ニュースストリームからの選言パターン抽出

清水　一宏†　　三浦　孝夫†

† 法政大学 工学研究科 電気工学専攻　〒 184–8584 東京都小金井市梶野町 3-7-2
E-mail: †shimizu@gs-eng.hosei.ac.jp, ††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，この考え方をニュースストリームに適用し，過去の頻度に対して忘却関数を用いた重み付け法を用いることで時間変化に適切に追従し，任意の時間において確率的にエラー保証された近似解を提供するオンライン型単一パスアルゴリズムを提案する．また実験によりその有用性を検証する．

キーワード　選言パターン，ニュースストリーム，オンライン型単一パスアルゴリズム

Mining Disjunctive Sequential Patterns from News Stream

Kazuhiro SHIMIZU† and Takao MIURA†

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

Abstract　Frequent disjunctive pattern 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 new *online and single-pass algorithm* which can extract current frequent *disjuctive patterns* by a weighting method for past events from a *news stream*. And we discuss some experimental results.

Key words　Disjunctive Sequential Pattern, News Stream, Online and Single-Pass Algorithm

1. 前　書　き

これまで，文書データから重要な語句や規則を抽出するために，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 [6], [9]．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 [23].

データマイニング手法の特徴は，頻度の数え上げや共起性の検出にあり [25], [26]，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.

一方，近年のネットワークや Web 技術の発達により，ネットワーク上を流れる膨大な量の動的なデータであるデータストリーム (data stream) に対するデータマイニング研究が注目されている [7]．一般にデータストリームは，(1) 大量，(2) 高速，(3) データ分布の動的変化，(4) 連続的という特徴を持っており，これを対象とする手法には，限られた計算資源で高速かつ長期間の解析が要求され，任意の時点において正しい解が出力されるような手法であることが望ましく，過去のデータマイニング手法を適用するのは困難であると言われている [7], [19].

データストリームの中でも特に，時系列順にニュース記事や放送内容のアノテーション・トランスクリプションが配信されるものを，ニュースストリーム (news stream) と呼ぶ．本稿ではこのニュースストリームを対象とする．一般に，ニュース記事の発行間隔は不定であるが，発行される記事は速報性が高いため，本稿では記事の発行時間と記事の内容時間は一致するものと仮定する.

ニュースストリーム環境下では，絶えず記事が発生しており，新しい記事と過去の記事が混在している．ユーザは，一般に新しい記事の内容に興味を持っていることが多く，両データを同等の重みで扱うことはできない．従って本稿では，各記事内に発生する語句については平等にオブジェクトとして扱うが，新しい記事に生じるものほど重要であるとする.

APRIORI など静的なデータを対象とした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を用いる．しかし，系列パターンでは成り立たない [2], [5]．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 [17]．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 [15].

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]．特にデータストリーム環境下では，その性質上，過去に遡った再計算が困難であるので，オンライン分析の考え方がより重要となる．筆者らは，オンライン分析を導入し，静的なデータ環境下で，効率的な選言パターン抽出法を提案した [24].

本稿では，ニュースストリーム環境下で，時間変化に適切に追従し，任意の時点において効率的に近似解を提供するオンライン型単一パスアルゴリズムを論じる．即ち，特定の形式のデータ構造を構築し，過去の頻度に対して適切な重み付けを行い，確率的にエラー保証された頻出な選言パターンを抽出する方法を提案する．2 章で関連研究を紹介し，3 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す．4 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の提案を行い，5 章でいくつかの実験結果を示し，6 章で結びとする．

2. 関 連 研 究

データストリームに対するデータマイニング手法としては，損失カウント (Lossy Counting) が提案されている [11]．これは，トランザクションデータストリームを対象としたオンライン型単一パスアルゴリズムであり，確率的にエラー保証された頻出アイテム集合を高速に生成する手法である．また，この考えを応用した研究も提案されており，例えば，センサーネットワークへの応用がある [8]．反面，時間変化への適応性が考慮されていない．

一方，StreamT [19] では忘却型減衰モデルを導入することで時間変化問題を解決している．しかし，半構造データストリームを対象としており，本稿で扱うようなニュースストリームへ応用することは困難である．

データストリーム環境下での研究としては，忘却関数による重み付け [14], [20], [21] や，Tilted-Time Window による重み付け [4] などが提案されているが，いずれの研究もニュースストリームに対するデータマイニングを想定していない．

厳密解を求めるアルゴリズムとしては, [12] が提案されており，上位 k 個のパターンを対象とすることで実行効率を改善しているが，本手法のように全ての頻出パターンの抽出を対象とするアルゴリズムではない．

3. 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 $I = \{i_1, .., i_L\}, L > 0$ に対し，**系列データ** (記事)S とはアルファベットの順序リスト $s_1...s_m, \infty > m > 0$ をいう．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．

選言パターン (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 または選択 $[a_1a_2...a_m], m > 0$ (各 a_j は相異なるアルファベット) である．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]” の部分パターンではない．また”[ab]” は ”ab” の部分パターンではない．□

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，$t_1 = a_1 = c_{i_1}, 1 \leqq i_1 \leqq m$ 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 ($\mathtt{a_1}, \mathtt{a_2}, \mathtt{a_6}$) する．また，パターン ab は 6 回適合 ($\mathtt{a_1b_3}, \mathtt{a_1b_4}, \mathtt{a_1b_5}, \mathtt{a_2b_3}, \mathtt{a_2b_4}, \mathtt{a_2b_5}$) 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 $\mathcal{M}$ 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 $\mathcal{M}_S(q) \geqq \mathcal{M}_S(p)$ 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 $\mathcal{M}$ が与えられ，また最小頻度を与えるしきい値 σ (最小支持度) が与えられているとする．このとき，$\mathcal{M}(q) < \sigma$ ならば $q \sqsubseteq p$ となる p は $\mathcal{M}(p) \geqq \sigma$ 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．しかし，例 2 のような単純な適合頻度では逆単調性が得られない．そこで本稿では逆単調性を満たす出現尺度を提案する [15].

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．

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

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 [15].

［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．
□

ここで，系列先頭頻度について S の長さに対して適切な重み付けを与える．これは，一般に長大な系列ほど単語の出現数が増加し，その頻度値が系列内容の重要性に対応しなくなる可能性が高いためである．ここでは，各頻出語の出現頻度として索引語頻度 (term frequency) を用いる [22]．すなわち，系列データ S における，要素数 n のパターン p に対する，重みを考慮した系列先頭頻度 $H_w(S,p)$ は，次で定義される．

$$H_w(S,p) = \frac{H(S,p)}{|S|-(n-1)}$$

ただし，$|S|$ は S 中の総単語数を表す．従って，全体出現頻度も以下のように書き直すことができる．

$$D_w(S,p) = \mathrm{MIN}\{H_w(S,p), D_w(S,p(2))\}$$

本稿では以降，この重み付けられた系列先頭頻度 $H_w(S,p)$ および全体出現頻度 $D_w(S,p)$ を用いる．

4. ニュースストリームからの選言パターン抽出

ニュースストリームからの選言パターン抽出問題とは以下で定義される．

> **入力**: ニュースストリーム NS, 最小支持度 σ , 許容誤差 ϵ
> **問題**: 現在時刻 $time_{now}$ において，出現頻度が $(\sigma - \epsilon)$ 以上となるような全てのパターンを出力せよ．

ニュースストリーム NS とは，系列データを要素とする無限長順序リスト $S_1...S_i...S_j...$ であり，S_i とは，S_1 を基準として NS 中に現れる i 番目の系列データであることを示す．また，NS 中の各 S_i はそれぞれ発行時刻 $time_i$ を持ち，$i \leqq j$ ならば $time_i \leqq time_j$ である．本研究では，発行時刻 $time_i$ が等しい複数の系列データ $S_{i_1}...S_{i_{wsize}}$ は，単一の系列データ S_i とみなす．ここで，$wsize$ をウインドウサイズと呼び，何らかの上限値を有する，一度の計算で同時に処理できる系列データ数とも考えられる．現在時刻 $time_{now}$ は問い合わせが行われた時刻を表すが，各時点で解釈が若干異なる．以下，本稿では最小支持度 σ 以上の頻度を持つパターンを**頻出パターン** (frequent pattern) と呼ぶ．データストリームを扱う問題の多くでは，データの規模が巨大でかつ長期的な解析を想定しており，一般に解として出力される頻出アイテム集合を近似解とすることで計算コストを抑える．本稿で提案する手法は，近似解を出力する手法の一つである**損失カウント** [11] の手法に基づく．損失カウントでは，各アイテムの推定頻度とその頻度の最大誤差の情報を用いて，次の規則に基づきデータ構造を構築する．

> **規則** 1: あるアイテムがデータ構造中に存在する時，その正確な頻度は推定頻度以上でかつ推定頻度と最大誤差の合計より小さい．
> **規則** 2: あるアイテムがデータ構造中に存在しない時，その正確な頻度は許容誤差以下である．

このようにして構築されたデータ構造から任意の時点で出力される解は，許容誤差 ϵ によって確率的に保証された頻出アイテム集合，すなわち $\sigma - \epsilon$ 以上の頻出パターン集合となる．

本稿では，上記の問題に対して以下の 3 つのフェーズからなるオンライン型単一パスアルゴリズムを提案する．

（1） 系列データからの候補パターン集合生成
（2） 候補パターン集合からの選言パターン束構築
（3） 選言パターン束からの頻出パターン集合抽出

アルゴリズムは，ある時点でニュースストリームから系列データが入力されるたびに (1),(2) を繰り返し実行し，ユーザからの任意の問合せに対して (3) を実行することで，その時点での結果を出力する．フェーズ (1) では，**出現位置リスト** と呼ぶ情報を用いることで効率的に候補パターン集合を生成する．フェーズ (2) では，**選言パターン束** と呼ぶ特殊なデータ構造を上記の規則に従い構築する．フェーズ (3) では，(2) で構築したデータ構造から，任意の時点において高速に頻出パターン抽出を行う．

また，本手法では，発行時刻に基づく重み ω を用い，発行時刻が新しい系列データ上に生じる頻度ほどより大きな重みを与える．ここではフェーズ (2) と (3) において，選言パターン束中の各節点へのアクセス時に**忘却関数** (decay function) による重みを計算し，発行時刻の古い系列データ上で生じる頻度を指数的に減衰させる．

$$\omega = u\lambda^{(time_{now} - time_{last})}$$

ここで $time_{last}$ とは，選言パターン束中の各節点の最終更新時刻である．λ は忘却定数 $0 \leqq \lambda \leqq 1$ を表し，この値が小さくなるほど ω も小さくなり，頻度の忘却の度合いが大きくなる．また u は，

$$u = \begin{cases} 1 & (time_{now} - time_{last} \leqq \tau) \\ 0 & (time_{now} - time_{last} > \tau) \end{cases}$$

で表される単位ステップ関数 (unit step function) であり，τ は有効期間を表す．従って，節点の最終更新時刻からの経過期間が有効期間外であれば，λ がどんな値をとっても，$\omega = 0$ となる．次章では各フェーズについて示す．

5. 提 案 手 法

5.1 系列データからの候補パターン集合生成

NS 中の系列データ S が入力されると，**候補パターン** (candidate pattern) の生成を行う．生成には，候補パターン 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**を用いる．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 $H(S, p)$ を表し，各 $head$ は同じリスト中では重複しない．

出現位置リストを用いて，系列データ S からパターンの要素数が n であるような n -候補パターン集合 C_n を生成する手順を以下に示す．

入力: 系列データ S
出力: n -候補パターン集合 C_n
手順:
$n = 1$ **の時**: S を 1 回走査して 1-候補パターンとその出現位置リストを生成し，C_1 に追加する．
$n > 1$ **の時**: $(n-1)$-候補パターンの出現位置リストから，順次 n -候補パターンとその出現位置リストを取得し，C_n に追加する．

上述の手順で，$(n-1)$-候補パターン p,q の出現位置リスト $list_p$, $list_q$ から n-候補パターン pq の出現位置リスト $list_{pq}$ を得る手順を示す．

入力: 出現位置リスト $list_p$, $list_q$
出力: 出現位置リスト $list_{pq}$
手順:
（1） $list_p$, $list_q$ を突き合わせ，$P_{p_i}(tail) < P_{q_j}(head)$ を満たす $P_{p_i}(head)$，$P_{q_j}(tail)$ をすべて見つけ，それぞれを P_{pq_k}のhead, tail として $list_{pq}$ に追加する．
（2） すべての出現位置を追加し終わったら $list_{pq}$ を出力する．

$$list_p = \{P_{p_i}(head, tail), ..., P_{p_{imax}}(head, tail)\}$$
$$list_q = \{P_{q_j}(head, tail), ..., P_{q_{jmax}}(head, tail)\}$$
$$list_{pq} = \{P_{pq_k}(P_{p_i}(head), P_{q_j}(tail)) | P_{p_i}(tail) < P_{q_j}(head)\}$$

［例 4］ 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，

$$list_a = \{P_{a_1}(2,2), P_{a_2}(3,3)\}$$
$$list_b = \{P_{b_1}(4,4), P_{b_2}(5,5), P_{b_3}(6,6)\}$$
$$list_c = \{P_{c_1}(1,1), P_{c_2}(7,7)\}$$

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

$$list_{ab} = \{P_{ab_1}(2,4), P_{ab_2}(3,4)\}$$
$$list_{[ac]} = \{P_{[ac]_1}(1,2), P_{[ac]_2}(2,7), P_{[ac]_3}(3,7)\}$$

となる．□

5.2 候補パターン集合からの選言パターン束構築

生成された候補パターン集合を用い，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 ，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 $(E \subseteqq V \times V)$ 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 n -候補パターンを表し，三項組 (推定頻度:f, 最大誤差:Δ, 節点最終更新時刻:$time_{last}$) を**ラベル**として持つ．ここで，f とは $time_{last}$ の時点までに計算されたパターンの推定頻度値であり，Δ とは DPL に節点が追加された時点までに生じた可能性のある，パターンの最大の頻度誤差値を示しており, [11] の手法に基づき計算される．$(n+1)$-候補パターン v' が v を部分パターンに持つとき，**有向辺** $e(v, v') \in E$ が存在する．

［例 5］ $S \in NS$ を cabcbba，$time = 1$, $\epsilon = 0.1$ とした時，DPL は図 1 のように構築される．例えば $v_{[a,\ b]}$ は，ラベルとして $(f : 0.29, \Delta : 0.0, time_{last} : 1)$ を持つ．また，$e(v_a, v_{[a,\ b]})$, $e(v_b, v_{[a,\ b]})$ から，v_a, v_b をそれぞれ部分パターンとして持つことがわかる．
□

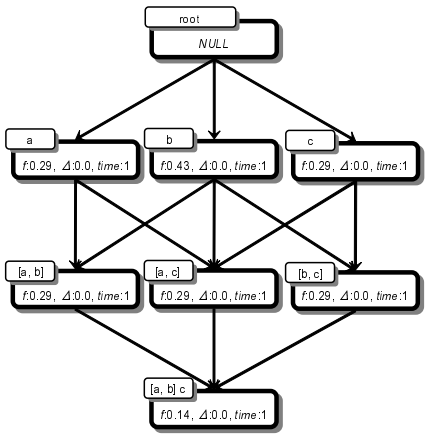

図 1 $\epsilon = 0.1$ で構築された DPL

系列データ S_i から生成された n -候補パターン集合中のパターン $p \in C_n$ について，DPL $\mathcal{D}$ への節点としての追加，更新，削除の各操作を次に示す．

入力: 候補パターン集合 C, 許容誤差 ϵ
出力: DPL $\mathcal{D}$
操作:
追加: $\exists v_p \notin \mathcal{D}$ かつ $\sum D_w(S_i, p) \geqq \epsilon(N_{all} - N_{last})$ であれば，ラベル $(\sum D_w(S_i, p), \epsilon N_{last}, time_i)$ を持つ節点として $\mathcal{D}$ へ v_p を追加する．

更新: $\exists v_p \in \mathcal{D}$ かつ $(\sum D_w(S_i,p) + \omega f + \Delta \geqq \epsilon N_{all})$ であれば，v_p のラベルを $(\sum D_w(S_i,p) + \omega f, \Delta, \mathrm{time}_i)$ として v_p を更新する．

削除: $\exists v_p \in \mathcal{D}$ かつ $(\sum D_w(S_i,p) + \omega f + \Delta < \epsilon N_{all})$ であれば，v_p およびその下位節点 $(n+1$ 以降) を全て削除する．

N_{all} とは S_i も含め，いままでに処理された全系列データ数を表し，N_{last} は S_i を除く，いままでに処理された系列データ数 $N_{all} - wsize$ を表す．

$\sum D_w(S_i,p)$ とは $N_{last}+1$ から N_{all} までの系列データの全体出現頻度 $D_w(S,p)$ の合計を表しており，次の式で表される．

$$\sum D_w(S_i,p) = \sum_{l=N_{last}+1}^{N_{all}} D_w(S_{i_l},p)$$

また，重み ω の計算に用いられる現在時刻 $time_{now}$ は，ここではアルゴリズム側から見た時刻を表しているので，S_i の発行時刻 $time_i$ と一致する．

［例 6］ 図 1 の DPL 中の，節点 v_a を更新することを考える．今，$\epsilon = 0.1$, $N_{all} = 3$, $time_{now} = 3$, $wsize = 1$, $\lambda = 0.98$, $\tau = 3$, $\sum D_w(S,a) = 0.2$ の時，$\omega = 0.98^2 = 0.96$, 頻度 $\sum D_w(S,p) + \omega f + \Delta = 0.2 + 0.28 + 0 = 0.48$ となる．$\epsilon N_{all} = 0.3$ であるので，v_a の新しいラベルは $(f : 0.48, \Delta : 0, time_{last} : 3)$ と更新される．また，ある時点で v_c を DPL 中から削除することを考える場合，その下位節点 $v_{[a,c]}, v_{[b,c]}, v_{[a,b]c}$ も削除される．それぞれの操作を反映した DPL を図 2 に示す．□

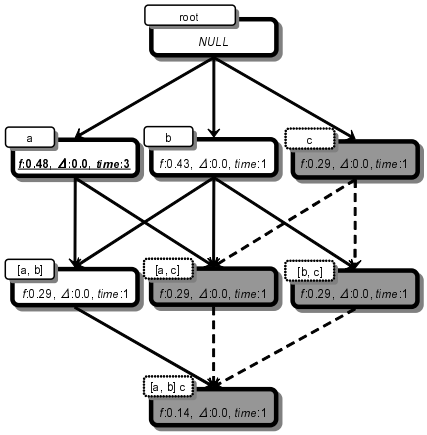

図 2　DPL に対しての更新・削除操作

5.3　選言パターン束からの頻出パターン集合抽出

構築された DPL 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: DPL $\mathcal{D}$, 最小支持度 σ, 許容誤差 ϵ

出力: 頻出パターン集合 $\mathcal{F}$

手順:

（1） NULL 節点から開始して，$(\sigma - \epsilon)N_{all} \leqq \omega f$ 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 $\mathcal{F}$ 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 $(\sigma - \epsilon)N_{all} \leqq \omega f$ を満たせば，節点を $\mathcal{F}$ 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 $\mathcal{F}$ を出力する．

ただし，重み ω の計算に用いられる現在時刻 $time_{now}$ は，ここではユーザ側から見た時刻となるので，ユーザの問い合わせた時刻そのものとなる．

［例 7］ 図 1 において，$\sigma = 0.5$, $\epsilon = 0.1$, $N_{all} = 1$, $time_{now} = 2$, $\lambda = 0.98$, $\tau = 3$ とするとき，$(\sigma - \epsilon)N_{all} = 0.4$ 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 3 のような順路に従い，頻出パターン集合 $\mathcal{F}$ を得る．図中の丸囲い数字は，探索の順序を示す．例えば，パターン a については，$\omega = 0.98$ であるので，$\omega f = 0.28$ となり頻出とはならない．しかし，パターン b については，$\omega f = 0.42$ となるので頻出パターンとして出力される．□

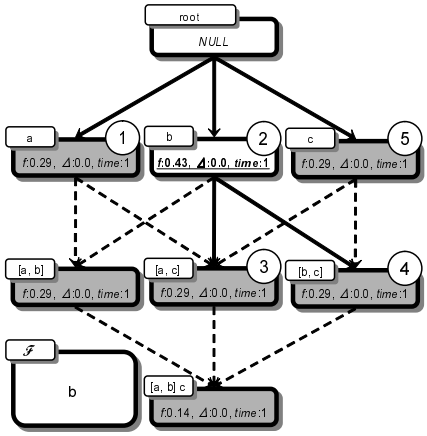

図 3　DPL からの頻出パターン抽出

6. 実　　験

6.1　実験の方法

本稿では，4 種類の実験を行う．実験 1 では，大量のニューススストリームに対して，提案手法の規模耐性を評価する．実験 2 では，各しきい値と DPL 構築に要する負荷の関係を調べるため，提案手法の DPL 構築に関する性能を評価する．実験 3 では，DPL からの頻出パターン抽出性能を評価する．実験 4 では，DPL のスペース効率および時間変化への追従性を評価する．また最後に，実験結果の妥当性を検討し，提案手法で用いた忘却関数による重み付け法の意味を考察する．

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, [abcd]e 等のように) 選言の深さを 1 レベル，パターン長を 5 までに制限する．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 2200 件分の記事 (同発行時刻の記事 100 件を 1 組として，時系列順に 22 組) を用いる．時刻は日単位で扱い，それぞれの記事の発行時刻は，1 組目の発行時刻を基準とした相対時刻で考える．各記事については，不要語 (*stop word*) の削除および単語のステミング [22] を行ったものを実験データとして使用する．以下に記事内容の一例と，表 1 に実験データの詳細を示す．

```
shower continu week bahia cocoa zone allevi drought
earli januari improv prospect temporao humid level
restor...
```

また実験では，4 種類の忘却定数 λ を用い，それぞれ 1.00(重み付け無し),1.00_7(有効期間 1 週間), 0.98(1 ヶ月で重み約 0.5), 0.91(1 週間で重み約 0.5) である．

データ [組]	1	2	3	4	5	6	7	8	9	10	11
時刻 [day]	0	5	7	11	14	15	19	21	25	26	28
差 [day]	-	5	2	4	3	1	4	2	4	1	2
組	12	13	14	15	16	17	18	19	20	21	22
時刻 [day]	32	34	36	40	41	46	95	96	113	235	236
差 [day]	4	2	2	4	1	5	49	1	17	122	1

表 1　実験データの詳細

6.2 実験結果

実験 1 では，許容誤差 ϵ を 0.005, 0.008, 0.010 と変化させ，忘却定数 λ を 0.98 に固定し，実験データ 2200 件に対して，50 件処理する毎に DPL 構築時間を計測し比較を行う．実験結果を図 4 に示す．

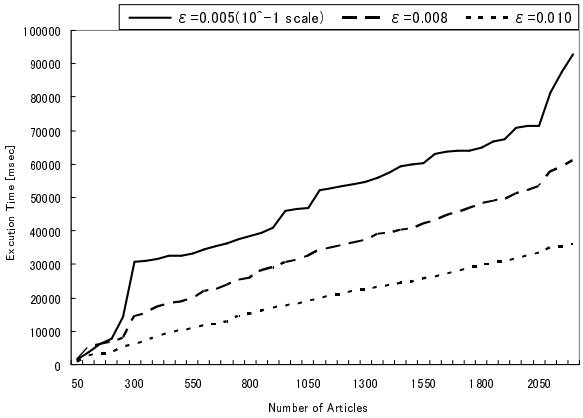

図 4　規模耐性実験

実験 2 では，(1) 許容誤差 ϵ, (2) 実験データの同時処理件数, (3) 忘却定数，の各しきい値と DPL 構築に要する負荷の関係を，DPL への節点の追加・更新・削除の各操作回数および実行時間で比較を行う．(1) では，実験データの同時処理件数を 50 件に固定し，ϵ を 0.005, 0.008, 0.010 と変化させ比較する．(2) では，ϵ を 0.008 に固定し，実験データの同時処理件数を 25, 50, 100 件と変化させ比較する．(3) では，実験データの同時処理件数を 50 件，ϵ を 0.008 に固定し比較する．実験結果を表 2, 表 3, 表 4 にそれぞれ示す．

λ	1.00			1.00_7		
ϵ	.005	.008	.01	.005	.008	.01
追加 [回]	1301	461	258	1360	483	275
更新 [回]	1579	614	364	1368	530	316
削除 [回]	1125	400	223	1206	425	241
実行時間 [sec]	1041.1	67.8	36.2	1054.3	66.8	36.9
λ	0.98			0.91		
ϵ	.005	.008	.01	.005	.008	.01
追加 [回]	1388	487	281	1453	521	296
更新 [回]	1230	486	289	903	359	223
削除 [回]	1233	429	247	1293	462	261
実行時間 [sec]	929.1	61.2	36.1	738.6	56.5	35.7

表 2　ϵ を変化させた時の負荷

λ	1.00			1.00_7		
同時処理数 [件]	25	50	100	25	50	100
追加 [回]	2597	461	172	2618	483	187
更新 [回]	1450	614	266	1328	530	221
削除 [回]	2494	400	134	2524	425	156
実行時間 [sec]	224.7	67.8	48.3	239.1	66.8	50.7
λ	0.98			0.91		
同時処理数 [件]	25	50	100	25	50	100
追加 [回]	2626	487	188	2643	521	203
更新 [回]	1210	486	196	1024	359	124
削除 [回]	2532	429	158	2549	462	172
実行時間 [sec]	208.9	61.2	44.3	196.4	56.5	39.4

表 3　実験データの同時処理件数を変化させた時の負荷

λ	1.00	1.00_7	0.98	0.91
追加 [回]	461	483	487	521
更新 [回]	614	530	486	359
削除 [回]	400	425	429	462
実行時間 [sec]	67.8	66.8	61.2	56.5

表 4　λ を変化させた時の負荷

実験 3 では，許容誤差 ϵ を 0.005，実験データの同時処理件数を 100 件，忘却定数 λ を 0.98 に固定し，実験データを 300 件分用いて構築した DPL から，最小支持度 σ を 0.005,...,0.010(0.001 刻み), 0.015, 0.020, 0.030 と変化させ，頻出パターン抽出の実行時間を比較する．実験結果を表 5 に示す．

実験 4 では，許容誤差 ϵ を 0.005，最小支持度 σ を 0.010，実験データの同時処理件数を 100 件に固定し，忘却定数 λ が 1.00 と 0.91 の時のそれぞれにおいて，DPL 中の節点 (候補パターン) 数と実際に頻出パターンとして抽出されるパターン数を比較する．また，同条件で実験データ 300 件分用いて構築し

σ	.005	.006	.007	.008	.009	.01	.015	.02	.03
時間 [msec]	47	46	32	31	31	31	31	31	16
パターン数	101	101	72	51	39	37	7	5	2

表 5　頻出パターン抽出の実行時間

た DPL から，7 日毎に DPL へ問い合わせを行い，各時点で抽出される頻出パターン数の変化を比較する．実験結果を図 5, 表 6 にそれぞれ示す．

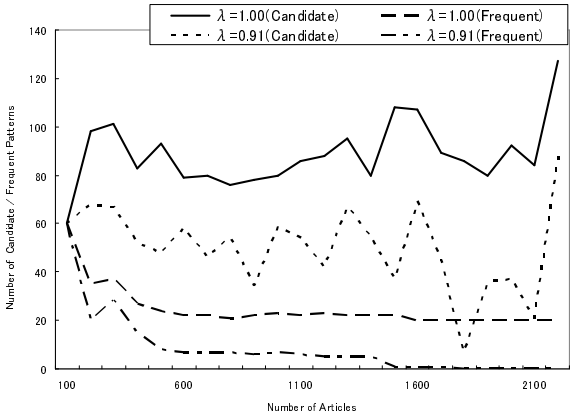

図 5　DPL 中の節点数と頻出パターン数

経過日数	0	7	14	21	28	35	42	49	56	63	70
$\lambda = 1.00$	37	37	37	37	37	37	37	37	37	37	37
$\lambda = 1.00_7$	35	0	0	0	0	0	0	0	0	0	0
$\lambda = 0.98$	37	31	20	18	8	7	7	6	5	4	2
$\lambda = 0.91$	28	7	2	0	0	0	0	0	0	0	0

表 6　経過日数による頻出パターン数

6.3 考　　察

実験結果の考察を行う．実験 1 において規模耐性実験を行っている．全体的には，許容誤差 ϵ が小さいほどパターンの探索範囲が増加するために効率は悪化し，ϵ が 2 倍で実行時間は 25.7 倍となる．また，入力される実験データに対して，実行時間はほぼ線形に推移するが，前半 (0 件から 300 件) では DPL への節点の追加が頻繁に行われるために効率が悪化し，後半 (2050 件から 2200 件) では大きな時間変化の影響により効率が悪化する．いずれの場合も，ϵ が小さいほどその影響も大きい．

実験 2 では DPL 構築の負荷を調べている．(1) 許容誤差 ϵ を変化させての比較では，ϵ が小さくなるほど負荷は増加しており，ϵ が 2 倍で忘却定数 λ が 1.00 の時に実行時間 28.8 倍，全操作回数 (追加, 更新, 削除の各操作の合計回数) は 4.74 倍の悪化，λ が 0.91 の時に実行時間 20.7 倍，全操作回数は 4.68 倍の悪化とそれぞれなる．これは，ϵ 減少に伴い生成される候補パターン数が増加することによると考えられる．

(2) 実験データの同時処理件数の比較を行っている．同時処理件数を増やすことで負荷は減少しており，同時処理件数が 4 倍で忘却定数 λ が 1.00 の時に実行時間 4.65 倍，全操作回数は 11.3 倍の改善，λ が 0.91 の時に実行時間 4.99 倍，全操作回数は 12.5 倍の改善とそれぞれなる．これは，各記事の処理ごとに発生する DPL への各操作が，同時処理件数を増やすことで一括的に実行されることによると考えられる．

(3) 忘却定数 λ の比較を行っている．λ が小さくなるほど負荷は減少しており，具体的には追加や削除の操作回数が増え，更新の操作回数が減少している．これは，λ を小さくすることで，時間変化の影響を大きく受けることとなり，DPL 中の候補パターンの頻繁な入れ替えによると考えられる．

実験 3 においては，DPL からの頻出パターン抽出実験を行っている．最小支持度 σ が小さいほどパターンの探索範囲が増加するため，実行時間も増加しており，σ が 6 倍で実行時間は 2.9 倍となる．しかし，最も遅い場合でも 47 [msec] であることから，オンライン分析により，非常に効率よく頻出パターンが抽出可能であるといえる．

実験 4 では，DPL のスペース効率と時間変化への追従性について調べている．スペース効率については，忘却定数 λ が小さくなると，削除前の候補パターンが DPL に残るため効率は悪くなり，λ が 1.00 の時に平均 28 %，λ が 0.91 の時に平均 17 %となる．これは，λ が小さいほど重みの影響を受け，非頻出となるまでの時間が早いことによると考えられる．

時間変化への追従性については，λ が 0.91 の時において 1600 件目以降，候補パターンおよび頻出パターン数が著しく減少していることがわかる．また，表 6 からも λ が小さいほど新しい話題をより重視する傾向が見られ，7 日後と 21 日後の頻出パターン数は構築直後 (0 日) と比べそれぞれ，λ が 0.98 の時で 84 %と 54 %，λ が 0.91 の時で 25 %と 0 %となり，λ が 1.00_7 では 7 日後以降，頻出パターンは抽出されていない．これらの結果から，適切に重み付けを与えることで過去の候補パターンを削除し，時間変化に適切に追従できていることがわかる．

最後に，実験結果の妥当性を検討し，提案手法で用いた忘却関数による重み付けの意味を考える．上記の実験では，忘却定数 λ の小さいもの程，実行効率の改善が見られるが，これは言い換えれば，他の部分でその対価を払っていると考えることができる．そこで，許容誤差 ϵ を 0.005, 実験データの同時処理件数を 100 件と設定し構築した DPL から，最小支持度 σ を 0.01 として抽出できる全頻出パターンの内，その合計頻度が全頻出パターンの合計頻度の 81 %を占めるような支配的な 13 パターンを選択し，各パターンについて λ が 1.00 の時に対するカバレージ $Cover$ を比較する．$Cover$ は次の式で表される．

$$Cover[\%] = \frac{\text{任意の}\lambda\text{時のパターン抽出回数 (100 件毎問合せ)}}{\lambda\text{=1.00 時のパターン抽出回数 (100 件毎問合せ)}} \times 100$$

表 7 は，実験データ 2200 件での各パターンのカバレージを示している．ここでは，λ が小さいほどカバレージが低下していることがわかり，0.91 の時では 33.3 [%] にまで低下している．つまり，実行効率は頻出パターンのカバレージを犠牲にすることで得られている．忘却関数による重み付けは，前回の頻度更新からの経過期間が長いほど頻度を早く減衰させる．即ち，局所的に頻出となるようなパターンを制限し，その結果として

実行効率が改善されている．

さらに，表 1 からもわかるとおり，実験データの 1800 件目以降は時間変化が急激になるため，これ以降は強制的に，前回更新からの経過期間が長くなる．そこで，この部分を除き，実験データ 1700 件でのカバレージを表 8 に示す．このとき，最大で λ が 1.00_7の時に，実験データ 1700 件目から 2200 件目まででカバレージが 21.4[%] 低下していることがわかる．このことからも，忘却関数による重み付けによって，全域的に頻出となるようなパターンを妥当な時間内に抽出できると言える．

Cover[%]	said	mln	dir	pct	year	billion	campani
$\lambda : 1.00$	100	100	100	100	100	100	100
$\lambda : 1.00_7$	77.3	77.3	77.3	77.3	77.3	100	77.3
$\lambda : 0.98$	90.9	86.4	77.3	77.3	72.7	40.0	77.3
$\lambda : 0.91$	77.3	63.6	63.6	45.5	18.2	6.7	18.2
Cover[%]	bank	share	ct	net	loss	sale	**合計**
$\lambda : 1.00$	100	100	100	100	100	100	*100*
$\lambda : 1.00_7$	61.5	77.3	77.3	77.3	100	100	*78.6*
$\lambda : 0.98$	0	72.7	77.3	50.0	0	33.3	*67.1*
$\lambda : 0.91$	0	18.2	45.5	13.6	0	33.3	*33.3*

表 7　各語のカバレージ (2200 件)

Cover[%]	said	mln	dir	pct	year	billion	campani
$\lambda : 1.00$	100	100	100	100	100	100	100
$\lambda : 1.00_7$	100	100	100	100	100	100	100
$\lambda : 0.98$	100	100	100	100	94.1	40.0	100
$\lambda : 0.91$	100	82.4	82.4	58.8	23.5	6.7	23.5
Cover[%]	bank	share	ct	net	loss	sale	**合計**
$\lambda : 1.00$	100	100	100	100	100	100	*100*
$\lambda : 1.00_7$	100	100	100	100	100	100	*100*
$\lambda : 0.98$	0	94.1	100	64.7	0	33.3	*82.6*
$\lambda : 0.91$	0	23.5	58.8	17.6	0	33.3	*44.6*

表 8　各語のカバレージ (1700 件)

7. 結　　論

本稿では，ニュースストリームからの高速な 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．提案した手法は，オンライン型の単一パスアルゴリズムであり，確率的にエラー保証がされた近似解を出力し，忘却関数による重み付け法を用いることで，時間変化に適切に追従することが可能である．また，いくつかの実験を通して，現実のデータに対して十分に有効であることを確認した．今後の研究として，データ分布の変化に対しての実行効率改善などが考えられる．

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] Chen, Y., Dong, G., Han, J., Wah, B. W. and Wang, J.: Multi-dimensional regression analysis of time-series data streams, International Conference on Very Large Databases, 2002, pp.323-334

[5] Goethals, B.: A Survey on Frequent Pattern Mining, Univ.of Helsinki, 2003

[6] Han,J. and Kamber, M.: Data Mining : Concepts and Techniques, Morgan Kaufmann, 2000

[7] Jian, N. and Gruenwald, L.: Research Issues in Data Stream Association Rule Mining, SIGMOD Record 35-1, 2006, pp.14-19

[8] Loo, K.K., Tong, I. and Kao, B.: Online Algorithms for Mining Inter-stream Associations from Large Sensor Networks, Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2005, pp.143-149

[9] Hand, D., Mannila, H. and Smyth, P.: Principles of Data Mining, MIT Press, 2001

[10] 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

[11] Manku G. S. and Motwani R.: Approximate frequency counts over data streams, the 28th International Conference on Very Large Data Bases, 2002, pp.346-357

[12] Metwally, A., Agrawal, D. and Abbadi, A. E.: Efficient computation of frequent and top-k elements indata streams, Proc. ICDT, 2005.

[13] Motoyoshi, M., Miura, T., Watanabe, K. and Shioya, I.: Temporal Class Mining for Time Series Data, proc. CIKM, 2002

[14] Ohuchi, H., Miura, T. and Shioya, I.: Document Retrieval using Projection by Frequency Distribution, IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2005, pp.356-361

[15] 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

[16] Srikant, R and Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements, proc.EDBT, 1996, pp.412-421

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

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

[19] 安部, 川副, 浅井, 有村, 有川: 大規模半構造データストリームからの知識発見，データ工学ワークショップ (DEWS), 7C-02, 2003

[20] 石川, 北川: 忘却の概念に基づくクラスタリング手法の改良方式, DBSJ Letters Vol.2 No.3, 2003, pp.53-56

[21] 上嶋，三浦，塩谷: 不完全なニュース集合からのタイムスタンプ推定，データ工学ワークショップ (DEWS), 3C-o4, 2005

[22] 北, 津田, 獅子堀: 情報検索アルゴリズム, 共立出版, 2002

[23] 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

[24] 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, 2005, pp.9-12

[25] 長尾 真: 自然言語処理, 岩波書店, 1996

[26] 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)