情報処理学会論文誌 Vol. 49 No. 11 3754–3764 (Nov. 2008)

文レベルの大域的な素性を用いた依存構造解析

中川 哲治[†1]

本稿では，文レベルの大域的な素性を用いた依存構造解析の手法を提案する．多くの既存の統計的依存構造解析手法では，文中の各単語の依存関係は互いに独立であると仮定しており，複数の単語の係り先を考慮するような文レベルの大域的な素性を利用することができないという問題があった．本稿では，依存構造木全体をモデル化する確率分布を考え，ギブスサンプリングを用いて効率的に依存構造解析を行う手法を提案する．提案手法では，依存構造木中の兄弟ノードに関する関係や，子ノードと祖父母ノードに関する関係のような，任意の文レベルの大域的な素性を利用することができる．5 つの言語のコーパスを用いて実験を行った結果，提案手法は既存手法と比較して同程度以上の高い解析精度を持つことを確認した．

Dependency Parsing Using Sentence-level Global Features

Tetsuji Nakagawa[†1]

In this paper, we present a method for dependency parsing using sentence-level global features. Many existing methods for statistical dependency parsing assume independence of the head of each token, and cannot incorporate sentence-level global features which handle heads of multiple tokens. In this paper, we study a probabilistic model of a whole dependency tree, and propose a method for efficient dependency parsing using Gibbs sampling. With the method, arbitrary sentence-level global features in a dependency tree can be used, which include relations between sibling nodes and relations between a child and its grandparent nodes. Experimental results on corpora of five languages showed that the performance of our method was competitive with other state-of-the-art methods.

†1 情報通信研究機構知識創成コミュニケーション研究センター
Knowledge Creating Communication Research Center, National Institute of Information and Communications Technology

1. 背景

単語間の依存関係を推定して文の構造を解析する依存構造解析（係り受け解析）は，重要な言語解析処理の 1 つであり，これまでに様々な方法が研究されている．内元らは，最大エントロピー法を用いた日本語の依存構造解析手法を提案した[20]．McDonald らは交差のない依存構造解析（projective dependency parsing）のための，マージン最大化に基づくオンライン学習アルゴリズムを用いた方法を提案し[9]，さらにその手法を交差のある依存構造解析（non-projective dependency parsing）へも適用した[10]．しかしながらこれらの手法において，文中の各単語（文節）の依存関係は互いに独立であると仮定されており，解析中に利用できる素性はその独立性の条件に合うものに限定されている．高精度な依存構造解析を行うためには，このような個々の単語の係り先を独立に扱って得られるトークンレベルの局所的な素性だけではなく，2 つ以上の単語の係り先を同時に考慮して得られる文レベルの大域的な素性を利用することが有用であると思われる．

Collins らは構文木中の任意の素性を使用して句構造解析を行うために，リランキングに基づく手法を提案し，句構造解析において大域的な素性が有用であることを示した[5]．依存構造解析においても，そのような大域的な素性を利用する方法がいくつか提案されている．McDonald らは二次の素性（second-order feature）を利用することができる方法を提案し[11]，Riedel らは言語学的な制約を利用することができる方法を提案したが[14]，そのような大域的な素性を用いることにより解析精度を大きく向上させられることが確認されている．本稿ではこれらのアプローチとは異なった，ギブスサンプリングを用いて大域的な素性を利用する方法を提案する．なお，提案手法は解析対象となる言語には依存せず，依存構造木が交差を含む場合も含まない場合も適用可能である．

以下，2 章ではギブスサンプリングを用いた依存構造解析手法と使用した素性について説明し，3 章では実験結果を報告する．4 章では関連研究について議論し，5 章で結論を述べる．

2. 手法

依存構造解析[⋆1]は，与えられた文に対してその依存構造（係り受け関係）を同定するタス

⋆1 日本語においては，文節間の修飾・被修飾関係を解析する観点から「係り受け解析」という言葉が類似の処理を表すのに使用されるが，本稿では「依存構造解析」と特に区別しないことにする．また日本語の係り受け解析においては，後述する構文構造を表すグラフにおいて，逆向きの矢印を使用することが多い．

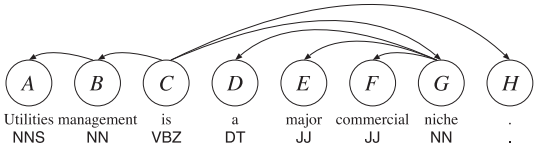

図 1　依存構造木の例
Fig. 1　Example of a dependency tree.

クである[6),12)]．依存構造は，**図 1** の例で示されるような依存構造木により表現することができる．この図において，グラフ中の各ノードはトークン[⋆1]を表し，トークン間の依存関係は係り先（head）から係り元（dependent）への矢印により示されている．各トークンはただ 1 つの係り先を持つが，依存構造木中のルートノードであるトークンは例外であり，そのようなトークンは係り先を持たない．依存構造木は多くの場合交差するエッジを持たないが，チェコ語等の語順が比較的自由な言語では依存構造木中に交差するエッジを持つことがある．以下の節では，依存構造解析のための確率モデルと，それを用いた解析方法，パラメータ推定方法，および使用した素性について説明する．

2.1　確率モデル

以下の説明では，入力文を $\mathbf{w}$ で表すことにし，$|\mathbf{w}|$ は文中のトークン数を，w_t は文中の t 番目のトークンを表すことにする：

$$\mathbf{w} = w_1 \cdots w_{|\mathbf{w}|}.$$

先頭から t 番目のトークン w_t の係り先のインデックス（何番目のトークンに係るか）を h_t で表すことにし，$\mathbf{h}$ は h_t の列を表すものとする．各トークンは自分自身が係り先になることはない．文中のルートノードは係り先を持たないが，そのようなルートノードの係り先のインデックスは 0 として考える：

$$\mathbf{h} = h_1 \cdots h_{|\mathbf{w}|},$$
$$h_t \in \{0, 1, \cdots, |\mathbf{w}|\} \setminus \{t\} \quad (t = 1, \cdots, |\mathbf{w}|).$$

このような定義のもとで，入力文 $\mathbf{w}$ に対する依存構造解析は，その文中のトークンに対する係り先 $\mathbf{h}$ を決定するタスクと考えることができる．

本研究では，入力文 $\mathbf{w}$ に対する依存構造 $\mathbf{h}$ の条件付き確率分布を，exponential model を用いて次のように定義する：

$$P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w}) = \frac{1}{Z_{\Lambda,\mathrm{M}}(\mathbf{w})} Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w}) \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(\mathbf{w}, \mathbf{h})\right\}, \tag{1}$$

$$Z_{\Lambda,\mathrm{M}}(\mathbf{w}) = \sum_{\mathbf{h}' \in \mathcal{H}(\mathbf{w})} Q_{\mathrm{M}}(\mathbf{h}'|\mathbf{w}) \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(\mathbf{w}, \mathbf{h}')\right\}. \tag{2}$$

ここで，$Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w})$ は初期分布，$f_k(\mathbf{w}, \mathbf{h})$ は k 番目の素性関数，K は素性関数の数，λ_k は k 番目の素性の重みである．$\mathcal{H}(\mathbf{w})$ は，入力文 $\mathbf{w}$ に対するすべての可能な係り先の組合せからなる集合である．本研究において $\mathcal{H}(\mathbf{w})$ は，すべての可能なグラフからなる集合（$|\mathbf{w}|$ 個のトークンがそれぞれ $|\mathbf{w}|$ 個の係り先の候補を持つため，集合の要素は $|\mathbf{w}|^{|\mathbf{w}|}$ 個存在する）として考える[⋆2]．Rosenfeld らは，文中の任意の素性を利用することができる whole-sentence language model を構築するために，このような exponential model を使用した[15)]．この確率モデルは最大エントロピーモデルを一般化したものであり，初期分布を一様分布とした場合に最大エントロピーモデルと等しくなる．ある訓練データに対するこの exponential model の最尤解は，素性の期待値がその訓練データにおける経験的期待値と等しいという制約のもとで，Kullback-Leibler 距離が初期分布に最も近いような分布と等しいことが知られている[13),15)]．式 (1) の確率分布 $P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w})$ は，入力文 $\mathbf{w}$ を条件とした文中のトークンの係り先の同時分布であり，この分布は文中のすべての係り受け関係をモデル化するため「文単位のモデル」と呼ぶことにする．素性関数 $f_k(\mathbf{w}, \mathbf{h})$ は係り先 $\mathbf{h}$ を持つ文 $\mathbf{w}$ 全体に関して定義され，各トークンの係り先に対して独立性を仮定せずに依存構造木中の任意の情報を利用することができるため，$f_k(\mathbf{w}, \mathbf{h})$ を「文単位の大域的な素性」と呼ぶことにする．この大域的な素性については，2.5 節で詳しく説明する．

初期分布 $Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w})$ は，各トークンの係り先の分布 $q^t_{\mathrm{M}}(h|\mathbf{w})$ の積として次のように定義する：

⋆1 本稿では，文を構成する基本的な単位を「トークン」と呼ぶことにする．依存構造解析において，日本語では文節が基本的な単位となることが多いが，英語等では単語が基本的な単位として使用されることが多い．なお，各トークンには品詞が付与されているものと仮定する．

⋆2 このように定義した $\mathcal{H}(\mathbf{w})$ は，交差のない依存構造木や交差のある依存構造木だけではなく，サイクルが存在するような依存構造木としては不適切なグラフも含むことになる．このように不必要に大きい確率空間を考慮することにより依存構造解析の精度が低下する可能性があるが，本研究では複雑な制約を課さずにサンプリング等の計算を効率的に行うために，このような定義を行った．

$$Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w}) = \prod_{t=1}^{|\mathbf{w}|} q_{\mathrm{M}}^{t}(h_t|\mathbf{w}). \tag{3}$$

ここで $q_{\mathrm{M}}^{t}(h|\mathbf{w})$ は，文 $\mathbf{w}$ が与えられた場合に t 番目のトークンの係り先が h である確率であり，次のように最大エントロピーモデルを用いて定義する：

$$q_{\mathrm{M}}^{t}(h|\mathbf{w}) = \frac{1}{Y_{\mathrm{M}}(\mathbf{w},t)} \exp\left\{\sum_{l=1}^{L} \mu_l g_l(\mathbf{w},t,h)\right\}, \tag{4}$$

$$Y_{\mathrm{M}}(\mathbf{w},t) = \sum_{\substack{h'=0 \\ h'\neq t}}^{|\mathbf{w}|} \exp\left\{\sum_{l=1}^{L} \mu_l g_l(\mathbf{w},t,h')\right\}. \tag{5}$$

ここで，$g_l(\mathbf{w},t,h)$ は l 番目の素性関数であり，L は素性関数の数であり，μ_l は l 番目の素性の重みである．$q_{\mathrm{M}}^{t}(h|\mathbf{w})$ は単一のトークン（t 番目のトークン）に関する係り先のモデルであり，他のトークンとは独立に計算されるため，$q_{\mathrm{M}}^{t}(h|\mathbf{w})$ を「トークン単位のモデル」と呼び，$g_l(\mathbf{w},t,h)$ を「トークン単位の局所的な素性」と呼ぶことにする．この局所的な素性については，2.4 節で説明する．

2.2　ギブスサンプリングを用いた解析

文 $\mathbf{w}$ と，文単位のモデルのパラメータ $\Lambda = \{\lambda_1, \cdots, \lambda_K\}$ と，トークン単位のモデルのパラメータ $\mathrm{M} = \{\mu_1, \cdots, \mu_L\}$ が与えられた場合に，依存構造解析結果 $\hat{\mathbf{h}}$ を求める方法を考えることにする．何らかの基準に基づいて最適な解を探索する場合，可能な解の候補が非常に多いため，厳密解を求めるのは困難である．文単位のモデルでは文単位の任意の素性を扱うことを考えており，変数間の独立性は仮定できないため，効率的な動的計画法等のアルゴリズムを用いることはできない．そこで本研究では，大規模な確率分布に対して，その系の状態を効率的に計算することができるギブスサンプリングを用いて依存構造解析を行うことを考える．ギブスサンプリング自体は確率分布からサンプルを生成するための手法であり，最適化を行うための手法ではない．そこで，サンプリングにより得られたサンプルから周辺確率を計算し，その周辺確率の積を最大化する解を最適な解と定義して，最大全域木の探索手法を利用して依存構造解析を行う．以下では，その具体的な方法を説明する．

次のような各トークンの係り先の周辺確率の積を最大化するような依存構造木を，最適な依存構造木と定義して求めることにする：

$$\hat{\mathbf{h}} = \operatorname*{argmax}_{\mathbf{h}\in\mathcal{T}(\mathbf{w})} \prod_{t=1}^{|\mathbf{w}|} P_t(h_t|\mathbf{w}). \tag{6}$$

ここで $P_t(h|\mathbf{w})$ は，文 $\mathbf{w}$ が与えられた場合における t 番目のトークンの係り先の周辺確率であり，$\mathcal{T}(\mathbf{w})$ は入力文 $\mathbf{w}$ に対する依存構造木として適切なグラフからなる集合である．この解を求める場合に，単純に $P_t(h|\mathbf{w})$ を最大化するように各トークンの係り先を選んでしまうと，サイクルを含むような依存構造木としては不適切な解が得られる可能性がある．そこで周辺確率の積を最大化し，なおかつ依存構造木として適切な解を求めるために，最大全域木（maximum spanning tree; MST）の探索手法を利用することにする．McDonald らは，交差のない依存構造解析の問題を最大全域木問題として解いた[9]．彼らは，文中の各トークン間の係り受け関係を，各トークンをノードとするグラフ中の有向エッジと見なし，トークン間の係りやすさを各エッジに対するスコアとして定義し，このグラフ中の最大全域木（グラフ中のすべてのノードを含む木で，エッジのスコアの和が最大であるもの）を探索することにより最適な依存構造木を求めた．彼らは交差のない最大全域木を探索するために，Eisner のアルゴリズム[7] を利用した．この手法は，交差のある最大全域木を探索するために Chu-Liu-Edmonds（CLE）のアルゴリズムを使用する方法[10] へと拡張された．ここでは，我々もこの最大全域木の探索に基づいて依存構造解析を行う枠組みを利用することにする．親ノード（係り先）h から子ノード（係り元）t へのエッジのスコアを $s(h,t)$ で表すことにする．そして，$s(h,t)$ を次のように定義する：

$$s(h,t) = \log P_t(h|\mathbf{w}). \tag{7}$$

上式において周辺確率の対数をスコアとしている理由は，最大全域木探索アルゴリズムはエッジのスコアの和を最大化するが，ここでは周辺確率の積を最大化したいからである．以上のようにして定義されたスコアと Eisner アルゴリズムを用いることで，最適な交差のない依存構造木を求めることができる．また，CLE アルゴリズムを用いることで，最適な交差のある依存構造木を求めることができる．

次に，周辺確率 $P_t(h|\mathbf{w})$ の計算方法を説明する．この確率は，確率分布 $P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w})$ に従う R 個のサンプル $\{\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(R)}\}$ が存在した場合に，次のようにして近似的に計算することができる：

$$P_t(h|\mathbf{w}) = \sum_{\substack{h_1,\cdots,h_{t-1},h_{t+1},\cdots,h_{|\mathbf{w}|} \\ h_t=h}} P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w}),$$

```
# 入力:
#     R: 生成するサンプルの数
#     w: 入力文
# 出力:
#     {h^(1), ..., h^(R)}: P(h|w) に従って生成されたサンプル

1  h^(0) を初期化する
2  for r := 1 to R
3    for t := 1 to |w|
4      条件付き確率分布
       P(h_t|w, h_1^(r), ..., h_{t-1}^(r), h_{t+1}^(r-1), ..., h_{|w|}^(r-1)) に
       従って発生させた値を h_t^(r) とする
```

図 2　ギブスサンプリング
Fig. 2　Gibbs sampling.

$$= \sum_{\mathbf{h}} P_{\Lambda,\mathrm{M}}(\mathbf{h}|\mathbf{w})\delta(h,h_t),$$
$$\simeq \frac{1}{R}\sum_{r=1}^{R}\delta(h,h_t^{(r)}). \tag{8}$$

ここで，$\delta(i,j)$ はクロネッカーのデルタである．サンプル $\{\mathbf{h}^{(1)},\cdots,\mathbf{h}^{(R)}\}$ を生成するために，ギブスサンプリングを利用することができる．ギブスサンプリングはマルコフ連鎖モンテカルロ法の一種であり，確率変数間に複雑な依存関係を持つような高次元の確率分布から，効率的にサンプルを生成することができる手法である[18]．**図 2** にそのアルゴリズムを示す．このアルゴリズムでは，まず最初に初期状態 $\mathbf{h}^{(0)}$ を決める．そして，ある 1 つの確率変数について，それ以外の確率変数をすべて固定した条件付き確率からのサンプリングを行い値を更新するという手続きを繰り返して，新しいサンプルを生成していく．ギブスサンプリングにより生成されるサンプルの分布は，もとの確率の定常分布へ収束することが知られている．本研究では，確率分布 $Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w})$ を最大化する係り先を初期状態 $\mathbf{h}^{(0)}$ とした．

ギブスサンプリングでは多数のサンプルを繰り返し生成する必要があるが，サンプリングを行うたびに文中の各トークンに対するすべての係り先の候補に対して確率を計算するため，計算量が比較的大きい．そこで，計算時間を削減するためにいくつかの手法を用いた．まず，$c(pos_d,pos_h,dir)$ という関数を考える．この関数は，訓練データ中で品詞 pos_d を持ち，その係り先の品詞が pos_h であり，係り先から係り元への方向（左または右）が dir であるようなトークンの数である．ギブスサンプリングにおいて，あるトークンに対する係り先の候補の確率を計算する際に，もしその係り元と係り先に対するこの関数の値が 0 であるならば，そのような係り先は考慮しないことにする．さらに，トークン単位のモデルにより計算された確率の値がある閾値 θ よりも小さい場合には，その係り先の候補は無視することにする．なぜならば，もしトークン単位のモデルによって計算された確率が十分に小さければ，文単位のモデルによって計算される確率も無視できる程度に小さいと考えられるからである．本研究では，θ の値は 0.5%とした．なお，ギブスサンプリングにおける確率の計算（図 2 の 4 行目）において，文中のあらゆる大域的な素性を使用して計算を行う必要はない．現在考慮している変数 h_t に関する大域的な素性しか確率には影響を及ぼさないため，そのような素性のみを使用して計算を行えばよい．

以上のように提案手法では，各エッジに対して互いに独立に定義されるスコア $s(h,t)$ を使用して正解と思われる依存構造木を求めるが，このスコアは文全体を考慮してギブスサンプリングを用いて計算されるものであり，スコアの計算には文全体の情報を利用することができる．

2.3　モデルパラメータの推定

N 個の事例からなる学習データ $\{\langle\mathbf{w}^1,\mathbf{h}^1\rangle,\cdots,\langle\mathbf{w}^N,\mathbf{h}^N\rangle\}$ が与えられた場合に，モデルのパラメータを推定することを考える．

はじめに，トークン単位のモデルのパラメータ $\mathrm{M}=\{\mu_1,\cdots,\mu_L\}$ を推定する方法について説明する．ここでは，Gaussian prior を用いて最大事後確率（maximum a posteriori; MAP）推定を行う．次のような目的関数 $\mathcal{M}$ を定義し，$\mathcal{M}$ の値を最大化する最適解を見つけることにする：

$$\mathcal{M} = \log\prod_{n=1}^{N} Q_{\mathrm{M}}(\mathbf{h}^n|\mathbf{w}^n) - \frac{1}{2\sigma^2}\sum_{l=1}^{L}\mu_l^2,$$
$$= \sum_{n=1}^{N}\sum_{t=1}^{|\mathbf{w}^n|}\left[-\log Y_{\mathrm{M}}(\mathbf{w}^n,t) + \sum_{l=1}^{L}\mu_l g_l(\mathbf{w}^n,t,h_t^n)\right] - \frac{1}{2\sigma^2}\sum_{l=1}^{L}\mu_l^2. \tag{9}$$

ここで，σ は Gaussian prior に関するハイパーパラメータである．目的関数の偏微分は次のようになる：

$$\frac{\partial\mathcal{M}}{\partial\mu_l} = \sum_{n=1}^{N}\sum_{t=1}^{|\mathbf{w}^n|}\left[-\frac{\partial}{\partial\mu_l}\log Y_{\mathrm{M}}(\mathbf{w}^n,t) + g_l(\mathbf{w}^n,t,h_t^n)\right] - \frac{1}{\sigma^2}\mu_l,$$

$$= \sum_{n=1}^{N} \sum_{t=1}^{|\mathbf{w}^n|} \left[- \sum_{\substack{h'=0 \\ h' \neq t}}^{|\mathbf{w}^n|} q_{\mathrm{M}}^{t}(h'|\mathbf{w}^n) g_l(\mathbf{w}^n, t, h') + g_l(\mathbf{w}^n, t, h_t^n) \right] - \frac{1}{\sigma^2}\mu_l. \quad (10)$$

最適なパラメータ M は，上述の $\mathcal{M}$ と $\partial\mathcal{M}/\partial\mu_l$ から，準ニュートン法の一種である L-BFGS 法を用いて求めることができる．

次に，すでにトークン単位のモデルのパラメータ M が得られているという条件のもとで，文単位のモデルのパラメータ $\Lambda = \{\lambda_1, \cdots, \lambda_K\}$ を推定する方法を説明する．MAP 推定を用いることにして，目的関数 $\mathcal{L}$ を次のように定義する：

$$\begin{aligned}\mathcal{L} &= \log \prod_{n=1}^{N} P_{\Lambda,\mathrm{M}}(\mathbf{h}^n|\mathbf{w}^n) - \frac{1}{2\sigma'^2}\sum_{k=1}^{K}\lambda_k^2, \\ &= \sum_{n=1}^{N}\left[-\log Z_{\Lambda,\mathrm{M}}(\mathbf{w}^n) + \sum_{k=1}^{K}\lambda_k f_k(\mathbf{w}^n, \mathbf{h}^n) \right] \\ &\quad -\frac{1}{2\sigma'^2}\sum_{k=1}^{K}\lambda_k^2 + \sum_{n=1}^{N}\log Q_{\mathrm{M}}(\mathbf{h}^n|\mathbf{w}^n). \end{aligned} \quad (11)$$

ここで，σ' は Gaussian prior に関するハイパーパラメータである．この目的関数の偏微分は次のようになる：

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\lambda_k} &= \sum_{n=1}^{N}\left[-\frac{\partial}{\partial\lambda_k}\log Z_{\Lambda,\mathrm{M}}(\mathbf{w}^n) + f_k(\mathbf{w}^n, \mathbf{h}^n) \right] - \frac{1}{\sigma'^2}\lambda_k, \\ &= \sum_{n=1}^{N}\left[-\sum_{\mathbf{h}'\in\mathcal{H}(\mathbf{w}^n)} P_{\Lambda,\mathrm{M}}(\mathbf{h}'|\mathbf{w}^n) f_k(\mathbf{w}^n, \mathbf{h}') + f_k(\mathbf{w}^n, \mathbf{h}^n) \right] - \frac{1}{\sigma'^2}\lambda_k. \end{aligned} \quad (12)$$

最適なパラメータ Λ は準ニュートン法により求めることができるが，$\mathcal{L}$ と $\partial\mathcal{L}/\partial\lambda_k$ の計算には，すべての可能な係り先の組合せについての和が含まれるため，厳密な計算を行うのは困難である．そこで無作為に生成されたサンプルを用いて，静的なモンテカルロ法により近似的にこれらの値を計算することにする[1),13),15),⋆1]．訓練データ中の n 番目の文 $\mathbf{w}^n$ に対して，S 個のサンプル $\{\mathbf{h}^{n(1)}, \cdots, \mathbf{h}^{n(S)}\}$ が確率分布 $Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w})$ から生成されるとする．確率分布 $Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w})$ の変数間には依存関係が存在しないため，単純に文中の各トークンの係り先を独立に選ぶことによって $Q_{\mathrm{M}}(\mathbf{h}|\mathbf{w})$ から容易にサンプルを生成することができる．これらのサンプルを用いると，係り先の組合せについての和を含む項は次のように近似的に計算できる：

$$\begin{aligned}\log Z_{\Lambda,\mathrm{M}}(\mathbf{w}^n) &= \log \sum_{\mathbf{h}'\in\mathcal{H}(\mathbf{w}^n)} Q_{\mathrm{M}}(\mathbf{h}'|\mathbf{w}^n)\exp\left\{\sum_{k=1}^{K}\lambda_k f_k(\mathbf{w}^n, \mathbf{h}')\right\}, \\ &\simeq \log\frac{1}{S}\sum_{s=1}^{S}\exp\left\{\sum_{k=1}^{K}\lambda_k f_k(\mathbf{w}^n, \mathbf{h}^{n(s)})\right\}, \end{aligned} \quad (13)$$

$$\begin{aligned}&\sum_{\mathbf{h}'\in\mathcal{H}(\mathbf{w}^n)} P_{\Lambda,\mathrm{M}}(\mathbf{h}'|\mathbf{w}^n) f_k(\mathbf{w}^n, \mathbf{h}') \\ &= \sum_{\mathbf{h}'\in\mathcal{H}(\mathbf{w}^n)} Q_{\mathrm{M}}(\mathbf{h}'|\mathbf{w}^n)\frac{f_k(\mathbf{w}^n, \mathbf{h}')}{Z_{\Lambda,\mathrm{M}}(\mathbf{w}^n)}\exp\left\{\sum_{k'=1}^{K}\lambda_{k'} f_{k'}(\mathbf{w}^n, \mathbf{h}')\right\}, \\ &\simeq \frac{1}{S}\sum_{s=1}^{S}\frac{f_k(\mathbf{w}^n, \mathbf{h}^{n(s)})}{Z_{\Lambda,\mathrm{M}}(\mathbf{w}^n)}\exp\left\{\sum_{k'=1}^{K}\lambda_{k'} f_{k'}(\mathbf{w}^n, \mathbf{h}^{n(s)})\right\}. \end{aligned} \quad (14)$$

2.4　使用した局所的な素性

局所的な素性としては，MSTParser version 0.4.2 [⋆2] で使用されているものと同一の素性を用いた．この素性には，係り元のトークン，係り先のトークン，それらの周辺および間に存在するトークンの，語形や品詞タグが含まれる．係り先のトークンから係り元のトークンへの方向と距離も含まれる．各トークンに対する詳細な形態素情報等が利用できる場合には，それらも利用される．

後述する実験では，訓練データ中に 5 回未満しか出現しなかった素性は削除した．

2.5　使用した大域的な素性

大域的素性として，9 種類の素性を利用した．これらの素性は Collins ら[5)] が使用した素性と似ているが，彼らの素性は句構造解析のために利用されたものであり，依存構造解析のためにここで用いる素性とはやや異なる．以下の説明では，図 1 の依存構造木を例として用いることにする．

2.5.1　Child Unigram+Parent+Grandparent

この素性テンプレートは，次の要素から構成される 4 つ組である：(1) 子ノード，(2) そ

⋆1 本研究では計算時間の観点からこのような静的なモンテカルロ法を用いて計算を行ったが，高次元の確率分布に対してより高い精度で期待値を計算できる方法を利用することも考えられる[18)]．

⋆2 http://sourceforge.net/projects/mstparser/より入手可能．

の親ノード，(3) 親ノードから子ノードへの方向（左「l」または右「r」），(4) 祖父母ノード（もし親ノードがルートノードの場合は特別な記号 ϕ とする）．

実際の素性は，素性テンプレート中の各ノード（トークン）をその語形と品詞に展開して作成する．実験では，5 つ未満の文にしか出現しなかった素性は削除した．これらの手続きは，他の素性についても同様に適用した．

例（図 1）：$\langle A, B, l, C\rangle$，$\langle B, C, l, \phi\rangle$，$\langle D, G, l, C\rangle$，$\langle E, G, l, C\rangle$，$\langle F, G, l, C\rangle$，$\langle G, C, r, \phi\rangle$，$\langle H, C, r, \phi\rangle$．

2.5.2　Child Bigram+Parent

この素性テンプレートは，次の要素から構成される 4 つ組である：(1) 子ノード，(2) その親ノード，(3) 親ノードから子ノードへの方向，(4) 最も近い外側（親ノードとは反対側）の兄弟ノード（該当するノードがない場合は特別な記号 ϕ とする）．この素性テンプレートは，McDonald ら[11] が二次の素性と呼んで使用したものとほぼ同じである．

例（図 1）：$\langle A, B, l, \phi\rangle$，$\langle B, C, l, \phi\rangle$，$\langle D, G, l, \phi\rangle$，$\langle E, G, l, D\rangle$，$\langle F, G, l, E\rangle$，$\langle G, C, r, H\rangle$，$\langle H, C, r, \phi\rangle$．

2.5.3　Child Bigram+Parent+Grandparent

この素性テンプレートは 5 つ組である．最初の 4 つの要素 (1)–(4) は **Child Bigram+Parent** 素性と同じである．残りの 1 つの要素 (5) は祖父母ノードである．

例（図 1）：$\langle A, B, l, \phi, C\rangle$，$\langle B, C, l, \phi, \phi\rangle$，$\langle D, G, l, \phi, C\rangle$，$\langle E, G, l, D, C\rangle$，$\langle F, G, l, E, C\rangle$，$\langle G, C, r, H, \phi\rangle$，$\langle H, C, r, \phi, \phi\rangle$．

2.5.4　Child Trigram+Parent

この素性テンプレートは 5 つ組である．最初の 4 つの要素 (1)–(4) は **Child Bigram+Parent** 素性と同じである．残りの 1 つの要素 (5) は 2 番目に近い外側の兄弟ノードである．

例（図 1）：$\langle A, B, l, \phi, \phi\rangle$，$\langle B, C, l, \phi, \phi\rangle$，$\langle D, G, l, \phi, \phi\rangle$，$\langle E, G, l, D, \phi\rangle$，$\langle F, G, l, E, D\rangle$，$\langle G, C, r, H, \phi\rangle$，$\langle H, C, r, \phi, \phi\rangle$．

2.5.5　Parent+All Children

この素性テンプレートは 1 つ以上の要素から構成される組である．最初の要素 (1) は親ノードであり，残りの要素はそのすべての子ノードである．親ノードの左側にある子ノードと右側にある子ノードは区別する必要があるため，下の例では後者にプライムを付けている．

例（図 1）：$\langle A\rangle$，$\langle B, A\rangle$，$\langle C, B, G', H'\rangle$，$\langle D\rangle$，$\langle E\rangle$，$\langle F\rangle$，$\langle G, D, E, F\rangle$，$\langle H\rangle$．

2.5.6　Parent+All Children+Grandparent

この素性テンプレートは 2 つ以上の要素から構成される組である．最後の要素以外は，**Parent+All Children** 素性と同じである．最後の要素は祖父母ノードである．

例（図 1）：$\langle A, B\rangle$，$\langle B, A, C\rangle$，$\langle C, B, G', H', \phi\rangle$，$\langle D, G\rangle$，$\langle E, G\rangle$，$\langle F, G\rangle$，$\langle G, D, E, F, C\rangle$，$\langle H, C\rangle$．

2.5.7　Child+Ancestor

この素性テンプレートは次の要素から構成される 2 つ組である：(1) 子ノード，(2) その祖先ノードの 1 つ．田村らは，このような情報を用いることにより日本語係り受け解析の精度を改善できたと報告している[19]．

例（図 1）：$\langle A, B\rangle$，$\langle A, C\rangle$，$\langle B, C\rangle$，$\langle D, G\rangle$，$\langle D, C\rangle$，$\langle E, G\rangle$，$\langle E, C\rangle$，$\langle F, G\rangle$，$\langle F, C\rangle$，$\langle G, C\rangle$，$\langle H, C\rangle$．

2.5.8　Acyclic

この素性は 2 つの値のうちのどちらか 1 つを持つ．もし依存構造木がサイクルを含まなければ *true* であり，サイクルを含めば *false* である．

例（図 1）：*true*.

2.5.9　Projective

この素性は 2 つの値のうちのどちらか 1 つを持つ．もし依存構造木が交差を含まなければ *true* であり，交差を含めば *false* である．

例（図 1）：*true*.

3. 実　　験

実験には，Penn Treebank WSJ corpus（英語）と，CoNLL-X shared task のデータセット（デンマーク語，オランダ語，ポルトガル語，スウェーデン語）を使用した．依存構造解析の精度を評価するために，下記の評価指標を用いた：

$$\textbf{係り先精度（DA）} = \frac{\langle\,\text{係り先が正しく推定されたトークン数}\,\rangle}{\langle\,\text{全トークン数}\,\rangle},$$

$$\textbf{ルート精度（RA）} = \frac{\langle\,\text{ルートノードが正しく推定された文数}\,\rangle}{\langle\,\text{文の総数}\,\rangle},$$

$$\textbf{文正解率（CM）} = \frac{\langle\,\text{正しく解析された文数}\,\rangle}{\langle\,\text{文の総数}\,\rangle}.$$

なお，従来研究[2),17)] と同様に句読点は除いて精度を計算した．

表 1　WSJ コーパスでの依存構造解析結果
Table 1　Results of dependency parsing on the WSJ corpus.

手法	DA（%）	RA（%）	CM（%）
提案手法–交差なし（局所素性）	90.6	95.1	34.9
提案手法–交差なし（＋大域素性）	**91.7**	95.3	42.0
提案手法–交差あり（局所素性）	90.3	**95.6**	32.9
提案手法–交差あり（＋大域素性）	91.5	95.3	40.8
MSTParser–交差なし（一次の素性）	91.0	93.8	37.5
MSTParser–交差なし（＋二次の素性）	**91.7**	94.9	**42.6**
MSTParser–交差あり（一次の素性）	90.4	94.2	34.1
MSTParser–交差あり（＋二次の素性）	91.5	94.4	40.6

3.1　WSJ コーパスでの実験結果

Penn Treebank WSJ コーパスを使用して実験を行った．WSJ コーパスは句構造文法のアノテーションを持つが，山田らの主辞規則[17] を用いて依存構造木に変換した．このコーパス中には，交差を持つ依存構造木は含まれない．section 2–21 を訓練用，section 22 をパラメータ調整用，section 23 を評価用に使用した．パラメータ調整用と評価用データに対する品詞タグは，訓練用データで学習した統計的品詞タグ付けシステム[22] を用いて付与した（評価用データに対する品詞タグ付けの精度は 97.1%であった）．ハイパーパラメータ等の値は次のように設定した：$\sigma = 0.25$，$\sigma' = 0.25$，$R = 100$，$S = 100$．局所的素性の数は 14,910,831 個であり，大域的素性の数は 4,946,085 個であった．実験は，Opteron 250 プロセッサと 8 GB のメモリを持つ計算機で行った．計算時にメモリが足りなかったため，学習データの素性ベクトルはディスク上に格納して処理を行った．トークン単位のモデルと文単位のモデルの両方のパラメータを学習するのに要した時間は約 28 時間であり，評価用データに対して交差のある依存構造解析を行うのに要した時間は約 71 分であった．

3.1.1　大域的素性の効果

提案手法を用いて依存構造解析を行った実験結果を**表 1** に示す．この表で，**提案手法–交差なし**は提案手法で Eisner アルゴリズムを利用して交差のない依存構造解析を行った場合の結果を，**提案手法–交差あり**は提案手法で CLE アルゴリズムを利用して交差のある依存構造解析を行った場合の結果を，**（局所素性）**はトークンレベルの局所的な素性のみを用いて文レベルの大域的な素性は用いなかった場合の結果を，**（＋大域素性）**は局所的な素性と大域的な素性の両方を用いた場合の結果をそれぞれ表す．大域的な素性を利用した場合，利用しなかった場合と比較して高い係り先精度（DA）と文正解率（CM）を得た．また，交差のない依存構造解析を行った方が，交差のある依存構造解析を行った場合よりも高い精度が得られている．これは，使用したコーパスには交差を持つ依存構造木は含まれていないため，交差のある依存構造解析を行っても正しく解析できるようになる事例が増えることはなく，むしろ不必要な解候補を多数考慮することにより誤った解析を行う可能性が増えるためと思われる．

表 2　各素性を利用した場合/利用しなかった場合の依存構造解析結果：各素性に対して，その素性のみを使用した場合（追加した場合）と，その素性のみを使用しなかった場合（削除した場合）の結果
Table 2　Results of dependency parsing with/without each feature: Two cases are examined for each feature; one is a case using only the feature and the other is a case using all features other than that one.

素性	DA（%） / CM（%）		素性の数
	追加した場合	削除した場合	
None	90.3 / 32.9	91.5 / 40.8	0
All	91.5 / 40.8	90.3 / 32.9	4,946,085
Child Unigram+Parent+Grandparent	90.7 / 35.5	91.4 / 40.4	587,836
Child Bigram+Parent	90.9 / 37.4	91.4 / 40.6	499,079
Child Bigram+Parent+Grandparent	91.2 / 38.5	91.3 / 40.2	1,835,514
Child Trigram+Parent	91.1 / 38.4	91.4 / 40.4	1,322,725
Parent+All Children	90.6 / 35.5	91.5 / 40.4	104,676
Parent+All Children+Grandparent	90.7 / 35.0	91.5 / 40.8	163,310
Child+Ancestor	90.4 / 33.4	91.4 / 40.4	432,941
Acyclic	90.2 / 32.5	91.3 / 40.7	2
Projective	90.3 / 32.9	91.5 / 40.6	2

個々の大域的な素性の効果を調べるために行った実験の結果を**表 2** に示す．各大域的素性に対して，2 通りの場合について交差のある依存構造解析を行った．1 つは特定の素性のみを使用した場合（**追加した場合**）であり，もう 1 つは特定の素性以外をすべて使用した場合（**削除した場合**）である．各素性を追加した場合，Acyclic と Projective 以外のすべての場合において精度が向上した．Acyclic と Projective の素性を利用することにより，より正解に近い依存構造木のサンプルがギブスサンプリングにより重点的に生成されて解析精度が向上することを期待したが，この実験結果からはこれらの素性の有用性は見られなかった．各素性を削除した場合，精度はあまり低下していない．このことから，本研究で用いた大域的な素性は比較的大きな冗長性を持っていると思われる．

次に，文単位のモデルのパラメータを学習した結果，各素性に対してどのような重みが与えられたかを調べた．一般的に，大きな重みが与えられた素性ほど，そのモデルを用いた分類に大きな影響を与えると考えられる．大域的な素性の中で，1 つも子ノードを持たない，または 2 つ以上の子ノードを持つ前置詞を表す素性は大きな負の値を持っており，ちょうど

1つの子ノードを持つ前置詞を表す素性は大きな正の値を持っていた．前置詞は多くの場合1つの子ノードしか持たないが，そのような性質等が大域的な素性により学習されることによって，精度の向上に寄与したと思われる．

3.1.2　既存手法との比較

他の依存構造解析手法との比較を行うために，既存の依存構造解析器の1つであるMSTParserを用いて実験を行った．MSTParserは交差のない依存構造解析[9]も，交差のある依存構造解析[10]も扱うことができ，また**一次の素性**（トークンレベルの局所的な素性と同一）だけではなく**二次の素性**（文レベルの大域的な素性の1つであるChild Bigram+Parent素性に相当）[11]も利用することができる．

実験結果を表1に示す．大域的な素性を用いた場合の提案手法–交差なしの係り先精度（DA）は，二次の素性を用いた場合のMSTParser–交差なしの精度と等しかった（91.7%）．しかしながら，文正解率（CM）はやや低かった（42.0%）[⋆1]．提案手法が，MSTParserよりも多くの大域的な素性を利用しているにもかかわらず文正解率が低かったことに対する考えられる原因の1つとしては，学習アルゴリズムの違いがあげられる．MSTParserではマージン最大化に基づくオンライン学習アルゴリズムを利用しているが，提案手法ではexponential modelを利用している．特に提案手法におけるトークン単位のモデルは，式(4)で定義されるような個々のトークンごとに局所的に最適化されるモデルであるが，MSTParserでは文全体での大域的な最適化が行われている．MSTParserで使用している一次の素性と，提案手法で使用している局所素性は同一であるにもかかわらず，一次の素性を使用したMSTParser–交差なしよりも局所素性を使用した提案手法–交差なしの精度が低いため（表1），トークン単位のモデル自体の性能が低いために大域的素性を使用しても十分な精度が得られなかった可能性がある．別の原因として，提案手法ではモンテカルロ法を用いた近似計算を行っており，これにより十分な精度が得られていない可能性も考えられる．

モンテカルロ法で使用するサンプル数を変えた場合の精度を調べるため，学習時に静的モンテカルロ法にて使用するサンプルの数（S）と，解析時にマルコフ連鎖モンテカルロ法（ギブスサンプリング）で使用するサンプルの数（R）を変えた実験を行ったところ，**表3**の結果が得られた．学習時や解析時にモンテカルロ法で使用するサンプル数を増やすと解析精度が向上する傾向が見られるが，$S = 200$，$R = 500$のときに最も高い精度が得られており，この場合はMSTParserに劣らない精度となっている．

表3　学習時に使用するサンプル数（S）と解析時に使用するサンプル数（R）を変えた場合の依存構造解析結果
Table 3　Results of dependency parsing for different numbers of samples.

	DA (%) / CM (%)				
R	S				
	10	20	50	100	200
10	91.0 / 38.9	91.1 / 39.1	91.2 / 39.7	91.3 / 39.8	91.3 / 40.0
20	91.2 / 39.7	91.3 / 40.4	91.3 / 40.0	91.5 / 41.2	91.5 / 41.2
50	91.3 / 40.6	91.5 / 41.3	91.6 / 41.7	91.6 / 41.9	91.6 / 41.7
100	91.5 / 41.0	91.5 / 41.6	91.6 / 42.0	91.7 / 42.0	91.7 / 41.8
200	91.5 / 41.2	91.5 / 41.3	91.6 / 41.8	91.7 / 42.1	91.7 / 42.3
500	91.5 / 41.1	91.6 / 41.7	91.7 / 41.8	91.7 / 42.2	**91.8** / **42.6**
1,000	91.5 / 41.0	91.6 / 41.7	91.7 / 42.2	**91.8** / 42.1	**91.8** / 42.4
2,000	91.5 / 40.9	91.6 / 41.4	91.7 / 42.0	**91.8** / 42.1	**91.8** / 42.4

3.2　CoNLL-X Shared Task データでの実験結果

CoNLL-X shared task（多言語依存構造解析）[2]で用いられたデータを使用して実験を行った．デンマーク語，オランダ語，ポルトガル語，スウェーデン語のデータ[⋆2]を利用した．これらのデータには交差を持つ依存関係が含まれているが，訓練データの中で交差を持つ依存構造木の割合は，デンマーク語，オランダ語，ポルトガル語，スウェーデン語のデータでそれぞれ15.6%，36.4%，18.9%，9.8%であった．提案手法で使われるハイパーパラメータ等の値は，3.1節のWSJコーパスでの実験で使用されたものと同じ値を用いた．このshared taskでは依存関係ラベルの正解率も含めてシステムの性能評価を行っているが，提案手法では依存関係ラベルの推定は行わないため，WSJコーパスの場合と同様に係り先精度を用いて評価を行った．

実験結果を**表4**に示す．デンマーク語，オランダ語，ポルトガル語に対しては，交差のある依存構造解析の精度が交差のない依存構造解析の精度よりも高かったが，これらの言語では交差のある依存構造木が比較的多く出現する（訓練データ中に10%以上と比較的多数含まれている）ためと考えられる．解析精度の比較を行うために，このshared taskに参加した上位3つのシステムの結果も示してある（**CoNLL 1st**，**2nd**，**3rd**）．提案手法は，4つのすべての言語において，最も高い係り先精度を得た．2番目に精度の高かったシステム

⋆1 ランダマイゼーション検定[4]により，2つのシステムの文正解率に差はないことを帰無仮説として統計的検定を行ったところ，有意確率は$p = 0.23$であった．

⋆2 http://nextens.uvt.nl/~conll/より入手可能．このshared taskでは13の言語に対して評価が行われたが，デンマーク語，オランダ語，ポルトガル語，スウェーデン語に関しては使用されたデータが公開されているため，本研究ではこれらの4つの言語を用いて実験を行った．

表 4　CoNLL-X shared task データでの依存構造解析結果
Table 4　Results of dependency parsing on the CoNLL-X shared task data.

手法	DA（%）			
	デンマーク語	オランダ語	ポルトガル語	スウェーデン語
提案手法–交差なし（局所素性）	89.94	80.73	90.84	88.83
提案手法–交差なし（＋大域素性）	90.82	81.55	91.38	**89.80**
提案手法–交差あり（局所素性）	90.04	83.71	90.66	88.73
提案手法–交差あり（＋大域素性）	**90.96**	**84.85**	**91.40**	89.62
CoNLL 1st	90.58	83.57	91.36	89.54
CoNLL 2nd	89.80	82.91	91.22	89.50
CoNLL 3rd	89.66	81.73	90.30	89.05

に対して，ランダマイゼーション検定[4] を用いて係り先精度の有意差検定を行った⋆1．その結果，オランダ語での結果に対して有意差（$p < 0.05$）が見られた（デンマーク語，オランダ語，ポルトガル語，スウェーデン語のデータでの p 値は，それぞれ 0.10，0.0047，0.46，0.29 であった）．

4. 関 連 研 究

大域的な素性を用いて依存構造解析を行う方法は，これまでにもいくつか研究されている．

工藤ら[21] は，決定的（deterministic）な日本語の係り受け解析手法を提案した．彼らの方法では決定的に解析が行われるため，すでに係り先が決まったトークンに関する情報を，それ以降のトークンを処理する際に利用することができる．動的素性と名付けられたそのような素性を利用することで，解析精度が大きく向上したと報告している．また英語の決定的な依存構造解析においても，同様の素性が有用であることが山田ら[17] により報告されている．決定的な解析手法では，計算量を増加させることなくこのような非局所的な素性を利用できる．しかしながら利用可能な素性は，処理を行う時点ですでに決定している構造に関する情報に限られる．また決定的な解析手法では，解析の途中で誤りが起きるとそれ以降の処理に影響が及んでしまうような，誤りが伝播するという問題が存在する．

McDonald ら[11] は二次の素性（依存構造木中で，2 つの隣接する兄弟ノードとその親ノード間を結ぶ，2 つのエッジに関する素性）を利用することができる依存構造解析手法を提案した．彼らの手法では二次の Eisner アルゴリズムを用いて，交差のない依存構造解析を行う．交差のある依存構造解析を行う場合には，そこで得られた交差のない依存構造木を greedy アルゴリズムを用いて変形することにより，より最適な交差のある依存構造木を得る．この方法は効率的で高い精度を得ることができる．しかしながら m 次の Eisner アルゴリズムの計算量は，$m > 1$ の場合に $O(n^{m+1})$ であるため（n は文中のトークン数）[11]，この手法でさらに高次の素性を扱うのは難しい．提案手法は厳密解を求めるのではなく，サンプリングを用いて近似的に解を求めるため，計算量は生成するサンプル数に依存するが素性の次数には依存しない．また，使用する素性の次数によって探索アルゴリズムを変更する必要はない．具体的な提案手法における解析時の計算量は次のようになる．図 2 のギブスサンプリングによりサンプルを生成する際の計算量は，この図の 4 行目の計算量を $O(n^3)$ とすると⋆2，2 行目と 3 行目で繰り返し計算を行うため，全体として $O(Rn^4)$ となる．サンプルを生成した後，式 (8) により周辺確率を計算し，Eisner 法（または CLE 法）を用いて解の探索を行うが，これらの計算量はそれぞれ $O(Rn^2)$ と $O(n^3)$（Eisner 法ではなく CLE 法の場合は $O(n^2)$）であるため，提案手法全体の計算量は $O(Rn^4)$ である⋆3．つまり，計算量はサンプル数 R に比例するが，素性の次数に関する変数は含まれないため素性の複雑さには依存しない．提案手法では，使用する素性やデータによっては，十分な精度で解析を行うために必要なサンプル数が多くなる可能性がある．しかしながら，複雑な素性を使用する場合でも計算が手に負えなくなるようなことはなく，サンプル数により計算時間と解析精度を調整することができ，探索アルゴリズムを変更する必要がないことは，提案手法の利点であると思われる．

Riedel ら[14] は，整数線形計画法[16] を用いた依存構造解析のための手法を提案した．彼らは，「等位接続詞が結ぶトークンは同じ品詞でなければならない」「動詞は 2 つ以上の主語を持ってはならない」というような言語学的な制約を取り入れて解析を行うために，整数線形計画法を利用した．この手法は文レベルの大域的な情報を利用することができる．しかしながら大域的な情報を，学習可能な重みが付与されて柔軟に制約の度合いが変えられる素性として利用するのではなく，必ず満たさなければならない制約条件として利用するため，ノイズの多いデータ等に対しては問題が生じることが報告されている[14]．

⋆1 検定には，shared task でも用いられた Bikel のプログラム（http://www.cis.upenn.edu/~dbikel/software.html#comparator）を使用した．

⋆2 2.5 節に示した大域的素性のうち，Child+Ancestor 以外の素性を使う場合は $O(n^2)$ で計算できる．Child+Ancestor だけは，1 つの文に最大で $n(n-1)/2$ 個の素性が存在する可能性があるため，計算量は $O(n^3)$ となる．

⋆3 ここで議論している計算量は最悪時の値である．実際に 2.5 節に示したすべての大域素性を利用して交差のない依存構造解析を行った場合に，解析全体で要した計算時間を測定して最小二乗法により曲線をあてはめたところ，2.2 節で述べた計算時間の削減手法を用いなかった場合で $O(Rn^{3.02})$，用いた場合で $O(Rn^{2.03})$ であった．

依存構造解析以外の言語処理においても，大域的な情報の利用は試みられている．Collinsらはリランキングに基づく句構造解析手法を提案した[5]．この方法では，まず基本的な構文解析器（base parser）を用いて入力された文を解析し，n-best 解を求める．次にその n-best 解に対して，文レベルの大域的な素性を利用して，別の統計的モデルにより再度順位付けを行う．この方法は文全体を考慮した任意の素性を用いることができる．しかしながら，n-best 解を得るために使用される基本的な解析器は一般的にあまり複雑な素性を扱うことはできず，もし n-best 解の中に正解が含まれていなければ最終的に正しい解を得ることはできない．そのため，より良い n-best 解を得るための研究も行われている[3]．

Johnson らは，単一化文法のための exponential model を提案した[8]．このモデルでは大域的な素性を利用することが可能であり，pseudo-likelihood estimator を用いてパラメータの推定を行っている．また，Rosenfeld らは whole-sentence exponential language model を提案した[15]．彼らは文中の任意の素性を利用して言語モデルを構築するために，exponential model を用いて確率モデルを定義し，ギブスサンプリング等のサンプリング手法を用いて計算を行っている．

5. 結　　論

本稿では，文中の任意の素性を利用することができる，ギブスサンプリングを用いた依存構造解析手法について検討した．この手法は最大全域木に基づく依存構造解析手法を応用したものであり，そのエッジのスコアには，exponential model により定義された構文木全体の確率分布からギブスサンプリングを用いて計算される周辺確率を使用する．実験の結果，大域的な素性を利用することにより高い精度が得られることを確認した．

本稿ではトークン間の依存関係のみに注目し，依存関係のラベルについては考慮しなかった．各トークンの係り先だけではなく依存関係のラベルも扱う場合，それらの組合せを考慮しなければならないため，依存関係のみを扱う場合に比べて探索空間が非常に大きくなる．しかしながら，そのような高次元の探索空間もギブスサンプリングで扱うことが可能であると思われる．依存関係のラベルも考慮した依存構造解析を行うことは今後の課題である．

謝辞　英語の主辞規則を提供していただいた（株）ジャストシステムの山田寛康氏と，多くのアドバイスをいただいた奈良先端科学技術大学院大学の松本裕治教授に深く感謝いたします．

参 考 文 献

1) Abney, S.P.: Stochastic Attribute-Value Grammars, *Computational Linguistics*, Vol.23, No.4, pp.597–618 (1997).
2) Buchholz, S. and Marsi, E.: CoNLL-X Shared Task on Multilingual Dependency Parsing, *Proc. CoNLL 2006*, pp.149–164 (2006).
3) Charniak, E. and Johnson, M.: Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking, *Proc. ACL 2005*, pp.173–180 (2005).
4) Chinchor, N.: The Statistical Significance of the MUC-4 Results, *Proc. MUC-4*, pp.30–50 (1992).
5) Collins, M. and Koo, T.: Discriminative Reranking for Natural Language Parsing, *Computational Linguistics*, Vol.31, No.1, pp.25–69 (2005).
6) Covington, M.A.: A Fundamental Algorithm for Dependency Parsing, *Proc. ACM Southeast Conference 2001*, pp.95–102 (2001).
7) Eisner, J.: Three New Probabilistic Models for Dependency Parsing: An Exploration, *Proc. COLING '96*, pp.340–345 (1996).
8) Johnson, M., Geman, S., Canon, S., Chi, Z. and Riezler, S.: Estimators for Stochastic "Unification-Based" Grammars, *Proc. ACL '99*, pp.535–541 (1999).
9) McDonald, R., Crammer, K. and Pereira, F.: Online Large-Margin Training of Dependency Parsers, *Proc. ACL 2005*, pp.91–98 (2005).
10) McDonald, R., Pereira, F., Ribarow, K. and Hajic, J.: Non-projective Dependency Parsing using Spanning Tree Algorithms, *Proc. HLT/EMNLP 2005*, pp.523–530 (2005).
11) McDonald, R. and Pereira, F.: Online Learning of Approximate Dependency Parsing Algorithms, *Proc. EACL 2006*, pp.81–88 (2006).
12) Nivre, J.: An Efficient Algorithm for Projective Dependency Parsing, *Proc. IWPT 2003*, pp.149–160 (2003).
13) Pietra, S.D., Pietra, V.D. and Lafferty, J.: Inducing Features of Random Fields, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.19, No.4, pp.380–393 (1997).
14) Riedel, S. and Clarke, J.: Incremental Integer Linear Programming for Non-projective Dependency Parsing, *Proc. EMNLP 2006*, pp.129–137 (2006).
15) Rosenfeld, R., Chen, S.F. and Zhu, X.: Whole-Sentence Exponential Language Models: A Vehicle For Linguistic-Statistical Integration, *Computers Speech and Language*, Vol.15, No.1, pp.55–73 (2001).
16) Roth, D. and Yih, W.: A Linear Programming Formulation for Global Inference in Natural Language Tasks, *Proc. CoNLL 2004*, pp.1–8 (2004).
17) 山田寛康，松本裕治：Support Vector Machine を用いた決定性上昇型依存構造解析，

情報処理学会論文誌，Vol.45, No.10, pp.2416–2427 (2004).
18) 伊庭幸人，種村正美，大森裕浩，和合　肇，佐藤整尚，高橋明彦：統計科学のフロンティア 12 計算統計 II マルコフ連鎖モンテカルロ法とその周辺，岩波書店 (2005).
19) 田村晃裕，高村大也，奥村　学：符号化問題として解く日本語係り受け解析，情報処理学会研究報告 2006-NL-176，pp.17–24 (2006).
20) 内元清貴，関根　聡，井佐原均：最大エントロピー法に基づくモデルを用いた日本語係り受け解析，情報処理学会論文誌，Vol.40, No.9, pp.3397–3407 (1999).
21) 工藤　拓，松本裕治：チャンキングの段階適用による係り受け解析，情報処理学会論文誌，Vol.43, No.6, pp.1834–1842 (2002).
22) 中川哲治，工藤　拓，松本裕治：Support Vector Machine を用いた形態素解析と修正学習法の提案，情報処理学会論文誌，Vol.44, No.5, pp.1354–1367 (2003).

(平成 20 年 2 月 6 日受付)
(平成 20 年 9 月 10 日採録)

中川　哲治（正会員）

2000 年筑波大学第三学群情報学類卒業．2002 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．同年沖電気工業（株）入社．2006 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．2007 年より情報通信研究機構専攻研究員．博士（工学）．統計的自然言語処理および機械学習に興味を持つ．