Support Vector Machine を用いた決定性上昇型依存構造解析

山田寛康[†] 松本裕治[††]

本論文では，英語文に対する統計的依存構造解析手法を提案する．様々な分野のテキストに対して高精度な統計的構文解析を実現するには，各分野ごとに解析済み訓練データを用意する必要がある．しかし Penn treebank のような句構造のタグ付け作業を行う場合，タグ付け者は対象言語を母国語とするだけでなく，言語学に関する専門知識が必要である．また対象とするテキストが，医学や法律文書などの専門分野では，言語学に関する知識だけでなく，その分野特有の専門知識が必要となり，タグ付け可能者の数はきわめて少なくなる．その結果，様々な分野で十分な訓練データを用意することが現実的に難しい．依存構造は単語間の修飾関係により文を表現するもので，句構造に比べ簡潔な構造である．したがって対象言語を母国語とする多くの人がタグ付け可能であり，構造の簡潔さはタグ付け者間での揺れを軽減し，高品質の訓練データを準備できる期待が持てる．我々は様々な分野に適用可能な統計的構文解析器の構築を目的とし，現実的に訓練データを準備可能という観点から，依存構造に注目する．対象分野で依存構造解析済みデータを獲得できた場合，それを訓練データとし直接解析規則を学習する必要がある．また学習したモデルを使用し未知の文を解析する統計的依存構造解析手法が必要となる．提案する依存構造解析手法は，文頭から順に 2 つの単語間の依存構造を決定的に構築する．またどのような文脈で適切な依存木を構築するかを，汎化性能の高い機械学習法 Support Vector Machine を用いて学習する．提案した依存構造解析手法を Penn treebank を依存木に変換したデータを使用して評価実験を行った結果，句構造情報を使用せずに，90%を超える高い精度を得た．

Bottom-up Deterministic Analysis of Dependency Structure Using Support Vector Machines

Hiroyasu Yamada[†] and Yuji Matsumoto[††]

In this paper, we propose a method for analyzing word-word dependencies using deterministic bottom-up manner using Support Vector Machines. To implement a statistical parser with high accuracy, we have to prepare annotated training data for each target domain. Phrase structure annotation is not easy because annotators need to be well acquainted with deep linguistic theories and phrase structure rules. This becomes a great hindrance to accumulation of large scale annotated corpus of phrase structures. On the other hand, word-word dependency relation is much easier to understand and is more amenable to annotators who have good knowledge of target text domain but may lack linguistic knowledge. Besides, since annotating simpler structure is useful for reaching a consensus among annotators, it is expected that construction of training data will become more noise-free. We experimented with dependency trees converted from Penn treebank data, and achieved over 90% accuracy of word-word dependency. Though the result is little worse than the most up-to-date phrase structure based parsers, it looks satisfactorily accurate considering that our parser uses no information from phrase structures.

1. はじめに

自然言語の構文解析は，計算機による言語理解の実現において重要な基礎技術の 1 つである．また近年の電子化テキストの増加にともない，様々な分野のテキストに対して構文解析処理を行う需要も高い．このような背景から構文解析には高い解析精度だけでなく，様々な分野のテキストに対して解析可能な頑健性も求められている．あらかじめ文法規則や解析規則を記述するといった方法では，限られた分野に対して高い解析精度を実現できたとしても，幅広い分野へ応用するには，規則の追加，および変更に多大なコストを必要とする．そのため近年では統計情報や機械学習法を使用し，解析に必要な規則を自動獲得することで，高い

† 北陸先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Japan Institute of Science and Technology
†† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

精度と頑健性を実現する統計的構文解析の研究が行われている．特に Penn treebank [10] コーパスに代表される句構造解析済データから，文法規則および規則の適用確率を教師付き学習を用いて自動獲得する手法では，高い解析精度が報告されている[3),5),9),12)]．

教師付き学習を用いて様々な分野で高い解析精度を実現するには，各分野ごとにある程度の解析済み訓練データを用意する必要がある．句構造解析済データの作成には，句構造木の構造とともに，各ノードに動詞句や名詞句といったラベルを付与する必要がある．このラベルを一貫性を維持し付与するには，対象言語を母国語とするだけでなく，言語学に関する専門知識が必要である．また対象とする分野が医学生物学や法律文書などの専門分野である場合，言語学の知識以外にその分野特有の専門知識も必要となるため，タグ付け可能者数は，きわめて少数になる．その結果，各分野で必要な訓練データを作成することが現実的に難しい．

依存構造は，単語間の修飾関係のみで文の構造を表現するもので，句構造に比べて簡潔な構造である．依存構造木のタグ付け作業には，文の任意の 2 つの単語間に修飾関係があるか否かのみを判断しタグ付けすればよいため，句構造木のように，木の各ノードにラベルを付与する負担はまったくない．またタグ付け者に要求される知識は，単語間の修飾関係が判断可能であるという比較的浅い言語学的知識のみである．したがって対象言語を母国語とする多くの人がタグ付け可能となり，専門性の高い分野においてもタグ付け作業可能な人を著しく限定しない．また簡潔な構造はタグ付け者間での一貫性を保つことにも役立ち，高品質の訓練データを多量に準備できる．

依存構造情報を他の自然言語処理タスクに利用した研究に Lin [4] の研究がある．彼は単語の語義曖昧性の解消に英語の依存構造を使用し，その有効性について報告している．また我々は医学生物学分野の文献に出現する専門用語に対し，用法や薬品名などのカテゴリへ自動的に分類する手法を提案した[16)]．我々の実験においても，専門用語のクラス分類にも依存構造情報が有用であることが分かった．以上のような先行研究からも，依存構造情報は様々な応用タスクにおいても有用な情報であると考える．

我々は，新たな分野でも現実的に訓練データを作成可能であるとの観点から，依存構造に注目する．日本語では係り受け解析として依存構造を扱う研究が行われてきた[15)]．日本語の係り受けには，係り先が右方向であるという原則があるが，これは日本語特有なものである．我々はより汎用的な依存構造解析を目指し，英語のような左右どちらの方向にも係り先が存在しうる依存構造を扱う．

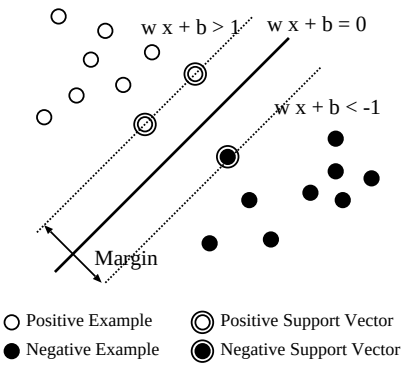

図 1 Support Vector Machine
Fig. 1 Support Vector Machine.

新たな分野で依存構造解析済データを用意した場合，依存構造解析済データを訓練データとし，そこから直接依存構造解析に必要な規則を学習する必要がある．また学習したモデルを使用し未知の文に対し依存構造を解析する手法が必要となる．本論文では，依存構造の基本性質である非交差性を満たしながら文頭から順に決定的に依存木を構築する上昇型解析手法を提案する．またどのような文脈で適切な依存木を構築するかを，機械学習法である Support Vector Machine（SVM）を用いて学習する．

以下次章では依存構造解析の学習に用いる SVM の概要について述べる．次に 3 章で本論文で提案する上昇型依存構造解析手法について説明し，4 章では SVM を用いた依存構造の学習方法について述べる．5 章では Penn treebank を用いた評価実験について報告し，6 章で実験に対する考察を行う．7 章で関連研究について述べ，最後に 8 章でまとめと今後の課題について述べる．

2. Support Vector Machine

Support Vector Machine（SVM）は Vapnik [14] により提案された二値線形分類器である．**図 1** に SVM の概要図を示す．SVM は n 次元ベクトル空間に分布する l 個の訓練事例 $(\mathbf{x}_i, y_i), (\mathbf{x}_i \in \mathbf{R}^n, y_i \in \{+1, -1\}, 1 \leq i \leq l)$ を正しく分離する超平面 $\mathbf{w} \cdot \mathbf{x} + b = 0$ $(\mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R})$ を求めるアルゴリズムである．訓練事例を正しく分離する超平面は多数存在する．SVM は図 1 の破線で示した，求める平面に平行な 2 つの平面間の距離マージンを最大にするような超平面を見つける．この超平面は式 (1) に示す制約付き最適化問題を解くことで求めることができる．

$$min\ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i}^{l}\xi_i$$
$$s.t.\ y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i,\ \ \xi_i \geq 0 \qquad (1)$$

ここで C はユーザの指定する定数（本論文の実験では $C = 1$ に固定）で，ξ_i は *slack variable* と呼ばれ，訓練データのエラーをある程度許容するソフトマージンの実現に必要な変数である（詳細は文献 14) を参照）．この最適化問題を解き $\mathbf{w}$，および b を求めることでマージンを最大にする分離超平面を得ることができる．未知のデータ $\mathbf{x}$ に対する分類は，分離超平面との位置を表す式 (2) の $f(\mathbf{x})$ の符号により決定される（α_i は，式 (1) の最適化問題を解くために導入した Lagrange 乗数である）．

$$f(\mathbf{x}) = \sum_{i=1}^{l}\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \qquad (2)$$

ここで $K(\mathbf{x}', \mathbf{x}'')$ は Kernel 関数と呼ばれ，引数である 2 つのベクトルの類似度を求める関数であり，Kernel 関数を適切に設定することで，非線形な学習が可能となる．SVM のマージン最大化に基づく戦略は，素性空間の次元に依存しない高い汎化性能を持つことが理論的に示されており，非常に高次元の素性空間における学習でも，過学習をできるだけ軽減することが可能となる．

依存構造解析では，品詞や単語自身の情報を学習の中心的な素性とするため，素性空間の次元数は数十万以上となり，過学習の危険性が高くなる．SVM を使用することで多数の素性を考慮したとしても過学習を回避できる期待が持てる．さらに Kernel 関数として多項式関数 $(\mathbf{x}'\cdot\mathbf{x}'' + 1)^d$ を使用することで，d 個までの素性の組合せを網羅的に考慮した学習が，計算量を大きく変化させることなく可能になる．

式 (2) の $f(\mathbf{x})$ の絶対値は，事例 $\mathbf{x}$ と分離超平面間の距離を表す．しかしこの値は正例または負例に対する適切な尤度である理論的根拠はない．確率的構文解析で用いられる，動的計画法を用いた文全体で最適な解析木の探索を行う場合の尤度としては適切でない．そのため本論文では SVM を用いて決定的な解析を行う．

3. 決定性上昇型依存構造解析

3.1 依存構造解析アクション

本論文で提案する依存構造解析手法は，与えられた英語文に対し文頭から順に，単語間に 3 つの**依存構造解析アクション** *Shift*，*Right*，および *Left* を適用し

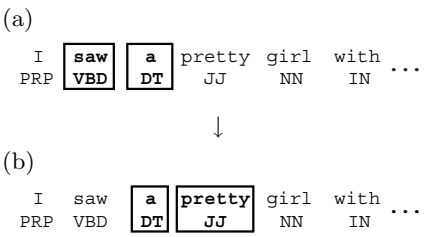

図 2 *Shift* の例：(a) *Shift* 実行前，(b) *Shift* 実行後
Fig. 2 An example of the action *Shift*: (a) shows states before the action, (b) is the result after the action.

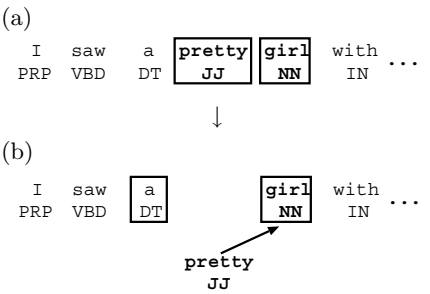

図 3 *Right* の例：(a) *Right* の実行前，(b) *Right* の実行後
Fig. 3 An example of the action *Right*: (a) shows states before the action, (b) is the result after the action.

依存木を構築する．以降，この 3 つの解析アクションを使用した解析モデルを 3 actions モデルと呼ぶ．また文中の各単語は依存木の各ノードに対応するので，ノードと呼ぶこととする．依存構造を構築する 2 つのノードを解析対象ノードと呼ぶ．

Shift は依存関係を構築せず，解析対象ノードを右に 1 つ移動する手続きである．**図 2** に *Shift* の例を示す．隣接した 2 つの解析対象ノード “saw” および “a” には依存関係を構築せず，解析対象ノードを 1 つ右の “a” と “pretty” に移す（図 2 (b)）．

Right は，隣接した解析対象ノード間に，左ノードが右ノードに係るという依存関係を構築する．**図 3** に *Right* の例を示す．解析対象ノードである “pretty” と “girl” において，手続き *Right* を実行することで，“pretty” が “girl” に係り “girl” を親とする依存木が構築される（図 3 (b)）．また *Right* の実行後の解析対象位置は “a” と “girl” とする．このような解析順序を保持することで，左から係りうるノードの解析を先に行うことが可能となる．

Left は，隣接した解析対象ノード間において，右ノードが左ノードに係るという依存関係を構築する．**図 4** に *Left* の例を示す．*Left* の実行により “girl” が

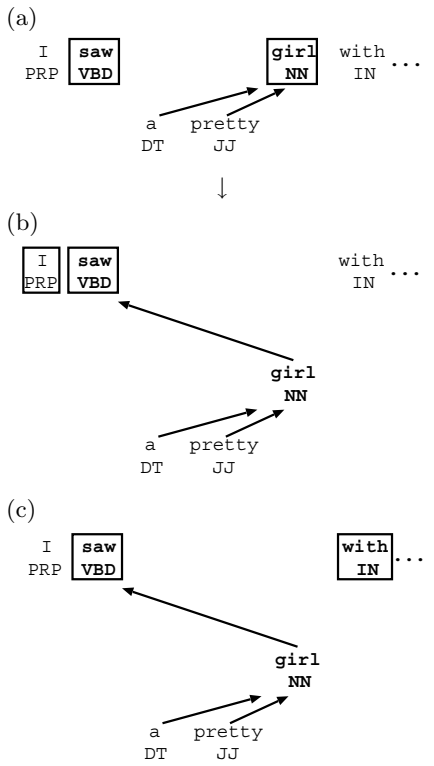

図 4 手続き *Left* の例：(a) *Left* の実行前，(b) *Left* 実行後（*Left-backward*）．(c) *Left* の実行後（*Left-forward*）．
Fig. 4 An example of the action *Left*: (a) shows states before the action, (b) is the result after the action *Left-backward*, (c) is the result after the action *Left-forward*.

“saw” に係り，“saw” を親に持つ依存木が構築される．*Left* の実行後の解析対象位置は “I” と “saw” もしくは “saw” と “with” の 2 通りがあり，本論文ではそれぞれの解析順序を *Left-backward*（図 4 (b)）と *Left-forward*（図 4 (c)）と呼ぶこととする．決定的な解析を行う場合，*Left-backward* は *Left-forward* に比べ，アクション実行後に解析位置が 1 つ戻るので，解析順序が異なる．本論文ではこれら 2 つの解析順序の違いが解析精度にどのような影響を与えるか実験的に示す．

依存構造には同じ文中でそれぞれの依存関係が交差してはいけないという非交差性を守る必要がある．決定的な解析を行い，かつ非交差性を保ちながら正しい解析を行うには，*Left* または *Right* の実行に留意が必要である．*Left* または *Right* を用いて 2 ノード間に依存関係を構築する場合には，係り元となるノードは，必ず他ノードから係りえないことが保証されなければならない．そこでたとえ解析対象ノード間に依存関係がある場合にでも，係り元ノードが他ノードから係りうる可能性がある場合は *Shift* を実行して，係り元の部分依存構造を先に解析する．

Input Sentence: $(w_1,p_1),(w_2,p_2),\cdots,(w_n,p_n)$
Initialize:
  $i = 1$;
  $\mathrm{T} = \{(w_1,p_1),(w_2,p_2),\cdots,(w_n,p_n)\}$;
  $no\_construction = true$;
Start:
while |T| > 1 **do begin**
  if i == |T| **then**
    if $no\_construction == true$ **then break;**
    $no\_construction = true$;
    i = 1 ;
  else
    $\mathbf{x} = get\_contextual\_features(\mathrm{T}, i)$;
    $y = estimate\_action(estimator, \mathbf{x})$;
    $execution(\mathrm{T}, i, y)$;
    if $y ==$ *Left* or *Right* **then**
      $no\_construction = false$;
    end;
  end;
end;

図 5 解析アルゴリズムの疑似コード
Fig. 5 Pseudo-code of our parsing algorithm.

3.2 依存構造解析アルゴリズム

前節で述べた依存構造解析アクションを使用して決定的に依存木を構築する解析アルゴリズムについて説明する．**図 5** に解析アルゴリズムの疑似コードを示す．n 個の単語列からなる入力文は，単語 w_k と品詞 p_k の組 (w_k, p_k) で表される $(1 \leq k \leq n)$．T は構築した各部分木を格納する変数であり，T の要素 t_m は構築した部分木のルートノードを表す．T の初期値は入力である単語と品詞の組 $(w_1,p_1),(w_2,p_2),\cdots,(w_n,p_n)$ である．解析は文頭から文末に向けて順に行われ，i は解析対象ノードのうち左ノードの添字を表し，$i+1$ が右ノードの添字を表す．解析は解析対象ノード t_i と t_{i+1} の間に，適切な依存構造解析アクション y を前後の文脈から推定することで実現する．

解析アクションの推定には 2 つの関数 *get_contextual_features* と *estimate_action* により実現する．*get_contextual_features*（T, i）は T 中の解析対象位置 i の前後の文脈から学習に使用する素性ベクトル $\mathbf{x}$ を返す関数である．素性に関する詳細は 4.1 節で述べる．関数 *estimate_action* は，抽出した素性ベクトル $\mathbf{x}$ と

estimator を用いて適切な解析アクション y を推定する．推定したアクション y は，手続き *execution* で実行することにより，依存木が構築される．解析アクション *Left* および *Right* のいずれかを実行すると，2 つの解析対象ノードのうちどちらかが，もう片方の子ノードとなり，T の要素から削除され，T の要素数は 1 減少する．変数 *no_construction* の役割は，文頭から文末にかけて，一度でも依存木を構築するアクションが起こったか否かを調べるものである．解析対象位置が文末に達したとき，もし *no_construction* が true であれば，どの文脈においても依存木を構築できなかったことを意味し，これ以上解析することは不可能となる．この場合，それまでに構築した部分木を出力し解析を終了する．解析対象位置が文末に達したとき，*no_construction* が false であれば，解析対象位置を文頭に戻し，T の要素数が 1 となる依存木完成まで，同様の解析を繰り返す．

3.3 依存構造解析モデル

3.1 節で述べた 3 actions モデルを拡張した 3 種類の解析モデルについて説明する．

4 actions モデル：3 actions モデルでは，次に示す 2 つの違いを区別できていない：(i) 真に依存関係があるノード間だが，係り元に係りうるノードがあるため，*Shift* する場合，(ii) 真に依存関係がないノード間に対して，*Shift* して解析位置を移動させる場合．(i) と (ii) の違いが 1 つのアクション *Shift* として表現されている．そこでこれら 2 つを区別するため (i) に該当するものを *Wait* という新たな解析アクションとして細分化する．この細分化を行った場合でも，解析順序は変化しない．すなわち *Wait* または *Shift* の実行は，解析位置を右へ移動する同一の振舞いである．しかし学習においては，異なる分類クラスとして学習される．

5 actions モデル：3 actions モデルでは依存木を構築するために *Left* と *Right* を使用して実現した．しかし依存木の各ノードにおける最左子ノードおよび最右ノードの解析時には特徴がある．たとえば，名詞句を形成する 1 つの部分依存木において最左子ノードは “a” や “the” などの冠詞などの出現が多い．そこで解析アクション *Right* の中で，最左子ノードが決定される場合を *First* という新たなアクションを用いる．また図 5 のアルゴリズムに従って文頭から順に解析を行うと，最右子ノード決定時における多くの場合で部分木の完成を知ることができ，これにより不要な *Shift* を避けることができる．**図 6** に “She said that he gave her the book.” という文の解析例を示す．今，解析対象ノードは “gave” と “her” であるとする．実線で描かれた依存関係はすでに解析された部分を表し，破線で描かれた関係は，まだ解析されていない関係を表す．*Left-backward* モデルを用いると，“gave” と “her” の間に *Left* によって依存木を構築した後，解析対象位置は “that” “gave” に移る．しかし “gave” を親とする部分依存木がまだ完成していないため必ず *Shift* を実行することになり，この *Shift* は無駄である．ここで “gave” の最右子ノード “book” が “gave” に係ることを認識した場合のみ，解析位置を戻すことができれば．先ほどの無駄な *Shift* を防ぐことができる．そこで *Left* のうち，最右子ノードの依存関係を決定する場合には *Last* というアクションを使用する．*Last* の実行後のみ解析対象位置を 1 つ戻す解析順序を *Last-backward* と呼び，*Last* も含め右ノードが左ノードに係るアクション後につねに解析位置を戻す *Left-backward* と区別する．

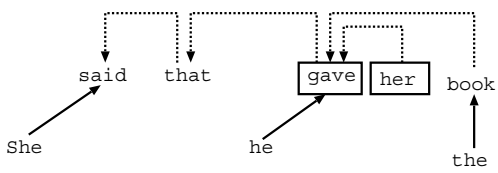

図 6 *Last* の例
Fig. 6 An example of the action *Last*.

6 actions モデル：3 actions モデルに *Wait*，*First*，および *Last* を追加したモデル．

4. 依存構造解析規則の学習

図 5 で示したアルゴリズムの実現には，*estimator* が解析の各過程で，文脈 $\mathbf{x}$ から適切な解析アクション y を推定する必要がある．これは文脈 $\mathbf{x}$ から解析アクション y への分類問題であり，我々は SVM を用いて学習を行う．訓練文自身を図 5 のアルゴリズムで解析し，*estimator* は訓練文にタグ付けされた正解依存関係を使用しつねに正しい解析アクションが選択される．そして各段階で得られた文脈ベクトルと解析アクションの組 $(\mathbf{x}, y)$ を 1 つの訓練事例とする．SVM は二値分類器であるのに対し，解析アクションは 3 種類以上あるため pairwise 法[13] により多値分類器へ拡張する．テスト文の解析では *estimator* は学習した SVM であり，テスト文の各文脈で適切な解析アクションを推定する．

4.1 素　　性

estimator が適切な解析アクションを推定するために，文脈中のどのような情報を手がかりとし，素性と

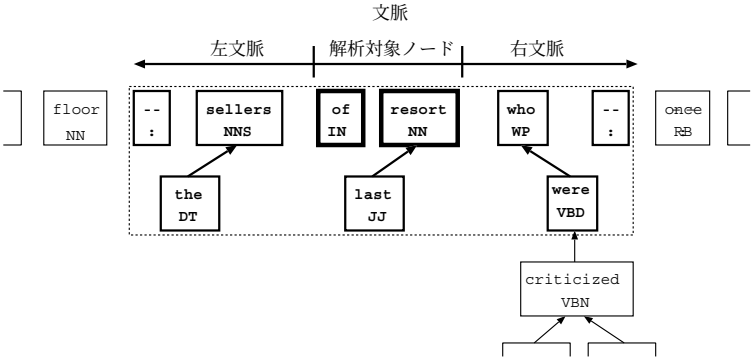

図 7 文脈と素性
Fig. 7 An example of the context and features.

表 1 学習に使用する素性とその値
Table 1 Summary of the feature types and their values.

種類	値
pos	品詞タグ
lex	単語
ch-L-pos	左から親ノードを修飾する子ノードの品詞
ch-L-lex	左から親ノードを修飾する子ノードの単語
ch-R-pos	右から親ノードを修飾する子ノードの品詞
ch-R-lex	右から親ノードを修飾する子ノードの単語

して考慮するかについて説明する．素性は解析対象ノードとその左右に位置する文脈から抽出し，それぞれを左文脈，右文脈と呼ぶこととする．解析対象ノードと左右の文脈を含めたものを単に文脈と呼ぶ．**図 7** に左右の文脈がそれぞれ 2 ノードの場合における例を示す．図 7 で解析対象ノードは “of” および “resort” であり，その 2 つの単語から左右 2 ノード分の情報を学習の素性として使用する．

素性は位置 p，種類 t，値 v の 3 つ組 (p, t, v) で表現される．位置 p は解析対象ノードからの $|p|$ ノード離れていることを表し $p < 0$ は左文脈，$p > 0$ は右文脈を表す．また解析対象ノードの左右のノードをそれぞれ $p = 0-$，$p = 0+$ で表す．使用する素性の種類を**表 1** に示す．pos，lex はそれぞれ，文脈中に現れた品詞および単語を表す．ch-で始まる素性を**子ノード素性**と呼び，解析の過程で決定した依存木の子ノード情報を表す．ch-L-，ch-R-は子ノードのうち親ノードをどちらの方向から修飾するかを表し，L ならば左から，R であれば右から修飾することを表す．修飾する方向を区別することで，たとえば動詞を左から修飾する名詞と，逆に右から修飾する名詞とで，主語，目的語の関係を区別することができる．ch-L-lex，ch-R-lex は子ノードの単語で，文字列自身が素性の値となる．ch-L-pos，ch-R-pos は子ノードの品詞タグが素性の値となる．図 7 の例では，使用する素性は以下のようになる．(-2, pos, :), (-2, lex, –), (-1, pos, NNS), (-1, lex, sellers), (-1, ch-L-pos, DT), (-1, ch-L-lex, the), (0-, pos, IN), (0-, lex, of), (0+, pos, NN), (0+, lex, resort), (0+, ch-L-pos, JJ), (0+, ch-L-lex, last), (+1, pos, WP),(+1, lex, who), (+1, ch-R-pos, VBD), (+1, ch-R-lex, were), (+2, pos, :), (+2, lex, –).

4.2 訓練事例のグループ化による学習時間の効率化

SVM の学習時間は，分類問題の難しさにも依存するが，おおむね訓練事例数の 2 乗から 3 乗に比例する．したがって訓練事例数が 100 万以上となる大規模な学習を行うと現実的な時間での学習が困難となる．本手法では解析の各ステップが 1 つの訓練事例に対応するため，Penn treebank の標準訓練データセットである約 40,000 文から 150 万以上の訓練事例が得られる．したがって一度にすべての訓練事例を使用した学習が難しい．そこで訓練事例をいくつかのグループに分割することで学習時間の効率化を行う．どのような基準でグループ化するかには，様々な方法が考えられるが，今回は暫定的に，解析対象ノードの品詞の種類でグループ化した．具体的には解析対象ノードのうち左ノードの品詞素性（0-, pos, POS）を使用し同一の品詞素性を持つ訓練事例を 1 つのグループとした．そして各グループごとに SVM を構築し，テスト文の解析では解析対象ノードのうち左ノードの品詞により，対応する SVM を用いて解析アクションの推定を行う．

5. 実　　験

本論文で提案した依存構造解析手法の有効性を検証するために，Penn treebank コーパスを使用した実験について報告する．

5.1 デ ー タ

実験では Penn treebank の標準データセットに従い，section 2 から 21 の 39,832 文を訓練データとし，

section 23 の 2,416 文をテストデータとした．Penn treebank にタグ付けされた構文情報は句構造であるため，Collins の提案した主辞規則[9] と同様の規則を使用し依存構造に変換したものを実験データとした．テストデータの品詞タグ付けは，Nakagawa らが提案した修正学習法に基づく品詞タグ付け手法[11] を使用した．Nakagawa らの手法をテストデータである section 23 に適用した場合，品詞タグ付け精度は 97.1%であった．

5.2 評価方法

依存構造解析の性能を以下に示す 3 つの指標を用いて評価する．

係り先精度 = **正解係り先数** / 総係り先数
ルート精度 = **ルート正解数** / 総文数
文正解率 = **完全正解数** / 総文数

正解係り先数は，各単語に対し正しい係り先が推定できた個数を表す．**ルート正解数**は，ルートノードの係り先が正しく推定できた個数を表す．**完全正解数**は，1 文すべての単語の係り先が正しく推定できた文数を表す．テストデータである section 23 は 2,416 文 56,684 単語を含んでいる．この中での ",", "." などの句読点を除いた 49,892 単語を評価の対象とした．またすべての実験結果において，異なる手法やパラメータ間で結果の比較を行う場合，各評価指標ごとに有意水準 5%の両側符号検定を行った．以後の実験結果では検定における p 値を断りなく使用する．

5.3 解析順序

まず，3 actions モデルにおいて，解析アクション *Left* の実行後で，解析対象位置が異なる *Left-forward* と *Left-backward* の違いを比較した．使用した Kernel 関数は $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$ で素性は表 1 に示したものすべてを使用した．**表 2** に結果を示す．表 2 で，アクション数は，各解析手法をテストデータに適用した場合に実行した解析アクション数であり，*Left-backward* を基準にした時の増減数で示した（() 内の数はアクション数の総数である）．また *Left*，*Right* および *Shift* は，それぞれのアクション数の内訳を表す．表 2 より *Left-forward* よりも *Left-backward* が高い精度を示していることが分かる．符号検定の結果は，すべての評価指標において有意な差が認められた（文正解率で $p = 0.011$，その他は $p < 0.001$）．*Left-foward* と *Left-backward* のアクション数の違いを見ると *Left-backward* の方が 42,258 も少ない．解析アクション別にみると，減少のすべてが *Shift* であり，*Left*，および *Right* はわずかに増加している．*Left-backward* では *Left* により依存木を構築した後すぐに，その構築したノードとの依存関係を推定することとなる．決定的な解析では，依存関係があるものをできるだけ先に解析する方が良い結果が得られた．以後の実験結果では，特に断りのない場合，*Left-backward* モデルを使用することとする．

表 2 アクション *Left* 実行後の解析順序と精度
Table 2 The parsing accuracies and two different types of parsing orders after executing action "*Left*".

	Left-forward	*Left-backward*
係り先精度	0.887	**0.900**
ルート精度	0.835	**0.896**
文正解率	0.361	**0.379**
アクション数	±0 (140,045)	−42,258 (97,787)
Left	(25,560)	+595
Right	(27,459)	+483
Shift	(87,026)	−43,336

表 3 多項式 Kernel の次数と解析精度
Table 3 The number of degree of polynomial Kernel function and dependency accuracies.

	d : $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$			
	1	2	3	4
係り先精度	0.854	**0.900**	0.897	0.886
ルート精度	0.811	**0.896**	0.894	0.875
文正解率	0.261	**0.379**	0.368	0.346

5.4 多項式 Kernel 関数の次数と解析精度

次に SVM に使用する多項式 Kernel 関数の次数と，解析精度の変化を調査した．素性は表 1 に示したものすべてを使用し，3 actions モデルで解析を行った．**表 3** に結果を示す．符号検定の結果，多項式 Kernel 関数の次数 d が 1 である場合に比べ，d が 2 以上の場合では，すべての評価指標について，有意な差が認められた（$p < 0.001$）．このことから依存構造解析には単語や品詞といった素性の組合せまで考慮することが重要といえる．また $d = 2$ のとき最も良い解析精度を得たが，$d > 2$ では次数が上がるにつれて精度が低下する．符号検定の結果，$d = 2$ と $d > 2$ ですべての指標に有意な差が認められた（$d = 2$ と $d = 3$ 間のルート精度のみ $p = 0.035$，その他は $p < 0.001$）．このことからも依存構造解析では 2 つの素性の組合せを考慮することで十分であることが分かる．

決定木や最大エントロピー法では，使用する素性に関して素性の組合せも含め明示的に定義する必要がある．そのため計算量や素性数の増加による過学習の危険性から，網羅的に素性の組合せまで考慮した学習は困難である．したがって事前に統計量や経験的な知識に基づく素性選択にかなりのコストを費やす必要があ

表 4 文脈長と解析精度
Table 4 The length of context and dependency accuracies.

	(l, r): l（左文脈長），r（右文脈長）							
	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(3, 2)	(3, 3)	(3, 4)	(3,5)
係り先精度	0.900	**0.903**	**0.903**	0.901	0.898	0.902	0.900	0.897
ルート精度	0.896	0.911	**0.916**	0.913	0.897	0.915	0.912	0.909
文正解率	0.379	0.382	0.384	0.375	0.373	**0.387**	0.373	0.366

る．しかし SVM では多項式 Kernel 関数の次数を 2 以上にすることで，素性の組合せを網羅的に考慮した学習が可能であり，大きな利点といえる．

5.5 文 脈 長

次に考慮する文脈の長さが解析精度にどのような影響を与えるかを調査した．**表 4** に結果を示す．使用した素性は表 1 に示したものすべてを用い，Kernel 関数は $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$ を使用した．解析モデルは 3 actions モデルとした．表 4 で (l, r) はそれぞれ，l が左文脈の長さ，r が右文脈の長さを表す．

最も高い係り先精度を得たのは (2, 3)，および (2, 4) のときで，ルート精度は (2, 4) のときが最良であった．しかしいずれも次に精度の高い (3, 3) とは有意な差はなかった．文正解率では (3, 3) が最良の結果であったが，次に精度の高い (2, 4) とは有意な差が認められなかった．この結果は，文脈の長さを変化させても，すべの評価指標の向上に貢献する文脈長はなく，最適な文脈長をあらかじめ静的に決定することが難しいことを示唆している．

5.6 子ノード情報の有効性

次に，依存構造解析に解析過程で動的に決定する子ノード情報が有効であるかを調べるため以下に示す 4 つの素性セットごとに解析精度を調査した．

(1) 子ノード情報を使用しない（表 1 の pos，lex のみ）．
(2) 子ノードの単語情報を使用（表 1 の pos，lex，ch-L-lex，ch-R-lex を使用）．
(3) 子ノードの品詞情報を使用（表 1 の pos，lex，ch-L-pos，ch-R-pos を使用）．
(4) 子ノードの単語および品詞情報を使用（表 1 のすべてを使用）．

結果を**表 5** に示す．単語，品詞の両方を使用する場合が，単語のみ，品詞のみと比べ，いずれの評価指標でも最良の結果を得た．しかし符号検定の結果で有意な差が認められたのは，単語のみ使用の係り先精度についてだけであった（$p = 0.007$）．しかし子ノード情報を使用しない場合と，使用する場合とを比べると，すべての指標において，有意な差が認められた（$p < 0.001$）．このことは解析過程で推定した依存木の情報が，依存構造解析に有効であることを示している．この結果は工藤らの提案した日本語係り受け解析[15] において，動的素性と呼ばれる解析過程で決定した係り関係を素性として考慮することで解析精度が向上するという報告と一致する．英語，日本語にかかわらず，どのような単語や品詞が係っているかという情報は依存構造解析に有用な情報といえる．

表 5 子ノード情報の効果
Table 5 The effect of children nodes' features.

	未使用	使用する子ノード情報		
		単語のみ	品詞のみ	品詞単語
係り先精度	0.890	0.901	0.902	**0.903**
ルート精度	0.882	0.915	0.912	**0.916**
文正解率	0.348	0.383	0.374	**0.384**

5.7 最適な解析モデル

提案した 4 つの解析モデルに関して解析精度を比較した．**表 6** に比較結果を示す．5 および 6 actions モデルでは，*Last* も含め右ノードが左ノードに係った後，解析位置を 1 つ戻す *Left-backward* モデルと *Last* の実行後のみ解析位置を 1 つ戻す *Last-backward* モデルの双方の結果を載せた．結果は 6 actions での *Last-backward* モデルが最も高い精度を得た．符号検定の結果，6 actions の *Last-backward* モデルは，係り先精度では，すべてのモデルと有意な差が認められた（5 actions *Last-backward* で $p = 0.044$，その他は $p < 0.001$）．しかし，文正解率では 3 actions モデルのみに対し有意な差が認められ（$p = 0.019$），ルート精度では 5 actions での *Left-forward* だけでしか有意な差がなかった（$p = 0.049$）．

次に *Left-backward* と *Last-backward* について解析効率について調べるためテストデータに適用したアクション数と解析時間について調査した．表 6 のアクション数は，テストデータを解析するのに要したアクションの数について 3 actions モデルに対しての増減で記載した．また *Shift/Wait* は適用したアクションのうち *Shift* と *Wait* 数に対する増減を表す．*Wait* は 3 actions モデルにおける *Shift* の一部がそのまま移行したものなので，*Shift* の増減だけでは，正確な

表 6 解析モデルと精度
Table 6 The dependency accuracies and 4 parsing models.

	3 actions	4 actions	5 actions		6 actions	
			Left	*Last*	*Left*	*Last*
係り先精度	0.903	0.903	0.903	0.905	0.905	**0.906**
ルート精度	0.916	0.916	0.910	0.912	0.914	**0.918**
文正解率	0.384	0.391	0.389	0.390	0.399	**0.400**
アクション数	±0(97,610)	−263	+96	−3,368	−150	−3,710
Shift/*Wait* 数	±0(43,377)	−259	+88	−3,368	−153	−3,714
解析時間（分）	61	84	87	79	111	100

表 7 他手法との比較
Table 7 Comparison with related work.

	Charniak	Collins			Our parser
		model 1	model 2	model 3	
係り先精度	**0.921**	0.912	0.915	0.915	0.906
ルート精度	**0.952**	0.950	0.951	**0.952**	0.918
文正解率	**0.452**	0.406	0.431	0.433	0.400
葉ノード精度	**0.943**	0.936	0.936	0.937	0.936

Shift と数の増減を観察できないためである．解析時間はテストデータに要した時間を分単位で記載した☆．5 actions，および 6 actions いずれも *Left-backwad* と *Last-backward* では 3,000 以上の無駄な *Shift* を削減している．*Left-backward* と比べ有意な差があったのは係り先精度のみであるが，精度を低下させることなく若干の解析時間短縮に貢献できた．

5.8 他手法との比較

本手法と Charniak の提案した Maximum-Entropy-Inspired Parser（MEIP）[5]，および Collins の提案した 3 種類の確率モデルに基づく手法[9] とを比較した．2 つの統計的構文解析手法は，いずれも句構造解析器であり，Penn treebank 標準データセットを使用した実験について報告している．そこで section 23 に対する解析結果を，我々が使用した主辞規則を用いて依存構造に変換し，精度を計算した．Collins の解析器はそれ自身で品詞タグ付けを行わないため，我々が使用した Nakagawa らの品詞タグづけ手法により品詞タグ付与を行った．MEIP はそれ自身で品詞タグ推定まで行うので，Nakagawa らの手法による品詞タグ付与は行わなかった．

表 7 に比較結果を示す．表 7 で葉ノード精度は，依存木の葉にあたるノードの係り先精度を表す．最も高い精度は MEIP であり，ついで Collins の model 3 であり，本手法は最も低い精度であった．MEIP や Collins のモデルは句構造解析をするために，いずれも句構造情報を使用し学習，および解析を行っている．それに対し我々は単語間の依存構造のみからしか学習できないため，素性として考慮可能な情報に大きな差がある．Penn treebank の句構造では，節や文または動詞句に対し，SBAR，S，VP という異なるラベルが付与されている．しかし依存構造木に変換すると，これらの構造はいずれも動詞をルートに持つ部分木で構成された類似した木構造である．したがってこれらの依存関係を正しく推定することは難しい．

依存木の葉に相当する部分の解析精度は Collins の方法と同等の解析精度で 93.6%であった．符号検定の結果，葉に関する部分の精度は本手法と Collins らの手法とでは有意な差は認められなかった．本手法が他手法に比べルート精度が大きく劣ることからも，依存木のルートに近い部分での解析誤りが精度低下の大きな原因となっている．しかし依存構造木から直接学習する場合，句構造情報に比べ使用できる素性に限りがあることを考慮すれば，90%以上の係り先精度を得たことは十分評価できると考えている．

6. 考　　察

6.1 品詞タグ付けの誤りの影響

品詞タグ付けの精度が依存構造解析の精度にどの程度影響を与えているのかを調査するため，品詞タグ付けを Nakagawa らの手法を用いた場合と，Penn treebank に付与された正解タグを使用する場合とで比較した．実験の設定は最も良い精度を得た 表 6 の 6 actions モデルの *Last-backward* の場合と同じ設定とした．結果を**表 8** に示す．表 8 より，品詞タグ付

☆ 解析時間の計測では Pentium 4 3.02 GHz を搭載した計算機を使用した．

表 8 品詞タグ付け誤りの影響
Table 8 The effect of error in part of speech tagging.

	Nakagawa	完全
係り先精度	0.906	0.919
ルート精度	0.918	0.934
文正解率	0.400	0.423

けが完全であった場合，係り先精度で 1.3%，ルート精度では 1.6%, 文正解率では 2.3%上昇した（すべての評価指標で $p < 0.001$）．本手法は入力文を単語と品詞のペアで表現したものを入力とし解析を行うため，品詞タグ付け精度の影響は後の解析に大きな影響を与える．特に “trading” のような動詞の現在進行形（VBG）と名詞（NN）の曖昧性を持つような単語に関する品詞タグ付け誤りは，品詞の種類によって係り先が大きく異なるため，後の依存構造解析に大きな影響を与える．

Penn treebank に付与された品詞タグは名詞 1 つをとっても，名詞の複数形（NNS），固有名詞（NNP）など，異なるタグで表現されている．しかし依存構造解析では，動詞と名詞の区別が重要で，名詞の詳細な分類が有用でない場合もある．このような詳細な分類タグをそのまま素性に使用すると，まったく異なる素性として扱われるため，正しい学習の妨げとなる．この問題に対処するため，品詞タグをその性質を表す属性の組で表現する必要がある．たとえば名詞であれば，品詞タグ N，固有名詞タグ P，複数形タグ S の 3 つの属性タグの組で表現することで，名詞単数形（NN）は（N, -, - ），固有名詞複数形（NNPS）は（N, P, S），名詞複数形（NNS）は（N, -, S）で表現できる．このような属性で品詞を表現し，各属性を学習の素性として使用すれば，Penn treebank の品詞タグでは異なる素性として扱われていた NN（名詞），NNS（名詞複数形），NNP（固有名詞単数形），および，NNPS（固有名詞複数形）は，名詞という共通な素性を持つことになり，より適切な学習が期待できる．

6.2 誤りの原因

本解析手法は，品詞推定が正しかった場合にでも Charniak の手法に比べ依然として大きな差がある．そこで品詞タグ付け以外の誤りの原因のうち，Charniak の手法で正しく解析でき，本解析手法で誤っている部分について調査した．本解析手法で，テストデータに対する品詞タグ付けを Penn treebank の正解タグを使用した場合にでも，MEIP より精度が劣っている文数は 683 文であった．この中の頻度の高い誤りについて調査した．**表 9** に 683 文中，誤り全体の 2 割以上の高頻出した箇所とそののべ文数を示す．以下に各誤り箇所の主要な原因について述べる．

表 9 解析誤り箇所とその発生文数
Table 9 The types of error in our parser and the number of sentences.

誤り箇所	文数
名詞	328
前置詞	274
動詞	242
副詞	138

名詞の係り先誤り：“,” および “and” などを含み，複数の名詞から構成される名詞句部分の解析に誤りが多い．“,” や “and” は句や節の境界にも使用されるため，短い文脈を考慮しただけでは，名詞句の一部か，節などの境界かを判断するのは難しい．また依存木のルートに近い文や節の主動詞を誤ると，その動詞に係る名詞の係り先も誤ることとなり，ルート精度の向上は大きな課題といえる．

前置詞の係り先誤り：前置詞の係り先決定は本質的な曖昧性を含む，難しい問題である．多くの場合，文中に現れた単語の表層的情報だけでは解決できない．前置詞句の係り先推定には前置詞に係る単語の意味情報が重要となる．前置詞に係る単語をクラスタリングし，前置詞に意味的に近い単語が係っているかどうかを考慮する必要がある．

動詞の係り先誤り：動詞の係り先を誤っているほとんどが，複数の節からなる文，挿入句のある文などで，文の主辞となる主動詞を誤っていることが原因である．これは表 7 にも示したとおり，我々の手法はルート精度が低く，ルートとなる動詞の係り先を誤ると，他の節の主辞となる動詞の係り先も誤ることとなり，大きな精度低下につながる．

副詞の係り先誤り：副詞の係り先の誤りの原因は，Penn treebank にタグ付けされた句構造に揺れがあるためである．たとえば “rather” と “than” がまとまってできる句として前置詞句（PP）と Multiword conjunction phrase（CONJP）という揺れがある．これら 2 つの句に対する主辞規則が異なるため，異なる依存構造に変換される．しかし依存構造に変換した場合，句の情報は失われるため，この 2 つを区別することは難しい．また副詞以外にも訓練データのタグ付けの揺れが原因となっているものがあった．“only about 2% to 3 %” という単語列が 1 つにまとまって quantifier phrase（QP）となり句構造がタグ付けされている一方で，“only 10%” という単語列に対しては，“only” と “10” がまとまり QP となり，その QP と “%” がまとまって名詞句のタグが付与されている．

このようなタグ付けの非一貫性は学習のノイズとなり，解析誤りの原因の 1 つになる．

本解析手法では，各解析アクションの推定時に，限られた文脈長の情報しか使用していないため，その文脈長外に重要な素性がある場合，正しい解析ができない．その結果，複数の名詞からなる名詞句，および複文におけるルートなど，短い文脈を考慮しただけではうまく扱えない部分が誤りの主要部分となっている．しかし不要に考慮する文脈長を伸ばすのは，学習時のデータ過疎性という点からも現実的でない．実際表 4 に示したとおり，文脈を伸ばすだけでは精度を向上することはできなかった．これらの問題に対処するには，考慮した文脈外に係り先の候補があるか否かを，あらかじめ別の手法で推定しておき，それを素性として考慮することが有用だと考えている．ある高い確率で係り先になりうるものが文脈外に存在することを素性として考慮できれば，*Shift* して解析を遅らせる重要な手がかりとなる．

7. 関連研究

英語の依存構造解析に Link grammar[2] がある．Link grammar は単語間の依存関係（link）と，主語（Subject）や目的語（Object）といった依存関係の種類（connector）の両方を解析する．本論文では Link grammar の link に該当する部分の解析のみを扱ったが，*Right-Subject* や *Left-Object* のように解析アクションに connector 情報を付与することで，Link grammar と同等な解析を実現可能である．

また英語の統計的な依存構造解析手法には Eisner の手法[6),7)] がある．Eisner は我々と同様に，依存構造木からの情報のみを使用した確率モデルに基づく解析手法を提案している．Eisner も Penn treebank を使用した実験について報告しているが，使用している訓練データおよびテストデータサイズが異なることと，接続詞を含むような比較的長い文は評価の対象としていないことから，直接精度を比較することはできない．しかし Eisner は Collins が提案した解析器[8)] との比較を行っている．Eisner が比較対象とした Collins の解析器は我々が比較した Collins の解析器[9)] よりも古いモデルで，解析精度も低い．Eisenr の報告によれば Collins 解析器が係り先精度で 92.6%であるデータに対し，彼の手法では 90.0%と報告している．テストデータ数が 400 文と少ないこと，また接続詞を含まないような比較的短い文しか評価していないこと，さらに我々の実験では Eisner の比較した Collins の手法よりも良い精度の解析手法でも 91.5%の係り先精度しか達していないことを考えると，比較的簡単な文に対する解析結果であることが予想される．したがって我々と同様の実験データで比較した場合，Eisner の手法の精度は我々よりも劣ることが予測できる．

8. まとめ

本論文では，様々な分野において現実的に解析済み訓練データを用意可能という観点から，英語の依存構造に注目し，機械学習法 SVM を用いた決定性上昇型依存構造手法を提案した．Penn treebank の句構造解析済データを依存構造に変換し，評価実験を行った結果，句構造情報を使用せず，90%を超える高い解析精度を達成することができた．

今後は，解析精度のさらなる向上を行う．品詞タグ推定の解析誤りが依存構造解析に大きな影響を与えるため，品詞タグ推定誤りが依存構造解析に及ぼす影響を軽減する必要がある．Penn treebank の品詞タグを名詞や動詞といった大きなグループに変換し，単数形や過去形など詳細な情報は属性として表現し素性に考慮することで，より適切な学習が可能となり精度改善に貢献できると考えている．また本手法は他手法に比べルート精度が低い．したがってルート精度の改善は大きな課題といえる．限られた長さの文脈情報だけでは，複文構造などのルートを正確にとらえることは難しい．ルートノードの候補をあらかじめ別の方法で推定し素性に考慮することで，ルート精度を改善する予定である．

また本手法を他分野へ適用し，依存構造解析の重要性および頑健性について検証する必要がある．本論文の実験で使用した Penn treebank は経済分野である Wall Street Journal に構文情報がタグ付けされているコーパスである．これとは異なった医学生物学分野の文献アブストラクト Medline[1)] を対象に，本解析手法を適用し，対象分野が大きく変化した場合の性能について調査する予定である．

謝辞 品詞タグ付けのツールを提供してくださった，沖電気株式会社，中川哲治氏に感謝いたします．また論文執筆にあったて，議論および様々な助言をくださった NTT コミュニケーション科学基礎研究所，工藤拓氏，東京工業大学，高村大也助手に感謝いたします

参考文献

1) Internet Grateful Med Development Team, National Libarary of Medline: MEDLINE (1996).
2) Sleator, D. and Temperley, D.: Parsing En-

glish with a Link Grammar, Technical report, Technical Report CMU-CS-91-196, Carnegie Mellon University (1991).
3) Magerman, D.M.: Statistical decision-tree models for parsing, *Proc. 33rd Annual Meeting of the Association for Computational Linguistics*, pp.276–283 (1995).
4) Lin, D.: Using Syntactic Dependency as Local Context to Resolve Word Sense Ambiguity, *Proc. 35th Annual Meeting of the Association for Computational Linguistics*, pp.64–71 (1997).
5) Charniak, E.: A Maximum-Entropy-Inspired Parser, *Proc. 2nd Meeting of North American Chapter of Association for Computational Linguistics* (*NAACL2000*), pp.132–139 (2000).
6) Eisner, J.: An Empirical Comparison of Probability Models for Dependency Grammar, Technical report, IRCS-96-11, IRCS, Univ. of Pennsylvania (1996).
7) Eisner, J.: Three New Probabilistic Models for Dependency Parsing: An Exploration, *Proc. 16th International Conference on Computational Linguistics* (*COLING-96*), pp.340–345 (1996).
8) Collins, M.: A New Statistical Parser Based on Bigram Lexical Dependencies, *Proc. 34th Annual Meeting of the Association for Computational Linguistics*, pp.184–191 (1996).
9) Collins, M.: Three Generative, Lexicalised Models for Statistical Parsing, *Proc. 35th Annual Meeting of the Association for Computational Linguistics* (*jointly with the 8th Conference of the EACL*), pp.16–23 (1997).
10) 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).
11) Nakagawa, T., Kudoh, T. and Matsumoto, Y.: Revision Learning and its Application to Part-of-Speech Tagging, *Proc. Association for Computational Linguistics*, pp.497–504 (2002).
12) Ratnaparkhi, A.: Learning to Parse Natural Language with Maximum Entropy Models, *Machine Learning*, Vol.34, No.1-3, pp.151–175 (1999).
13) KreBel, U.H.-G.: *Advances in Kernel Methods*, chapter Pairwise Classification and Support Vector Machines, MIT Press (1999).
14) Vapnik, V.N.: *Statistical Learning Theory*, A Wiley-Interscience Publication (1998).
15) 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).
16) 山田寛康, 新保 仁, 松本裕治：文脈情報を用いた医学用語分類, 情報処理学会研究会報告, 知能と複雑系研究会 ICS-128-5, pp.23–28 (2002).

(平成 15 年 4 月 17 日受付)
(平成 16 年 9 月 3 日採録)

山田 寛康

1974 年生．1997 年山梨大学工学部電子情報工学科卒業．1999 年同大学大学院工学研究科博士前期課程修了．2002 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．同年北陸先端科学技術大学院大学助手．自然言語処理の研究に従事．

松本 裕治（正会員）

1955 年生．1977 年京都大学工学部情報工学科卒業．1979 年同大学大学院工学研究科修士課程情報工学専攻修了．同年電子技術総合研究所入所．1984 年〜1985 年英国インペリアルカレッジ客員研究員．1985 年〜1987 年（財）新世代コンピュータ技術開発機構に出向．京都大学助教授を経て，1993 年より奈良先端科学技術大学院大学教授，現在に至る．工学博士．専門は自然言語処理．人工知能学会，日本ソフトウェア科学会，言語処理学会，認知科学会，AAAI，ACL，ACM 各会員．