1B1-R-3-1

係り受け解析における状態書き換え規則適用の学習

Learning for Applying Rules that Rewrite States in a Dependency Parser

山岡歩[*1] 猪口博明[*1] 松本裕治[*2] 鷲尾隆[*1]
Ayumu Yamaoka Hiroaki Inokuchi Yuji Matsumoto Takashi Washio

[*1]大阪大学 産業科学研究所
The Institute of Scientific and Industrial Research, Osaka University

[*2]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

This paper presents a novel application of Graph Sequence Mining to Dependency Analysis in Natural Language Processing. Graph Sequence Mining is a method for finding common changes from sequences of graphs, and it have been applied to social networks, co-author networks, citation networks and so on. We applied a Graph Sequence Mining method to a transition-based dependency parser to mine rules for rewriting states in the parser. In this paper, we propose a method for deciding whether the rules are applied to rewrite states or not.

1. はじめに

様々な構造を持つデータから特徴的なパターンを取り出すマイニング手法が提案されている中で，グラフ系列からの頻出パターンマイニングが注目されはじめている [2, 3, 7, 10, 11]．図 1(a) に 4 つの状態と 5 つの ID（頂点 ID）からなるグラフ系列の例を示す．グラフ系列を用いてモデル化することが適している実世界データの例を挙げると，人間関係ネットワークや遺伝子ネットワークなどがある．我々は文献 [3] において，グラフ系列中のグラフは徐々に変化するという仮定のもとで，グラフ系列の集合から図 1(b) に示す部分グラフ系列をマイニングする手法 GTRACE (Graph TRAnsformation sequenCE mining) を提案し，エンロン社の電子メールデータに適用した．

また，我々は自然言語処理における係り受け解析を対象とするグラフ系列マイニングの応用を提案した [4]．これは状態遷移系に基づく係り受け解析を対象とし，正解の係り受け構造に至ることのできない状態から正解へ至ることのできる状態へ書き換える規則（書き換え規則）をマイニングする手法である．この手法では適用可能な全ての状態に対して書き換え規則を適用する．しかし，適用可能な状態の中には書き換え規則を適用することで正しい係り受け構造に至ることのできない状態へ遷移してしまう状態も存在する．そこで，本稿では書き換え規則を適切に運用する手法を提案し，その評価を行う．

2. 状態遷移系に基づく係り受け解析

係り受け解析は，自然言語処理の基本技術の 1 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きた [5, 8]．本節では，係り受け解析に用いられるモデルを説明し，状態遷移系に基づく解析手法を紹介する．

定義 1. n 文節からなる文 $x = \langle w_1, w_2, \cdots, w_n \rangle$ の係り受け構造は，有向グラフ $g = (V, E)$ で表される．ここで，$V = \{1, 2, \cdots, n\}$，及び $E \subseteq V \times V$ である[*1]． ■

連絡先: 山岡歩，大阪大学 産業科学研究所，567-0047 大阪府茨木市美穂ケ丘 8-1，yamaoka@ar.sanken.osaka-u.ac.jp

*1 日本語における係り受け構造の有向辺は，係り元 w_i から係り先 w_j $(i < j)$ への辺 (i, j) で表される．

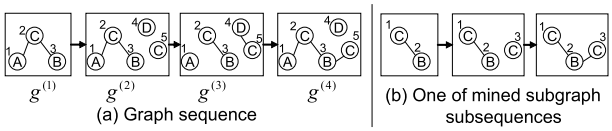


図 1 グラフ系列とマイニングされる部分グラフ系列

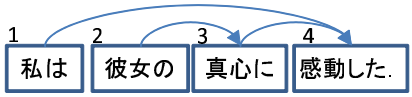


図 2 係り受け構造の例

本稿では，日本語の文節係り受け解析を対象とする．よって，係り受け構造において各文節はその後方に係り先を持つ．また，係り受け構造は頂点 n を根とする木で表される整形式であるとする．さらに，有向辺が交差しない係り受け構造を対象とする．

例 1. $x = \langle$ 私は, 彼女の, 真心に, 感動した. $\rangle$ という文の係り受け構造は図 2 の有向グラフで表される．

次に，状態遷移系に基づく係り受け解析器を定義する．ここでの入力は $x = \langle w_1, w_2, \cdots, w_n \rangle$ であり，出力は $g = (V, E)$ である．

定義 2. 状態遷移系に基づく係り受け解析器は $S = (C, T, c_s, C_F)$ で表される．ここで

- C は状態の集合であり，各状態 $c = (N, A) \in C$ は頂点集合 $N \subseteq \{1, 2, \cdots, n\}$ と辺集合 $A \subseteq N \times N$，
- T は遷移集合であり，$t \in T$ は $t : C \to C$ を満たす部分関数，
- c_s は初期関数であり，文 $x = \langle w_1, w_2, \cdots, w_n \rangle$ に対して $c_s(x) = (\{1\}, \emptyset)$，
- $C_F \subseteq C$ は最終状態の集合であり，$c_F \in C_F$ は n を根とする木である． ■

$S = (C, T, c_s, C_F)$ を状態遷移系とする．S における文 $x = \langle w_1, w_2, \cdots, w_n \rangle$ に対する状態遷移系列は以下を満たす系列 $C_{1,m} = \langle c_1, c_2, \cdots, c_m \rangle$ である．

Parse$(x = \langle w_1, w_2, \cdots, w_n \rangle)$
1) $c \leftarrow c_s(x)$
2) while $c \notin C_F$
3) $c \leftarrow [o(c)](c)$
4) return $c_m = (N, A)$

図 3　状態遷移系に基づく係り受け解析アルゴリズム

Transitions	
Arc	$(N, A) \Rightarrow (N, A \cup \{(i, j)\})$
	where $\{i, j\} = roots((N, A))$
Shift	$(N, A) \Rightarrow (N \cup \{\|N\| + 1\}, A)$
Preconditions	
Arc	c is not a tree, but a forest.
Shift	$j \neq n$, where $\{i, j\} = roots((N, A))$

図 4　遷移集合とその前提条件

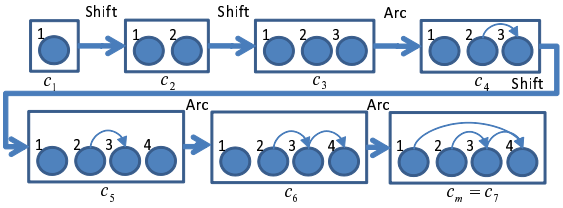


図 5　状態遷移系列

- $c_1 = c_s(x)$,
- $c_m \in C_F$,
- $1 < i \leq m$ である任意の i に対して，ある遷移関数 $t \in T$ が存在し，$c_i = t(c_{i-1})$ を満たす．

これ以降，状態 c に対する頂点集合を N_c，辺集合 A_c と表す．

定義 3. $S = (C, T, c_s, C_F)$ を状態遷移系とし，$N_c \subseteq N_{t(c)}$，及び $A_c \subseteq A_{t(c)}$ を満たすとき，S を逐次的と呼ぶ．■

定義 3 より，状態 c のグラフ (N_c, A_c) の頂点数，辺数は単調に増加する．また，状態 $c = (N_c, A_c)$ は森（木の集合）であり，グラフ (N_c, A_c) は最終状態 $c_m = (N_{c_m}, A_{c_m})$ の部分グラフである．

状態遷移系に基づく係り受け解析器のアルゴリズムを図 3 に示す．ここで o はある状態 c から次の状態へ遷移する関数 t を一意に定める関数である．具体的には，o は図 4 に示す Arc か Shift の遷移のいずれかを選択する関数であり，$roots$ は森 (N, A) の根のうち，頂点 ID が大きい 2 つの頂点対 $\{i, j\}$ $(i < j)$ を返す関数である．関数 o は $roots(c)$ の戻り値である i 番目の文節が j 番目に係るか (Arc)，否か (Shift) を判定する関数であるので，SVM(Support Vector Machine) などの二値分類器を用いて実装される．すなわち，i 番目の文節と j 番目の文節を特徴づけるベクトル $\Phi(\langle w_i, w_j \rangle) \in \mathbb{R}^n$ を入力とし，Arc か Shift かのいずれかを返す二値分類器で構成される．二値分類器の適用法に関しては，紙面の都合上割愛するが，文献 [5, 8] などを参照されたい．

例 2. 図 5 に初期状態から図 2 に示す係り受け構造を得るまでの状態遷移の系列を示す．初期状態 c_1 から最終状態 c_m が得られるまで，*Shift*, *Shift*, *Arc*, *Shift*, *Arc*, *Arc* の順に遷移が選択される．紙面の都合上，図 5 では単語の表記を省略している．

状態遷移系に基づく係り受け解析器の探索空間は，状態を節点，遷移を枝，初期状態を根，最終状態を葉とする木である．従って，初期状態から正解の最終状態に至る状態遷移系列は 1 通りである．また，木の分岐数は高々 2 である．関数 o は，2 つの枝のうち，どちらか一方を選択する関数であるので，関数 o が一度，枝の選択を誤ると正解の最終状態には辿り着けない．

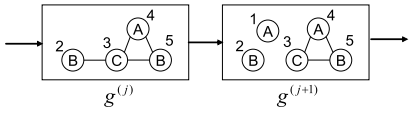


図 6　連続する 2 つのグラフの変化

頂点追加 $vi^{(j)}_{[u,l]}$	ラベルが l である頂点 u を $g^{(j)}$ へ追加し，$g^{(j+1)}$ へ変換
頂点削除 $vd^{(j)}_{[u,\bullet]}$	頂点 u を $g^{(j)}$ から削除し $g^{(j+1)}$ へ変換
頂点ラベル変更 $vr^{(j)}_{[u,l]}$	頂点 u のラベルを l に変更し，$g^{(j)}$ を $g^{(j+1)}$ へ変換
辺追加 $ei^{(j)}_{[(u_1,u_2),l]}$	頂点 u_1 と u_2 の間にラベル l の辺 (u_1, u_2) を追加し，$g^{(j)}$ を $g^{(j+1)}$ へ変換
辺削除 $ed^{(j)}_{[(u_1,u_2),\bullet]}$	頂点 u_1 と u_2 の間から辺を削除し，$g^{(j)}$ を $g^{(j+1)}$ へ変換
辺ラベルの変更 $er^{(j)}_{[(u_1,u_2),l]}$	頂点 u_1 と u_2 の間の辺ラベルを l に変更し，$g^{(j)}$ を $g^{(j+1)}$ へ変換

表 1　グラフ系列データのための変換規則

3. グラフ系列マイニング

図 1(a) はグラフ系列の例である．グラフ $g^{(j)}$ はグラフ系列中において j 番目のグラフである．頻出グラフ系列マイニング問題とは，グラフ系列の集合が与えられたとき，それらに頻繁に現れる構造の変化を列挙する問題である．文献 [3] において，我々はグラフ系列中のグラフは徐々に変化するという仮定のもとでグラフ系列を簡潔に表現する変換規則を提案した．具体的には，グラフ系列中の連続する 2 つのグラフ $g^{(j)}$ と $g^{(j+1)}$ で，その一部のみが変化し，残りの大部分は変化しないという仮定であり，例えば，図 6 の連続するグラフ $g^{(j)}$ と $g^{(j+1)}$ の間の変化は変換系列 $\langle vi^{(j)}_{[1,A]}, ed^{(j)}_{[(2,3),\bullet]} \rangle$ で表わされる．この系列は ID が 1 でラベルが A である頂点が $g^{(j)}$ に追加 (vi) され，ID が 2 と 3 である頂点間の辺が $g^{(j)}$ から削除 (ed) された結果，$g^{(j)}$ が $g^{(j+1)}$ に変換されたことを意味している．このようにグラフの頂点や辺が多い場合でも，上記の仮定のもとでグラフ系列を簡潔に表現することが可能である．2. 節で示した係り受け解析における状態遷移系列はグラフ系列そのものであり，その系列において，頂点あるいは辺が 1 つ追加されるだけなので，上記の仮定を満たす．

ここで，$g^{(j)}$ を $g^{(j+1)}$ へ変換する変換規則を $tr^{(j)}_{[o,l]}$ で表す．tr は表 1 に示す 6 種のいずれかである．また，o は変換される頂点あるいは辺であり，l は変換される頂点あるいは辺のラベルである．如何なるグラフ系列も表 1 に示す 6 種の変換規則で表現可能である．

変換規則からなる系列である変換系列から頻出変換部分系列を列挙するために，$s'_d \sqsubseteq s_d$ を変換系列 s'_d が変換系列 s_d の部分系列であると定義する．詳細な定義については，文献 [3] を参照されたい．

グラフ系列の集合 $DB = \{\langle tid, d \rangle \mid d = \langle g^{(1)} \cdots g^{(z)} \rangle\}$ に対し，変換部分系列 s_p の支持度 $\sigma(s_p)$ を $\sigma(s_p) = |\{tid \mid \langle tid, d \rangle \in DB, s_p \sqsubseteq s_d\}|/|DB|$ と定義する．ここで，tid はグラフ系列の ID であり，s_d は d の変換系列である．最小支

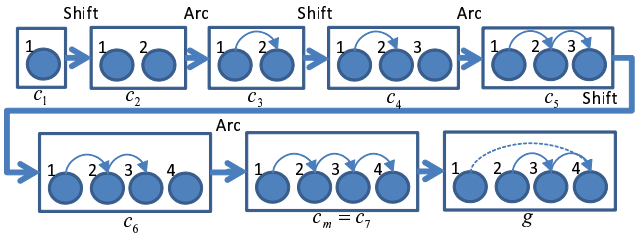


図 7　状態遷移系列からグラフ系列の生成

持度 σ' 以上の支持度を有する部分系列を頻出変換部分系列 (Frequent Transformation Subsequence: FTS) と呼び，文献 [2] において，我々は FTS を効率良く列挙する手法 GTRACE を提案した．

4.　書き換え規則の抽出

2. 節で述べた係り受け解析器は一度誤った状態に陥ると正しい係り受け構造を出力することができない．そこで，正解に至ることができない状態から正解へ至ることができる状態へ，状態を書き換えるための規則をグラフ系列マイニングを用いて抽出する．

抽出したい書き換え規則は，ある誤った状態に陥ったとき，正解に至る状態に戻すためにグラフを変換する規則である．そこで係り受け解析器の状態遷移の系列 $s = \langle c_1, c_2, \cdots, c_m \rangle$ に正解の係り受け構造 g を加えた，グラフ系列 $s = \langle c_1, c_2, \cdots, c_m, g \rangle$ の集合から頻出する構造の変化である FTS を見つけることを考える．$c_m = g$ であれば，c_m は正解の最終状態であり，c_m を g に変換する変換規則は存在しない．一方，$c_m \neq g$ の場合，c_m は不正解の最終状態であり，c_m を g に変換する変換規則は

- gに存在する辺の内, c_mに存在しない辺を追加する変換規則,
- c_mに存在する辺の内, gに存在しない辺を削除する変換規則.

である．いま抽出したい書き換え規則は，c_m から g への変換規則を含む FTS である．そこで，c_m から g への変換規則とそれ以外の変換規則を区別するために，g に存在する辺のうち，c_m に存在しない辺にラベル l_2 を，それ以外の辺に l_1 を付与する．

例 3. 図 7 に文 $x =$ ⟨ 私は, 彼女の, 真心に, 感動した. ⟩ に対する状態遷移系列の末尾にグラフ g を付加した例を示す．ただし，ここでの例は，c_2 から c_3 の遷移において，誤った遷移が関数 o によって選択された例である．この図において辺 (1, 4) は，c_m にはなく，g に存在する辺であるので，他の辺とは異なるラベルをもつことを表すために破線で図示されている．このグラフ系列の変換系列は以下で与えられる．

$\langle vi^{(0)}_{[1]} vi^{(1)}_{[2]} ei^{(2)}_{[(1,2),l_1]} vi^{(3)}_{[3]} ei^{(4)}_{[(2,3),l_1]} vi^{(5)}_{[4]} ei^{(6)}_{[(3,4),l_1]} ed^{(7)}_{[(1,2),\bullet]} ei^{(7)}_{[(1,4),l_2]} \rangle$

グラフ系列の集合からは，非常に膨大な数の FTS が得られるが，全てが書き換え規則として適しているわけではない．そこで得られた FTS を書き換え規則として採用するかを評価するために，書き換え規則を FTS ではなくルール形式で定義し，その確信度を以下のように定義する．

定義 4. FTS の集合 F が与えられたとき，書き換え規則を以下の条件を満たす $r =$ "$s_1 \Rightarrow s_2$" と定義する．

1. $s_1, s_2 \in F$.
2. ラベルが l_2である辺を追加する変換規則が s_1に含まれない.
3. ラベルが l_2 である辺を追加する変換規則 tr を s_1 の末尾に付加すると s_2 と等しくなる.

また，r の確信度を $conf(r) = \sigma(s_2)/\sigma(s_1)$ と定義する．　■

条件 1 は少数の文にしか適用できない規則を排除するためである．また条件 2 と条件 3 は正解に至る状態に戻すためにグラフを変換する規則を取り出すためである．

例 4. 書き換え規則 r の例を以下に示す.

$\langle vi^{(0)}_{[1]} vi^{(1)}_{[2]} ei^{(2)}_{[(1,2),l_1]} vi^{(3)}_{[4]} \rangle$
$\Rightarrow \langle vi^{(0)}_{[1]} vi^{(1)}_{[2]} ei^{(2)}_{[(1,2),l_1]} vi^{(3)}_{[4]} ei^{(4)}_{[(1,4),l_2]} \rangle$

上記の規則の条件部にはラベルが l_2 である辺の追加の変換規則が含まれていない．また，条件部と結論部の差は 1 つの変換規則であり，ラベルが l_2 である辺の追加の変換規則を条件部の末尾に付加すると，結論部と等しくなる．

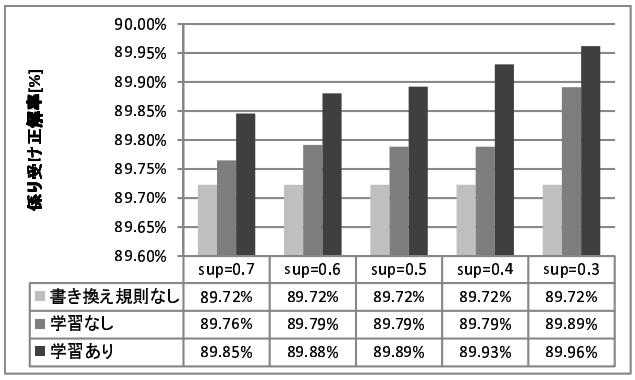
例 4 で与えられた書き換え規則 r があり，例 3 の状態遷移が c_6 まで遷移していたとすると，r を c_6 に適用して，辺 $(1, 4)$ を追加し，辺 $(1, 2)$ を削除する．つまり，r により c_6 から新たな状態 $c_{m'} = g$ に遷移する．従って，書き換え規則を適用することにより正解へ至らない状態から正解へ至ることができる状態へ，状態を書き換えることができる．

5.　書き換え規則適用の学習

4. 節で得た書き換え規則は正解へ至ることができない状態を正解へ至ることができる状態へ書き換えることを目的とした規則である．ここで，状態遷移系列 $C_{1,m'} = \langle c_1, c_2, \cdots, c_{m'} \rangle$ $(1 \leq m' \leq m)$ の変換系列を s' とし，書き換え規則 $r =$"$s_1 \Rightarrow s_2$" を状態 $c_{m'}$ に対して適用することを考える．このとき，$s_1 \sqsubseteq s'$ であれば，$c_{m'}$ に r を適用可能である．文献 [4] では，$s_1 \sqsubseteq s'$ を満たすとき，状態 $c_{m'}$ に対して必ず書き換え規則 r を適用する．しかし，$c_{m'}$ に対して r を適用することで正解へ至ることができない状態へ書き換えてしまう場合がある．そこで，書き換え規則を適用するか否かを分類器に学習させることで書き換え規則を適切に運用する事を考える．書き換え規則を適用するか否かは二値分類問題であるので, 二値分類器である SVM(Support Vector Machine) を利用できる．ここで，グラフ編集距離 [9] に基づき，ある状態 c_i と正解の状態 g の編集距離を $dist$，c_i に書き換え規則を適用した状態 c'_i と g の編集距離を $dist'$ とする．本稿では，書き換え規則を適用することによって $dist > dist'$ となる場合を正例とし，$dist \leq dist'$ となる場合を負例とする．いま，書き換え規則によって i 番目の文節と j 番目の文節の間にある辺 (i, j) が削除され，i 番目の文節と k 番目の文節の間にある辺 (i, k) が追加されたとすると，SVM の訓練データとテストデータは，i 番目の文節と j 番目の文節を特徴付けるベクトル $\Phi(\langle w_i, w_j \rangle) \in \mathbb{R}^n$ と，i 番目の文節と k 番目の文節を特徴付けるベクトル $\Phi(\langle w_i, w_k \rangle) \in \mathbb{R}^n$ を連結した $2n$ 次元のベクトルとする．

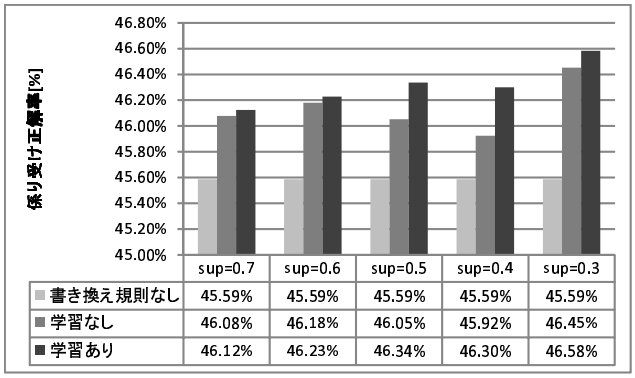
6.　評価実験

京大コーパス Version 4.0[6] を使用して評価実験を行った．1 月 1 日から 1 月 8 日までのデータを係り受け解析器内の SVM の訓練データとした．ここでの SVM には CaboCha-0.60 [1] 内の SVM を使用し，SVM のパラメータには CaboCha-0.60 のデフォルト値を設定した．また 1 月 9 日から 1 月 17 日までの 9 日分のデータのうち，1 日分のデータを除いた 8 日分のデータを書き換え規則の集合を抽出する訓練データとし，さらに同データより書き換え規則適用の学習のための訓練データを



	sup=0.7	sup=0.6	sup=0.5	sup=0.4	sup=0.3
書き換え規則なし	89.72%	89.72%	89.72%	89.72%	89.72%
学習なし	89.76%	89.79%	89.79%	89.79%	89.89%
学習あり	89.85%	89.88%	89.89%	89.93%	89.96%

図 8　係り受け正解率



	sup=0.7	sup=0.6	sup=0.5	sup=0.4	sup=0.3
書き換え規則なし	45.59%	45.59%	45.59%	45.59%	45.59%
学習なし	46.08%	46.18%	46.05%	45.92%	46.45%
学習あり	46.12%	46.23%	46.34%	46.30%	46.58%

図 9　文正解率

作成した．ここでの学習に利用した SVM は解析器内の SVM と同様，CaboCha-0.60 内の SVM を使用した．除いた 1 日分のデータをテストデータとし，これを 9 回繰り返した．

表 2 に学習に用いた素性を示す．ここで，主辞とは文節において品詞が特殊，助詞，接尾辞でないものの中で最も文末近くにある形態素であり，語形とは文節において品詞が特殊でないものの中で最も文末近くにある形態素である．

係り受け解析器の精度を評価する場合，一般に係り受け正解率と文正解率が用いられる [8]．係り受け正解率とは，文末の 2 文節を除く全文節のうち，係り先が正しい文節の割合であり，文正解率とは，解析対象となった文のうち，文全体の係り受け構造が正しい文の割合である．

図 8 と図 9 はそれぞれ係り受け正解率と文正解率を示している．横軸は書き換え規則を抽出する際に与えた最小支持度 *sup* である．また，学習なしとは書き換え規則の条件部を含むすべての状態に書き換え規則を適用した場合で，学習ありとは書き換え規則を適用するか否かを判定する SVM を導入した場合である．それぞれの図において学習ありの場合における係り受け正解率および文正解率が，学習なしの場合におけるそれらを上回っている．また，最小支持度が減少するに従って，正解率が上昇しているが，これは取り出される書き換え規則の数が増加し，より多くの誤りが修正されているためである．以上の実験結果は書き換え規則適用の学習が有効であることを示している．しかし一方で，学習ありの場合と学習なしの場合の正解率の差は係り受け正解率で最大 0.14%，文正解率で最大 0.38% と十分な改善とはいえず，改良の余地が残されていると考える．

静的素性	文節（係り元，書き換え前の係り先，書き換え後の係り先）	主辞（見出し，品詞，品詞細分類，活用，活用形），語形（見出し，品詞，品詞細分類，活用，活用形），括弧，句読点，文節の位置（文頭，文末）
	文節間	距離，助詞，括弧，句読点
動的素性	文節（係り元，書き換え前の係り先，書き換え後の係り先）と係り受け関係のある文節の語形見出し	

表 2　使用した素性

7. まとめ

本稿では正解に至ることのできない状態から正解に至ることのできる状態へ状態を書き換えるために抽出した書き換え規則を適切に運用する手法を提案した．また，書き換え規則を適用するか否かを二値分類器を用いて学習することで，係り受け正解率及び文正解率を改善できることを評価実験によって示した．今後の課題としては書き換え規則を適用するか否かを判定する SVM の精度を向上させ，書き換え規則による正解率の改善をより大きくすることである．そのためには，SVM に与える訓練データを増加させることが必要である．現在は 1 つの書き換え規則当たりの訓練データが文書数にして 30 から 200 程度と少ないため，ビーム探索法 [10, 11] などを利用することで訓練データの規模を拡大する予定である．

参考文献

[1] CaboCha-0.60. `http://code.google.com/p/cabocha/`
[2] 生田, 猪口, 鷲尾. : 逆探索法によるグラフ系列マイニングの高速化. 第 *3* 回データ工学と情報マネジメントに関するフォーラム, B10-3. (2011)
[3] A. Inokuchi and T. Washio. : A Fast Method to Mine Frequent Subsequences from Graph Sequence Data. *Proc. of IEEE Int'l Conf. on Data Mining*, pp. 303–312. (2008)
[4] 猪口, 山岡, 鷲尾, 松本, 浅原, 岩立, 賀沢. : 係り受け解析における状態書き換え規則のマイニング. *SIG-DOCMAS* 第 1 回研究会, (2011)
[5] T. Kudo and Y. Matsumoto. : Japanese Dependency Analysis using Cascaded Chunking. *Proc. of Conf. on Computational Natural Language Learning* (CoNLL 2002), pp. 63–69. (2002)
[6] 黒木, 長尾. : 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)
[7] Liang Huang, Kenji Sagae. : Dynamic Programming for Linear-Time Incremental Parsing. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1077-1086, (2010)
[8] S. Kubler, R. McDonald, and J. Nivre. : Dependency Parsing, *Morgan and Claypool Publishers*, (2009)
[9] A. Sanfeliu and K. Fu. : A Distance Measure Between Attributed Relational Graphs for Pattern Recognition. *IEEE Transactions on Systems, Man and Cybernetic*, Vol. 13, pp. 353–362. (1983)
[10] Y. Zhang and S. Clark. : A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing. *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing* pp. 562-571, (2008)
[11] Y. Zhang and S. Clark. : Syntactic Processing Using the Generalized Perceptron and Beam Search. *Computational Linguistics*, vol.37, No.1: pp. 105-151 (2011)