選言パターン抽出のオンライン分析

Online Analysis for Disjunctive Sequential Patterns

清水 一宏♣ 三浦 孝夫♦

Kazuhiro SHIMIZU Takao MIURA

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，オンライン分析手法に基づいて，頻出な選言パターンの実用的発見の為のデータ構築法を提案する．また，実験によりその有用性を検証する．

Disjunctive patterns are known as useful for text mining in a single document. In fact, due to anti-monotonicity, we can discuss efficient algorithm as APRIORI. In this work, we propose online analysis method for extracting frequent disjunctive patterns in trials and errors manner. And we show the usefulness of the approach by examining experimental results.

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．しかし，テキストの特徴を数量化し解析しようとする研究(テキストマイニング)が注目されるにつれ, 文書データ分析にデータマイニング手法を適用する研究が開始されている[1].

データマイニング手法の特徴は, 頻度(重要な項目は何度も生じるという考え方)や共起性(複数の項目が同時に生じる)にある. このデータマイニング手法を, 系列データ(sequence data)に適用するアプローチをテキストマイニングと呼ぶ. 系列データは語の並びで構成された意味を有し，テキストマイニング手法により，高頻度パターン(語の並びや句)を抽出し，内容の要約(抽象化)やラベル付けを行うことができる.

パターンは多様な解釈を持ち，出現頻度は各系列における興味の度合いを示す．しかし，系列データを扱う問題は，単一の系列データを扱う場合と複数系列データを扱う場合で大きく異なる．系列データの集合を単位として捉え，系列パターンを含む系列数を出現頻度とする問題を複数系列データ問題と呼ぶ.

一方，単一系列データでは，出現頻度とは系列におけるパターンの重要度と考えることができる．複数系列データとは異なって，単一の著者によって執筆されていることが期待できるため，ひとつのパターンが異なる解釈を有することは少ないと考えられる．従って，単一系列データではパターンの出現頻度がそのまま重要性を表すと考えられるが，既存のデータマイニング手法の多くは複数系列データを対象としており，単一系列中におけるパターン出現頻度については考慮されないため，テキストマイニングには直接利用できない.

一般的に，長大なテキストから発生頻度の高い語句すべてを発見することは，探索空間が巨大であるため，組合わせ的な探索問題となっている[2]. このため，何らかの工夫を要する必要がある．データマイニングの主要な研究のほとんどでは，探索空間の削減にパターンの逆単調性が用いられてきた. 例えば, APRIORIでは探索時間を大幅に下げることができる[3]. しかし，系列パターンについてはこの逆単調性が成り立たず，根源にさかのぼった検討が必要である．系列パターンに対して正規表現を適用する方法が提案されているがNP完全な問題を含み実用的計算量が得られない[4]. 著者らは，**選言**(disjunctive)パターンを導入し，この上で逆単調性を満たす出現尺度を提案した[5].

APRIORIを用いたテキストマイニング操作は，探索問題であるため，一般的にかなりの計算資源を用い，簡単な問題でも数時間を要することは珍しくない．実際に適用する場合, 様々な条件により繰り返して探索することから，小規模な問題あるいはサンプリング手法などにより問題を回避する[6]. 実務などで利用されるOLAP(*Online Analytical Processing*)では，多種大量のデータを扱うため，(例えばAPRIORIでは支持度, 確信度などの)種々のパラメタを予め設定することは容易ではなく，かなり時間を要する作業となっている.

一般的に，条件を充足するパターンの発見問題では，繰り返しデータを走査する必要がある．しかし，各データ走査で非常に多くの読み込みが発生する場合，それがボトルネックとなってしまう．また，計算結果の再利用ができないため，パラメタの設定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[7]. ここでは，予めデータを前処理してその結果を保持し，これを繰り返して利用することにより負荷を軽減する．考え方自身は新しいものではないが，更新頻度が少ない状況，特にOLAPでは相性が良い.

本稿では, テキストマイニングに対するオンライン分析手法[7]を論じる．即ち，特定の形式でデータを表現し，繰り返して生じる問い合わせに対し効率よく選言パターンを抽出する方法を提案する. [7]では，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だが，相関ルールのオンライン生成を扱うため選言パターンは扱わない. SPADE[8]は，複数系列データを対象としているため，テキストマイニングには直接利用できない．また，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で抽出し，それぞれでデータ走査を行う必要がある．2 章では選言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 章で結びとする.

2. 選言パターン(Disjunctive 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$I = \{i_1,\ldots,i_L\}$, $L>0$ に対し，**系列データ** Sとはアルファベットの順序リスト$S = s_1\ldots s_m$, m＞0 である．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.

選言パターン(あるいは単にパターン)pとは$t_1t_2\ldots t_n$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$p = t_1t_2\ldots t_n$, $q = v_1v_2\ldots v_m$, $m \leqq n$があるとき，qがpの**部分パターン**である($q \subseteq p$ と記す)とは，$1 \leqq j_1 < \ldots < j_m \leqq n$があり，各$v_k$は$t_{jk}$に対応して次を満たす(混乱の無い限

♣ 学生会員 法政大学工学部情報電気電子工学科
c02d3051@k.hosei.ac.jp
♦ 正会員 法政大学工学部情報電気電子工学科
miurat@k.hosei.ac.jp

り $v_k \subseteq t_{jk}$と記す):

v_kがアルファベットaのとき，$t_{jk} = a$もしくはaを含む選択

v_kが選択$[a_1a_2..a_m]$のとき，$t_{jk} = [b_1b_2..b_l]$でかつ$\{a_1,...,a_m\} \subseteq \{b_1,...,b_l\}$

［**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．□

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，$t_1 = a_1 = c_{i1}$, $1 \leqq i_1 \leqq m$でありかつ部分パターン$t_2...t_n$が系列データ$c_{i1+1}...c_m$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$a_{j1}...a_{jm}$が系列データ$c_1...c_{i1}$に適合し，かつ部分パターン$t_2...t_n$が系列データ$c_{i1+1}...c_m$に適合するときを言う．

［**例 2**］系列データSが$a_1a_2b_3b_4b_5a_6$(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(a_1, a_2, a_6)し，パターンabは6回適合(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．□

系列データSに関してパターンから非負整数への関数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$q \subseteq p$ のとき$M_S(q) \geqq M_S(p)$ が成り立つときを言う．以下で考慮する系列データは単一でありSを略す．

逆単調な 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，$M(q) < m$ ならば $q \subseteq p$ となる p は $M(p) \geqq m$ 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 は探索範囲を表すため，特に最小支持度(minimum support)と呼ぶ．

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．しかし，本稿で論じるような系列データには“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系列”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[5].

系列データ$S = s_1s_2...s_r$，パターンpを$t_1t_2...t_n$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$t_2t_3...t_n$が$S(i+1)$に適合する．t_1が選択$[a_1a_2...a_m]$のとき，$s_i = a_j$となるjがあり(例えば$j = 1$)，$[a_2a_3...a_m]t_2...t_n$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．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．

$$D(S, p) = MIN\{H(S, q) \mid q \subseteq p\}$$

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表せる．系列データ S，パターンを p とすると，

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

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[5].

［**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，単一系列データSに対して，図1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**選言パターン束**(Disjunctive Pattern Lattice，以下DPL)と呼ぶ．DPL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($E \subseteq V \times V$)を表す．DPL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，$D(S,v)$を**ラベル**として持つ．(n+1)-頻出パターンvがvを部分パターンに持つとき，**有向辺**$(v, v') \in E$が存在する．

［**例4**］Sをcabcbba，m = 1とした時，DPLは図1のように構築される．例えば節点$v_{[a,b]}$は，ラベルとして全体出現頻度2を持つ．また，有向辺$(v_a, v_{[a,b]})$, $(v_b, v_{[a,b]})$から，節点v_a, v_bをそれぞれ部分パターンとして持つ．□

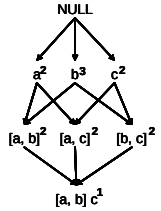

図 1　m=1 で構築された DPL の例

Fig.1　DPL of cabcbba with m = 1

3.1　系列データからの DPL 構築

系列データSからDPLを構築するため，頻出パターンpのS中での出現開始位置$head$と終了位置$tail$を保持した**出現位置リスト**を定義する．すなわち，pのS中での出現位置を$P_p(head, tail)$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．

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

ここでmaxは系列先頭頻度を表し，各$head$は同じリスト中では重複しないようにとる．

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，系列データSからDPLを構築する手順を以下に示す．

入力: 系列データS

出力: DPL L

（1）Sを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，$P_{pi}(tail) < P_{qj}(head)$を満たす$P_{pi}(head)$，$P_{qj}(tail)$をすべて見つけ，それぞれを$P_{pqk}$の$head$, $tail$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．

$$list_p = \{P_{p_1}(head,tail),...,P_{p_{i\max}}(head,tail)\}$$
$$list_q = \{P_{q_1}(head,tail),...,P_{q_{j\max}}(head,tail)\}$$
$$list_{pq} = \{P_{pq_1}(P_{p_i}(head),P_{q_j}(tail)) \mid P_{p_i}(tail) < P_{q_j}(head)\}$$

最大パターンサイズをNとするとき, APRIORIではN+1回のSの走査が必要であるのに対し，上述の手順ではSの走査はNに関係せず，S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］Sをcaabbbcとすると，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，

$$list_a = \{P_{a1}(2,2),P_{a2}(3,3)\}$$
$$list_b = \{P_{b1}(4,4),P_{b2}(5,5),P_{b3}(6,6)\}$$
$$list_c = \{P_{c1}(1,1),P_{c2}(7,7)\}$$

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

$$list_{ab} = \{P_{ab1}(2,4),P_{ab2}(3,4)\}$$
$$list_{[ac]} = \{P_{[ac]_1}(1,2),P_{[ac]_2}(2,7),P_{[ac]_3}(3,7)\}$$

となる．□

3.2 希望語を含む DPL の構築

系列データSからDPLを構築する際に特定の語句(希望語)を指定し，これを含むパターンとその部分パターンのみでDPLを構築する方法を考えることができる．実際，n-頻出パターン中に希望語を含むパターンが生成されなければ, (出現尺度が逆単調性を満たすので)以降希望語を含むパターン生成はありえず，この時点で構築を終了すれば良い．

3.3 DPL からのパターン抽出

構築されたDPLからのパターン抽出は, 深さ優先探索で行う．以下にその手順を示す．

入力: DPL L, 最小支持度m
出力: 出力リストL_o
（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L_o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L_o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L_oを出力する．

［例 6］図 1 において，m = 2 以上の度数を有するすべてのパターンを抽出する．この探索は図 2 のような順路に従い，出力リストL_oを得る．図中の丸囲い数字は，探索の順序を示す．□

4. 実験

4.1 実験の方法

本稿では，3 種類の実験を行う．実験 1 では，DPL を用いたオンライン分析と APRIORI による方法を用いて，選言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，DPL 構築に要する負荷を調べるため，出現位置リストと APRIORI を用いて実行時間の比較を行う．APRIORI を用いての DPL 構築とは，出現位置リストを用いずに，データ走査を行うことで頻出パターンを取得する方法である．実験 3 では，希望語を指定した DPL 構築の負荷を調べる．このため，頻出パターンを制限する場合としない場合の DPL 構築の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パターンが 1 回だけ出現するものに限定する．

実験データとして，Reuters-21578 text categorization test collection を使用する．これは 1987 年のロイター社の発信記事を日時順に並べたコーパスであり, 全記事数は 21,578 件である．今回は，この記事から 925 件分の記事を用い，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を行ったものを使用する．

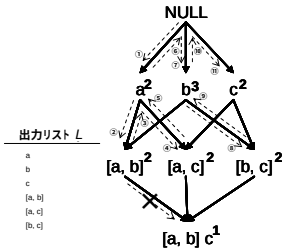

図 2 DPL からの頻出パターン抽出
Fig.2 Extracting Frequent Patterns from a DPL (m = 2)

4.2 実験結果の評価方法

一般に長大な記事ほど単語の出現数が増加し, その頻度値は記事内容の重要性に対応しなくなる可能性が高い．このため, 本実験では, 記事の長さに対して適切な重み付けを行う．ここでは，各頻出語の出現頻度として対数化索引語頻度(logarithmic TF)を用いる．実験データ中の記事jにおける，パターンpに対する出現頻度$LTF_{j,p}$は，次で定義される．

$$LTF_{j,p} = \frac{\log(H(j,p)+1)}{\log(N_j+1)}$$

ただし，j中の総単語数をN_jとする．従って，重みを考慮した先頭系列頻度$Hfreq(S,p)$は，

$$Hfreq(S,p) = \sum_{j=1}^{j\max} LTF_{j,p}$$

となる．$Hfreq(S,p)$は重み付けられた頻度であるので, $H(S,p)$とは異なる．全体出現頻度についての変更はない．

4.3 実験結果

実験 1 では，頻度しきい値 m を 20 から 80 まで 10 刻みで変化させ，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パターンの一部を以下に示す．

[mln, oper, shr, net] note
[loss, profit, shr, ct, net] note
[mln, dlr, 1, shr, ct, net] year

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各手法での DPL 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 を連結したもの(従って 2 倍の長さになる)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．

実験 3 では，しきい値を 8 に固定し，希望語として fell および analyst を用い，指定した場合と指定しない場合の DPL 構築にかかる実行時間の比較を行う．実験 3 の結果を

表 4 にまとめ，各希望語と実行時間の関係を示す．

また，実験 1 においてしきい値が 20 における DPL と APRIORI による，リスト(パターン)数，1 リストあたりの平均ディスク使用量及び総ディスク使用量の比較を表 5 に示す．

表 1　しきい値とパターン数の関係

Table.1　The number of Extracted Patterns

しきい値	8	10	20	30	40	50	60	70	80
パターン数	7766	4060	517	145	58	25	14	9	6

表 2　各手法における選言パターン抽出の実行時間比較

Table.2　Execution Time (sec.) for Extraction

しきい値		20	30	40	50	60	70	80
実行時間[s]	DPL	4.70	3.82	3.05	2.19	1.77	1.15	0.47
	APRIORI	26020	4655	1729	467	242	155	127

表 3　各データサイズにおける DPL 構築の実行時間比較

Table.3　Construction Time (sec.) of DPL

データサイズ	実行時間[s]	
	DPL	APRIORI
データ1	473	26020
データ2	950	128471

表 4　希望語の有無による DPL 構築時間の比較

Table.4　Construction Time (sec.) with/without Specific Words

希望語	実行時間[s]	
	希望語有	希望語無
fell	5245	7734
analyst	7345	7734

表 5　各手法におけるディスク使用量の比較

Table.5　Space Utilization (bytes), m = 20

手法	リスト数（パターン数）	ディスク使用量[byte]	
		平均使用量	総使用量
DPL	517	3954	2043969
APRIORI		27	14037

4.4　考察

実験1のパターン抽出において,一旦DPLを構築すれば、これを用いて選言パターン抽出を輪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．これは APRIORI のような全件データ走査と比較して本質的に効率が良い．さらに，選言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，選言パターンをそのまま残した DPL 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．他方，APRIORI のような全件データ走査方式では選言パターンは選択肢ごとの探索を個別に実行して結果を重ねることに対応し, 重複した走査を要するため, 本質的に効率が悪い．実際，本実験結果からもわかるように，APRIORI を用いた方法に比べおよそ 270 倍～5500 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 4 倍に対して DPL 法で 10 倍なのに対し，APRIORI では 200 倍以上に低下する．

実験 2 では DPL 構築の負荷を調べている．2 種類のデータとも，出現位置リストを用いた方法は APRIORI を用いた方法に比べおよそ 50～130 倍高速となる．これは，出現位置リスト法は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．データサイズの増加に伴い実行時間は増加する．出現位置リスト法では実験データの全走査が 1 度実行され，ちょうど 2 倍悪化するのに対し，APRIORI 法では，N+1 回のデータ走査を行うことから，データサイズの影響が 5 倍の悪化を生んだと考えられる．

実験 3 では，希望語の影響を調査している．2 つの語句 fell, analyst それぞれを指定した場合は，希望語を指定しない場合に比べおよそ 47%及び 5%の高速化となっているに過ぎない．本実験では，DPL 構築に際し，“ 希望語を含む頻出パターン ” の探索を各記事内に閉じて探索している．つまり複数の記事にわたる出現を考慮していないため，全記事に対する探索を避けることができる．しかし，希望語に適合するテキストの検索対象範囲は当該記事全体に及ぶため，希望語の有無が大幅な改善を生むことは無い．

DPL 構築のためのディスク使用量を考察する．DPL を用いた手法では，APRIORI を用いた手法に比べ，およそ 146 倍に悪化する．実際，DPL 法では計算の途中で出現位置リストを保持するため，このような結果を生む．本実験では，実験を簡単にするため計算中すべての出現位置リストを保持したが，実際には(n +1)-頻出アイテムを計算するのには n -頻出アイテムの出現位置リストだけで良く, 計算中のディスク使用量は減らすことが可能である．

5. 結論

本稿ではオンライン分析の考え方を用い, 単一系列データ上からの高速な DPL 構築法および DPL からの頻出パターン抽出法を提案した．また，APRIORI を用いた手法とのいくつかの比較実験を行い，同じ条件下でより高速に頻出パターンを抽出および DPL 構築できることを示し，本手法の有用性を確認した．また本稿では，テキストデータを用いたが，これは(MIDI などのような)一般的な系列データにも拡張可能だと考えられる．

[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] 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

[2] 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

[3] Agrawal, R. and Srikant, R.: Fast Algorithm for Mining Association Rules, VLDB, 1994, pp.487-499

[4] Albert-Lorincz, H. and Boulicaut, J-F.: Mining Frequent Sequential Patterns under Regular Expressions: A Highly Adaptative Strategy for Pushing Constraints, SIAM DM, 2003, pp.316-320

[5] 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

[6] Toivonen, H.: Sampling Large Databases for Association Rules, VLDB, 1996, pp134-145

[7] Aggarwal, C.C.. and Yu, P.S.: Online Generation of Association Rules, ICDE, 1998, pp.402-411

[8] Zaki, M.J.: Efficient Enumeration of Frequent Sequences, CIKM, 1998, pp.68-75

清水 一宏　Kazuhiro SHIMIZU
法政大学工学部情報電気電子工学科在学中.

三浦 孝夫　Takao MIURA
法政大学工学部情報電気電子工学科教授.