自 然 言 語 処 理 152－8
（2002．11．12）

Support Vector Machine を用いた英語依存構造解析

山田 寛康†，松本 裕治††
†北陸先端科学技術大学院大学 情報科学研究科
††奈良先端科学技術大学院大学 情報科学研究科
h-yamada@jaist.ac.jp, matsu@is.aist-nara.ac.jp

本稿では，機械学習アルゴリズム Support Vector Machine(SVM) を用いた英語の依存構造解析法を提案する．統計的構文解析を様々な分野のテキストに適用する場合，適用分野毎にある程度の構文解析済み訓練データを用意する必要がある．Penn Treebank のような複雑な句構造木は，そのタグ付け作業に文法的な専門知識を必要とするため，タグ付け作業者が限定される．その結果，様々な分野で訓練データを用意することが現実的に困難となる．依存構造は句構造木のような複雑さは大幅に削減されるため，対象言語を母国語とする多くの人が，タグ付け可能となる．また，構造が簡潔なためタグ付け作業者間での揺れを軽減し，様々な分野で高品質の訓練データを作成することが期待できる．そこで，直接依存構造を学習する高精度な依存構造解析器を構築することが重要となる．提案する依存構造解析手法は，3 種類の手続きを用いて決定的に依存木を構築する．また解析の各段階で，適切な手続きを推定するために SVM を用いて学習を行う．Penn Treebank を主辞規則を用いて依存木に変換し評価実験を行った結果，係り先精度で 88.4% という結果を得た．

キーワード：統計的構文解析，決定性構文解析，依存構造解析，サポートベクター学習

Dependency Analysis of English Sentences with Support Vector Machines

YAMADA Hiroyasu †，MATSUMOTO Yuji ††
†Japan Advanced Institute of Science and Technology
††Graduate School of Information Science, Nara Institute Science and Technology
h-yamada@jaist.ac.jp, matsu@is.aist-nara.ac.jp

In this paper, we propose a method for dependency analysis of English sentences using Support Vector Machines(SVMs). In order to build statistical parser with high accuracy in various domains, we have to prepare parsed training data annotated by human. However, to annotate sentences in a phrase structure format like Penn Treebank, the requirement for annotators is to have profound linguistic knowledge about the target language. As a result, it is difficult to build large size of training data annotated with phrase structure such as Penn Treebank. Annotating word-word dependencies in sentences is easier than phrase structures annotation for many native speakers of the target language. The conciseness of dependencies as sentence structure is useful for reaching a consensus among annotators. It is is expected that we can make a large amount training data with less inconsistency. Therefore, it is important to develop the statistical parser which can directly analyze dependencies of words in a sentence. Our parser deterministically constructs dependency trees using three types of parsing actions, and uses SVMs for estimating the correct parsing action at each parsing step. In our experiments using Penn Treebank converted into dependency trees, the results show that our parser achieves dependency accuracy of 88.4% for the Penn Treebank standard data set.

Keywords : Statistical Parsing, Deterministic Parsing, Dependency Analysis, Support Vector Learning

1 はじめに

構文解析は, 自然言語処理における基礎技術の一つであり, 機械翻訳や情報検索など幅広い応用に使用されるため, 汎用的で高精度な解析を実現する必要がある. 統計的構文解析は, 解析済コーパスから, 解析に必要な規則を自動学習することで, 高い汎用性と解析精度を実現する. 近年, より高い精度で解析するために, 人手で解析された訓練コーパスから教師有り学習を用いて解析規則を自動学習する研究が多く行われている [3, 10, 7]. これらの研究の多くは Penn Treebank[8] の句構造木を学習データとして使用している. Penn Treebank から学習した構文解析器を使用し, 多分野へ適用することは可能であるが, 高精度の解析を実現するには, 適用分野毎にある程度の学習データを用意する必要がある.

しかし Penn Treebank には 品詞タグ以外に 26 種類のラベル (句) から構成される複雑な句構造解析木がタグ付けされており, これらのタグ付け作業を行う人は, 英語を母国語としていることだけでなく, 英文法に関する専門的な知識を持っている必要がある. また適用分野が, 新聞記事のような一般的な分野ではなく, 医学生物学分野のような専門性の高い分野では, 言語に関する知識でだけでなく, その分野に関する専門知識を必要する可能性もあり, タグ付けを行うことができる人の数は更に少なくなる. このような状況を考えると, 様々な分野のテキストに対して, Penn Treebank のような複雑な句構造をタグ付けし, 訓練データを準備することは現実的でない. また複雑な構造のタグ付けは, タグ付け者間での一貫性を保持することが難しくなる. データ中のタグ付けに一貫性がない場合, 学習のノイズとなり解析精度を低下させる原因となる.

依存構造は, 語間の修飾関係で文の構造を表現した簡潔なもので, 句構造にみられる複雑さは大幅に軽減される. 従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る. また簡潔な構造であるために, タグ付け者間で揺れが小さく一貫性の保たれた高品質の訓練データを作成することが期待できる.

また依存構造解析結果を応用した研究として, Lin は依存構造を利用して単語の語義曖昧性解消を行っている [2]. 我々は以前に医学生物学用語の単語クラス分類タスクに依存関係を利用し, その有効性を確認している [14]. 応用を考えた場合にでも, 依存構造は十分利用価値がある情報だと考える.

我々は, 依存構造* であれば, 様々な分野においても高品質な訓練データが現実的に作成可能であると考える. 依存木からは句構造木を構築することができないため, 直接依存構造を解析する手法が必要となる. 本稿では 3 種類の解析手続きを用いて決定的に依存木を構築する手法を提案する. 各解析過程において, 適切な解析手続きを推定するために, 機械学習アルゴリズム Support Vector Machine を用いてを学習を行う. また句構造木がタグ付けされた Penn Treebank を主辞規則を用いて依存木に変換し, このデータを用いた評価実験について報告する.

以下, 次章では, 機械学習アルゴリズム Support Vector Machine についての簡単に説明する. 次に 3 章では. 依存構造解析を行うアルゴリズムについて述べ, 4 章では依存構造を SVM を用いて学習する方法について説明する. 5 章で Penn Treebank を用いた評価実験について報告する. 最後に 6 章で, まとめと今後の課題について述べる.

*本稿で扱う依存構造は単語同士の係り関係にとどめる. Link grammer [1] のような主語, 述語などの関係については扱わない. しかし, このような関係も容易にアルゴリズムに取り込み拡張可能である.

2 Support Vector Machine

Support Vector Machine (SVM)[12] はマージン最大化に基づく二値分類器であり, 素性空間の次元数に依存しない高い汎化性能は, 多くの自然言語処理タスクにおいて, その有効性が報告されている [11, 13, 9]. 依存構造解析では, 単語自身や品詞が学習の重要な素性となる. 従って素性空間は数万以上の高次元空間なることから, 過学習を軽減することが可能な SVM を使用する.

SVM は, n 次元ベクトル空間上に分布する l 個の正・負を表す訓練事例 $(\mathbf{x}_i, y_i), (i \leq l, \mathbf{x}_i \in \mathbf{R}^n, y_i \in \{+1, -1\})$ から, 与えられた正・負事例を正しく分離する超平面 $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b (\mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R})$ を求めるアルゴリズムである. SVM によって求めた分離平面は, 最終的に次式で表され, 未知のデータ $\mathbf{x}$ に関する分類結果は 関数 $f(\mathbf{x})$ の符号により決定される.

$$f(\mathbf{x}) = sign\left(\sum_{i}^{l} \alpha_i y_i \mathrm{K}(\mathbf{x}_i, \mathbf{x}) + b\right)$$

ここで α_i は, 超平面を求めるため, 最適化問題を解くとき使用するラグランジュ乗数であり, $\mathrm{K}(\mathbf{a}, \mathbf{b})$ は Kernel 関数で, 引数である 2 つのベクトルの類似度を計算する関数である. SVM は Kernel 関数を適切に選択することで非線型な学習も可能となる. 本稿では Kernel 関数として多項式関数 $(\mathbf{a} \cdot \mathbf{b} + 1)^d$ を用いた. これは素性の d 個までの組み合わせを考慮した学習が計算時間を大幅に増加させることなく実現できるためである. また依存構造解析では, 複数のクラスへの分類が必要となるため, 二値分類器である SVM を one-versus-rest 法を使用し

多値分類へ拡張した.

3 依存構造解析アルゴリズム

本稿で提案する依存構造解析アルゴリズムについて説明する. 提案する解析アルゴリズムは, 以下に示す3つの手続き Left, Right, 及び Shift を繰り返し適用し, 決定的に依存木を構築していく. 図 1, 2, 及び 3 にそれぞれの手続きの例を示す. 図において波線で囲まれた2つのノードが現在注目している解析位置を表す. また図の上方が手続き適用前を表し, 下方が手続きを適用した結果を表す.

- Left : 隣接した2つのノード間で, 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 1).

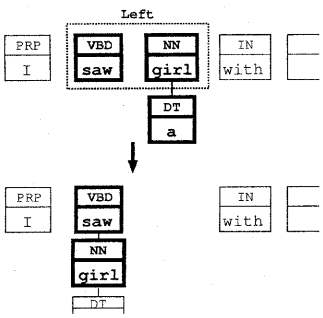

図 1: 手続き 'Left'

- Right : 隣接した2つのノード間で, 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 2).

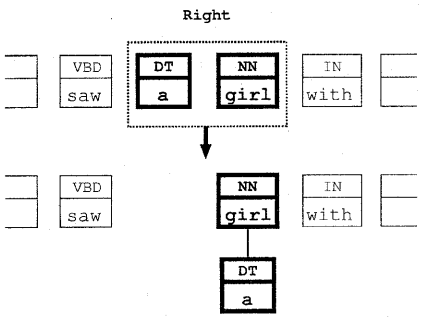

図 2: 手続き 'Right'

- Shift : 解析位置を一つ移動する. 即ちこの時点では依存木を構築せず, 次のノード間の解析に移る (図 3).

図4に3つの手続きを使用した解析アルゴリズムの疑似コードを示す. アルゴリズムへの入力は, 入力文 n 単語

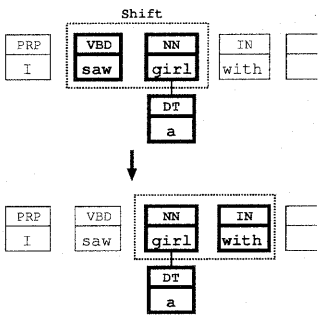

図 3: 手続き 'Shift'

に対し品詞が付与された単語列 $c_1, c_2, \cdots, c_n$ とする. 図4で, s は解析された部分依存木のルートノードを保存する配列変数である. 解析の初期では, 入力単語列である n 個のノードが s にセットされる. $|s|$ は解析された部分木数を表し, s_i は s の i 番目の部分木を表す. 関数 $get\_contextual\_feature(s,i)$ は, i 番目の解析位置において s_i の周りの文脈から素性を抽出し, その素性ベクトル $\mathbf{x}$ を返すものである. 具体的な素性については 4.3 節で述べる. $model$ は学習した SVM を表し, 関数 $classify$ は, $model$ と関数 $get\_contextual\_information$ で抽出した素性ベクトル $\mathbf{x}$ から, 現在の解析位置における適切な解析手続き $y \in \{\text{Left}, \text{Right}, \text{Shift}\}$ を推定する. $construction$ は解析手続き y と s を引数に, 実際に手続きを実行し, s に依存木を構築する.

解析は, 文頭もしくは文末から順に 隣接した2つのノード間の依存関係を, 3つの手続きにより決定していく. 依存関係が決定した時点で, 係り先のノード (親ノード) のみが解析対象ノードとなり s に残される. 解析位置が文末に達した場合は, 文頭に戻り同様の動作を繰り返す. 最終的に1つのノード (ルートノード) が決定されるか, または解析位置 i が $i = |s|$ となるまでに, 依存木が何も構築されなかった場合 (変数 no_construction が文末に到達した時点で true であった場合) に解析終了となる.

4 SVMによる依存構造の学習

4.1 訓練事例

訓練時は, 依存構造解析済みの訓練文に対し, 図4に示したアルゴリズムを使用し, 依存構造解析を行う. ここで $model$ は SVM の代わりに訓練文にタグ付けされた正解解析木とする. 従って関数 $classify$ による解析手続き y の推定は, 正解解析木から, 正しい解析手続きが選択される. このようにして解析の各ステップで得ら

Input Sentence: $c_1, c_2, \cdots, c_i, \cdots c_n$
Initialize:
  $i = 1$; $s = c_1, c_2, \cdots, c_i, \cdots c_n$
  no_construction = *true* ;
Start:
while *true* do begin
  if $i == |s|$ then
    if no_construction == *true* then break;
    no_construction = *true*
    $i = 1$;
  else
    $\mathbf{x} = get\_contextual\_features(s, i)$;
    $y = classify(model, \mathbf{x})$;
    if $y ==$ **Left** or **Right** then
      $s = construction(s, i, y)$;
      no_construction = *false* ;
    else if $y ==$ **Shift** then
      $i = i + 1$
    end;
  end;
end;

図 4: 依存構造解析アルゴリズム疑似コード

れた, 素性ベクトル $\mathbf{x}$ と解析手続き y とのペア $(\mathbf{x}, y)$ が, 一つの事例に対応する.

本稿で提案する依存構造解析アルゴリズムは, 決定的に依存木を構築するため, 訓練時に 2 つのノードに依存関係を構築する Left 及び Right の解析手続きを選択する場合は, 次の注意が必要である. 隣接した 2 つのノード間に Left 及び Right のいずれかの解析手続きを選択する場合は, 係り元となるノードは, 他のどのノードからも係られないことが必要である. 英語は日本語と異なり, 左右どちらからも係り関係が存在する. このため隣接ノード間に依存関係がある場合にでも, 係り元ノードに他から係り得るノードが存在する場合, 先にその係り関係を決定させる必要がある.

4.2 文脈長

学習の素性として考慮する文脈について説明する. 解析は前後の文脈から, 適切な解析手続き y を推定する. このとき考慮する文脈の長さを, 文脈長とよび, 解析位置より左にある文脈を左文脈, 解析位置より右にある文脈を右文脈と呼ぶ. 学習に使用する素性は, 文脈長内にある各ノードから抽出する. また本稿で "文脈長 (l, r)" と表記した場合, 左右の文脈長がそれぞれ l, r であることを意味する.

4.3 素性

素性には, 考慮する文脈にある各ノードから, 単語や品詞, 各ノードの係り元情報を用いる. 素性は, 現在の解析対象としているノードからの位置 $p \in \{\cdots, -2, -1, 0^-, 0^+, 1, 2, \cdots\}$, 素性の種類 $t \in$ {pos, lex, pos(L), lex(L), pos(R), lex(R)}, 及び素性の値 $v \in$ {NN, VBZ, $\cdots$, a, small, telescope, $\cdots$} の 3 つ組み $p.t.v$ で表現される. $p < 0$ である場合その素性は解析位置より左に $|p|$ 番目のノードを表し, $p > 0$ なら右に p 番目のノードを表す. 0^-, 0^+ は隣接する解析対象の 2 ノード中, それぞれ, 左ノード, 右ノードを表す. 素性の種類 t が pos であれば品詞を表し, lex は単語を表す. pos(L) は係り元ノードのうち, 左から係るノードの品詞を表し pos(R) は右から係るノードの品詞を表す. 同様に lex(L), lex(R) は, 係り元ノードのうちそれぞれ左から係るノードの単語, 右から係る単語を表す. 素性の値 v は 素性の種類 t が pos, pos(L), 及び pos(R) であれば NN, VBD, IN など品詞文字列が値となり, t が lex, lex(L), 及び lex(R) であれば, 単語自身の文字列が値となる. 図 5 に左右の文脈長をそれぞれ 1 とした場合に, 使用される素性の例を示す.

4.4 訓練事例のグループ化による大規模な学習の計算量軽減

訓練事例は, 訓練文を依存構造解析する 1 ステップが一つの事例に対応する. そのため訓練事例数は, Penn Treebank の訓練セット section 02 から section 21 の約 40,000 文 では 1,400,000 以上となり, SVM を用いて一度にすべての訓練事例を学習することが難しい. そこで訓練データを, 解析位置における品詞素性 0^-.pos の値 (品詞) 毎に分割し, 分割した個々のグループで学習を行う†. 解析位置における品詞素性 0^-.pos の値が, POS である事例集合から学習した SVM を SVM_{POS} とする. テストデータの解析時のあるステップで, 品詞素性 0^-.pos の値が 品詞 POS' であった場合, 対応する学習モデル $\mathrm{SVM}_{POS'}$ を用いて解析手続き y を推定する.

†今回 0^+.pos ではなく 0^-.pos を利用してグループ化しのは暫定的なものである

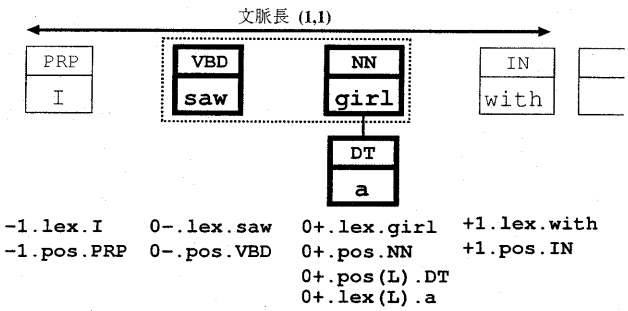

図 5: 使用する素性の例

5 実験

5.1 データ

実験では, Penn Treebank の標準データセットである section 02 から 21 を訓練データ, section 23 をテストデータとして使用した. Penn Treebank は句構造であるため, これを依存木に変換する必要がある. これは Collins [6, 7] と同様, 主辞規則を用いて, 各句の主辞を決定し, 非主辞ノードが主辞ノードに係るという処理により依存木に変換した.

評価

依存構造解析器の性能を評価するために, 次に定義する 3 つの指標を用いた.

係り先精度 (Dep. Acc.) = 正解係り先数/総係り先数
ルート精度 (Root Acc.) = ルート正解数/総文数
文正解率 (Comp. Rate) = 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. **完全正回数**は, 一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.

品詞タグ付け

品詞タグ付けには中川らの修正学習法による品詞タグ付け手法 [9] を使用した. 実験では section 02 から 21 で学習を行ったモデルにより品詞タグ付けを行った.

5.2 予備実験

実装した依存構造解析器の性能を調査するため, いくつかの予備実験を行った. SVM の学習には多大な時間を要するため, 予備実験での訓練文は section 02 から 06 の 9,695 文を使用した. また解析する方向を左から右にした場合 (前向き解析) と, 右から左 (後ろ向き) で解析する場合とでは, 右から左に解析した場合の方が訓練事例数が少ない. そこで学習時間を軽減するため, 予備実験では後ろ向き解析を中心に実験を行った.

多項式 Kernel 関数の次数

先ず, SVM で使用する 多項式 kernel 関数の次数による精度の変化を調査した. 使用した多項式 kernel 関数 $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$ の次数 d を 1 から 4 まで変化させ実験を行った. 表 1 に結果を示す.

表 1: 多項式 Kernel の次数の変化と解析精度

	$d : (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$			
	1	2	3	4
Dep. Acc.	0.798	**0.853**	0.842	0.807
Root Acc.	0.711	**0.819**	0.798	0.734
Comp. Rate	0.206	**0.290**	0.276	0.248

訓練データ: section 02-06 (9,695 文), テストデータ: section 23(2,416), 文脈長 (2,2), 素性:各ノードの単語及び品詞, 係り元の単語及び品詞, 解析方向:後ろ向き

多項式 Kernel 関数 $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$ の d を変化させることは, 学習に d 個までの素性の組み合わせを考慮することと等価である. $d = 1$ と $d > 1$ とでは精度に大きな差があり, 依存構造の学習には素性の組み合わせを考慮する必要があることがわかった.

文脈長の違いによる精度の変化

次に, 素性として考慮する文脈長を変化させることで, 解析精度にどの程度影響があるか調査した. 結果を表2に示す. 表2の数字は, 左右の文脈長をそれぞれ1から4まで変化させたときの係り先精度を示す. 文脈長(3,2)で最も高い精度を得た.

表 2: 文脈長と解析精度:(係り先精度)

		右文脈			
		1	2	3	4
左文脈	1	0.842	0.848	0.848	0.845
	2	0.851	0.853	0.853	0.849
	3	0.850	**0.854**	0.851	0.849
	4	0.847	0.852	0.848	0.847

訓練データ: section 02-06 (9,695 文), テストデータ: section 23(2,416), Kernel 関数: $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$, 素性:各ノードの単語及び品詞, 係り元の単語及び品詞, 解析方向:後ろ向き

解析方向の違いによる解析精度

解析する方向を, 左から右へむけて解析する (前向き) 場合と, 右から左へ解析する (後ろ向き) 場合とを比較した. 結果を表3に示す.

表 3: 解析方向と精度

	前向き	後ろ向き
Dep. Acc.	0.846	**0.853**
Root Acc.	0.791	**0.819**
Comp. Rate	0.268	**0.290**

訓練データ: section 02-06 (9,695 文), テストデータ: section 23(2,416), Kernel 関数: $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$, 文脈長 (2,2), 素性:各ノードの単語及び品詞, 係り元の単語及び品詞

同じ素性, 同じ文脈長では後ろ向き解析のほうが精度が高いことがわかる. これは英語は右から左に修飾する場合が多いため, 後ろから依存関係を解析することで, 推定した依存関係を後の解析に利用できるためであろう.

素性の違いによる精度の変化

次に使用する素性の違いによる解析精度を調査した. 工藤らは, 日本語係り受け解析において, 動的素性と呼ばれる, 解析の過程で決定された係り関係を使用することで, 精度が向上すると報告している [11, 13]. 本稿でも動的素性にあたる係り元情報を使用している. そこでこの係り元情報が解析精度にどの程度影響を与えるかを調べるために, 文脈中ノードの単語及び品詞情報に加え, 次の5種類の素性セットによる実験を行った.

(1) 係り元情報を使用しない
(2) 係り元情報のうち単語情報のみを使用
(3) 係り元情報のうち品詞情報のみを使用
(4) 係り元情報のうち品詞情報のみを使用, 但し係り先の品詞が前置詞 IN の場合は単語情報も使用
(5) 係り元情報すべてを使用

表 4: 素性の種類と解析精度

	素性の種類				
	(1)	(2)	(3)	(4)	(5)
Dep. Acc.	0.829	0.855	0.857	**0.858**	0.853
Root Acc.	0.784	0.831	0.837	**0.839**	0.819
Comp. Rate	0.237	0.289	0.285	0.286	**0.290**

訓練データ: section 02-06 (9,695 文), テストデータ: section 23(2,416), Kernel 関数: $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$, 文脈長 (2,2)

結果を表4に示す. 係り元情報を使用しない (1) は係り元情報を使用する (2) から (5) より大きく精度が劣っている. 決定的に解析した場合にも, 解析の過程で決定された係り元情報を動的に使用することが, 精度向上に大きく貢献することがわかった.

5.3 比較実験

現在 Penn Treebank のテストセットに対して最も高い精度が報告されている Charniak の Maximum Entropy Inspired Parser[3] (以下 MEIP) を比較対象とした. テスト文に対し MEIP を用いて句構造解析した結果を, 主辞規則[‡]を用いて依存木に変換し, 精度を計算した. 結果を表5に示す.

評価は, Penn Treebank の句構造木から依存木に変換し, それを正解としている. そのため Penn Treebank の句構造を捕らえることができる MEIP は, テストデータに対して係り先精度で 91.4%という高精度で解析できて

[‡]Charniak は Penn Treebank オリジナルの動詞に関して助動詞 have などを動詞と区別するために AUX, AUXG という新たな品詞タグ追加している. そのため動詞句 VP に関する主辞規則を MEIP 用に拡張した.

表 5: Charniak 00 との比較

	Charniak 00		Our parser	
	02-21	23	02-21	23
Dep. Acc.	0.955	0.914	0.998	0.884
Root Acc.	0.969	0.952	0.997	0.877
Comp. Rate	0.680	0.430	0.984	0.339

Kernel 関数: $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$, 文 脈長 (2,2), 素性: 表 4 の (4) と同一

いる. 一方, 本手法は, テストデータに対しては MEIP に比べ 3% 精度が低い. しかし句構造情報をまったく使用せず, 単語と品詞, 及び解析過程で決定した係り元情報のみで, この精度に達したことは良い結果であると考えている.

また, 訓練データに対しては, 学習モデルの差がそのまま結果に現れている. MEIP が使用している Maximum Entropy モデルは, 訓練データに対し 95 % 程度の係り先精度である. このため, 一文が完全に解析できているかを示す文正解率の値は 7 割を下回っている. 本稿で採用した SVM では, 訓練データに対して 99.8% の係り先精度に達しており, ほぼ完璧に解析できている. そのため文正解率も 98.4 % と非常に高い.

誤りの原因

- **品詞推定の誤りによる影響**

  本稿で提案した解析アルゴリズムは, 決定的に依存木を構築する. また学習の効率化のために, 訓練データを, 解析位置での品詞ごとにグループ化し個別に学習をしている. そのため解析の入力となる品詞タグ付けの精度は, 依存構造解析に大きな影響を及ぼす. そこで品詞タグ付けの誤りが解析精度にどの程度の影響を与えるかを調べるために, 品詞タグ付けが完全に正しい場合と比較を行った. 結果を表 6 に示す.

表 6: 品詞タグ付け精度の影響

	推定	完全
Dep. Acc.	0.884	**0.899**
Root Acc.	0.877	**0.899**
Comp. Rate	0.339	**0.370**

結果から, 品詞推定の誤りが解析精度に影響を与えることがわかる. 訓練事例を品詞素性毎にグループ化することで大規模な学習はできるが, 品詞推定を誤ると, 適切な学習モデルを使用して解析手続きを推定できず, 解析誤りに直結する. 今後, 精度改善のためには品詞以外の別の観点でグループ化するなど, 品詞推定の影響を軽減する方法を検討する必要がある.

- **前置詞句の係り先の誤り**

  前置詞句の係り先決定は本質的に難しい問題である. しかし本手法は, MEIP と比べ係り先が離れている場合や長い文において, 推定を誤る. MEIP は句構造木の トップダウンからの情報を利用しているため, 係り先が離れている場合にでも比較的誤りが少ない.

  それに対して, 本手法は, 解析位置から限られた長さの文脈情報で依存関係を推定していることが原因として考えられる. しかし考慮する文脈を不要に長くすることは逆に精度の低下につながる. 前置詞句の係り先を推定する場合には長め文脈を考慮するなど, 考慮する文脈を可変長にするなどの対処が必要である.

- **挿入文, 複文, 重文の誤り**

  本手法では, 挿入文や, 複数の節からなる比較的長い文に, 誤りが多い. ルートとなるノードを誤って推定したり, 解析が途中で終了している場合もある. また複数ある節内の依存関係は正しいが, 節間の依存関係を正しく推定できていない場合も多い. MEIP に比べルート精度が 8% と近く劣っていることからも, トップダウン情報を解析に考慮する必要がある. また複数の節間の依存関係を決定的に解析していくことにも限度があり, 今後, 採用した決定的な解析方法を冗長に解析する方法へ拡張することも検討していきたい.

関連研究

Eisner は CKY 法に類似した上昇型解析アルゴリズムと, 確率モデルを使用した依存構造解析法を提案している [5]. 更に Eisner は文献 [4] で, Penn Treebank を用いた大規模な実験を行っており, 係り先精度が 90%に達したと報告している. しかし Eisner らの結果と我々の実験結果とでは, 次の 3 つ点から平等に比較できない.

(1) 訓練とテストに用いているデータサイズが異なる
(2) 接続詞を含む文を解析対象外としている
(3) punctuation を評価対象にしていない

特に (2) は, 我々の行った実験よりも簡単なタスクでの評価と言える. また我々も (3) と同様 punctuation を評価対象から除外した精度を計算したところ 89.5 % とい

う係り先精度を得た．従ってタスクの難しを考慮すれば，本手法は Eisner の手法と同等以上であることが期待できる．Eisner の方法は，我々と同様に句構造情報を使用せず，直接依存構造解析できる点では MEIP より適切な比較対象と言える．また採用している CKY 法に類似したアルゴリズムは，今後，本手法を冗長解析させる拡張方法の参考にしたい．

6 まとめと今後の課題

本稿では，様々な分野において，統計的構文解析の実現を目標に，現実的に訓練データを作成可能な依存構造に注目した．そして SVM を用いて直接依存構造を解析する手法を提案した．提案手法は 3 種類の解析手続きを用いて決定的に依存木を構築し，各解析過程を文脈からその文脈に適した解析手続きへの分類問題とみなし，SVM を適用した．また Penn Treebank を主辞規則を用いて依存木に変換し，評価実験を行った．実験では，Penn Treebank の section 23 のテストデータに対し，88.4% の係り先精度を得た．決定的に解析する方法でも，解析の過程で決定された依存関係を動的に素性として利用することが精度向上に貢献することを確認した．

今後，精度向上のために，調査したいくつかの誤りの原因について対処する必要がある．また構築した依存構造解析木を，他分野のテキスト，特に医学生物学分野のテキストに適用し，解析器の頑健性について検証する予定である．

謝辞

修正学習法による品詞タグ付けの学習 ツールを提供して下さった中川哲治氏に心から感謝致します．

参考文献

[1] Daniel Sleator and Davy Temperley. Parsing English with a Link Grammar. Technical report, Technical Report CMU-CS-91-196, Carnegie Mellon University, October 1991.

[2] Dekang Lin. Using Syntactic Dependency as Local Context to Resolve Word Sense Ambiguity. In *Proceedings of 35th Annual Meeting of the Association for Computational Linguistics*, pp. 64–71, 1997.

[3] Eugene Charniak. A Maximum-Entropy-Inspired Parser. In *Proceedings of the Second Meeting of North American Chapter of Association for Computational Linguistics (NAACL2000)*, pp. 132–139, 2000.

[4] Jason Eisner. An Empirical Comparison of Probability Models for Dependency Grammar. Technical report, IRCS-96-11, IRCS, Univ. of Pennsylvania, 1996.

[5] Jason Eisner. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pp. 340–345, 1996.

[6] Michael Collins. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 184–191, 1996.

[7] Michael Collins. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (jointly with the 8th Conference of the EACL)*, pp. 16–23, 1997.

[8] Mitchell P. Marcus and Beatrice Santorini and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, Vol. 19, No. 2, pp. 313–330, 1993.

[9] Tetsuji Nakagawa, Taku Kudoh, and Yuji Matsumoto. Revision learning and its application to part-of-speech tagging. In *Proceedings of Association for Computational Linguistics*, pp. 497–504, 2002.

[10] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, Vol. 34, No. 1-3, pp. 151–175, 1999.

[11] Taku Kudo and Yuji Matsumoto. Japanese Dependency Analysis using Cascaded Chunking. In *Proceedings of Sixth Conference on Natural Language Learning (CoNLL-2002)*, pp. 63–69, 2002.

[12] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. New York, 1995.

[13] 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, June 2002.

[14] 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.