スマートハウスのセンサデータに対する SAX を利用したイベント検出の検討

大西 史花† 渡辺 知恵美‡

†‡お茶の水女子大学理学部 〒112-8610 東京都文京区大塚 2-1-1

E-mail: †‡{onishi.ayaka, chiemi}@is.ocha.ac.jp

あらまし 近年，ネットワークやセンサ技術が普及してきたことにより，様々なセンサが身の回りで使われるようになってきている．この動向に注目し，Ocha House プロジェクトでは，近未来における新しい住宅のかたちを提案している．我々は，住宅に設置されたセンサの情報に着目し，他のアプリケーションの基盤となるような，住宅の情報を格納し効率的にイベントが検出できる DB システムを提案する．我々は，高速な検索のために Symbolic Aggregate Approximation (SAX) を用いて時系列データの文字列化を行い，加えて接尾辞木を適用し索引を作成する．本稿では，作成した SAX 索引を用いた，住宅内でのセンサデータからのイベント検索を行うための問合せ方法を検討する．

キーワード Symbolic Aggregate approximation，イベント検出，センサデータ

Investigations for Event Detections in a Smart House from Archived Sensor Data Using SAX

Ayaka ONISHI† Chiemi WATANABE‡

†‡Ochanomizu University Otsuka 2-1-1,Bunkyo-Ku, Tokyo 112-8610 Japan

E-mail: †‡{onishi.ayaka, chiemi}@is.ocha.ac.jp

1. はじめに

近年，ネットワークやセンサ技術が普及してきたことにより，それらの技術を利用する場は，研究所や工場などの専門家が作業するための場所だけでなく，一般オフィスや店舗などにも広がっており，様々なセンサが身の回りで使われるようになってきている．この動向に注目し，我々が参加している Ocha House プロジェクトでは，進歩の早い情報技術を家に組み込むために，センサ，ネットワーク等の変化に柔軟に対応できる，近未来における新しい住宅のかたちを提案している[9]．このプロジェクト内で提案された手法やシステムを Ocha House と呼ばれる実験住宅に実装することで，一般の生活者が必要とするユビキタスコンピューティングアプリケーションの実証実験を行っている[8]．図 1 は Ocha House の外観である．

我々は，特にセンサ情報に着目し，他のアプリケーションの基盤となるような，住宅の情報を格納し効率的に利用することが出来るデータベースシステムの提案と実装を行っている．ここでいう住宅の情報とは，例えばドアに設置した加速度センサの値から分かるドアの開閉時間や，部屋に設置した照度センサの値から分かる部屋の照明の点灯時間等である．これらの情報を取得出来れば，入退室管理を行うことや，1 日の電気代を算出することなどが出来ると考えられる．

図 1：実験住宅 Ocha House の外観

このようなシステムの実現を考えた時，センサから送信されるデータを格納するデータベースには，常に新しい情報が到着し続けるため，短時間で大量のデータが生成されることになる．この大量のセンサデータ

を格納した場合，そのデータに対して検索や分析を行う場合に時間がかかりすぎる問題が生じる．この問題を解決するために，我々はセンサデータに対して高速な問合せが可能となるような索引の実装を行った[7]．その索引では，Lin らが提案した Symbolic Aggregate approximation (SAX) [1] という手法を用いているため，以後“SAX 索引”と呼ぶ．

SAX 索引を用いた問合せでは，ユーザが指定した時系列データをクエリとし，SAX によって文字列化されたクエリ文字列と一致または類似するセンサデータ中の部分文字列を検索する．これによって例えばドアの向きを長時間測定しているセンサデータからドアの開閉というイベントが行われた部分のみを検索できる．また SAX を用いたモチーフ検索[11]や外れ値検索[10]などの検索手法なども実現されており，音量センサデータから家庭内での生活パターンをマイニングしたり格納したセンサデータを有効に扱うなど，多様な分析に適用させることができる．

センサデータは長時間とり続けているため，SAX 索引は非常に長い文字列となる．そこで我々は，SAX 索引に対して，長い文字列から部分文字列を高速に検索する手法として 接尾辞木を適用することとした．しかしながら，接尾辞木はクエリ文字列に対して完全一致する部分文字列を高速に検索するための索引であり，SAX で実現できるような類似度検索にはそのままでは適用できない．ドアの開閉のようなイベント検索を行うにはセンサデータの細かな違いなどを吸収するために類似検索が必要である．また動作のスピードを吸収できるような柔軟性も持ち合わせなければイベント検出に対応させるのは難しい．

そこで我々は SAX 索引の文字列に対し，編集距離を用いた類似検索を適用することを提案する．SAX における近似距離の定義に基づいて編集距離の編集コスト定義を行い，トライ木 による編集距離検索手法である Error-tolerant Recognition アルゴリズム[3]を用いて高速な類似度検索を実現した．

本稿では，第 2 節で SAX について，第 3 節で SAX 索引への接尾辞木の付与について，第 4 節で蓄積されたセンサデータへの索引の付与について，第 5 節で編集距離を用いた類似検索について，第 6 節でイベント検索に対する検討について，第 7 項で関連研究について述べる．そして第 8 節でまとめと今後の課題を提示する．

2. SAX

SAX とは，時系列データの表現手法の 1 つで，時系列データを文字列に量子化する手法である．パラメータ値として，あらかじめ文字列長 w（もしくは 1 つの文字に変換するデータ数）と文字種類 a の 2 つを決定する．これらの値はデータに依存するため実験的に求める必要がある．

SAX による時系列データの文字列化の手順を以下と図 2 によって示す．図 2 では破線が SAX を適用するセンサデータである．

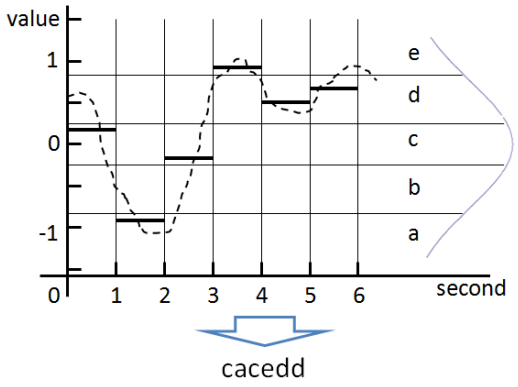


図 2：SAX による時系列データの文字列化

(1) 時間軸を等間隔に区分（縦実線）．
(2) 区間ごとの平均値を算出（横太線）．
(3) 正規分布に従って， a ，b ， c …とアルファベットを割り振り，正規分布の各面積が等しくなるような分割線（横実線）を定める．
(4) 2 で求めた平均値を 3 で定めた分割線に従って文字に変換（下部）．

SAX の利点として，実装が容易であること，後述するように近似距離が実距離の下界を抑えることが保障されていること，文字列用に定義された手法等も適用出来るようになること等が挙げられる．

本研究では以上の手法を実装し，センサデータを文字列化した SAX 索引を作成している[1]．なお，作成された SAX 索引に対して問合せを行う場合は，クエリも文字列化する必要がある．問合せの結果として該当した文字列の位置から時刻を算出すれば，時刻の取得も可能である．

我々が SAX 索引に対して期待することは類似検索である．類似検索とは，2 つのデータ Q と C を比較した際の距離がしきい値以内のものを一致とみなす検索のことで，時系列データにおいては距離尺度としてユークリッド距離（図 3 の(A)）等を使用することが多い．それに対して SAX での文字列（$\hat{Q}$, $\hat{C}$）間の距離は図 3 の式(B)で示されるように文字間の距離 dist($\hat{q}$, $\hat{c}$)の二乗の和で定義されている．n は元データのデータ数である．

$$(A)\quad \text{元データ}\quad D(Q,C)\equiv\sqrt{\sum_{i=1}^{n}(q_i-c_i)^2}$$

$$(B)\quad SAX\text{表現}\quad MINDIST(\hat{Q},\hat{C})\equiv\sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(dist(\hat{q}_i,\hat{c}_i))^2}$$

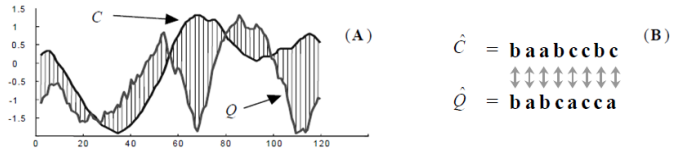
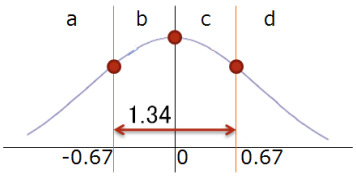


図 3：SAX における文字列間の距離の定義と概念図

文字間の距離 dist(q̂, ĉ) は文字 q̂ がとりうる値の最大値と ĉ がとりうる値の最小値によって定義される．例えば図 4 の左側は時系列のデータの値の分布を示している．SAX 索引は量子化の前に値の正規化を行うため，正規分布に従っている．この図では時系列の値を 4 種類のアルファベットに分割している．ここで dist(a,d) は a のとりうる最大値 -0.67 と d のとりうる最小値 0.67 の差となるので 1.34 となる．SAX は文字間の距離の表（図 4 の例の場合は図中右の表）をあらかじめ用意しておくことができるため，一度距離表を作成すれば文字列間の距離を求める際に毎回文字間の距離を測る必要はない．



	a	b	c	d
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

図 4：SAX における文字間の距離の図示と表

このような距離の定義により，SAX によって求められる文字列間の距離（つまり元の時系列データの近似距離）は元の時系列データの距離の下界を抑えていることが保障されている．

3. SAX 索引への接尾辞木の付与

SAX 索引を用いた問合せでは，SAX によって文字列化されたクエリ文字列と一致または類似する SAX 索引の部分文字列を検索する．しかし，SAX によって文字列化された SAX 索引は長大であり，これに対しシーケンシャルスキャンで検索を行うことは効率的ではない．そこで，我々は高速な文字列検索を可能にするデータ構造として接尾辞木を SAX 索引へ付与する．

接尾辞木とは，与えられた文字列の接尾部を木構造で表すデータ構造のことで，木構造の枝には文字列が対応し，根から葉までの経路ごとにそれぞれひとつの接尾辞が対応する．葉ノードは対応する接尾辞が元テキストの何文字目に存在しているかを示している．例として babac という SAX 索引データに対する接尾辞木の構築を示す．文字列 babac の接尾辞は，「babac」，「abac」，「bac」，「ac」，「c」の 5 つである．

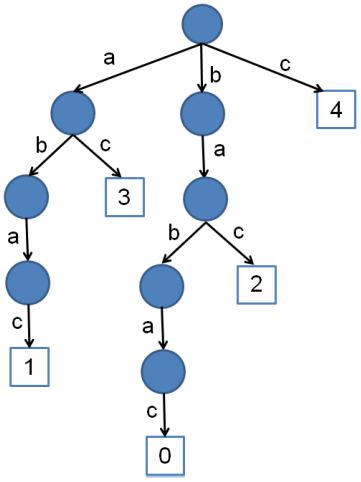


図 5：文字列 babac に対する接尾辞木の構築

検索を行う際には接尾辞木をルートから辿っていく．例えば図 5 の木に対して“ac”を検索したい場合，根から比較しつつ辿っていくと葉ノード 3 に辿りつく．結果“ac”は元テキストの先頭から 3 番目の位置にあることが分かる．

接尾辞木の効果を確かめるため，2日分の SAX 索引（2230000文字）に対して検索を行った際の速度を比較した（表1）．結果，接尾辞木を適用した索引のほうが高速な検索が行えることを確認した．

表1：速度比較実験結果

接尾辞木あり	接尾辞木なし（シーケンシャルスキャン）
3ms	267ms

4. 蓄積されたセンサデータへの索引の付与

センサデータは常にデータが取得されるため，基本的には終わりがない．そのため，センサデータに対する SAX 索引の生成と接尾辞木の付与には以下の問題が生じる

(1) SAX 索引を作成するためには正規化が必要であるため必要なデータがそろってから分布を取る必要があるが，センサデータの場合は常にデータが蓄積されるため，正確な正規化を行うのは難しい．

(2) 接尾辞木は接尾辞パターンの木構造であるが，センサデータが到着するごとに接尾辞が追加されるためツリーが際限なく深くなってしまう．

1 つ目の問題に対しては，現時点ではある程度の期

間とったデータに対して統計を取り正規化のためのパラメータである平均値と分散を求め，それ以降はその値を使用することとした．ただしこの手法に関してはあくまで現時点での対応であり，今後の課題として対応が必要である．特にアルファベットに割り当てられる値が正規分布に基づいているため，統計を取った後に大幅な時系列の変化が起こった場合には SAX での近似距離が下界を抑えられるという特徴が保障できない可能性があるため対応が必要である．

2 つ目の問題に関しては，接尾辞木で扱う木の深さに制限を設けることによって対応することができる．元の時系列に対して滑走窓を設けて部分時系列を検索する場合と同様の対応になると考えられ，SAX[1]においても次元（文字列長）の削減方法として記述されているため非常に長いイベントを抽出したい場合以外には適切な手法と言える．

5. イベント検索に対する検討

本節では，SAX 索引に対する接尾辞木を使用した住宅のイベント検索について検討する．課題点はセンサデータのバリエーションへの対応法である．例えば「ドアの開閉」についてのセンサデータには，ドアの開閉スピードや開け幅等の個人差，開けっ放しにする等の状況の差などが含まれる．これらセンサデータのバリエーションに対して，我々はユーザの要求，例えば単に「ドアが開閉した」時刻を知りたい，「ドアを開けっ放し」にした回数を知りたい，「○○さんがドアを開閉した」時刻を知りたいなどの要求に柔軟に対応できるような検索システムを構築したいと考えている．しかし，そのために解決しなければならない課題が 2 点ある．

まず 1 点目は，接尾辞木そのままでは類似検索が行えないということである．SAX では比較する 2 つの文字列の近似距離から類似検索を行うことができる．しかしながら我々は長い SAX 索引から部分文字列を検索するために接尾辞木を適用している．基本的には接尾辞木は完全一致する部分文字列を検索するデータ構造なのでこのままでは類似検索を行うことが出来ない．例えば図 6 の上部は 1 人の被験者が同一のドアを 1 回開閉した時のセンサデータの値の挙動 2 つ（A, B）と，開けっ放しにしたあと閉めた時のセンサデータの値の挙動（C）で，下部はその値を SAX 索引（w=53, a=8）に変換したものである．なお，見やすさのため以降の図では SAX 索引はランレングス圧縮表記となっている．このとき例えばクエリとして時系列 Q の検索を行う場合，1 文字でも違えば一致と判断されない．

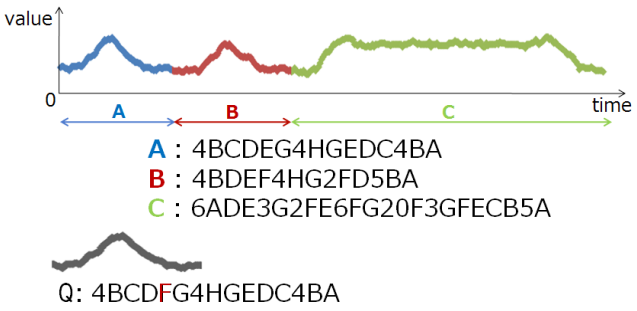


図 6：ドアの開閉による実際のセンサ値の挙動

2 点目はドアの開閉スピードの違いを吸収できないということである．例えば図 7 の上部は普通にドアを開閉した時とゆっくりドアを開閉した時のセンサ値の挙動であり，図 7 の下部はその値を SAX 索引（w=41, a=8）に変換したものである．時系列 A と時系列 D は，同じ「ドアの開閉」イベントであるが，そのスピードの差により SAX 索引に変換したときの文字列に若干の違いが生じている．このような違いも現状では対応できない．

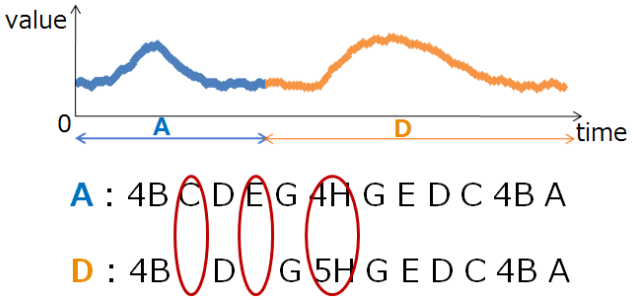


図 7：ドアの開閉スピードの違い

6. 編集距離を用いた類似検索

前節で述べたような状況や個人差を考慮した検索を行うため，課題への対応策として我々は文字列間の距離の定義として編集距離を採用する．

編集距離は，2 つの文字列 Q と C があったとき，文字列 Q を文字列 C に変換するのに必要なコスト（置換，削除，挿入）の最小合計値を文字列 Q と文字列 C の距離 dist(Q,C) として計算する．例えば文字列 Q="AFC"，文字列 C="ABCE" とした時，文字列 Q の 2 文字目 F を B に置換，4 文字目に E を挿入すると文字列 C と一致する．置換および挿入にかかるコストをそれぞれ 1 と定義すると，dist(Q,C)=2 となる．

編集距離を用いた文字列間の類似度を使って部分類似文字列を接尾辞木により検索する手法は Error-tolerant Recognition アルゴリズム[3]によって提案されており，これを採用することによって接尾辞木

を用いた類似検索を行うことができる．

また，前節で述べた 2 点目の問題を考慮する際は，動的タイムワーピング (Dynamic Time Warping: DTW) 距離を用いることで対応することができると考えられる．DTW とは時系列の距離測度の一つであり，時間軸上のずれやスケーリングを吸収した距離を求めることができる．Chen らの論文[5]において DTW を編集距離のコスト値に当てはめる記述があり，これを適用することによって DTW に従った編集距離を求めることができる．

以下，6.1 節では SAX 索引における編集距離の定義について，6.2 節では「ドアの開閉」イベント検索の実験について述べる．

6.1 SAX 索引における編集距離の定義

編集距離における編集コストは DTW に従って以下のように定義する．

- 置換コスト

  置換コストは，2 節で説明した SAX における文字間の距離 dist に従う．

- ギャップ（挿入，削除）コスト

  挿入コストは挿入する文字 x と挿入する位置の一つ前にある文字 y との距離とする．つまり"ABC"と"AC"を比較する際，"AC"の A と C の間に B を挿入するコストは dist(A,B)とする．削除コストに関しても同様の考え方を適用し，削除する文字 x と削除する文字の一つ前にある文字 y との距離とする．

この考え方を元に，SAX 索引 Q と C の各々i 文字目までと j 文字目までの文字列の距離は以下のように定義できる．

$$P(Q_i,C_j) = \min\{P(Q_{i-1},C_{j-1})+dist(q_i,c_j), \quad \#置換$$
$$P(Q_{i-1},C_j)+dist(q_{i-1},c_j), \quad \#削除$$
$$P(Q_i,C_{j-1})+dist(q_i,c_{j-1})\} \quad \#挿入$$

ただし我々のケースに関しては，例えばドアの開閉のスピードを考慮したい場合とそこまで考慮しない場合を調節したいという要求が考えられる．そこでスピードを考慮したい場合の検索では以下の式を使用することとする．

$$P(Q_i,C_j) = \min\{P(Q_{i-1},C_{j-1})+dist(q_i,c_j), \quad \#置換$$
$$P(Q_{i-1},C_j)+\alpha*dist(q_{i-1},c_j), \quad \#削除$$
$$P(Q_i,C_{j-1})+\alpha*dist(q_i,c_{j-1})\} \quad \#挿入$$

挿入コストと削除コストにユーザが指定できる調節パラメータ α を追加した．α を大きくすれば，「ドアの開閉」イベントの場合開閉スピードを考慮した検索が出来ると考えられる．

6.2 「ドアの開閉」イベント検索の実験

前節のように定義された編集コストを用いて，実際に「ドアの開閉」イベントの検索実験を行った．図 8 は実験対象データの挙動とクエリとした部分の SAX 文字列である．このデータには 4 人の被験者が同一のドアに対して「普通に開閉」した挙動，1 人の被験者が「遅いスピードで開閉」した挙動，「速いスピードで開閉」した挙動，「開けっ放しにしたあと閉めた」挙動の計 7 つのイベントが含まれている．今回は赤い部分 Q をクエリとした場合の検索実験を行う．

図 9 はしきい値と削除・挿入コストを変えた検索実験結果の図示で，橙色の部分が結果として返ってきた部分文字列である．

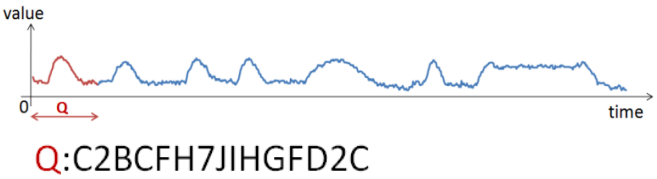


図 8：実験対象データの挙動（w=287，a=10）

(1)は，しきい値 0.2，α を 1 にした場合，(2)はしきい値 0.2，α を 10 にした場合である．

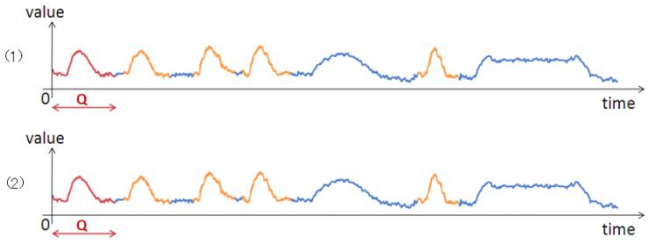


図 9：検索実験結果

(1)について、クエリ Q と似たような動きをしている最初の 3 つのイベントと、速いスピードで開閉したイベントは検索結果として返すことが出来た．しかし，(2)のようなしきい値はそのままにαだけを変化させた場合であるが、期待としては全てのイベントを検索したかったのだが、パラメータ α が上手く機能せず、α が 1 の時と同様、遅いスピードで開閉したイベントと開けっ放しにした後閉めたイベントは検出することが出来なかった．これは削除コストを小さくしすぎたためと考えられ，今後は挿入と削除のコストを一律に設

定するのではなく，別々に調節する必要があると考えられる．

7. 関連研究

我々が利用している SAX は時系列データを一定の区間ずつまとめて量子化する PAA(Piecewise Aggregate Approximation)[13]に基づき，近似距離が実距離の下界を抑えることが保障できるよう拡張したものであり，文字列に対する様々な検索技術やマイニング技術を適用することができる．

今回我々は部分文字列検索の高速化のための索引である接尾辞木を用い編集距離によって部分時系列データの検索を行った．同様に編集距離を用いて時系列データの類似検索を行ったものに Chen らの研究[4][5]がある．[4]では元の値は量子化せず，二つの時系列内の 2 時点間の距離が閾値内にあるか否かでコストを決定している．[5]では距離の公理を満たすようなコスト定義を行い，測度索引を利用して検索を高速化している．また SAX を長大な時系列検索に対応させた iSAX[2]では，滑走窓を用いて短く区切った SAX 文字列に対する索引を生成し，長大な文字列に対する高速な部分文字列の検索を実現している．我々が採用した接尾辞木とは異なり，iSAX の索引では，滑走窓で切り出した n 文字の文字列を n 次元のデータとし，n 次元ベクトルによる特徴空間に対しての索引を生成している．またこの手法では時系列のスケーリングには対応していない．

またストリームデータに関する部分類似検索の研究は Gao ら[6]や Zhu ら[15]などにより数多く行われている．また櫻井ら[14]豊田ら[12]などによってストリームデータにおける高速な DTW による類似検索も研究されている．今回我々は SAX によって量子化された時系列データにおいて文字列検索の高速化技術を用いて類似検索を行う試みを行ったが，これらの既存手法に対する有効性を今後検証していく必要がある．

8. まとめと今後の課題

本稿では，住宅内に設置された大量のセンサから送られるデータに対して問合せを行った際に高速な問合せが可能となるような SAX 索引の実装を行った．また，SAX の距離定義に基づく編集コストを定義した接尾辞木による高速な類似部分イベント検索を実現し，イベント検索実験を行った．

今後は，今回提案したコスト計算式の改良と，それに伴う再実験を行い状況差や個人差を区別できるかを検証する．また，他のイベント例でも同様に実験と検証を行っていく．

参 考 文 献

[1] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. : A Symbolic Representation of Time Series, with Implications for Streaming Algorithms," In SIGMOD workshop, 2003.

[2] Jin Shieh and Eamonn Keogh:"iSAX: indexing and mining terabyte sized time series," In 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008.

[3] Kemal Oflazer:"Error-tolerant finite-state recogniti-on with applications to morphological analysis and spelling correction," In Journal Computational Linguistics Volume 22 Issue 1, March 1996.

[4] L.Chen,M.T.Özsu, and V.Oria: "Robust and efficient similarity search for moving object trajectories," In CS Tech. Report. CS-2003-30, School of Computer Science, University of Waterloo.

[5] Lei Chen and Raymond Ng:"On the marriage of Lp-norms and edit distance," In Thirtieth International Conference on Very Large Data Bases (VLDB 2004), 2004.

[6] L. Gao and X.S.Wang: "Continually Evaluating Similarity-Based Pattern Queries on a Streaming Time Series," In the 2002 ACM SIGMOD International Conference on Management of Data, pp370-pp381, June 2002.

[7] 中島沙希:"センサデータに対する問い合わせ高速化のための索引の実装," In お茶の水女子大学 2009 年度卒業研究, February 2010.

[8] Ocha House : http://www.siio.jp/index.php?OchaHouse

[9] Ocha House Projects : http://www.siio.jp/index.php?OchaHouseProjects

[10] 岡部正幸,三輪多恵子,梅村恭司:"文字列解析に基づくネットワークトラフィックデータからの異常発見," In インタ-ネットコンファレンス論文集, pp67-pp74,2006.

[11] Tanaka, Y. & Uehara, K: "Motif Discovery Algorithm from Motion Data," In 18th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI), June 2004.

[12] 豊田真智子,櫻井保志,市川俊一:"データストリームのための部分シーケンスマッチング," In DEIM Forum, 2009.

[13] Yi, B, K, and Faloutsos, C: "Fast Time Sequence Indexing for Arbitrary Lp Norms," In 26st Int'l Conference on Very Large Databases, pp385-pp394, 2000.

[14] Y.Sakurai, C.Faloutsos,and M.Yamamuro:"Stream Monitoring under the Time Warping Distance," In IEEE 23rd International Conference on Data Engineering (ICDE 2007), pp1046-pp1055, April 2007.

[15] Y.Zhu and D.Shasha:" StatStream: statistical monitoring of thousands of data streams in real time," In 28th International Conference on Very Large Data Bases (VLDB 2002), pp358-pp369, August 2002.