解　説

人はどういうものを似ていると思うのか？

時系列データの類似検索と順序保存照合問題 基般

成澤和志（東北大学）

時系列データの解析

2000年代初頭から時系列データと呼ばれるデータの解析手法に関する研究が盛んになってきた．株価や音声，心電図など以前からある，分かりやすいデータはもちろん，近年ではさまざまな小型センサの出現によりライフログやドライブレコーダなど，身近な行動でさえもが時系列データとして活用されるようになっている．このような時系列データの解析において必須ともいえる近似照合の新しい手法が順序保存照合である．順序保存照合問題（Order Preserving Pattern Matching Problem）は，2012年頃から文字列処理アルゴリズムをベースとして盛んに研究がされるようになってきている．本稿では，順序保存照合問題に関して，照合の特徴や現在の研究に関する状況などについて概説する．

▶ パターン照合問題

データを解析するといってもさまざまな手法がある．最も基本的な方法の1つが，パターン照合（Pattern Matching）と呼ばれるものである．一般的に分かりやすい言い方をすると，検索であるが本稿では照合という言葉を使うことにする．“パターン”という言葉は分野によってさまざまな意味で使われるが，ここでは検索したいものを指す．たとえば，`This is a pen.`という文字列の中に`is`という文字列があるかどうかを探すとき，`is`をパターンといい，`This is a pen.`をテキストという．このテキストでは，パターン`is`は2カ所に出現している．

データ分類やクラスタリング，頻出パターン発見，異常値検出などさまざまな解析を行う上で，パターン照合は最も基本的な技術である．この技術の精度や計算速度を向上させることは，すべての解析を向上させることにつながるため，文章などの文字列データに限らず，時系列データにおいてもパターン照合を高速に行うことが重要となる．

本稿では，時系列データは一定時間ごとに取得された数値を並べたものとする．そのため，時系列データでのパターン照合では，テキストは(10, 20, 10, 30, 30, 20, 10, 30, 20)，パターンは(11, 21, 11)というような形で与えられる．すぐに分かるだろうが，このテキストにパターンと完全に一致する部分は存在しない．しかし，テキストの先頭の(10, 20, 10)はパターンと非常に似ているといえるため，これをパターンと一致するものとして見つけても実用的にはよさそうである．

このように，時系列データは単語や文字列と違い，ノイズを含んでいる場合が多く，パターンと完全に同じものが存在する可能性は非常に低い．そのため，似ているものを探す，いわゆる近似照合を行う必要がある．ここで，問題となるのは，“似ている”とは何なのかということである．人が思う“似ている”を表現した順序保存照合について，既存の手法と比較しながら紹介していこう．

本題に入る前に1つ注意してほしいことがある．それは，順序保存照合問題はアルゴリズムの改良を重視した研究が多く，実用的な応用がほとんどないということである．そこで，読者にはぜひ，今抱えている問題の解決にこの手法が使えるのではないか

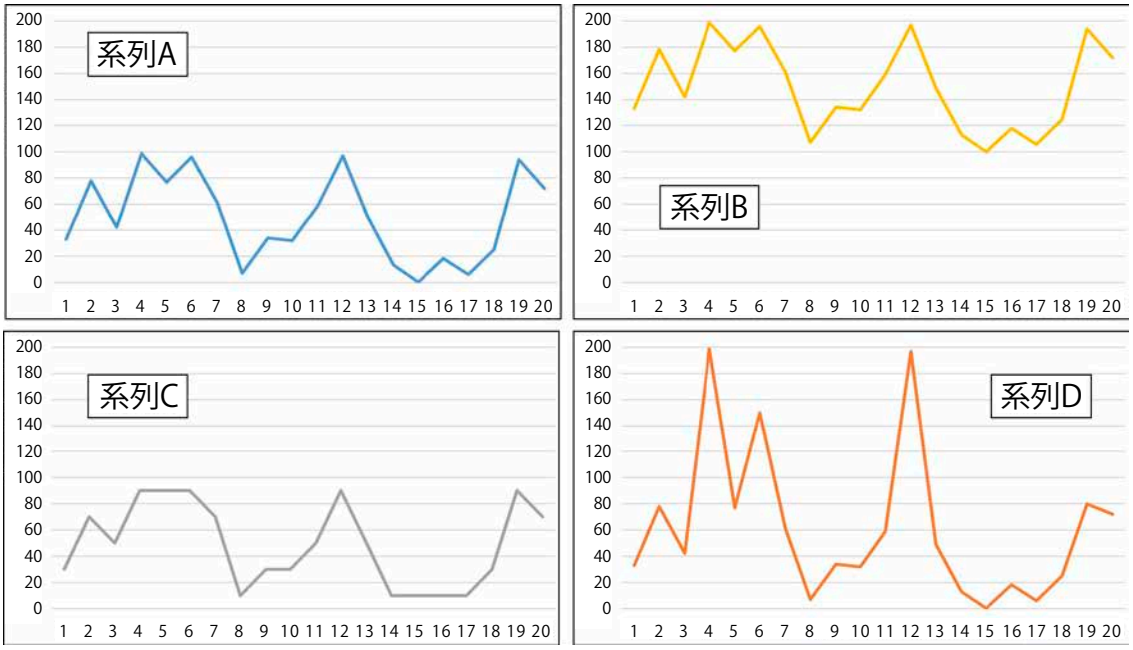

図 -1
時系列データ

という視点を持ちながら読んでほしい．先ほども述べたようにここ数年，時系列データを扱う分野は格段に広くなっている．教育や文学，農業，スポーツ，政治，料理など，一見関係がなさそうな分野であってもビッグデータというキーワードの下，時系列データ解析が必要になっているのではないだろうか．あまり難しく考えず，数値を並べれば時系列データだというくらい簡単な気持ちで多くの分野の人に読んでもらい，利用してもらえると幸いである．

なぜ順序保存照合なのか

▶ 時系列データでのパターン照合

時系列データにおいて，人（あなた）はどのようなものを“似ている”と思うのだろうか．実際に考えてもらいたい．図 -1 の 4 つの時系列データを見てほしい．

系列 A と“似ている”系列はどれか？

という問いに，あなたならどれを選ぶだろうか．どれも似ているといえば似ているし，似ていないといえば似ていないように見えるのではないだろうか．

時系列データにおいては，どのようなものを似ていると判断するかという基準が重要になる．順序保存照合問題に入る前に，まずはそれぞれの系列について，人間的な判断と，時系列データで用いられる既存の類似性指標を比較してみよう．

系列 A と系列 B

おそらく，系列 A と似ている系列として系列 B を選ぶ人が最も多いのではないだろうか．答えを言うと，系列 B は系列 A を上方向にシフトしただけの系列である．そのため，人間的な感覚では，3 つの中では一番似ているといえるだろう．しかし，各時間（横軸）同士での値を比べてみると大きな差がある．そのため，各時間での差分を基にした類似性指標であるユークリッド距離

$$D(\mathbf{x},\mathbf{y}) = \sqrt{\Sigma_{i=1}^{n}(x_i - y_i)^2}$$

や，ユークリッド距離を長さが異なるデータでも対応できるように拡張した DTW（Dynamic Time Warping）を用いると似ていないという判断になる．

系列 A と系列 C

系列 C は，系列 B と比べるとあまり似ていないようにも見えるが，系列 A のように各値に対して大きな差はなさそうであり，これも似ているという判断をしてもよさそうな気もする．系列 C は系列 A に対して，SAX（Symbolic Aggregate approXimation）という操作を行ったものである．この操作は，たとえば，0 から 200 までの値を 20 刻みで分割し，0 か

手法	系列 B	系列 C	系列 D
ユークリッド距離 （値が小さいほど似ている）	447.2	**29.9**	152.03
DTW （値が小さいほど似ている）	2000	**118**	231
相関係数 （値が 1 に近いほど似ている）	**1.000**	0.979	0.886
SAX （似ている：○，似ていない：×）	×	○	×

表 -1　系列 A との類似性

ら 20 までにある値はすべて 10 とする，というように区間ごとに代表する文字（図では平均値）で表現したものである．

この手法自体は，類似性指標ではないが，時系列データにおける近似照合においてよく利用される．このメリットは，数値列を文字列に変換することで，通常の文字列処理アルゴリズムが適用できるため高速な処理が可能となることである．ただし，区間の区切り方や，区間の端にある値などによっては，一見似ているようなものでも似ていないと判断してしまうことがある．

系列 A と系列 D

系列 D は，なんとなくは似ているけど数カ所が飛び出ているのが気になる，という人が多いのではないだろうか．系列 D は系列 A の各値に対して多少のノイズを入れ，いくつかの値だけ異常値として変化させた系列である．系列 A とほとんど同じ高さにあるため，ユークリッド距離で判断すると系列 B よりも似ていることになるが，SAX による判断では異常値のせいで似ていないと判断されてしまう．

このような系列に対しては，相関係数を用いることがある．相関係数は 2 つの系列の関連性を見るための指標であり，1 に近い方が関連があるという判断になる．

結局どれが似ているの？

さて，これまでの議論を元に改めて考えてみるとどうだろう．系列 A と似ている系列はどれだろうか．参考のために，これまで出てきた 4 つの類似性指標による評価の結果を**表 -1** にまとめた．それぞれ太字にしているものがその指標における最も似ているというものになる．これを見ると系列 C や系列 B が似ていると判断する指標が多い．

この判断は正しいのだろうか．

答えは，状況によって変わる．たとえば，音楽データとして考えると系列 A と系列 B は転調のような関係を表すことになるため似ていると判断したいだろう．株価などのような上下の動きが重要なデータとして考えると系列 C のように大まかな判断で似ているといってもよいだろう．センサデータなどのようにノイズや異常値が入りやすいデータとして考えると系列 D のようなデータを似ていると判断したい場合もあるだろう．

では，我々は似ている系列を探すときに，どのようなものを似ているとするかという基準を明確に持っていなければ探すことはできないのだろうか．また，3 つの系列すべてを似ているということはできないのだろうか．

その答えとなるのが順序保存照合である．

順序保存照合問題

順序保存照合は，**順序同型**（order isomorphic）である系列を似ていると判断し，時系列データでパターン照合を行うことである．順序同型とは，長さ n の 2 つの系列 $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ において，任意の i, j に対して $x_i \leq x_j \Leftrightarrow y_i \leq y_j$ が成り立つことをいう．

別の表現をすると，系列内の各値を順位で表現した系列が同じになるとき順序同型であるという．たとえば，

$$\mathbf{x} = (85, 97, 74, 79, 43)$$
$$\mathbf{y} = (69, 83, 61, 64, 59)$$

という 2 つの系列について考えてみる．それぞれの系列を順位を使って表現すると

$$code\ (\mathbf{x}) = (4, 5, 2, 3, 1)$$
$$code\ (\mathbf{y}) = (4, 5, 2, 3, 1)$$

となり，順位による系列が一致する．そのため，$\mathbf{x}$ と $\mathbf{y}$ は順序同型であるといえる．

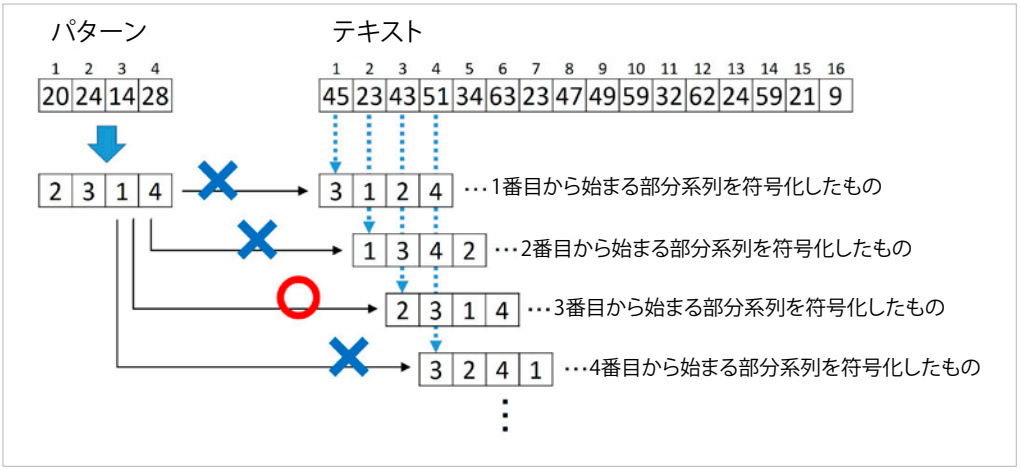

図 -2　順序保存照合のナイーブな方法

▶ どう似ているのか

順序保存照合では，系列の各値の**順位の列**が同じものを似ていると判断している．それは系列内での値の上がり方や下がり方などが相対的に似ている，つまり時系列データとしての**形状が大まかに**似ているといえるだろう．そのため，系列全体が縦方向にシフトしたものはもちろん，多少のノイズや異常値が入っていても似ていると判断することができる．順序同型であるかを基準にみると，図 -1 の 4 つの系列はすべて似ていると判断することができる．

▶ 長所と短所

順序保存照合を行う特徴としては以下のようなものがある．

1. シフト，ノイズ，異常値に強い
2. 思いがけないものを似ていると判断する
3. 閾値がいらない
4. 計算が速い

1. に関してはこれまで述べてきた通りで，順序保存照合を行う最大のメリットであるといえるだろう．ただし，ノイズや異常値に関してはあくまでも順位を変えない場合に限るところがデメリットであるともいえる．

2. に関しては 1. にも関連したことであるが，実際の値ではなく相対的な関係によって似ていると判断するため，人間が目で見て分かりにくいものを見つけてくれる可能性もある．しかし，それは同時に，人間からしたら似ていないものでも，似ていると判断してしまうというデメリットとなる可能性もある．

3. の閾値がいらないという点は，実用上かなりメリットとなり得る．どのくらいの数値があれば似ていると判断するか，閾値を決める上で迷ったことがある人は多いと思う．

4. に関してはこれまで紹介した類似性指標でも，ユークリッド距離などは十分に高速である．しかし，DTW や相関係数などは計算に比較的時間がかかる．大規模なデータを解析する上では，わずかな計算速度の向上が全体として大きな差となる．

次からは順序保存照合問題に対する高速なアルゴリズムについて，簡単に紹介していきたい．

高速なアルゴリズム

ここからは，計算量に関する話をするため，順序保存照合問題において，パターン系列の長さを m，テキスト系列の長さを n とする．

▶ ナイーブな方法

順序同型かどうかを調べるための方法としてすぐに思いつくのは，パターンとテキストの部分系列をソートして順位を求めるという方法である．図 -2 にナイーブな方法（すぐに思いつく最も簡単な方法）の概要を示す．この方法では，まずパターンを順位

の系列に変換する．この操作には一般的なソートを用いれば，$O(m \log m)$ 時間で行うことができる．次に，テキストの先頭からパターン長 m（図では長さ 4）と同じ長さのテキストの部分系列を順位の系列に変換する．この操作も 1 つの部分系列に対して，やはり $O(m \log m)$ 時間かかる．その後，変換したパターンと一致しているかを判定する．この判定には $O(m)$ 時間かかる．この操作をテキストの最後まで行うと，全部で $n-m+1$ 回行うことになる．そのため，全体で $O(nm \log m)$ 時間が必要となる．順序保存照合に関するアルゴリズムでは，これが基準であり，これより高速なアルゴリズムの開発が要求される．

▶ 順序保存照合問題特有の難しさ

順序保存照合において通常の文字列のパターン照合と大きく異なるところは，順序同型を考慮する点である．これまでの説明では，系列内の各値を順位で表現した系列に変換した．このような順序同型を判定するために行う変換のことを**順序保存符号化**，または単に**符号化**という．先ほどのような順位に変換する符号化方法は，自然表現（natural representation）と呼ばれている．この他にもアルゴリズムごとに適した符号化方法が提案されている．

順序保存照合問題を考える上で最も難しい点が，符号化による影響である．図 -2 のテキストの部分系列を符号化したものを見てほしい．2 番目から始まる符号化部分系列（1, 3, 4, 2）と，3 番目から始まる符号化部分系列（2, 3, 1, 4）は，元のテキスト上で 3 つ分が同じであるにもかかわらず，まったく違うものになっている．文字列照合アルゴリズムでは，このような問題は起きない．それどころか，ずらしたときに同じ部分があるという性質を効率的に使うことで，比較回数を減らすなどの工夫がされている．順序保存照合問題では，この問題のため文字列照合アルゴリズムが単純には適用できない場合が多い．

▶ 問題解決の方針

文字列照合アルゴリズムが単純に適用できないとはいえ，順序保存照合問題に対する研究は，やはり文字列照合アルゴリズムを基盤としたものが多く，大きな方針も文字列でのパターン照合問題と同じである．パターン照合問題における方針は大きく 3 つある．

1. 逐次アルゴリズムを用いる
2. 索引構造を用いる
3. フィルタを用いる

順序保存照合では，それぞれの方針やアルゴリズムごとに符号化の方法も提案されている．それぞれの方針ごとに現在の研究状況について，符号化方法とあわせて紹介していく．

逐次アルゴリズム

逐次アルゴリズムは，テキストを先頭もしくは後ろから順に見ていきパターンを探していくアルゴリズムである．先ほどのナイーブな手法もこの 1 つである．文字列処理では，ボイヤームーア法やクヌースモリスプラット法（KMP 法）などのアルゴリズムが古くから提案されている．ナイーブな手法ではパターンと比較する部分を 1 つずつずらしながらテキストと比較していたが，これらの手法では，パターンに前処理を行い，ずらす量を複数にすることで高速化を実現している．

しかしながら，図 -2 の例からも分かるように自然表現ではパターンをずらした場合にすべての値が変わってしまうことがあるため，そのまま適用することが難しい．そのため，自然表現ではない別の符号化を行うことでこのような難しさを緩和している．たとえば，KMP 法を基にした順序保存照合アルゴリズムでは接頭辞表現（prefix representation）と呼ばれる符号化を用いている．この符号化では，系列の i 番目の値を符号化するときに，系列全体ではなく，先頭から i 番目までででの i 番目の値の順位を用いる．たとえば，$\mathbf{x} = (21, 34, 19, 47, 38)$ を接頭辞表現を用いて符号化すると $code_P(\mathbf{x}) = (1, 2, 1, 4, 4)$ となる．2 番目の 34 という値は，全体でみれば 3 番目の順位であるが，この表現では先頭から 2 番目の間で考えるため 2 となる．この符号化の特徴は，右側に値を追加（または削除）したとしてもそれより

も左側にあるほかの値に影響しないところである．

順序保存照合問題では，このように符号化の方法を工夫することで既存の文字列アルゴリズムを適用させている．接頭辞表現を用いた場合 $O(n \log m)$ 時間で，近傍表現（nearest neighbor representation）と呼ばれる符号化を用いれば $O(n + m \log m)$ 時間で照合できる手法が提案されている [1]．

索引構造

1つのテキストに対して違うパターンを何度も探したり，複数のテキストから複数のパターンを探すというような場合，逐次アルゴリズムでは1つのパターンに対して毎回テキストを最初から最後まで探すことは非効率である．そのため，あらかじめテキストに対する索引構造を作っておくことで，1つのパターンに対する照合時間を高速化することができる．多くは，$O(n)$ 時間で索引構造を構築し，1つのパターンに対する照合を $O(m)$ 時間で行うものが多い．

文字列照合においては，接尾辞木（Suffix Tree）が最も有名な索引構造の1つである．順序保存照合に対しても接尾辞木を基にした順序保存不完全接尾辞木（order-preserving incomplete suffix tree）や順序保存完全接尾辞木（order-preserving complete suffix tree）と呼ばれる索引構造が提案されている [2]．これらに対しては，$\alpha\beta$ 表現や計数表現（counting representation）と呼ばれる符号化が用いられている．順序保存不完全接尾辞木を構築するには $O(n \log \log n)$ 時間，順序保存完全接尾辞木を構築するには $O(\frac{n \log n}{\log \log n})$ 時間がかかる．

文字列処理においては，接尾辞木よりも実用的で高速かつ省メモリな方法として接尾辞配列（suffix array）というデータ構造がある．しかしながら，順序保存照合においては接尾辞配列に対応するものは現在のところ，残念ながら存在しない．正確に言うと，高速な直接構築法が存在しない．接尾辞配列は，接尾辞木を作ることができれば，そこから作ることができる．しかし，それでは時間もメモリもかかるため接尾辞配列としての良さが失われてしまう．

接尾辞木や接尾辞配列を高速に構築することが難しい理由は，やはり符号化の影響である．これらの索引構造では，すべての接尾辞を考慮しなければならない．つまり，任意の区間に対してある値の符号化されたものを高速に求めなければならない．たとえば，系列（5, 20, 15, 30, 10）において，15という値は系列全体で見れば順位は3であるが，[1 : 4] の区間で見れば2であり，[2 : 4] の区間で見れば1であるということを高速に求めなければならない．このような問題に対しては，順序統計量木やY高速木と呼ばれるデータ構造などを利用している．計算量に複雑な log が入っているのはこれらのデータ構造の構築等に必要なためである．また，我々のグループではウェーブレット木と呼ばれるデータ構造を用いた $O(\log \sigma)$ 時間（σ は文字の種類数）での符号化方法も提案している [3]．

フィルタリング

順序保存照合で最もボトルネックになっているのは，順序同型かをチェックすることである．そのため，フィルタを用いてだいたい近いものを候補として高速に検索し，見つかった候補に対して本当に正解かどうかをチェックしていくことで，全体として高速に検索を行う手法が提案されている．

代表的な方法として，系列を次の値と比べて大きい場合は1，小さい場合は0としたバイナリ系列に変換したものをフィルタとして利用することで解の候補を絞り，計算時間がかかる順序同型の判定回数を減らす方法がある [4]．たとえば，図-2の例で考えると，パターンは010，テキストは100101000101011と変換できる．このバイナリ系列で照合を行うと，候補位置は3, 5, 9, 11の4カ所になる．そのため，本来は13回の順序同型判定を行わなければならなかったのが，たった4回の判定でよくなる．

バイナリへの変換は単純に隣の数字との大小比較だけなので，系列を1度なぞるだけで可能である．また，バイナリに変換した後の照合でも，bit演算やSIMD演算などを利用した照合アルゴリズムを用いることができるため，通常の照合アルゴリズムと比べても，実験的に高速に候補を見つけることが

できる．

フィルタリングを用いた方法では，逐次アルゴリズムをベースとしているため，計算量としてはそれらとあまり変わらない．しかしながら，ボトルネックになっている順序保存符号化や順序同型の判定などを，計算時間のかからないフィルタを用いて減らすことで，大幅な高速化を行っている．さらに，bit 演算や SIMD 演算などを利用すれば，パターン長の制限という問題があるものの，劇的に計算を速くすることができる．

応用的な解析のために

これまでは，順序同型を用いた近似パターン照合に対する手法を紹介してきた．しかし，順序保存照合に限らず，パターン照合ではパターンが長くなれば長くなるほど，照合が起こりにくいという欠点がある．たとえば，Web 検索を行うときに，文章を入れたら 1 件もヒットせず，仕方なく単語に区切って検索しなおしたことがある人は多いだろう．筆者の経験上であるが，順序保存照合の場合，長さが 10 を超えるようなものに関しては，一致するような系列が存在しなくなる場合が多くなる．これはパターンが長くなると，順序を変えてしまうようなノイズや異常値が含まれる可能性が高くなるためである．そのため，系列中の k 個までは不一致を許した順序同型なども提案されている．これによって，順序を変えてしまうようなノイズや異常値に対しても対応することが可能となる．

また，複雑な解析を行うためにはパターン照合だけでは十分ではない．このような問題に対して，分類問題などに利用できるよう順序同型を用いたカーネルを提案している [5]．このカーネルを用いるとサポートベクターマシンなどの既存の分類手法に適用できるだけでなく，これまで似ているかどうかだけだった順序同型を類似度として利用することができる．これによって，閾値がいらないというメリットはなくなるが既存の手法との親和性は高くなるため利用しやすくなる．さらに，文字列処理においてもよく利用される 2 つの文字列の最長共通部分列（Longest Common Subsequence）を順序同型においても定義し，高速に計算するという研究も行っている [6]．

冒頭にも述べたが，現在，順序保存照合問題に対する研究はアルゴリズム的側面からのアプローチがほとんどであり，実用的な応用はほとんどない．そのため，多くの分野の人に知ってもらい，どのような状況で，どのように似ているものを探すことが有用であるかを教えてもらえると幸いである．

多くのアルゴリズムを考える研究者にとって，自分の技術がどのように使われるか分からず，応用先を思いつかないということは多いだろう．少なくとも筆者はそうである．筆者が初めて読んだ時系列データの論文も，物体の輪郭の座標を時系列データとして表現し近似照合を行うことで，物体を識別するというものであり，筆者の想像していた時系列データの外にある使い方であった．このように，時系列データといっても時間に関連している必要はなく，ただデータを並べただけのものでも時系列データと見なすこともできるだろう．一見関係がないと思えるようなところにこそ，素晴らしい結果が得られる要素が隠れているかもしれない．

参考文献

1) Kim, J., Eades, P., Fleischer, R., Hong, S. H., Iliopoulos, C. S., Park, K., Puglisi, S. J. and Tokuyama, T. : Order-preserving Matching, *The-oretical Computer Science*, Vol.525, No.13, pp.68-79 (2014).
2) Crochemore, M., Iliopoulos, C. S., Kociumaka, T., Kubica, M., Langiu, A., Pissis, S. P., Radoszewski, J., Rytter, W. and Walen, T. : Order-preserving Incomplete Suffix Trees and Order-preserving Indexes, In *SPIRE*, pp.84-95 (2013).
3) 佐藤雄介，成澤和志，篠原　歩：順序保存符号化 n-gram の高速な出現頻度計算手法，情報処理学会研究報告アルゴリズム 2015-AL-151(1), pp.1-8 (2015).
4) Chhabra, T., Giaquinta, E. and Tarhio, J. : Filtration Algorithms for Approximate Order-preserving Matching, In *SPIRE*, pp.177-187 (2015).
5) 柏葉祐輝，成澤和志，篠原　歩：順序保存カーネルを用いた時系列データ分類，人工知能基本問題研究会，第 B5 巻，pp.1-6 (2016).
6) 栗原理聡，成澤和志，篠原　歩：順序同型な部分系列を用いた数値列に対する最長共通部分列問題，電子情報通信学会コンピュテーション研究会，pp.11-20 (2016).

（2016 年 1 月 12 日受付）

成澤和志（正会員）　narisawa@ecei.tohoku.ac.jp

2010 年九州大学大学院システム情報科学府情報理学専攻博士課程修了．博士（理学）．同年より東北大学大学院情報科学研究科助教．文字列処理などに関する研究に従事．