確率文脈自由文法及びその拡張文法の高速EM学習法

森 高志　　亀谷 由隆　　佐藤 泰介
〒 152-8552 東京工業大学 大学院情報理工学研究科 計算工学専攻
東京都目黒区大岡山 2-12-1

tmori@mi.cs.titech.ac.jp　　kame@mi.cs.titech.ac.jp　　sato@mi.cs.titech.ac.jp

概要
現在，統計的な言語モデルの一クラスとして，確率文脈自由文法 (PCFG) が知られている．PCFG の EM 学習法として Inside-Outside (I-O) アルゴリズムがあるが，計算速度に問題があることが知られている．本報告では，事前に CFG の骨格を与えることで，効率的に PCFG の確率パラメータの EM 学習を行う手法を提案する．提案手法では，WFST (well-formed substring table) の部分構文木を支持グラフと呼ばれる構造に変換し，その上でグラフィカル EM アルゴリズムを走らせ学習パラメータを得る．支持グラフは WFST における部分木共有を保存し，また入力文に対して構文木を構成する要素だけを持つため，提案手法は高速な学習が期待できる．また，本手法を PCFG に文脈を加えたモデルに拡張することも可能である．我々は ATR コーパスと人手で作られた日本語文法を用いて実験を行なった．学習速度の比較実験では，I-O アルゴリズムと比べ，訓練時間の大幅な短縮が確認された．

キーワード: 確率文脈自由文法，EM アルゴリズム，Inside-Outside アルゴリズム，WFST，疑似確率文脈依存文法，規則 bigram モデル

Efficient EM learning of probabilistic CFGs and their extentions

Takashi Mori　　Yoshitaka Kameya　　Taisuke Sato
Department of Computer Science, Graduate School of Information Science and Engeneering, Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo, 152-8552, Japan

tmori@mi.cs.titech.ac.jp　　kame@mi.cs.titech.ac.jp　　sato@mi.cs.titech.ac.jp

Abstract
Probabilistic context-free grammars (PCFGs) are widely-known as a class of statistical language models. It is also known that the Inside-Outside (I-O) algorithm (an EM algorithm tailored for PCFGs) requires much computational cost. In this report, we present a new framework for efficient EM learning for PCFGs, assuming that CFG skeleton is given in advance. In this framework, we first generate a *support graph* with partial parse trees in the WFST (well-formed substring table), and then run the *graphical EM algorithm* on the graph. Taking each advantage of Fujisaki et al.'s algorithm and the I-O algorithm, high-speed learning is expected. This framework is also applicable for various extensions of PCFGs which include context dependencies. We have conducted an experiment with the ATR corpus and a hand-crafted Japanese grammar, and the results show that, in comparison with I-O algorithm, our new framework has achieved drastic improvement in efficiency.

keyword: probabilistic context-free grammars, EM algorithm, Inside-Outside algorithm, WFST, pseudo PCSGs, rule bigram model

1 はじめに

現在，統計的な言語モデルの一クラスとして確率文脈自由文法 (以下 PCFG) が広く知られている．PCFG の確率パラメータの獲得には，構造付きコーパスに出現する文法規則を数えあげる方法 (以下，相対頻度法) が広く用いられるが，構造付きコーパスを作成する人的コストが大きいという問題がある．一方，形態素解析済みの括弧なしコーパスを用いた PCFG の EM (Expectation-Maximization) 学習法として Inside-Outside (以下 I-O) アルゴリズム [1] が知られているが，文長 (文の語数) L に対し $O(L^3)$ の計算量を要するため [8]，大規模な文法やコーパスからの学習は困難であった．

本報告では，事前に CFG の骨格が与えられているとき，効率的に PCFG の確率パラメータを EM 学習する手法を提案する．我々は学習過程を構文解析と EM 学習の 2 つに分離する．まず，構文解析過程で括弧なしコーパス中の各文の全構文木をパーザ固有の WFST (well-formed substring table) [1] の形で獲得する．次に EM 学習過程で，WFST の構文木を支持グラフと呼ばれるデータ構造に変換し，新たに考案されたグラフィカル EM (以下 gEM) アルゴリズム [4] を支持グラフ上で走らせて学習パラメータを得る．構文解析過程では，WFST を用いるパーザであればどのようなものでも利用可能である．I-O アルゴリズムでは対象文法が Chomsky 標準形に限られるのに対し，本手法は一般化 LR (GLR) パーザなどを利用することで Chomsky 標準形でない文法も利用可能である．3 節で提案手法のアルゴリズムを示す．I-O アルゴリズムは三角行列の共有構造を用いて，複数の構文木に対する部分木共有 (sub-tree sharing) を行なっているが，入力文を導出しない要素も走査している．一方，Fujisaki らは構文木を展開した形で取り出して走査する方法 [3] を用いており，I-O アルゴリズムのように入力文を導出しない部分を走査することはないが，部分木共有は行なっていない．支持グラフは構文木を取り出し，部分木共有も行なっているため，両者のメリットを合わせ持っている．また，Pereira らは，I-O アルゴリズムによる文法学習において，人間が部分的に括弧を与えた括弧付きコーパスを訓練コーパスとすることで学習モデルに基づく解析精度が上がることを実験により示した [11]．それに対し，本提案手法では人間が与える情報として CFG の骨格を用意し，学習の対象をパラメータにしぼることで，学習の困難さを軽減している．

本手法は Charniak らの疑似文脈依存文法 (以下 pseudo PCSG) [2] や北らの規則 bigram [5] など，PCFG に文脈を加えた拡張文法に対する EM アルゴリズムも包含している．これらの多くは，相対頻度法による実験で PCFG よりも優れた解析精度が確認されているので，EM 学習でも精度の向上が期待される．ただし，PCFG の拡張文法ではパラメータ数が増大するため，提案手法の実装ではいくつかの対応策を用いた．提案手法への拡張文法の実装については，4 節で述べる．

我々は ATR 対話コーパスを用いて，提案手法の実験を行なった．その結果を 5 節に示す．学習速度についての実験では，I-O アルゴリズムと比べ，コーパス平均文長においておよそ 1,500 倍の速度向上が確認された．これは，提案手法 (学習過程の分離及び支持グラフの利用) の有効性を示すものである．最後に 6 節で本報告をまとめ，関連研究との比較を行なう．

2 準備

2.1 確率文脈自由文法

まず，文脈自由文法 G を 4 つ組 $\langle V_{\rm n}, V_{\rm t}, R, S\rangle$ で定義する．$V_{\rm n}$，$V_{\rm t}$，R はそれぞれ非終端記号，終端記号，規則の集合，S は開始記号 ($S \in V_{\rm n}$) である．S から規則適用を繰り返すことで導出可能な終端記号列 $\boldsymbol{w} = w_1 w_2 \cdots w_n (w_i \in V_{\rm t})$ を文と呼ぶ．G に基づく PCFG を $G(\boldsymbol{\theta})$ で表し，G を「$G(\boldsymbol{\theta})$ における CFG の骨格」と呼ぶ．$\boldsymbol{\theta}$ は $|R|$ 次元ベクトルであり，以降パラメータと呼ぶ．$\boldsymbol{\theta}$ の各要素を $\theta(r)$ で参照し，$0 \leq \theta(r) \leq 1$, $\sum_{\zeta:(A\rightarrow\zeta)\in R_A} \theta(A\rightarrow\zeta) = 1$ が成り立つとする ($r \in R,\ A \in V_{\rm n}$)．適用規則列 $\boldsymbol{r} = \langle r_1, r_2, \ldots, r_K\rangle$ の出現確率 $P(\boldsymbol{r})$ は，$P(\boldsymbol{r}) = \prod_{k=1}^{K} \theta(r_k)$ と計算される．また，S から $\boldsymbol{w}$ を導出する全ての適用規則列の集合を $\psi(\boldsymbol{w})$ とおくと，$\boldsymbol{w}$ が S から $\boldsymbol{r}$ によって生成される確率 $P(\boldsymbol{r}, \boldsymbol{w})$ は，

$$P(\boldsymbol{r}, \boldsymbol{w}) = \begin{cases} P(\boldsymbol{r}) & \text{if } \boldsymbol{r} \in \psi(\boldsymbol{w}) \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

[1] WFST は構文解析の途中において解析済みの部分木を記録するための表 (データ構造) である [9]．CYK (Cocke-Younger-Kasami) パーザでは三角行列，Earley パーザではアイテム集合の集まり，一般化 LR パーザでは共有圧縮統語森 (packed shared parse forest) がそれぞれ WFST となっている．

となる．よって S から $\boldsymbol{w}$ が導出される確率 $P(\boldsymbol{w})$ は

$$P(\boldsymbol{w}) = \sum_{\mathrm{all}\,\boldsymbol{r}} P(\boldsymbol{r}, \boldsymbol{w}) = \sum_{\boldsymbol{r} \in \psi(\boldsymbol{w})} P(\boldsymbol{r}) \qquad (2)$$

と書くことができる．後に示す提案手法では $\psi(\boldsymbol{w})$ に含まれる規則列だけを WFST から取り出して学習する．また，提案手法では，文法には ε 生成規則や $A \stackrel{+}{\Rightarrow} A$ のようなサイクルになる規則は含まないと仮定する．

2.2 相対頻度法と I-O アルゴリズム

PCFG の確率パラメータを獲得する方法として，構造つきコーパスに現れる規則を数え上げる相対頻度法が知られている．相対頻度法では，パラメータは $\theta(r) = \frac{\sigma(r)}{\sum_{\zeta} \sigma(\mathtt{A} \to \zeta)}$ で求まる．ただし $\sigma(r)$ は構造付きコーパスに現れる r の数である．相対頻度法は精度の高いパラメータを学習できるが，構造付きコーパスの作成に要する人的コストが大きいという問題がある．

一方 I-O アルゴリズムは括弧なしコーパスからパラメータ推定を行なう EM アルゴリズムで，括弧なしコーパス $\mathcal{C} = \langle \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \rangle$ と CFG G が与えられたとき，尤度 $P(\mathcal{C}|\boldsymbol{\theta}) = \prod_{\ell=1}^{N} P(\boldsymbol{w}_\ell \mid \boldsymbol{\theta})$ に対して $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} P(\mathcal{C}|\boldsymbol{\theta})$ なる (局所) 最尤推定値 $\boldsymbol{\theta}^*$ を見つける．G は Chomsky 標準形であることを前提としている．パラメータ更新手続きの中心は内側確率と外側確率という 2 種類の確率値の計算である．これらの計算は三角行列 (図 1; 次頁) の上で行なわれ，文長 L に対して計算量は $O(|V_{\mathrm{n}}|^3 L^3)$ となる．この計算コストが I-O アルゴリズムの問題の一つであり，大きな文法やコーパスを用いての学習は現実的でない．

3 提案手法

我々の手法は，CFG の骨格と括弧なしコーパスを入力とし，PCFG の学習パラメータを出力として返す．学習過程全体を次の 2 つのステップに分離する．ステップ 1 は，任意のパーザを用いて実行可能である．

ステップ 1 (構文解析): 括弧なしコーパス中の各文に構文解析を施し，その文の構文木すべてを得る．ただし，構文木は実際に構築せずに途中で構築される WFST のままでとどめておく．

ステップ 2 (EM 学習): ステップ 1 で得られた WFST から支持グラフと呼ばれるデータ構造を抽出し，新たに考案されたグラフィカル EM (graphical EM; 以下 gEM) アルゴリズム [4] を支持グラフ上で走らせ，学習パラメータを得る．

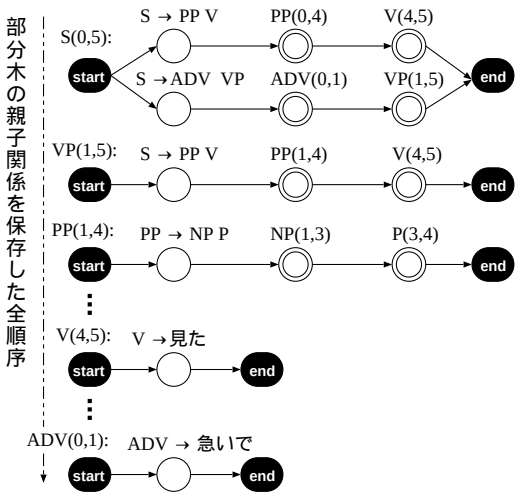

図 2: 三角行列 T から抽出された支持グラフ．

3.1 支持グラフ

ステップ 2 では，まず WFST から支持グラフを抽出する．ここでは I-O アルゴリズムとの比較のため CYK パーザの WFST である三角行列を用いる．

図 2 に示すように，支持グラフは再帰遷移ネットワーク (RTN) [7] に似た非循環有向グラフ (DAG) の集合である．個々の部分グラフにはラベル $A(d, d')$ が付与されており，start と end と書かれたノード以外は規則 $A \to \zeta$ もしくはラベル $A(d, d')$ が付与されている．ラベル $A(d, d')$ は，$A \stackrel{*}{\Rightarrow} \boldsymbol{w}_{d,d'}$[2]に対応する部分構文木を表す．各部分グラフは確率的な関係を表しており，例えば図 2 の一番上の部分グラフは

$$\begin{aligned} P(\mathtt{S} \stackrel{*}{\Rightarrow} \boldsymbol{w}_{0,5}) = {} & \theta(\mathtt{S} \to \mathtt{PP\ V}) P(\mathtt{PP} \stackrel{*}{\Rightarrow} \boldsymbol{w}_{0,4}) P(\mathtt{V} \stackrel{*}{\Rightarrow} \boldsymbol{w}_{4,5}) \\ & + \theta(\mathtt{S} \to \mathtt{ADV\ VP}) P(\mathtt{ADV} \stackrel{*}{\Rightarrow} \boldsymbol{w}_{0,1}) P(\mathtt{VP} \stackrel{*}{\Rightarrow} \boldsymbol{w}_{4,5}) \end{aligned}$$

という関係を表現する．また，支持グラフ中の各部分グラフは各自のラベルの部分木の親子関係を保持するように全順序で並べられている．

支持グラフはコーパス中の各文 $\boldsymbol{w}_\ell$ に対して構築され，$(O_\ell, \tilde{\psi}_\ell)$ で表される $(\ell = 1 \ldots N)$．O_ℓ は支持グラフの部分グラフに付与されているラベルのリストで，部分木の親子関係を保存する

[2] $\boldsymbol{w}_{d,d'}$ は文 $\boldsymbol{w}$ の部分単語列 $w_{d+1} w_{d+2} \cdots w_{d'}$，$P(A \stackrel{*}{\Rightarrow} \boldsymbol{w}_{d,d'})$ は規則適用列により A から $\boldsymbol{w}_{d,d'}$ が生成される確率である．

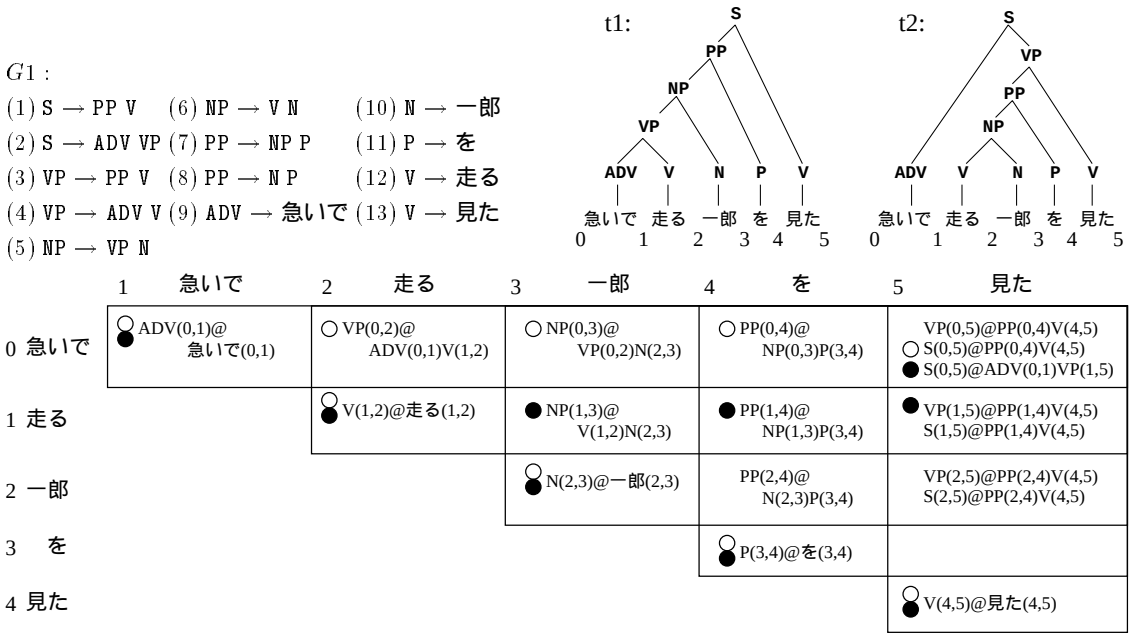

図 1: $G1$ と文 ⟨ 急いで, 走る, 一郎, を, 見た ⟩ に対する三角行列 T(下), 三角行列 T から抽出された 2 つの構文木 (右上) .

ように並べられている．図 1 の例では O_ℓ = $\langle \mathtt{S}(0,5), \mathtt{VP}(1,5), \mathtt{PP}(1,4), \mathtt{NP}(1,3), \mathtt{V}(4,5), \mathtt{PP}(0,4), \mathtt{P}(3,4), \mathtt{NP}(0,3), \mathtt{N}(2,3), \mathtt{VP}(0,2), \mathtt{V}(1,2), \mathtt{ADV}(0,1)\rangle$ となる．また，$\tilde{\psi}_\ell$ は各部分グラフの構造を表しており，図 2 のグラフでは以下のようになっている．

$$
\begin{aligned}
\tilde{\psi}_\ell(\mathtt{S}(0,5)) &= \{\ \{\mathtt{S}\rightarrow\mathtt{PP\ V},\ \mathtt{PP}(0,4),\ \mathtt{V}(4,5)\},\\
&\qquad \{\mathtt{S}\rightarrow\mathtt{ADV\ VP},\ \mathtt{ADV}(0,1),\ \mathtt{VP}(1,5)\}\ \}\\
\tilde{\psi}_\ell(\mathtt{VP}(1,5)) &= \{\ \{\mathtt{VP}\rightarrow\mathtt{PP\ V},\ \mathtt{PP}(1,4),\ \mathtt{V}(4,5)\}\ \}\\
\tilde{\psi}_\ell(\mathtt{PP}(1,4)) &= \{\ \{\mathtt{PP}\rightarrow\mathtt{NP\ P},\ \mathtt{NP}(1,3),\ \mathtt{P}(3,4)\}\ \}\\
\tilde{\psi}_\ell(\mathtt{NP}(1,3)) &= \{\ \{\mathtt{NP}\rightarrow\mathtt{V\ N},\ \mathtt{V}(1,2),\ \mathtt{N}(2,3)\}\ \}\\
\tilde{\psi}_\ell(\mathtt{V}(4,5)) &= \{\ \{\mathtt{V}\rightarrow\text{見た}\ \}\ \}\\
\tilde{\psi}_\ell(\mathtt{PP}(0,4)) &= \{\ \{\mathtt{PP}\rightarrow\mathtt{NP\ P},\ \mathtt{NP}(0,3),\ \mathtt{P}(3,4)\}\ \}\\
\tilde{\psi}_\ell(\mathtt{P}(3,4)) &= \{\ \{\mathtt{P}\rightarrow\text{を}\ \}\ \}\\
\tilde{\psi}_\ell(\mathtt{NP}(0,3)) &= \{\ \{\mathtt{NP}\rightarrow\mathtt{VP\ N},\mathtt{VP}(0,2),\ \mathtt{N}(2,3)\}\ \}\\
\tilde{\psi}_\ell(\mathtt{N}(2,3)) &= \{\ \{\mathtt{N}\rightarrow\text{一郎}\ \}\ \}\\
\tilde{\psi}_\ell(\mathtt{VP}(0,2)) &= \{\ \{\mathtt{VP}\rightarrow\mathtt{ADV\ V},\ \mathtt{ADV}(0,1),\ \mathtt{V}(1,2)\}\ \}\\
\tilde{\psi}_\ell(\mathtt{V}(1,2)) &= \{\ \{\mathtt{V}\rightarrow\text{走る}\ \}\ \}\\
\tilde{\psi}_\ell(\mathtt{ADV}(0,1)) &= \{\ \{\mathtt{ADV}\rightarrow\text{急いで}\ \}\ \}
\end{aligned}
$$

三角行列は複数の構文木に対して部分木共有を行なっているが，構文木を構成しない要素も含まれている．一方，支持グラフに抽出するのは，三角行列中の構文木を構成する (式 2 の $\psi(\boldsymbol{w})$ に含まれる，図 1 では○ ● の) 部分だけであるので，三角行列よりも無駄のない構造になっていることがわかる．CYK パーザ用の支持グラフ抽出アルゴリズムを以下に示す．

```
1: procedure Extract-CYK() begin
2:   for ℓ := 1 to N do begin
3:     Initialize all ψ̃_ℓ(·) to ∅ and all Visited[·] to NO;
4:     ClearStack(U);
5:     Visit-CYK(ℓ, S, 0, n_ℓ);
6:     for k := 1 to |U| do τ_k := PopStack(U);
7:     O_ℓ := ⟨τ_1, τ_2, ..., τ_|U|⟩
8:   end
9: end.
```

```
1: procedure Visit-CYK(ℓ, A, d, d') begin
2:   Put τ = A(d, d'); Visited[τ] := YES;
3:   if d' = d + 1 and A(d, d')@w_d'(d, d') ∈ T^(ℓ)_{d,d'} then
4:     Add a set {A → w_d'} to ψ̃_ℓ(τ)
5:   else foreach A(d, d')@B(d, d'')C(d'', d') ∈ T^(ℓ)_{d,d'}
6:     do begin
7:       Add a set {A → BC, B(d, d''), C(d'', d')}
8:           to ψ̃_ℓ(τ);
9:       if Visited[B(d, d'')] = NO then
10:        Visit-CYK(ℓ, B, d, d'');   /* 再帰 */
11:      if Visited[C(d'', d')] = NO then
12:        Visit-CYK(ℓ, C, d'', d')   /* 再帰 */
13:    end;
14:  PushStack(τ, U)
15: end.
```

O_ℓ はトポロジカルソーティングの考えに基づいて獲得している．*Visit-CYK* 中の $T^{(\ell)}_{d,d'}$ は w_ℓ の三角行列 $T^{(\ell)}$ の d 行 d' 列の要素を表す．

3.2 グラフィカル EM アルゴリズム

I-O アルゴリズムと同様に，gEM アルゴリズムでも内側確率，外側確率という 2 つの確率値の計算が中心になる．gEM アルゴリズムを以下に示す．

```
 1: procedure Graphical-EM() begin
 2:   Initialize all parameters θ(A→ζ)
 3:     such that P(w_ℓ|θ) > 0 for all ℓ = 1...N;
 4:   Get-Inside-Probs();
 5:   λ^(0) := Σ_{ℓ=1}^{N} log P[ℓ, S(0, n_ℓ)];
 6:   repeat
 7:     Get-Expectations();
 8:     foreach (A→ζ) ∈ R do
 9:       θ(A→ζ) := η[A→ζ] / Σ_{ζ'} η[A→ζ'];
10:     m += 1;
11:     Get-Inside-Probs();
12:     λ^(m) += Σ_{ℓ=1}^{N} log P[ℓ, S(0, n_ℓ)]
13:   until λ^(m) − λ^(m−1) becomes sufficiently small
14: end.
```

```
 1: function Get-Inside-Probs() begin
 2:   for ℓ := 1 to N do begin
 3:     Put O_ℓ = ⟨τ_1, τ_2, ..., τ_{|O_ℓ|}⟩;
 4:     for k := |O_ℓ| downto 1 do begin
 5:       foreach E ∈ ψ̃_ℓ(τ_k) do begin
 6:         R[ℓ, τ_k, E] := 1;
 7:         foreach τ' ∈ E do
 8:           if τ' = (A→ζ) then
 9:             R[ℓ, τ_k, E] *= θ(A→ζ)
10:           else R[ℓ, τ_k, E] *= P[ℓ, τ'];
11:       end; /* foreach E */
12:       P[ℓ, τ_k] := Σ_{E∈ψ̃_ℓ(τ_k)} R[ℓ, τ_k, E]
13:     end /* for k */
14:   end /* for ℓ */
15: end.
```

```
 1: procedure Get-Expectations() begin
 2:   foreach (A→ζ) ∈ R do η[A→ζ] := 0;
 3:   for ℓ := 1 to N do begin
 4:     Put O_ℓ = ⟨τ_1, τ_2, ..., τ_{|O_ℓ|}⟩;
 5:     Q[ℓ, τ_1] := 1;   /* τ_1 は特別に 1 に初期化 */
 6:     for k := 2 to |O_ℓ| do Q[ℓ, τ_k] := 0;
 7:     for k := 1 to |O_ℓ| do
 8:       foreach E ∈ ψ̃_ℓ(τ_k) do
 9:         foreach τ' ∈ E do
10:           if τ' = (A→ζ) then
11:             η[A→ζ] +=
12:                 Q[ℓ, τ_k] · R[ℓ, τ_k, E] / P[ℓ, S(0, n_ℓ)]
13:           else if P[ℓ, τ'] > 0 then
14:             Q[ℓ, τ'] += Q[ℓ, τ_k] · R[ℓ, τ_k, E] / P[ℓ, τ']
15:   end /* for ℓ */
16: end.
```

gEM のメインルーチン *Graphical-EM* はサブルーチン *Get-Inside-Probs* で内側確率を，*Get-Expectations* で外側確率とパラメータの期待値を計算している (計算結果はそれぞれ配列 $\mathcal{P}[\ell, \tau]$，$\mathcal{Q}[\ell, \tau]$，$\eta[A \rightarrow \zeta]$ に格納される)．

計算時間は支持グラフのサイズ (start と end を除いたノード数) のオーダとなる．支持グラフのサイズは最悪の場合 $O(|V_n|^3 L^3)$ となるので，最悪計算量は I-O アルゴリズムと同じである．しかし，三角行列の要素が疎な場合には，学習速度の向上が期待される．

4 PCFG の拡張文法

PCFG は文脈とは無関係に句構造を決めてしまうため，自然言語のような複雑な確率現象を表現するモデルとしては問題が多いことが指摘されている．それに対し，北らの規則 bigram モデル [5] や Charniak らの pseudo PCSG [2] など，PCFG に文脈情報を採り入れたモデル (PCFG の拡張文法と呼ぶ) が数多く提案されている．PCFG $G(\theta)$ では，構文木中で規則 r が適用される確率を $\theta(r)$ としていたが，拡張文法では $\theta(r \mid F(\pi))$ とする．ただし，π は構文木中で r が適用されるまでの部分過程の情報である．関数 $F(\pi)$ は，規則を適用する際に π 中のどの情報を文脈として採用するかを決める関数である．F の選び方によって様々な拡張文法が考えられ，規則 bigram モデルでは，最左導出において直前に適用された規則を，pseudo PCSG では r の左辺に現れる非終端記号の親ノードを文脈として採用している．

我々は図 3 の 4 つの拡張文法，すなわち pseudo PCSG，最左導出において直前に生成された終端記号を文脈とするモデル (以後 Bigram と呼ぶ)，さらにこれら 2 つの文脈を両方用いるモデル (Bigram+pseudo PCSG と呼ぶ), 規則 bigram モデルを提案手法に実装した．各拡張文法に対する支持グラフ抽出ルーチンを用意すれば，EM 部分のアルゴリズムは $A \rightarrow \zeta$ を $A \rightarrow \zeta | F(\pi_\ell)$ とし，*Get-Expectations* の行 2 と *GraphicalEM* の行 8-9 のループに文脈についてのループを重ねるだけでよい．

PCFG の拡張文法では，r の適用が文脈情報により細分化されるので，パラメータ数が増大し，データの過疎性の問題が生じる．それに対し，我々は EM 学習に Laplace smoothing を採り入れた[3]．紙面の都合上

[3] 自然言語処理分野においては，Nigam らが EM アルゴリズム

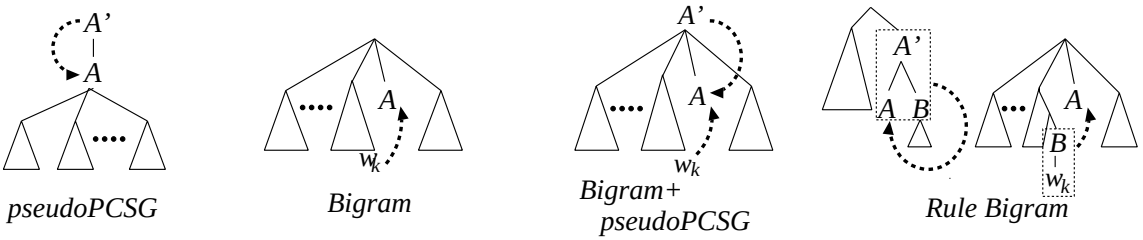

図 3: 提案手法に実装した 4 つの拡張モデル (規則 bigram では最左導出に固定している)．

説明は省略するが, 最大化の対象を尤度 $P(\mathcal{C}|\theta)$ ではなく事後確率 $P(\theta|\mathcal{C})$ とする．また，Laplace smoothing を採り入れることで，収束までに要する更新回数が減少した[4]．また，パラメータが増加し尤度 (事後確率) 関数が複雑になると，局所解に陥りやすくなるという問題が起こる．局所解の対策としては初期パラメータを変えて繰り返し学習を行なう再出発法がある．ただ，我々の実験では初期の更新で尤度が大幅に上昇するが，以降は尤度の更新幅が非常に小さくなることが確認されたため，どの初期パラメータに対しても収束まで学習を行なうのは効率的でない．そこで，我々はヒューリスティクスとして選択的な再出発法を用いることにした．まず，いくつかの初期パラメータに対して，定数回更新を繰り返した時点の尤度 (smoothing 適用時は事後確率) を比較し，その中で最大のパラメータのみ収束まで学習を行なう．そして，これを 1 回の学習として，再出発法のように繰り返し学習を行なうという方法である．

5 実験

構造付きコーパスである ATR 対話コーパス (SLDB) と，田中らによって開発された CFG G^* [13] を用いて実験を行なった．G^* は品詞を細分化したカテゴリを終端記号とした CFG で，規則数 860，非終端記号数 173，終端記号数 441 である．ATR 対話コーパス中の文では，(実際の単語ではなく) 上記カテゴリの列を対象とした．コーパスは 10,995 文からなり，文長 (終端記号数) は平均で 9.97，最短 2，最長 49 である．G^* は Chomsky 標準形でない規則も含むので，本実験では東京工業大学，田中・徳永研究室で開発・公開されている MSLR パーザとの組合せを採用している．

5.1 学習時間に関する実験

ATR コーパス $\mathcal{C}$ と先述した文法 G^* が与えられた場合のパラメータ更新に要する計算時間を提案手法と I-O アルゴリズムの間で比較する．具体的には，文長 L を変化させたときにパラメータを一回更新するのに要する計算時間 (更新時間と呼ぶ) が変化する様子を比較する．まず，我々は ATR コーパス $\mathcal{C}$ の非終端記号と括弧情報を除いたコーパス $\mathcal{C}'$ を作成し，$\mathcal{C}'$ の中で文長 $L-1$ と L の文をグループ化し，各々から無作為に取り出した 100 文を $\mathcal{C}_L$ とする $(L = 2, 4, \ldots 26)$[5]．そして，各 $\mathcal{C}_L$ を訓練コーパス，G^* を対象文法として，I-O アルゴリズムと提案手法でそれぞれ学習を行ない，1 回のパラメータ更新時間を比較した[6]．提案手法には，PCFG と先述の 4 つの拡張文法を実装している．

更新時間を計測した結果を図 4 に示す．*"I-O"* は I-O アルゴリズムの，それ以外は各モデルを実装した提案手法の更新時間を示している．見やすさのため，縮尺を変えた 3 つのグラフを用意し, 図の左と中央のグラフでは PCFG だけを表示している．グラフからわかるように，各モデルの gEM アルゴリズムは，I-O アルゴリズムに比べてはるかに少ない更新時間で学習を行なっている．I-O アルゴリズムと同じ PCFG で比較すると，平均文長 (9.97) に近い $L = 10$ ではおよそ 1,500 倍の速度向上が得られた[7]．また，I-O アルゴリズムは理論値どおり L^3 の曲線を描いているのに対し，PCFG モデルの gEM アルゴリズムは，$2 \leq L \leq 26$ の範囲では L に対しほぼ線形時間で計算できている．

次に，PCFG について，提案手法の学習時間全体の内訳 (構文解析，支持グラフ抽出，gEM 実行) を計測

による文書分類で Laplace smoothing を用いている [10]．

[4] 5 節の実験で用いる文法とコーパス全文からの学習では，Bigram モデル (パラメータ数 14,612) の更新回数は Laplace smoothing を用いなかった場合は 2,773，用いた場合は 301 となった．

[5] 長さ 27 以上の 176 文 (全体の 1.6 %) はデータ不足のため，この実験では考慮しなかった．

[6] I-O アルゴリズムでは Chomsky 標準形でしか動作しないので，あらかじめ G^* を Chomsky 標準形に変換した．その結果規則数が 2,307 (非終端記号数 210, 終端記号数 441) の文法になった．

[7] 以前の計測では 1,000 倍だったが，実装を見直した結果，この数値となった

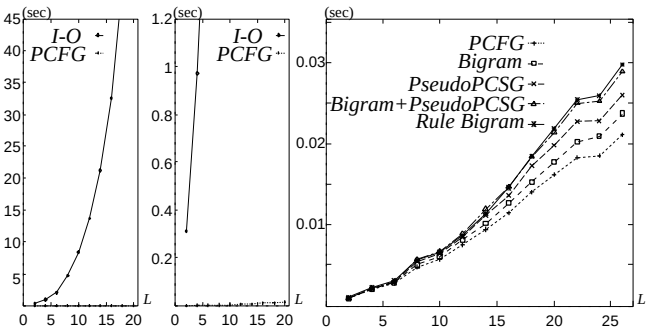

図 4: I-O アルゴリズムと, 各モデルの gEM アルゴリズムにおける，文長に対するの更新時間の変化．

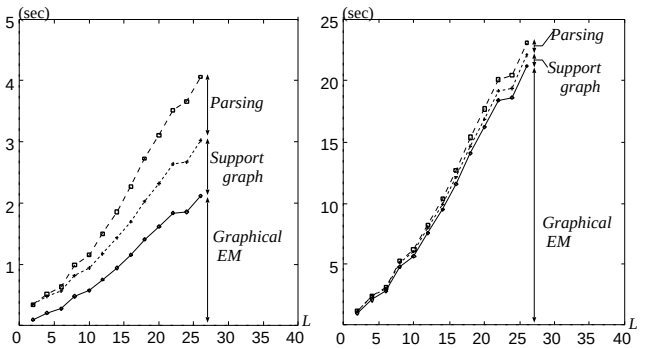

図 5: 学習時間に占める各過程の処理時間の内訳 (左) 再出発なしの場合 $(h = 1)$ (右) 再出発回数 $h = 10$ の場合．

した．gEM 実行時間は先述の通り，更新時間，収束までの更新回数，再出発回数 h の積である．

先ほどの $\mathcal{C}_L$ を用いて，再出発なし $(h = 1)$ の場合と，10 回の再出発を行なう $(h = 10)$ 場合について学習を行なった．ただし，収束までの更新回数はパラメータの初期値やコーパスによって異なるので，更新回数を 100 回に固定した．図 5 が結果のグラフである．左のグラフと比べて，右グラフでは構文解析と支持グラフ抽出の割合が非常に小さい．これは再出発の際に改めて構文解析と支持グラフ抽出を行なう必要がないためである．構文解析部と EM 学習部に分離したことが学習の高速化につながっていることがわかる．

5.2 パラメータの解析精度に関する実験

前節の実験結果より，gEM アルゴリズムがある程度の大きさのコーパスや文法に対しても，現実的な時間で学習を行なえることが確認できた．本節では前節でも用いた 5 つのモデルの学習パラメータに対する解析精度を，相対頻度法と EM アルゴリズムの間で比較する．EM 学習の訓練コーパスとしては，括弧なしコーパスと括弧付きコーパスを用いた．括弧付きコーパスは，構造付きコーパスから，非終端記号と $A \to B$ の形の規則に関する括弧を取り除いたものである．以下に括弧付きコーパスの例を示す．

PA(1)	PCFG	Bigram	PPCSG	B+PPCSG	RB
相対頻度	81.09	83.25	84.90	85.17	85.60
gEM(U)	71.88	73.75	76.03	77.65	77.15
gEM(B)	73.01	77.59	77.64	81.79	77.09
Baseline	64.37	—	—	—	—

PA(3)	PCFG	Bigram	PPCSG	B+PPCSG	RB
相対頻度	93.74	95.62	95.67	96.03	96.40
gEM(U)	88.89	88.79	89.74	90.40	89.91
gEM(B)	90.60	91.90	90.47	93.44	90.73
Baseline	83.27	—	—	—	—

0-CB	PCFG	Bigram	PPCSG	B+PPCSG	RB
相対頻度	87.89	91.73	92.08	92.71	92.77
gEM(U)	86.83	86.19	89.24	86.68	89.67
gEM(B)	88.49	91.96	92.59	92.77	92.86

表 1: 各モデルの相対頻度法と gEM アルゴリズムの評価．

(はい (((シングルルーム で)(ございま す)) ね))

ただし，学習では単語列ではなく，品詞を細分化したカテゴリ列を用いている．括弧付きコーパスからの学習では，MSLR パーザの，括弧に矛盾しない構文木のみを出力する機能を利用している．また，実験データに偏りがないように，我々は 11-fold の交差検定 (cross-validation) を行なう．

11 回の実験の評価値の平均を表 1 に示す．(U) は括弧なし，(B) は括弧付きコーパスによる学習結果である．PA(n) は，上位 n 番目までの予測構文木のいずれかが正解構文木と完全に一致した文の割合 (%) ，0-CB は最上位の予測構文木と正解構文木の括弧づけに矛盾がなかった文の割合 (%) である．また，Baseline は，$100 \times \sum_{\ell=1}^{N} \min\{1, n/|\psi(\boldsymbol{w}_\ell)|\}$ ，すなわち，ランダムに構文木を予測した場合の PA(n) の期待値である．

G^* は曖昧性の少ない文法であるため，各学習で 70% 以上の精度を上げている．相対頻度法と同様，提案手法による学習パラメータでも，PCFG に文脈情報を加えることで解析精度が上がっていることがわかる．また，0-CB に関しては，提案手法でも相対頻度法にかなり接近している．特に括弧付きコーパスを用いると，どのモデルも相対頻度法を上回っている．この原因については，現在調査中である．

6 まとめと関連研究

本報告では，CFG の骨格が与えられていることを前提として，PCFG 及びその拡張文法を括弧なしコーパスから学習する高速かつ一般的な枠組を提案した．拡張文法では，パラメータ増加にともなうデータの過疎化への対策として Laplace smoothing を，局所解の増加への対策として選択的再出発法を用いた．これらの提案手法と I-O アルゴリズムとの学習速度の比較実験を行ない，大幅な速度向上を確認した．PCFG との比較実験では，コーパスの平均文長においておよそ 1,500 倍速度が向上した．また，各モデルに対し学習パラメータの解析精度を比較した．その結果，EM 学習でも文脈依存性を考慮することで解析精度が上がることを確認した．

PCFG とその拡張文法の EM 学習については，いくつかの研究がなされている．Kupiec は，RTN を用いることで Chomsky 標準形でない PCFG に対して学習を行なう方法を示している [7]．しかし，I-O アルゴリズムと同様，1 回の更新ごとに構文木を探索するので，我々のような高速学習は実現していない．Stolcke は，確率的 Earley パーザで PCFG のパラメータ学習を行なっている [12]．確率的 Earley パーザでは ε 規則やサイクルになる規則も扱える．我々の手法でこのような規則を扱うことは今後の課題であるが，現段階でも充分に実用的である．その点を除けば，Earley パーザと提案手法を連結した場合は Stolcke の手法と等価である．また，Stolcke は PCFG の拡張文法については言及していない．よって，提案手法はあるクラスの PCFG については Stolcke の方法の一般化といえる．PCFG の拡張文法の EM 学習としては，Charniak らが I-O アルゴリズムをもとにした pseudo PCSG の学習を提案している [2]．ただし学習速度についての言及はない．また本実験のように，様々な拡張文法に対する EM 学習の比較は行なわれていない．

提案手法では CFG が与えられることが前提となるが，パラメータ学習とともに文法規則の探索をどう行なっていくかが今後の課題である．また，他のコーパス，文法についても同様の実験を行なっていきたい．

謝辞

実験に用いた ATR コーパス，日本語文法の改訂版は，東京工業大学 田中・徳永研究室のご厚意により提供頂きました．記して感謝致します．また，同研究室 白井清昭助手には上記コーパス・文法に関する情報や文献紹介など大変お世話になりました．重ねて感謝申し上げます．なお，本研究の一部は平成 11 年度 科学研究費補助金 特定領域研究「発見科学」の補助を受けています．

参考文献

[1] Baker, J. K. (1979). Trainable grammars for speech recognition, *Proc. of Spring Conf. of the Acoustical Society of America*, pp.547–550.

[2] Charniak, E. and Carroll, G. (1994). Context-sensitive statistics for improved grammatical language models, *Proc. of AAAI-94*, pp.728–733,

[3] Fujisaki, T., Jelinek, F., Cocke, J., Black, E. and Nishino, T. (1989). A probabilistic parsing method for sentence disambiguation. *Proc. of IWPT-89*, pp.85–94.

[4] Kameya, Y. and Sato, T. (2000). Efficient EM learning for parameterized logic programs. *Proc. of the 1st Conf. on Computational Logic (CL2000)*, pp.269–294.

[5] Kita, K., Morimoto, T., Ohkura, K., Sagayama, S. and Yano, Y. (1994). Spoken Sentence Recognition Based on HMM-LR with Hybrid Language Modeling, *IEICE Trans. on Information and Systems*, Vol.E77-D(2).

[6] 北研二 (1999). 確率的言語モデル. 東京大学出版会.

[7] Kupiec, J. An algorithm for estimating the parameters of unrestricted hidden stochastic context-free grammars, *Proc. of COLING-92*, pp.387–393.

[8] Manning, C. D. and Schütze, H.(1999).*Foundations of Statistical Natural Language Processing*, The MIT Press.

[9] 永田 昌明 (1999). “形態素，構文解析.” 自然言語処理—基礎と応用—(田中穂積監修)第 1 章 (pp.2–45), 電子情報通信学会.

[10] Nigam, K. Mccallum, A. Thrun, S. and Mitchell, T.(2000). The Classification from Labeled and Unlabeled Documents using EM. Machine Learning, 38(2-3),pp.103–134.

[11] Pereira, F and Schabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. *Proc of ACL-92*, pp.128–135.

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

[13] 田中穂積, 竹澤寿幸, 衛藤 純司 (1997). MSLR 法を考慮した音声認識用日本語文法 — LR 表工学 (3) —. 音声言語情報処理研究会, 情報処理学会, Vol.97.