ISSN 0918-2802

Technical Report

Fast EM Learning of a Family of PCFGs

Sato, Taisuke (TIT, sato@mi.cs.titech.ac.jp)
Kameya, Yoshitaka (NS solutions)
Abe, Shigeru (TIT, abe@mi.cs.titech.ac.jp)
Shirai, Kiyoaki (JAIST, kshirai@jaist.ac.jp)

TR01-0006 May

Department of Computer Science
Tokyo Institute of Technology
Ôokayama 2-12-1 Meguro Tokyo 152-8552, Japan
http://www.cs.titech.ac.jp/

Abstract

The purpose of this report is to optimize the Inside-Outside algorithm and realize fast EM learning of various extensions of PCFGs. The point of our optimization is the introduction of a new data structure called *support graphs* hierarchically representing a set of parse trees and a new EM learning algorithm called the *graphical EM algorithm* that runs on them. Learning experiments with PCFGs using Japanese corpora of moderate size with contrasting characters indicate that in terms of updating time per iteration, the graphical EM algorithm can learn parameters of PCFGs orders of magnitude faster than the Inside-Outside algorithm. We also experimentally show that the graphical EM algorithm requires at most about twice as much time as a pure PCFG to learn parameters of extended PCFGs (a Pseudo PCSG and and a lexicalized PCFG).

1 Introduction

PCFGs (probabilistic context free grammars) have been recognized as a basic mechanism to express a probability distribution over syntactic objects described by CFGs (Manning and Schütze, 1999). Despite their popularity however, they have two major problems. One is their poor modeling capability as a statistical language model and the other is their slow speed of unsupervised parameter learning (Pereira and Schabes, 1992; Lari and Young, 1990). As solutions to the first problem, several elaborations of PCFGs were proposed including Pseudo PCSGs (probabilistic context sensitive grammars) (Charniak and Carroll, 1994) and lexicalizations (Charniak, 1997; Collins, 1997; Carroll and Rooth, 1998; Beil et al., 1999; Charniak, 2000).

The second problem, the computational cost of unsupervised parameter learning, on the other hand, has been paid relatively little attention since the proposal of the I-O (inside-outside) algorithm by Baker (Baker, 1979) as the EM algorithm for PCFGs.[1] Time complexity in one iteration of the I-O algorithm is known to be $O(N^3L^3)$ where N is the number of symbols in a grammar and L the length of an input sentence (Lari and Young, 1990). And it is not unusual for the I-O algorithm to take days to learn parameters from a large corpus.[2]

The purpose of this paper is to optimize the I-O algorithm and realize fast EM learning for various extensions of PCFGs. The optimization is done by introducing a new data structure called *support graph*s representing a set of parse trees and a new EM learning algorithm called the *graphical EM algorithm* (Kameya and Sato, 2000) that runs on them. They are explained in Section 3. Learning experiments with PCFGs in Section 4 using Japanese corpora of moderate size with contrasting characters indicate that the graphical EM algorithm can learn parameters of PCFGs orders of magnitude faster than the I-O algorithm, thus solving the second problem to a large extent.

We also experimentally show by learning experiments with the graphical EM algorithm in Section 5 that although extended PCFGs mentioned above require learning time longer than pure PCFGs, they do so only marginally (at most twice), which makes it computationally feasible to adopt even more linguistically sophisticated models in natural language processing, thereby indirectly contributing to the solution of the first problem.

[1] The EM algorithm is an iterative algorithm for maximum likelihood estimation. It finds parameters values that maximize the likelihood of observations by repeating the E(-expectation) step and the M(-aximization) step alternately (Dempster et al., 1977).

[2] It is reported in (Beil et al., 1999) that to train a PCFG with 5,508 rules from a corpus of 450,526 German subordinate clauses whose average ambiguity is 9,202 trees/clause, it takes 2.5 hours for 4 CPUs to complete *one iteration* of the I-O algorithm.

2 Previous EM algorithms for PCFGs

In this section, we review previous EM algorithms for PCFGs. The first EM algorithm for PCFGs was the I-O algorithm proposed by Baker (Baker, 1979). The grammar it uses must be in Chomsky normal form and contains all possible rules expressed by numbers like $i \rightarrow j, k$ ($1 \leq i, j, k \leq N$, for N non-terminals). Let $w_1, \ldots, w_n$ be an input sentence of length n and S (= 1) the starting symbol.[3] In each iteration, the algorithm first computes in a bottom up manner inside probabilities $e(s, t, i)$, i.e. the probability $P(i \overset{*}{\Rightarrow} w_s, \ldots, w_t)$ while trying every possible combination of rules. It next computes outside probabilities $f(s, t, i)$, i.e. the probability $P(S \overset{*}{\Rightarrow} w_1, \ldots, w_{s-1}\ i\ w_{t+1}, \ldots, w_n)$ in a top-down manner using previously computed inside probabilities. After computing inside and outside probabilities, parameters are updated by using them. Update iterates until the likelihood of the input sentence saturates.

When the grammar does not allow arbitrary rules, it is apparent that only combinations of rules appearing in parse trees are relevant to the computation of inside and outside probabilities. Nonetheless the I-O algorithm repeats in every iteration all possible combinations of the rules that derive the input sentence. In other words, the I-O algorithm lacks parsing.

By contrast the EM algorithm proposed by Fujisaki et al. used probability $P(D_m)$ of the m-th derivation D_m of an input sentence (Fujisaki et al., 1989).[4] So irrelevant rules are excluded by parsing. Unfortunately, while inside probabilities are recursively computed like the I-O algorithm, outside probabilities are naively computed as a sum $\sum_m P(D_m)\#(i \rightarrow j, k \in D_m)$ in each iteration, where $\#(i \rightarrow j, k \in D_m)$ is the number of occurrences of the rule $i \rightarrow j, k$ in the m-th derivation. Since m can be exponential in the length of an input sentence, it is difficult to perform large scale learning by their approach.

Later Stolcke used an Earley chart generated by an Earley parser to compute inside and outside probabilities (Stolcke, 1995). The chart is comprised of items augmented with two types of probabilities and inside and outside probabilities are computed from them. Accordingly, unlike (Fujisaki et al., 1989), his approach incorporates both parsing and the sharing of processes of sentence derivations. The problem is that the Earley chart may possibly contain items not contributing to the final outcome, a parse tree, but probabilities are still computed for them as well.

To solve these problems, *we completely separate EM learning from parsing.* We first introduce a new data structure, *support graph*s for EM learning. A *support graph* (explained in next section) is constructed from a WFST (well-formed substring table) such as a triangular table used by a CYK parser and an Earley chart used by an Earley parser. It is an ordered set of applied rules compactly representing all parse trees by structure sharing. We also introduce a new EM algorithm called the *graphical EM algorithm* that runs on the graph to compute inside and outside probabilities.

Computationally, construction of a support graph is an extra burden, but it pays off just like compilation of programs pays off as time per iteration in the EM algorithm can improve by orders of magnitude as we will see in Section 4.

3 Support graphs and the graphical EM algorithm

Suppose we are given a PCFG $\mathcal{G}$[5] and T sampled sentences $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(T)}$ whose distribution is supposedly modeled by $\mathcal{G}$. In our approach to EM learning, for each sentence $\mathbf{w}^{(\ell)}$ ($1 \leq \ell \leq T$), we first parse $\mathbf{w}^{(\ell)}$, construct a WFST and extract from the WFST a *support graph* $\Delta_\ell = \langle \tau_1, \ldots, \tau_{|\Delta_\ell|} \rangle$ for $\mathbf{w}^{(\ell)}$. We then run the *graphical EM algorithm* on

[3] We assume only one input sentence for simplicity.

[4] $P(D_m)$ is computed as the products of probabilities associated with production rules involved in the derivation.

[5] We assume that there is neither the empty production rule nor non-terminal A such that $A \overset{+}{\Rightarrow} A$ in the grammar.

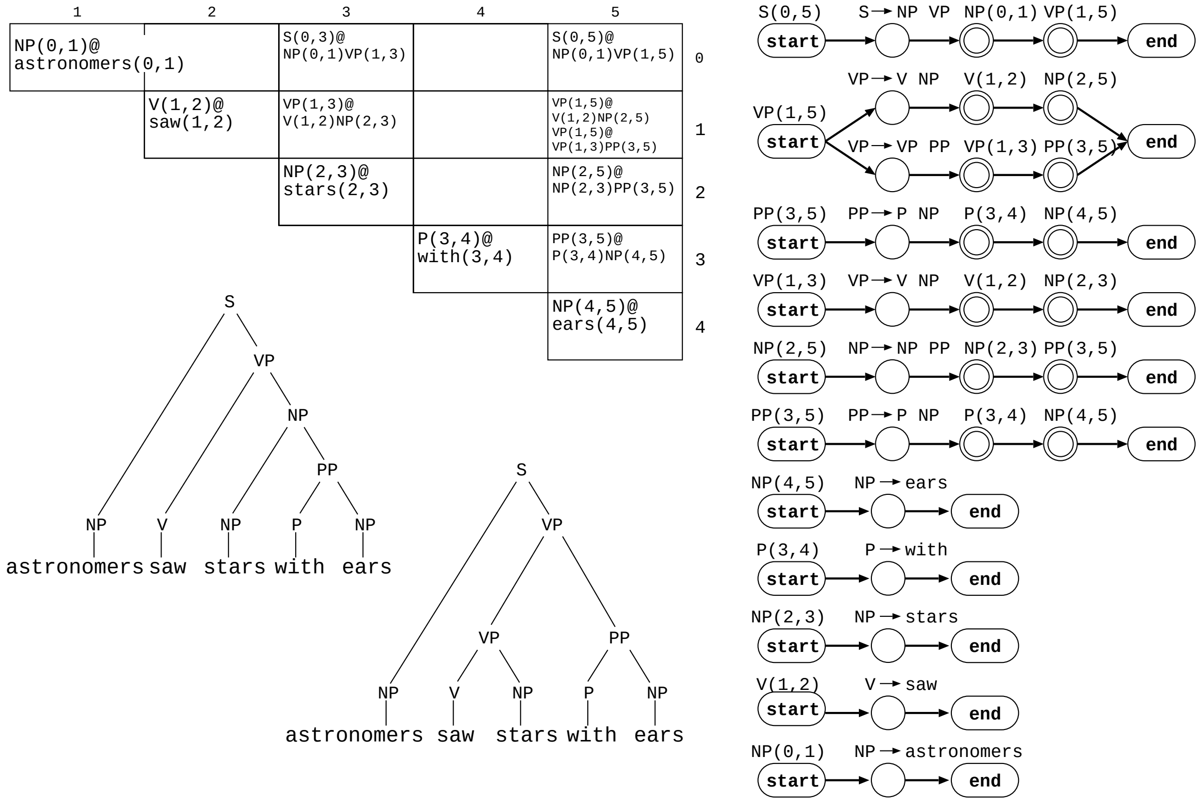

Figure 1: Parsing "astronomers saw stars with ears" and its support graph

$\{\Delta_\ell \mid 1 \leq \ell \leq T\}$ to statistically learn parameters associated with $\mathcal{G}$.

In the following, we use R for the rule set of $\mathcal{G}$, $V_{\rm n}$ for its non-terminal set and $\theta(A \to \zeta)$ for the parameter associated with a rule $A \to \zeta \in R$. n_ℓ stands for $|\mathbf{w}^{(\ell)}|$, i.e. the length of $\mathbf{w}^{(\ell)}$ and $\mathbf{w}^{(\ell)}[i,j]$ denotes a substring of $\mathbf{w}^{(\ell)}$ starting at the $i+1$-th word and ending at the j-th word.

A *support graph* Δ_ℓ for $\mathbf{w}^{(\ell)}$ is a *linearly ordered* set $\langle \tau_1, \ldots, \tau_M \rangle$ of disconnected subgraphs τ_k $(1 \leq k \leq M)$ which can be read off from the WFST for $\mathbf{w}^{(\ell)}$. Each τ_k is labeled like $A(i,j)$ and comprised of linear graphs ν of the form

$$\texttt{start-}(A \to B_1 \cdots B_M)\text{-} \\ B_1(i,h_1)\text{-}\cdots\text{-}B_M(h_M,j)\texttt{-end}$$

which records that A is expanded by rule $A \to B_1 \cdots B_M$ to span $\mathbf{w}^{(\ell)}[i,j]$ in some parse of $\mathbf{w}^{(\ell)}$. Here `start` is a fork node and `end` is a join node and $B_k(h_{k-1},h_k)$ $(1 \leq k \leq M, h_0 = i, h_M = j)$ is the label of the corresponding subgraph which is recursively constructed. Define $\mathcal{N}(\nu)$ and $\tilde{\psi}_\ell(\cdot)$ by

$$\begin{aligned} \mathcal{N}(\nu) &\stackrel{\rm def}{=} \{A \to B_1 \cdots B_M, \\ &\qquad B_1(i,h_1), \ldots, B_M(h_M,j)\} \\ \tilde{\psi}_\ell(\tau_k) &\stackrel{\rm def}{=} \{\mathcal{N}(\nu) \mid \nu \text{ is a linear graph in } \tau_k\} \end{aligned}$$

Suppose for example A is expanded by $A \to BC$ and $A \to D$ respectively in some parse of $\mathbf{w}^{(\ell)}$ and happens to span the same substring $\mathbf{w}^{(\ell)}[i,j]$. Then the disconnected subgraph τ labeled $A(i,j)$ contains two linear graphs, $\texttt{start-}(A \to BC)\text{-}B(i,k)\text{-}C(k,j)\texttt{-end}$ and $\texttt{start-}(A \to D)\text{-}D(i,j)\texttt{-end}$, and we have $\tilde{\psi}(\tau)) = \{\{A \to BC, B(i,k), C(k,j)\}, \{A \to D, D(i,j)\}\}$.

The total ordering $\tau_1 < \cdots < \tau_M$ over the disconnected subgraphs is introduced as follows. First define a partial ordering $\prec$ over the τ_k's as $A(i,j) \prec B(i',j')$ iff B is expanded in a partial derivation from A of $\mathbf{w}^{(\ell)}[i,j]$ and spans a substring $\mathbf{w}^{(\ell)}[i',j']$ of $\mathbf{w}^{(\ell)}[i,j]$ $(i \leq i' \leq j' \leq j)$ in some successful parse of $\mathbf{w}^{(\ell)}$. A support graph Δ_ℓ for $\mathbf{w}^{(\ell)}$ is then ob-

```
procedure Extract-CYK() begin
  for $\ell := 1$ to $T$ do begin
    Initialize all $\tilde{\psi}_\ell(\cdot)$ to $\emptyset$ and
      all $Visited[\cdot]$ to NO;
    $ClearStack(U)$;
    $Visit\text{-}CYK(\ell, S, 0, n_\ell)$;
    for $k := 1$ to $|U|$ do $\tau_k := PopStack(U)$;
    $\Delta_\ell := \langle \tau_1, \tau_2, \ldots, \tau_{|U|} \rangle$
  end
end.
```

```
procedure Visit-CYK($\ell, A, i, j$) begin
  Put $\tau = A(i,j)$; $Visited[\tau]$ := YES;
  if $j = i+1$ and $A(i,j)@\mathbf{w}_j^{(\ell)} \in T_{i,j}^{(\ell)}$
  then Add a set $\{A \to \mathbf{w}_j^{(\ell)}\}$ to $\tilde{\psi}_\ell(\tau)$
  else
    foreach
      $A(i,j)@B(i,k)C(k,j) \in T_{i,j}^{(\ell)}$
    do begin
      Add to $\tilde{\psi}_\ell(\tau)$ a set
        $\{A \to BC,\ B(i,k),\ C(k,j)\}$;
      if $Visited[B(i,k)]$ = NO
      then $Visit\text{-}CYK(\ell, B, i, k)$;
      if $Visited[C(k,j)]$ = NO
      then $Visit\text{-}CYK(\ell, C, k, j)$
    end;
  $PushStack(\tau, U)$
end.
```

Figure 2: A routine for support graph extraction from a triangular table generated by a CYK parser

tained by topologically sorting the τ_k's so that $\tau_k < \tau_{k'}$ implies $\tau_k \prec \tau_{k'}$. As a result, the label of τ_1 always takes the form $S(0, n_\ell)$ where S is the starting symbol of the grammar.

In what follows, for descriptive convenience, we sometimes treat $\Delta_\ell = \langle \tau_1, \ldots, \tau_M \rangle$ as an ordered labels of the τ_k's together with a function $\tilde{\psi}_\ell(\cdot)$ defined over them.

Fig. 1 illustrates parses of sentence "astronomers saw stars with ears" (taken from (Manning and Schütze, 1999)) by a CYK parser and the resulting triangular table, i.e. a WFST for CYK parsing. "@" in the table is a symbol separating a parent label from child labels. The right graph in Fig. 1 is a support graph extracted from the triangular table by the procedure *Extract-CYK*() in Fig. 2.[6] It is (we only show labels)

$$\Delta = < \mathrm{S(0,5),\ VP(1,5),\ PP(3,5),\ VP(1,3),\ NP(2,5),\ PP(3,5),\ NP(4,5),\ P(3,4),\ NP(2,3),\ V(1,2),\ NP(0,1)} > .$$

Note that in the support graph, two disconnected graphs VP(1,5) and NP(2,5) have PP(3,5) as a child node and precede it in Δ.

After constructing support graphs $\{\Delta_\ell \mid 1 \leq \ell \leq T\}$, we run the graphical EM algorithm on them. It is driven by one main routine *gEM*() (Fig. 3) for updating parameters[7] and two subroutines, *Get-Inside-Probs*() (Fig. 4) for computing inside probabilities and *Get-Expectations*() (Fig. 5) for computing outside probabilities and the expected number of occurrences $\eta[A \leftarrow \zeta]$ of a production rule $A \leftarrow \zeta \in R$ given the sentence.

Let $A(i,j)$ be the label of a disconnected subgraph τ_k in Δ_ℓ. In each iteration of *gEM*(), *Get-Inside-Probs*() is called and recursively computes in a bottom-up manner the inside probability $P(A \overset{*}{\Rightarrow} \mathbf{w}^{(\ell)}[i,j])$ and stores it in array $\mathcal{P}[\ell, A(i,j)]$.[8] *Get-Inside-Probs*() also uses array $\mathcal{R}[\ell, A(i,j), E]$ to store probability $P(E)$ where $E = \{e_0, \ldots, e_M\} \in \tilde{\psi}(\tau_k)$. $P(E)$ is computed as a product $P(e_1) \cdots P(e_M)$ where $P(e) = \theta(A \leftarrow \zeta)$ if node e is a rule $A \leftarrow \zeta \in R$, or $P(e) = \mathcal{P}[\ell, B(i',j')]$ if e is a label $B(i',j')$. We see $\mathcal{P}[\ell, A(i,j)] = \sum_{E \in \tilde{\psi}(\tau_k)} P(E)$.

After computing all $\mathcal{P}[\ell, A(i,j)]$s, *Get-Expectations*() is called to compute each $\mathcal{Q}[\ell, A(i,j)]$, the outside probability

[6] Here U in *Extract-CYK*() is a stack holding disconnected subgraphs (or equivalently their labels) constituting a support graph Δ_ℓ. $T_{i,j}^{(\ell)}$ in *Visit-CYK*() is an element stored at (i,j) in the triangular table $T^{(\ell)}(\cdot,\cdot)$ for $\mathbf{w}^{(\ell)}$. $\mathbf{w}_j^{(\ell)}$ is the j-th word of $\mathbf{w}^{(\ell)}$. Since there exist only $O(N^3 L^3)$ elements possibly filling $T^{(\ell)}(\cdot,\cdot)$ where $L = |\mathbf{w}^{(\ell)}|$ and $V = |V_{\mathrm{n}}|$, and filled elements are all different, time complexity of *Extract-CYK*() is $O(N^3 L^3)$. Another support graph extraction routine for an Earley parser is described in (Kameya et al., 2001).

[7] $P(\cdot \mid \theta)$ in *gEM*() stands for a distribution under the current parameter values θ.

[8] For intuitiveness and readability, we use $A(i,j)$ as a substitute for τ_k.

```
procedure gEM() begin
    Initialize all parameters $\theta(A \rightarrow \zeta)$;
    such that $P(\mathbf{w}^{(\ell)}|\theta) > 0$ for all $\ell = 1, \ldots, T$;
    Get-Inside-Probs();
    $\lambda^{(0)} := \sum_{\ell=1}^{T} \log \mathcal{P}[\ell, S(0, n_\ell)]$;
    repeat
        Get-Expectations();
        foreach $(A \rightarrow \zeta) \in R$ do
            $\theta(A \rightarrow \zeta) := \eta[A \rightarrow \zeta] / \sum_{\zeta'} \eta[A \rightarrow \zeta']$;
        $m$ += 1;
        Get-Inside-Probs();
        $\lambda^{(m)} := \sum_{\ell=1}^{T} \log \mathcal{P}[\ell, S(0, n_\ell)]$
    until $\lambda^{(m)} - \lambda^{(m-1)}$ is sufficiently small
end.
```

Figure 3: Main routine *gEM*

$P(S \stackrel{*}{\Rightarrow} \mathbf{w}^{(\ell)}[0,i]\ A\ \mathbf{w}^{(\ell)}[j,n_\ell])$ in a top-down manner from $\tau_1 = \mathtt{S(0,}n_\ell\mathtt{)}$ where $n_\ell = |\mathbf{w}^{(\ell)}|$ (the length of $\mathbf{w}^{(\ell)}$), while incrementing the expected counts $\eta[A \leftarrow \zeta]$ of $A \leftarrow \zeta$ in a parse of $\mathbf{w}^{(\ell)}$.

gEM() iterates to update parameters until an increase in the log likelihood of $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(T)}$ is less than a threshold, say 10^{-6}.

It is proved in (Kameya, 2000; Kameya et al., 2001) that the graphical EM algorithm computes the same values (inside and outside probabilities, update values of parameters) as the I-O algorithm, hence the difference only lies in computational efficiency. Since time complexity of parsing and that of one iteration for updating all parameters in one iteration by the graphical EM algorithm are both $O(N^3L^3)$ where N is the number of terminal symbols in a grammar and L the length of an input sentence,[9] we can say that the graphical EM algorithm is as efficient as the I-O algorithm. Likewise we can analyze time complexity by the graphical EM algorithm of various extensions of PCFGs. For example, parsing and parameter updating in one iteration for Pseudo PCSGs (Charniak and Carroll, 1994) take $O(N^4L^3)$. In what follows, we focus on the actual behavior of the graphical EM algorithm w.r.t. real corpora.

```
procedure Get-Inside-Probs() begin
    for $\ell$ := 1 to $T$ do begin
        Put $\Delta_\ell = \langle \tau_1, \tau_2, \ldots, \tau_{|\Delta_\ell|} \rangle$;
        for $k := |\Delta_\ell|$ downto 1 do begin
            foreach $E \in \tilde{\psi}(\tau_k)$ do begin
                $\mathcal{R}[\ell, \tau_k, E] := 1$;
                foreach $e \in E$ do
                    if $e = (A \rightarrow \zeta)$
                    then $\mathcal{R}[\ell, \tau_k, E]$ *= $\theta(A \rightarrow \zeta)$
                    else $\mathcal{R}[\ell, \tau_k, E]$ *= $\mathcal{P}[\ell, e]$;
            end;
            $\mathcal{P}[\ell, \tau_k] := \sum_{E \in \tilde{\psi}(\tau_k)} \mathcal{R}[\ell, \tau_k, E]$
        end    /* for $k$ */
    end    /* for $\ell$ */
end.
```

Figure 4: Subroutine *Get-Inside-Probs*

4 Learning experiments with pure PCFGs using two corpora

In this section, we experimentally compare the graphical EM algorithm with the I-O algorithm in terms of time per iteration (= time for updating parameters) by letting them learn PCFG parameters from a corpus. Support graphs were generated by using a Tomita (Generalized LR) parser.[10] All measurements were made on a 296MHz Sun UltraSPARC-II with Solaris 2.6.

We used two corpora. One was a POS (part of speech)-tagged corpus converted from ATR corpus and the other is EDR corpus. ATR corpus is a Japanese-English corpus (we used only the Japanese part) developed by ATR (Uratani et al., 1994). It contains 10,995 conversational sentences, whose minimum length, average length and maximum length are respectively 2, 9.97 and 49. As a skeleton of PCFG, we employed a context free

[9] It is easy to see that time complexity of the graphical EM algorithm in one iteration is proportional to the size of a support graph which is $O(N^3L^3)$ (Kameya, 2000; Kameya et al., 2001).

[10] More specifically, we combined MSLR (Morphological and Syntactic LR) parser with our routine for the graphical EM algorithm. MSLR parser is a Tomita parser, developed by Tanaka-Tokunaga Laboratory in Tokyo Institute of Technology (Tanaka et al., 1997). Although MSLR parser provides several useful functions (e.g. morphological analysis), we used only the basic one (i.e. parsing of the tagged sentence).

```
procedure $Get\text{-}Expectations()$ begin
  foreach $(A \rightarrow \zeta) \in R$ do $\eta[A \rightarrow \zeta] := 0$;
  for $\ell := 1$ to $T$ do begin
    Put $\Delta_\ell = \langle \tau_1, \tau_2, \ldots, \tau_{|\Delta_\ell|} \rangle$;
    $\mathcal{Q}[\ell, \tau_1] := 1$;
    for $k := 2$ to $|\Delta_\ell|$ do $\mathcal{Q}[\ell, \tau_k] := 0$;
    for $k := 1$ to $|\Delta_\ell|$ do
      foreach $E \in \tilde{\psi}(\tau_k)$ do
        foreach $e \in E$ do
          if $e = (A \rightarrow \zeta)$ then $\eta[A \rightarrow \zeta] \mathrel{+}= \mathcal{Q}[\ell, \tau_k] \cdot \mathcal{R}[\ell, \tau_k, E] / \mathcal{P}[\ell, S(0, n_\ell)]$
          else if $\mathcal{P}[\ell, e] > 0$ then $\mathcal{Q}[\ell, e] \mathrel{+}= \mathcal{Q}[\ell, \tau_k] \cdot \mathcal{R}[\ell, \tau_k, E] / \mathcal{P}[\ell, e]$
  end    /* for $\ell$ */
end.
```

Figure 5: Subroutine *Get-Expectations*

grammar G_{atr} comprising 860 rules (172 non-terminals and 441 terminals) manually developed for ATR corpus (Tanaka et al., 1997) which generates 958 parses/sentence.

Because the I-O algorithm only accepts CFGs in Chomsky normal form, we converted G_{atr} into Chomsky normal form G^*_{atr}. G^*_{atr} contains 2,105 rules (196 non-terminals and 441 terminals). We then divided the corpus into subgroups of similar length like $(L = 1,2), (L = 3,4), \ldots, (L = 25,26)$, each containing randomly chosen 100 sentences. After these preparations, we compare the graphical EM algorithm applied to G_{atr} and G^*_{atr} against the I-O algorithm applied to G^*_{atr} in terms of updating time per iteration.[11] The results are shown in Fig. 6.

In the left graph, the I-O algorithm draws a cubic curve labeled "I-O".[12] The curves drawn by the graphical EM algorithm coincide with the x axis and are not plotted. The middle graph magnifies the left graph. The curve labeled "gEM (original)" is drawn by the graphical EM algorithm applied to the original grammar G_{atr} whereas the one labeled "gEM (Chomsky CF)" used G^*_{atr}. Seeing "gEM (original)" at length 10, the average sentence length, we know that the graphical EM algorithm runs about 850 times faster than the I-O algorithm. The right graph shows (almost) linear dependency of updating time of the graphical EM algorithm on the sentence length.

This rather striking difference needs explanation, but the reason looks simple. When the I-O algorithm is applied to a specific grammar such as G^*_{atr}, it tries all possible combinations of CFG rules *in every iteration*. In other words, it parses a sentence anew in each iteration. The graphical EM algorithm on the other hand uses support graphs representing already parsed trees.

It is conceivable however that such a big difference in updating time is ascribed to the fact that ATR corpus only contains short sentences and G_{atr} is not very ambiguous so that the resulting WFST is sparse. If the WFST is not sparse, and hence the support graph is larger, we might not be able to expect such a big difference.

We therefore conducted the same experiment with another corpus, EDR Japanese corpus which contains much longer sentences using a highly ambiguous grammar. EDR Japanese corpus (Japan Electronic Dictionary Research Institute, 1995) contains 220,000 Japanese news article sentences. It is however under the process of re-annotation, and only part of it (randomly sampled 9,900 sentences) is available as a labeled corpus at the moment. Compared with ATR corpus, sentences are much longer (the average length

[11] The stopping threshold was 10^{-6}.

[12] An x-axis is the length L of an input sentence and a y-axis is time taken by the EM algorithm to update all parameters in the grammar in one iteration.

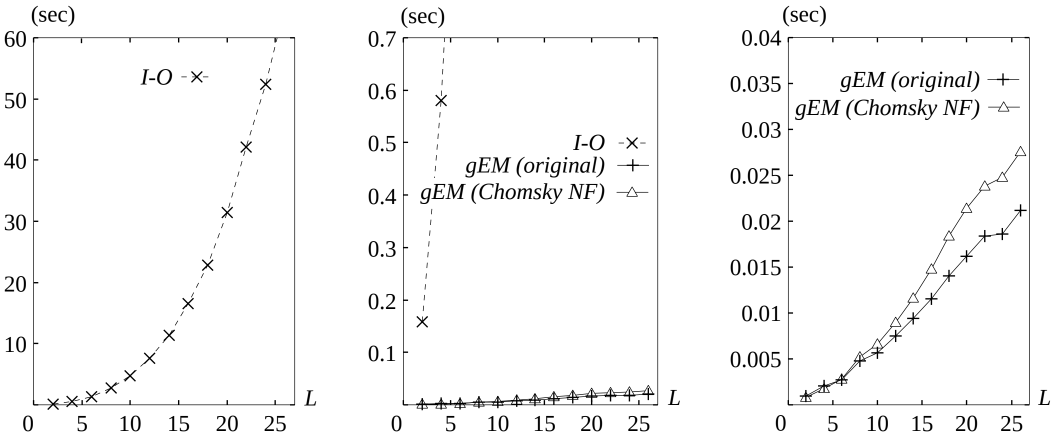

Figure 6: Time per iteration : I-O vs. gEM (ATR)

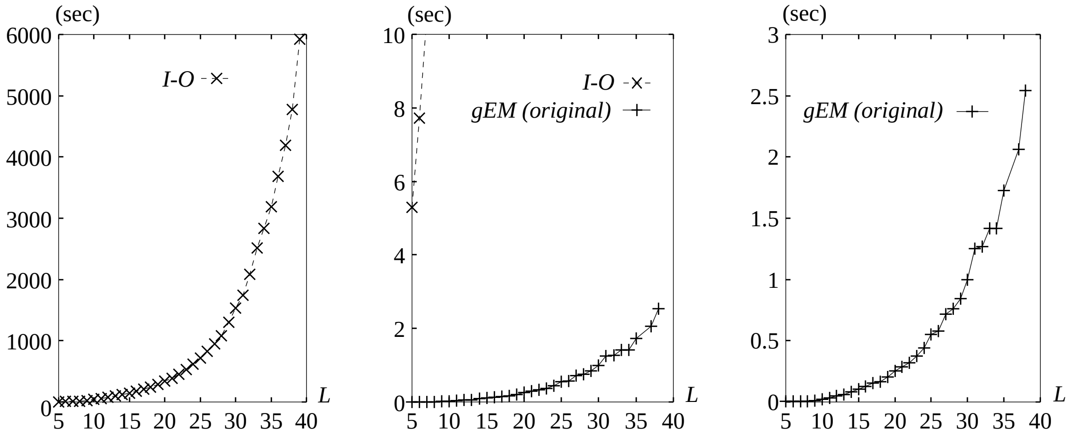

Figure 7: Time per iteration : I-O vs. gEM (EDR)

of 9,900 sentences is 20, the minimum length 5, the maximum length 63) and a CFG grammar $G_{\rm edr}$ (2,687 rules, converted to Chomsky normal form grammar $G^{*}_{\rm edr}$ containing 12,798 rules) developed for it is very ambiguous, having 3.0×10^{8} parses/sentence at length 20 and 6.7×10^{19} at length 38. The graphs in Fig. 7 show the results of the same experiment with EDR corpus as with ATR corpus, i.e. the graphical EM algorithm applied $G_{\rm edr}$ vs. the I-O algorithm applied to $G^{*}_{\rm edr}$.

We found that at average sentence length 20, the former runs 1,300 times faster the latter per iteration. Thus the speed ratio even widens compared to the case of ATR corpus. This can be explained by the mixed effects of time complexity $O(L^{3})$ for the I-O algorithm and a slow increase in the size of support graphs for the graphical EM algorithm.

One might argue that the comparison we made is not fair as what we are comparing is time per iteration of the EM algorithms, but proper comparison must be based on the total learning time which is calculated as

$$\begin{aligned}&(\mathrm{A} : \text{time for constructing support graphs})\\ &\quad + (\mathrm{B} : \text{time per iteration})\\ &\qquad \times (\mathrm{C} : \text{the number of iterations})\end{aligned}$$

for the graphical EM algorithm whereas it is merely B×C for the I-O algorithm. To answer this question, we estimated total learning time for both of the EM algorithms using the entire ATR corpus as input.

B for the I-O algorithm applied to $G^{*}_{\rm atr}$ was 1,215 seconds (average of 20 iterations) whereas for the graphical EM algorithm applied to $G_{\rm atr}$, A was 279 seconds and B was

0.661 second respectively.[13] Assuming C = 100 iterations,[14] the ratio of total learning time using the entire corpus is calculated as $(1215 \times 100)/(279 + 0.661 \times 100) = 352$, and hence the graphical EM algorithm will still be orders of magnitude faster than the I-O algorithm in terms of total learning time.[15]

5 Learning experiments with extended PCFGs

Finally we turn to extensions of PCFGs incorporating context sensitiveness in rule application. We consider two extensions. One is Pseudo PCSGs (pseudo probabilistic context sensitive grammar) (Charniak and Carroll, 1994) in which the probability of selecting a rule for expanding a non-terminal depends on its parent category. The other is lexicalized PCFGs (Charniak, 1997; Collins, 1997; Carroll and Rooth, 1998; Beil et al., 1999; Charniak, 2000) but in the most basic form such that the probability of rule application is conditioned on a "head" associated with the non-terminal to be expanded.

We conducted learning experiments with these extensions by the graphical EM algorithm just like the experiments in Section 4, using ATR corpus and G_{atr}. We, however, had to modify the parser to generate appropriate support graphs for each grammar model, but the modification was straightforward; just make every thing conditioned. For example, in the case of a Pseudo PCSG, the rule application of $A \rightarrow \zeta$ is conditioned on the parent category A' of A so that the label of a disconnected graph takes the form $A(i, j \mid A')$ and so on.

As Fig. 8 shows, despite higher time complexity, extended models draw similar curves to the PCFG model and they require at most twice as much time as the PCFG model.

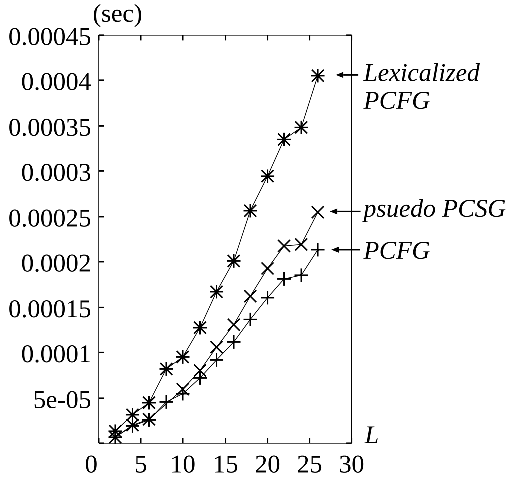

Figure 8: Time per iteration w.r.t. various extensions of PCFGs

For the sake of completeness, we summarize the performance measures in the Table 1 which is obtained by 11 fold cross-validation of various PCFGs using ATR corpus, G_{atr} and the Viterbi-parser combined with learned parameters.[16] Clearly these extended models perform better than the pure PCFG, and it appears that provided the grammar is not very ambiguous, we may expect that performance using parameters obtained by unsupervised learning (EM) comes close to the performance using those obtained by supervised learning (counting).

6 Conclusion

We have proposed a new data structure, support graphs, and a new EM algorithm, the graphical EM algorithm for PCFGs (and their extensions) that runs on support graphs. We have conducted learning experiments using two corpora, ATR corpus and EDR corpus which are contrasting in sentence length and

[13] An attempt to measure B for the EDR corpus available (9,900 sentences) was abandoned due to memory overflow.

[14] The plausible number of iterations until convergence by the graphical EM algorithm using the entire ATR corpus is 260 (this is the average of 3 trials), which would give more favorable ratio to the graphical EM algorithm.

[15] We also conducted a learning experiment to measure time per iteration for the entire ATR corpus using an implementation by Mark Johnson of the I-O algorithm down-loadable from `http://www.cog.brown.edu/%7Emj/`. This implementation turned out to be twice as fast as our naive implementation but gave B=630 seconds which is orders of magnitude slower than the graphical EM algorithm.

[16] PPCSG stands for Pseudo PCSG and LLPCFG for lexicalized PCFG. "Labeled Tree" means exact match and 0-CB is zero cross brackets. The base line is the performance in the case of randomly choosing a sentence and its labeled tree.

Labeled Tree	PCFG	PPCSG	LPCFG
counting	81.09	84.90	83.97
EM	71.88	76.03	78.68
Baseline	64.37		

0-CB	PCFG	PPCSG	LPCFG
counting	87.89	92.08	92.39
EM	86.83	89.24	89.75

Table 1: Evaluation of extended PCFGs

the ambiguity of the CFG grammars developed for them. In both cases however, the graphical EM algorithm outperformed the I-O algorithm by orders of magnitude in terms of updating time per iteration (and also in terms of total learning time with ATR corpus). We have also observed, experimentally using ATR corpus, that learning time by the graphical EM algorithm applied to extended PCFGs (a Pseudo PCSG and a lexicalized PCFG) does not deteriorate much compared to a pure PCFG.

References

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Proceedings of Spring Conference of the Acoustical Society of America*, pages 547–550.

F. Beil, G. Carroll, D. Prescher, S. Riezler, and M. Rooth. 1999. Inside-outside estimation of a lexicalized pcfg for german. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages –.

G. Carroll and M. Rooth. 1998. Valence induction with a head-lexicalized pcfg. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing (EMNLP 3)*.

E. Charniak and G. Carroll. 1994. Context-sensitive statistics for improved grammatical language models. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, pages 728–733.

E. Charniak. 1997. Statistical parsing with a conttext-free grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97)*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of 1st Meeting of the North American Cahpter of the Association for Computational Lingusitics.*

M. Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*, pages 16–23.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Royal Statistical Society*, B39(1):1–38.

T. Fujisaki, F. Jelinek, J. Cocke, E. Black, and T. Nishino. 1989. A probabilisitic parsing method for sentence disambiguation. In *Proceedings of International Parsing Workshop '89*, pages 85–94.

Ltd. Japan Electronic Dictionary Research Institute. 1995. Edr electronic dictionary technical guide (2nd edition). Technical Report EDR TR–045.

Y. Kameya and T. Sato. 2000. Efficient EM learning for parameterized logic programs. In *Proceedings of the 1st Conference on Computational Logic (CL2000)*, volume 1861 of *Lecture Notes in Artificial Intelligence*, pages 269–294. Springer.

Y. Kameya, T. Mori, and T. Sato. 2001. Fast em learning based on wfst of pcfgs and their extensions (in j). *Journal of Natural Language Processing*, Vol.8(No.1):49–84.

Y. Kameya. 2000. *Learning and Representation of Symbolic-Statistical Knowledge (in Japanese)*. Ph. D. dissertation, Tokyo Institute of Technology.

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

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing.* The MIT Press.

F. C. N. Pereira and Y. Schabes. 1992. Inside-Outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL'92)*, pages 128–135.

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

H. Tanaka, T. Takezawa, and J. Etoh. 1997. Japanese grammar for speech recognition considering the MSLR method. In *Proc. of the meeting of SIG-SLP (Spoken Language Processing)*, 97-SLP-15-25, pages 145–150. Information Processing Society of Japan. In Japanese.

N. Uratani, T. Takezawa, H. Matsuo, and C. Morita. 1994. ATR integrated speech and language database. Technical Report TR-IT-0056, ATR Interpreting Telecommunications Research Laboratories. In Japanese.