Vol.2012-MPS-88 No.12
2012/5/17

Pitman-Yor過程に基づく確率的木挿入文法モデル

進藤 裕之[1,a)]　藤野 昭典[1]　永田 昌明[1]

概要：Pitman-Yor 過程に基づく木挿入文法の確率モデルを提案する．木挿入文法は，部分木の挿入操作により少数の規則で様々な構文パターンを表現できるが，計算コストが高いという問題がある．そこで，挿入操作を含む文法モデルを等価な文脈自由文法へ変換し，効率的に確率モデルの学習ができることを示す．提案手法を用いて構文解析の実験を行ったところ，提案モデルは木置換文法のモデルと比較して約 18 %コンパクトでありながら，文脈自由文法や木置換文法と比較して解析精度の向上を実現した．

Probabilistic modeling of Tree Insertion Grammars based on the Pitman-Yor process

Hiroyuki Shindo[1,a)]　Akinori Fujino[1]　Masaaki Nagata[1]

Abstract: We propose a probabilistic model of Tree Insertion Grammars (TIG) based on the Pitman-Yor process. Tree insertion is helpful for modeling syntax patterns accurately with compact grammar rules, however, it suffers from high computational cost. In this paper, we propose an efficent method for learning TIG by mapping our model to equivalent context-free grammars. The experimental parsing results show that our model outperforms a standard CFG and Tree Substitution Grammars (TSG) for a small dataset, making the number of grammar rules approximately 18% smaller than with TSG.

1. はじめに

木挿入文法（Tree Insertion Grammars）[16] は，構文木を生成する形式文法モデルの一つであり，文法情報を利用した機械翻訳などの言語処理アプリケーションへ応用されている [6], [7]．文脈自由文法が非終端記号の再帰的な書き換え（置換操作）により文の構造を導出するのに対して，木挿入文法では，部分木（subtree）の再帰的な書き換え（置換・挿入操作）により文の構造を導出する．したがって，木挿入文法は文脈自由文法の自然な拡張であると言える．木挿入文法では任意の大きさの部分木を文法規則として利用するため，例えば述語項構造などの言語学的な特徴を単一の規則で捉えることができる [15].

従来より，木挿入文法の規則を人手で記述したり，Penn Treebank などの構文木コーパスから発見的手法を用いて自動獲得する研究が行われてきた [1], [2], [18]．発見的手法とは，例えば構文木のノードが主辞であるか，または付加詞であるか等の言語学的な知見に基づいて人手で作成されたルールであり，統計的な情報は考慮していない．木挿入文法の規則を人手で記述するには，言語学に関する高度な専門知識が必要となり，多大な時間や費用を要するという問題点がある．また，構文木コーパスから規則を獲得するための発見的手法は，言語やコーパスの仕様に依存するため汎用性が低く，データに過学習しやすいという問題点が指摘されている [15].

これらの問題を解決するため，我々は木挿入文法の確率モデルを考案し，木挿入文法の規則を構文木コーパスから統計的に学習する手法を提案する．ただし，構文木コーパスには木挿入文法の規則に関する情報は付与されていないため，教師なし学習によってそれらを獲得する．木挿入文

[1] NTT コミュニケーション科学基礎研究所
[a)] shindo.hiroyuki@lab.ntt.co.jp

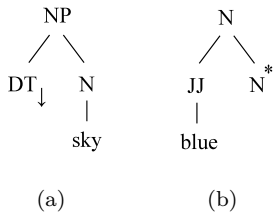

図 1 (a) 初期木の例．境界ノードは記号“↓”で表している．(b) 補助木の例．末尾ノードは記号“∗”で表している．

法では任意の大きさの部分木を規則として扱うため，膨大な部分木の可能性を考慮しなければならない．例えば，部分木を確率変数とする多項分布を考えた場合，部分木の種類は可算無限通りあるため，可算無限次元の多項分布が必要となる．したがって，全ての部分木の可能性を計算機のメモリ上へ展開することは困難であり，EM アルゴリズムのような教師なし学習手法を直接適用して確率モデルの学習を行うことができないという問題がある．そのため，本研究ではノンパラメトリックベイズモデルの一種である Pitman-Yor 過程 [14] を事前分布として利用することで，可算無限次元の部分木の確率分布を構築する．また，確率モデルの学習にマルコフ連鎖モンテカルロ（MCMC）法を利用することで，部分木の全可能性をあらかじめ列挙するのではなく，必要に応じて適応的に生成するというアプローチを取る．このとき，木挿入文法分解という手法を新たに提案し，木挿入文法を等価な文脈自由文法に変換して効率的に確率モデルの学習が可能であることを示す．本手法は，言語やコーパスに関する先見的知見を必要とせず，様々なデータへ適用可能である．

英語の構文木コーパスを用いて提案モデルの学習を行った結果，木挿入文法の挿入操作は全体の規則数を大幅に減らす効果があり，コンパクトな文法で様々な構文パターンを表現可能であった．また，提案モデルの学習によって獲得された木挿入文法の部分木を構文解析タスクへ適用した結果，従来の木置換文法に基づく手法よりも約 18 %少ない規則数であるにも関わらず，構文解析精度の向上を実現した．

2. 木挿入文法

木挿入文法は，様々な基本木を組み合わせて文全体の構文木を生成するモデルである．構文木の生成には，初期木による置換操作と，補助木による挿入操作の二種類がある．図 1 に初期木と補助木の例を示す．図 1 に初期木と補助木の例を示す．図 1(a)(b) はそれぞれ，“(NP (DT) (N sky))” や “(N (JJ blue) (N))” とも表記される．

初期木は，各ノードに非終端記号または終端記号の付与された木構造である．初期木の葉ノードの中で非終端記号が付与されているノードは，特に境界ノード（frontier node）と呼ばれ，他の初期木または補助木によって書き換えられることを表す目印となる（図 1(a) の記号“↓”）．補助木は，初期木と同様に，各ノードに非終端記号または終端記号の付与された木構造であるが，根ノードと同じ非終端記号が付与されている特別な葉ノードが必ず一つ存在する．この葉ノードは末尾ノード（foot node）と呼ばれる（図 1(b) の記号“∗”）．末尾ノードは，補助木の挿入によって切り取られた部分木が結合するノードであることを示しており，詳細は後述する．初期木と補助木をまとめて基本木と呼び，基本木の各ノードの中で葉ノードでも根ノードでもないノードを内部ノード（internal node）と呼ぶ．

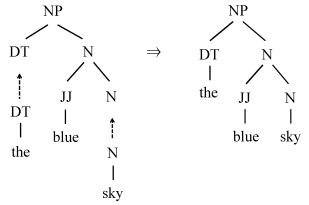

図 2 置換操作の例．

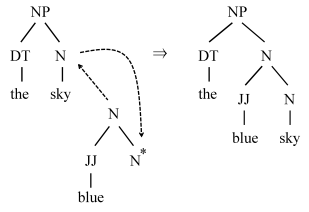

図 3 挿入操作の例．

木挿入文法の導出過程は，まず開始記号 S を根ノードとして，置換または挿入と呼ばれる二種類の操作によって基本木を結合していく．図 2 および図 3 に置換操作と挿入操作の例をそれぞれ示す．置換操作は，部分的に構築された構文木の境界ノードを初期木で置き換える操作である．ただし，境界ノードと，初期木の根ノードの非終端記号は同じでなければならない．一方，挿入操作は，部分的に構築された構文木の内部ノードに対して，補助木を挿入する操作である．ただし，内部ノードと，補助木の根ノードの非終端記号は同じでなければならない．図 3 で示されているように，補助木の挿入によって切り取られた内部ノード以下の部分木は，補助木の末尾ノードと結合する．本研究の木挿入文法モデルでは，図 1(b) のような末尾ノードが根ノードの直接の子供となるような単純な補助木のみを仮定する．

3. 提案手法

3.1 確率モデル

我々の提案する木挿入文法の確率モデルは，基本木の確率的生成モデルと，基本木の置換・挿入操作に基づく構文木の確率的生成モデルとで構成される．

3.1.1 基本木の生成モデル

G_X を，根ノードの非終端記号が X である基本木の確率分布とし，G_X の事前分布として Pitman-Yor 過程 [14] を

仮定する．このとき，根ノードの非終端記号が X である基本木 e の確率分布は，以下のようにモデル化できる．

$$\begin{aligned} e\,|X &\sim G_X \\ G_X\,|d_X,\theta_X &\sim \mathrm{PYP}\,(d_X,\theta_X,P_0\,(\cdot\,|X\,)) \end{aligned} \tag{1}$$

ただし，PYP は Pitman-Yor 過程を表し，$d_X,\theta_X,P_0\,(\cdot\,|X\,)$ は Pitman-Yor 過程のパラメータであり，それぞれ割引パラメータ（discount parameter），強度パラメータ（strength parameter），基底分布（base distribution）と呼ばれる．Pitman-Yor 過程はベキ則に従う確率分布を生成することができ，言語データの統計的性質と親和性が高いことが知られている [8], [17].

式 1 に基づいて基本木 e を i 回生成し，これらを $\mathbf{e}_{1:i}=e_1,e_2,\ldots,e_i$ とする．このとき，$i+1$ 番目の基本木 e_{i+1} の生成確率は，式 1 の G_X を積分消去することにより以下のように計算される．

$$\begin{aligned} p\,(e_{i+1}\,|\mathbf{e}_{1:i},X,d_X,\theta_X\,) &= \alpha_{e_{i+1},X}+\beta_X P_0\,(e_{i+1}\,|X\,) \\ \alpha_{e_{i+1},X} &= \frac{n_{e_{i+1},X}-d_X\cdot t_{e_{i+1},X}}{\theta_X+n_{\cdot,X}} \\ \beta_X &= \frac{\theta_X+d_X\cdot t_{\cdot,X}}{\theta_X+n_{\cdot,X}} \end{aligned} \tag{2}$$

ただし，$n_{e_{i+1},X}$ は，$\mathbf{e}_{1:i}$ のうち e_{i+1} と同じ基本木が生成された回数を表す．$t_{e_{i+1},X}$ は，e_{i+1} のラベルが付与されたテーブルの数を表す．Pitman-Yor 過程では，基本木 e を複数のクラスタに分けて保存しており，各クラスタをテーブルと呼ぶ．また，$n_{\cdot,X}=\sum_e n_{e,X}$，$t_{\cdot,X}=\sum_e t_{e,X}$ である．式 2 において，第一項目は基本木 e_{i+1} と同じ種類の基本木が既に何回生成されたかに基づいて計算される確率，第二項目は基本木 e_{i+1} のスムージング確率と捉えることができる．スムージング確率は，係数 β_X と基底確率 $P_0\,(e_{i+1}\,|X\,)$ の積で表される．

Pitman-Yor 過程の基底分布 P_0 は，基本木 e の基底となる確率分布である．我々は，Cohn ら [4] による木置換文法の確率モデルと同様の基底分布を設定する．具体的には，基本木 e を文脈自由文法の規則（高さが 1 の部分木）に分解し，それらの積として以下のようにモデル化する．

$$\begin{aligned} P_0\,(e\,|X\,) = &\prod_{r\in\mathrm{CFG}(e)} P_{\mathrm{MLE}}\,(r) \\ &\times \prod_{A\in\mathrm{LEAF}(e)} s_A \times \prod_{B\in\mathrm{INTER}(e)} (1-s_B) \end{aligned} \tag{3}$$

ただし, $\mathrm{CFG}(e)$ は, e を高さが 1 となるように分解した部分木の集合で，$P_{\mathrm{MLE}}\,(r)$ は構文木コーパスから計算される部分木 r の最尤推定値を表す．また, $\mathrm{LEAF}\,(e)\,,\mathrm{INTER}\,(e)$ はそれぞれ，e の葉ノードおよび中間ノードにおける非終端記号の集合を表す．s_X は基本木の生成がノード X で停止する確率（停止確率）である．つまり，基底分布 P_0 に基づく基本木 e の生成は，まず根ノード X から始まり，高さが 1 の部分木を結合して部分木を成長させていくモデルである．葉ノードが非終端記号である場合，停止確率 s によって生成を停止するか，または確率 $(1-s)$ でさらに基本木を成長させる．

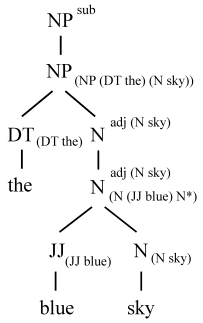

図 4 図 3 の導出過程に木挿入文法分解を適用した例.

分解規則		確率
$\mathrm{NP}^{\mathrm{sub}}$	$\to \mathrm{NP}_{(\mathrm{NP\ (DT\ the)\ (N\ sky)})}$	$\alpha_{(\mathrm{NP\ (DT\ the)\ (N\ sky)}),\mathrm{NP}}$
$\mathrm{NP}_{(\mathrm{NP\ (DT\ the)\ (N\ sky)})}$	$\to \mathrm{DT}_{(\mathrm{DT\ the})}\,\mathrm{N}^{\mathrm{adj\ (N\ sky)}}$	$(1-a_{\mathrm{DT}})\times a_{\mathrm{N}}$
$\mathrm{DT}_{(\mathrm{DT\ the})}$	$\to$ the	1
$\mathrm{N}^{\mathrm{adj\ (N\ sky)}}$	$\to \mathrm{N}^{\mathrm{adj\ (N\ sky)}}_{(\mathrm{N\ (JJ\ blue)\ N*})}$	$\alpha'_{(\mathrm{N\ (JJ\ blue)\ N*}),\mathrm{N}}$
$\mathrm{N}^{\mathrm{adj\ (N\ sky)}}_{(\mathrm{N\ (JJ\ blue)\ N*})}$	$\to \mathrm{JJ}_{(\mathrm{JJ\ blue})}\,\mathrm{N}_{(\mathrm{N\ sky})}$	$(1-a_{\mathrm{JJ}})\times 1$
$\mathrm{JJ}_{(\mathrm{JJ\ blue})}$	$\to$ blue	1
$\mathrm{N}_{(\mathrm{N\ sky})}$	$\to$ sky	1

表 1 図 4 の分解規則と確率.

我々は，初期木と補助木の確率分布を，それぞれ独立に式 1 で定義する．ただし，基底分布は互いの確率分布で共通である．以降，初期木を e，初期木の確率分布に用いる Pitman-Yor 過程を $\mathrm{PYP}\,(d_X,\theta_X,P_0\,(\cdot\,|X\,))$ とし，補助木を e'，補助木の確率分布に用いる Pitman-Yor 過程を $\mathrm{PYP}\,(d'_X,\theta'_X,P_0\,(\cdot\,|X\,))$ と区別する．

3.1.2 構文木の生成モデル

木挿入文法における構文木の生成は，まず開始記号 S を根ノードとして始まり，全ての葉ノードが終端記号となるまで基本木の置換・挿入操作を繰り返す．本節では，この導出過程の確率モデルを新たに提案する．

構文木の導出過程において，構文木の葉ノードが非終端記号であるときには，必ず初期木による置換操作が起こる．したがって，非終端記号が X である構文木の葉ノードが初期木 e によって置換される確率は，式 2 より $1\times p\,(e\,|\mathbf{e},X,d_X,\theta_X\,)$ となる．ただし，e の下付き文字 i は簡単のため省略してある．一方，構文木の内部ノードでは，補助木による挿入操作が確率 a_X で起こると仮定する．つまり，挿入確率は内部ノードの非終端記号 X に依存する．このようなモデル化によって，例えば動詞句では挿入操作が起こりやすく，名詞句では起こりにくいといった現象を捉えることができる．以上のことから，非終端記号が X である構文木の内部ノードが，根ノードの非終端記号が X である補助木 e' によって置換される確率は $a_X\times p\,(e'\,|\mathbf{e}',X,d'_X,\theta'_X\,)$ とモデル化できる．我々のモデルでは，挿入操作は同じ内部ノードで高々一度しか起こらないと制約する．

3.2 推定

構文木コーパス $\mathbf{t}$ が与えられたときに，MAP 推定によっ

て確率モデルを学習し，構文木を構成する木挿入文法の基本木を獲得するという問題を考える．このとき，構文木 t を構成する導出過程 $\mathbf{e}$ の事後分布は，ベイズの定理を用いて以下のように計算できる．

$$P(\mathbf{e}|t) \propto P(t|\mathbf{e})\,P(\mathbf{e}) \tag{4}$$

ただし，$P(\mathbf{e})$ は前述の Pitman-Yor 過程に基づく構文木の生成モデルであり，$P(t|\mathbf{e})$ は，導出過程 $\mathbf{e}$ によって得られる構文木が t と一致すれば $P(t|\mathbf{e}) = 1$，そうでなければ $P(t|\mathbf{e}) = 0$ の値を取る．したがって，構文木 t を生成可能な導出過程のみを考慮して，$P(\mathbf{e})$ からのサンプリングを行えば良い．$P(\mathbf{e}|t)$ からの効率的なサンプリングを実現するために，まず木挿入文法分解法について説明し，次にブロック化メトロポリス・ヘイスティングスアルゴリズム（blocked Metropolis-Hastings algorithm，以下ブロック化 MH アルゴリズムと略す）について述べる．

3.2.1 木挿入文法分解

木挿入文法分解は，我々の木挿入文法モデルを MH アルゴリズムにより効率的に学習するための手法である．基本的なアイディアは，木挿入文法の導出過程を等価な文脈自由文法へ変換することである．その結果，文脈自由文法の標準的な学習アルゴリズムである動的計画法をそのまま適用することができる．同様の手法は Cohn ら [3] によって木置換文法のモデルに使用されているが，我々の手法は挿入操作を考慮している点において異なる．まず，木挿入文法分解の具体例を示す．

例えば，図 3 の導出過程は，まず根ノード NP が初期木 “(NP (DT the) (N sky))” によって置換され，さらに，内部ノード N に補助木 “(N (JJ blue) (N))” が挿入されたものである．これを木挿入文法分解によって文脈自由文法に変換したものが図 4 である．図 4 において，$\mathrm{NP}^{\mathrm{sub}}$ は NP で置換操作が実行されたことを表し，$\mathrm{NP}_{(\mathrm{NP\ (DT\ the)\ (N\ sky)})}$ は，初期木 “(NP (DT the) (N sky))” を必ず生成する特殊な NP ノードである．したがって，NP ノードが初期木 “(NP (DT the) (N sky))” によって置換されたことを表している．また，$\mathrm{N}^{\mathrm{adj\ (N\ sky)}}$ は，内部ノード $\mathrm{N}_{(\mathrm{N\ sky})}$ で挿入操作が起こったことを表しており，$\mathrm{N}^{\mathrm{adj\ (N\ sky)}}_{(\mathrm{N\ (JJ\ blue)\ N*})}$ は，$\mathrm{N}_{(\mathrm{N\ sky})}$ ノードで補助木 “(N (JJ blue) N*)” が挿入されたことを表す．ここで，図 4 の生成規則を文脈自由文法の規則に分解し，表 1 のように確率を割り当てると，式 2 の生成確率に従う導出過程と一致する．表 1 の例では，基本木は全て式 2 の第一項から生成されたと仮定しているが，第二項から生成する場合は $\alpha_{e_i,X}$ や $\alpha'_{e_i,X}$ の代わりに $\beta_{e_i,X}$ や $\beta'_{e_i,X}$ を割り当てればよい．以上のように，木挿入文法の導出過程を図 4 のように変換し，基本木の生成を表 1 のように文脈自由文法の確率の積で表現する方法が木挿入文法分解である．

前述の木挿入文法分解を一般化すると以下のようになる．木挿入文法分解では，式 2 に従う基本木 e_{i+1} の生成が第一項目 $\alpha_{e_{i+1},X}$ からの生成なのか，または第二項目 $\beta_X P_0(e_{i+1}|X)$ からの生成なのかを明示的に区別する．第一項目からの生成は，以前生成した基本木 $\mathbf{e}_{1:i}$ の中から確率的にいずれかを選択することに相当する．一方で，第二項目からの生成は，基底分布 $P_0(e_i|X)$ にしたがって新たに基本木を生成することに相当する．また，非終端記号 X を以下の六種類に分類する．

初期木のノードに使用される非終端記号

- X^{sub}：置換操作が起こるノードであり，非終端記号が X であることを表す．X^{sub} は，確率 $\alpha_{e,X}$ で子ノード $X_{(e)}$ を生成するか，または確率 β_X で子ノード $X_{(\mathrm{base})}$ を生成する．これはつまり，式 2 の第一項目から初期木 e を生成して置換操作を行うのか，または第二項目から初期木 e を生成して置換操作を行うのかを表している．
- $X_{(e)}$：式 2 の第一項目から生成された，初期木の根ノードまたは内部ノードの非終端記号を表す．$X_{(e)}$ は，必ず e の情報に基づいて子ノードを生成しなければならない特殊なノードである．例えば，X を NP とし，e を “NP (DT) (N girl)” とすると，$\mathrm{NP}_{(\mathrm{NP\ (DT\ )\ (N\ girl)})}$ は二つの子ノード：$\mathrm{DT}^{\mathrm{sub}}$ と $\mathrm{N}_{(\mathrm{N\ girl})}$ を確率 1 で生成する．同様に，$\mathrm{N}_{(\mathrm{N\ girl})}$ は終端記号 “ girl ” を確率 1 で生成する．
- $X_{(\mathrm{base})}$：初期木の基底分布から生成された，根ノードまたは内部ノードの非終端記号を表す．$X_{(\mathrm{base})}$ は，初期木の基底分布に従って子ノードを生成する．

補助木のノードに使用される非終端記号

- $X^{\mathrm{adj}(type)}$：挿入操作が起こるノードであり，非終端記号が $X_{(type)}$ であることを表す．ただし，“$type$” は，“ base ” または e を取る．X^{sub} と同様に，$X^{\mathrm{adj}(type)}$ は確率 $\alpha'_{e,X}$ で子ノード $X^{\mathrm{adj}(type)}_{(e')}$ を生成するか，または確率 β'_X で子ノード $X^{\mathrm{adj}(type)}_{\mathrm{base}}$ を生成する．
- $X^{\mathrm{adj}(type)}_{(e')}$：式 2 の第一項目から生成された，補助木の根ノードまたは内部ノードの非終端記号を表す．$X_{(e)}$ と同様に，補助木 e' の情報に基づいて子ノードを生成する．
- $X^{\mathrm{adj}(type)}_{\mathrm{base}}$：補助木の基底分布から生成された，根ノードまたは内部ノードの非終端記号を表す．補助木の基底分布に従って子ノードを生成する．

以上の分類は，構文木の各ノードで置換操作または挿入操作が起こったかどうか，そのときに式 2 の第一項目から基本木が生成されたか，または第二項目から生成されたのかという細かな情報を非終端記号に付与したものと考えられる．このように，各ノードに導出過程に関する全ての情報を付与することで，木挿入文法モデルを表 1 のように文脈自由文法として扱い，動的計画法に基づく効率的な導出過

程の推定が可能となる．

3.2.2 ブロック化 MH アルゴリズム

Johnson ら [9] によって提案されたブロック化 MH アルゴリズムを用いると，構文木コーパス $\mathbf{t}$ から我々の確率モデルを学習することができる．MH アルゴリズムは MCMC 法の一種であり，任意の確率分布からランダムサンプルを得る汎用的な手法である．我々の確率モデルでは，導出過程の事後分布 $p(\mathbf{e}\,|\mathbf{t},\mathbf{d},\boldsymbol{\theta},\mathbf{s},\mathbf{a})$ に従う基本木のサンプルを得ることが目標となる．これは，前述の木挿入文法分解を利用すれば，構文木 t を文脈自由文法でモデル化し，各ノードの非終端記号 X に $X_{(e)}, X_{(\mathrm{base})}$ などの情報を教師なし学習で付与する問題と等価である．また，モデルのハイパーパラメータ $(\mathbf{d},\boldsymbol{\theta},\mathbf{s},\mathbf{a})$ も同様に MH アルゴリズムで学習することができるが，紙面の都合上省略する．

ブロック化 MH アルゴリズムは，構文木 t ごとに以下の三ステップで構成される．

(1) 構文木 t の各ノードにおいて内側確率 [10] を計算する．我々のモデルでは，X が $X^{\mathrm{sub}}, X_{(e)}, X_{(\mathrm{base})}$ などの全ての可能性を考慮して内側確率を計算する．内側確率の計算は，構文木の葉ノードからボトムアップに行われる．

(2) 1 で計算された内側確率に基づいて，図 4 のような導出過程のサンプルを得る．これは，構文木の根ノードからトップダウンに行われる．

(3) MH アルゴリズム [9] によって，サンプルを受理または棄却する．MH アルゴリズムは，マルコフ連鎖が正しい事後分布へ収束するために必要な確率計算である．サンプルが受理された場合は，現在の導出過程を新たなサンプルで置き換える．棄却された場合は，現在の導出過程をそのまま保持する．

上記の三ステップを，全ての構文木について順番に繰り返すことで，導出過程の事後分布に基づくサンプルを得ることができる．ブロック化 MH アルゴリズムの詳細は，文献 [3], [9] に示されている．

4. 実験

4.1 設定

英語の構文木コーパスである WSJ Penn Treebank[11] を用いて提案手法の評価を行った．Penn Treebank コーパスは，学習用データをセクション 2（約 2000 文），テスト用データをセクション 22（約 2000 文）とした．データの前処理として，まずコーパス中の全構文木を “ CENTER-HEAD ” 法 [12] により二分木へ変換した．また，コーパス中に一度しか現れない単語は，単純なルールによって “ UNKNOWN ”, “ DIGIT ”, “ PROPER_NOUN ” のいずれかに置き換えた．実験では，それぞれの学習データに対して MH アルゴリズムを用いて確率モデルの学習を行った．また，MH アルゴリズムで獲得された木挿入文法の基本木を用いて構文解析器を作成し，構文木の情報を取り除いたテストデータに構文解析を行なって精度を評価した．構文解析の精度評価は，EVALB[*1]で計算されるブラケティング（bracketing）F 値 [5] を用いた．本実験で用いた計算機は，CPU が Xeon 5600 3.33GHz，メモリが 48GB である．

反復回数	対数尤度	時間（sec）	F 値
500	-282418.4	1258	78.2
1000	-276171.9	2583	79.3
3000	-273591.3	7681	79.1
5000	-272766.8	12794	79.0

表 2 MH アルゴリズムの反復回数と対数尤度，構文解析精度の関係．

文法モデル	(b_1, b_2)	＃基本木（＃補助木）	F 値
文脈自由文法	-	5957 (0)	65.0
木置換文法（$a_X = 0$ for all X）	-	9135 (0)	77.9
木挿入文法（提案モデル）	(1,1)	7928 (3)	**79.3**
	(100,1)	7462 (16)	75.4
	(100,100)	8057 (15)	78.0
Berkley Parser[13]		-	77.9
木置換文法（Cohn ら [3] の結果）		-	78.4

表 3 実験結果（MH アルゴリズムの反復 1000 回）．“ ＃基本木 ” は学習後に得られた基本木が何種類であるかを表し，“ ＃補助木 ” は得られた基本木の中で補助木が何種類であるかを表す．

(NP (NP) (ADVP (RB aloft)))
(N̄P (N̄P) (ADVP (RB respectively)))
(P̄P (P̄P) (, ,))
(FR̄AG (, ,) (FR̄AG))
(V̄P (V̄P) (RB then))
(V̄P (V̄P) (RB not))
(Q̄P (Q̄P) (IN of))
(S̄ (S̄) (: ;))
(S̄ (S̄) (RB so))

表 4 獲得された補助木の例．非終端記号の上線は，構文木の二分化によって生成されたノードであることを表す．また，“ NP ” は名詞句，“ ADVP ” は副詞句，“ RB ” は副詞，“ PP ” は前置詞句，“ FRAG ” はフラグメント，“ VP ” は動詞句，“ QP ” は数量詞，“ S ” は文全体を表す非終端記号である．

4.2 結果

構文木データを用いて提案モデルの評価を行った．まず，MH アルゴリズムの反復回数と対数尤度，構文解析精度の関係を表 2 に示す．反復回数を増やしていくと提案モデルの対数尤度は増加したが，構文解析精度は反復回数が 1000 回より多ければほぼ同等の結果であった．また，1 反復にかかる計算時間は約 2.5 秒で，反復回数が増えてもほぼ変化がない．したがって，本データと同じような規模のデータでは，現実時間内に提案モデルの学習を完了することができる．

次に，本データを用いて提案モデルの詳細な評価を行っ

[*1] http://nlp.cs.nyu.edu/evalb/

た．表 3 に結果を示す．表中の木置換文法の結果は，提案モデルの挿入確率 a_X を全ての非終端記号 X において 0 に設定することにより得た．また，(b_1, b_2) は提案モデルにおける挿入操作の起こりやすさを調節するベータ分布のパラメータである．b_1 に大きな値を設定すると補助木による挿入操作が起こる確率を高め，b_2 に大きな値を設定すると挿入操作を抑制する働きがある．実際，$b_1 = 100$ のときに補助木設定した場合は，$b_1 = 1$ と比較して獲得された補助木の数が増加していることがわかる．提案モデルでは，獲得される基本木の数や構文解析精度が (b_1, b_2) の値に影響されやすいことがわかる．

木置換文法と木挿入文法のモデルは，文脈自由文法のモデルと比較して構文解析の精度が極めて高いことがわかる．文脈自由文法は高さが 1 の部分木のみを使用して構文木を生成するのに対して，木置換文法と木挿入文法では任意の大きさの部分木を使用できる．したがって，これらのモデルでは頻出する構文パターンを一つの部分木として学習し，高い確率を付与できる．その結果，木置換文法や木挿入文法は文脈自由文法よりも基本木の数は増加するが，より高精度を実現できていると考えられる．

木置換文法と木挿入文法を比較すると，木挿入文法は適切な (b_1, b_2) の値を設定すれば，木置換文法よりも約 1.4 ポイントの構文解析精度向上が確認できた．また，基本木の数を比較すると，木挿入文法は木置換文法よりも約 18 %少なかった．したがって，補助木による挿入操作が汎用性の高い構文パターンの獲得に寄与していると考えられる．実際に獲得された補助木の例は後の実験結果で示す．

最後に，提案モデルと既存モデルとを比較した．提案モデルは，現在標準的に使用されている Berkley Parser[13] や，Cohn らの木置換文法モデル [3] よりも高精度であることがわかる．したがって，本手法により，本データと同等の規模のデータでは非常に高精度な構文解析器が得られることが期待できる．

5. 結論

我々は，Pitman-Yor 過程に基づく木挿入文法の確率モデルを構築し，構文木コーパスから統計的に基本木を獲得する学習法を提案した．提案モデルは，可算無限次元の確率分布を表現するために Pitman-Yor 過程を利用し，挿入操作を含む確率モデルの効率的な学習のために木挿入文法分解法を新たに考案した．提案手法を英語の構文木コーパスに適用したところ，提案モデルは木置換文法のモデルと比較して約 18 %コンパクトでありながら，文脈自由文法や木置換文法と比較して構文解析精度の向上を実現した．

参考文献

[1] Chen, J., Bangalore, S. and Vijay-Shanker, K.: Automated extraction of Tree-Adjoining Grammars from treebanks, *Natural Language Engineering*, Vol. 12, No. 03, pp. 251–299 (2006).

[2] Chiang, D.: *Statistical Parsing with an Automatically Extracted Tree Adjoining Grammar*, chapter 16, pp. 299–316, CSLI Publications (2003).

[3] Cohn, T. and Blunsom, P.: Blocked Inference in Bayesian Tree Substitution Grammars, *In Proc. of ACL*, Uppsala, Sweden, pp. 225–230 (2010).

[4] Cohn, T., Goldwater, S. and Blunsom, P.: Inducing Compact but Accurate Tree-Substitution Grammars, *In Proc. of HLT-NAACL*, Boulder, Colorado, Association for Computational Linguistics, pp. 548–556 (2009).

[5] Collins, M.: Head-Driven Statistical Models for Natural Language Parsing, *Computational Linguistics*, Vol. 29, No. 4, pp. 589–637 (2003).

[6] DeNeefe, S. and Knight, K.: Synchronous Tree Adjoining Machine Translation, *In Proc. of EMNLP*, pp. 727–736 (2009).

[7] DeNeefe, S., Knight, K. and Vogler, H.: A decoder for probabilistic synchronous tree insertion grammars, *In Proc. of the 2010 Workshop on Applications of Tree Automata in Natural Language Processing*, pp. 10–18 (2010).

[8] Johnson, M. and Goldwater, S.: Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars, *In Proc. of HLT-NAACL*, pp. 317–325 (2009).

[9] Johnson, M., Griffiths, T. and Goldwater, S.: Bayesian Inference for PCFGs via Markov Chain Monte Carlo, *In Proc. of HLT-NAACL*, pp. 139–146 (2007).

[10] Lari, K. and Young, S.: Applications of stochastic context-free grammars using the inside-outside algorithm, *Computer Speech & Language*, Vol. 5, No. 3, pp. 237–257 (1991).

[11] Marcus, M. P., Santorini, B. and Marcinkiewicz, M. A.: Building a Large Annotated Corpus of English: The Penn Treebank, *Computational Linguistics*, Vol. 19, No. 2, pp. 313–330 (1993).

[12] Matsuzaki, T., Miyao, Y. and Tsujii, J.: Probabilistic CFG with latent annotations, *In Proc. of ACL*, pp. 75–82 (2005).

[13] Petrov, S., Barrett, L., Thibaux, R. and Klein, D.: Learning Accurate, Compact, and Interpretable Tree Annotation, *In Proc. of ICCL-ACL*, pp. 433–440 (2006).

[14] Pitman, J. and Yor, M.: The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator, *The Annals of Probability*, Vol. 25, No. 2, pp. 855–900 (1997).

[15] Post, M. and Gildea, D.: Bayesian Learning of a Tree Substitution Grammar, *In Proc. of the ACL-IJCNLP*, pp. 45–48 (2009).

[16] Schabes, Y. and Waters, R.: Tree insertion grammar: a cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced, *Fuzzy Sets and Systems*, Vol. 76, No. 3, pp. 309–317 (1995).

[17] Teh, Y. W.: A Hierarchical Bayesian Language Model based on Pitman-Yor Processes, *In Proc. of ICCL-ACL*, pp. 985–992 (2006).

[18] Xia, F.: Extracting tree adjoining grammars from bracketed corpora, *In Proc. of NLPRS*, pp. 398–403 (1999).