Vol.2014-DBS-159 No.20
Vol.2014-IFAT-115 No.20
2014/8/2

確率的データストリームにおける パターン問合せ結果のグループ化

杉浦 健人[1,a)]　石川 佳治[1,2,b)]　佐々木 勇和[3,c)]

概要：確率的データストリームに対してパターンマッチングを行うことで，より高次のイベントを発見しようとする試みがある．しかし，各イベントに生起確率が付与された確率的データストリームでは，パターンマッチングを行うと重複した時間帯に複数のマッチが発見されてしまう．このような重複して存在するマッチはいずれもその時間帯にパターンが存在した可能性を示しており，個々を区別することは必ずしも重要ではない．そこで本稿では，重複したマッチをグループとしてひとまとめにする手法を提案する．また，グループの確率的な意味について考えるために，確率的データストリームに対してパターンマッチングを行った際の確率空間を定義する．その後，グループを生成するためのセマンティクスやグループに対して付与する確率の定義，マッチの発見とグループの生成を効率的に行う手法などについて提案する．

1. はじめに

近年，スマートフォンのように多数のセンサを持つ機器が普及したことにより，得られたセンサデータを有効活用しようとする動きが盛んである．特にセンサデータを用いて人の行動をモニタリングする研究は，医療や介護への応用なども考えられ活発に行われている [4], [8]．行動認識の結果はセンサデータに混入するノイズや行動認識の誤りによって不確かなものとなるため，**図 1** のように各時刻の行動を確率で表す**確率的データストリーム**として表現される．さらに，行動認識からは各時刻におけるユーザの行動しか得られないため，**パターンマッチング**によってより高次のイベントを発見しようとする試みが注目されている [1], [3]．パターンマッチングを利用することで「立ち止まっていた人が歩き出し，その後走りだした」というような複雑な行動の検出も可能になり，様々な応用が期待できる．

しかし，確率的データストリームでのパターンマッチングには，同じ時間帯に複数のマッチが検出されるという問題がある．例えば**図 2** は，図 1 の確率的データストリームに対して正規表現 `stand walk+ run` でパターンマッチングを行った結果の一部を示している． すべてのマッチを確認するのはユーザにとって煩雑であるため，既存研究では確率の値が高いマッチを優先的に報告するというアプローチがとられている [5], [6], [7]．しかし，発見されるマッチはいずれもこの時間帯にパターンが存在した可能性を示しており，個々を区別することは必ずしも適切ではない．むしろこれらのマッチを統合して，「時刻 t_1 から t_6 のどこかでパターンに一致する行動をとった確率」を考えた方が有益である．例えば，図 1 から発見されるマッチの中で最も確率の高いマッチは m_2 であるが，確率の値は高々 35% 程しかない．一方で，発見されるマッチをすべて統合して時刻 t_1 から t_6 のどこかでパターンに一致する行動をとった確率を考えると，その確率は約 94% にもなる．

そこで，本稿では同じ時間帯に存在する複数のマッチを一つの**グループ**にまとめる手法を提案する．本稿の構成は

時刻		t_1	t_2	t_3	t_4	t_5	t_6
行動	stand	1.0	0.3	0.1	0.1	0	0
	walk	0	0.7	0.8	0.7	0.9	0
	run	0	0	0.1	0.2	0.1	1.0

図 1　確率的データストリームの例

	t_1	t_2	t_3	t_4	t_5	t_6	確率
m_1	stand	walk	run				0.0700
m_2	stand	walk	walk	walk	walk	run	0.3528
m_3		stand	walk	walk	run		0.0168
m_4				stand	walk	run	0.0900

図 2　パターン `stand walk+ run` に対応するマッチの一部

[1] 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya University
[2] 国立情報科学研究科
National Institute of Informatics
[3] 名古屋大学未来社会創造機構
Institute of Innovation for Future Society, Nagoya University
[a)] sugiura@db.ss.is.nagoya-u.ac.jp
[b)] ishikawa@is.nagoya-u.ac.jp
[c)] yuya@db.ss.is.nagoya-u.ac.jp

以下の通りである．まず，パターンマッチングの結果について厳密に議論するために，確率的データストリームに対してパターンマッチングを行った際の確率空間を定義する．次に，マッチをひとまとまりのグループにするためのセマンティクスを提案し，グループの定義を述べる．その後，グループの生成方法とグループ確率の効率的な計算方法を紹介する．最後に，グループの特徴を評価するために行った実験について説明し，本稿のまとめを述べる．

2. 確率空間の定義

本章では，確率的データストリームに対してパターンマッチングを行う際に必要な確率空間を定義する．まず，本研究で用いる二つの仮定を以下に示す．

(1) 単位時間毎にイベントが発生し，イベントは発生した順に処理される．また，イベントの欠落は起こらない．
(2) ある時刻 t_1 におけるイベントの生起確率は他の時刻 t_2 におけるイベントの生起確率と独立している．

例えば図 1 の確率的データストリームにおいて，生起確率の独立性により $\texttt{stand}_{t_2}$ と $\texttt{walk}_{t_3}$ が共に発生する確率は $0.3 \times 0.8 = 0.24$ である．なおこれ以降，ある時刻 t に対して一単位時間進んだ時刻を $t+1$ で表す．

以上の仮定の元で，まず対象となる確率的データストリームを次のように定義する．

定義 1 **確率的データストリーム** PDS は，$PDS = \langle e_1, e_2, ..., e_t, ... \rangle$ で与えられる無限の系列である．e_t は時刻 t におけるイベントの確率分布である．また，V はイベントのドメインであり，離散的であるとする．イベント e_t の各イベント値 $\alpha \in V$ には生起確率 $P(e_t = \alpha)$ が付与されており，$\sum_{\alpha \in V} P(e_t = \alpha) = 1$ が成り立つ． □

例えば，図 1 の確率的データストリームは $PDS = \langle e_{t_1}, e_{t_2}, e_{t_3}, e_{t_4}, e_{t_5}, e_{t_6} \rangle$ のように表すことができる．また，イベント e_{t_3} の確率分布の和は $\sum_{\alpha \in \{\texttt{stand},\texttt{walk},\texttt{run}\}} P(e_{t_3} = \alpha) = 0.1 + 0.8 + 0.1 = 1$ である．

次に，確率的データストリーム上の具体的なイベントの系列として，シーケンス $s_{[t_1,t_2]}$ を定義する．

定義 2 **シーケンス** $s_{[t_1,t_2]}$ は，$s_{[t_1,t_2]} = \langle \alpha_{t_1}, \alpha_{t_1+1}, ..., \alpha_{t_2} \rangle$ で与えられる時刻 t_1 から時刻 t_2 までのイベントの系列である．ただし，$\alpha_i \in V$ とする．シーケンスの確率は $P(s_{[t_1,t_2]}) = \prod_{i=t_1}^{t_2} P(e_i = \alpha_i)$ で与える． □

例えば，図 1 におけるシーケンスの例としては，$s_{[t_1,t_3]} = \langle \texttt{stand}_{t_1}, \texttt{stand}_{t_2}, \texttt{walk}_{t_3} \rangle$ などがあり，$P(s_{[t_1,t_3]}) = 1.0 \times 0.3 \times 0.8 = 0.24$ である．また，ユーザによってウィンドウ $w = [t_1, t_2]$ が指定されているとき，始まりと終わりがウィンドウ w と一致するシーケンスを s_w，シーケンス s_w の全体集合を S_w で表すことにする．なお，ウィンドウとは無限に存在するデータストリームを有限に区切るためのもの [2] であり，例えば図 1 の確率的データストリームはウィンドウ $[t_1, t_6]$ で有限に区切られた例である．

以上を用いて，パターンマッチングにおける確率空間とマッチの定義を以下に示す．

定義 3 ウィンドウ w が与えられたとき，確率的データストリーム上でのパターンマッチングにおける確率空間を $(2^{S_w}, P)$ で与える．2^{S_w} は S_w のべき集合を表し，P は $P(x \in F_{S_w}) = \sum_{s_w \in x} P(s_w)$ によって事象 x に確率を与える．なお，$P(S_w) = 1$ である． □

定義 4 **マッチ** m は，与えられた正規表現パターンを部分系列に含むシーケンスの集合である．また，確率空間 $(2^{S_w}, P)$ が与えられたとき，マッチの確率は $P(m) = \sum_{s_w \in m} P(s_w)$ である． □

例えば，図 1 の確率的データストリームに対してウィンドウ $w = [t_1, t_4]$ を与えたとき，図 2 のマッチ $m_1 = \langle \texttt{stand}_{t_1}\ \texttt{walk}_{t_2}\ \texttt{run}_{t_3} \rangle$ は次の三つのシーケンスの集合である．

- $s_1 = \langle \texttt{stand}_{t_1}, \texttt{walk}_{t_2}, \texttt{run}_{t_3}, \texttt{stand}_{t_4} \rangle$
- $s_2 = \langle \texttt{stand}_{t_1}, \texttt{walk}_{t_2}, \texttt{run}_{t_3}, \texttt{walk}_{t_4} \rangle$
- $s_3 = \langle \texttt{stand}_{t_1}, \texttt{walk}_{t_2}, \texttt{run}_{t_3}, \texttt{run}_{t_4} \rangle$

従って，確率空間 $(2^{S_w}, P)$ 上において，マッチ m_1 の確率は $P(m_1) = P(s_1) + P(s_2) + P(s_3) = 0.07$ となる．

3. グループの定義

本章では，グループの定義について述べる．まず，マッチをグループにまとめるための方針として**完全オーバラップ**と**部分オーバラップ**の二つを定義する．なお，$ts\_overlap(m, m')$ は二つのマッチ m と m' の時区間に交わりがあるときに真となる述語記号である．

定義 5 あるマッチの集合 M が次の式を満たすとき，集合 M は完全オーバラップの性質を持つと言う．

$$\forall m, m' \in M \ such\ that\ ts\_overlap(m, m') \quad (1)$$

□

定義 6 あるマッチの集合 M が次の式を満たすとき，集合 M は部分オーバラップの性質を持つと言う．

$$\forall m \in M, \exists m' \in M \ such\ that\ m \neq m' \wedge ts\_overlap(m, m') \quad (2)$$

□

以上二つのオーバラップを用いて，グループを以下のように定める．

定義 7 **グループ** g は，完全オーバラップまたは部分オーバラップの性質を持つ極大なマッチの集合である．グループ g は，開始時刻 t_s，終了時刻 t_e，確率 p の三つ組として $g = (t_s, t_e, p)$ で表現する． □

例えば，**図 3** は完全オーバラップと部分オーバラップによるグループの生成例である．図 3 には四つのマッチが存在しているが，完全オーバラップでは実線で表した三つのグループ $g_1 = (t_1, t_4, p_1), g_2 = (t_2, t_5, p_2), g_3 = (t_3, t_6, p_3)$ に，部分オーバラップでは点線で表した一つのグループ $g_4 = (t_1, t_6, p_4)$ にまとめられる．

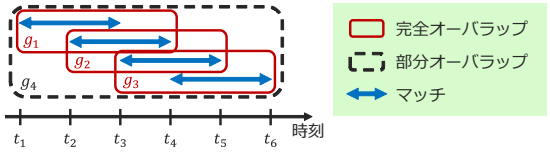

図 3　オーバラップによるグループの生成例

次にグループの確率を考える．定義 7 に示したようにグループはマッチの集合である．また，定義 4 で述べたようにマッチはシーケンスの集合である．従って，グループもシーケンスの集合として表現できる．ゆえに，定義 3 の確率空間に基づけばグループの確率は次のように定義できる．

定義 8　グループ $g = \{m_1, ..., m_n\}$ と確率空間 $(2^{S_w}, P)$ が与えられたとき，その確率 $P(g)$ を次式で与える．

$$P(g) = P\left(\bigcup_{m_i \in g} m_i\right) = \sum_{s_w \in \bigcup_{m_i \in g} m_i} P(s_w) \qquad (3)$$

□

4. グループの生成方法

本章では，ユーザからパターンを受け取ったときに，確率的データストリームからどのようにグループを生成するかについて述べる．まず，完全オーバラップによってグループを生成するアルゴリズムを図 4 に示す．アルゴリズム中の r_t は時刻 t に生成される新しいマッチの候補を表し，一度グループに追加された候補は各時刻で更新されるものとする．例えばパターン `stand walk+ run` に対して，r_t は $\langle \mathtt{stand_t} \rangle$ を表し，時刻 $t+1$ には $\langle \mathtt{stand_t\ walk_{t+1}} \rangle$ に更新される．グループを生成する際には確率的データストリーム上のすべてのマッチを用いることも考えられるが，パターンがクリーネ閉包を含んでいると生成されるマッチの数が多くなりグループ生成の効率が悪くなる．そこで，本稿ではマッチの確率に閾値を設定し，閾値未満の確率を持つマッチを除くことにする．そのため，マッチの候補 r_t は更新とともに確率の計算を行い，確率がマッチ確率の閾値を下回った時点で削除される．

具体的に，図 1 の確率的データストリームとマッチ確率の閾値 0.01 を用いてどのようにグループが生成されるか説明する．まず，入力として確率的データストリームが与えられ，時刻 t_1 のイベント e_{t_1} から処理を始める（3〜5 行目）．イベント e_{t_1} から $r_{t_1} = \langle \mathtt{stand_{t_1}} \rangle$ が生成された後（9 行目），r_{t_1} を持つグループ g_1 が生成される（23 行目）．その後時刻 t_2 では候補 r_{t_2} が，時刻 t_3 では候補 r_{t_3} がグループ g_1 に追加され（13 行目），時刻 t_3 終了時点で g_1 は一つのマッチ $m_1 = \langle \mathtt{stand_{t_1}\ walk_{t_2}\ run_{t_3}} \rangle$ と三つのマッチ候補 $r_{t_1} = \langle \mathtt{stand_{t_1}\ walk_{t_2}\ walk_{t_3}} \rangle, r_{t_2} = \langle \mathtt{stand_{t_2}\ walk_{t_3}} \rangle, r_{t_3} = \langle \mathtt{stand_{t_3}} \rangle$ を持つ．時刻 t_4 では，グループ g_1 はマッチ m_1 を t_3 で得ているため，r_{t_4} を持つ新しいグループ g_2 が生成される（17 行目）．その後は新しい候補はすべて g_2 へと追加され，g_1 は候補の更新のみが行われる．なお，同様の処理を時刻 t_6 まで続けていくが，確率の値が閾値 0.01 未満になるようなマッチや候補 (例：$\langle \mathtt{stand_{t_3}\ walk_{t_4}\ run_{t_5}} \rangle$) は適宜削除されていく点に注意する．そして，g_1, g_2 ともにマッチの候補 r_t が無くなる時刻 t_6 に出力され（20 行目），処理が終了する．

```
1:  function GenerateGroup(PDS)
2:    G = ∅ // グループの集合を表す大域変数
3:    while ストリーム PDS からイベント e_t が得られる do
4:      ManageGroupUsingCompleteOverlap(e_t)
5:    end while
6:  end function
7:  procedure ManageGroupUsingCompleteOverlap(e_t)
8:    e_t を用いてマッチの候補 r_{t_i} を更新し，閾値の判定を行う
9:    e_t から時刻 t のマッチの候補 r_t を生成する
10:   if G ≠ ∅ then
11:     for each g ∈ G do
12:       if g が初めてマッチを得た時刻 t_f に対し t ≤ t_f then
13:         グループ g にマッチの候補 r_t を加える
14:       end if
15:     end for
16:     if r_t がどのグループにも追加されていない then
17:       r_t を要素として持つグループ g_new を生成し G に加える
18:     end if
19:
20:    if マッチ候補 r_{t_i} を一つも持たない g ∈ G が存在する then
21:      グループ g を出力し，グループ集合 G から除く
22:    end if
23:   else
24:     r_t を要素として持つグループ g_new を生成し G に加える
25:   end if
26: end procedure
```

図 4　完全オーバラップによるグループ生成のためのアルゴリズム

部分オーバラップによるグループの生成を行うには，図 4 のアルゴリズムの 19 行目に次の処理を追加する．

19-1: **if** 時刻 t にマッチが受理される **then**
19-2:　　G 中のグループを一つに統合する
19-3: **end if**

この処理は，前の時刻の時点ではオーバラップするかわからなかったため別々に生成しておいたグループを，ひとまとめにする処理である．完全オーバラップの場合と同様に，図 1 を用いて具体的に説明する．時刻 t_5 までは完全オーバラップの場合と同じで，グループ g_1 と g_2 が生成されている．時刻 t_5 にグループ g_1 は図 2 のマッチ m_3 を得るため，この時点でグループ g_1 のマッチ m_3 と g_2 のマッチ候補 r_{t_4}, r_{t_5} がオーバラップする．つまり，グループ g_1 と g_2 に含まれるマッチがオーバラップすることが確定するため，二つのグループを一つのグループ g にまとめることができる．その後はマッチ候補の追加やマッチの受理を繰り返し，時刻 t_6 にグループ g を出力する．

5. グループ確率の効率的な計算方法

定義 8 に示したように，グループ中に含まれるすべての

Vol.2014-DBS-159 No.20
Vol.2014-IFAT-115 No.20
2014/8/2

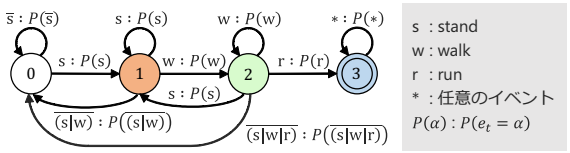

図 5 パターン `stand walk+ run` に対応するトランスデューサ

シーケンスの確率を計算すれば，グループの確率は計算できる．しかし，シーケンスの数は非常に多くなるため，すべてのシーケンスの確率を計算するのは効率が悪い．そこで本章では，トランスデューサを利用することで効率良くグループ確率を計算する手法について述べる．まず部分オーバラップによって生成されたグループの計算方法について説明し，次に完全オーバラップの場合について述べる．

5.1 部分オーバラップの場合

部分オーバラップにより生成されたグループは，図 3 に示すように，その開始時刻 t_s から終了時刻 t_e までに存在するすべてのマッチをひとまとめにしたものとして見ることができる．従って部分オーバラップの確率を求めるという問題は，時区間 $[t_s, t_e]$ において，パターンに一致する部分系列を持つシーケンスの確率の総和を求める問題として考えられる．

提案手法では，ユーザから与えられたパターンを基に確率計算用のトランスデューサを作成し，各グループで一つ保持する．トランスデューサを作成する際はパターンに対応する遷移の他に，受理状態には任意のイベントで受理状態に移る遷移を，他の状態にはパターンの最初のイベントで状態 1 に移る遷移及び一致するイベントがないとき初期状態 0 に戻る遷移を加える．実際にパターン `stand walk+ run` からトランスデューサを作成した例を**図 5** に示す．なお各遷移の添字は，遷移に必要なイベントとその遷移確率を示す．このトランスデューサはパターンに一致する部分系列を持つシーケンスをすべて受理できる．ゆえに，グループが出力される時点でこのトランスデューサの受理状態に到達している確率，つまりこのトランスデューサの出力を，部分オーバラップによって生成されたグループの確率として扱うことができる．

実際の処理では，4 章で述べたグループの生成と並行して確率計算を行う．まず，グループが生成された時点でそのグループが保持するトランスデューサの各状態にいる確率を，初期状態 0 にいる確率が 1.0 であるように初期化する．その後は各時刻で与えられるイベントの確率分布 e_t を用いて，トランスデューサの各状態にいる確率を更新する．確率の更新は，現在の状態にいる確率とその状態から別の状態に遷移する確率を乗算し，それを遷移後の状態の確率とすることで行う．例えば図 5 の状態 2 の場合，「時刻 $t-1$ に状態 1 もしくは 2 にいる確率 × 時刻 t におけるイベント `walk` の生起確率 $P(\mathtt{w})$」が時刻 t における状態 2 の確率となる．このように各状態にいる確率を更新し，グループが出力される段階で受理状態に到達している確率がそのグループの確率となる．

時刻		-	t_1	t_2	t_3	t_4	t_5	t_6
状態	0	1.0	0	0	0.03	0.047	0.0563	0.0563
	1	0	1.0	0.3	0.10	0.093	0	0
	2	0	0	0.7	0.80	0.630	0.6507	0
	3	0	0	0	0.07	0.230	0.2930	0.9437

図 6 入力が図 1 の確率的データストリームである場合に，図 5 のトランスデューサの各状態に到達する確率を計算した例

具体的な例として，図 1 の確率的データストリームに対してパターン `stand walk+ run` のマッチングを行った場合を考える．部分オーバラップの場合，4 章で述べた手続きにより，ウィンドウ $[t_1, t_6]$ 中に存在するマッチをすべてひとまとめにしたグループ $g = (t_1, t_6, p)$ が生成される．このグループ g の生成と並行して，トランスデューサの各状態にいる確率を計算したものを**図 6** に示す．まず，グループ g が生成された時点で，各状態にいる確率が初期化される．そして，時刻 t_1 では状態 0 から状態 1 への遷移が実行され，$P(\mathtt{s}) = 1.0$ であるため状態 1 にいる確率が 1.0 となる．次に時刻 t_2 では状態 1 から状態 1 と 2 への遷移が実行され，$P(\mathtt{s}) = 0.3, P(\mathtt{w}) = 0.7$ であるため，状態 1 にいる確率が 0.3，状態 2 にいる確率が 0.7 となる．このような計算をグループ g が出力される時刻 t_6 まで行い，時刻 t_6 に受理状態にいる確率 0.9437 がグループの確率 p となる．

5.2 完全オーバラップの場合

グループが完全オーバラップによって生成される場合，図 6 の計算ではグループに含まれないマッチの確率まで加えてしまうため，正確なグループ確率を求めることができない．例えば，図 1 においてパターン `stand walk+ run` のマッチングを行い，完全オーバラップでグループを生成した場合を考える．このとき，図 2 のマッチ m_1 とマッチ m_4 がオーバラップしないため，マッチ m_4 以外をまとめたグループが生成される．このグループの確率を計算する際に図 6 のような方法を用いるとマッチ m_4 の確率も含めて計算してしまうため，実際とは異なる確率となってしまう．

完全オーバラップにより生成されたグループの確率を正確に計算出来ない原因は，グループに含まれないマッチの確率も足しあわせてしまっていることである．従って，グループに含まれないマッチが受理されないトランスデューサを使用すれば，5.1 節で述べた方法でグループ確率を計算できる．しかし，そのようなトランスデューサは単にマッチを受理するだけのトランスデューサよりも複雑になるため，効率的な計算には不向きである．そこで，グループ確率の計算途中でトランスデューサを変形することで，グループに含まれないマッチを受理しないようにすることを

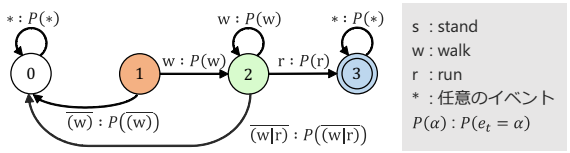

図 7 図 5 のトランスデューサを完全オーバラップにより生成されたグループの確率を計算するために変形した例

時刻		-	t_1	t_2	t_3	t_4	t_5	t_6
状態	0	1.0	0	0	0.03	0.14	0.140	0.140
	1	0	1.0	0.3	0.10	0	0	0
	2	0	0	0.7	0.80	0.63	0.567	0
	3	0	0	0	0.07	0.23	0.293	0.860

図 8 図 7 のトランスデューサを利用して確率を計算した例

考える．完全オーバラップの場合，一番最初のマッチが受理された時点でそれ以降に生成されるマッチは同じグループに含まれない．ゆえに，一番先頭のマッチを受理した時点でそれ以降のマッチが確率の計算に含まれないようトランスデューサを変形すれば，5.1 節で述べたものと同じ方法で完全オーバラップの場合のグループ確率も計算できる．

例えば図 6 の場合，初めてマッチを受理した時刻 t_3 の時点で図 5 のトランスデューサを**図 7** に変形することで，完全オーバラップにより生成されたグループの確率も計算できる．図 7 のトランスデューサはパターンの先頭である `stand` による遷移がすべて除かれており，新たにマッチの候補を生成することができなくなっている．実際にこのトランスデューサを利用して確率を計算した例を**図 8** に示す．時刻 t_3 までは図 6 と同じであるが，時刻 t_4 から図 5 に替わって図 7 のトランスデューサを使用したことにより，時刻 t_4 以降に状態 1 に到達する確率がすべて 0 になっている．これはグループに含まれないマッチ m_4 の確率が除かれたことを表しており，完全オーバラップによって生成されたグループの確率が計算できることを示している．

6. 評価実験

本章では，グループの特徴を評価するために行ったシミュレーションによる実験について述べる．まず，実験で使用する確率的データストリームについて説明する．本実験では，確率的データストリームは人工データを用いた．人工データは，曖昧性のない決定的なデータストリームを生成し，それに対して人工のノイズを加えることで作成した．決定的なデータストリームは単位時間 1，イベントのドメイン $V = \{a, b, c, d\}$ を使用して，合計 100,000 イベントのストリーム $\langle \alpha_1, \alpha_2, ..., \alpha_{100000} \rangle$ を生成した．人工的なノイズは $[0.1, 0.3]$ の範囲で各時刻ランダムに決定した．決定した値だけ各時刻のイベント α_t の生起確率から減算し，他のイベントの生起確率へ分配することで，イベントの確率分布 e_t を生成した．

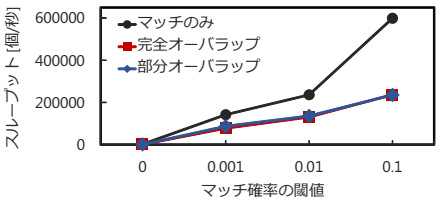

図 9 グループを用いる場合と用いない場合のスループットの比較

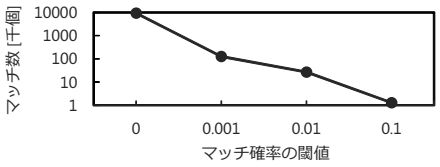

図 10 確率の閾値とマッチの生成数の関係

次に，実験で使用するパターンとウィンドウサイズ，確率の閾値について説明する．まず，パターンは `a b+ c` を用いた．前述の決定的なストリームに対してこのパターンを適用させると，長さ 3 から 30 程度のマッチが発見される．次に，ウィンドウサイズは 100 を，マッチ確率の閾値は $\{0, 0.001, 0.01, 0.1\}$ の 4 種類を使用した．実験ではマッチ確率の閾値を変化させることで，グループの生成における処理速度や生成されるグループの変化について調査した．

6.1 処理速度の評価

まず，グループを生成する際の処理速度を評価する実験を行った．この実験では与えるマッチ確率の閾値を変化させることで，グループを使用した際のスループットがどのように変化するかを測定した．実験結果を**図 9** に示す．なお，図 9 にはグループを使用しない場合のスループットも比較のために表示している．

図 9 から，グループを使用した場合スループットが約 0.4〜0.6 倍になっていることがわかる．このような結果となったのは，グループを生成する際の追加処理によって処理時間が増加したためである．また，マッチ確率の閾値が小さいほどスループット減少の割合も小さくなっているが，これはマッチの生成における処理時間が確率の閾値に大きく依存しているためである．**図 10** に示すように，確率の閾値が小さくなるほど生成されるマッチの数は多くなる．そのため，確率の閾値が小さいときは全体の処理時間に対してマッチ生成の占める割合が大きくなり，相対的にグループ生成によるスループット減少の割合が小さくなる．

また，図 9 から，完全オーバラップと部分オーバラップのどちらを使用してもスループットはほぼ変わらないことがわかる．これは，どちらのオーバラップを使用した場合でも一度に処理するグループの数がほぼ変わらないためである．本稿で提案したグループの生成方法では，どちらのオーバラップでも処理の際メモリ上に保持されるグループは数個しか存在しない．そのため，最終的に出力されるグ

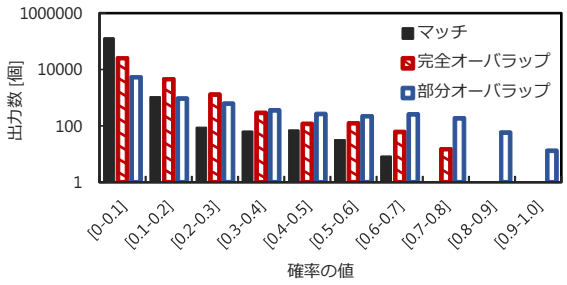

図 11 マッチ及びグループを確率の大きさで分類し，各確率の範囲で出力数を調査した結果

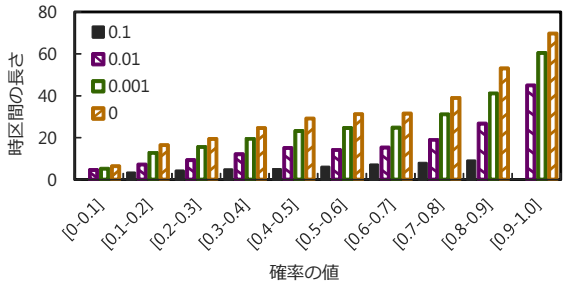

図 12 部分オーバラップを用いたグループを確率の大きさで分類し，各確率の範囲で時区間の長さの平均値を求めた結果

ループには差が生まれるが，スループットに関してはオーバラップの違いによる差は出なくなっている．

6.2 処理結果の評価

次に，生成されたグループを評価するための実験について説明する．この実験ではグループ確率の値に対して，グループの出力数がどのように分布しているかを調査した．実験結果を**図 11** に示す．図 11 中の $[0-0.1]$ などは確率の範囲を示しており，各棒グラフはこの範囲内の確率を持つグループないしマッチがいくつ存在しているかを表している．なお，どのマッチ確率の閾値を用いても同様の傾向が得られたため，図 11 には閾値 0.001 を使用した場合の結果のみを示している．

図 11 から，高い確率のマッチはほとんど出力されていないのに対し，グループは確率の高いものもある程度出力されていることがわかる．特に，確率 0.7 以上の範囲においてマッチは一つも出力されていないのに対し，完全オーバラップによるグループは 15 個，部分オーバラップによるグループは 258 個出力されている．この結果は複数のマッチをまとめることで確率の値を補正できたことを示しており，本手法の有効性を示している．

ただし，本稿の手法で生成されたグループの出力結果がすべて適当であるわけではない．**図 12** に，部分オーバラップによって生成されたグループの長さ $(t_e - t_s + 1)$ の平均をとり，グループ確率の値毎に表したものを示す．図 12 から，グループ確率の値が高いほどグループの長さも長くなっていることがわかる．この結果は，ひとまとめにするには離れ過ぎているマッチまでグループ化していることを示しており，部分オーバラップによるグループ化の条件が緩過ぎることを意味している．完全オーバラップによって生成されたグループであればこのような問題は発生しないが，完全オーバラップはグループ化の条件が厳し過ぎるため，図 1 や図 2 のような直感的にはひとまとめにしたいマッチもグループにまとめられないという問題がある．今後より適当なグループを生成するためには，完全と部分の中間に位置するようなオーバラップを考える必要がある．

7. おわりに

本稿では，確率的データストリームに対してパターンマッチングを行った際に生成される大量のマッチをグループにまとめる手法を提案した．グループにまとめるための方針として完全オーバラップと部分オーバラップの二つを定義し，それらを使用してグループを定義した．また，グループを生成するためのアルゴリズムや，トランスデューサを使用することでグループの確率を効率的に計算する手法を提案した．最後に提案手法を評価するための実験を行い，グループの特徴について考察した．

今後の課題としては，グループを構成するマッチをどのように決定するかについての再考や，グループをより効率的に生成する手法の開発などが挙げられる．また，現在はパターンは正規表現で与えるものとしているが，SQL 風の問合せなどより適当なパターンの表現方法についても考えていく必要がある．

謝辞 本研究の一部は，科件費（25280039）による．

参考文献

[1] Agrawal, J., Diao, Y., Gyllstrom, D. and Immerman, N.: Efficient Pattern Matching over Event Streams, *Proc. ACM SIGMOD*, pp. 147–160 (2008).
[2] Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J.: Models and Issues in Data Stream Systems, *Proc. ACM PODS*, pp. 1–16 (2002).
[3] Cugola, G. and Margara, A.: Processing Flows of Information: From Data Stream to Complex Event Processing, *ACM Comput. Surv.*, Vol. 44, No. 3, pp. 15:1–15:62 (2012).
[4] Hattori, Y. and Inoue, S.: A Large Scale Gathering System for Activity Data using Mobile Devices, *Journal of Information Processing*, Vol. 20, No. 1, pp. 177–184 (2012).
[5] Li, Z., Ge, T. and Chen, C. X.: ε-Matching: Event Processing over Noisy Sequences in Real Time, *Proc. ACM SIGMOD*, pp. 601–612 (2013).
[6] Nishimura, N. and Kawashima, H.: Accelerating CEP queries over Uncertain Data with Event Pruning and Link Aggregation, *WebDB Forum 2013* (2013).
[7] Ré, C., Letchner, J., Balazinksa, M. and Suciu, D.: Event Queries on Correlated Probabilistic Streams, *Proc. ACM SIGMOD*, pp. 715–728 (2008).
[8] HASC (Human Activity Sensing Consortium) ホームページ：http://www.hasc.jp/.