情 報 学 基 礎 61−14
自 然 言 語 処 理 142−14
（2001. 3. 5）

チャンキングの段階適用による係り受け解析

工藤 拓　　松本 裕治

奈良先端科学技術大学院大学 情報科学研究科
〒630-0101 奈良県生駒市高山町 8916-5
{taku-ku,matsu}@is.aist-nara.ac.jp

本稿では，チャンキングの段階適用による日本語係り受け解析手法を提案し，その評価を行う．従来法は，任意の二文節間の係りやすさを数値化した行列を作成し，そこから動的計画法を用いて文全体を最適にする係り受け関係を求めるというモデルに基づいていた．しかし，解析時に候補となるすべての係り関係の尤度を計算する必要があるため効率が良いとは言えない．本提案手法は，直後の文節に係るか係らないかという観点のみで決定的に解析を行うため，従来方法に比べ，モデル自身が単純で，実装も容易であり，高効率である．さらに，従来法では，個々の係り関係の独立性を前提としているが，本提案手法はその独立性を一部排除することが可能である．本提案手法を用い，京大コーパスを用いて実験を行った結果，従来法と比較して効率面で大幅に改善されるとともに，従来法以上の高い精度 (89.29%) を示した．

キーワード: 構文解析，係り受け解析，チャンキング，機械学習，Support Vector Machine

Applying Cascaded Chunking to Japanese Dependency Structure Analysis

Taku Kudoh　　Yuji Matsumoto

Graduate School of Infomation Science, Nara Institute Science and Technology
8916-5 Takayama, Ikoma Nara 630-0101 Japan
{taku-ku,matsu}@is.aist-nara.ac.jp

In this paper, we apply cascaded chunking to Japanese dependency structure analysis. A conventional approach consists of two steps: First, dependency matrix is constructed, in which each element represents the probability of a dependency. Second, an optimal combination of dependencies are determined from the matrix. However, this method is not always efficient since we have to calculate all the probabilities of candidates. Our proposed cascaded chunking model is quite simple and efficient, since it estimates whether current segment modifies immediately right-hand side segment to parse a sentence. In addition, proposed model does not assume the independence constraints in dependency relation. Experimental results on Kyoto University corpus show that our system achieves accuracy of 89.29%, higher than that of our previous system, as well as improves the efficiency of parsing.

Keywords : Parsing, Dependency Structure Analysis, Chunking, Machine Learning, Support Vector Machines

1 はじめに

日本語の構文解析において係り受け解析は, 自然言語処理の基本技術の一つとして認識されており, 従来から多くの研究が行われている. 初期の研究では, 二文節間の係りやすさを決定するルールを人手で作成していたが, 網羅性, 一貫性という面で問題が多い. 近年では, 構文情報が付与された大規模コーパスが利用可能になったことで, 機械学習アルゴリズムを用いた統計的な構文解析技術が提案されるようになってきた.

従来の係り受け解析は, 文中の任意の二文節間の係りやすさを数値化した行列を作成し, その中から動的計画法を用いて文全体を最適にする係り受け関係を求めるというモデルに基づくものであった. しかしながら, 解析時や学習時にすべての係り関係の候補を対象としなければならないために, 効率が良いとは言えない. さらに, 各二文節の係り関係は他と独立と仮定しているが, 並列構造のように独立のモデルでは扱いにくい問題もある.

本稿では, チャンキングの段階適用による係り受け解析モデルを提案する. このモデルは極めて単純で実装も容易である. また, 必要最低限の係り受け候補のみを解析, 学習対象としているために従来法に比べ高効率である. さらに個々の係り関係の独立性の仮定を部分的に排除することが可能である.

本稿で提案する係り受け解析も, 実際の係り関係の同定に機械学習アルゴリズムを用いる. 提案手法は, 機械学習アルゴリズムに依存しない手法であるが, 我々は, Support Vector Machine (SVM) [7] を用いる. その理由として, SVM は, 他の学習モデルと比較して極めて汎化能力が高く , 高次元の素性集合を用いても過学習しにくいとされていこと. さらに, Kernel 関数を変更することで, 非線形のモデル空間を仮定したり, 複数の素性の組み合せを考慮した学習が可能な点が挙げられる. 自然言語処理においては, 文書分類や, 係り受け解析, 英語の単名詞句同定などに応用され高い精度を示している [1, 16, 3, 2, 13].

本稿の構成は以下の通りである. 2 章で従来法と提案手法の違いを述べ, 3 章で SVM の説明を行う. さらに 4 章で京大コーパスを用いた評価実験を提示し, 最後に 5 章で本稿をまとめる.

2 係り受け解析モデル

2.1 係り受け解析モデル (従来法)

本節では, これまでに日本語係り受け解析によく用いられているモデルについて説明する. まず, あらかじめ文節にまとめられ属性付けされた文節列 $\{b_1, b_2, \ldots, b_m\}$ を B, 係り受けパターン列 $\{Dep(1), Dep(2), \ldots, Dep(m-1)\}$ を D と定義する. ただし,$Dep(i)$ は, 文節 b_i の係り先文節番号を示す.

これ以降, D は以下の制約を満たすものと仮定する.

1. 文末を除き, 各文節はその文節の後方側に必ず一つの係り先を持つ.
2. 係り受け関係は交差しない.

統計的係り受け解析とは, 上記の二つの制約のもとで, 入力文節列 B に対する条件付き確率 $P(D|B)$ を最大にする係り受けパターン列 D を求めることと定義できる.

$$D_{best} = \underset{D}{\operatorname{argmax}} P(D|B)$$

ここで, それぞれの係り関係は独立である と仮定すると, $P(D|B)$ は,

$$P(D|B) = \prod_{i=1}^{m-1} P(Dep(i)=j \mid \mathbf{f}_{ij})$$
$$\mathbf{f}_{ij} = \{f_1, \ldots, f_n\} \in \mathrm{R}^n$$

のように変形できる. ここで, 確率 $P(Dep(i)=j \mid \mathbf{f}_{ij})$ は文節 b_i と文節 b_j が言語的素性集合 $\mathbf{f}_{ij}$ を持つ時に, 文節 b_i が文節 b_j に係る確率を示す. $\mathbf{f}_{ij}$ は文節 b_i と文節 b_j に関する種々の言語的特徴を表わす n 次元の特徴ベクトルである. 最終的に, これらの確率値をもとに D_{best} を決定する. この時, 確率 $P(Dep(i)=j \mid \mathbf{f}_{ij})$ を格納する行列は係り受け行列と呼ばれている. 従来法は, 与えられた入力文に対し, 係り受け行列を作成する処理と, 係り受け行列から最尤の係り受け候補選択し出力する処理の, 基本となる二つの処理から構成される.

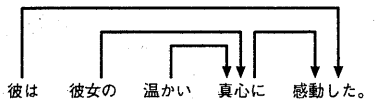
2.2 チャンキングの段階適用による解析

チャンキングの段階適用による構文解析は, 英語の統計的構文解析においては古くから適用されてきた [5, 4]. このアルゴリズムのおおまかな流れは以下のようになっている.

1. 入力として基本句列を与える.
2. 文を先頭から眺めて行き, 任意の基本句の連続を, 新しい非終端ノードをまとめ上げる (チャンキング). (フェーズ 1.)
3. まとめ上げられた句の連続から, 主辞のみを残し, それ以外は削除する. (フェーズ 2.)
4. 非終端ノードが一つになれば終了, それ以外は 2 に戻る

チャンキングはチャンクの状態を示すタグ付与することと解釈できるため, このモデルは基本句に対するタグ付けを段階的に適用した形になっている.

ここで, チャンキングの段階適用手法を日本語係り受けに適用することを考える. 日本語の係り受け構造は, 後方参照であること, 三つ以上の文節で一つの



初期化

Input:	彼は	彼女の	温かい	真心に	感動した。
Tag:	O	O	O	O	O
Input:	彼は	彼女の	温かい	真心に	感動した。
Tag:	O	O	D 削除	D	O
Input:	彼は	彼女の	真心に	感動した。	
Tag:	O	D 削除	D	O	
Input:	彼は	真心に	感動した。		
Tag:	O	D 削除	O		
Input:	彼は	感動した。			
Tag:	D 削除	O			
Input:	感動した。				
Tag:	O	終了			

図 1: 係り受け解析例

非終端ノードを形成しないことを考慮すれば, アルゴリズムのおおまかな流れは以下のようになる.

1. 入力文節すべてに対し, 係り受けが未定という意味の O タグを付与する.
2. 文末の文節を除く O タグが付与された文節に対し, 直後の文節に係るか推定. 係る場合は D タグを付与. 後ろから 2 番目の文節は無条件に D タグを付与 (フェーズ 1.)
3. O タグの直後にあるすべての D タグおよびその文節を削除する. (フェーズ 2.)
4. 残った文節が一つ (文末の文節) の場合は終了, それ以外は 2. に戻る.

このアルゴリズムによる解析例を図 1 に, 具体的な擬似コードを 図 2 に示す.

我々は, 提案するチャンキングの段階適用による係り受け解析は, 従来法と比較して以下のような利点を持つと考える.

[モデルの簡潔性, 効率性]
従来法は, 解析時に候補となるすべての係り関係の尤度を計算しなければならないため効率が良いとは言えない. また, 学習時においてもすべての二つの文節を学習データとして考慮する必要があるため, SVM のように学習時間が事例数に対し非線形に増加する

```
Input := { 文節 [1], .., 文節 [n]} // 入力文節列
Tag := {O, O, .., O} // タグ列
Del := {0, 0, , .., 0} // 削除フラグ
Tag[0] := O // ダミー
Output := φ // Hash, キー:係り元, 値:係り先
function estimate(src, dst)
begin
    feature_set := {src, dst}
    estimate := 任意の学習器 (feature_set)
end
while (length(Input) > 1)
begin
    // フェーズ 1. 直後に係るか決定
    for i := 1 to length(Input) − 1
    begin
        Del[i] := 0 // 削除フラグ
        // 後ろから 2 番目は無条件に係る
        if (i = (length(Input) − 1)) then
            Tag[i] := D
        else if (Tag[i] = O) then
            Tag[i] := estimate(Input[i],
                               Input[i + 1])
        if (Tag[i] = D) then
        begin
            Output[Input[i]] := Input[i + 1]
            if (Tag[i − 1] = O) then
                Del[i] := 1
        end
    end
    // フェーズ 2. O の直後の D を削除
    for i := length(Input) − 1 downto 1
    begin
        if (Del[i] = 1) then
        begin
            delete(Input[i])
            delete(Tag[i])
        end
    end
end
```

図 2: アルゴリズムの擬似コード

```
function estimate(src, dst)
begin
    if (src, dst が学習データで係る場合) then
        estimate := D
    else
        estimate := O
    // 学習データとして保存
    feature_set := {src, dst}
    push(TrainigData, {estimate, feature_set})
end
```

図 3: 学習データ作成時の関数 *estimate*

ような学習モデル[1]では, 扱いが困難となる. 提案手法はモデルのアルゴリズム自身極めて単純であり, 実装も容易である. また, 必要最小限の推定しか行わず, 多くの係り受けが直後の文節に係る事を考えると, 極めて効率がよい.

[非交差条件の自動考慮]
従来法にはモデル自身に非交差条件を考慮する仕組みが備わっていない. 実際の解析時に, 交差する条件を逐次考慮する必要があり, それらの処理は各パーザに依存する. 一方, 提案手法は, それ自身簡潔であるにもかかわらず, モデル自身が非交差条件を考慮する機能を備えている.

[独立性の制約の部分的排除]
従来法は, それぞれの係り関係は独立であると仮定している. しかしながら並列構造のように, 独立なモデルでは解決できない問題がある[2]. 提案手法は, 解析と推定を同時に行い, 着目している 二文節よりスコープの狭い係り受け関係はすべて素性として考慮する事が可能である. このことは「静的素性と動的素性」の節で詳しく述べる.

[文頭からの自然な解析]
従来法では, 実際の解析において, 文末に近い文節の方が係り受けの候補が少ない理由から, 文末から解析するものもあった [12]. しかし, 実際, 我々人間は文頭から解析しており, この点でギャップを感じる. 提案手法は, 自然な形で文頭から解析を行う.

[学習事例抽出と解析の同一視]
学習事例を抽出する作業は, 関数 *esitmate* を図 3 のように置き換えることで実現できる. つまり, 学習事例抽出は, あたかも解析を行いながら, 実際には学習コーパスを参照する形になる. 多くの係り受けが直後の文節に係る事を考えると, 学習事例 (係る事例, 係らない事例のペア) を必要最小限に抑える事ができ, 学習時間が事例数に非線形に増加するような学習モデルでもある程度大量な学習データを扱う事が可能である.

[機械学習アルゴリズムの非依存性]
提案手法は, 直後の文節に係るか係らないかという観点で決定的に解析を行うため, 二値分類が行える学習器であれば, あらゆるものが適用可能である. 係りやすさの尤度や確率値は本提案手法には必ずしも必要ではない.

[係りタイプの考慮]
従来法は, 係り受け解析に特化した手法であり, そのままの形では係り受けのタイプ (通常の係り受け, 並列, 同格 等) を出力しにくい. 本提案手法は, 係りタイプに応じて使用するタグを使い分けるだけでよく, アルゴリズムそのもに手を入れる必要はない.

[1]SVM は $O(n^3)$ のモデルである
[2]並列構造の場合, 前件の係り受け構造, 後件の構造および全体の並列構造は互いに従属関係にある場合が多い.

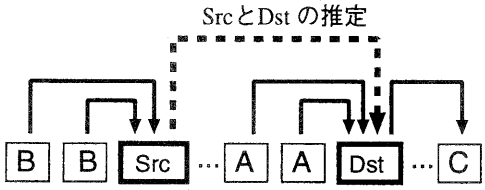


図 4: 三つの動的素性

2.3 静的素性と動的素性

統計的日本語係り受け解析に有効とされてきた素性には, 着目している 2 文節の主辞の語彙や品詞, 語形の活用形, 二文節間の距離, 句読点, 引用符の有無などがある. これらの素性は文節の作成時に決定される素性であり, このような素性集合のことをまとめて**静的素性**と呼ぶこととする. 日本語の係り受け関係は, 語の活用形が大きな制約となり, 静的素性だけで係り先の大部分を限定することができる. しかし, 複数の係り先の候補がある場合や, 並列構造, 複文構造の場合, 個々の係り受け関係の従属性を考慮しないと係り関係を決定しにくいことがある.

我々は以前, 係り関係そのものを素性とする **動的素性** の概念を取りいれ, 静的素性のみの場合より高い精度を示すことに成功した [2]. しかし, [2] では, 動的素性を従来法の係り受けモデルの拡張として取り入れ, 実際の解析には, 関根の提案する文末から解析するアルゴリズム [12] を用いた. そのため, 動的素性として着目している二文節より後方にある文節しか対象にすることができないという問題点があった.

ここで, 我々は, この動的素性の概念をチャンキングの段階適用による係り受け解析に適用することを考える. 提案手法は, 文頭から解析し, 係り関係の推定と解析を同時に行っていくため, 極めて自然な形で動的素性を投入する事ができる. ただし, 基本的にはボトムアップに解析していくために, 動的素性として, 着目している二文節よりスコープの大きい係り関係は投入できない.

具体的に, 我々は動的素性として以下の三つの素性を考慮する (図 4).

1. 着目してる係り先に係る文節 (A)
2. 着目している係り元に係る文節 (B)
3. 着目している係り先が係る文節 (C)

これらの素性は, 解析中, 二つの係り関係が決まった時点で互いに素性情報を提供しあうことで容易に実現することが可能である.

3 Support Vector Machines

正例, 負例 の二つのクラスに属す学習データのベクトル集合を,

$$(\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_l, y_l) \qquad \mathbf{x}_i \in \mathbf{R}^n,\ y_i \in \{+1, -1\}$$

とする. ここで $\mathbf{x}_i$ はデータ i の特徴ベクトルで, 一般的に n 次元の素性ベクトル ($\mathbf{x}_i = (f_1, f_2, \ldots, f_n) \in \mathbf{R}^n$) で表現される. y_i はデータ i が, 正例 (1), 負例 (-1) かを表わすスカラーである. パターン認識とは, この学習データ $\mathbf{x}_i \in \mathbf{R}^n$ から, クラスラベル出力 $y \in \{\pm 1\}$ への識別関数 $y = f(\mathbf{x}, \theta)$ を導出することにある. 一般には, あるコスト関数 $Cost(y, \mathbf{x}, f(\mathbf{x}, \theta))$ を定義し, それを最小化するような関数 $f(\mathbf{x}, \theta)$ を求める問題に置き代わる.

従来からあるモデルの多くは, 学習データとモデルとの誤差をコスト関数として選択していた[3]. これらは**経験的リスク最小化原理**に基づく学習モデルと呼ばれている. しかし, 経験的リスク最小化原理に基づくモデル推定は, 例外的な事例 (ill-posed data) に対し過学習してしまい頑健な推定ができないことが指摘されている [7].

一方, Support Vector Machine (SVM) [7] では, 関数クラスとして $f(\mathbf{x}, \mathbf{w}, b) = sgn(\mathbf{wx} + b)$ といった線形分類器を仮定し, 以下のような**構造化リスク最小化原理**に基づくコスト関数を選択する.

$$Cost(y, \mathbf{x}, f(\mathbf{x}, \mathbf{w}, b)) = C \sum_{i=1}^{l} \max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i + b)) + \frac{1}{2}||\mathbf{w}||^2 \quad (1)$$

コスト関数の第一項は, 学習データとの誤差を示す経験的リスク最小化原理に基づくコストである. 第二項に モデル自身の複雑さを考慮するコストが付与されている. C はこれらのバランスを決めるパラメータである. 第二項は分類関数が持つ VC 次元と密接に関連しており, VC 次元が小さいモデルほど真のテストデータに対する精度を向上させる. 分類関数クラスが線形関数の場合 VC 次元の上限値 は $||\mathbf{w}||^2$ と比例関係にある事から, SVM のコスト関数が導出される.

この最適化問題は, 一般的な 凸二次計画問題に帰着され, すべての局所解が大域解になる事が保証されている. さらに, 事例間の内積を一般化した Kernel 関数を用い, 低次元空間での非線形分類器を高次元空間上での線形分類器とみなすことで, 上記の原理や計算量を変更することなく非線形の分類が可能となっている.

[3]例えば 最小二乗誤差や対数尤度など

4 実験および考察

4.1 実験環境, 設定

実験に用いたコーパスは京大コーパス (Version 2.0) [14] の一部で, 学習には 1 月 1 日から 8 日分 (7958 文), テストには 1 月 9 日分の (1246 文) を用いた. 実験には自作の SVM 学習ツール TinySVM を用いた[4]. また, 式 (1) におけるパラメータ C は, すべての実験を通して 1 に固定した. Kernel 関数には多項式 Kernel を用い, 次元数は [2] と共通にするため, 3 に固定した.

次に, 学習に用いた静的素性を, 表 1 に示す. これらは 若干な差異はあるものの [2, 11, 10, 9] 等で用いられた素性であり, 日本語係り受け解析に用いられる素性として一般的なものである. ただし, **主辞**とは文節内で品詞が特殊, 助詞, 接尾辞となるものを除き, 文末に一番近い形態素, **語形**とは文節内で品詞が特殊となるものを除き, 文末に一番近い形態素のことを指す. また, 見出し語の選択に関しては適当な頻度による閾値を設けず, 学習データ中のすべての語彙を用いた.

次に, 動的素性の取り扱いについて説明する, 一般にタイプ A, B の動的素性の候補は複数あり, それぞれに対し, 主辞, 語形の品詞や活用形等をばらばらに登録すると, どの活用形がどの文節のものか区別できなくなる. そこで文節内の機能語の部分を一つの表現に縮約する必要がある. この縮約表現を便宜的に機能語と定義した. 具体的には, 助詞, 副詞, 連体詞, 接続詞 については見出し語そのものを, 活用形のあるものはその活用形を, その他の品詞については, 品詞と品詞細分類を与えた. また, タイプ C の動的素性には, 主辞の品詞と品詞細分類を与えた.

4.2 実験結果

我々の提案手法と, 従来方法 [2] の結果を表 2 にまとめた. ただし, 係り受け正解率とは, 文末の一文節を除くすべての文節に対して, 正しく係り先が同定できたものの割合, 文正解率とは, 文全体の解析が正しいものの割合を示す.

結果, 係り受け正解率, 文正解率ともに, 従来方法に比べ精度が向上している. 効率面でも, 学習に要する時間が, 2 週間程度という非現実的な時間から, 10 時間程度と実用に耐えうる時間に短縮され, 一文あたりの平均解析時間は, 2.1 秒程度から, 0.5 秒程度に改善された[5] [6]. 実際の解析精度の低下がないことから, 提案手法は, 従来法に比べ高効率であるといえる.

[4]http://cl.aist-nara.ac.jp/~taku-ku/ より入手可能.

[5]学習は AlphaServer 8400 上で, 解析は PentiumIII(1GHz) の Linux 上で行った.

[6]我々は, 素性空間がバイナリである特徴をうまく利用し, SVM の分類器を高速化している. 定義どおりの単純な分類器の場合, 解析時間は, この 4-6 倍程度の時間になるだろう.

4.3 動的素性の効果

表3に, 動的素性のうちそのいくつかを削除した場合の精度の増減を示した. 結果から, どのタイプの動的素性も係り受け解析にも有効であることが分かる.

さらに, 図4に, 動的素性を用いた場合と用いなかった場合の精度と学習データのサイズの関係を示した. 学習データが少量の時は, 動的素性は有効に機能していない. これは, 学習データが少量の時は, 動的素性が過学習の要因として働き, 十分に汎化が行われていないことを意味している. しかし, 学習データを増加させるとともに, 動的素性の効果は次第と大きくなっている. 学習曲線から考察するに, 学習データをさらに増やすと, 精度はさらに向上し, 動的素性の有無による精度差がさらに大きくなると予想される.

4.4 従来法, 関連研究との比較

[解析モデルの観点から]

我々は, 従来法に基づき, 機械学習アルゴリズムとして SVM を用いた係り受け解析を以前提案した [2]. 従来法では, 係り受けの尤度 (確率値) を要求するため, 我々は, 学習データ中から実際に係った関係を正例, 係らなかった事例を負例として学習を行い, 分離平面からの距離を尤度とすることで, 従来モデルの枠組で解析を行った. 直感的には, すべての係り受け候補を学習データとし, 学習データの量が多いぶん従来法のほうが高い精度が得られると期待できる. しかしながら, 実際には提案手法の方が高い精度を示している. この要因はいったい何なのであろうか.

従来法ではすべての係り受け候補を学習データとするため, 係った文節と似た語形情報を持つ後方の文節は極めて例外的な事例となってしまう. さらに, 非交差条件により手前に係らなかった事例で, 同じように係った文節と似た語形情報を持つものも例外的な事例となる. 多くの文節は直後の文節に係る事を考えると, 上記のような事例は無視できない量となり, 結果として不必要に多くの例外的事例までも学習してしまう. つまり, 実際の学習データにおいて, 同じ係らない現象でも, 越えて係ったのか手前に係ったのか, さらに非交差条件から係らなかったのか, これら三つを区別する必要があると考える. 特に, 「文節はできるだけ近い文節に係る」というヒューリスティックスを考えると, 実際に係った事例と越えて係った事例の事例差を優先して学習することが重要であろう.

我々の提案手法は, 上記のヒューリスティックスに沿いながら, スコープの小さい係り関係から優先的に解析していく. さらに, 学習データの抽出の際も, あたかも解析を行いながら抽出するため, 学習データ中で係ったものと越えて係ったもののみが学習対象となり, 残りの候補は考慮されない. これらの抽出された事例は, 絶対数は少ないものの, 係り受け解析に有

前/後文節	主辞見出し, 主辞品詞, 主辞品詞細分類, 主辞活用, 主辞活用形, 語形見出し, 語形品詞, 語形品詞細分類, 語形活用, 語形活用形, 括弧の有無, 句読点の有無, 文節の位置 (文頭, 文末)
文節間	距離 (1,2-5,6 以上), すべての助詞 (は, が, を, に ...), 括弧, 句読点の有無

表 1: 使用した静的素性

提案法	係り受け正解率	89.29% (10057/11263)
	文正解率	47.53% (589/1239)
	学習時間	約 10 時間
	解析時間	0.5 秒/文
従来法	係り受け正解率	89.09% (10034/11263)
	文正解率	46.17% (572/1239)
	学習時間	約 2 週間
	解析時間	2.1 秒/文 (beam 幅 1)

表 2: 実験結果

削除した動的素性	精度の増減	
	係り受け正解率	文正解率
A	-0.28%	-0.89%
B	-0.10%	-0.89%
C	-0.28%	-0.56%
AB	-0.33%	-1.21%
AC	-0.55%	-0.97%
BC	-0.54%	-1.61%
ABC	-0.58%	-2.34%

表 3: 動的素性の効果

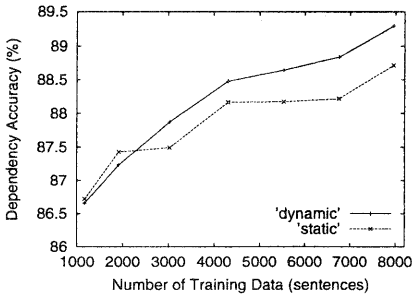


表 4: 学習データと解析精度

効な事例集合となり, 上記の理由からくる例外的な事例の混入は少ない.

宇津呂らは, 係り関係を,「係る」「越える」のみに限定したモデルを, 内元らは, 係り関係を,「係る」「越える」「手前」という三つに分けるモデルを提案し, 従来の「係る」「係らない」のみを考慮するモデルよりも精度が向上したと報告している [17, 10]. これらの手法は, 上記の考察による例外的事例を切り分けて考えることができるため, 有効な戦略だと言える. しかしながら, ある任意の係り関係の確率値一つを計算するのに, 残りのすべての係り受け候補について, それぞれのクラスに属する確率値を計算する必要があり効率が悪い. また, SVM のように真の意味での尤度や確率値を出力しないアルゴリズム[7]の適用は難しく, 学習アルゴリズムに依存した手法と言える.

また, 我々や内元が実際に採用した文末から解析するアルゴリズム [12] は, 並列構造に有効な タイプBの動的素性を考慮できない. この点から見ても提案手法は従来法に比べ優位であると考える.

[学習の観点から]

係り受け解析は, 係り元と係り先の素性の組を学習する必要があるため, 学習器は素性の非線形性を考慮できなければならない. 内元 [11, 10], 金山 [15] らは学習アルゴリズムに, Maximum Entropy (ME) を選択しているが, ME は素性の独立性を前提としているため, 係り元と係り先の素性の組を新たな素性として追加しなければならず, 係り受け解析の学習には不向きであると考える. 実際に, 同じ学習, テストコーパスを用いて実験を行っている内元らの手法は, 我々の静的素性のみの手法と比較しても 1% 程度劣っている (87.93%). 使用した素性がほとんど変わらないことを考えると, 上記の ME の持つ弱点が性能差の要因の一つとなっていると考える.

決定木 (決定リスト) は, 貪欲的に非線形性を考慮することが可能であるが, それ自身では過学習に陥いりがちなので, Boosting の弱学習器として用いるのが一般的である. Boosting と SVM は, 共に**マージン最大化** という戦略に基づくアルゴリズムであり, 過学習を起こしにくいとされている[8]. しかし, 弱学習器としての決定木は, 語彙を投入しても逆に精度が下がるという報告 [9] がある事から, 慎重な素性選択 (どの語彙を用いるか等) を要求する点で劣る.

一方 SVM は, Kernel 関数により計算量を変える事なく素性の非線形性を考慮できる学習器であり, 係り受け解析に適している. また, SVM は与えられた素性数に依存しない高い汎化能力を持つため, 他の手法に比べ, 慎重な素性選択を要求しない. 実際に, 文書分類の応用例において, 素性となる単語を増やした場合, 決定木の場合は精度が低下したが, SVM は過学習することなく精度が向上したという報告 [16] があることが, その事実を裏付けている.

[7]分離平面からの距離は真の意味での尤度とは言い難い.

[8]Boosting と SVM はマージンの捉え方, 計測方法が異なる. 詳細は, 文献 [6] を参照されたい.

	KNP	提案手法
係り受け正解率	89.68%	89.41%
並列構造正解率	76.32%	65.87%

表 5: KNP との比較

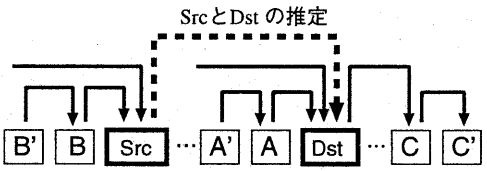


表 6: 再帰的な動的素性

4.5 今後の課題

[並列構造解析]

並列句の類似性に着目して並列構造を精度良く検出するルールベースの日本語係り受け解析器 KNP [8] がある. 提案手法と KNP を以下の二つの観点から比較した.

1. 係り受け解析の精度
2. 並列構造を含めた精度

後者は, 京大コーパスに付与されている係りタイプのうち, 通常の係りタイプと, 並列のタイプを区別し, 別のタグを付与することで実現できる. このためには, SVM を多値分類器に拡張する必要があるが, 二値分類器を多値分類器に拡張する手法の一つである pairwise 法を用いた. 実験には, テストデータの中から, KNP の文節区切り認定と一致する文のみを選び比較を行った. その結果を表 5 に示す. ただし, 並列構造正解率とは, 並列タグが付与されたすべての文節のうち, 係り先とタグを正しく同定できた割合を示す.

係り受け自身の解析精度は, KNP に比べ若干劣ってはいるものの, ほぼ同程度の結果となった. KNP が京大コーパスを基に人手でチューンしたルールから構成されるており, この精度が, いわゆるクローズドデータに対するものであることを考えると, 我々の提案手法は十分価値のある精度であると考える.

一方, 並列構造の精度は, KNP に比べ 10% 以上劣っている. これは, 本提案手法が, 局所的な関係のみを考慮しながら解析するモデルであることに起因する. KNP は, 並列構造の前半部分, 後半部分の類似性を動的計画法を用い探索し, 係り受けの前処理として並列構造部分を検出している [8]. しかしながら, 類似度の尺度は少量のデータを参考にしながら, 発見的に導出されており, 網羅性, 一貫性という意味で疑

問が残る．我々は，コーパスから統計的に学習する枠組は変更せず，一文全体に渡る広範囲の情報を利用をしながら並列構造を同定し，その情報を本手法に取り入れることで，さらなる精度向上を計っていきたいと考えている．

[動的素性の詳細な調査]
本稿では，基本となる三種の動的素性 (A,B,C) を提案し，それらが係り受け解析に有効である事を示した．一方で我々の提案手法は，注目している二文節よりスコープの狭いあらゆる係り関係が動的素性として考慮可能であるため，図 6 のように，これらの基本三素性を再帰的に繋げた素性 (A',B',C') も動的素性の候補となりうる．しかし，不必要に素性を増やしていくと，過学習に陥いったり，効率性の面で問題がある．また，実際問題として，これらの再帰的な動的素性をどのように表現し個々の学習器に与えるのかも問題となる．今後，再帰的な動的素性を含め，動的素性に関し，さらなる調査を行いたいと考えている．

[後方文脈の考慮]
係り関係の同定に，係り先よりも後方にある文節列 (後方文脈) の有効性が明らかになってきている．内元らは，係り関係を「係る」「越える」「手前」と 3 つに分けることで後方文脈を考慮している [10]．しかし，それぞれの関係の独立性を前提としなければならない．また，全ての後方文脈を考慮するため，効率が悪くなり，ノイズとなる不必要な関係までも学習してしまう可能性がある．金山らは，文法を用いて係り先の候補を絞り，それらの候補を確率の条件部に入れることで，後方文脈を考慮している [15]．この手法は，独立性を仮定する必要がなく，解析に有効な限られた後方文脈のみを用いるのでノイズの混入も少ない．我々は，金山らの手法に基づき，後方文脈を考慮したいと考える．その際，金山らは既存の文法を用いているが，既存の文法が手元に無い状態でも解析が行なえる手法の提案を試みたい．

5 まとめ

本稿では，チャンキングの段階適用による新しい係り受け解析モデルを提案した．このモデルは，従来法と比べ解析アルゴリズム自身が単純で高効率であるにもかかわらず，従来法と同等かそれ以上の高い精度 (89.29%) を示すことが分かった．さらに，着目している二文節より狭いスコープの係り関係はすべて素性として使用可能であり，それらが係り受け解析に極めて有効であることを示した．

参考文献

[1] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *European Conference on Machine Learning (ECML)*, 1998.

[2] Taku Kudoh and Yuji Matsumoto. Japanese Dependency Structure Analysis Based on Support Vector Machines. In *Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 18–25, 2000.

[3] Taku Kudoh and Yuji Matsumoto. Use of Support Vector Learning for Chunk Identification. In *Proceedings of the 4th Conference on CoNLL-2000 and LLL-2000*, pp. 142–144, 2000.

[4] Adwait Ratnaparkhi. A Liner Observed Time Statistical Parser Based on Maximum Entropy Models. In *Proceedings of EMNLP '97*, 1997.

[5] Abney S. Parsing By Chunking. In *Principle-Based Parsing*. Kluwer Academic Publishers, 1991.

[6] Alexander J. Smola, Peter L. Bartlet, Bernhard Schölkopf, and Dale Schuurmans. Introduction to Large Margin Classifiers. In *Advances in Large Margin Classifiers*. MIT Press, 1999.

[7] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[8] 黒橋禎夫, 長尾眞. 並列構造の検出に基づく長い日本語文の構文解析. 自然言語処理, Vol. 1, No. 1, pp. 35–57, 1994.

[9] 春野雅彦, 白井諭, 大山芳史. 決定木を用いた日本語係り受け解析. 情報処理学会論文誌, Vol. 39, No. 12, p. 3117, 1998.

[10] 内元清貴, 村田真樹, 関根聡, 井佐原均. 後方文脈を考慮した係り受けモデル. 自然言語処理, Vol. 7, No. 5, pp. 3–17, 2000.

[11] 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.

[12] 関根聡, 内元清貴, 井佐原均. 文末から解析する統計的係り受け解析アルゴリズム. 自然言語処理, Vol. 6, No. 3, pp. 59–73, 1999.

[13] 工藤拓, 松本裕治. Support Vector Machine を用いた Chunk 同定. 情報処理学会 自然言語処理研究会 NL140, pp. 9–16, 2000.

[14] 黒橋禎夫, 長尾眞. 京都大学テキストコーパス・プロジェクト. 言語処理学会 第 3 回年次大会, pp. 115–118, 1997.

[15] 金山博, 鳥澤健太郎, 光石豊, 辻井潤一. 3 つ以上の候補から係り先を選択する係り受けモデル. 自然言語処理, Vol. 7, No. 5, pp. 71–91, 2000.

[16] 平博順, 春野雅彦. Support Vector Machine によるテキスト分類における属性選択. 情報処理学会論文誌, Vol. 41, No. 4, p. 1113, 2000.

[17] 宇津呂武仁, 西岡山滋之, 藤尾正和, 松本裕治. コーパスからの日本語従属節係り受け選好情報の抽出およびその評価. 自然言語処理, Vol. 6, No. 7, pp. 29–60, 1999.