP^{\overline{\beta_d\alpha_{n2}\alpha_s}}\$, P_{AP}, F_{AP},$ *a nice big new red table* $\langle/s\rangle)$. According to the equations in Section 3,

$F_{AP} = P_0P(\overline{\alpha_{n2}} \overset{NP}{\Longrightarrow} \overline{\beta_d})LAP(NP^{\overline{\beta_d\alpha_{n2}\alpha_s}}\$, a)$

Since $f_{SA}(\beta_d, \epsilon) = \emptyset$, $LAP(NP^{\overline{\beta_d\alpha_{n2}\alpha_s}}\$, a) = 1$. Thus, $F_{AP} = P_0P(\overline{\alpha_{n2}} \overset{NP}{\Longrightarrow} \overline{\beta_d})$.

Likewise,

$F_{SP} = P_0P(\overline{\alpha_{n2}} \overset{NP}{\Longrightarrow} \overline{\beta_{a4}})P(\overline{\beta_{a4}} \overset{NP}{\Longrightarrow} \overline{\beta_{a3}})$
$P(\overline{\beta_{a3}} \overset{NP}{\Longrightarrow} \overline{\beta_{a2}})P(\overline{\beta_{a2}} \overset{NP}{\Longrightarrow} \overline{\beta_{a1}})P(\overline{\beta_{a1}} \overset{NP}{\Longrightarrow} \overline{\beta_d})$

Suppose all the rules for each node have equivalent probabilities. Since $f_{SA}(\alpha_{n2}, \epsilon) = \{\beta_{a1}, \beta_{a2}, \beta_{a3}, \beta_{a4}, \beta_d\}$ and $f_{OA}(\alpha_{n2}, \epsilon) = true$, $F_{AP} = P_0P(\overline{\alpha_{n2}} \overset{NP}{\Longrightarrow} \overline{\beta_d}) = \frac{1}{5}P_0$.

Considering the ordering hierarchy of stacked adjectival modification (Scott, 2002) with $f_{SA}(\alpha_{a4}, \epsilon) = \{\beta_{a1}, \beta_{a2}, \beta_{a3}, \beta_{a4}, \beta_d\}$, $f_{SA}(\alpha_{a3}, \epsilon) = \{\beta_{a1}, \beta_{a2}, \beta_{a3}, \beta_d\}$, $f_{SA}(\alpha_{a2}, \epsilon) = \{\beta_{a1}, \beta_{a2}, \beta_d\}$, $f_{SA}(\alpha_{a1}, \epsilon) = \{\beta_{a1}, \beta_d\}$, $F_{SP} = \frac{1}{5} \times \frac{1}{6} \times \frac{1}{5} \times \frac{1}{4} \times \frac{1}{3}P_0$, far less than F_{AP}.

To generalize, if the stacked adjectives in a nominal phrase are $a_1 \ldots a_m$ generated from n paradigms ($m, n \geq 1$), the ratio of probabilities for the first word of the nominal phrase will be $\frac{F_{AP}}{F_{SP}} = \frac{(n+2)!}{(n-m+2)!}$ if $a_1 \ldots a_m$ are distinct words ($m \leq n$). If there are repetitive words in the nominal phrase, in the worst case, the ratio of probabilities can be $\frac{F_{AP}}{F_{SP}} = (n+2)^m$.

References

Aravind K Joshi. 1985. How much context sensitivity is necessary for characterizing structural descriptions: Tree adjoining grammars. *Natural language parsing: Psychological, computational and theoretical perspectives*, pages 206–250.

Aravind K Joshi, Leon S Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of computer and system sciences*, 10(1):136–163.

Aravind K Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer.

Laura Kallmeyer. 2010. *Parsing beyond context-free grammars*. Springer Science & Business Media.

Jungo Kasai, Robert Frank, R Thomas McCoy, Owen Rambow, and Alexis Nasr. 2017. Tag parsing with neural networks and vector representations of supertags.

Marco Kuhlmann and Giorgio Satta. 2012. Tree-adjoining grammars are not closed under strong lexicalization. *Computational Linguistics*, 38(3):617–629.

Mark-Jan Nederhof. 1999. The computational complexity of the correct-prefix property for tags. *Computational Linguistics*, 25(3):345–360.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational linguistics*, 27(2):249–276.

Yves Schabes and Stuart M Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

Gary-John Scott. 2002. Stacked adjectival modification and the structure of nominal phrases. *Functional structure in DP and IP: The cartography of syntactic structures*, 1:91–120.

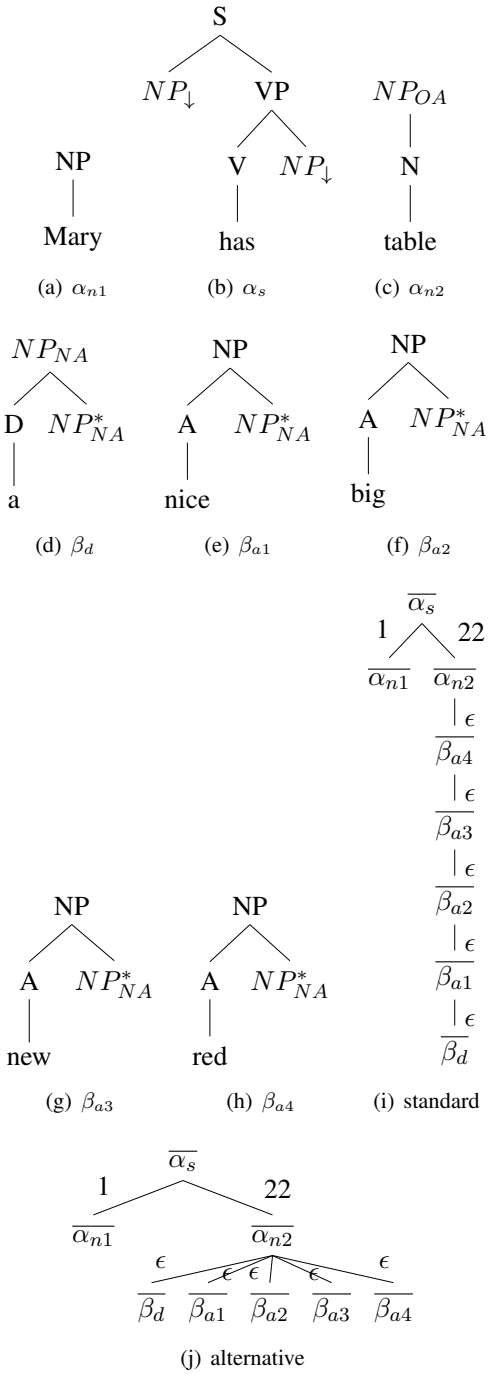
Edward P Stabler. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in cognitive science*, 5(3):611–633.

K Vijay-Shankar and Aravind K Joshi. 1985. Some computational properties of tree adjoining grammars. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pages 82–93. Association for Computational Linguistics.

Krishnamurti Vijay-Shanker and David J Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical systems theory*, 27(6):511–546.

K Vijayashanker and Aravind K Joshi. 1988. *A study of tree adjoining grammars*. University of Pennsylvania Philadelphia.

A Appendices



(a) α_{n1} (b) α_s (c) α_{n2}

(d) β_d (e) β_{a1} (f) β_{a2}

(g) β_{a3} (h) β_{a4} (i) standard

(j) alternative

Step	Standard	ω_{i+1}	Alternative	ω_{i+1}
1	$NP^{\overline{\beta_{a4}\alpha_{n2}\alpha_s}}\$$	a	$NP^{\overline{\beta_d\alpha_{n2}\alpha_s}}\$$	a
2	$NP^{\overline{\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	a	$N^{\overline{\beta_d\alpha_{n2}\alpha_s}}\ NP^{\overline{\beta_d\alpha_{n2}\alpha_s}}\$$	a
3	$NP^{\overline{\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	a	$NP^{\overline{\beta_d\alpha_{n2}\alpha_s}}\$$	nice
4	$NP^{\overline{\beta_{a1}\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	a	$NP^{\overline{\alpha_{n2}\alpha_s}}\$$	nice
5	$NP^{\overline{\beta_d\beta_{a1}\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	a	$NP^{\overline{\beta_{a1}\alpha_{n2}\alpha_s}}\$$	nice
6	$N^{\overline{\beta_d\beta_{a1}\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\ NP^{\overline{\beta_d\beta_{a1}\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	a	$N^{\overline{\beta_{a1}\alpha_{n2}\alpha_s}}\ NP^{\overline{\beta_{a1}\alpha_{n2}\alpha_s}}\$$	nice
7	$NP^{\overline{\beta_d\beta_{a1}\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	nice	$NP^{\overline{\beta_{a1}\alpha_{n2}\alpha_s}}\$$	big
8	$N^{\overline{\beta_{a1}\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\ NP^{\overline{\beta_{a1}\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	nice	$NP^{\overline{\alpha_{n2}\alpha_s}}\$$	big
9	$NP^{\overline{\beta_{a1}\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	big	$NP^{\overline{\alpha_{a2}\alpha_{n2}\alpha_s}}\$$	big
10	$N^{\overline{\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\ NP^{\overline{\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	big	$N^{\overline{\beta_{a2}\alpha_{n2}\alpha_s}}\ NP^{\overline{\beta_{a2}\alpha_{n2}\alpha_s}}\$$	big
11	$NP^{\overline{\beta_{a2}\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	new	$NP^{\overline{\beta_{a2}\alpha_{n2}\alpha_s}}\$$	new
12	$N^{\overline{\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\ NP^{\overline{\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	new	$NP^{\overline{\alpha_{n2}\alpha_s}}\$$	new
13	$NP^{\overline{\beta_{a3}\beta_{a4}\alpha_{n2}\alpha_s}}\$$	red	$NP^{\overline{\beta_{a3}\alpha_{n2}\alpha_s}}\$$	new
14	$N^{\overline{\beta_{a4}\alpha_{n2}\alpha_s}}\ NP^{\overline{\beta_{a4}\alpha_{n2}\alpha_s}}\$$	red	$N^{\overline{\beta_{a3}\alpha_{n2}\alpha_s}}\ NP^{\overline{\beta_{a3}\alpha_{n2}\alpha_s}}\$$	new
15	$NP^{\overline{\beta_{a4}\alpha_{n2}\alpha_s}}\$$	table	$NP^{\overline{\beta_{a3}\alpha_{n2}\alpha_s}}\$$	red
16	$N^{\overline{\alpha_{n2}\alpha_s}}\$$	table	$NP^{\overline{\alpha_{n2}\alpha_s}}\$$	red
17	$\$$	$\langle/s\rangle$	$NP^{\overline{\beta_{a4}\alpha_{n2}\alpha_s}}\$$	red
18			$N^{\overline{\beta_{a4}\alpha_{n2}\alpha_s}}\ NP^{\overline{\beta_{a4}\alpha_{n2}\alpha_s}}\$$	red
19			$NP^{\overline{\beta_{a4}\alpha_{n2}\alpha_s}}\$$	table
20			$NP^{\overline{\alpha_{n2}\alpha_s}}\$$	table
21			$N^{\overline{\alpha_{n2}\alpha_s}}\$$	table
22			$\$$	$\langle/s\rangle$

Table 2: Stack trace for standard and alternative top-down parsing of S_1 (ω_{i+1}: look-ahead word).