シンボル細分化を適用した階層 Pitman-Yor 過程に基づく木置換文法獲得法と構文解析への応用

進藤 裕之 †　宮尾 祐介 ‡　藤野 昭典 †　　永田 昌明 †
†NTT コミュニケーション科学基礎研究所
‡ 国立情報学研究所
shindo.hiroyuki@lab.ntt.co.jp
yusuke@nii.ac.jp
{fujino.akinori, nagata.masaaki}@lab.ntt.co.jp

1　はじめに

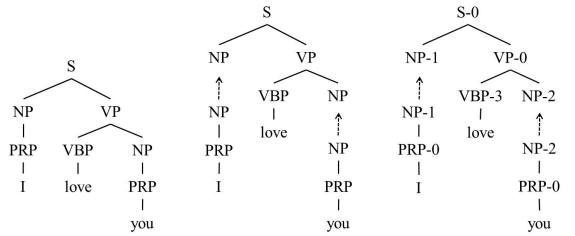
木置換文法（TSG: Tree Substitution Grammar）は，近年盛んに研究されている文法モデルの一つである [1, 4, 11]．TSG は CFG（文脈自由文法）の拡張であり，任意の大きさ（サイズ）の部分木を結合していくことにより構文木を生成するモデルである．TSG で用いられる規則（部分木）を基本木と呼び，非終端シンボルでラベル付けされている葉ノードに対して，新たな基本木が結合する操作を置換操作と呼ぶ．図 1(a) に構文木の例を，図 1(b) に TSG の導出過程の例をそれぞれ示す．TSG の基本木は，述語項構造やイディオムといった文法的なパターンを明示的に表現できるという利点があり，構文解析だけでなく自動要約や言語モデルへの応用例がある．

既存の CFG や TSG に基づく構文解析の問題点として，学習データとして用いられる人手で構築された構文木コーパス（例えば Penn Treebank）では，構文木の各ノードに付与されたタグ（シンボル）の粒度が粗いという点が挙げられる．例えば，NP（名詞句）は動詞の主語となる場合と目的語になる場合とで，子ノードになり得るシンボルの分布が異なる．このとき，CFG や TSG では双方の NP を区別せずに扱うため，構文木コーパスから獲得された基本木を用いて構文解析を行った場合，解析に多くの曖昧性が生じ，その結果，高精度化を実現できない．TSG は CFG と異なり，サイズの大きな部分木を利用できるため，例えば NP ノードとその子・孫ノードを一つの基本木として扱うことで前述の問題をある程度解消することが可能であるが，置換操作が起こるノード（以降，置換ノードと呼ぶ）では CFG と同様に文脈自由を仮定しているため，上記の例では双方の NP を区別できず根本的な解決には至らない．実際，既存の TSG モデルに基づく構文解析器は，単純な CFG モデルの構文解析器よりも高精度であるが，最先端の構文解析器と比較すると精度が低いことが報告されている [4]．

一方，近年の高性能な構文解析器は，CFG にシンボル細分化技術を適用した手法に基づいているものが多い．シンボル細分化とは，構文木の非終端シンボルをいくつかのサブカテゴリに分割する手法の総称であり，粗いタグ付けの構文木コーパスであっても，周辺ノードの情報を上手く利用してターゲットとなるシンボルを細分化することにより，構文木から得られる文脈情報を含んだ基本木を獲得できる．細分化の方法は様々なものが提案されており，例えば主辞となる単語の情報によって細分化するもの [5] や，統計モデルを用いて自動的にいくつかのクラスタに細分化するもの [10] などがある．

本研究では，シンボル細分化を適用した木置換文法（SR-TSG: Symbol-Refined TSG）を提案する．既存の TSG と異なり，SR-TSG では学習データとして与えられる構文木コーパスから，自動的にシンボルが細分化された基本木が獲得される．図 1(c) に SR-TSG の導出過程の例を示す．細分化されたシンボルは，例えば主語となる NP や目的語となる NP の違いを捉えることができるので，粗いタグ付けの構文木コーパスでもモデルをデータに適合させることができる．従来より，TSG とシンボル細分化を組み合わせた手法は提案されているが，従来法 [1] ではヒューリスティクスによりシンボル細分化を行なっている．一方，SR-TSG ではシンボル細分化と TSG の獲得が全て構文木コーパスから自動的に実行されるため，提案法はコーパスや言語に依存せず汎用性の高い手法だと言える．構文木コーパスから SR-TSG 規則を学習する場合，限られた量の学習データでは統計的に信頼性の高い確率値を推定することが困難である（データスパースネス問題）．したがって，我々は，階層 Pitman-Yor 過程に基づいたバックオフモデルを新たに考案し，複雑な SR-TSG 基本木の生成確率を，より単純な基本木のモデルから得られる確率値で補正することによりデータスパースネスの問題に対処する．

提案手法は，WSJ English Penn Treebank データ



(a) (b) (c)

図 1 (a) 構文木の例．(b) TSG の導出過程の例．(c) SR-TSG の導出過程の例．ハイフンで結ばれた数字はシンボルの細分化カテゴリを表す．

を用いた構文解析タスクにおいて，92.4% の F 値を達成した．これは，現在広く用いられている Berkeley parser [10] や Charniak parser [2] よりも高精度であり，英語の構文解析器としては state-of-the-art である．

2 SR-TSG

2.1 確率モデル

我々は，三階層（SR-TSG，SR-CFG，RU-CFG）で構成される階層 Pitman-Yor 過程を用いて，SR-TSG の確率モデルを定義する．表 1 に各階層における部分木の例を示す．最上位の階層（SR-TSG）の確率モデルは，シンボルが x_k であるルートノードから生成される基本木 e が Pitman-Yor 過程に従うと仮定して，以下の式でモデル化する．

$$\begin{aligned} e\,|x_k &\sim G_{x_k} \\ G_{x_k} &\sim \mathrm{PYP}\left(d_{x_k}, \theta_{x_k}, P^{\text{sr-tsg}}(\cdot\,|x_k)\right), \end{aligned}$$

ここで，x_k は構文木コーパスに付与されているシンボル x を細分化して得られた k 番目の細分化カテゴリを表す．例えば，x が NP で細分化カテゴリが 0 のとき，x_k は NP_0 となる．$P^{\text{sr-tsg}}(\cdot\,|x_k)$ は基底分布と呼ばれ，e の生成確率におけるスムージングの値を与える確率分布である．d_{x_k} と θ_{x_k} は Pitman-Yor 過程のパラメータで，スムージング値の大きさを調整する働きをする．提案モデルにおいて，基底分布における基本木 e の確率 $P^{\text{sr-tsg}}(e\,|x_k)$ は，e を構成する CFG 規則，すなわちシンボル細分化 *CFG*（*SR-CFG*）の部分木を用いて以下のようにモデル化する．

$$\begin{aligned} P^{\text{sr-tsg}}(e\,|x_k) = &\prod_{f\in F(e)} s_{c_f} \times \prod_{i\in I(e)} (1-s_{c_i}) \\ &\times H\left(\text{cfg-rules}\,(e\,|x_k)\right) \\ \alpha\,|x_k &\sim H_{x_k} \\ H_{x_k} &\sim \mathrm{PYP}\left(d_x, \theta_x, P^{\text{sr-cfg}}(\cdot\,|x_k)\right), \end{aligned}$$

ただし，$F(e)$ は e に含まれる葉ノードの集合で，$I(e)$ は e に含まれる内部ノードの集合である．c_f と

SR-TSG	SR-CFG	RU-CFG

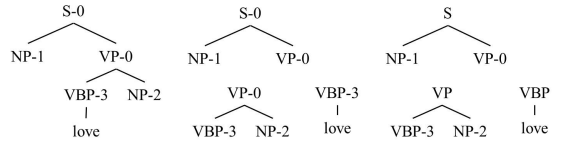


表 1 SR-TSG モデルの各階層における部分木の例．

c_i は，それぞれノード f とノード i のシンボルを表す．s_c はシンボル c のラベルが付与されているノードが葉ノードになる確率，つまり，シンボル c のノードが子ノードの生成を停止する停止確率を表す．SR-CFG は，表 1 に示すように，全てのノードが細分化された CFG 規則である．$\text{cfg-rules}\,(e\,|x_k)$ は，e を構成する全ての SR-CFG 規則を返す関数であり，$x_k \to \alpha$ という形式を取る．各 SR-CFG 規則 $x_k \to \alpha$ のスムージング値は基底分布 H_{x_k} で与えられ，H_{x_k} は Pitman-Yor 過程から生成される．このとき，SR-CFG 規則の基底確率 $P^{\text{sr-cfg}}(\alpha\,|x_k)$ は，SR-CFG のルートシンボルが細分化されていない CFG である，ルート非細分化 *CFG*（*RU-CFG*）を用いて以下のように定義される．

$$\begin{aligned} P^{\text{sr-cfg}}(\alpha\,|x_k) &= I\left(\text{root-unrefine}\,(\alpha\,|x_k)\right) \\ \alpha\,|x &\sim I_x \\ I_x &\sim \mathrm{PYP}\left(d'_x, \theta'_x, P^{\text{ru-cfg}}(\cdot\,|x)\right) \end{aligned}$$

ただし，$\text{root-unrefine}\,(\alpha\,|x_k)$ は，α のルートシンボルを非細分化した RU-CFG 規則を返す関数であり，$x \to \alpha$ という形式を取る．上記の階層モデルと同様に，RU-CFG 規則 $x \to \alpha$ のスムージング値は基底分布 I_x で与えられ，I_x は Pitman-Yor 過程から生成される．最後に，RU-CFG 規則の基底確率を $P^{\text{ru-cfg}}(\alpha\,|x) = \frac{1}{|x\to\cdot|}$ と一様分布で定義する．ただし，$|x \to \cdot|$ は，学習データから得られる，ルートシンボルが x である RU-CFG 規則の数を表す．以上のように，SR-TSG の確率モデルは，SR-TSG から SR-CFG，そして SR-CFG から RU-CFG へと階層的な構造を有しており，複雑な部分木の確率は，より単純な部分木の確率によってバックオフスムージングが行われる．

2.2 学習

構文木コーパスから SR-TSG 規則を獲得するために，我々はマルコフ連鎖モンテカルロ法（MCMC）を用いた学習法を提案する．MCMC は，ある確率分布に従うサンプルを得るための汎用的な手法である．SR-TSG の学習では，構文木コーパス $\mathbf{t}$ が与えられた下で，SR-TSG 導出過程の事後分布 $p\left(\mathbf{e}\,|\mathbf{t}, \mathbf{d}, \boldsymbol{\theta}, \mathbf{s}\right)$ に従うサンプルを得ることが目標である．SR-TSG の確率モデルには，構文木に含まれる各シンボルの細分化カテゴリと，各ノードが置換ノードかどうか，という二種類の潜在変数を含む．全体の学習ステップは，まず始めに細分化カテゴリを学習し，その後に置換ノードを学習

するという段階的なアプローチを取る．

2.2.1 細分化カテゴリの推定

細分化カテゴリの学習には，Petrov ら (2006) によって提案された split-merge 学習法を用いる．この手法は，まず split ステップにおいて，構文木に含まれる全種類のシンボルをそれぞれ二つのサブカテゴリへ分類する．例えば，学習データの NP シンボルは，モデルの尤度が最も高くなるように NP_0 または NP_1 と細分化される．その後の merge ステップでは，細分化された状態と細分化されていない元の状態との尤度を比較し尤度の差を計算する．そして，過分割を抑制するために，尤度の差が小さかった上位 50% のシンボルを merge して細分化前の状態に戻す．この split-merge ステップを指定回数繰り返すことで，最終的な細分化カテゴリが決定される．Petrov らは細分化カテゴリの学習に EM アルゴリズムを用いたが，我々は Pitman-Yor 過程のようなベイズモデルの学習で標準的な手法である MCMC を用いる．

各 split ステップでは，文レベルの Metropolis-Hastings サンプラー（MH サンプラー）[3] と，部分木レベルの Gibbs サンプラーを交互に用いる．文レベルの MH サンプラーでは，構文木ごとに，1) 各シンボルの各細分化カテゴリについて内側確率を計算する，2) トップダウンに SR-TSG の導出過程を一つサンプルする，3) MH テストによってサンプルを受理または棄却する，という三ステップを繰り返し実行することで事後確率の高い解を探索する．MH サンプラーの詳細は Cohn ら (2010) に記述されている．部分木レベルの Gibbs サンプラーでは，SR-TSG 規則の種類ごとに，順番に細分化カテゴリのサンプリングが行われる．例えば，ある MCMC イテレーションにおいて，構文木コーパス中の $\mathrm{S}_0 \rightarrow \mathrm{NP}_1\mathrm{VP}_2$ と細分化されている全ての部分木を $\mathrm{S}_0 \rightarrow \mathrm{NP}_2\mathrm{VP}_0$ と更新する．

文レベルの MH サンプラーでは文ごとに細分化カテゴリを更新するのに対し，部分木レベルの Gibbs サンプラーでは複数の構文木に含まれる同じ種類の部分木の細分化カテゴリを同時に更新する．上記の二種類の MCMC サンプラーを併用することで，より高い尤度を与える局所最適解を探索できる．部分木レベルの Gibbs サンプラーは，Adaptor grammar [7] の学習に用いられる Table label resampling と似ている方法だが，本稿で示した Gibbs サンプラーは，複数の table を一度に更新できるという点において Johnson らの手法とは異なる．

2.2.2 置換ノードの推定

置換ノードの推定には，既存の TSG モデルの学習で用いられている Gibbs サンプラーを用いる [4, 11]．まず，構文木データの全てのノードに二値変数（0 or 1）を割り当てる．変数が 0 の場合はそのノードが置換ノードでない，すなわち基本木の中間ノードであることを表し，1 の場合にはそのノードが置換ノードである，すなわち，基本木のルートノードかつ葉ノードであることを表す．各 Gibbs イテレーションでは，構文木データのノードをランダムな順番で一つずつ辿り，事後分布に基づいて二値変数の値を更新していく．指定回数のイテレーションが完了したとき，二値変数の値が最終的に 1 となっているノードを置換ノードであると推定する．

Model	F1 (small)	F1 (full)
CFG	61.9	63.6
*TSG	77.1	85.0
SR-TSG ($P^{\text{sr-tsg}}$)	73.0	86.4
SR-TSG ($P^{\text{sr-tsg}}$, $P^{\text{sr-cfg}}$)	79.4	89.7
SR-TSG ($P^{\text{sr-tsg}}$, $P^{\text{sr-cfg}}$, $P^{\text{ru-cfg}}$)	**81.7**	**91.1**

表 2　少量学習データと標準学習データでの構文解析精度の比較．*Cohn et al. (2010) のモデルを我々が再実装した．

2.2.3 ハイパーパラメータの推定

我々は，SR-TSG モデルのハイパーパラメータ $\{\mathbf{d}, \boldsymbol{\theta}\}$ を確率変数として扱い，MCMC イテレーション毎に値を更新する．そのために，$d \sim \mathrm{Beta}(1,1)$，$\theta \sim \mathrm{Gamma}(1,1)$ と事前確率を設定し，$\{\mathbf{d}, \boldsymbol{\theta}\}$ も事後確率に基づいて適切な値をサンプリングし，最適な値を推定する．

3 実験

3.1 設定

SR-TSG モデルの評価のために，我々は構文木データとして WSJ English Penn Treebank [8] を用いた．構文木データは，学習セット（セクション 2-21），開発セット（セクション 22），テストセット（セクション 23）に分割した．学習セットは確率モデルの学習に用い，開発セットは停止確率 $\mathbf{s}$ の調整に用いた．確率モデルの学習後，テストセットの文に対して構文解析を行い，構文解析精度を F1 スコアで評価した．また，言語資源の少ない状況を想定して，少量の学習データ（セクション 2）を用意した．以降，少量の学習セット（セクション 2）と標準の学習セット（セクション 2-21）を区別する．学習セットは，Petrov ら (2006) と同じ方法で二分木に変換し，頻度が 5 回以下の単語は接尾辞や接頭辞の情報を用いて 50 種類の未知語の何れかに置き換えた．詳細は Petrov ら (2006) に記述されている．SR-TSG の細分化カテゴリの学習は，標準学習セットでは split-merge ステップを 6 回行い，各シンボルを最大で $2^6 = 64$ に細分化した．同様に，少量学習セットでは，split-merge ステップを 3 回行った．各 split-merge ステップでは MCMC を 1000 イテレーション繰り返し，その後の置換ノードの学習には Gibbs サンプラーを 2000 イテレーション繰り返した．

Model	F1 (≤ 40)	F1 (all)
TSG (no symbol refinement)		
Post and Gildea (2009) [11]	82.6	-
Cohn et al. (2010) [3]	85.4	84.7
TSG with Symbol Refinement		
Bansal et al. (2010) [1]	88.7	88.1
SR-TSG (single)	**91.6**	**91.1**
SR-TSG (multiple)	**92.9**	**92.4**
CFG with Symbol Refinement		
Collins (2003) [5]	88.6	88.2
Petrov and Klein (2007) [10]	90.6	90.1
Petrov (2010) [9]	-	91.8
Discriminative		
Charniak and Johnson (2005) [2]	92.0	91.4
Huang (2008) [6]	-	91.7

表 3　SR-TSG と既存の構文解析器の精度比較．*開発セット (≤ 100) での結果.

3.2 結果と考察

表 2 は，SR-TSG と CFG，TSG の構文解析精度を比較した結果である．SR-TSG は，階層モデルによるバックオフの効果を評価するために，階層を 1 つずつ減らしたモデルでの結果も併せて示した．例えば，SR-TSG ($P^{\text{sr-tsg}}$) はバックオフを行わない一階層のモデルであり，SR-TSG ($P^{\text{sr-tsg}}$, $P^{\text{sr-cfg}}$, $P^{\text{ru-cfg}}$) は 2 節で示した三階層のモデルでの構文解析結果である．表 2 で示されるように，SR-TSG ($P^{\text{sr-tsg}}$, $P^{\text{sr-cfg}}$, $P^{\text{ru-cfg}}$) は学習データの規模に関わらず，従来の CFG や TSG モデルよりも高い構文解析精度を達成した．TSG と比較すると，SR-TSG の基本木は文脈に応じて適切なカテゴリ情報を付与しているため，学習に用いた構文木コーパスの文脈情報を上手くモデルに取り込むことが可能となり，その結果構文解析の高精度化が達成できたと考えられる．また，SR-TSG の構文解析精度は，バックオフを行わない場合と行った場合で大きな性能差がみられた．これは，我々の予期したように，シンボルが細分化された部分木を基本木として用いることによりデータスパースネスの問題が起こっていることを示唆している．我々の提案した階層 Pitman-Yor 過程に基づくスムージングは，SR-TSG より単純な SR-CFG や RU-CFG の部分木の確率を利用して SR-TSG の部分木の確率を補正しているため，この問題を解消できたと考えられる．その結果，バックオフを行わない場合と比較して，我々の三階層のモデルは標準学習セットにおいて 4.7 ポイントの精度向上が見られた.

表 3 は，既存の高精度な構文解析モデルと SR-TSG とを比較した結果である．既存のモデルは，シンボル細分化を行わない通常の TSG，シンボル細分化と CFG または TSG を組み合わせたモデル，識別学習に基づいたモデルとに分類できる．SR-TSG による構文解析精度をさらに高めるため，我々は Petrov(2010) で提案されている複数の学習モデルを組み合わせる手法を SR-TSG に適用した（SR-TSG(multiple)）．これは，独立に学習された複数のモデル（本実験では 16）から k-best（本実験では 100-best）の構文解析候補を出力し，その中で最も複数のモデルにおける尤度の積が高いものを選ぶという手法である．SR-TSG (multiple) は，単一モデルによる SR-TSG（single）と比較して，1.3 ポイントの精度向上が見られたとともに，既存手法と比較して最も良い F1 値（92.4%）を達成した．以上の結果から，我々は構文解析タスクにおける SR-TSG の有効性を確認した.

4 まとめ

本稿では，既存の TSG モデルにシンボル細分化を組み込んだ SR-TSG モデルを提案した．また，データスパースネスの問題に対処するため，階層 Pitman-Yor 過程に基づく新たな確率モデルを構築し，構文木コーパスから MCMC を用いて SR-TSG 規則を獲得する方法を示した．SR-TSG モデルは既存の TSG モデルやシンボル細分化を用いた他の既存手法と比較して最も良い結果であった．今後は，SR-TSG を教師なし学習や係り受け解析へ応用する予定である.

参考文献

[1] Mohit Bansal and Dan Klein. Simple, Accurate Parsing with an All-Fragments Grammar. In *In Proc. of ACL*, number July, pages 1098–1107. Association for Computational Linguistics, 2010.

[2] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *In Proc. of ACL*, 1(June):173–180, 2005.

[3] Trevor Cohn, Phil Blunsom, and Sharon Goldwater. Inducing Tree-Substitution Grammars. *Journal of Machine Learning Research*, 11(1994):3053–3096, 2010.

[4] Trevor Cohn and Mirella Lapata. Sentence Compression as Tree Transduction. *Journal of Artificial Intelligence Research*, 34(1):637–674, 2009.

[5] Michael Collins. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*, 29(4):589–637, 2003.

[6] Liang Huang. Forest Reranking : Discriminative Parsing with Non-Local Features. *In Proc. of ACL*, 19104:0, 2008.

[7] Mark Johnson and Sharon Goldwater. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *In Proc. of HLT-NAACL*, NAACL '09, pages 317–325. Association for Computational Linguistics, 2009.

[8] Mitchell P Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[9] Slav Petrov. Products of Random Latent Variable Grammars. *In Proc. of HLT-NAACL*, (June):19–27, 2010.

[10] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, number April, pages 404–411. Association for Computational Linguistics, 2007.

[11] Matt Post and Daniel Gildea. Bayesian Learning of a Tree Substitution Grammar. In *In Proc. of ACL-IJCNLP*, number August, pages 45–48. Association for Computational Linguistics, Association for Computational Linguistics, 2009.