社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-DOCMAS-B101-5

係り受け解析における状態書き換え規則のマイニング

Mining Rules to Rewrite States in a Dependency Parser

猪口 明博[1] *　山岡 歩[1] 鷲尾 隆[1]
松本 裕治[2] 浅原 正幸[2] 岩立 将和[2] 賀沢 秀人[3]
Akihiro Inokuchi[1] Ayumu Yamaoka[1] Takashi Washio[1]
Yuji Matsumoto[2] Masayuki Asahara[2] Masakazu Iwatate[2] and Hideto Kazawa[3]

[1] 大阪大学 産業科学研究所
[1] 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
[2] Graduate School of Information Science, Nara Institute of Science and Technology
[3] Google Inc.

Abstract: This paper presents a novel application of Graph Sequence Mining. 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. In this paper, we apply Graph Sequence Mining to a dependency parser based on transitions to mine rules to rewrite states in the parser.

1　はじめに

近年，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．AGM [7], gSpan [16]，Gaston [10] はラベル付きグラフの集合から閾値以上に頻繁に現れる頻出部分グラフパターンを列挙する手法である．上記のグラフマイニング手法は実用上非常に効率的であるが，グラフ系列のようなさらに複雑な構造をもつデータに対して適用することは困難である．

しかしながら，グラフ系列でモデル化するのが適している実世界の対象は多く存在する．例えば，ある時点での人間関係ネットワークは人が頂点，関係が辺であるグラフで表現できる．さらに，人がコミュニティ（ネットワーク）に参加，脱退することで頂点や辺が増減するため，その構造が変化するグラフはグラフの系列により表現できる．同様に，遺伝子が頂点，相互関係が辺である遺伝子ネットワークは進化の過程で遺伝子を新規獲得，欠落，突然変異するグラフの系列で表現できる．

近年，グラフ系列[1]から頻出パターンをマイニングが注目されはじめている [8, 6]．例えば，図 1(a) は 4 つの状態と 5 つの ID（頂点 ID）からなるグラフ系列を示している．我々は文献 [8] 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．これまでグラフ系列マイニングが適用された応用例は，引用ネットワーク [1]，HTML ネットワークなど文書を頂点，引用関係を辺とするネットワークや，論文共著ネットワーク [2]，ソーシャルネットワーク [2]，電子メールネットワーク [3] など人（ユーザ）を頂点，人間関係を辺とするネットワークであった．

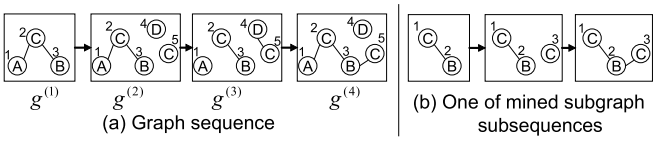


図 1: グラフ系列とマイニングされる部分グラフ系列

本稿では，グラフ系列マイニングの新たな応用として，自然言語処理における係り受け解析を対象とする．状態遷移系に基づく係り受け解析器の内部状態はグラフで表現でき，その状態遷移系列をグラフ系列データとして扱うことができる [12, 9]．本稿では，正解の係り受け構造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マイニングする手法を提案する．得られた書き換え規則は人間にとって可読であり，理解が容易であるという利点をもつ．また，書き換え規則を用いた提案手法

*連絡先：大阪大学 産業科学研究所
〒 567-0047 大阪府茨木市美穂ヶ丘 8-1
E-mail: inokuchi@ar.sanken.osaka-u.ac.jp

[1]dynamic graph [3] や evolving graph [2] とも呼ばれる．

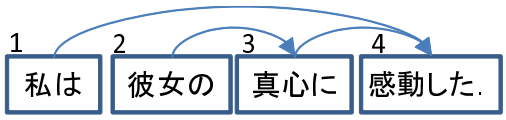


図 2: 係り受け構造の例

の計算量は，文節数 n に対して線形であり，従来の係り受け解析器と同等である．本稿では説明の簡単化のために日本語の係り受け解析を対象とし，さらにその手法として arc-standard, stack-based parser [11] を対象とするが，本稿で提案する原理は，状態遷移系に基づく係り受け解析器全般に適用可能である．

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 [12]．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[2]． ■

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，以下の定義を満たす projective な係り受け構造を対象とする．

定義 2 係り受け構造 $g = (V, A)$ が，全ての辺 $(i, j) \in A$ と頂点 $k \in V$ $(i < k < j)$ に対して，k から j に至る有向路が存在するときに限り，g を *projective* と呼ぶ． ■

文献 [12] の実験によると，日本語の単語係り受け解析において，94.7%の文が projective であり，今回取り扱う日本語の文節係り受け解析は取り扱うコーパスの定義より全ての文が projective である．

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．この図に示されるように，*projective* な係り受け構造の辺は交差なしに図示することができる．

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 3 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

[2]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．

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	
Left-arc	$(N, A) \Rightarrow (N, A \cup \{(i, j)\})$
	where $\{i, j\} = roots((N, A))$
Shift	$(N, A) \Rightarrow (N \cup \{\lvert N \rvert + 1\}, A)$
Preconditions	
Left-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

- 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．

- $c_1 = c_s(x)$,
- $c_m \in C_F$,
- $1 < i \leq m$ 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 4 $S = (C, T, c_s, C_F)$ 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 4 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 Left-arc か Shift の遷移のいずれかを選択する関数[3]であり，roots は森 (N, A) の根のう

[3]図 4 の遷移関数は文献 [11] において，arc-standard, stack-based parser と呼ばれるアルゴリズムの遷移関数である．本稿では，

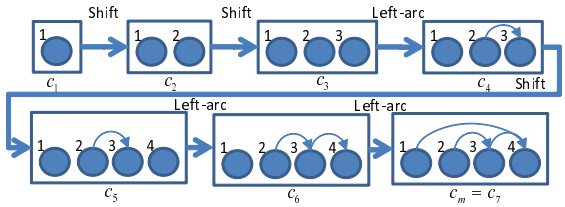


図 5: 状態遷移系列

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 により Left-arc が選択された場合，グラフ (N, A) に辺 (i, j) を加え，c から $t(c)$ に遷移する．一方，Shift が選択された場合，(N, A) に含まれない頂点の 1 つをグラフ (N, A) に加え，c から $t(c)$ に遷移する．

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 (Left-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を入力とし，Left-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 [12, 9] などを参照されたい．

本稿で対象とする係り受け解析器は，逐次的であるので，初期状態から終了状態に到達するまでに，Left-arc が $n-1$ 回，Shift が $n-1$ 回選択される．従って，関数 o による状態遷移選択の計算量の上限を θ とすると，ある文の係り受け構造を同定するための計算量は $O(n\theta)$ 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 [12]．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数の割合であり，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．

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*, *Left-arc*, *Shift*, *Left-arc*, *Left-arc* の順に遷移が選択される．紙面の都合上，この図では単語の表記を省略している．

3 状態遷移系に基づく係り受け解析器の課題

前節において，状態遷移系に基づく係り受け解析器の一例を示した．本節では，その解析器の課題について述べる．

係り受け解析器の内部状態とグラフ系列の関連の述べるにあたり，スタックを用いずに説明するため，その名称を用いないが，本稿での「状態遷移系に基づく係り受け解析器」は上記の parser を指している．

補題 1 係り受け解析器 $S = (C, T, c_s, C_F)$ の状態 C を節点，状態遷移を辺とする探索空間 T としたとき，図 *3* に示したアルゴリズムの探索空間 T は初期状態 c_1 を根，最終状態 $C_F \subseteq C$ を葉とする木で表される． ■

証明 1 ある状態 $c = (N, A)$ を考える．$v_{max} = max(N)$ とし，v_{max} を係り先とする辺が A に存在しないとき，c に至る状態遷移は，状態 $(N \backslash \{v_{max}\}, A)$ から *Shift* により状態遷移したときに限る．一方，状態 $c = (N, A)$ において，v_{max} を係り先とする辺が存在する場合，$v_{min} = min(\{v \in V \mid (v, v_{max}) \in A\})$ とすると，c に至る状態遷移は，状態 $(N, A \setminus \{(v_{min}, v_{max})\})$ から *Left-arc* により状態遷移したときに限る．従って，任意の c に対して，その直前の状態が一意に定まる．本稿で対象とする係り受け解析器は逐次的であるので，直前の状態を順に辿ることで，任意の c から c_1 へ至る路が一意に存在する．すなわち，探索空間 T は初期状態を根，最終状態を葉とする木で表される． □

状態によって構成される探索空間は木であるので，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．遷移を選択する関数 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．これを回避するための単純な方法は，バックトラック探索，確率的アルゴリズムを使う方法であるが，計算量が $O(n\theta)$ である利点が損なわれる．また，バックトラック探索などの探索法を取り入れると，各状態において尤度などの評価計算が必要となる．

そこで本稿では，計算量を n に対して線形であることを維持したまま，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マイニングする方法を提案する．そのために，次節では，グラフ系列を対象としたマイニング手法 GTRACE を紹介する．

4 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 [8] 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 $g^{(j+1)}$ の間の変化は変換規則系列 $\langle vi^{(j)}_{[1,A]}, ed^{(j)}_{[(2,3),\bullet]} \rangle$

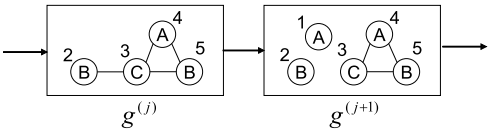


図 6: 連続する 2 つのグラフの変化

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．

ラベル付きグラフ g を $g = (V, E, L, f)$ で表す．ここで，$V \subseteq \{1, \cdots, n\}$ は頂点集合，$E \subseteq V \times V$ は辺集合，L は頂点と辺のラベル集合であり，$f : V \cup E \to L$ である．また観測グラフ系列を $d = \langle g^{(1)} \cdots g^{(z)} \rangle$ と表す．$g^{(j)}$ は j 番目に観測されたグラフである．

グラフ系列を簡潔に表現するため，グラフ系列中の連続する 2 つのグラフ $g^{(j)}$ と $g^{(j+1)}$ の差異に着目する．

定義 5 観測グラフ系列 $d = \langle g^{(1)} \cdots g^{(z)} \rangle$ の各グラフ $g^{(j)}$ を外部状態と呼ぶ．さらに，連続する *2* グラフ $g^{(j)}$ と $g^{(j+1)}$ の間を補間するグラフ系列を $d^{(j)} = \langle g^{(j,1)} \cdots g^{(j,m_j)} \rangle$ で表し，各 $g^{(j,k)}$ を内部状態と呼ぶ．ただし，$g^{(j,1)} = g^{(j)}$ かつ $g^{(j,m_j)} = g^{(j+1)}$ とする．グラフ系列 d は補間系列 $d = \langle d^{(1)} \cdots d^{(z-1)} \rangle$ で表される． ■

外部状態の順序は観測グラフ系列中のグラフの順序であるが，内部状態の順序は人工的に補間されたグラフの順序であり，$g^{(j)}$ と $g^{(j+1)}$ の間に様々な補間系列が考えられる．GTRACE は，グラフ系列マイニングの計算コストと空間コストを抑えるために，グラフ編集距離 [14] に基づき最短の補間系列を選択する．

定義 6 頂点や辺の追加，削除，ラベル変更を変換の最小単位とし，それらの変換を編集距離 *1* とする．本稿の内部状態系列 $d^{(j)} = \langle g^{(j,1)} \cdots g^{(j,m_j)} \rangle$ は，その連続する *2* つの内部状態の編集距離は *1* であり，かつ内部状態系列中の任意の *2* つの内部状態の編集距離は最小であるグラフ系列である． ■

本稿では，最小単位の変換を変換規則を用いて表す．

定義 7 $g^{(j,k)}$ を $g^{(j,k+1)}$ へ変換する変換規則を $tr^{(j,k)}_{[o_{jk},l_{jk}]}$ で表す．

- tr は頂点や辺の追加，削除，ラベル変更のいずれか
- o_{jk} は変換される頂点，あるいは辺
- l_{jk} は変換される頂点や辺のラベル ■

表 1: グラフ系列データのための変換規則

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

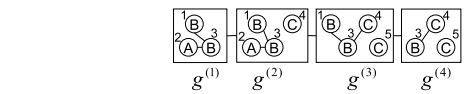
本稿では簡単化のため変換規則 $tr^{(j,k)}_{[o_{jk},l_{jk}]}$ を $tr^{(j,k)}_{[o,l]}$ と略記する．GTRACE が用いる 6 種の変換規則を表 1 に示す．以上より，変換系列を以下のように定義する．

定義 8 内部状態系列 $d^{(j)} = \langle g^{(j,1)} g^{(j,2)} \cdots g^{(j,m_j)} \rangle$ を変換規則を用いて $s^{(j)}_d = \langle tr^{(j,1)}_{[o,l]} tr^{(j,2)}_{[o,l]} \cdots tr^{(j,m_j-1)}_{[o,l]} \rangle$ と表し，内部状態変換系列と呼ぶ．さらに，外部状態系列 $d = \langle g^{(1)} \cdots g^{(z)} \rangle$ を内部状態変換系列の系列である外部状態変換系列 $s_d = \langle s^{(1)}_d s^{(2)}_d \cdots s^{(z-1)}_d \rangle$ で表す． ■

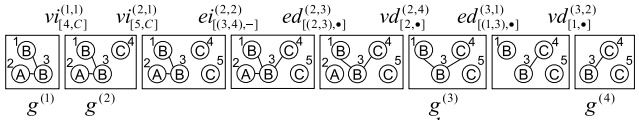
変換系列によるグラフ系列の表記は，グラフが徐々に変化するという仮定の下で，連続するグラフの差異のみに注目した表現形式であるので，グラフによる直接の系列表記に比べ簡潔である．また，如何なるグラフ系列も表 1 に示す 6 種の変換規則で表現可能である．

例 3 図 *7(a)* に示されるグラフ系列は図 *7(b)* に示される変換規則によって表され，変換系列 $\langle vi^{(1,1)}_{[4,C]} vi^{(2,1)}_{[5,C]} ei^{(2,2)}_{[(3,4),-]} ed^{(2,3)}_{[(2,3),\bullet]} vd^{(2,4)}_{[2,\bullet]} ed^{(3,1)}_{[(1,3),\bullet]} vd^{(3,2)}_{[1,\bullet]} \rangle$ で表される．ここで，“$-$” は辺ラベルを表している．

上記で説明した外部状態の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 の部分系列と定義し，s'_d に現れる頂点の ID を $ID(s'_d)$ と表し，$ID(s'_d)$ から $ID(s_d)$ への写像を ϕ とする．詳細な定義については，文献 [8] を参照されたい．



(a) An example of a graph sequence d.



(b) Representation of the graph sequence d by using TRs.

図 7: グラフ系列とその変換規則

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) と呼ぶ．関連研究同様，$s_{p1} \sqsubset s_{p2}$ ならば $\sigma(s_{p1}) \geq \sigma(s_{p2})$ である支持度の逆単調性が成り立つ．グラフ系列から FTS をマイニングする手法である GTRACE は，PrefixSpan [13] と同様に，FTS の末尾に変換規則を再帰的に 1 つずつ追加しながら，全ての FTS を列挙するアルゴリズムである．

5 書き換え規則の抽出

3 節で述べたように，図 3 に示し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抽出することを考える．

抽出したい書き換え規則は，ある誤った変換がグラフに与えられたとき，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 に存在する辺のうち，c_m に存在しない辺を追加する変換規則，
- c_m に存在する辺のうち，g に存在しない辺を削除する変換規則

である．前述の通り，抽出したい書き換え規則は，ある誤った変換がグラフに与えられたとき，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．従って，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 4 図 *8* に文 $x = \langle$ 私は, 彼女の, 真心に, 感動した. $\rangle$ 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．

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

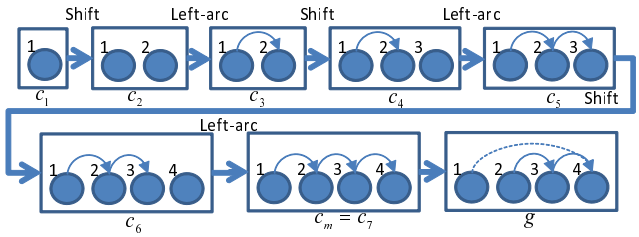


図 8: 状態遷移系列からグラフ系列の生成

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 9 *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，tr が適用される辺を返す関数を $lastEdge$ とし，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 5 書き換え規則 r の例を以下に示す．

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

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．さらに，$lastEdge(r) = (1, 4)$ である．

[4] 各頂点は文節を構成する単語，品詞などの情報によりラベル付けされるが，用いる素性選択法に依存するため 6 節で議論する．

[5] 我々は各状態が森であり，各頂点は高々1 つの親しか持たないという事前知識を持っている．従って，$ei^{(7,2)}_{[(1,4),l_2]}$ が系列 s に存在することは，ラベルが l_1 で，係り元が 1 である辺を削除する変換規則が s に必ず 1 つ存在することを示唆している．このため，我々は実装上，$ed^{(7,1)}_{[(1,2),l_1]}$ を s に含めていない．これより s の系列長を短くできるため，GTRACE の計算時間を削減することができる．

RuleMiner(D, $minsup$){
1) $R \leftarrow \emptyset$
2) $r \leftarrow null$
3) while
4) $TS \leftarrow \emptyset$
5) for sentence $(x = \langle w_1, w_2, \cdots, w_n \rangle, g) \in D$
6) $s \leftarrow$ ParseWithRules $(x, R \cup \{r\})$
7) $TS \leftarrow TS \cup \{CreateTransSeq(s \diamondsuit g)\}$
8) $evaluete(c_m, g)$, where $s = \langle c_1, c_2, \cdots, c_m \rangle$
9) if $R \neq \emptyset$, かつ正解率が改善しない
10) return R
11) if $r \neq null$
12) $R \leftarrow R \cup \{r\}$
13) $r \leftarrow MineAssociationRule(TS, minsup)$

ParseWithRules($x = \langle w_1, w_2, \cdots, w_n \rangle, R$)
1) $c \leftarrow c_s(x)$
2) $s \leftarrow ()$
3) while $c \notin C_F$
4) $c \leftarrow [o(c)](c)$
5) $s \leftarrow s \diamondsuit c$
6) $ts \leftarrow CreateTransSeq(s)$
7) for $(r = s_1 \Rightarrow s_2) \in R$
8) $(a, b) \leftarrow lastEdge(r)$
9) if $s_1 \sqsubseteq ts$, where $\phi : ID(s_1) \rightarrow ID(ts)$
10) $(i, j) \leftarrow (\phi(a), \phi(b))$
11) $c \leftarrow (N_c, A_c \cup \{(i, j)\})$
12) if $\exists j'$ s.t. $(i, j') \in A_c \wedge j' \neq j$
13) $c \leftarrow (N_c, A_c \setminus \{(i, j')\})$
14) $s \leftarrow s \diamondsuit c$
15) return s

図 9: 書き換え規則マイニングアルゴリズム

例 5 で与えられた書き換え規則 r があり，例 4 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．

図 9 の上段に書き換え規則 R をマイニングする手法を示す．D を文節区切りされた文 $x = \langle w_1, \cdots, w_n \rangle$ とその係り受け構造 g からなるコーパス $D = \{(x, g)\}$ とする．6 行目において，ParseWithRules は書き換え規則の集合 R を用いて，文 x を係り受け解析し，遷移系列 s を返す．次に 7 行目において，s の末尾に g を加え，その変換系列を CreateTransSeq によって生成し，TS に追加する．ここで $s \diamondsuit g$ は s の末尾に g を付加する操作である．続いて，8 行目において，最終状態 c_m と g を比較し，文正解率，係り受け正解率を更新する．9 行目では，$|R \cup \{r\}|$ 個の場合の正解率が，書き換え規則が $|R|$ 個のときに比べて改善しなければ，R を出力する．一方，正解率が改善するなら，r を R に追加し，13 行目で指定された最小支持度 $minsup$ のもとで，最大の確信度をもつ書き換え規則 r をマイニングする．

一方，図 9 の下段は，書き換え規則の集合 R を用いて，文 x を係り受け解析し，遷移系列 s を返す関数である．7 行目から 14 行目以外は，図 3 と同じである．この関数の 9 行目において，条件部が ts に含まれる書き換え規則 r が存在するなら，状態 c を 11 行目，あるいは 13 行目によって書き換え，14 行目において状態を遷移する．10 行目の辺 (i, j) は ts 内の辺 (a, b) に対応する s_1 内の辺である．11 行目は A_c に辺 (i, j) を追加する．また，13 行目では，i が親 j とは異なる親 j' を持つ場合は，A_c から辺 (i, j') を削除する．

続いて，ParseWithRules の計算量について，議論する．2 節で議論したように，3 行目のループ文の内部は $2n - 2$ 回繰り返される．また，7 行目のループ文の内部は，R 回繰り返される．さらに，グラフ系列が一般のグラフの系列であれば，$s_1 \sqsubseteq ts$ の計算問題は部分グラフ同型問題と同様に NP 完全であるが，グラフ系列は順序木の系列であるので $s_1 \sqsubseteq ts$ を解く計算量は，部分順序木同型問題 [5] と同様に頂点数 n に比例する．従って，ParseWithRules の計算量は $O(n(\theta + |R|n))$ である．しかし，s_1 と $CreateTransSeq(s)$ の計算において，その頂点の対応関係をメモリ上に記憶しておけば，s_1 と $CreateTransSeq(s \diamondsuit c)$ の計算は定数時間で計算可能である．すなわち ParseWithRules の計算量は $O(n(\theta + |R|))$ である．従って，計算量は文節数 n に対して線形であり，従来法と同等である．

6 評価実験

提案法を実装し，京大コーパス Version 4.0 を使用して評価実験を行った．具体的には，1 月 1 日から 1 月 8 日までの記事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 [4] 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 日までの記事のうち，1 日分の記事を除いて書き換え規則の集合 R を抽出する訓練データとし，除いた記事データをテストデータとした．これを 9 回繰り返した．なお，全ての評価実験は Intel Xeon X5570 (2.93GHz) × 2 の Windows Server 2008 上で行われた．

5 節で提案したアルゴリズムを効率化するために，以下に示すように提案手法をチューニングした．

グラフ系列の生成　グラフ系列の各頂点は 1 つのラベ

ルを持つのではなく，ラベル集合をもつように拡張した．図 10 は，$x = \langle$ 私は, 彼女の, 真心に, 感動した. $\rangle$ を入力としたときの CaboCha-0.60 の素性選択結果である[6]．ここで 1 番目の文節「私は」は {F_H0:私,F_H1:名詞,F_H2:代名詞,F_H3:一般,F_F0:は,F_F1:助詞,F_F2:係助詞,F_BOS:1} の 8 個の素性で特徴づけられている．そこで，この文節に対する頂点は 8 つの頂点ラベルをもち，この頂点の追加は $\langle vi^{(0,1)}_{[1,F\_H0:私]}, vi^{(0,2)}_{[1,F\_H1:名詞]}, vi^{(0,3)}_{[1,F\_H2:代名詞]}, vi^{(0,4)}_{[1,F\_H3:一般]}, vi^{(0,5)}_{[1,F\_F0:は]}, vi^{(0,6)}_{[1,F\_F1:助詞]}, vi^{(0,7)}_{[1,F\_F2:係助詞]}, vi^{(0,8)}_{[1,F\_BOS:1]} \rangle$ で表される．

GTRACE によりマイニングされる FTS の制約

1. 変換系列中における，辺の追加のための変換規則の最大数を 4 とした．辺の追加のための変換規則数が 4 より大きい FTS は，$vi^{(j,k)}_{[o,F\_H1:名詞]}$ や $vi^{(j',k')}_{[o',F\_F1:助詞]}$ など，どの文にも現れる素性のみで構成されることが多く，これらを排除するためである．
2. 関連のある FTS のみをマイニングする[7]．これにより，文節間の前後関係が不明な FTS を排除できる．
3. 変換系列 s における各々 $o \in ID(s)$ について，少なくとも 1 つの $vi^{(j,k)}_{[o,l]}$ が存在する FTS のみをマイニングする．これにより文節に関する情報のない FTS を排除できる．
4. ある FTS s は，あるグラフ系列の変換系列 s_d に複数回出現する．GTRACE は s が s_d に複数回出現しても，1 回とカウントするが，SiGTRACE [15] では出現回数だけカウントする．後者を重み付き支持度 $\sigma_w(s)$ と呼ぶと，$\mu = \sigma_w(s)/\sigma(s)$ は 1 以上であり，μ の値が 1 に近くないとき，書き換え規則により追加される辺の端点が一意に決まらない可能性がある．このために，$\mu \leq 1.25$ を満たす FTS のみをマイニングする．

条件 1 と 2 により GTRACE の計算時間を削減でき，条件 1〜4 により出力される FTS の数を削減できる．

書き換え規則の抽出に関する改良　MineAssociationRule により抽出される書き換え規則は 1 つではなく確信度の高い上位 100 個とし，正解率がもっとも改善する書き換え規則を選択した．

図 11 と図 12 に，提案手法により最小支持度 0.5%で抽出された書き換え規則の結論部に対するグラフ系列を示す．ここで，h, i, j, k は文節番号であり，$h < i < j < k$ である．図 11 の支持度，確信度は，それぞれ 0.58%，89.7%であり，図 12 の支持度，確信度は，それぞれ 0.56%，77.0%であった．また，図 11 と図 12 に示

[6]A,B,G タイプの素性を頂点ラベルとして使用しなかったため，この図ではそれらの素性を削除している．

[7]関連のある FTS の定義については，文献 [8] を参照されたい．

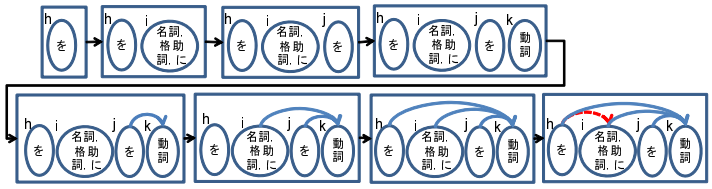


図 11: 抽出された書き換え規則 1

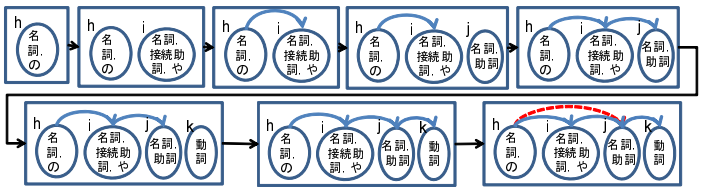


図 12: 抽出された書き換え規則 2

した書き換え規則により，書き換えられた例文を図 13 と図 14 に示す[8]．スラッシュ記号が文節の区切りであり，文中の h, i, j, k は，書き換え規則の h, i, j, k のそれぞれに対応する．さらに，1 列目に○と記された文は，書き換え規則により正しく状態が書き換えられた文であり，△は正しく状態が書き換えられなかった文，×は誤って状態が書き換えられた文である．例えば，従来法による $x = \langle$ 各行は, 今月, 下旬を, めどに, 結論を, 出す。$\rangle$ に対する係り受け構造では,「下旬を」,「めどに」,「結論を」の係り先は，全て「出す。」である．しかし，図 11 に示す書き換え規則により,「下旬を」の係り先を「めどに」に書き換え，正しい係り受け構造を得られたことを表している．1 つ目の書き換え規則は，k 番目の文節である動詞の目的語は j 番目の文節であり，h 番目の文節ではないことを示唆し，h 番目の係り先を適切に書き換える規則である．一方，2 番目の書き換え規則は i 番目の文節の名詞と j 番目の名詞が接続詞「や」によって接続された場合，従来法では h 番目の文節の係り先は i 番目の文節とする傾向があるが，h 番目の文節は j 番目の文節が適切であることを示唆している．この規則は必ずしも成り立つわけではないが，この規則の確信度は 77.0%であるので，成り立つ可能性は高い．

以上のように，提案手法により得られる書き換え規則は，人間にとって可読であり，理解が容易であるという利点をもつ．また，書き換え規則が人間の理解に対して妥当であった場合，その規則で状態が書き換えられたことにより誤った係り受け構造が得られる文は，人手によって作成されたコーパスデータが誤りの可能性がある．従ってコーパスの修正にも役に立つと考えられる．

図 15 と図 16 は，それぞれ係り受け正答率と文正答率を表している．図の横軸は書き換え規則の数 $|R|$ であり，図 9 の 3 行目のループ文を繰り返すたびに，規則

[8]紙面の都合上，文節数の少ない文を選択して掲載する．

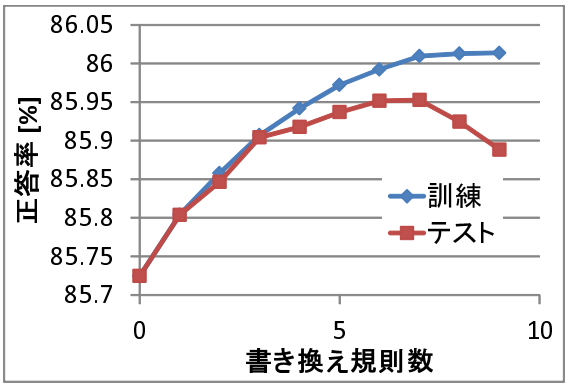


図 15: 係り受け正解率

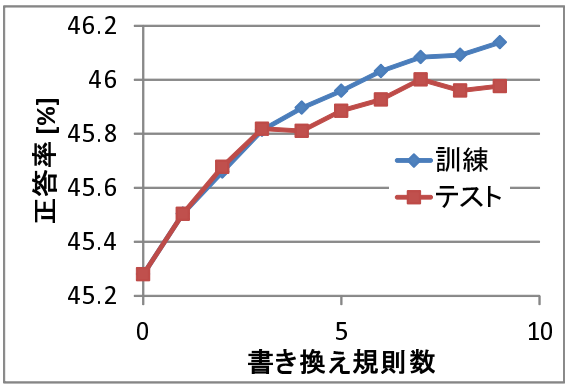


図 16: 文正解率

の数は 1 つ増える．$|R| = 0$ が，従来法の正答率であり，書き換え規則が増えると正答率は増加する．$|R| = 9$ で，係り受け正答率と文正答率は，それぞれ 0.16%，0.7%改善したが，得られた書き換え規則数が少ないため，正答率の十分な改善が得られたとは言えないが，得られた書き換え規則によりテストデータの正解率が改善していることから，得られた書き換え規則は妥当だと言える．

7 まとめ

本稿では，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マイニングする手法を提案した．得られた書き換え規則は人間にとって可読であり，理解が容易であるという利点をもつ．また，書き換え規則を用いた係り受け解析器 ParseWithRules の計算量は，文節数 n に対して線形であり，従来法と同等である．提案法の課題は以下の通りである．6 節で，マイニングされる FTS に様々な制約を課した．しかし制約を課してもなお，出力される FTS の数は 1.4 億パターンを超え，これらの FTS のマイニングための計算時間は 4 時間を超える．今後の課題は，書き換え規則を得るための計算時間を削減することである．

参考文献

[1] R. Ahmed and G. Karypis Algorithms for Mining the Evolution of Conserved Relational States in Dynamic Networks *Proc. of IEEE Int'l Conf. on Data Mining*,(2011), to appear.

[2] M. Berlingerio, F. Bonchi, B. Bringmann, and A. Gionis. Mining Graph Evolution Rules. *Proc. of European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pp. 115–130. (2009)

[3] K. M. Borgwardt, H. Kriegel, and P. Wackersreuther. Pattern Mining in Frequent Dynamic Subgraphs. *Proc. of IEEE Int'l Conf. on Data Mining*, pp. 818–822. (2006)

[4] CaboCha-0.60 `http://code.google.com/p/cabocha/`

[5] E. Makinen. On the Subtree Isomorphism Problem for Ordered Trees. *Information Processing Letter*, Vol. 32, No. 5, pp. 271–273. (1989)

[6] 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)

[7] A. Inokuchi, T. Washio, and H. Motoda. An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data. *Proc. of European Conf. on Principles of Data Mining and Knowledge Discovery*, pp. 13–23. (2000)

[8] 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)

[9] 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)

[10] S. Nijssen and J. N. Kok. A Quickstart in Frequent Structure Mining can Make a Difference. *Proc. of Int'l Conf. on Knowledge Discovery and Data Mining*, pp. 647–652. (2004)

[11] J. Nivre Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics*, Vol. 34, No. 4, pp. 513–553. (2008)

[12] S. Kubler, R. McDonald, and J. Nivre. Dependency Parsing. *Morgan and Claypool Publishers*, (2009)

[13] J. Pei, et. al. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth, *Proc. of Int'l Conf. on Data Engineering* (ICDE), pp. 2–6. (2001)

[14] 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)

[15] 山岡, 猪口, 鷲尾. 単一グラフ系列からの頻出パターン列挙. 人工知能学会 第 *25* 回全国大会 予稿集 1D1-2in. (2011)

[16] X. Yan and J. Han. gSpan: Graph-Based Substructure Pattern Mining. *Proc. of IEEE Int'l Conf. on Data Mining*, pp. 721–724. (2002)

```
* 0 -1D 0/1 0.0 F_H0:私,F_H1:名詞,F_H2:代名詞,F_H3:一般,F_F0:は,F_F1:助詞,F_F2:係助詞,F_BOS:1
私 名詞, 代名詞, 一般,*,*,*, 私, ワタシ, ワタシ O
は 助詞, 係助詞,*,*,*,*, は, ハ, ワ O
* 1 -1D 0/1 0.0 F_H0:彼女,F_H1:名詞,F_H2:代名詞,F_H3:一般,F_F0:の,F_F1:助詞,F_F2:連体化
彼女 名詞, 代名詞, 一般,*,*,*, 彼女, カノジョ, カノジョ O
の 助詞, 連体化,*,*,*,*, の, ノ, ノ O
* 2 -1D 0/1 0.0 F_H0:真心,F_H1:名詞,F_H2:一般,F_F0:に,F_F1:助詞,F_F2:格助詞,F_F3:一般
真心 名詞, 一般,*,*,*,*, 真心, マゴコロ, マゴコロ O
に 助詞, 格助詞, 一般,*,*,*, に, ニ, ニ O
* 3 -1D 1/2 0.0 F_H0:し,F_H1:動詞,F_H2:自立,F_H5:サ変・スル,F_H6:連用形,F_F0:た,F_F1:助動詞,F_F5:特殊・タ,F_F6:基本形,F_EOS:1
感動 名詞, サ変接続,*,*,*,*, 感動, カンドウ, カンドー O
し 動詞, 自立,*,*, サ変・スル, 連用形, する, シ, シ O
た 助動詞,*,*,*, 特殊・タ, 基本形, た, タ, タ O
.  記号, 句点,*,*,*,*,.  ,.  ,.  O
EOS
```

図 10: CaboCha-0.60 による素性選択

○	各行は/今月/下旬を (h)/めどに (i)/結論を (j)/出す。(k)
○	同課は/新宿を (h)/拠点に (i)/麻薬などを (j)/密売していたと (k)/みている。
○	苦手意識を (h)/きっかけに (i)/塾通いを (j)/始めた (k)/子供は/七割近く/いた。
○	中国は/「互恵平等」主義を (h)/盾に (i)/協定締結時から/平等を (j)/保っている。(k)
○	民間信用調査機関の/データを (h)/もとに (i)/全国/約四十八万社を (j)/分析した。(k)
○	「戦後五十周年」を (h)/節目に、(i)/戦後処理問題に/引き続き/取り組む/考えを (j)/示す。(k)
○	保守党が/これを (h)/機に、(i)/古い/体質を/抱えた/労働党攻撃を (j)/行う (k)/ことは/確実だ。
○	この/ため/農水省は/今月、/全国の/獣医師を (h)/対象に (i)/薬管理の/実態調査を (j)/実施する。(k)
○	所有者と/担保設定手続きを/代行した/司法書士を (h)/相手に (i)/損害賠償請求訴訟を (j)/起こした。(k)
○	円天井は/深い/青を (h)/地に (i)/金色の/石片を (j)/ちりばめ、(k)/それこそ/満天の/星空を/思わせる。
○	同省は/調査結果を (h)/もとに (i)/今秋にも/監視体制の/強化など/対策を (j)/打ち出す (k)/ことに/している。
○	反戦機運の/高まりを (h)/背景に (i)/政権への/揺さぶりを (j)/かけようとした (k)/改革派の/狙いは/失敗した。
○	都市銀行などの/金融機関は、/兵庫県、/大阪府を (h)/中心に (i)/支店の/業務が/大きく/影響を (j)/受けている。(k)
△	「成人の/日」に/あたり、/詩人に/「明」を、(h)/精神医学者に (i)/「暗」を (j)/語ってもらった。(k)
△	小沢氏は/新進党に/ついて、/その/性格を (h)/「生活者に (i)/重きを (j)/置き、(k)/都市に/住む/人の/気持ちを/しっかり/つかんだ/政党だ」と/解説。

図 13: 書き換え規則 1 が適用された文

○	昨年の/新進党結成の (h)/背景や (i)/目的などを (j)/説明した。(k)
○	新幹線の (h)/新型車両や (i)/試験車両が (j)/相次いで (k)/登場している。
○	１５日までの/日程で、/各国の (h)/蔵相や (i)/首脳クラスと (j)/会談する。(k)
○	サンクリストバルの (h)/陸軍基地や (i)/市庁舎などを (j)/次々に/攻撃した。(k)
○	鳥取や/島根の (h)/日本海側や (i)/山間部では、(j)/雪が/断続的に/降り続いた。(k)
○	同県我孫子市内の (h)/井手口幹事長後援会事務所や (i)/自宅などを (j)/家宅捜索した。(k)
○	「一年の (h)/夢や (i)/希望が (j)/天に/届いて、(k)/かなうように」と/学校側が/考案。
○	そんなこんなで/「ええい、/茶髪の (h)/１つや (i)/２つ！」と (j)/意気込む (k)/私……。
○	武器輸出の (h)/規制強化や (i)/軍縮を (j)/やりつつ (k)/国連平和維持活動を/やるべきだ。
○	関西では/一部の (h)/信用金庫や (i)/信用組合が (j)/取り扱っているが、(k)/銀行では/初めて。
○	我孫子市に/本社が/あり、/ゴルフ練習場の (h)/経営や (i)/保険代理業などを (j)/営んでいる。(k)
○	付録では、/事故が/起きた/場合の/応急措置の (h)/方法や (i)/手配に (j)/ついて (k)/記載されている。
×	八〇年に/弁護士登録し、/リッカーの (h)/商法違反事件や (i)/平和相互銀行事件などを (j)/担当した。(k)
×	一方、/分党案は/久保氏の (h)/周辺や、(i)/山花氏の/新党構想に/賛同している/党幹部らが (j)/検討している。(k)
×	だが、/食料や/飲料水が/欠乏し、/市民は/道路の (h)/穴や (i)/溝に (j)/ある (k)/水を/集めて/飲んでいると/いう。

図 14: 書き換え規則 2 が適用された文