情報処理学会論文誌 Vol. 52 No. 7 2277–2290 (July 2011)

決定的な解析と相対的な比較による解析の2側面を持つ日本語係り受け解析

山本悠二[†1] 増山繁[†1]

日本語係り受け解析の手法は大きく分けて，1. 決定的な解析方法と，2. 係り先候補の確信度に基づく解析方法がある．前者は係り先候補間の比較が行えないことから，特に長距離依存の係り先を同定するときに誤りを生じやすいという傾向がある．また，後者は係り先候補集合のすべての要素を探索するため，計算時間の点で問題がある．提案手法では，係り先候補の確信度に基づく解析方法での解析時間を減らすために，決定的な解析が容易な文節について先に係り先を定めた後に，相対的な比較による係り先を同定する方法を示す．京都テキストコーパス 4.0 を用いて提案手法を評価したところ，係り先候補の確信度に基づく解析方法の 1 つである相対モデルと比較してほぼ同等の解析性能を持ち，かつ，実行時間が 2.4 倍程度高速であることが確認された．

Bilateral Japanese Dependency Parsing – Deterministic and Comparative by Relative Preferences of Dependency

Yuji Yamamoto[†1] and Shigeru Masuyama[†1]

Japanese dependency parsers fall into two main methods, 1) deterministic parsing and 2) parsing based on dependency certainties among modifee candidates. The former methods tend to make errors especially for identifying long-distance dependencies because these methods do not opt the candidate by comparing candidates. On the other hand, the latter methods have difficulty with their parsing speed due to searching the most preferable candidate from all modifee candidates. The proposed method identifies easily-analyzable dependencies by deterministic parsing and identifies the rest dependencies by parsing based on dependency certainties among modifee candidates later. Experiments using the Kyoto Text Corpus 4.0 show that the proposed method runs 2.4 times faster than the relative-model parser that belongs to the latter methods while the dependency accuracy of the proposed method is nearly comparable with the relative-model's.

1. はじめに

係り受け解析は自然言語処理における基本技術として認識されている．特に，その精度向上は，解析後の応用タスクにおける結果に直接影響することが多いため，研究課題として重要な位置を占めている．また，計算効率についての研究も実用的な自然言語処理のアプリケーションにとって重要である．

従来提案されてきた統計的日本語係り受け解析は大きく分けて，i) 決定的な解析方法[1),2)]と，ii) 係り先候補の確信度に基づく解析方法[3),4)]がある．ここで，前者の方法は，Shift-Reduce 法による決定的な解析[⋆1]を指す．つまり，プッシュダウンオートマトン上で Shift，Reduce と呼ばれる操作を組み合わせることによってボトムアップに文を決定的に解析するものである．この解析方法は，広義の決定的な解析である「スタック内の任意の要素を取り出して解析するもの」とは異なることを強調する[⋆2]．Shift-Reduce 法による決定的な解析では，ある係り元について，その係り先を決定する過程を観察すると，多くの場合，係り先候補集合のいくつかの候補を取り出して係り先となりうるか否かを調べるだけで係り先を決定できていることが確認できる．そのため，後者の係り先候補集合のすべての要素を取り出して係りやすさの確信度を求める方法に比べて高速に係り受け解析を行うことができる．特に論文 1) では，文節数に対して線形時間で解析が行えるアルゴリズムが示されている．

しかしながら，決定的な解析では，特に長距離依存の係り先を同定するときに誤りを生じやすいという傾向がある．これは，係り先候補 A も係り先候補 B も係りやすいものであったとしても，後方の文脈をいっさい考慮せずに，近い方の候補から Reduce 操作を行うか否かを選択する必要があるためである．特に精度良く長距離依存を解析することは，文短縮要約[5)]や，節単位を基本とした表現の獲得[6)]といった文全体の構造を把握する必要があるタスクにおいて重要である．このような場合，統計的日本語係り受け解析のもう 1 つの方法である，係り先候補の確信度に基づく解析方法を用いることが好ましい．この解析方法では，ある係り元の係り先を求める場合，係り元とその候補の文節対の係りやすさの確信度をすべての候補について求める．そして，確信度が最も高い候補を係り先として選択する．係り先

†1 豊橋技術科学大学
Toyohashi University of Technology

⋆1 決定的な解析とは，つねに唯一の文法規則とその適用箇所を選択する構文解析をいう[8)]．

⋆2 論文 22)，28)，および，係り先候補の確信度に基づく解析をビーム幅 1 で行うものは，広義の決定的な解析にあたる．

候補の確信度に基づく解析方法は，係り先候補集合のすべての要素[⋆1]について確信度を求める必要があるため，決定的な解析方法に比べて解析時間がかかる．たとえば，論文 4) による解析アルゴリズムを使用すると，解析時間の上限は文節数の二乗に比例する．

そこで本稿では，係り先候補の確信度に基づく解析方法での解析時間を減らすために，決定的な解析が容易な文節について先に係り先を定めた後に，残りの文節について係り先の相対的な比較による係り先の同定を行う手法を提案する．ここで，基本的な考え方について述べる．先に述べたように，決定的な解析では係り先候補のいくつかが係りやすい候補であるとき，後方の文脈をいっさい考慮せずに，近い方の候補から Reduce 操作を行うか否かを選択する必要があるため誤りが生じやすい．一方，文献 8) で示されているように，このような曖昧性がある係り受けは，係り元，係り先候補に特徴がある．提案手法は，このような係り受けに関して，先に行う決定的な解析では係り先の同定を保留しておき，後の係り先の相対的な比較によって定める．

本稿の構成は以下のとおりである．まず，2 章で統計的日本語係り受けについての記号の定義を行う．3 章で統計的係り受け解析手法である決定的な解析と相対的な比較による解析について概説する．4 章で提案手法について述べる．5 章で実験とその結果，6 章で関連研究について述べ，7 章でまとめを行う．

2. 統計的日本語係り受け解析

依存文法に基づく係り受け解析でよく用いられているモデルについて説明する．前提として日本語文における係り受けは以下の制約を満たすものとする．

(1)　係り受けは前方から後方に向いている（後方修飾）．
(2)　係り受けは交差しない（非交差条件）．
(3)　係り受けは係り先を 1 つだけ持つ．
(4)　文末は係り先を持たない．

以下では記号の定義を行う（**図 1** に例を示す）．まず，N 個の文節列で構成される日本語文について，文節列を保持する配列を `B[]` とする．以降，配列は 0 から始まるものとする．つまり，`B[]` は `B[0]` から `B[N-1]` までアクセスすることができる．また，係り受け解析後の，推定された係り先文節の添字番号が格納されている配列を `estlink[]` とする．また，訓練データの文節列については，正しい係り先文節の添字番号が格納されている配列

`B[]`	私は	興味本位で	この	本を	読んだ。
添字番号	0	1	2	3	4
`anslink[]`	4	4	3	4	−1

図 1　例文における B[]，anslink[]
Fig. 1　B[] and anslink[] on an example sentence.

`anslink[]` が与えられる．

統計的係り受け解析では，初めに訓練データとして与えられた複数個の文節列を用いて，係り先を求めるための識別モデルを作る．ここで，`B[]` の i 番目と j 番目の文節対を表す素性ベクトルを $F(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})$ と表記する．また，このとき使用する機械学習に非線形カーネルを導入することが多い[⋆2]ため，特徴空間への非線形写像についてのいくつかの定義を導入する．まず，入力として与えられた素性ベクトルを特徴空間へ非線形写像する関数を $\phi(\cdot)$ とおく．つまり，先に示した文節対の素性ベクトルを非線形写像したものは $\phi(F(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B}))$ である．以降，簡略化のため，$\phi(F(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B}))$ を $\psi(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})$ と表記する．また，4 章は $\psi(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})$ を単位ベクトルに正規化したものを使用する．これも簡略化のために $\omega(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B}) = \psi(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})/||\psi(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})||$ と表記する．なお，非線形カーネルは特徴空間内での内積を，入力された素性ベクトル上の空間（入力空間）内での内積を非線形写像したものとして計算することができる（詳細はたとえば文献 9) を参照）．たとえば，統計的日本語係り受け解析でよく用いられる多項式カーネル（次元数を d とする）の場合，

$$\psi(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})\cdot\psi(\langle \mathtt{k},\mathtt{l}\rangle,\mathtt{B'}) = \{1 + F(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})\cdot F(\langle \mathtt{k},\mathtt{l}\rangle,\mathtt{B'})\}^d \quad (1)$$

で計算することができる．

3. 決定的な解析，相対的な比較による解析

3.1 決定的な解析

Shift-Reduce 法による決定的な解析とは，スタックを利用し，Shift と Reduce という 2 つの操作を組み合わせることによってボトムアップに文を解析するものである．なお，Shift 操作とは処理する語を入力文から取り出しスタックにプッシュすること，Reduce 操作とは生成規則を適用することをそれぞれ表す．ここでは，文節数に対して線形時間で日本語係り受け解析が行える，颯々野の係り受け解析[1] について説明する．擬似コードを**図 2** に示

⋆1 正確には非交差条件を満たす係り先候補に限定している．

⋆2 特に素性集合の要素の組合せを考慮するために多項式カーネルが用いられる．

```
1: // 出力: estlink[]: 推定された係り先文節の添字番号が格納された配列
2: function sr_parsing( w, B[], N, estlink[] )
3: estlink = [-1] * N;  // −1 が N 個入った配列
4: push(stack, -1);  // −1 は番兵
5: push(stack,  0);
6: for (j = 1; j < N; j++) {
7:   i = pop(stack);
8:   while ( (i != -1)
9:           && ( (j == N - 1) || ( w·ψ(⟨i,j⟩,B) > 0 ) ) ) {
10:    // Reduce 操作
11:    estlink[i] = j;  // 推定された係り先文節の添字番号を代入
12:    i = pop(stack);
13:  }
14:  // Shift 操作
15:  push(stack, i);
16:  push(stack, j);
17: }
```

図 2　擬似コード—颯々野の係り受け解析アルゴリズム
Fig. 2　Pseudo code – Sassano's dependency parsing algorithm.

す[⋆1]．なお，擬似コード中の関数 pop，push はそれぞれ以下の機能を持つ．

- 関数 pop：第 1 引数のスタックから，先頭の要素を取り除き，その要素を返す．
- 関数 push：第 1 引数のスタックの先頭に第 2 引数の値を追加する．

ここで，識別モデルの重みベクトルを $\mathbf{w}$ と定義する．このとき，文節列 B[] における i 番目と j 番目の文節対を識別モデルで判定したときの値は $\mathbf{w}\cdot\psi(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})$ となる．この識別モデルは，文節対を判定したときに，値が正の値をとれば「文節対は係る」と対応付けられ，負の値をとれば「文節対は係らない」と対応付けられるように学習されているものとする．図 2 に示したアルゴリズムにおいて，スタックには係り元文節の添字番号が格納されている．また，Shift 操作は「係り元文節になりうる文節の添字番号をスタックにプッシュすること」，Reduce 操作は「生成した文節対が係ると定めること」をそれぞれ表す．したがって，このアルゴリズムは，係り先文節を左から右へ走査するときに，スタックから取り出した係り元と着目している係り先で文節対を作り，識別モデルを用いて Reduce 操作を行うか否かを決める．ただし，係り先が末尾に到達した場合はスタックにある係り元文節は必ず文末の文節を係り先として定める．

3.2　相対的な比較による解析

先に示した決定的な解析では，識別モデルを用いて，ある係り元について最短の係り先となりうる候補を求めていた．しかし，係り受け解析においては，依存関係の曖昧性から，複数の候補のうちで，より係り先になりやすい候補を選択する必要性が生じる．相対的な比較による解析では，ある係り元の係り先を求める場合，識別モデルを用いて，係り元とその候補の文節対の係りやすさの確信度をすべての候補について求める．そして，確信度の最も高い候補を係り先として選択する．ここで，文節対の係りやすさの確信度を求める識別モデルの重みベクトルを $\mathbf{w}$ とする．このとき，文節列 B[] における i 番目と j 番目の文節対の係りやすさの確信度は $\mathbf{w}\cdot\psi(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})$ となる．この識別モデルは，ある文節対が他の文節対と比較したときに，より係りやすいものであれば確信度の値もより大きくなるように学習されているものとする．なお，このような学習方法を優先度学習[7] と呼ぶ．解析アルゴリズムとしては，たとえば，文末から解析するもの[4] がある．

4.　決定的な解析と相対的な比較による解析の 2 側面を持つ日本語係り受け解析

4.1　基本的な考え

提案手法では，係り先候補の確信度に基づく解析方法での解析時間を減らすために，決定的な解析が容易な文節について先に係り先を定めた後に，相対的な比較により係り先を同定する方法を示す．先に述べたように，Shift-Reduce 法による決定的な解析では長距離依存の係り先を同定するときに誤りが生じやすいという傾向がある．これは，係り先候補のいくつかが係りやすい候補であったとしても，後方の文脈をいっさい考慮せずに，近い方の候補から Reduce 操作を行うか否かを選択する必要があるためである．

一方，文献 8) で示されているように，曖昧性が生じる係り受けは，たとえば以下のように，ある程度は類型化が可能である（A，B，C などは名詞，V_1，V_2 などは動詞を表す．また，下線部は曖昧性がある係り元，矩形で囲まれているものは係り先となりうるものを表す）．

⋆1 論文 1) の擬似コードでは，i は係り先文節番号の添字，j は係り元文節番号の添字を表しているが，本稿では i は係り元文節番号の添字，j は係り先文節の添字の意味で用いているため注意．

(1)「A の [B の] [C]」のような連体修飾語の係り先の曖昧性

(2)「A が [V_1 した] B を [V_2 した]」のような格要素の係り先の曖昧性

(3)「…V_1 したが … [V_2 したので] … [V_3 した]」のような従属節の係り先の曖昧性

(4)「A を V_1 したので B が [V_2 し] C を [V_3 したが] D に [V_4 された]」のような並列構造の範囲の曖昧性[⋆1]

このような類型を用いれば，決定的な解析において，上記のパターンにあてはまるものを Reduce 操作しないことで係り先の同定を保留し，後の相対的な比較による解析で係り先候補を精査することができる．しかしながら，曖昧性が生じる係り受けについての類型化を人手で行うことは網羅性の点で問題があることから，機械学習の範疇で Reduce 操作を保留する機構を組み入れることが望ましい．つまり，決定的な解析において，先のようなパターンに近い文節対が出現したときに，識別モデルが負の値を返すように学習を行えるようにする．ただし，実際のタグ付けコーパスに付与されているタグは，係り先に曖昧性があるか否かという情報はないため，単純な分類問題に帰着することはできない．そこで，1 つの識別モデルを二値分類学習と優先度学習[7] を組み合わせてモデルを作ることで，先の性質を持つ識別モデルを作る．

4.2 提案手法

以下では，決定的な解析と相対的な比較による解析の 2 側面を持つ日本語係り受け解析 “bilateral parsing” を提案する．

4.2.1 学習アルゴリズム

提案手法の学習アルゴリズムについて示す．重みベクトル $\mathbf{w}$ を用い，正解の係り先情報が分かっている文節列に対して，決定的な解析，および，相対的な比較による解析を行う．もし，解析途中で誤った出力になる場合や，正しい出力であったとしても十分なマージンがとれていない場合は重みベクトルを更新することで正解の係り先情報がない場合でも正しく解析できるように補正する．重みベクトルを更新するための学習アルゴリズムについては，Online Passive-Aggressive Algorithm [10] の PA-I（以下 OPA と略記）を用いた．なお，本項の最後に詳細を示すが，提案手法は，いくつかの条件を満たせば，他の重みベクトルを更新する学習アルゴリズムを使用することが可能である．

擬似コードを示す前に，入力として与える引数について説明する．まず，T は，訓練データの文集合である．T の要素は，「訓練データの文節列，文節数，正しい係り先文節の添字番号の配列」の三つ組で構成される．C は，OPA で使用する引数で，マージン違反を起こす事例に対してどれだけ積極的に重みベクトルを更新するかについて決めるパラメータである．C が大きければ，与えられた事例について，OPA の定式化によって指定されたマージンを忠実に確保するようになる．I は，訓練データセット単位で何回学習を繰り返すかについて指定するパラメータである．

```
1: // 出力: 更新された重みベクトル w
2: funciton train(T, C, I)
3: w ← 0;
4: for (iter = 1; iter <= I; iter++) {
5:   foreach ( ⟨B, N, anslink⟩ ∈ T ) {
6:     estlink = [-1] * N;  // 文節数分 −1 が入った配列
7:     ⟨w, estlink⟩ = bilateral_sr_train(w, C, B, N, anslink, estlink);
8:     ⟨w, estlink⟩ = bilateral_comp_learn(w, C, B, N, anslink, estlink);
9:   }
10: }
```

図 3　擬似コード—Bilateral Parsing の学習アルゴリズム
Fig. 3　Pseudo code for training the bilateral parser.

擬似コードを**図 3** に示す．このコードでは，訓練データから文を取り出し，決定的な係り受け解析で部分的な係り受けができるように学習（関数は `bilateral_sr_train`）し，その後で残りの文節において，係り先の相対的な比較による係り受けができるように学習（関数は `bilateral_comp_learn`）する．なお，5 章での実験で用いた学習アルゴリズムは，論文 11) に掲載されている重みベクトルの平均化を行った．

擬似コードで示した決定的な係り受け解析での学習アルゴリズム（関数 `bilateral_sr_train`）を**図 4** に示す．このアルゴリズムは，基にした解析アルゴリズムである颯々野の係り受け解析[1] で用いる識別モデルをオンライン学習を使用して学習するのとは以下の 2 点で異なる．

- 提案手法のアルゴリズムは，末尾まで探索しても係り先が見つからなかった場合は，係り先がない（つまり，`estlink[]` の要素が -1 のまま変わらない）とする．これらの文節は，後の相対的な比較による解析で係り先を定める．
- 文節対 $\langle \mathtt{i}, \mathtt{j} \rangle$ が正しい係り受けであるとする．そして，$\langle \mathtt{i}, \mathtt{j} \rangle$ を読み込んだ時点での重

⋆1 各下線部の文節は，その後方の矩形で囲まれた文節が係り先になりうるものであることを表している．

```
1: // 出力: w: 更新された重みベクトル
2: //       estlink[]: 推定された係り先文節の添字番号が格納された配列
3: function bilateral_sr_train(w, C, B[], N, anslink[], estlink[])
4: push(stack, -1);  // −1 は番兵
5: push(stack,  0);
6: for (j = 1; j < N; j++) {
7:   i = pop(stack);
8:   while (i != -1) {
9:     if ( (i == N - 2) && (j == (N - 1) ) )
10:      estlink[i] = j;
11:    else {
12:      scr = w · ω(⟨i, j⟩, B);
13:      y = (j == anslink[i]) ? +1 : -1;
14:      τ = min{C, max{ 0, (1 − y · scr) / ||ω(⟨i, j⟩, B)||^2 }};
15:      if (j != anslink[i]) {
16:        w ← w + y τ ω(⟨i, j⟩, B);
17:        break;
18:      }
19:      else if ( (j == anslink[i]) && (scr >= 0) ) {
20:        estlink[i] = j;
21:        w ← w + y τ ω(⟨i, j⟩, B);
22:      }
23:      else  //  (j == anslink[i]) && (scr < 0) の場合
24:        break;
25:    }
26:    i = pop(stack);
27:  }
28:  push(stack, i);
29:  push(stack, j);
30: }
```

図 4　擬似コード—Bilateral Parsing の決定的な解析側の学習アルゴリズム
Fig. 4　Pseudo code for training the deterministic parsing of the bilateral parser.

```
1: // 出力: w: 更新された重みベクトル
2: //       estlink[]: 推定された係り先文節の添字番号が格納された配列
3: function bilateral_comp_learn(w, C, B[], N, anslink[], estlink[])
4: for (i = N - 3; i >= 0; i--) {
5:   if (estlink[i] != -1)
6:     continue;  // すでに係り先が決定している場合は係り先の学習は不要
7:   j = i + 1;
8:   while (j != -1) {
9:     if (j != anslink[i]) {
10:      link_scr = w · ω(⟨i, anslink[i]⟩, B);
11:      nlink_scr = w · ω(⟨i, j⟩, B);
12:      τ = min{C, max{ 0, (1 − link_scr + nlink_scr)
13:                                / ||ω(⟨i, anslink[i]⟩, B)−ω(⟨i, j⟩, B)||^2 }};
14:      w ← w + τ{ ω(⟨i, anslink[i]⟩, B) − ω(⟨i, j⟩, B) };
15:    }
16:    j = estlink[j];
17:  }
18: }
```

図 5　擬似コード—Bilateral Parsing の相対的な比較による解析側の学習アルゴリズム
Fig. 5　Pseudo code for training the preference-based parsing of the bilateral parser.

みベクトルを $\mathbf{w}$ とする．このとき，$\langle \mathtt{i}, \mathtt{j} \rangle$ をより正確に識別できるように重みベクトルを更新するための条件は $\mathbf{w} \cdot \omega(\langle \mathtt{i}, \mathtt{j} \rangle, \mathtt{B}) \geq 0$ である．つまり，分類した値が負をとる場合は重みベクトルは更新されない．これは，文節対に係り受けの曖昧性があることを考慮しているためである．仮に，文節対に係り受けの曖昧性がある場合に正例として学習すると，よく似た形式の未知の文節対に対して Reduce 操作を行い，長距離依存の係り先同定に誤りが生じる可能性があるためである．

関数 `bilateral_comp_learn`（**図 5**）は，論文 3)，4) を基にして，決定的な解析で係り先が同定できなかった係り元について優先度学習を用いて重みベクトル $\mathbf{w}$ を更新するように変更を加えたものである．優先度学習とは次のようなものである．たとえば，文節列 `B[]` において，i 番目の係り先がまだ決まっていないものとする．このとき，文節対 $\langle \mathtt{i}, \mathtt{j} \rangle$ を正しい係り受けである文節対，$\langle \mathtt{i}, \mathtt{k} \rangle$ を誤った係り受けである文節対である

とする．優先度学習では，正しい係り受けである文節対の識別関数の出力値が，正しくない係り受けである文節対の識別関数の出力値よりも大きくなるように重みベクトルを更新する．つまり，$\mathbf{w}\cdot\omega(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B}) > \mathbf{w}\cdot\omega(\langle \mathtt{i},\mathtt{k}\rangle,\mathtt{B})$ のような関係になるように重みベクトルを更新する．ここで，更新前の重みベクトルと更新後の重みベクトルの違いが分かるように，前者を $\mathbf{w}_t$，後者を $\mathbf{w}_{t+1}$ と表記する．先に示した 2 つの文節対について，更新前の重みベクトルでは識別関数の出力値について大小関係が逆転していて，これを OPA で正しい関係になるように重みベクトルを更新するという設定を考える．このとき，$\mathbf{w}_{t+1}=\mathbf{w}_t+\tau\{\omega(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})-\omega(\langle \mathtt{i},\mathtt{k}\rangle,\mathtt{B})\}$ となる（ただし $\tau>0$）．$\mathbf{w}_{t+1}$ と $\mathbf{w}_t$ の関係式から，$\mathbf{w}_{t+1}\cdot\omega(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})=\mathbf{w}_t\cdot\omega(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})+\tau\{||\omega(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})||^2-\omega(\langle \mathtt{i},\mathtt{k}\rangle,\mathtt{B})\cdot\omega(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})\}$ である．$\omega(\cdot)$ は単位ベクトル，$\tau>0$ であることに注意すると，$\mathbf{w}_{t+1}\cdot\omega(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})\geq\mathbf{w}_t\cdot\omega(\langle \mathtt{i},\mathtt{j}\rangle,\mathtt{B})$ が成立する．また，同様にして，$\mathbf{w}_{t+1}\cdot\omega(\langle \mathtt{i},\mathtt{k}\rangle,\mathtt{B})\leq\mathbf{w}_t\cdot\omega(\langle \mathtt{i},\mathtt{k}\rangle,\mathtt{B})$ が成立することが確認できる．これらの性質は次のことを意味する．まず，ある文節対が一貫して正しい係り受けになる場合，ある程度学習が進むと識別関数の出力値が大きくなり，負の値から正の値をとりやすくなる．この場合，決定的な係り受け解析において，正例として重みベクトルを更新する対象になるため，決定的な係り受け解析のほうで係り受けの同定が行える．一方，ある文節対が係り受けに曖昧性を持つ場合，i) 決定的な解析側の学習アルゴリズムにおいて負例として重みベクトルを更新する対象になるケースがあることと，ii) 一貫して正しい係り受けになる文節対とは異なり，識別関数の出力値が小さくなるように重みベクトルが更新されるケースがある．これらのことから識別関数の出力値はあまり増加せず，負のままになりやすい．したがって，曖昧性を持つ文節対は相対的な比較による解析を行うまで係り先の同定が遅延されることが期待できる．

提案手法では，決定的な解析の重みベクトルの更新，および，相対的な比較による重みベクトルの更新を行っているが，それらの重みベクトルは同一の $\mathbf{w}$ である．一方で，それぞれの解析について別々の重みベクトルで学習することも考えられる．つまり，関数 `bilateral_sr_train` で得た重みベクトルを $\mathbf{w}_{sr}$，関数 `bilateral_comp_learn` で得た重みベクトルを $\mathbf{w}_{comp}$ として学習を行うことである．しかしながら，この場合，$\mathbf{w}_{sr}$ を用いた識別関数の出力値は，学習の結果，識別関数がすべての文節対について負の値をとるという問題が生じる．理由を以下に述べる．まず，関数 `bilateral_sr_train` の説明で述べたように，正しい係り受けである文節対を学習事例として重みベクトルが更新される条件は，更新前の重みベクトルで分類した値が正である（正しく分類されている）ときのみである．この関数は，関数 `bilateral_comp_learn` と重みベクトルを共有することで，関数 `bilateral_comp_learn` により，一貫して正しい係り受けを持つ文節対の識別関数の出力値が 0 より大きい値をとりやすくなるようにしている．しかしながら，前提として 2 つの関数は別々の重みベクトルで学習すると仮定しているため，関数 `bilateral_sr_train` では，最終的に正しい係り受けではない文節対についての学習事例（負例）のみしか重みベクトルの更新対象にならなくなる．$\mathbf{w}_{sr}$ を用いた識別関数の出力値が，どのような文節対についても負の値をとるということは，決定的な解析では，末尾の 2 つの文節を除いたすべての文節について係り先が決定しないことを意味するため，単純に相対的な比較による解析を行うことと同じになり，係り受け解析の高速化にまったく貢献しない．したがって，関数 `bilateral_sr_train` と関数 `bilateral_comp_learn` は，同一の重みベクトルを使用する必要がある．

ここまで，重みベクトルを更新するための学習アルゴリズムとして OPA を使用した擬似コードを示したが，提案手法は別の重みベクトルを更新する学習アルゴリズムを使用することも可能である．ただし，その学習アルゴリズムは以下の 3 つの条件を満たす必要がある．

(1) 決定的な解析，および，相対的な比較による解析のそれぞれの学習が，オンラインで重みベクトルを更新するものである．

(2) 相対的な比較による解析側の重みベクトルの更新則が，$\mathbf{w} \leftarrow \mathbf{w}+\tau\{\ \omega(\langle \mathtt{i},\mathtt{anslink[i]}\rangle,\mathtt{B})-\omega(\langle \mathtt{i},\mathtt{j}\rangle,B)\ \}$ を満たす（ただし，$\mathtt{j}\neq\mathtt{anslink[i]}$．また，$\tau$ は 0 より大きい値）．

(3) 決定的な解析側において，正しい係り受けである文節対を学習事例として重みベクトルを更新する場合，識別関数の出力値をなるべくマージン γ（ただし，γ は 0 より大きい定数）以上になるようにする．

条件 (1) は，決定的な解析側の学習アルゴリズムと相対的な比較による解析側の学習アルゴリズムを独立させるために必要である．また，条件 (2) は，関数 `bilateral_comp_learn` の説明で述べたように，相対的な比較による解析側の重みベクトルの更新により，正しい係り受けを持つ文節対に関する識別関数の出力値が大きくなるようにするために必要である．条件 (3) は，一貫して正しい係り受けを持つ文節対について，決定的な解析アルゴリズムで係り受け同定ができるようにするために必要である．この理由について以下に示す．まず，関数 `bilateral_sr_train` の説明で述べたように，正しい係り受けである文節対を学習事例として重みベクトルが更新される条件は，更新前の重みベクトルで分類した値が正である（正しく分類されている）ときのみである．したがって，パーセプトロンのように，誤って分類した場合にのみ重みベクトルを更新する学習アルゴリズムでは重みベク

```
1: // 出力: estlink: 推定された係り先文節番号が格納された配列
2: function bilateral_parsing(w, B[], N)
3: estlink = [-1] * N; // 文節数分 −1 が入った配列
4: ⟨estlink⟩ = bilateral_sr_parsing(w, B, N, estlink);
5: ⟨estlink⟩ = bilateral_comp_parsing(w, B, N, estlink);
```

図 6　擬似コード—Bilateral Parsing
Fig. 6　Pseudo code for the bilateral parser.

トルが更新される条件を満たさないため正例は学習されない．このような場合でも，関数 bilateral_comp_learn により，一貫して正しい係り受けを持つ文節対の識別関数の出力値が 0 より大きくなりやすくなるが，正例が学習されないことにより他の負例による影響を大きく受ける．そのため，学習が進んだ結果，ほとんどの文節対の識別関数の出力値が負の値をとり，決定的な解析アルゴリズムで係り受け同定が行えなくなる．したがって，正しい係り受けを持つ文節対で識別関数の出力値が 0 より大きくなるものに対しては，ある程度マージンをとるように重みベクトルを更新する必要がある．なお，このようなマージンをとるオンライン学習手法は OPA のほかにマージンパーセプトロン[12),13)] がある．

4.2.2　解析アルゴリズム

提案手法の解析アルゴリズムについて示す．**図 6** は，解析アルゴリズムの本体で，4.2.1 項の提案手法の学習アルゴリズム（関数 train）によって得た重みベクトル **w** を用いて，入力された文節列 B[] の係り受け同定を行う．この関数は，決定的な解析側の係り受け同定の関数 bilateral_sr_parsing（**図 7** 参照），相対的な比較による解析側の係り受け同定の関数 bilateral_comp_parsing（**図 8** 参照）を呼び出すことで係り受け解析を行っている．これらの関数においても 4.2.1 項の提案手法の学習アルゴリズム（関数 train）によって得た重みベクトル **w** を用いている．

ここで，それぞれのアルゴリズムの時間計算量について示す．

関数 bilateral_sr_parsing の時間計算量は文節数に比例する．この理由について述べる．まず，関数 bilateral_sr_parsing と，基のアルゴリズムである颯々野の係り受け解析[1)] は，コード中の while 文が以下のように異なっているだけであることに注目する．

- 関数 bilateral_sr_parsing 中の while 文

```
while ( (i != -1)
        && ( ( (i == N - 2) && (j == N - 1) ) ||
```

```
1: // 出力: estlink[]: 推定された係り先文節番号が格納された配列
2: function bilateral_sr_parsing( w, B[], N, estlink[] )
3: push(stack, -1);  // −1 は番兵
4: push(stack,  0);
5: for (j = 1; j < N; j++) {
6:   i = pop(stack);
7:   while ( (i != -1)
8:           && ( ( (i == N - 2) && (j == N - 1) ) ||
9:                ( w·ω(⟨i,j⟩,B) > 0 ) ) ) {
10:    estlink[i] = j;  // 推定された係り先文節の添字番号を代入
11:    i = pop(stack);
12:  }
13:  push(stack, i);
14:  push(stack, j);
15: }
```

図 7　擬似コード—Bilateral Parsing の決定的な解析側の係り受け同定
Fig. 7　Pseudo code for the deterministic parsing of the bilateral parser.

```
( w·ω(⟨i,j⟩,B) > 0 ) ) )
```

- 颯々野の係り受け解析[1)] アルゴリズム中の while 文

```
while ( (i != -1)
        && ( (j == N - 1) || ( w·ω(⟨i,j⟩,B) > 0 ) ) )
```

このことから，while 文のネストの実行回数は，関数 bilateral_sr_parsing のほうが少ないか同じである．さらに，颯々野の係り受け解析の時間計算量は文節数に比例することから，関数 bilateral_sr_parsing の時間計算量は文節数に比例することがいえる．

また，関数 bilateral_comp_parsing の時間計算量の上限は文節数の二乗に比例する．この理由は，N 個（$N > 1$）の文節列 B[] を関数 bilateral_sr_parsing で解析した時点で，最も係り先が決定できなかった場合の estlink[] を考える．このとき，estlink[N-2]=N-1 を除き，estlink[] の要素は -1 となる．この estlink[] を入力として関数 bilateral_comp_parsing で係り先を決定する過程は，基にした解析アルゴリズ

```
1: // 出力: estlink[]: 推定された係り先文節番号が格納された配列
2: function bilateral_comp_parsing( w, B[], N, estlink[] )
3: for (i = N - 3; i >= 0; i--) {
4:   if (estlink[i] != -1)
5:     continue;  // すでに係り先が決定している場合は係り先の探索は不要
6:   max_scr_idx = i + 1;
7:   max_scr = w · ω(⟨i,i + 1⟩,B);
8:   j = estlink[i + 1];
9:   while (j != -1) {
10:    scr = w · ω(⟨i,j⟩,B);
11:    if (scr > max_scr) {
12:      max_scr_idx = j;
13:      max_scr = scr;
14:    }
15:    j = estlink[j];
16:  }
17:  estlink[i] = max_scr_idx;
18: }
```

図 8　擬似コード—Bilateral Parsing の相対的な比較による解析側の係り受け同定
Fig. 8　Pseudo code for the preference-based parser of the bilateral parser.

ム[3),4)] とちょうど同じである．そして，基のアルゴリズムの時間計算量の上限は文節数の二乗に比例することから，関数 bilateral_comp_parsing の時間計算量の上限は文節数の二乗に比例することがいえる．

なお，関数 bilateral_sr_parsing の文節対 $\langle \mathtt{i}, \mathtt{j} \rangle$ の分類スコア $\mathbf{w} \cdot \omega(\langle \mathtt{i}, \mathtt{j} \rangle, \mathtt{B})$ は，動的素性が一致している場合，関数 bilateral_comp_parsing の識別関数の出力値としても再度使用することができる．5 章での実験で用いた解析アルゴリズムは，動的素性情報込みの文節対をキーとして識別モデルのスコアを保存することで動作の効率化を図っている．

5. 実　　験

5.1 実験設定

京都テキストコーパス 4.0 [⋆1] を以下の 3 つに分けて実験を行った．

- 訓練データ：一般記事[⋆2] 1 月 1，3–11 日，社説 1–8 月，合計 24,280 文，234,639 文節
- 開発データ：一般記事 1 月 12，13 日，社説 9 月，合計 4,833 文，47,571 文節
- 評価データ：一般記事 1 月 14–17 日，社説 10–12 月，合計 9,284 文，89,874 文節

これらの記事の分け方は，論文 3) と同じである．

係り受け解析手法としては，提案手法である Bilateral Parsing，比較手法である颯々野の線形時間係り受け解析，同じく比較手法である相対モデル[3)] を使用した．なお，論文 3) については相対モデルと決定的な解析手法の 1 つであるチャンキングモデルの組合せ処理も提案しているが，5.2 節から 5.4 節については Bilateral Parsing と，Bilateral Parsing の構成要素の基となった係り受け解析手法についての比較を行うため，組み合わせ処理を行っていない解析手法を使用している．論文 3) で提案されている組み合わせ処理を用いた場合の提案手法との比較は 5.5 節で行う．

次に使用した学習器について述べる．Bilateral Parsing が Online Passive-Aggressive Algorithm の PA-I; OPA [10)]，颯々野の線形時間係り受け解析がもとの論文に従い Support Vector Machine; SVM を使用した．また，相対モデルについては，容易にカーネル関数が使える点から Margin Infused Relaxed Algorithm; MIRA [14),15)] を使用した．ただし，機械学習の性質の違いによって実験結果が大きく変わる可能性があるため，颯々野の線形時間係り受け解析，ならびに，相対モデルについても OPA で学習した実験も行った．この学習についても Bilateral Parsing の学習と同様に，論文 11) に掲載されている重みベクトルの平均化を行った．

相対モデルの学習については，論文 16) と同様に，訓練データから文節列 B[] と正しい係り先文節の添字番号が格納されている配列 anslink[] を取り出し，文末の文節から順に係り元，正しい係り先，係り先候補集合（係り元の後方に存在するすべての文節）を用いて重みベクトルを更新する．また，それぞれの係り受け解析手法で用いる識別モデルのカーネルは 3 次の多項式を使用した．なお，今回の実験で用いる識別モデルは，すべて Polynomial Kernel Inverted Representation; PKI [17)] による計算の高速化を行っている．

以下にパラメータの設定について述べる．それぞれの学習器は，コストのパラメータ（C）を定める必要がある．また，学習器に OPA，もしくは，MIRA を用いた場合，反復回数 I を定める必要がある．これらのパラメータは，$C = \{\text{1e-06}, \text{1e-05}, \cdots, 10\}$，$I = \{1, 2, \cdots, 30\}$

⋆1 http://nlp.kuee.kyoto-u.ac.jp/nl-resource/corpus.html
⋆2 ただし，以下の ID を持つ文は文節番号とその文節の係り先番号が同一であるというタグ付けの誤りがあったため，訓練データから除外した；950101159-010，950106177-017，950106192-002.

表 1　結果　係り受け正解率，文正解率，解析時間，分類器の呼び出し回数
Table 1　Results of dependency accuracy, sentence accuracy, parsing time and number of times that each classifier was called and number of support vectors for each classifier.

解析手法	係り受け正解率（%）	文正解率（%）	解析時間（秒）	分類器の呼び出し回数	サポートベクトル数
Bilateral（OPA, I=29, C=0.1）	91.09	55.13	486	143,370	198,375
相対モデル（MIRA, I=25, C=1e-05）	91.19	54.84	1179	265,666	228,426
颯々野（SVM, C=0.0001）	90.74 ‡	54.06 ‡	146	112,975	75,117
相対モデル（OPA, I=29, C=1e-05）	91.12	54.64	1176	264,947	231,630
颯々野（OPA, I=9, C=0.0001）	90.92 ‡	55.01	193	113,284	97,197

‡：二項検定 [有意水準 1%両側] で，提案手法との有意差が認められるもの．

のうちで開発データの係り受け正解率が最も高くなる値を使用した．

学習に用いた素性は，CaboCha 0.53 中にある素性抽出プログラム selector.pl の出力を使用した．また，他の語彙的な素性は固有のコーパスに過度に依存する可能性があるため使用していない．ただし，同素性抽出プログラムで出力される動的素性は使用している．動的素性とは，係り元もしくは係り先において解析途中ですでに得られている係り受けをもとにした素性のことである．Bilateral Parsing，および，颯々野の線形時間係り受け解析では A. 係り元にすでに係る文節，B. 係り先にすでに係る文節についての動的素性を使用した．また，相対モデルは，B. 係り先にすでに係る文節と C. 係り先が係る文節についての動的素性を使用した．

なお，実験は Xeon E5504，主記憶 32 GByte の Linux 上で行った．また，実装は C++ で行い，gcc 4.4.3 の O3 オプションでコンパイルしたプログラムを実行している．

5.2　実 験 結 果

前の節で示した 3 つの手法の評価データにおける結果を**表 1** に示す．以降，表中では，提案手法である Bilateral Parsing は「Bilateral」，颯々野の線形時間係り受け解析は「颯々野」と略記する．ここでの係り受け正解率は，文末の 1 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，文正解率は，文単位で全体の文節の係り先が正しく同定できたものの割合を示す．また，提案手法以外の係り受け正解率，文正解率については，二項検定により，両側 1%で提案手法との有意差が認められるものについてダブルダガー（‡）を付記している．加えて，本稿での目的は相対モデルの高速化であるので，評価データ 9,284 文の解析に要する時間（秒），分類器の呼び出し回数，および，分類器を構成するサポートベクトル数を記載している．なお，この解析時間には素性抽出プログラム selector.pl の実行時間は含まれていない．また，4.2.2 項で述べたように，Bilateral Parsing は同一の素性ベクトルについて分類器が 2 回呼び出されることがあるが，これは識別モデルのスコアを再利用することで 1 回の呼び出しで抑えるという工夫を施している．

提案手法の Bilateral Parsing は，係り受け正解率の点で颯々野の線形時間係り受け解析に比べ有意に向上している．ただし，文正解率については，同一の学習器を用いた場合に有意差が認められなかったことから同等の性能であると考える．

次に Bilateral Parsing と相対モデルとの比較を行う．Bilateral Parsing は相対モデルと比べて文正解率では若干向上しているものの，係り受け正解率では若干下がっている．ただし，これらの二項検定による有意差は認められなかった．したがって，Bilateral Parsing と相対モデルはほぼ同等の性能であると考える．

5.3　解析時間についての比較

解析時間について見てみる．一般に，解析時間はそれぞれの解析手法によって呼び出される分類器の回数と，1 回あたりの分類器の呼び出し時間の積に比例する．また，SVM に代表されるカーネルマシンを用いた分類器の呼び出し時間は，素性の組合せを展開するといった高速化手法を用いない場合，分類器の持つサポートベクトル数に比例する．

表 1 の解析時間について見ると，颯々野の線形時間係り受け解析が最も速い．Bilateral Parsing は，線形時間係り受け解析を行った後で，相対的な比較による係り受け解析を行っているため，線形時間係り受け解析よりも遅くなる．一方，Bilateral Parsing と相対モデルを比較すると前者のほうが 2.4 倍程度高速に動作していることが確認できた．この比率については，MIRA で学習した相対モデル，ならびに，OPA で学習した相対モデルでも同じであった．

表 1 に示した解析時間は，多項式カーネルの厳密計算ができ，かつ，簡易な実装である PKI を使用した．このとき，Bilateral Parsing は決定的な解析での重みベクトルの更新と相対的な重みベクトルの更新の両方を行うため，Bilateral Parsing における分類器のサポートベクトル数は相対モデルのものに近い値となっている．そのことが理由で Bilateral Parsing の分類器の呼び出し回数が颯々野の線形時間係り受け解析の呼び出し回数の 1.27 倍程度であるにもかかわらず，解析時間においては 2.5 倍以上となった．なお，多項式カーネルの計算についての高速化として，Polynomial Kernel Expanded Representation; PKE [17] や SplitSVM [18] がある．このような高速化手法を用いた場合，1 回あたりの分類器の呼び出し時間はサポートベクトルに依存しないため，それぞれの解析手法における解析時間は呼び出される分類器の回数に比例するようになることが期待できる．

提案手法が相対モデルと比較して分類器の呼び出し回数を減らすことができた理由について，いくつかの調査を行った．まず，Bilateral Parsing が相対モデルに比べて高速に動作するのは，決定的な解析の時点で係り先が決定している係り元文節が多いことが考えられる．そこで，実際に決定的な解析で，どれくらいの文節数が，どれぐらい正確に定めることができているのかを調べた．Bilateral Parsing の決定的な解析のみを行った結果を用いて適合率・被覆率を求めた．これらは論文 19) を参考にして次のように定めた．(適合率) = (解析器が出力した文節のうち正解した数)/(解析器が係り先を出力した文節数)，(被覆率) = (解析器が係り先を出力した文節数)/(末尾の文節を除いた総文節数)．ここで，`estlink[]` の要素で −1 のものは係り先を出力した文節とカウントしないことを強調する．結果は適合率が 0.9274，被覆率が 0.9491 であった．これより，ほとんどの文節について係り先が決定的な解析の時点で精度良く決定できていることが確認できる．

先の調査で，ほとんどの文節について係り先が決定的な解析の時点で決定できていることを確認した．しかしながら，相対的な比較による解析まで係り先が決定できない文節について，その係り先候補が非常に多い場合，相対的な比較による解析による時間が支配的になり，高速化に寄与しない可能性がある．そこで，相対的な解析の際にどのくらい係り先候補があるかについて調べた．まず，評価データを解析したときの，相対的な処理で探索した係り先候補の総数について示すと 18,008 文節であった．なお，相対的な処理の素朴な実装では，係り先候補の総数 18,008 回の文節対素性ベクトルを生成して分類器を呼び出すことになるが，4.2.2 項で述べたように，決定的な解析で同一の素性ベクトルが生成される場合，識別モデルのスコアを再利用することができる．実際，18,008 回のうち 2,050 回はスコアの再利用を行うことにより高速化を図ることができた．また，相対的な処理で探索した係り元の総数は 4,099 文節であるので，1 つの係り元に対して平均 4.4 文節程度の係り先候補が存在することになる．これは評価データの平均文節数が 9.7 文節であることから，候補数としては比較的多いといえる．ただし，実際には，決定的な処理での分類器の呼び出し回数は，相対的な処理での分類器の呼び出し回数よりも多い．評価データで解析したとき，決定的な処理における分類器の呼び出し回数は 127,412 回であった．決定的な処理と相対的な処理にかかった時間も分類器の呼び出し回数を反映しており，それぞれ 428 秒と 58 秒となった．以上のことから，相対的な処理における係り先候補に対する探索は比較的多いが，適合率が高いことから相対的な比較による解析の時間が支配的になるということはないといえる．

次に，各手法における 1 文あたりの処理時間を**図 9** に示す．相対モデルは時間計算量の上限が文節数の二乗に比例するアルゴリズムであり，颯々野の線形時間係り受け解析は時間計算量が文節数に比例するアルゴリズムである．また，提案手法は時間計算量の上限が文節数の二乗に比例するアルゴリズムである．参考として，プロット図に，それぞれの手法の時間計算量に基づく回帰曲線（直線）を引いている．

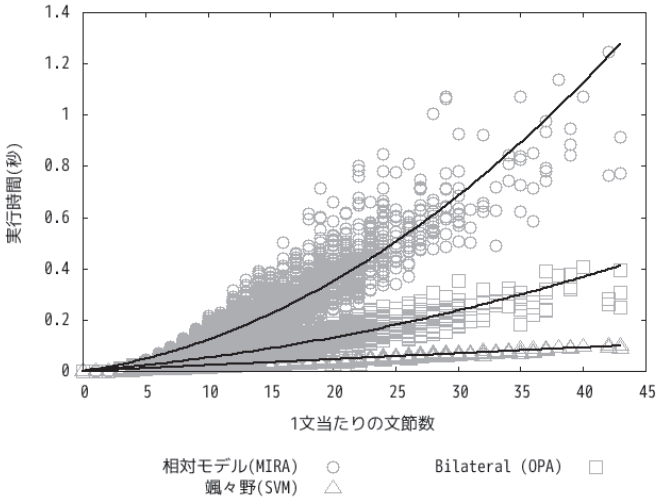

図 9　各手法における 1 文あたりの処理時間
Fig. 9　Parsing time per sentence on each method.

この図からも提案手法は，相対モデルに比べて高速に解析できていることが確認できる．これは，i) 決定的な解析の時点で 9 割方の文節の係り先が同定できる（被覆率が 0.9491 であることより）ことと，ii) 係り先は同定されている係り受けに対して非交差を満たすものに限定することから，決定的な解析の時点で比較的長距離の係り先が同定できる場合，相対的な比較による解析での係り先候補を大幅に減らすことができることによる．

5.4　距離ごとの比較

決定的な解析手法と相対的な比較による解析手法，および，提案手法は，係り受け解析の過程で係り先候補すべてを調べるか否かについて違いがある．この違いが係り先距離ごとの解析性能にどのような影響が現れるかについて調べた．**表 2** に係り先距離ごとでの F 値，精度，再現率を示す．ただし，距離 n での精度は，係り受け解析器が出力した距離 n の係り受け結果のうちで正解だった割合を表す．また，距離 n の再現率は，正解データ中の距離 n の係り受けのうちで係り受け解析器が正しい係り受けを出力した割合を示す．な

表 2　係り先距離ごとの比較：F 値（精度/再現率）（%）
Table 2　Relation between dependency distance and accuracy (F-measure/Precision/Recall).

解析手法	1	2–5	6–9	10 以上
Bilateral [決定的のみ]	97.12	80.59	66.82	56.97
（OPA, I=29, C=0.1）	(97.01/97.24)	(85.45/76.26)	(76.50/59.31)	(74.31/46.20)
Bilateral [決定的と相対的な比較]	97.19	81.83	73.55	73.71
（OPA, I=29, C=0.1）	(96.87/97.52)	(83.79/79.97)	(72.13/75.02)	(67.94/80.54)
相対モデル	97.06	82.04	74.06	74.94
（MIRA, I=25, C=1e-05）	(96.32/97.81)	(83.96/80.21)	(74.39/73.72)	(72.10/78.00)
颯々野	96.95	81.29	72.38	73.62
（SVM, C=0.0001）	(96.54/97.38)	(82.99/79.65)	(71.48/73.30)	(69.58/78.17)
相対モデル	97.05	81.95	73.82	74.47
（OPA, I=29, C=1e-05）	(96.38/97.73)	(83.82/80.15)	(73.84/73.79)	(71.46/77.75)
颯々野	97.14	81.61	72.64	73.29
（OPA, I=9, C=0.0001）	(96.92/97.36)	(82.81/80.44)	(71.68/73.63)	(69.77/77.17)

お，提案手法 Bilateral Parsing は，決定的な解析の時点と，決定的な解析の後に相対的な比較による解析を行った時点の結果における F 値，精度，再現率を載せている．

Bilateral Parsing [決定的のみ] の結果について見る．この時点では，部分的な係り受けを出力するため，他手法と比較して再現率は低くなる．それでも，距離 1 の係り受けについては，精度 97.01%，再現率 97.24%と他手法に接近する値となっている．これは，日本語係り受けは近い文節に係りやすいという性質から直近の文節が係るか否かの判定においては，他の係り先候補との係り受けの曖昧性が生じることが少ないためであると考える．特に再現率 97.24%と高い値をとることから，直近の係り先を確定するのに相対モデルが必要とする係り先候補すべてを調べ上げる計算が大幅に削減できていることが分かる．

次に，Bilateral Parsing [決定的と相対的な比較] と颯々野の線形時間係り受け解析（SVM）について見ると，どの距離についても F 値で向上が見られる．4.1 節で述べたように，Shift-Reduce 法に決定的な解析では係り先候補のいくつかが係りやすい候補であったとしても，後方の文脈をいっさい考慮せずに近い候補から Reduce 操作を行うか否かを選択する必要がある．そのため，6 以上の距離といった長距離における再現率の低下を起こしやすい．提案手法では，決定的な解析を行いにくい文節については，相対的な比較による解析まで係り受け同定を遅らせるため，再現率の点で有利であったと考える．

5.5　独立した係り受け解析器の組合せ手法との比較

論文 3) では，決定的な解析による結果と，相対的な比較による解析による結果を組み合わせることにより，より正解率の高い係り受け解析結果を得る方法を提案している．基本的な考え方は以下のとおりである．決定的な解析では短距離の解析性能に優れ，相対的な比較による解析では長距離の解析性能に優れていることが知られている．そこで，決定的な解析で定めた d 以下の係り受けは無条件に採用し，それ以外は相対的な比較による解析の結果を採用するという組合せを行うというものである．ただし，組合せにより，非交差性が崩れる場合は，a) 決定的な解析を優先する，b) 相対的な比較による解析を優先する，の 2 つの場合を試みる．距離 d，および，解析結果の優先順序については開発データを用いて選択する．

表 3　結果　解析結果の組合せによる実験結果
Table 3　Results of model combination.

解析手法	係り受け正解率（%）	文正解率（%）
Bilateral（OPA, I=29, C=0.1）	91.09	55.13
相対モデル（MIRA, I=25, C=1e-05）	91.19	54.84
颯々野（SVM, C=0.0001）	90.74 ‡	54.06 ‡
Bilateral [決定的のみ] & 相対モデル（MIRA）（$d = 3$, 交差戦略=b）	91.35 ‡	55.12
颯々野（SVM）& 相対モデル（MIRA）（$d = 1$, 交差戦略=b）	91.13	54.45
颯々野（OPA）& 相対モデル（MIRA）（$d = 3$, 交差戦略=b）	91.25 †	54.99

‡：二項検定 [有意水準 1%両側] で，提案手法との有意差が認められるもの．
†：二項検定 [有意水準 5%両側] で，提案手法との有意差が認められるもの．

5.4 節では，Bilateral Parsing について決定的な解析と相対的な解析を行った結果を示した．この結果から，工藤らが提案した独立した係り受け解析器の組合せ手法を適用すると，Bilateral Parsing の決定的な解析のみの結果と相対モデルの解析結果を組み合わせることで係り受け正解率や文正解率が向上する可能性が考えられる．そこで，Bilateral Parsing の決定的な解析のみの結果と相対モデルの解析結果の組合せによる解析結果を求めた．結果を **表 3** に示す．表には，颯々野の線形時間係り受け解析と相対モデルの組合せによる解析結果についても載せている．表から，Bilateral Parsing の決定的な解析のみの結果と相対モデルの解析結果の組合せについて，係り受け正解率が提案手法の結果よりも有意に向上していることが確認された．また，学習器にもよるが，颯々野の線形時間係り受け解析と相対モデルの組合せによる解析結果でも，係り受け正解率が提案手法の結果よりも向上していることが示唆される．ただし，文正解率については提案手法との有意差は見られなかった[⋆1]．なお，Bilateral Parsing の決定的な解析のみの結果と相対モデルの解析結果の組合せが，颯々野の線形時間係り受け解析と相対モデルの組合せの結果よりも解析性能が若干上がっている

⋆1 論文 3) の実験結果においても，相対モデルと組合せ手法との比較で文正解率については有意差は見られなかった．

ことが確認された．これは，Bilateral Parsing の決定的な解析は，係り受けがあるか否かの判定が難しい箇所には無理に係り先を定めないためであると考える．

このように，決定的な解析結果と，相対的な解析結果の組合せは，係り受け正解率向上の点では貢献することが確認された．ここで，独立した係り受け解析器の組合せ手法と Bilateral Parsing の違いについて述べる．前者は決定的な解析と相対的な解析の両方を行う必要がある．このとき，相対的な解析が最も時間を要する．一方，後者は相対的な解析の部分の解析時間を短縮するための方法である．したがって，解析時間については後者の手法のほうが速いところが利点としてあげられる．逆に，前者の利点は，係り受け解析の組合せが固有の決定的な解析と相対的な解析結果に依らないところがあげられる．

6. 関 連 研 究

論文 21) では，コスト付きの Shift-Reduce 法とビームサーチを用いることにより，実行時の解析時間が文節数に比例するアルゴリズムを提案している．この方法は，探索空間内の同値の状態をまとめあげて解析するといった一種の動的計画法を用いて，それを用いないコスト付きの Shift-Reduce 法に比べて，同一のビーム幅でより多様な構文木を探索できるところが特徴である．一般に，コスト付きの解析における探索範囲は，Bilateral Parsing の前段階で用いられているような決定的な解析における探索範囲よりも最低限ビーム幅分は広くなる．したがって，決定的な解析の処理時間に大きく近づけることを目的として，コスト付きの解析を採用するのは得策とはいえない．ただし，Bilateral Parsing は，実験的には文節数に対して線形に近い解析時間を達成しているものの，相対的な比較による解析では解析時間の上限が二乗に比例するアルゴリズムを使用している．相対的な比較による解析の部分をコスト付きの決定的な解析に置き換えることにより，Bilateral Parsing の計算量を文節数に比例させるといった拡張は可能性として考えられる．

論文 22) では，英語の係り受け解析において，部分木の Most Probable Head を求める際にトーナメントモデル[27] を用いて候補の探索を行ってから決定的にアクションをとる手法を提案している．この方法は主辞候補集合の Most Probable Head を求めるところでトーナメントモデルを用いて相対的な比較を行っており，ちょうど広義の決定的な解析に該当する．そのため，本稿での Shift-Reduce 法での決定的な解析とは異なり，日本語係り受け解析での係り先候補集合の中から係り先候補を比較によって選択する[28] 状況に近い．

論文 23) では，Shift-Reduce 法による決定的な解析手法の 1 つである Nivre の方法[24] と全域木を求めることで係り受けを求める McDonald の方法[25] の組合せ手法として，一方の出力を他方の素性として使用する方法を提案している．なお，McDonald の方法は，根となる節点から到達できる有向辺を求めるときに接続する節点を節点集合から探索しているため，相対的な比較による解析に対応する．この手法は，一方の解析結果を他方の素性として組み入れるため，5.5 節で示した距離を基準とした解析結果の組合せ手法よりも多様な特徴量を基に最終的な係り受けを定めることができ，Bilateral Parsing と比べて解析性能の点で有利であるといえる．ただし，この手法は 5.5 節の組合せ手法と同様に，決定的な解析と相対的な比較による解析の両方を行う必要がある．一方で，Bilateral Parsing は相対的な解析の部分の解析時間を短縮する方法であるため，Bilateral Parsing は解析時間の点では有利である．

論文 26) では，Shift-Reduce 法による決定的な解析手法を行った後に，別途用意した分類器を用いて解析結果を修正する手法を提案している．なお，修正するために用いる分類器の出力ラベルは適用する修正ルールを表す．この手法では，決定的な係り受け解析，および，解析結果の修正のそれぞれの時間計算量が語数に対して線形であることから，Bilateral Parsing と比べて解析時間の点で有利である．一方，この手法は，修正ルールを人手で作成する必要があり，どのようにして網羅率が高くなるように修正ルールを定めるかについて難点がある．Bilateral Parsing は，Shift-Reduce 法で判定が難しい箇所は無理に係り先を定めることはしないため，後で誤った係り先を修正するということは行わない．したがって，提案手法は，修正ルールを別途用意する必要がない点で利点があるといえる．

提案手法の決定的な解析の時点での結果は，係り受け解析器において信頼性の高い解析結果を部分係り受けとして出力するものに似ている．訓練データサイズや素性設定が異なるので若干公平さに欠けるが，論文 20) によると，5.2 節で示した適合率・被覆率と同程度のスコアを達成する解析器をトーナメントモデル[28] を用いて実現できることが示されている．ただし，この方法は高精度の部分解析を求めるための手法であり，提案手法のような決定的な解析を行っていない．実際，トーナメントモデルによる解析手順は相対モデルの解析手順と同じである．そのため，解析速度の向上を目的として，決定的な解析の代わりにトーナメントモデルを利用することはできない．

相対モデルについては，論文 16) で優先度学習で事例を作る際に相対位置素性を加えることで係り元文節からの相対的な距離を反映させるようにした係り受け解析モデルがある．この手法では相対位置素性を加えることで係り受け正解率，文正解率が有意に向上したとの

報告がある[⋆1]．このモデルも相対モデルと同様に優先度学習で定式化されている．今回提案した決定的な解析方法との併用が可能かどうかについては興味深い課題である．

7. ま と め

本稿では，係り先候補の確信度に基づく解析方法での解析時間を減らすために，決定的な解析が容易な文節について先に係り先を定めた後に，残りの文節について係り先の相対的な比較による係り先の同定を行う手法を提案した．実験結果から，係り先候補の確信度に基づく解析方法の 1 つである相対モデルと比較してほぼ同等の解析性能を持ち，かつ，実行時間が 2.4 倍程度高速であることが確認された．

Shift-Reduce 法は構文解析に限らず，日本語固有表現抽出[29] や形態素解析[30] などにも応用されている．これらの分野に本手法が適用できないかについては今後の課題である．

謝辞　本研究は文部科学省グローバル COE プログラム「インテリジェントセンシングのフロンティア」，総務省戦略的情報通信研究開発推進制度地域振興型，および，日本学術振興会科研費基盤研究（C）22500129 の支援により行われた．

参 考 文 献

1) 颯々野学：日本語係り受け解析の線形時間アルゴリズム，自然言語処理，Vol.14, No.1, pp.3–18 (2007).
2) 工藤　拓，松本裕治：チャンキングの段階適用による日本語係り受け解析，情報処理学会論文誌，Vol.43, No.6, pp.1832–1842 (2002).
3) 工藤　拓，松本裕治：相対的な係りやすさを考慮した日本語係り受け解析モデル，情報処理学会論文誌，Vol.46, No.4, pp.1082–1092 (2005).
4) 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).
5) Nomoto, T.: A Generic Sentence Trimmer with CRFs, *Proc. ACL-08: HLT*, pp.299–307 (2008).
6) Sakaji, H., Sekine, S. and Masuyama, S.: Extracting Causal Knowledge Using Clue Phrases and Syntactic Patterns, *Proc. PAKM 2008*, pp.111–122 (2008).
7) Herbrich, R., Graepel, T., Bollmann-Sdorra, P. and Obermyer, K.: Learning Preference Relations for Information Retrieval, *Proc. ICML-98 Workshop: Text Categorization and Machine Learning*, pp.80–84 (1998).
8) 長尾　眞，佐藤理史，黒橋禎夫，角田達彦：自然言語処理（岩波講座 ソフトウェア科学 15），第 4 章，岩波書店 (1996).
9) Shawe-Taylor, J. and Cristianini, N.: *Kernel Methods for Pattern Analysis*, Chapter 2, Cambridge University Press (2004).
10) Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S. and Singer, Y.: Online Passive-Aggressive Algorithms, *Journal of Machine Learning Research*, Vol.7, pp.551–585 (2006).
11) Daumé III, H.: Practical Structured Learning Techniques for Natural Language Processing, Ph.D. Thesis, University of Southern California (2006).
12) Krauth, W. and Mézard, M.: Learning algorithms with optimal stablilty in neural networks, *Journal of Physics A*, Vol.20, No.11, pp.745–752 (1987).
13) Kazama, J. and Torisawa, K.: A New Perceptron Algorithm for Sequence Labeling with Non-local Features, *Proc. EMNLP 2007*, pp.315–324 (2007).
14) Crammer, K. and Singer, Y.: Ultraconservative Online Algorithms for Multiclass Problems, *Journal of Machine Learning Research*, Vol.3, pp.951–991 (2003).
15) McDonald, R., Crammer, K. and Pereira, F.: Online large-margin training of dependency parsers, *Proc. ACL-05*, pp.91–98 (2005).
16) 山本悠二，増山　繁：係り元文節からの相対的な距離を反映した統計的日本語係り受け解析，電子情報通信学会論文誌 D，Vol.J93-D, No.6, pp.1036–1047 (2010).
17) 工藤　拓，松本裕治：カーネル法を用いた言語処理における高速化手法，情報処理学会論文誌，Vol.45, No.9, pp.2177–2185 (2004).
18) Goldberg, Y. and Elhadad, M.: splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications, *Proc. ACL-08: HLT*, pp.237–240 (2008).
19) 藤尾正和，松本裕治：語の共起確率に基づく係り受け解析とその評価，情報処理学会論文誌，Vol.40, No.12, pp.4201–4212 (1999).
20) 岩立将和，浅原正幸，松本裕治：係り受け解析器の部分解析精度評価とその応用，情報処理学会研究報告，Vol.NL189, pp.41–48 (2009).
21) Huang, L. and Sagae, K.: Dynamic Programming for Linear-Time Incremental Parsing, *Proc. ACL 2010*, pp.1077–1086 (2010).
22) Kitagawa, K. and Tanaka-Ishii, K.: Tree-Based Deterministic Dependency Parsing – An Application to Nivre's Method, *Proc. ACL 2010*, pp.189–193 (2010).
23) Nivre, J. and McDonald, R.: Integrating Graph-Based and Transition-Based Dependency Parsers, *Proc. ACL-08: HLT*, pp.950–958 (2008).
24) Nivre, J., Hall, J., Nilsson, J., Eryiğit, G. and Marinov, S.: Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines, *Proc. CoNLL-X*, pp.221–225 (2006).
25) McDonald, R.: Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing, Ph.D. Thesis, University of Pennsylvania (2006).

⋆1 実際に 5.1 節の実験設定で学習を行ったところ，係り受け正解率 91.46%，文正解率 55.61%となり相対モデルに比べて有意に向上していることが確認された．

26) Attardi, G. and Ciaramita, M.: Tree Revision Learning for Dependency Parsing, *Proc. NAACL HLT 2007*, pp.388–395 (2007).
27) 飯田　龍，乾健太郎，松本裕治：文脈的手がかりを考慮した機械学習による日本語ゼロ代名詞の先行詞同定，情報処理学会論文誌，Vol.45, No.3, pp.906–918 (2004).
28) 岩立将和，浅原正幸，松本裕治：トーナメントモデルをを用いた日本語係り受け解析，自然言語処理，Vol.15, No.6, pp.169–185 (2008).
29) 山田寛康：Shift-Reduce 法に基づく日本語固有表現抽出，情報処理学会研究報告，Vol.NL179, pp.13–18 (2007).
30) 岡野原大輔，辻井潤一：Shift-Reduce 操作に基づく未知語を考慮した形態素解析，第 14 回言語処理学会年次大会論文集，pp.77–80 (2008).

(平成 22 年 9 月 23 日受付)
(平成 23 年 4 月 8 日採録)

山本　悠二

2008 年豊橋技術科学大学大学院修士課程知識情報工学専攻修了．現在，同大学院博士後期課程電子・情報工学専攻在学中．自然言語処理の研究に従事．

増山　繁（正会員）

1977 年京都大学工学部数理工学科卒業．1982 年同大学院博士後期課程単位取得退学．1983 年工学博士．1982 年日本学術振興会奨励研究員．1984 年京都大学工学部数理工学科助手．1989 年豊橋技術科学大学知識情報工学系講師，1990 年同助教授，1997 年同教授．2010 年 4 月同大学大学院工学研究科情報・知能工学専攻教授．自然言語処理，特に，テキストマイニング，情報抽出，テキスト自動要約等情報アクセス支援，および，アルゴリズム工学，特に，グラフ・ネットワークのアルゴリズム，AGV，列車スケジューリング等の研究に従事．