# Grammaires Stochastiques Lexicalisées d'Arbres Adjoints

Résumé du papier
*Stochastic Lexicalized Tree-adjoining Grammars*
*Yves Schabes*

## Motivations

Les techniques stochastiques bénéficient aujourd'hui d'un regain de popularité. Cependant, les modèles stochastiques utilisés sont clairement inadéquats pour l'analyse syntaxique des langues naturelles. Les formalismes probabilistes qui ont été proposés dans le domaine de la théorie de la communication (processus de Markov et n-grammes) (Pratt, 1942; Shannon, 1948; Shannon, 1951) ont éte rapidement réfutés en linguistique. En effet, ces modèles sont incapables de décrire la syntaxe de manière hiérarchique (sous forme d'arbre). De plus, les phénomènes portant sur de longues distances ne peuvent pas être pris en compte par ces formalismes. Les grammaires stochastiques hors contexte (Booth, 1969) permettent d'élaborer une description hiérarchique de la syntaxe. Cependant, aucune approche utilisant les grammaires stochastiques hors contexte (Lari and Young, 1990; Jelinek, Lafferty, and Mercer, 1990) est en pratique aussi efficace que les processus de Markov ou les n-grammes. En effet, les règles hors contexte ne sont pas directement sensibles au mot et donc à une distribution de mots.

## Grammaires Stochastiques Lexicalisées d'Arbres Adjoints

Les grammaires lexicalisées d'arbres adjoints consistent d'un ensemble d'arbres, chacun associé à un mot. Elles permettent de localiser la plupart des contraintes syntaxiques (par exemple, sujet-verbe, verbe-objet) tout en décrivant la syntaxe sous forme d'arbres.

Dans ce papier, la notion de derivation des grammaires lexicalisées d'arbres adjoints (tree-adjoining grammars) est modifiée au cas de derivations stochastiques. Le nouveau formalisme, les grammaires stochastiques lexicalisées d'arbres adjoints (stochastic lexicalized tree-adjoining grammars ou SLTAG) , a des propriétés uniques car il maintient la notion de distribution entre mot tout en manipulant la syntaxe de manière hiérarchique.

## Algorithmes

Un algorithme pour calculer la probabilité d'une phrase est présenter dans le papier.

Ensuite, un algorithme qui permet de réestimer les paramètres d'une grammaire stochastique lexicalisée d'arbres adjoints est décrit. Cette algorithme permet de réestimer les paramètres de façon à augmenter après chaque itération la probabilité du corpus. Cette algorithme peut être utilisé comme algorithme d'apprentissage. La grammaire initiale d'entrée génère tous les mots de toutes les façons possibles. L'algorithme permet ensuite d'inférer une grammaire à partir du corpus.

## Evaluation Expérimentale

Nous avons testé l'algorithme de réestimation sur un corpus artificiel (Figure 1) et aussi sur les sequences de parties du discours (Figure 2) du corpus 'ATIS' (Hemphill, Godfrey, and Doddington, 1990). Dans les deux cas, l'algorithme pour les grammaires stochastiques lexicalisées d'arbres adjoints converge plus rapidement que celui pour les grammaires hors contexte (Baker, 1979). Ces expériences confirment le fait que les grammaires stochastiques lexicalisées d'arbres adjoints permettent de modéliser des distributions entre mots que les grammaires stochastiques hors contexte ne peuvent pas exprimer.

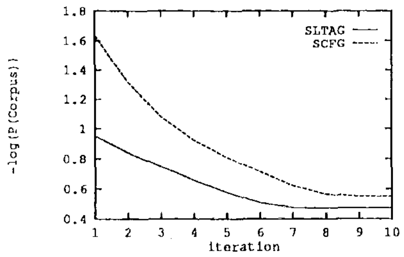


Figure 1: Convergence avec un corpus de phrases du language $\{a^n b^n | n \geq 0\}$

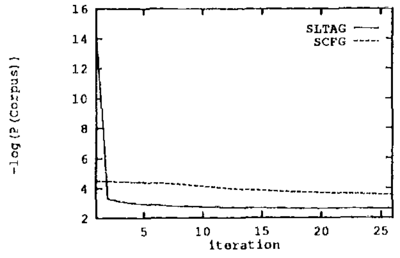


Figure 2: Convergence sur le ATIS Corpus

# Stochastic Lexicalized Tree-Adjoining Grammars *

*Yves Schabes*
Dept. of Computer & Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389, USA
schabes@unagi.cis.upenn.edu

## Abstract

The notion of stochastic lexicalized tree-adjoining grammar (SLTAG) is formally defined. The parameters of a SLTAG correspond to the probability of combining two structures each one associated with a word. The characteristics of SLTAG are unique and novel since it is lexically sensitive (as N-gram models or Hidden Markov Models) and yet hierarchical (as stochastic context-free grammars).

Then, two basic algorithms for SLTAG are introduced: an algorithm for computing the probability of a sentence generated by a SLTAG and an inside-outside-like iterative algorithm for estimating the parameters of a SLTAG given a training corpus.

Finally, we should how SLTAG enables to define a lexicalized version of stochastic context-free grammars and we report preliminary experiments showing some of the advantages of SLTAG over stochastic context-free grammars.

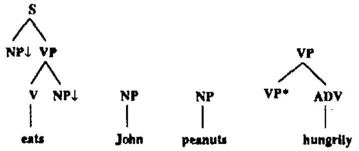
## 1 Motivations

Although stochastic techniques applied to syntax modeling have recently regained popularity, current language models suffer from obvious inherent inadequacies. Early proposals such as Markov Models, N-gram models (Pratt, 1942; Shannon, 1948; Shannon, 1951) and Hidden Markov Models were very quickly shown to be linguistically not appropriate for natural language (e.g. Chomsky (1964, pages 13-18)) since they are unable to capture long distance dependencies or to describe hierarchically the syntax of natural languages. Stochastic context-free grammar (Booth, 1969) is a hierarchical model more appropriate for natural languages, however none of such proposals (Lari and Young, 1990; Jelinek, Lafferty, and Mercer, 1990) perform as well as the simpler Markov Models because of the difficulty of capturing lexical information. The parameters of a stochastic context-free grammar do not correspond directly to a distribution over words since distributional phenomena over words that are embodied by the application of more than one context-free rule cannot be captured under the context-freeness assumption. This leads to the difficulty of maintaining a standard hierarchical model while capturing lexical dependencies.

This fact prompted researchers in natural language processing to give up hierarchical language models in the favor of non-hierarchical statistical models over words (such as word N-grams models). Probably for lack of a better language model, it has also been argued that the phenomena that such devices cannot capture occur relatively infrequently. Such argumentation is linguistically not sound.

Lexicalized tree-adjoining grammars (LTAG)[1] combine hierarchical structures while being lexically sensitive and are therefore more appropriate for statistical analysis of language. In fact, LTAGs are the simplest hierarchical formalism which can serve as the basis for lexicalizing context-free grammar (Schabes, 1990; Joshi and Schabes, 1991).

LTAG is a tree-rewriting system that combines trees of large domain with ***adjoining*** and ***substitution***. The trees found in a TAG take advantage of the available extended domain of locality by localizing syntactic dependencies (such as filler-gap, subject-verb, verb-object) and most semantic dependencies (such as predicate-argument relationship). For example, the following trees can be found in a LTAG lexicon:



Since the elementary trees of a LTAG are minimal syntactic and semantic units, distributional analysis of the combination of these elementary trees based on a training corpus will inform us about relevant statistical aspects of the language such as the classes of words appearing as arguments of a predicative element, the distribution of the adverbs licensed by a specific verb, or the adjectives licensed by a specific noun.

This kind of statistical analysis as independently suggested in (Resnik, 1991) can be made with LTAGs because of their extended domain of locality but also because of their lexicalized property.

*This work was partially supported by DARPA Grant N0014-90-31863, ARO Grant DAAL03-89-C-0031 and NSF Grant IRI90-16592. We thank Aravind Joshi for suggesting the use of TAGs for statistical analysis during a private discussion that followed a presentation by Fred Jelinek during the June 1990 meeting of the DARPA Speech and Natural Language Workshop. We are also grateful to Peter Braun, Fred Jelinek, Mark Liberman, Mitch Marcus, Robert Mercer, Fernando Pereira and Stuart Shieber for providing valuable comments.

[1] We assume familiarity throughout the paper with TAGs and its lexicalized variant. See, for instance, (Joshi, 1987), (Schabes, Abeillé, and Joshi, 1988), (Schabes, 1990) or (Joshi and Schabes, 1991).

In this paper, this intuition is made formally precise by defining the notion of a stochastic lexicalized tree-adjoining grammar (SLTAG). We present an algorithm for computing the probability of a sentence generated by a SLTAG, and finally we introduce an iterative algorithm for estimating the parameters of a SLTAG given a training corpus of text. This algorithm can either be used for refining the parameters of a SLTAG or for inferring a tree-adjoining grammar from a training corpus. We also report preliminary experiments with this algorithm.

Due to the lack of space, in this paper the algorithms are described succinctly without proofs of correctness and more attention is given to the concepts and techniques used for SLTAG.

## 2 SLTAG

Informally speaking, SLTAGs are defined by assigning a probability to the event that an elementary tree is combined (by adjunction or substitution) on a specific node of another elementary tree. These events of combination are the stochastic processes considered.

Since SLTAG are defined on the basis of the derivation and since TAG allows for a notion of derivation independent from the trees that are derived, a precise mathematical definition of the SLTAG derivation must be given. For this purpose, we use stochastic linear indexed grammars (SLIG) to formally express SLTAGs derivations.

Linear Indexed grammar (LIG) (Aho, 1968; Gazdar, 1985) is a rewriting system in which the non-terminal symbols are augmented with a stack. In addition to rewriting non-terminals, the rules of the grammar can have the effect of pushing or popping symbols on top of the stacks that are associated with each non-terminal symbol. A specific rule is triggered by the non-terminal on the left hand side of the rule and the top element of its associated stack.

The productions of a LIG are restricted to copy the stack corresponding to the non-terminal being rewritten to at most one stack associated with a non-terminal symbol on the right hand side of the production.[2]

In the following, $[\cdot\cdot\rho]$ refers to a possibly unbounded stack whose top element is $\rho$ and whose remaining part is schematically written as '$\cdot\cdot$'. $[\$]$ represents a stack whose only element is the bottom of the stack. While it is possible to define SLIGs in general, we define them for the particular case where the rules are binary branching and where the left hand sides are always incomparable.

A *stochastic linear indexed grammar*, $G$, is denoted by $(V_N, V_T, V_I, S, Prod)$, where $V_N$ is a finite set of non-terminal symbols; $V_T$ is a finite set of terminal symbols; $V_I$ is a finite set of stack symbols; $S \in V_N$ is the start symbol; $Prod$ is a finite set of productions of the form:

$$\begin{array}{lcll} X_0[\$\rho_0] & \rightarrow & a & \\ X_0[\cdot\cdot\rho_0] & \rightarrow & X_1[\cdot\cdot\rho_1] & X_2[\$\rho_2] \\ X_0[\cdot\cdot\rho_0] & \rightarrow & X_1[\$\rho_1] & X_2[\cdot\cdot\rho_2] \\ X_0[\$\rho_0] & \rightarrow & X_1[\$\rho_1] & X_2[\$\rho_2] \end{array}$$

where $X_k \in V_N$, $a \in V_T$ and $\rho_0 \in V_I$, $\rho_1, \rho_2 \in V_I^\star$; $P$, a probability distribution which assigns a probability, $0 \leq P(X[\cdot\cdot x] \rightarrow \Delta) \leq 1$, to a rule, $X[\cdot\cdot x] \rightarrow \Delta \in Prod$ such that the sum of the probabilities of all the rules that can be applied to any non-terminal annotated with a stack is equal to one. More precisely if, $\forall X \in V_N, \forall \rho \in V_I$:

$$\sum_{\Delta} P(X[\cdot\cdot\rho] \rightarrow \Delta) = 1$$

$P(X[\cdot\cdot\rho] \rightarrow \Delta)$ should be interpreted as the probability that $X[\cdot\cdot\rho]$ is rewritten as $\Delta$.

A derivation starts from $S$ associated with the empty stack ($S[\$]$) and each level of the derivation must be validated by a production rule. The *language* of a SLIG is defined as follows: $L = \{w \in V_T^\star \mid S[\$] \overset{*}{\Rightarrow} w\}$.

The *probability of a derivation* is defined as the product of the probabilities of all individual rules involved (counting repetition) in the derivation, the derivation being validated by a correct configuration of the stack at each level. The *probability of a sentence* is then computed as the sum of the probabilities of all derivations of the sentence.

Following the construction described in (Vijay-Shanker and Weir, 1991), given a LTAG, $G_{tag}$, we construct an equivalent LIG, $G_{slig}$. The constructed LIG generates the same language as $G_{tag}$ and each derivation of $G_{tag}$ corresponds to a unique LIG derivation corresponds to a unique derivation in $G_{slig}$ (and conversely). In addition, a probability is assigned to each production of the LIG. For simplicity of explanation and without loss of generality we assume that each node in an elementary tree in $G_{tag}$ is either a leaf node (i.e. either a foot node or a non-empty terminal node) or binary branching.[3] The construction of the equivalent SLIG follows.

The non-terminal symbols of $G_{slig}$ are the two symbols 'top' ($t$) and 'bottom' ($b$), the set of terminal symbols is the same as the one of $G_{tag}$, the set of stack symbols is the set of nodes (not node labels) found in the elementary trees of $G_{tag}$ augmented with the bottom of the stack ($\$$), and the start symbol is 'top' ($t$).

For all root nodes $\eta_0$ of an initial tree whose root is labeled by $S$, the following starting rules are added:

$$t[\$] \xrightarrow{P} t[\$\eta_0] \qquad (1)$$

These rules state that a derivation must start from the top of the root node of some initial tree. $P$ is the probability that a derivation starts from the initial tree associated with a lexical item and rooted by $\eta_0$.

Then, for all node $\eta$ in an elementary tree, the following rules are generated.

- If $\eta_1\eta_2$ are the 2 children of a node $\eta$ such that $\eta_2$ is on the spine (i.e. subsumes the foot node), include:
  $$b[\cdot\cdot\eta] \xrightarrow{p=1} t[\$\eta_1]t[\cdot\cdot\eta_2] \qquad (2)$$
  Since (2) encodes an immediate domination link defined by the tree-adjoining grammar, its associated probability is one.

- Similarly, if $\eta_1\eta_2$ are the 2 children of a node $\eta$ such that $\eta_1$ is on the spine (i.e. subsumes the foot node), include:
  $$b[\cdot\cdot\eta] \xrightarrow{p=1} t[\cdot\cdot\eta_1]t[\$\eta_2] \qquad (3)$$
  Since (3) encodes an immediate domination link defined by the tree-adjoining grammar, its associated probability is one.

[2] LIGs have been shown to be weakly equivalent to Tree-Adjoining Grammars (Vijay-Shanker, 1987).

[3] The algorithms explained in this paper can be generalized to lexicalized tree-adjoining grammars that need not be in Chomsky Normal Form using techniques similar the one found in (Schabes, 1991).

- If $\eta_1\eta_2$ are the 2 children of a node $\eta$ such that none of them is on the spine, include:
  $$b[\$\eta] \overset{p=1}{\to} t[\$\eta_1]t[\$\eta_2] \qquad (4)$$
  Since (4) also encodes an immediate domination link defined by the tree-adjoining grammar, its associated probability is one.

- If $\eta$ is a node labeled by a non-terminal symbol and if it does not have an obligatory adjoining constraint, then we need to consider the case that adjunction might not take place. In this case, include:
  $$t[\cdot\cdot\eta] \overset{p}{\to} b[\cdot\cdot\eta] \qquad (5)$$
  The probability of rule (5) corresponds to the probability that no adjunction takes place at node $\eta$.

- If $\eta$ is an node on which the auxiliary tree $\beta$ can be adjoined, the adjunction of $\beta$ can be predicted, therefore (assuming that $\eta_r$ is the root node of $\beta$) include:
  $$t[\cdot\cdot\eta] \overset{p}{\to} t[\cdot\cdot\eta\eta_r] \qquad (6)$$
  The probability of rule (6) corresponds to the probability of adjoining the auxiliary tree whose root node is $\eta_r$, say $\beta$, on the node $\eta$ belonging to some elementary tree, say $\alpha$.[4]

- If $\eta_f$ is the foot node of an auxiliary tree $\beta$ that has been adjoined, then the derivation of the node below $\eta_f$ must resume. In this case, include:
  $$b[\cdot\cdot\eta_f] \overset{p=1}{\to} b[\cdot\cdot] \qquad (7)$$
  The above stochastic production is included with probability one since the decision of adjunction has already been made in rules of the form (6).

- Finally, if $\eta_1$ is the root node of an initial tree that can be substituted on a node marked for substitution $\eta$, include:
  $$t[\$\eta] \overset{p}{\to} t[\$\eta_1] \qquad (8)$$
  Here, $p$ is the probability that the initial tree rooted by $\eta_1$ is substituted at node $\eta$. It corresponds to the probability of substituting the lexicalized initial tree whose root node is $\eta_1$, say $\delta$, at the node $\eta$ of a lexicalized elementary tree, say $\alpha$.[5]

The SLIG constructed as above is well defined if the following equalities hold for all nodes $\eta$:

$$P(t[\cdot\cdot\eta] \to b[\cdot\cdot\eta]) + \sum_{\eta_1} P(t[\cdot\cdot\eta] \to t[\cdot\cdot\eta\eta_1]) = 1 \quad (9)$$

$$\sum_{\eta_1} P(t[\$\eta] \to t[\$\eta_1]) = 1 \qquad (10)$$

$$\sum_{\eta_0} P(t[\$] \to t[\$\eta_0]) = 1 \qquad (11)$$

A grammar satisfying (12) is called *consistent*.[6]

$$\sum_{w\in\Sigma^*} P(t[\$] \overset{*}{\Rightarrow} w) = 1 \qquad (12)$$

Beside the distributional phenomena that we mentioned earlier, SLTAG also captures the effect of adjoining constraints (selective, obligatory or null adjoining) which are required for tree-adjoining grammar.[7]

# 3 Algorithm for Computing the Probability of a Sentence

We now define an bottom-up algorithm for SLTAG which computes the probability of an input string. The algorithm is an extension of the CKY-type parser for tree-adjoining grammar (Vijay-Shanker, 1987). The extended algorithm parses all spans of the input string and also computes their probability in a bottom-up fashion.

Since the string on the frontier of an auxiliary is broken up into two substrings by the foot node, for the purpose of computing the probability of the sentence, we will consider the probability that a node derives two substrings of the input string. This entity will be called the *inside probability*. Its exact definition is given below.

We will refer to the subsequence of the input string $w = a_1 \cdots a_N$ from position $i$ to $j$, $w_i^j$. It is defined as follows:

$$w_i^j \overset{def}{=} \begin{cases} a_{i+1}\cdots a_j & \text{, if } i < j \\ \varepsilon & \text{, if } i \geq j \end{cases}$$

Given a string $w = a_1 \cdots a_N$ and a SLTAG rewritten as in (1-8) the *inside probability*, $I^w(pos, \eta, i, j, k, l)$, is defined for all nodes $\eta$ contained in an elementary tree $\alpha$ and for $pos \in \{t, b\}$, and for all indices $0 \leq i \leq j \leq k \leq l \leq N$ as follows:

(i) If the node $\eta$ does not subsume the foot node of $\alpha$ (if there is one), then $j$ and $k$ are unbound and:
$$I^w(pos, \eta, i, -, -, l) \overset{def}{=} P(pos[\$\eta] \overset{*}{\Rightarrow} w_i^l)$$

(ii) If the node $\eta$ subsumes the foot node $\eta_f$ of $\alpha$, then:
$$I^w(pos, \eta, i, j, k, l) \overset{def}{=} P(\ pos[\$\eta] \overset{*}{\Rightarrow} w_i^j b[\$\eta_f] w_k^l)$$

In (ii), only the top element of the stack matters since as a consequence of the construction of the SLIG, we have that if $pos[\$\eta] \overset{*}{\Rightarrow} w_i^j b[\$\eta_f] w_k^l$ then for all string $\gamma \in V_I^*$ we also have $pos[\$\gamma\eta] \overset{*}{\Rightarrow} w_i^j b[\$\gamma\eta_f] w_k^l$.[8]

Initially, all inside probabilities are set to zero. Then, the computation goes bottom-up starting from the productions introducing lexical items: if $\eta$ is a node such that $b[\$\eta] \to a$, then:

$$I^w(b, \eta, i, -, -, l) = \begin{cases} 1 & \text{if } l = i+1 \wedge a = w_i^{i+1} \\ 0 & otherwise. \end{cases} \qquad (13$$

Then, the inside probabilities of larger substrings are computed bottom-up relying on the recurrence equa-

[4] Since the grammar is lexicalized, both trees $\alpha$ and $\beta$ are associated with lexical items, and the site node for adjunction $\eta$ corresponds to some syntactic modification. Such rule encapsulates $S$ modifiers (e.g. sentential adverbs as in "*apparently* John left"), $VP$ modifiers (e.g. verb phrase adverbs as in "John left *abruptly*)", $NP$ modifiers (e.g. relative clauses as in "The man *who left* was happy"), $N$ modifiers (e.g. adjectives as in "*pretty* woman"), or even sentential complements (e.g. *John thinks that* Harry is sick).

[5] Among other cases, the probability of this rule corresponds to the probability of filling some argument position by a lexicalized tree. It will encapsulate the distribution for selectional restriction since the position of substitution is taken into account.

[6] We will not investigate the conditions under which (12) holds. We conjecture that the techniques used for checking the consistency of stochastic context-free grammars (Booth and Thompson, 1973) can be adapted to SLTAG.

[7] For example, for a given node $\eta$ setting to zero the probability of all rules of the form (6) has the effect of blocking adjunction.

[8] This can be seen by observing that for any node on the path from the root node to the foot node of an auxiliary tree, the stack remains unchanged.

tions stated in Appendix A. This computation takes in the worst case $O(|G|^2N^6)$-time and $O(|G|N^4)$-space for a sentence of length $N$.

Once the inside probabilities computed, we obtain the ***probability of the sentence*** as follows:

$$P(w) \overset{def}{=} P(t[\$] \overset{*}{\Rightarrow} w) = I^w(t, \$, 0, -, -, |w|) \qquad (14)$$

We now consider the problem of re-estimating a SLTAG.

# 4 Inside-Ouside Algorithm for Reestimating a SLTAG

Given a set of positive example sentences, $W = \{w_1 \cdots w_K\}$, we would like to compute the probability of each rule of a given SLTAG in order to maximize the probability that the corpus were generated by this SLTAG. An algorithm solving this problem can be used in two different ways.

The first use is as a reestimation algorithm. In this approach, the input SLTAG derives structures that are reasonable according to some criteria (such as a linguistic theory and some a priori knowledge of the corpus) and the intended use of the algorithm is to refine the probability of each rule.

The second use is as a learning algorithm. At the first iteration, a SLTAG which generates all possible structures over a given set of nodes and terminal symbols is used. Initially the probability of each rule is randomly assigned and then the algorithm will re-estimate these probabilities.

Informally speaking, given a first estimate of the parameters of a SLTAG, the algorithm re-estimates these parameters on the basis of the parses of each sentence in a training corpus obtained by a CKY-type parser. The algorithm is designed to derive a new estimate after each iteration such that the probability of the corpus is increased or equivalently such that the cross entropy estimate (negative log probability) is decreased:

$$H(W,G) \quad = \quad -\frac{\sum\limits_{w \in W} log_2(P(w))}{\sum\limits_{w \in W} |w|} \qquad (15)$$

In order to derive a new estimate, the algorithm needs to compute for all sentences in $W$ the inside probabilities and the ***outside probabilities***. Given a string $w = a_1 \cdots a_N$, the ***outside probability***, $O^w(pos, \eta, i, j, k, l)$, is defined for all nodes $\eta$ contained in an elementary tree $\alpha$ and for $pos \in \{t, b\}$, and for all indices $0 \leq i \leq j \leq k \leq l \leq N$ as follows:

(i) If the node $\eta$ does not subsume the foot node of $\alpha$ (if there is one), then $j$ and $k$ are unbound and:

$$O^w(pos, \eta, i, -, -, l) \overset{def}{=}$$
$$P(\exists \gamma \in V_I^\star \ s.t.\ t[\$] \overset{*}{\Rightarrow} w_0^i \ pos[\$\gamma\eta] \ w_l^N)$$

(ii) If the node $\eta$ does subsume the foot node $\eta_f$ of $\alpha$ then:

$$O^w(pos, \eta, i, j, k, l) \overset{def}{=}$$
$$P(\exists \gamma \in V_I^\star \ s.t.$$
$$t[\$] \overset{*}{\Rightarrow} w_0^i \ pos[\$\gamma\eta] \ w_l^N \quad \text{and } b[\$\gamma\eta_f] \overset{*}{\Rightarrow} w_j^k)$$

Once the inside probabilities computed, the outside probabilities can be computed top-down by considering smaller spans of the input string starting with $O^w(t, \$, 0, -, -, N) = 1$ (by definition). This is done by computing the recurrence equations stated in Appendix B.

In the following, we assume that $\eta$ subsumes the foot node $\eta_f$ within a same elementary tree, and also that $\eta\prime$ subsumes the foot node $\eta_f\prime$ (within a same elementary tree). The other cases are handled similarly. Table 1 shows the reestimation formulae for the adjoining rules (16) and the null adjoining rules (17).

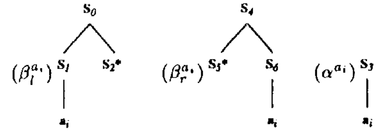
(16) corresponds to the average number of time that the rule $t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta\prime]$ is used, and (17) to the average number of times no adjunction occurred on $\eta$. The denominators of (16) and of (17) estimate the average number of times that a derivation involves the expansion of $t[\cdot\cdot\eta]$. The numerator of (16) estimates the average number of times that a derivation involves the rule $t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta\prime]$. Therefore, for example, (16) estimates the probability of using the rule $t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta\prime]$.

The algorithm reiterates until $H(W, G)$ is unchanged (within some epsilon) between two iterations. Each iteration of the algorithm requires at most $O(|G|N^6)$-time for each sentence of length $N$.

# 5 Grammar Inference with SLTAG

The reestimation algorithm explained in Section 4 can be used both to reestimate the parameters for a SLTAG derived by some other mean or to infer a grammar from scratch. In the following, we investigate grammar Inference from scratch.

The initial grammar for the reestimation algorithm consists of all SLIG rules for the trees in Lexicalized Normal Form (in short LNF) over a given set $\Sigma = \{a_i | 1 \leq i \leq T\}$ of terminal symbols, with suitably assigned non zero probability:[9]



The above normal form is capable not only to derive any lexicalized tree-adjoining language, but also to impose any binary bracketing over the strings of the language. The latter property is important as we would like to be able to use bracketing information in the input corpus as in (Pereira and Schabes, 1992).

The worst case complexity of the reestimation algorithm given in Section 4 with respect to the length of the input string ($O(N^6)$) makes this approach in general impractical for LNF grammars.

However, if only trees of the form $\beta_l^{a_i}$ and $\alpha_l^{a_i}$ (or only of the form $\beta_r^{a_i}$ and $\alpha_l^{a_i}$), the language generated is a context-free language and can be handled more efficiently by the reestimation algorithm.

[9] Adjoining constraints can be used in this normal form. They will be reflected in the SLIG equivalent grammar. Indices have been added on $S$ nodes in order to be able to uniquely refer to each node in the grammar.

$$\widehat{P}(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta\prime]) = \frac{\sum_{w \in W} \frac{1}{P(w)} \times Q^w(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta\prime])}{\sum_{w \in W} \frac{1}{P(w)} \times [R^w(\eta) + \sum_{\eta\prime} Q^w(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta\prime])]} \tag{16}$$

$$\widehat{P}(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]) = \frac{\sum_{w \in W} \frac{1}{P(w)} \times R^w(\eta)}{\sum_{w \in W} \frac{1}{P(w)} \times [R^w(\eta) + \sum_{\eta\prime} Q^w(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta\prime])]} \tag{17}$$

$$Q^w(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta\prime]) = \sum_{i,r,j,k,s,l} P(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta\prime]) \times I^w(t,\eta\prime,i,r,s,l) \times I^w(b,\eta,r,j,k,s) \times O^w(t,\eta,i,j,k,l) \tag{18}$$

$$R^w(\eta) = \sum_{i,j,k,l} P(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]) \times I^w(t,\eta,i,j,k,l) \times O^w(b,\eta,i,j,k,l) \tag{19}$$

Table 1: Reestimation of adjoining rules (16) and null adjoining rules (17)

It can be shown that if, only trees of the form $\beta_l^{a_i}$ and $\alpha^{a_i}$ are considered, the reestimation algorithm requires in the worst case $O(N^3)$-time.[10]

The system consisting of trees of the form $\beta_l^{a_i}$ and $\alpha^{a_i}$ can be seen as a *stochastic lexicalized context-free grammars* since it generates exactly context-free languages while being lexically sensitive.

In the following, due to the lack of space, we report only few experiments on grammar inference using these restricted forms of SLTAG and the reestimation algorithm given in Section 4. We compare the results of the TAG inside-outside algorithm with the results of the inside-outside algorithm for context-free grammars (Baker, 1979).

These preliminary experiments suggest that SLTAG achieves faster convergence (and also to a better solution) than stochastic context-free grammars.

## 5.1 Inferring the Language $\{a^n b^n | n \geq 0\}$

We consider first an artificial language. The training corpus consists of 100 sentences in the language $L = \{a^n b^n | n \geq 0\}$ randomly generated by a stochastic context-free grammar.

The initial grammar consists of the trees $\beta_l^a$, $\beta_l^b$, $\alpha^a$ and $\alpha^b$ with random probability of adjoining and null adjoining.

The inferred grammar models correctly the language L. Its rules of the form (1), (5) or (6) with high probability follow (any excluded rule of the same form has probability at least $10^{-33}$ times lower than the rules given below). The structural rules of the form (2), (3), (4) or (7) are not shown since their probability always remain 1.

$$\begin{aligned}
&t[\$] \xrightarrow{1.00} t[\$\eta_3^b]\\
&t[\cdot\cdot\eta_0^b] \xrightarrow{1.00} t[\cdot\cdot\eta_0^b\eta_0^a]\\
&t[\$\eta_1^b] \xrightarrow{0.63} t[\$\eta_1^b\eta_0^a]\\
&t[\$\eta_1^b] \xrightarrow{0.37} t[\$\eta_1^b\eta_0^b]\\
&t[\$\eta_3^b] \xrightarrow{0.64} t[\$\eta_3^b\eta_0^a]\\
&t[\$\eta_3^b] \xrightarrow{0.36} t[\$\eta_3^b\eta_0^b]\\
&t[\cdot\cdot\eta_0^a] \xrightarrow{1.00} b[\cdot\cdot\eta_0^a]\\
&t[\$\eta_1^a] \xrightarrow{1.00} b[\$\eta_1^a]\\
&t[\cdot\cdot\eta_2^a] \xrightarrow{1.00} b[\cdot\cdot\eta_2^a]\\
&t[\cdot\cdot\eta_2^b] \xrightarrow{1.00} b[\cdot\cdot\eta_2^b]
\end{aligned}$$

In the above grammar, a node $S_k$ in a tree $\alpha^a$ or $\beta_l^a$ associated with the symbol $a$ is referred as $\eta_k^a$, and a node $S_k$ in a tree associated with $b$ as $\eta_k^b$.

We also conducted a similar experiment with the inside-outside algorithm for context-free grammar (Baker, 1979), starting with all possible Chomsky Normal Form rules over 4 non-terminals and the set of terminal symbols $\{a, b\}$ (72 rules). The inferred grammar does not quite correctly model the language $L$. Furthermore, the algorithm does not converge as fast as in the case of SLTAG (See Figure 1).

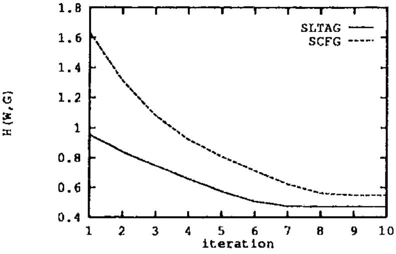


Figure 1: Convergence for the Language $\{a^n b^n | n \geq 0\}$

## 5.2 Experiments on the ATIS Corpus

We consider the part-of-speech sequences of the spoken-language transcriptions in the Texas Instruments sub-

[10] This can be seen by observing that, for example in $I(pos, \eta, i, j, k, l)$, it is necessary the case that $k = l$, and also by noting that $k$ is superfluous.

set of the Air Travel Information System (ATIS) corpus (Hemphill, Godfrey, and Doddington, 1990). This corpus is of interest since it has been used for inferring stochastic context-free grammars from partially bracketed corpora (Pereira and Schabes, 1992). We use the data given by Pereira and Schabes (1992) on raw text and compare with an inferred SLTAG.

The initial grammar consists of all trees (96) of the form $\beta_l^a$, $\alpha^a$ for all 48 terminal symbols for part-of-speech. As shown in Figure 2, the grammar converges very rapidly to a lower value of the log probability than the stochastic context-free grammar reported by Pereira and Schabes (1992).

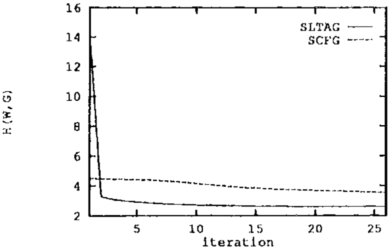


Figure 2: Convergence for ATIS Corpus

## 6 Conclusion

A novel statistical language model and fundamental algorithms for this model have been presented.

SLTAGs provide a stochastic model both hierarchical and sensitive to lexical information. They combine the advantages of purely lexical models such as N-gram distributions or Hidden Markov Models and the one of hierarchical modes as stochastic context-free grammars without their inherent limitations. The parameters of a SLTAG correspond to the probability of combining two structures each one associated with a word and therefore capture linguistically relevant distributions over words.

An algorithm for computing the probability of a sentence generated by a SLTAG was presented as well as an iterative algorithm for estimating the parameters of a SLTAG given a training corpus of raw text. Similarly to its context-free counterpart, the reestimation algorithm can be extended to handle partially parsed corpora (Pereira and Schabes, 1992).

Preliminary experiments with a context-free subset of SLTAG confirms that SLTAG enables faster convergence than stochastic context-free grammars (SCFG). This is the case since SCFG are unable to represent lexical influences on distribution except by a statistically and computationally impractical proliferation of nonterminal symbols, whereas SLTAG allows for a lexically sensitive distributional analysis while maintaining a hierarchical structure.

Furthermore, the techniques explained in this paper apply to other grammatical formalisms such as combinatory categorial grammars and modified head grammars since they have been proven to be equivalent to tree-adjoining grammars and linear indexed grammars (Joshi, Vijay-Shanker, and Weir, 1991).

Due to the lack of space, only few experiments with SLTAG were reported. A full version of the paper will be available by the time of the meeting and more experimental details will be reported during the presentation of the paper.

In collaboration with Aravind Joshi, Fernando Pereira and Stuart Shieber, we are currently investigating additional algorithms and applications for SLTAG, methods for lexical clustering and automatic construction of a SLTAG from a large training corpus.

# References

Aho, A. V. 1968. Indexed grammars -- An extension to context free grammars. *J. ACM*, 15:647–671.

Baker, J.K. 1979. Trainable grammars for speech recognition. In Jared J. Wolf and Dennis H. Klatt, editors, *Speech communication papers presentaed at the 97th Meeting of the Acoustical Society of America*, MIT, Cambridge, MA, June.

Booth, Taylor R. and Richard A. Thompson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450, May.

Booth, T. 1969. Probabilistic representation of formal languages. In *Tenth Annual IEEE Symposium on Switching and Automata Theory*, October.

Chomsky, N., 1964. *Syntactic Structures*, chapter 2-3, pages 13–18. Mouton.

Gazdar, G. 1985. Applicability of indexed grammars to natural languages. Technical Report CSLI-85-34, Center for Study of Language and Information.

Hemphill, Charles T., John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *DARPA Speech and Natural Language Workshop*, Hidden Valley, Pennsylvania, June.

Jelinek, F., J. D. Lafferty, and R. L. Mercer. 1990. Basic methods of probabilistic context free grammars. Technical Report RC 16374 (72684), IBM, Yorktown Heights, New York 10598.

Joshi, Aravind K. and Yves Schabes. 1991. Tree-adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Definability and Recognizability of Sets of Trees*. Elsevier. Forthcoming.

Joshi, Aravind K., K. Vijay-Shanker, and David Weir. 1991. The convergence of mildly context-sensitive grammatical formalisms. In Peter Sells, Stuart Shieber, and Tom Wasow, editors, *Foundational Issues in Natural Language Processing*. MIT Press, Cambridge MA.

Joshi, Aravind K. 1987. An Introduction to Tree Adjoining Grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

Lari, K. and S. J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56.

Pereira, Fernando and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In $20^{th}$ *Meeting of the Association for Computational Linguistics (ACL'92)*, Newark, Delaware.

Pratt, Fletcher. 1942. *Secret and urgent, the story of codes and ciphers.* Blue Ribbon Books.

Resnik, Philip. 1991. Lexicalized tree-adjoining grammar for distributional analysis. In *Penn Review of Linguistics*, Spring.

Schabes, Yves, Anne Abeillé, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proceedings of the* $12^{th}$ *International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August.

Schabes, Yves. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars.* Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, August. Available as technical report (MS-CIS-90-48, LINC LAB179) from the Department of Computer Science.

Schabes, Yves. 1991. An inside-outside algorithm for estimating the parameters of a hidden stochastic context-free grammar based on Earley's algorithm. Manuscript.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Shannon, C. E. 1951. Prediction and entropy of printed english. *The Bell System Technical Journal*, 30:50–64.

Vijay-Shanker, K. and David J. Weir. 1991. Parsing constrained grammar formalisms. In preparation.

Vijay-Shanker, K. 1987. *A Study of Tree Adjoining Grammars.* Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

# A Computing the Inside Probabilities

In the following, the inside and outside probabilities are relative to the input string $w$. $\mathcal{F}$ stands for the the set of foot nodes, $\mathcal{S}$ for the set of nodes on which substitution can occur, $\mathcal{R}$ for the set of root nodes of initial trees, and $\mathcal{A}$ for the set of non-terminal nodes of auxiliary trees. The inside probability can be computed bottom-up with the following recurrence equations. For all node $\eta$ found in an elementary tree, it can be shown that:

1. If $b[\$\eta] \rightarrow a$, $I(b,\eta,i,-,-,l) = d1$ if $l = i+1$ and if $a = w_i^{i+1}$, 0 otherwise.
2. If $\eta_f \in \mathcal{F}$, $I(b,\eta_f,i,j,k,l) = 1$ if $i = j$ and if $k = l, 0$ otherwise.
3. If $b[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta_1]t[\$\eta_2]$: $I(b,\eta,i,j,k,l) =$
$$\sum_{m=k}^{l-1} I(t,\eta_1,i,j,k,m) \times I(t,\eta_2,m,-,-,l)$$
4. If $b[\cdot\cdot\eta] \rightarrow t[\$\eta_1]t[\cdot\cdot\eta_2]$, $I(b,\eta,i,j,k,l) =$
$$\sum_{m=i+1}^{j} I(t,\eta_1,i,-,-,m) \times I(t,\eta_2,m,j,k,l)$$
5. If $b[\$\eta] \rightarrow t[\$\eta_1]t[\$\eta_2]$, $I(b,\eta,i,-,-,l) =$
$$\sum_{m=i+1}^{l-1} I(t,\eta_1,i,-,-,m) \times I(t,\eta_2,m,-,-,l)$$
6. For all node $\eta$ on which adjunction can be performed: $I(t,\eta,i,j,k,l) =$
$$I(b,\eta,i,j,k,l) \times P(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]) + \sum_{r=i}^{j}\sum_{s=k}^{l}\sum_{\eta_1} \left( \begin{array}{ll} & I(t,\eta_1,i,r,s,l) \\ \times & I(b,\eta,r,j,k,s) \\ \times & P(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta_1]) \end{array} \right)$$
7. For all node $\eta \in \mathcal{S}$: $I(t,\eta,i,-,-,l) =$
$$\sum_{\eta_1} I(t,\eta_1,i,-,-,l) \times P(t[\$\eta] \rightarrow t[\$\eta_1])$$
8. $I(t,\$,i,-,-,l) = \sum_{\eta} I(t,\eta,i,-,-,l) \times P(t[\$] \rightarrow t[\$\eta])$

# B Computing the Outside Probabilities

The outside probabilities can be computed top-down recursively over smaller spans of the input string once the inside probabilities have been computed. First, by definition we have: $O(t,\$,0,-,-,N) = 1$. The following recurrence equations hold for all node $\eta$ found in an elementary tree.

1. If $\eta \in \mathcal{R}$, $O(t,\eta,0,-,-,N) = P(t[\$] \rightarrow t[\$\eta])$.
   And for all $(i,j) \neq (0,N)$, $O(t,\eta,i,-,-,j) =$
$$O(t,\eta_0,i,-,-,j) \times P(t[\$\eta_0] \rightarrow t[\$\eta])$$
2. If $\eta$ is an interior node which subsumes the foot node of the elementary tree it belongs to, $O(t,\eta,i,j,k,l) =$
$$\sum_{q=l+1}^{N} \left( \begin{array}{ll} & O(b,\eta_0,i,j,k,q) \\ \times & I(t,\eta_2,l,-,-,q) \\ \times & P(b[\cdot\cdot\eta_0] \rightarrow t[\cdot\cdot\eta]t[\$\eta_2]) \end{array} \right) + \sum_{p=0}^{i-1} \left( \begin{array}{ll} & O(b,\eta_0,p,j,k,l) \\ \times & I(t,\eta_1,p,-,-,i) \\ \times & P(b[\cdot\cdot\eta_0] \rightarrow t[\$\eta_1]t[\cdot\cdot\eta]) \end{array} \right)$$
3. If $\eta$ is an interior node which does not subsume the foot node of the elementary tree it belongs to, we have: $O(t,\eta,i,-,-,l) =$
$$\sum_{q=l+1}^{N} \left( \begin{array}{ll} & O(b,\eta_0,i,-,-,q) \\ \times & I(t,\eta_2,l,-,-,q) \\ \times & P(b[\$\eta_0] \rightarrow t[\$\eta]t[\$\eta_2]) \end{array} \right) + \sum_{p=0}^{i-1} \left( \begin{array}{ll} & O(b,\eta_0,p,-,-,l) \\ \times & I(t,\eta_1,p,-,-,i) \\ \times & P(b[\$\eta_0] \rightarrow t[\$\eta_1]t[\$\eta]) \end{array} \right)$$
$$+ \sum_{j=l}^{N}\sum_{k=j+1}^{N}\sum_{q=k}^{N} \left( \begin{array}{ll} & O(b,\eta_0,i,j,k,q) \\ \times & I(t,\eta_2,l,j,k,q) \\ \times & P(b[\cdot\cdot\eta_0] \rightarrow t[\$\eta]t[\cdot\cdot\eta_2]) \end{array} \right) + \sum_{p=0}^{i-1}\sum_{j=p}^{i}\sum_{k=j}^{i} \left( \begin{array}{ll} & O(b,\eta_0,p,j,k,l) \\ \times & I(t,\eta_1,p,j,k,i) \\ \times & P(b[\cdot\cdot\eta_0] \rightarrow t[\cdot\cdot\eta_1]t[\$\eta]) \end{array} \right)$$
4. If $\eta \in \mathcal{A}$, then: $O(t,\eta,i,j,k,l) =$
$$\sum_{\eta_0}\sum_{p=j}^{k-1}\sum_{q=p+1}^{k} \left( \begin{array}{ll} & O(t,\eta_0,i,p,q,l) \\ \times & I(t,\eta_0,j,p,q,k) \\ \times & P(t[\cdot\cdot\eta_0] \rightarrow t[\cdot\cdot\eta_0\eta]) \end{array} \right) + \sum_{\eta_0} \left( \begin{array}{ll} & O(t,\eta_0,i,-,-,l) \\ \times & I(t,\eta_0,j,-,-,k) \\ \times & P(t[\$\eta_0] \rightarrow t[\$\eta_0\eta]) \end{array} \right)$$
5. If $\eta$ is a node which subsumes the foot node of the elementary tree it belongs to, we have: $O(b,\eta,i,j,k,l) =$
$$O(t,\eta,i,j,k,l) \times P(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]) + \sum_{\eta_0}\sum_{p=0}^{i}\sum_{q=l}^{N} \left( \begin{array}{ll} & O(t,\eta,p,j,k,q) \\ \times & I(t,\eta_0,p,i,l,q) \\ \times & P(t[\cdot\cdot\eta_0] \rightarrow t[\cdot\cdot\eta_0\eta]) \end{array} \right)$$
6. And finally, if $\eta$ is a node which does not subsume the foot node of the elementary tree it belongs to: $O(b,\eta,i,-,-,l) =$
$$O(t,\eta,i,-,-,l) \times P(t[\$\eta] \rightarrow b[\$\eta]) + \sum_{\eta_0}\sum_{p=0}^{i-1}\sum_{q=l}^{N} \left( \begin{array}{ll} & O(t,\eta,p,-,-,q) \\ \times & I(t,\eta_0,p,i,l,q) \\ \times & P(t[\$\eta_0] \rightarrow t[\$\eta_0\eta]) \end{array} \right)$$