Prefix Probabilities from Stochastic Tree Adjoining Grammars*

Mark-Jan Nederhof
DFKI
Stuhlsatzenhausweg 3,
D-66123 Saarbrücken,
Germany
nederhof@dfki.de

Anoop Sarkar
Dept. of Computer and Info. Sc.
Univ of Pennsylvania
200 South 33rd Street,
Philadelphia, PA 19104 USA
anoop@linc.cis.upenn.edu

Giorgio Satta
Dip. di Elettr. e Inf.
Univ. di Padova
via Gradenigo 6/A,
35131 Padova, Italy
satta@dei.unipd.it

Abstract

Language models for speech recognition typically use a probability model of the form $\Pr(a_n|a_1, a_2, \ldots, a_{n-1})$. Stochastic grammars, on the other hand, are typically used to assign structure to utterances. A language model of the above form is constructed from such grammars by computing the prefix probability $\sum_{w \in \Sigma^*} \Pr(a_1 \cdots a_n w)$, where w represents all possible terminations of the prefix $a_1 \cdots a_n$. The main result in this paper is an algorithm to compute such prefix probabilities given a stochastic Tree Adjoining Grammar (TAG). The algorithm achieves the required computation in $\mathcal{O}(n^6)$ time. The probability of subderivations that do not derive any words in the prefix, but contribute structurally to its derivation, are precomputed to achieve termination. This algorithm enables existing corpus-based estimation techniques for stochastic TAGs to be used for language modelling.

1 Introduction

Given some word sequence $a_1 \cdots a_{n-1}$, speech recognition language models are used to hypothesize the next word a_n, which could be any word from the vocabulary Σ. This is typically done using a probability model $\Pr(a_n|a_1, \ldots, a_{n-1})$. Based on the assumption that modelling the hidden structure of natural language would improve performance of such language models, some researchers tried to use stochastic context-free grammars (CFGs) to produce language models (Wright and Wrigley, 1989; Jelinek and Lafferty, 1991; Stolcke, 1995). The probability model used for a stochastic grammar was $\sum_{w \in \Sigma^*} \Pr(a_1 \cdots a_n w)$. However, language models that are based on trigram probability models out-perform stochastic CFGs. The common wisdom about this failure of CFGs is that trigram models are lexicalized models while CFGs are not.

Tree Adjoining Grammars (TAGs) are important in this respect since they are easily lexicalized while capturing the constituent structure of language. More importantly, TAGs allow greater linguistic expressiveness. The trees associated with words can be used to encode argument and adjunct relations in various syntactic environments. This paper assumes some familiarity with the TAG formalism. (Joshi, 1988) and (Joshi and Schabes, 1992) are good introductions to the formalism and its linguistic relevance. TAGs have been shown to have relations with both phrase-structure grammars and dependency grammars (Rambow and Joshi, 1995), which is relevant because recent work on *structured* language models (Chelba et al., 1997) have used dependency grammars to exploit their lexicalization. We use stochastic TAGs as such a *structured* language model in contrast with earlier work where TAGs have been exploited in a class-based n-gram language model (Srinivas, 1996).

This paper derives an algorithm to compute prefix probabilities $\sum_{w \in \Sigma^*} \Pr(a_1 \cdots a_n w)$. The algorithm assumes as input a stochastic TAG G and a string which is a prefix of some string in $L(G)$, the language generated by G. This algorithm enables existing corpus-based estimation techniques (Schabes, 1992) in stochastic TAGs to be used for language modelling.

* Part of this research was done while the first and the third authors were visiting the Institute for Research in Cognitive Science, University of Pennsylvania. The first author was supported by the German Federal Ministry of Education, Science, Research and Technology (BMBF) in the framework of the VERBMOBIL Project under Grant 01 IV 701 V0, and by the Priority Programme Language and Speech Technology, which is sponsored by NWO (Dutch Organization for Scientific Research). The second and third authors were partially supported by NSF grant SBR8920230 and ARO grant DAAH0404-94-G-0426. The authors wish to thank Aravind Joshi for his support in this research.

2 Notation

A stochastic tree adjoining grammar (STAG) is represented by a tuple $(NT, \Sigma, \mathcal{I}, \mathcal{A}, \phi)$ where NT is a set of nonterminal symbols, Σ is a set of terminal symbols, $\mathcal{I}$ is a set of **initial** trees and $\mathcal{A}$ is a set of **auxiliary** trees. Trees in $\mathcal{I} \cup \mathcal{A}$ are also called **elementary** trees.

We refer to the root of an elementary tree t as R_t. Each auxiliary tree has exactly one distinguished leaf, which is called the **foot**. We refer to the foot of an auxiliary tree t as F_t. We let V denote the set of all nodes in the elementary trees.

For each leaf N in an elementary tree, except when it is a foot, we define $label(N)$ to be the label of the node, which is either a terminal from Σ or the empty string ϵ. For each other node N, $label(N)$ is an element from NT.

At a node N in a tree such that $label(N) \in NT$ an operation called **adjunction** can be applied, which excises the tree at N and inserts an auxiliary tree.

Function ϕ assigns a probability to each adjunction. The probability of adjunction of $t \in \mathcal{A}$ at node N is denoted by $\phi(t, N)$. The probability that at N no adjunction is applied is denoted by $\phi(\mathbf{nil}, N)$. We assume that each STAG G that we consider is **proper**. That is, for each N such that $label(N) \in NT$,

$$\sum_{t \in \mathcal{A} \cup \{\mathbf{nil}\}} \phi(t, N) = 1.$$

For each non-leaf node N we construct the string $cdn(N) = \widehat{N_1} \cdots \widehat{N_m}$ from the (ordered) list of children nodes $N_1, \ldots, N_m$ by defining, for each d such that $1 \leq d \leq m$, $\widehat{N_d} = label(N_d)$ in case $label(N_d) \in \Sigma \cup \{\epsilon\}$, and $\widehat{N_d} = N_d$ otherwise. In other words, children nodes are replaced by their labels unless the labels are nonterminal symbols.

To simplify the exposition, we assume an additional node for each auxiliary tree t, which we denote by $\perp$. This is the unique child of the actual foot node F_t. I.e. we change the definition of cdn such that $cdn(F_t) = \perp$ for each auxiliary tree t. We set

$$V^{\perp} = \{N \in V \mid label(N) \in NT\} \cup \Sigma \cup \{\perp\}.$$

We use symbols $a, b, c, \ldots$ to range over Σ, symbols $v, w, x, \ldots$ to range over Σ^*, symbols $N, M, \ldots$ to range over $V^{\perp}$, and symbols $\alpha, \beta, \gamma, \ldots$ to range over $(V^{\perp})^*$. We use $t, t', \ldots$ to denote trees in $\mathcal{I} \cup \mathcal{A}$ or subtrees thereof.

We define the predicate dft on elements from $V^{\perp}$ as $dft(N)$ if and only if (i) $N \in V$ and N dominates $\perp$, or (ii) $N = \perp$. We extend dft to strings of the form $N_1 \ldots N_m \in (V^{\perp})^*$ by defining $dft(N_1 \ldots N_m)$ if and only if there is a d $(1 \leq d \leq m)$ such that $dft(N_d)$.

For some logical expression p, we define $\delta(p) = 1$ iff p is true, $\delta(p) = 0$ otherwise.

3 Overview

The approach we adopt in the next section to derive a method for the computation of prefix probabilities for TAGs is based on transformations of equations. Here we informally discuss the general ideas underlying equation transformations.

Let $w = a_1 a_2 \cdots a_n \in \Sigma^*$ be a string and let $N \in V^{\perp}$. We use the following representation which is standard in tabular methods for TAG parsing. An **item** is a tuple $[N, i, j, f_1, f_2]$ representing the set of all trees t such that (i) t is a subtree rooted at N of some derived elementary tree; and (ii) t's root spans from position i to position j in w, t's foot node spans from position f_1 to position f_2 in w. In case N does not dominate the foot, we set $f_1 = f_2 = -$. We generalize in the obvious way to items $[t, i, j, f_1, f_2]$, where t is an elementary tree, and $[\alpha, i, j, f_1, f_2]$, where $cdn(N) = \alpha\beta$ for some N and β.

To introduce our approach, let us start with some considerations concerning the TAG parsing problem. When parsing w with a TAG G, one usually composes items in order to construct new items spanning a larger portion of the input string. Assume there are instances of auxiliary trees t and t' in G, where the yield of t', apart from its foot, is the empty string. If $\phi(t, N) > 0$ for some node N on the spine of t', and we have recognized an item $[R_t, i, j, f_1, f_2]$, then we may adjoin t at N and hence deduce the existence of an item $[R_{t'}, i, j, f_1, f_2]$ (see Fig. 1(a)). Similarly, if t can be adjoined at a node N to the left of the spine of t' and $f_1 = f_2$, we may deduce the existence of an item $[R_{t'}, i, j, j, j]$ (see Fig. 1(b)). Importantly, one or more other auxiliary trees with empty yield could wrap the tree t' before t adjoins. Adjunctions in this situation are potentially nonterminating.

One may argue that situations where auxiliary trees have empty yield do not occur in practice, and are even by definition excluded in the

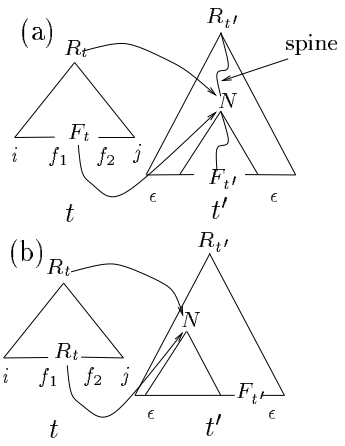

Figure 1: Wrapping in auxiliary trees with empty yield

case of lexicalized TAGs. However, in the computation of the prefix probability we must take into account trees with non-empty yield which behave like trees with empty yield because their lexical nodes fall to the right of the right boundary of the prefix string. For example, the two cases previously considered in Fig. 1 now generalize to those in Fig. 2.

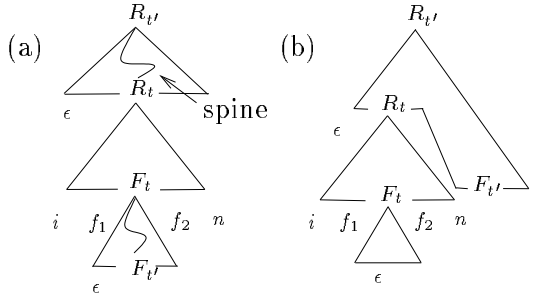

Figure 2: Wrapping of auxiliary trees when computing the prefix probability

To derive a method for the computation of prefix probabilities, we give some simple recursive equations. Each equation *decomposes* an item into other items in all possibly ways, in the sense that it expresses the probability of that item as a function of the probabilities of items associated with equal or smaller portions of the input.

In specifying the equations, we exploit techniques used in the parsing of incomplete input (Lang, 1988). This allows us to compute the prefix computation problem as a by-product of computing the inside probability.

In order to avoid the problem of nontermination outlined above, we transform our equations to remove infinite recursion, while preserving the correctness of the probability computation. The transformation of the equations is explained as follows. For an item I, the **span** of I, written $\sigma(I)$, is the 4-tuple representing the 4 input positions in I. We will define an equivalence relation on spans that relates to the portion of the input that is covered. The transformations that we apply to our equations produce two new sets of equations. The first set of equations are concerned with all possible decompositions of a given item I into set of items of which one has a span equivalent to that of I and the others have an empty span. Equations in this set represent endless recursion. The system of all such equations can be solved independently of the actual input w. This is done once for a given grammar.

The second set of equations have the property that, when evaluated, recursion always terminates. The evaluation of these equations computes the probability of the input string modulo the computation of some parts of the derivation that do not contribute to the input itself. Combination of the second set of equations with the solutions obtained from the first set allows the effective computation of the prefix probability.

4 Computing Prefix Probabilities

This section develops an algorithm for the computation of prefix probabilities for stochastic TAGs.

4.1 General equations

The prefix probability is given by:

$$\sum_{w \in \Sigma^*} \Pr(a_1 \cdots a_n w) \quad = \quad \sum_{t \in \mathcal{I}} P([t, 0, n, -, -]),$$

where P is a function over items recursively defined as follows:

$$P([t, i, j, f_1, f_2]) = P([R_t, i, j, f_1, f_2]); \tag{1}$$

$$P([\alpha N, i, j, -, -]) = \tag{2}$$
$$\sum_{k (i \le k \le j)} P([\alpha, i, k, -, -]) \cdot P([N, k, j, -, -]),$$
$$\text{if } \alpha \neq \epsilon \wedge \neg dft(\alpha N);$$

$$P([\alpha N, i, j, f_1, f_2]) = \tag{3}$$
$$\sum_{k (i \le k \le f_1)} P([\alpha, i, k, -, -]) \cdot P([N, k, j, f_1, f_2]),$$
$$\text{if } \alpha \neq \epsilon \wedge dft(N);$$

$$P([\alpha N, i, j, f_1, f_2]) \; = \tag{4}$$
$$\sum_{k(f_2 \le k \le j)} P([\alpha, i, k, f_1, f_2]) \cdot P([N, k, j, -, -]),$$
if $\alpha \neq \epsilon \wedge dft(\alpha)$;

$$P([N, i, j, f_1, f_2]) \; = \tag{5}$$
$$\phi(\mathbf{nil}, N) \cdot P([cdn(N), i, j, f_1, f_2]) \; +$$
$$\sum_{f_1', f_2'(i \le f_1' \le f_1 \wedge f_2 \le f_2' \le j)} P([cdn(N), f_1', f_2', f_1, f_2]) \; \cdot$$
$$\sum_{t \in \mathcal{A}} \phi(t, N) \cdot P([t, i, j, f_1', f_2']),$$
if $N \in V \wedge dft(N)$;

$$P([N, i, j, -, -]) \; = \tag{6}$$
$$\phi(\mathbf{nil}, N) \cdot P([cdn(N), i, j, -, -]) \; +$$
$$\sum_{f_1', f_2'(i \le f_1' \le f_2' \le j)} P([cdn(N), f_1', f_2', -, -]) \; \cdot$$
$$\sum_{t \in \mathcal{A}} \phi(t, N) \cdot P([t, i, j, f_1', f_2']),$$
if $N \in V \wedge \neg dft(N)$;

$$P([a, i, j, -, -]) \; = \tag{7}$$
$$\delta(i + 1 = j \wedge a_j = a) + \delta(i = j = n);$$

$$P([\bot, i, j, f_1, f_2]) \; = \delta(i = f_1 \wedge j = f_2); \tag{8}$$

$$P([\epsilon, i, j, -, -]) \; = \delta(i = j). \tag{9}$$

Term $P([t, i, j, f_1, f_2])$ gives the inside probability of all possible trees derived from elementary tree t, having the indicated span over the input. This is decomposed into the contribution of each single node of t in equations (1) through (6). In equations (5) and (6) the contribution of a node N is determined by the combination of the inside probabilities of N's children and by all possible adjunctions at N. In (7) we recognize some terminal symbol if it occurs in the prefix, or ignore its contribution to the span if it occurs after the last symbol of the prefix. Crucially, this allows us to reduce the computation of prefix probabilities to the computation of inside probabilities.

4.2 Terminating equations

In general, the recursive equations (1) to (9) are not directly computable. This is because the value of $P([A, i, j, f, f'])$ might indirectly depend on itself, giving rise to nontermination. We therefore rewrite the equations.

We define an equivalence relation over spans, that expresses when two items are associated with equivalent portions of the input:

$(i', j', f_1', f_2') \approx (i, j, f_1, f_2)$ if and only if

$$((i', j') = (i, j)) \wedge$$
$$((f_1', f_2') = (f_1, f_2) \vee$$
$$((f_1' = f_2' = i \vee f_1' = f_2' = j \vee f_1' = f_2' = -) \wedge$$
$$(f_1 = f_2 = i \vee f_1 = f_2 = j \vee f_1 = f_2 = -)))$$

We introduce two new functions P_{low} and P_{split}. When evaluated on some item I, P_{low} recursively calls itself as long as some other item I' with a given elementary tree as its first component can be reached, such that $\sigma(I) \approx \sigma(I')$. P_{low} returns 0 if the actual branch of recursion cannot eventually reach such an item I', thus removing the contribution to the prefix probability of that branch. If item I' is reached, then P_{low} switches to P_{split}. Complementary to P_{low}, function P_{split} tries to decompose an argument item I into items I' such that $\sigma(I) \not\approx \sigma(I')$. If this is not possible through the actual branch of recursion, P_{split} returns 0. If decomposition is indeed possible, then we start again with P_{low} at items produced by the decomposition. The effect of this intermixing of function calls is the simulation of the original function P, with P_{low} being called only on potentially nonterminating parts of the computation, and P_{split} being called on parts that are guaranteed to terminate.

Consider some derivation tree spanning some portion of the input string, and the associated derivation tree τ. There must be a unique elementary tree which is represented by a node in τ that is the "lowest" one that entirely spans the portion of the input of interest. (This node might be the root of τ itself.) Then, for each $t \in \mathcal{A}$ and for each i, j, f_1, f_2 such that $i < j$ and $i \le f_1 \le f_2 \le j$, we must have:

$$P([t, i, j, f_1, f_2]) \; = \tag{10}$$
$$\sum_{t' \in \mathcal{A}, f_1', f_2'((i,j,f_1',f_2') \approx (i,j,f_1,f_2))} P_{low}([t, i, j, f_1, f_2],\; [t', f_1', f_2']).$$

Similarly, for each $t \in \mathcal{I}$ and for each i, j such that $i < j$, we must have:

$$P([t, i, j, -, -]) \; = \tag{11}$$
$$\sum_{t' \in \{t\} \cup \mathcal{A}, f \in \{-, i, j\}} P_{low}([t, i, j, -, -],\; [t', f, f]).$$

The reason why P_{low} keeps a record of indices f_1' and f_2', i.e., the spanning of the foot node of the lowest tree (in the above sense) on which P_{low} is called, will become clear later, when we introduce equations (29) and (30).

We define $P_{low}([t, i, j, f_1, f_2], [t', f_1', f_2'])$ and $P_{low}([\alpha, i, j, f_1, f_2], [t', f_1', f_2'])$ for $i < j$ and $(i, j, f_1, f_2) \approx (i, j, f_1', f_2')$, as follows.

$$P_{low}([t,\ i,\ j,\ f_1,\ f_2],\ [t', f_1', f_2']) \ = \tag{12}$$
$$\quad P_{low}([R_t,\ i,\ j,\ f_1,\ f_2],\ [t', f_1', f_2']) +$$
$$\quad \delta((t, f_1, f_2) = (t', f_1', f_2')) \cdot$$
$$\qquad P_{split}([R_t,\ i,\ j,\ f_1,\ f_2]);$$

$$P_{low}([\alpha N, i, j, -, -],\ [t, f_1', f_2']) \ = \tag{13}$$
$$\quad P_{low}([\alpha, i, j, -, -],\ [t, f_1', f_2']) \cdot$$
$$\qquad P([N, j, j, -, -]) +$$
$$\quad P([\alpha, i, i, -, -]) \cdot$$
$$\qquad P_{low}([N, i, j, -, -],\ [t, f_1', f_2']),$$
$$\quad \text{if } \alpha \neq \epsilon \wedge \neg dft(\alpha N);$$

$$P_{low}([\alpha N, i, j, f_1, f_2],\ [t, f_1', f_2']) \ = \tag{14}$$
$$\quad \delta(f_1 = j) \cdot P_{low}([\alpha, i, j, -, -],\ [t, f_1', f_2']) \cdot$$
$$\qquad P([N, j, j, f_1, f_2]) +$$
$$\quad P([\alpha, i, i, -, -]) \cdot$$
$$\qquad P_{low}([N, i, j, f_1, f_2],\ [t, f_1', f_2']),$$
$$\quad \text{if } \alpha \neq \epsilon \wedge dft(N);$$

$$P_{low}([\alpha N, i, j, f_1, f_2],\ [t, f_1', f_2']) \ = \tag{15}$$
$$\quad P_{low}([\alpha, i, j, f_1, f_2],\ [t, f_1', f_2']) \cdot$$
$$\qquad P([N, j, j, -, -]) +$$
$$\quad \delta(i = f_2) \cdot P([\alpha, i, i, f_1, f_2]) \cdot$$
$$\qquad P_{low}([N, i, j, -, -],\ [t, f_1', f_2']),$$
$$\quad \text{if } \alpha \neq \epsilon \wedge dft(\alpha);$$

$$P_{low}([N, i, j, f_1, f_2],\ [t, f_1', f_2']) \ = \tag{16}$$
$$\quad \phi(\mathbf{nil}, N) \cdot$$
$$\qquad P_{low}([cdn(N), i, j, f_1, f_2],\ [t, f_1', f_2']) +$$
$$\quad P_{low}([cdn(N), i, j, f_1, f_2],\ [t, f_1', f_2']) \cdot$$
$$\qquad \textstyle\sum_{t' \in \mathcal{A}} \phi(t', N) \cdot P([t', i, j, i, j]) +$$
$$\quad P([cdn(N), f_1, f_2, f_1, f_2]) \cdot$$
$$\qquad \sum_{t' \in \mathcal{A}} \phi(t', N) \cdot P_{low}([t', i, j, f_1, f_2],\ [t, f_1', f_2']),$$
$$\quad \text{if } N \in V \wedge dft(N);$$

$$P_{low}([N, i, j, -, -],\ [t, f_1', f_2']) \ = \tag{17}$$
$$\quad \phi(\mathbf{nil}, N) \cdot$$
$$\qquad P_{low}([cdn(N), i, j, -, -],\ [t, f_1', f_2']) +$$
$$\quad P_{low}([cdn(N), i, j, -, -],\ [t, f_1', f_2']) \cdot$$
$$\qquad \textstyle\sum_{t' \in \mathcal{A}} \phi(t', N) \cdot P([t', i, j, i, j]) +$$
$$\quad \sum_{f_1'', f_2''(f_1'' = f_2'' = i \vee f_1'' = f_2'' = j)} P([cdn(N), f_1'', f_2'', -, -]) \cdot$$
$$\qquad \sum_{t' \in \mathcal{A}} \phi(t', N) \cdot P_{low}([t', i, j, f_1'', f_2''],\ [t, f_1', f_2']),$$
$$\quad \text{if } N \in V \wedge \neg dft(N);$$

$$P_{low}([a, i, j, -, -],\ [t, f_1', f_2']) = 0; \tag{18}$$

$$P_{low}([\perp, i, j, f_1, f_2],\ [t, f_1', f_2']) = 0; \tag{19}$$

$$P_{low}([\epsilon, i, j, -, -],\ [t, f_1', f_2']) = 0. \tag{20}$$

The definition of P_{low} parallels the one of P given in §4.1. In (12), the second term in the right-hand side accounts for the case in which the tree we are visiting is the "lowest" one on which P_{low} should be called. Note how in the above equations P_{low} must be called also on nodes that do not dominate the footnode of the elementary tree they belong to (cf. the definition of $\approx$). Since no call to P_{split} is possible through the terms in (18), (19) and (20), we must set the right-hand side of these equations to 0.

The specification of $P_{split}([\alpha, i, j, f_1, f_2])$ is given below. Again, the definition parallels the one of P given in §4.1.

$$P_{split}([\alpha N, i, j, -, -]) \ = \tag{21}$$
$$\quad \sum_{k(i < k < j)} P([\alpha, i, k, -, -]) \cdot P([N, k, j, -, -]) +$$
$$\quad P_{split}([\alpha, i, j, -, -]) \cdot P([N, j, j, -, -]) +$$
$$\quad P([\alpha, i, i, -, -]) \cdot P_{split}([N, i, j, -, -]),$$
$$\quad \text{if } \alpha \neq \epsilon \wedge \neg dft(\alpha N);$$

$$P_{split}([\alpha N, i, j, f_1, f_2]) \ = \tag{22}$$
$$\quad \sum_{k(i < k \leq f_1 \wedge k < j)} P([\alpha, i, k, -, -]) \cdot P([N, k, j, f_1, f_2]) +$$
$$\quad \delta(f_1 = j) \cdot P_{split}([\alpha, i, j, -, -]) \cdot$$
$$\qquad P([N, j, j, f_1, f_2]) +$$
$$\quad P([\alpha, i, i, -, -]) \cdot P_{split}([N, i, j, f_1, f_2]),$$
$$\quad \text{if } \alpha \neq \epsilon \wedge dft(N);$$

$$P_{split}([\alpha N, i, j, f_1, f_2]) \ = \tag{23}$$
$$\quad \sum_{k(i < k \wedge f_2 \leq k < j)} P([\alpha, i, k, f_1, f_2]) \cdot P([N, k, j, -, -]) +$$
$$\quad P_{split}([\alpha, i, j, f_1, f_2]) \cdot P([N, j, j, -, -]) +$$
$$\quad \delta(i = f_2) \cdot P([\alpha, i, i, f_1, f_2]) \cdot$$
$$\qquad P_{split}([N, i, j, -, -]),$$
$$\quad \text{if } \alpha \neq \epsilon \wedge dft(\alpha);$$

$$P_{split}([N, i, j, f_1, f_2]) \ = \tag{24}$$
$$\quad \phi(\mathbf{nil}, N) \cdot P_{split}([cdn(N), i, j, f_1, f_2]) +$$
$$\quad \sum_{f_1', f_2' \ (i \leq f_1' \leq f_1 \wedge f_2 \leq f_2' \leq j \wedge (f_1', f_2') \neq (i, j) \wedge (f_1', f_2') \neq (f_1, f_2))} P([cdn(N), f_1', f_2', f_1, f_2]) \cdot$$
$$\qquad \sum_{t \in \mathcal{A}} \phi(t, N) \cdot P([t, i, j, f_1', f_2']) +$$
$$\quad P_{split}([cdn(N), i, j, f_1, f_2]) \cdot$$
$$\qquad \sum_{t \in \mathcal{A}} \phi(t, N) \cdot P([t, i, j, i, j]),$$

if $N \in V \wedge dft(N)$;

$$P_{split}([N,i,j,-,-]) = \quad (25)$$
$$\phi(\mathbf{nil}, N) \cdot P_{split}([cdn(N),i,j,-,-]) +$$
$$\sum_{f'_1,f'_2 \ (i \le f'_1 \le f'_2 \le j \wedge (f'_1,f'_2) \neq (i,j) \wedge \neg(f'_1 = f'_2 = i \vee f'_1 = f'_2 = j))} P([cdn(N),f'_1,f'_2,-,-]) \cdot$$
$$\sum_{t \in \mathcal{A}} \phi(t,N) \cdot P([t,i,j,f'_1,f'_2]) +$$
$$P_{split}([cdn(N),i,j,-,-]) \cdot$$
$$\sum_{t \in \mathcal{A}} \phi(t,N) \cdot P([t,i,j,i,j]),$$

if $N \in V \wedge \neg dft(N)$;

$$P_{split}([a,i,j,-,-]) = \delta(i+1=j \wedge a_j = a); \quad (26)$$
$$P_{split}([\bot,i,j,f_1,f_2]) = 0; \quad (27)$$
$$P_{split}([\epsilon,i,j,-,-]) = 0. \quad (28)$$

We can now separate those branches of recursion that terminates on the given input from the cases of endless recursion. We assume below that $P_{split}([R_t,i,j,f'_1,f'_2]) > 0$. Even if this is not always valid, for the purpose of deriving the equations below, this assumption does not lead to invalid results. We define a new function P_{outer}, which accounts for probabilities of subderivations that do not derive any words in the prefix, but contribute structurally to its derivation:

$$P_{outer}([t,i,j,f_1,f_2],\ [t',f'_1,f'_2]) = \quad (29)$$
$$\frac{P_{low}([t,i,j,f_1,f_2],\ [t',f'_1,f'_2])}{P_{split}([R_{t'},i,j,f'_1,f'_2])};$$
$$P_{outer}([\alpha,i,j,f_1,f_2],\ [t',f'_1,f'_2]) = \quad (30)$$
$$\frac{P_{low}([\alpha,i,j,f_1,f_2],\ [t',f'_1,f'_2])}{P_{split}([R_{t'},i,j,f'_1,f'_2])}.$$

We can now eliminate the infinite recursion that arises in (10) and (11) by rewriting $P([t,i,j,f_1,f_2])$ in terms of P_{outer}:

$$P([t,i,j,f_1,f_2]) = \quad (31)$$
$$\sum_{t' \in \mathcal{A}, f'_1,f'_2((i,j,f'_1,f'_2) \approx (i,j,f_1,f_2))} P_{outer}([t,i,j,f_1,f_2],\ [t',f'_1,f'_2]) \cdot$$
$$P_{split}([R_{t'},i,j,f'_1,f'_2]);$$
$$P([t,i,j,-,-]) = \quad (32)$$
$$\sum_{t' \in \{t\} \cup \mathcal{A}, f \in \{-,i,j\}} P_{outer}([t,i,j,-,-],\ [t',f,f]) \cdot$$
$$P_{split}([R_{t'},i,j,f,f]).$$

Equations for P_{outer} will be derived in the next subsection.

In summary, terminating computation of prefix probabilities should be based on equations (31) and (32), which replace (1), along with equations (2) to (9) and all the equations for P_{split}.

4.3 Off-line Equations

In this section we derive equations for function P_{outer} introduced in §4.2 and deal with all remaining cases of equations that cause infinite recursion.

In some cases, function P can be computed independently of the actual input. For any $i < n$ we can consistently define the following quantities, where $t \in \mathcal{I} \cup \mathcal{A}$ and $\alpha \in V^{\bot}$ or $cdn(N) = \alpha\beta$ for some N and β:

$$H_t = P([t,i,i,f,f]);$$
$$H_\alpha = P([\alpha,i,i,f',f']),$$

where $f = i$ if $t \in \mathcal{A}$, $f = -$ otherwise, and $f' = i$ if $dft(\alpha)$, $f = -$ otherwise. Thus, H_t is the probability of all derived trees obtained from t, with no lexical node at their yields. Quantities H_t and H_α can be computed by means of a system of equations which can be directly obtained from equations (1) to (9). Similar quantities as above must be introduced for the case $i = n$. For instance, we can set $H'_t = P([t,n,n,f,f])$, f specified as above, which gives the probability of all derived trees obtained from t (with no restriction at their yields).

Function P_{outer} is also independent of the actual input. Let us focus here on the case $f_1, f_2 \notin \{i,j,-\}$ (this enforces $(f_1,f_2) = (f'_1,f'_2)$ below). For any $i,j,f_1,f_2 < n$, we can consistently define the following quantities.

$$L_{t,t'} = P_{outer}([t,i,j,f_1,f_2],\ [t',f'_1,f'_2]);$$
$$L_{\alpha,t'} = P_{outer}([\alpha,i,j,f_1,f_2],\ [t',f'_1,f'_2]).$$

In the case at hand, $L_{t,t'}$ is the probability of all derived trees obtained from t such that (i) no lexical node is found at their yields; and (ii) at some 'unfinished' node dominating the foot of t, the probability of the adjunction of t' has already been accounted for, but t' itself has not been adjoined.

It is straightforward to establish a system of equations for the computation of $L_{t,t'}$ and $L_{\alpha,t'}$, by rewriting equations (12) to (20) according to (29) and (30). For instance, combining (12) and (29) gives (using the above assumptions on f_1 and f_2):

$$L_{t,t'} = L_{R_t,t'} + \delta(t = t').$$

Also, if $\alpha \neq \epsilon$ and $dft(N)$, combining (14) and (30) gives (again, using previous assump-

tions on f_1 and f_2; note that the H_α's are known terms here):

$$L_{\alpha N,t'} = H_\alpha \cdot L_{N,t'}.$$

For any $i, f_1, f_2 < n$ and $j = n$, we also need to define:

$$L'_{t,t'} = P_{outer}([t, i, n, f_1, f_2],\ [t', f'_1, f'_2]);$$

$$L'_{\alpha,t'} = P_{outer}([\alpha, i, n, f_1, f_2],\ [t', f'_1, f'_2]).$$

Here $L'_{t,t'}$ is the probability of all derived trees obtained from t with a node dominating the foot node of t, that is an adjunction site for t' and is 'unfinished' in the same sense as above, and with lexical nodes only in the portion of the tree to the right of that node. When we drop our assumption on f_1 and f_2, we must (pre)compute in addition terms of the form $P_{outer}([t, i, j, i, i],\ [t', i, i])$ and $P_{outer}([t, i, j, i, i],\ [t', j, j])$ for $i < j < n$, $P_{outer}([t, i, n, f_1, n],\ [t', f'_1, f'_2])$ for $i < f_1 < n$, $P_{outer}([t, i, n, n, n],\ [t', f'_1, f'_2])$ for $i < n$, and similar. Again, these are independent of the choice of i, j and f_1. Full treatment is omitted due to length restrictions.

5 Complexity and concluding remarks

We have presented a method for the computation of the prefix probability when the underlying model is a tree adjoining grammar. Function P_{split} is the core of the method. Its equations can be directly translated into an effective algorithm, using standard functional memoization or other tabular techniques. It is easy to see that such an algorithm can be made to run in time $\mathcal{O}(n^6)$, where n is the length of the input prefix.

All the quantities introduced in §4.3 (H_t, $L_{t,t'}$, etc.) are independent of the input and should be computed off-line, using the system of equations that can be derived as indicated. For quantities H_t we have a non-linear system, since equations (2) to (6) contain quadratic terms. Solutions can then be approximated to any precision degree using standard iterative methods, as for instance those exploited in (Stolcke, 1995). Under the hypothesis that the grammar is consistent, that is $\Pr(L(G)) = 1$, we have that all quantities H'_t and H'_α evaluate to one. For quantities $L_{t,t'}$ and the like, §4.3 provides linear systems whose solutions can easily be obtained using standard methods. Note also that quantities $L_{\alpha,t'}$ are only used in the off-line computation of quantities $L_{t,t'}$, they do not need to be stored for the computation of prefix probabilities (cfr. (31) and (32)).

We can easily develop implementations of our method that can compute prefix probabilities incrementally. That is, after we have computed the prefix probability for a prefix $a_1 \cdots a_n$, on input a_{n+1} we can extend the calculation to prefix $a_1 \cdots a_n a_{n+1}$ without having to recompute all intermediate steps that do not depend on a_{n+1}. This step takes time $\mathcal{O}(n^5)$.

In this paper we have assumed that the parameters of the stochastic TAG have been previously estimated. In practice, smoothing of these parameter values plays an important role. Smoothing can be directly incorporated into a language model that uses prefix probabilities. For example, if smoothing of a stochastic TAG model after initial estimation was done by using combining word-based models with class-based models using linear interpolation, each corresponding prefix probability model obtained by application of our algorithm, can be combined similarly. The interpolation parameters are trained using held-out data used by the estimation algorithm.

References

C. Chelba et al. 1997. Structure and performance of a dependency language model. In *Proc. of Eurospeech 97*, volume 5, pages 2775–2778.

F. Jelinek and J. Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323.

A. K. Joshi and Y. Schabes. 1992. Tree-adjoining grammars and lexicalized grammars. In M. Nivat and A. Podelski, editors, *Tree automata and languages*, pages 409–431. Elsevier Science.

A. K. Joshi. 1988. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

B. Lang. 1988. Parsing incomplete sentences. In *Proc. of the 12th International Conference on Computational Linguistics*, volume 1, pages 365–371, Budapest.

O. Rambow and A. Joshi. 1995. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Leo Wanner, editor, *Current Issues in Meaning-Text Theory*. Pinter, London.

Y. Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proc. of COLING '92*, volume 2, pages 426–432, Nantes, France.

B. Srinivas. 1996. "Almost Parsing" technique for language modeling. In *Proc. ICSLP '96*, volume 3, pages 1173–1176, Philadelphia, PA, Oct 3-6.

A. Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

J. H. Wright and E. N. Wrigley. 1989. Probabilistic LR parsing for speech recognition. In *IWPT '89*, pages 105–114.