選言的木パターンマイニング

清水　一宏†　　三浦　孝夫†

† 法政大学 工学部 情報電気電子工学科　〒 184–8584 東京都小金井市梶野町 3-7-2
E-mail: †{c02d3051,miurat}@k.hosei.ac.jp

あらまし　頻出な選言パターンは，単一系列データ上で APRIORI に基づいた逆単調性を満たす効率的なアルゴリズムを論じることができ，精巧なテキストマイニング技法として知られている．本稿では，この選言パターンの考え方を木パターンに拡張し，半構造データ上で逆単調性を満たす出現尺度を用いることで，高速な選言的木パターンマイニング手法を提案する．また実験によりその有用性を検証する．

キーワード　選言的木パターン，半構造データ, 逆単調性

Mining Disjunctive Tree Patterns

Kazuhiro SHIMIZU† and Takao MIURA†

† Dept.of Elect.& Elect. Engr., HOSEI University　3-7-2, KajinoCho, Koganei, Tokyo, 184–8584 Japan
E-mail: †{c02d3051,miurat}@k.hosei.ac.jp

Abstract　Frequent disjunctive patterns is known to be a sophisticated method of text mining in a single document that satisfies *anti-monotonicity*, by which we can discuss efficient algorithm based on APRIORI. In this work, we propose a data construction method for practical mining of frequent *disjuctive tree patterns* based on *Online analysis approach*. And we discuss some experimental results.

Key words　Disjunctive Tree Pattern, Semi-Structured data, Anti-Monotonicity

1. 前　書　き

これまで，文書データから重要な語句を抽出するために，統計的推定やデータマイニング手法などの定量的な分析を適用することは容易ではないと言われてきた [9], [10]．しかし，テキストを対象とした研究が注目されるにつれ，この手法を適用する研究が開始されている [8], [18].

データマイニング手法の特徴は，頻度や共起性にある [22], [23]．この手法を，系列データ (sequence data)，特に文字列テキストに適用するアプローチをテキストマイニングと呼ぶ．系列データを扱う問題には，単一系列と複数系列を扱うものがあるが，既存のデータマイニング手法の多くは後者を対象としており，個々のパターン出現の探索による空間量の増大を，パターンの出現系列数を重要度と定義することで抑える．しかし，各パターンの出現頻度を調べることは，データの偏りや分布，TF*IDF といった重要度の算出や検索に有効である．複数の単一系列データを分析することは，各系列でのパターン頻度分析を多重化するのと同時に，複数系列の系列毎の頻度分析を行うことになり，より精密な解析が可能となる．従って，これまでに提案されている手法は適用できない．

APRIORI などデータマイニングの主要な研究の多くでは，探索空間の削減にパターンの逆単調性が用いられているが，系列パターンでは成り立たない [2], [7]．また，APRIORI を用いたテキストマイニング操作は組み合わせ探索問題であり，一般的に多くの計算資源を要するため，実際には小規模な問題やサンプリング手法などによりこれを回避する [15]．著者らは，**選言** (disjunctive) パターンを導入し，この上で逆単調性をみたす出現尺度を提案した [13].

一般的に利用者が興味あるパターンを発見する場合，検出結果を見ながら設定条件を変化させるというプロセスを繰り返すことが多い．各ステップは効率よく処理できても，全体としてデータを繰り返し走査するため，膨大な読込みが生じ，最終結果を得るまでに多くの時間を要する．また，各処理が独立であるため，設定条件を変更するたびに再計算が必要となる．これらの問題を回避するため，オンライン分析手法が提案されている [1]．また筆者らは, [13] に対するオンライン分析を導入し，効率的な選言パターン抽出法を提案した [20].

本稿では，半構造データ (semi structured data) を単一系列データの集合と見ることで，この単一系列データの集合を同時実行的に効率よく解析することを考える．複数系列データとは違い，パターンの頻度計算量が膨大となる単一系列データでは，処理を同時実行することで大幅な計算量削減が期待できる．ここでは単一系列データの構文的特長を利用し，(トライ構造などのような) 先頭から一致する複数の系列を木構造として捉え，

重複計算を回避する方式を論じる．また，各単一系列データをニュース記事などのように互いに独立であると考え，タグ値とテキスト値を同等に木の経路上に配置する．従って解析されるパターンは深さ方向にのみ展開されるため，構造抽出に特化した木パターンマイニング手法 [18] などは単純には適用できない．

ここで提案する解析方式の特徴は，木の探索時に子孫節点が上位節の情報を利用し解析結果を継承する点にある．実際には，上位節における各パターンの出現位置と頻度を継承し，下位節での頻度計算に利用することで上位節の重複した計算を避ける．このようにして得られた解析結果を特定の形式でデータを表現し，オンライン分析の考え方を用いて，繰り返して生じる問い合わせに対し，効率よく選言的木パターンを抽出する方法を提案する．2 章では選言的木パターンとその逆単調性を満たす出現尺度を示し，3 章でデータの構築法及びそのデータからの選言的木パターン抽出法を提案する．4 章でいくつかの実験結果を示す．5 章で関連研究をまとめ，6 章で結びとする．

2. 選言的木パターン (Disjunctive Tree Pattern)

与えられたアルファベット (あるいはアイテムとも言う) の集合 $I = \{i_1, .., i_L\}, L > 0$ に対し，**系列データ** S とはアルファベットの順序リスト $s_1...s_m, m > 0$ である．S には同じアルファベットが複数回生じてよく，S に含まれる (重複を含めた) 語の数 m に対し，S を m-系列データという．

ここで，S の i 番目に出現するアルファベットが v であるとき，節 i でこれをラベルにもつ有向グラフを考える．例えば図 1(a) に $S = a_1b_2c_3$ に対応する有向グラフを示す．2 つの系列 $S = a_1a_2...a_n$, $T = b_1b_2...b_m$ に対してそのアルファベットが先頭から第 i 番目まで一致するとき ($i \geqq 0$)，先頭からの長さ i の経路が一致する有向グラフに対応させる．i 番目で初めて異なれば位置 i で分岐し，$i = 0$ であれば非連結となる．一般に，系列データの集合 $G = \{S_1...S_n\}, n > 0$ があたえられたとき，2 つの系列 S_i, S_j に対してアルファベットが先頭から第 i 番目まで一致するとき，先頭からの長さ i の経路を共有する森グラフ (木構造グラフの集合) に対応させれば，すべての系列を経路と対応させることができる．例えば $G = \{aca, abc, abb, aa\}$ であるとき図 1(b) G を得る。なお $|G| = n$ とする．

選言的木パターン (あるいは単にパターン) p とは $t_1t_2...t_n$ で表され，各 t_i はアルファベット a または $[a_1a_2...a_m], m > 0$ (各 a_j は相異なるアルファベット) の形である．この $[a_1a_2...a_m]$ を選択と呼ぶ．パターン $p = t_1t_2...t_n$，$q = v_1v_2...v_m, m \leqq n$ があるとき，q が p の**部分パターン** である ($q \sqsubseteq p$ と記す) とは，$1 \leqq j_1 < ... < j_m \leqq n$ があり，各 v_k は t_{j_k} に対応して次を満たす (混乱の無い限り $v_k \sqsubseteq t_{j_k}$ と記す)：

> v_k がアルファベット a のとき， $t_{j_k} = a$ もしくは a を含む選択
> v_k が 選択 $[a_1a_2...a_m]$ のとき，$t_{j_k} = [b_1b_2...b_l]$ でかつ $\{a_1, .., a_m\} \subseteqq \{b_1, .., b_l\}$

［例 1］ “ac” は “abcd” の部分パターンである．同様に，“[ac]” は “[abcd]” の，“bd” は “[ab]b[cd]de” の，“b” は “[ab]” の，そして “ac”は “[ab][cd]” の部分パターンである．

しかし，“ab” は “[ab]” の部分パターンでも，“[ab]” は “ab” の部分パターンでもない．実際，選択 “[ab]” は選択でない “ab” と一致せず，“a” は “ [ab]”に生じるが “ b” が対応しない．□

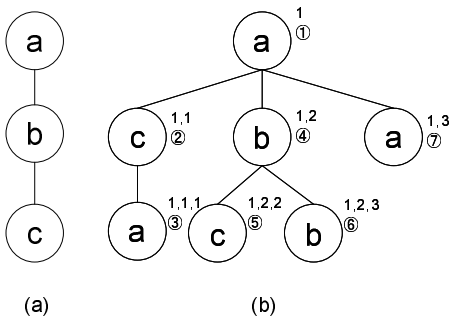

図 1　(a) 系列データの例　(b) 半構造データの例

系列データ $S = c_1c_2...c_m$ にパターン $p = t_1t_2...t_n$ が**適合する** (match) とは，t_1 がアルファベット a_1 のとき，$t_1 = a_1 = c_{i_1}, 1 \leqq i_1 \leqq m$ でありかつ 部分パターン $t_2...t_n$ が系列データ $c_{i_1+1}...c_m$ に適合するときを言う．t_1 が選択 $[a_1a_2...a_m]$ のとき $a_1, ..., a_m$ のある並びからなるパターン $a_{j_1}...a_{j_m}$ が系列データ $c_1...c_{i_1}$ に適合し，かつ 部分パターン $t_2...t_n$ が系列データ $c_{i_1+1}...c_m$ に適合するときを言う．

［例 2］ 系列データ S が $\mathtt{a_1a_2b_3b_4b_5a_6}$(各アルファベットの添え字はS中における出現位置) であるとき，パターン a は Sに 3 回適合 ($\mathtt{a_1, a_2, a_6}$) する．また，パターン ab は 6 回適合 ($\mathtt{a_1b_3, a_1b_4, a_1b_5, a_2b_3, a_2b_4, a_2b_5}$) する．一方 [ab] は 9 回適合する.□

半構造データ G に関してパターンから非負整数への関数 $\mathcal{M}$ が**逆単調性** (Anti Monotonicity, AM) を満たすとは，パターン p, q に対して $q \sqsubseteq p$ のとき $\mathcal{M}_G(q) \geqq \mathcal{M}_G(p)$ が成り立つときを言う．以下で考慮する半構造データは単一であり G を略す．

逆単調な $\mathcal{M}$ が与えられ，また最小支持度 m が与えられているとする．このとき，$\mathcal{M}(q) < m$ ならば $q \sqsubseteq p$ となる p は $\mathcal{M}(p) \geqq m$ ではない．この性質を用いれば，部分パターンの探索範囲を減少させることができる．整数パラメタ m は探索範囲を表すとも見れるため，特に最小支持度 (minimum support) と呼ぶ．

$\mathcal{M}$ は出現頻度を表すことが多い．例えば APRIORI ではパターンが生じるトランザクションレコード数であると定義される [2]．しかし，本稿で論じるような半構造データには “トランザクションレコード”のような明確な区別が無いため，出現頻度の定義自体を論じる必要がある．複数の半構造 (系列) データを扱う場合，出現頻度 (適合頻度) は “当該パターンを含むデータ”の数であると定義されることが多い．この場合，逆単調性は自明に成り立つ．しかし，単一な半構造 (系列) データの場

合，例 2 のように単純な適合頻度では逆単調性が得られず，効率よい探索を期待することができない．これが本稿で逆単調性を有する出現尺度を提案する理由である [13].

系列データ $S = s_1 s_2 ... s_r$，パターン p を $t_1 t_2 ... t_n$ とすると系列先頭頻度は以下のように表される．

$$H(S,p) = \sum_{i=1}^{r} Val(S,i,p)$$

ただし $Val(S,i,p)$ は次を満たすとき 1, さもなければ 0 であるとする:

> $S(i)$ を S の i 番目からの接尾辞 $s_i ... s_r$ とする．t_1 がアルファベット a のとき，$s_i = a$ かつ $t_2 t_3 ... t_n$ が $S(i+1)$ に適合する．t_1 が選択 $[a_1 a_2 ... a_m]$ のとき，$s_i = a_j$ となる j があり (例えば $j = 1$)，$[a_2 a_3 ... a_m] t_2 ... t_n$ が $S(i+1)$ に適合する．

$H(S,p)$ は S またはその接尾辞において p が先頭から適合する回数を表している．また，G に対する系列先頭頻度 $H(G,p)$ は以下のように表される．

$$H(G,p) = \sum^{|G|} H(S,p)$$

しかし，系列先頭頻度は逆単調性を満たさない．そこで， p のすべての部分パターン q を調べ，その $H(S,q)$ のうちの最小の値を全体出現頻度とする．

$$D(S,p) = \mathrm{MIN}\{H(S,q) | q \sqsubseteq p\}$$

また，この計算は p の接尾辞 (n 個存在) のうち，長さの小さいものから順にその最小値を決定するとすれば以下のように書き直すことができる．系列データ S, パターンを p とすると，

$$D(S,p) = \mathrm{MIN}\{H(S,p), D(S,p(2))\}$$

ただし，$p(2)$ は p より 1 つ長さの短い部分パターンである．このとき，$D(S,p)$ は逆単調性を示すことが知られている [13]. また，G に対する全体出現頻度 $D(G,p)$ は以下のように表される．

$$D(G,p) = \mathrm{MIN}\{H(G,p), D(G,p(2))\}$$

[例 3] S を `caabbbc`, $p = ab$ とすれば $H(S,p) = 2$, $D(S,p) = 2$ である．$p = ac$ とすれば $H(S,p) = 2$, $D(S,p) = 2$ である．また $p = [ac]$ とすれば $H(S,p) = 3$, $D(S,p) = 2$ である (実際の適合頻度は 4 回)．ac や $[ac]$ の部分系列 a,c が分離して生じていても適合するとみなすため，系列先頭頻度や適合回数とは合わない．
□

3. 選言的木パターン束

以下，本稿ではしきい値 m 以上の全体出現頻度を持つパターンを**頻出パターン** (frequent pattern) という．選言的木パターンに対するオンライン分析を行うため，アイテム集合 I，単一半構造データ G に対して，図 2 のような，あるしきい値を持つアイテムに関するべき集合 2^I 上の**束** (lattice) を構築する．この束を**選言的木パターン束**（Disjunctive Tree Pattern Lattice, 以下 DTPL）と呼ぶ．DTPL とは，閉路の無いラベル付き根付き有向グラフ (V,E) であり，V は節点の有限集合, E は有向辺の集合 ($E \subseteqq V \times V$) を表す．DTPL にはただ一つの **根節** (root) が含まれ，ラベルを有さない．根からの距離が n の**節点** $v \in V$ は 要素数 n の頻出パターン (n-頻出パターン) を表し，$D(G,v)$ を**ラベル**として持つ．$(n+1)$-頻出パターン v' が v を部分パターンにもつとき，**有向辺** $(v,v') \in E$ が存在する．
[例 4] G を 図 1(b)，しきい値を $m = 1$ とした時，DTPL は図 2 のように構築される．
□

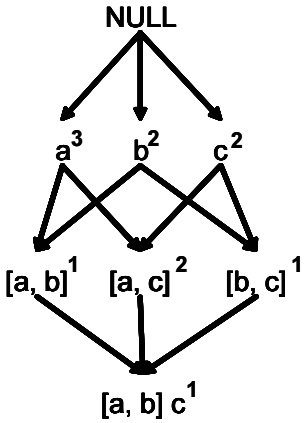

図 2　$m = 1$ で構築された DTPL の例

3.1 半構造データからの DTPL 構築

半構造データ G から DTPL を構築するため，頻出パターン p の G 中での前置順序で表される出現開始位置 $head$ と終了位置 $tail$，同一親での兄弟順序を表す深さ符号 d の順序リストである**深さ符号列** $depth = \{d_1, ..., d_n\}$ をそれぞれ保持した**出現位置リスト**を定義する．深さ n は G の根 r の深さを 1 とした時，r から $tail$ までの経路上に存在するアイテム数である．例えば，図 1(b) において，図中の数字列は深さ符号列を，丸囲い数字は前置順序による出現位置をそれぞれ表している．p の G 中での出現位置を $P_p(head, tail, depth)$ とするとき，p に関する出現位置リスト $list_p$ を次のように定義する．

$$list_p = \{P_{p_i}(head, tail, depth), ..., P_{p_{max}}(head, tail, depth)\}$$

ここで max は系列先頭頻度を表す．与えられた頻出規準値 (しきい値 m) に対して，出現位置リストを用いて，半構造データ G から DTPL を構築する手順を以下に示す．

> 入力:半構造データ G
> 出力: DTPL L
> （1） G をプリオーダー順に 1 回走査し，全ての 1-頻出パターン p の出現位置リスト $list_p$ を生成し，L に節点と辺として追加する．n を 1 とする．

（2） n-頻出パターンの出現位置リストから，順次 $(n+1)$-頻出パターン p' とその出現位置リスト $list_{p'}$ を取得し，L に節点と辺として追加する．n を $n+1$ とする．

（3） 頻出パターンが生成できなくなるまで (2) を繰り返し，最後に L を出力する．

上述の手順 (2) で，n-頻出パターン p,q の出現位置リスト $list_p$, $list_q$ から $(n+1)$-頻出パターン pq の出現位置リスト $list_{pq}$ を得る手順を精密に示す．

入力:出現位置リスト $list_p$, $list_q$
出力:出現位置リスト $list_{pq}$

（1） $list_p$, $list_q$ を突き合わせ，$P_{p_i}(tail) < P_{q_j}(head)$ かつ $P_{p_i}(depth) \subseteqq P_{q_j}(depth)$ を満たす $P_{p_i}(head)$, $P_{q_j}(tail)$, $P_{q_j}(depth)$ をすべて見つけ，それぞれを P_{pq_k} の $head, tail, depth$ として $list_{pq}$ に追加する．

（2） $list_{pq}$ の要素数を $kmax$ とする．すべての出現位置を追加し終わったとき，$kmax \geqq m$ ならば $list_{pq}$ を出力する．

$$list_p = \{P_{p_1}(head, tail, depth), ..., P_{p_{imax}}(head, tail, depth)\}$$

$$list_q = \{P_{p_1}(head, tail, depth), ..., P_{p_{jmax}}(head, tail, depth)\}$$

$$list_{pq} = \{P_{pq_1}(P_{p_i}(head), P_{q_j}(tail), P_{q_j}(depth))) \\ |P_{p_i}(tail) < P_{q_j}(head), P_{p_i}(depth)) \subseteqq P_{q_j}(depth))\}$$

最大パターンサイズを N とするとき，上述の手順では G の走査は N に関係せず，G から 1-頻出アイテムの出現位置リストを生成する時に 1 回だけ必要となる．

［例 5］ 図 1(b) におけるパターン a,b,c の出現位置リスト $list_a$, $list_b$, $list_c$ はそれぞれ，

$$list_a = \{P_{a_1}(1,1,\{1\}), P_{a_2}(3,3,\{1,1,1\}), P_{a_3}(7,7,\{1,3\})\}$$

$$list_b = \{P_{b_1}(4,4,\{1,2\}), P_{b_2}(6,6,\{1,2,3\})\}$$

$$list_c = \{P_{c_1}(2,2,\{1,1\}), P_{c_2}(5,5,\{1,2,2\})\}$$

またパターン bc と $[ac]$ の出現位置リスト $list_{bc}$, $list_{[ac]}$ は，

$$list_{bc} = \{P_{bc_1}(4,5,\{1,2,2\})\}$$

$$list_{[ac]} = \{P_{[ac]_1}(1,2,\{1,1\}), P_{[ac]_2}(1,5,\{1,2,2\}), \\ P_{[ac]_3}(2,3,\{1,1,1\})\}$$

となる．□

3.2 DTPL からのパターン抽出

構築された DTPL からのパターン抽出は，**深さ優先探索** (*Depth-First Search*) で行う．以下にその手順を示す．

入力:DTPL L, 最小支持度 m
出力:出力リスト $\mathcal{L}$

（1） NULL 節点から開始して，m 以上の度数を満たす 1-頻出パターン節点を 1 つ選択し，その節点をスタック $\mathcal{L}$ へ出力する．

（2） その節点の持つ 1 辺を選択し辿る．

（3） 次の節点が，過去に訪れたことがなく，かつ m 以上の度数を有すれば，節点を $\mathcal{L}$ へ出力し (2) へ戻る．

（4） 過去に訪れていれば，1 つ前の節点に戻り (2) から開始する．

（5） すべての訪問可能な節点の探索が終わったら $\mathcal{L}$ を出力する．

［例 6］ 図 2 において，$m = 2$ の最小支持度を有するすべてのパターンを抽出する．この探索は図 3 のような順路に従い，出力リスト $\mathcal{L}$ を得る．図中の丸囲い数字は，探索の順序を示す．□

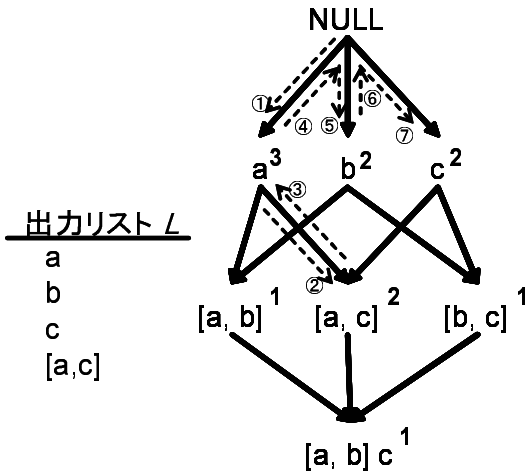

図 3 DTPL からの頻出パターン抽出

4. 実　　験

4.1 実験の方法

本稿では，2 種類の実験を行う．実験 1 では，DTPL を用いたオンライン分析手法と，毎回しきい値を指定し出現位置リストを利用する手法を用いて，選言的木パターン抽出に要する実行時間の比較を行う．実験 2 では，DTPL 構築に要する負荷を調べるため，使用する実験データのサイズを変えて，構築にかかる実行時間の比較を行う．

なお，以下の実験で生成される候補パターンは，(a, [ab], [ab]c, [abc]d 等のように) 選言の深さを 1 レベルに限り，また，1 つのパターン中に選択が 1 回だけ出現するものに限定する．

実験データとして，オンライン論文目録データベース $DBLP$ で公開されている XML データ [5] を用いる．全データサイズは $296MB$ であるが，その中から 500 件分のデータを用い，各記事について不要語 (*stop word*) の削除および単語のステミング [19] を行ったものを実験データとして使用する．

今回の実験では，タグ値の他にテキスト値 ($value$) もアイテムとして扱う．両者の扱いは基本的に変わらないが，テキスト値の深さは直前のタグ値によって決められるものとする．また，タグ値内の属性情報については，予め除去する．以下に実験データの一部を示す．

$\langle dblp \rangle$
 $\langle articl \rangle$
 $\langle author \rangle$ $abraham\ kandel$ $\langle /author \rangle$
 $\langle author \rangle$ $mordechai\ schneider$ $\langle /author \rangle$
 $\langle titl \rangle$ $fuzzi\ set\ applic\ artifici\ intellig$ $\langle /titl \rangle$
…
 $\langle /articl \rangle$
…
$\langle /dblp \rangle$

4.2 実験結果

実験 1 では，頻度しきい値 m を 5 から 50 まで変化させ，各手法でのパターン抽出にかかる実行時間の比較を行う．しきい値とそれに対する抽出パターン数を表 1 に，実行時間を表 2 に示す．表 2 は，パターン数と実行時間の関係を示す．実際に抽出した選言的木パターンの一部を以下に示す．

```
[<articl>, <journal>]advanc
[<titl>, softwar]engin
[<journal>, advanc]comput
```

実験 2 では，生成されるパターン数を固定し，データ長のみを変えて DTPL 構築にかかる実行時間の比較を行う．ここでデータ 1 は実験データ全体，データ 2 はデータ 1 を 2 つ連結したもの (従って 2 倍の長さになる)，同様にデータ 3 はデータ 1 を 3 つ連結したものである．実験 2 の結果を表 3 に示す．この表では，各データと実行時間の関係を示す．

しきい値	パターン数
5	1266
10	394
15	223
20	169
25	123
30	97
50	81

表 1　しきい値とパターン数の関係

しきい値	実行時間[s]	
	オンライン分析	しきい値毎回指定
5	0.93	833
10	0.90	293
15	0.73	203
20	0.63	177
25	0.60	150
30	0.57	138
50	0.55	128

表 2　各手法における選言パターン抽出の実行時間比較

データサイズ	実行時間[s]
データ1	293
データ2	726
データ3	1170

表 3　各データサイズにおける DTPL 構築の実行時間比較

4.3 考　　察

実験 1 のパターン抽出において，いったん DTPL を構築すれば，これを用いて選言的木パターン抽出を閉路無し有向グラフの探索問題に帰着させることができる．逆単調性を有する頻度定義であることは，いったん対象外であると判明したノードから先はたどらなくて良いことに対応し，探索範囲が削減できることを意味する．さらに，選言的木パターンが含まれれば組み合わせ的に探索空間が広がる．しかし，本方式で提案するように，選言的木パターンをそのまま残した DTPL を構築することで空間の拡大を抑え，空間自体の走査を 1 回だけ実施することで，効率よく探索することが可能になる．実際，本実験結果からもわかるように，毎回しきい値を指定して抽出する方法に比べおよそ 230 倍〜900 倍の高い性能を得る．

しきい値（最小支持度）が低いほど探索空間は増加する．しかし，上記 2 つの違いは，重複した計算部分の肥大化となって表れるため，本方式との差が拡大する原因となる．実際，本実験結果ではその割合はしきい値 10 倍に対して DTPL を用いたオンライン分析を利用した方法でおよそ 1.7 倍なのに対し，オンライン分析を行わず，毎回しきい値を指定して抽出する方法ではおよそ 6.5 倍に低下する．

実験 2 では DTPL 構築の負荷を調べている．データサイズが 2 倍増加に対して実行時間はおよそ 2.5 倍，データサイズが 3 倍増加に対して実行時間はおよそ 4.0 倍となっており，データサイズの増加に対して，実行時間の急激な増加は見られない．これは，出現位置リストを用いた DTPL 構築が実験データの局地的走査のみの計算で構築できることによるからだと考えられる．

5. 関連研究

半構造データ上で頻出順序木を高速に発見するアプローチとしては，TreeMiner [17] や FREQT [4]，左右木結合を利用した手法 [24] などがある．また頻出無順序木に対しては UNOT [25] や WTIMiner [6] などがある．しかし，いずれの手法も複雑な頻出木パターンの発見を目的とし，計算効率が悪い．一方，本研究で扱う選言的木パターンは，各データが独立しており，深さ方向にのみ展開されるという性質上，深さ方向へのパターン生成にのみ注目すれば良いため，その分効率良く頻出パターンが発見可能である．またこれらの研究では，構造の抽出に特化しているため，タグ値のみを対象としているものが多いが，本

手法では，テキストマイニング手法である [13] を拡張しているため，タグ値とテキスト値を同時に扱うことができ，より多くの情報をパターンとして抽出することが期待できる．

単一系列データ上で頻出パターンを発見するアプローチとしては，これまで事象列に関するエピソード手法 [11] が知られている．またテキストマイニング問題として [21] があるが，いずれも半構造データマイニングを前提には論じられていない．

本稿で扱うような，オンライン分析の考え方を利用した高速化手法として，隣接束 (Adjacency Lattice) が知られている [1]．隣接束とは，しきい値以上の頻度を有する頻出アイテムの集合から構成されており，この集合に対する問い合わせにより，最小支持度を満たす頻出アイテム集合を高速に得ることが可能である．しかし，相関ルールのオンライン生成が前提となっており，本研究のような選言的木パターンは扱わない．

また，束を用いた形式化によるパターン列挙法として SPADE が知られている [16]．この手法は複数系列データを対象としているため，本稿のような半構造データには直接利用できない．SPADE はすべての頻出パターンを列挙するのに，1-頻出パターン，2-頻出パターン，それ以上の頻出パターンの各抽出ステップを要し，それぞれでデータ走査を行う．一方，本提案手法では DTPL 構築に要するデータ走査は，先に述べた通り 1-頻出パターンの出現位置リストを生成する際の 1 回のみで良い．

6. 結　　論

本稿ではオンライン分析の考え方を用い，半構造データ上からの高速な DTPL 構築法および DTPL からの頻出な選言的木パターン抽出法を提案した．また，実データを利用したいくつかの比較実験を行い，本手法の有用性を確認した．選言的木パターンは，実用的には辞書のような構造を持つ半構造データに有効であると考えられるため，引き続きその有効性について検証していきたい．

謝　　辞

本研究の一部は文部科学省科学研究費補助金 (課題番号 16500070) の支援をいただいた．

文　　献

[1] Aggarwal, C.C.. and Yu, P.S.: Online Generation of Association Rules, ICDE, 1998, pp.402-411
[2] Agrawal, R. and Srikant, R.: Fast Algorithm for Mining Association Rules, proc. VLDB, 1994, pp.487-499
[3] Agrawal, R. and Srikant, R.: Mining Sequential Patterns, proc. ICDE, 1995, pp.3-14
[4] Asai, T. Abe, K. Kawasoe, S. Arimura, H. Sakamoto, H. and Arikawa, S.: Efficient substructure discovery from large semistructured data, Proceedings of the 2nd SIAM International Conference on Data Mining (SDM '02), 2002, pp.158-174
[5] DBLP, http://dblp.uni-trier.de/
[6] Feng, F. Hsu, W. Lee, M.L.: Efficient Pattern Discovery for Semistructured Data, International Conference on Tools with Artificial Intelligence (ICTAI), 2005, pp.294-301
[7] Goethals, B.: A Survey on Frequent Pattern Mining, Univ.of Helsinki, 2003
[8] Goethals, B., Hoekx, E. and Bussche, J.V.d.: Mining Tree Queries in a Graph, ACM SIGKDD, 2005, pp.61-69
[9] Han,J. and Kamber, M.: Data Mining : Concepts and Techniques, Morgan Kaufmann, 2000
[10] Hand, D., Mannila, H. and Smyth, P.: Principles of Data Mining, MIT Press, 2001
[11] Mannila, H. and Toivonen, H. and Verkamo, I.: Discovery of Frequent Episodes in Event Sequences, *Data Mining and Knowledge Disvocery* 1(3), 1997, pp.259-289
[12] Motoyoshi, M., Miura, T., Watanabe, K. and Shioya, I.: Temporal Class Mining for Time Series Data, proc. CIKM, 2002
[13] Shimizu, K. and Miura, T.:: Disjunctive Sequential Patterns on Single Data Sequence and its Anti-Monotonicity, International Conference on Machine Learning and Data Mining(MLDM), 2005, pp.376-383
[14] Srikant, R and Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements, proc.EDBT, 1996, pp.412-421
[15] Toivonen, H.: Sampling Large Databases for Association Rules, VLDB, 1996, pp134-145
[16] Zaki, M.J.: Efficient Enumeration of Frequent Sequences, proc.CIKM, 1998, pp.68-75
[17] Zaki, M.J.: Efficiently mining frequent trees in a forest, ACM SIGKDD, 2002, pp.71-80
[18] 浅井, 有村: 半構造データマイニングにおけるパターン発見技法, 電子情報通信学会論文誌 VOL.J87-D1 No.2, 2004, pp.79-96
[19] 北, 津田, 獅子堀: 情報検索アルゴリズム, 共立出版, 2002
[20] 清水, 三浦: 選言パターン抽出のオンライン分析, DBSJ Letters Vol.4 No.3, 2006, pp.9-12
[21] 高野，岩沼，鍋島: 単一の長大なデータ系列上の系列パターンの出現尺度とその逆単調性, 第 3 回情報科学技術フォーラム (FIT), 2004, pp.115-118
[22] 長尾 真: 自然言語処理, 岩波書店, 1996
[23] 永田, 平田: "テキスト分類–学習理論の「見本市」–", 情報処理,vol.42(1),pp:32–37(2001)
[24] 比土, 河野: 頻出順序木発見のための効率的な列挙手法の提案, DBSJ Letters Vol.4 No.1, 2005, pp.165-168
[25] 房延, 浅井, 有村, 宇野, 中野: 半構造データマイニングのための高速な無順序木パターン発見手法, データ工学ワークショップ (DEWS), 2004, 6A-o3