センサデータに対する SAX を利用したイベント検索の検討

大西 史花　(指導教員：渡辺 知恵美)

1　はじめに

近年，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用されている．この動向に注目し，我々は，他のアプリケーションの基盤となる，住宅の情報を格納し効率的に利用することが出来る DB システムの提案と実装を行っている．しかし，センサデータのような短時間で大量に生成されるストリームデータに問合せを行うには時間がかかる．そこで，我々は先行研究 [1] にて，センサデータに対して高速な問合せが可能となる索引の実装を行った．その索引では，Lin らが提案した Symbolic Aggregate approximation (SAX) [2] を使用している．以後これを“ SAX 索引 ”と呼ぶ．

SAX 索引はセンサデータを符号化した長い文字列となる．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 によって符号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することができる．

本研究では，長い文字列から部分文字列を高速に検索する Suffix Array を用いてイベント部分を高速に検索するシステムを実装した．Suffix Array はクエリ文字列に一致する文字列を高速に検索する．しかし，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となる．そこで我々は SAX における近似距離の定義に基づいて編集距離による距離定義を行い，Suffix Array による編集距離検索手法 [3] を用いて高速な類似度検索を実現した．

2　SAX を用いたセンサデータの索引

2.1　SAX 索引

SAX 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符号化する．パラメータとして文字列長 w と文字種類 a の 2 つを決めることが出来る．この手法を採用する利点としては，実装が簡単であること，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が挙げられる．

SAX による時系列データの文字列化の手順を以下と図 1（$w = 6$，$a = 5$）によって示す．図 1 では破線が SAX を適用するセンサデータである．

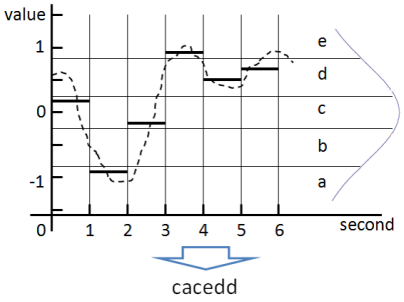

図 1: SAX による時系列データの符号化

(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（PAA 近似，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 の大きな特徴は二つの時系列データ Q，C の距離 d(Q,C) に対して，対応する SAX データ Q’，C’ の近似距離 d’(Q’,C’) が元の距離の下界を抑えることである．近似距離と実際の距離との幅は w と a を調節することによりパフォーマンスを考慮したうえで近づけることができる．

文字列間の距離は図 2 に示すように対応する各文字の距離の二乗の総和である．文字間の距離は各文字がとり得る数値のうちで互いに最も近い値を取った場合の距離とする．

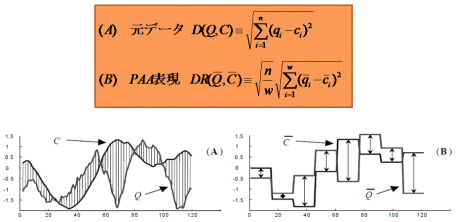

図 2: SAX 文字列の近似距離測定法

また我々は SAX 索引をランレングス圧縮によって圧縮して格納した．イベントが頻繁でないようなセンサの場合は同様の値が続くことが多いため，データを大きく圧縮することができる．

2.2　SAX 索引に対する Suffix Array の付与

先行研究の段階では，SAX 索引に対して問合せを行う際のデータ探索方法はシーケンシャルスキャンであった．そこで本研究では Suffix Array を採用し実装する．この手法の採用によりデータ探索方法が 2 分探索となり，更なる問合せの高速化が期待される．

Suffix Array とは，文字列の接尾辞にインデックス値を付与し，その値を保持したまま接尾辞を辞書順に並べ替える配列構造のことである．SAX 索引に対する検索を高速に行うため，我々は圧縮された SAX 索引に対し Suffix Array を適用した．

例えば，文字列 abac の接尾辞は「abac」「bac」「ac」「c」の 4 つである．この 4 つの接尾辞を 0〜3 のインデックスを持つ配列に格納し，その後接尾辞を辞書順に並べ替えると，インデックス値は {0，2，1，3} となる．これを接尾辞配列と呼ぶ．このインデックス値は元テキストにおける各接尾辞の開始位置を示していることになる．例えば，接尾辞配列の最後にある { 3 } は，対応する接尾辞「c」が元テキスト abac の 3 文字目に存在していることを示している．

本研究の索引は，1 日分のセンサデータに対して 1

つの SAX 索引を作成している．よって「1 月 1 日にドアが開閉した時刻」を検索する際には，文字列化したクエリと，1 月 1 日のデータに対して作成された SAX 索引とで 2 分探索を行い，一致があればその位置から時刻を逆算することで結果を返す．

3　イベントに対する検討

本節では，SAX 索引に対する Suffix Array 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．

例えば図 3 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 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（C）で，下部はその値を SAX 索引に変換したものである．

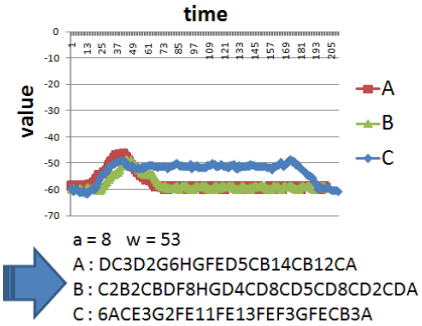

図 3: 状況差によるドア開閉のセンサデータ値

我々は時系列データから「ドアがいつ開閉したか」だけでなく，ユーザの要求によって，例えば同じドアの開き方の特徴を持つデータだけを検索することによってドアの開いた状況（複数人出入りした，開いたまま会話していた）を考慮した検索をしたり，個人差を考慮してある人物がドアを開閉した時刻だけを検出するなどの柔軟な検索が行えるようにしたい．

また，時系列 A と時系列 B のようにドアが開いてから閉じるまでの間隔に違いがある場合もある．このような時系列の動きに対するスケールの違いに関しても，ユーザの要求によって考慮する場合と考慮しない場合が考えられ，十分に対応できるようにしたい．

4　編集距離を用いた類似度検索

SAX では比較する二つの文字列の近似距離を求めることができ，それによって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 Suffix Array を適用している．基本的には Suffix Array は完全一致する部分文字列を検索するためこのままでは上記のような状況や個人差を考慮することができない．そこで我々は文字列間の距離の定義として編集距離を採用した．

編集距離とは，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 Q と C の距離 dist(Q,C) として計算する．例えば Q=”ABCE”，C=”AFCD” とした時，Q の 2 文字目に F を挿入，4 文字目の E を F に置換すると C と一致する．挿入および置換にかかるコストをそれぞれ 1 とすると，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 Suffix Array により検索する手法は山下ら [3] によって提案されており，我々はこれを採用する．

編集距離における編集コストは SAX における距離定義にしたがって以下のように定義する．

■置換コスト

置換コストは SAX における文字間の距離に従う．文字間の距離は文字種類 $a*a$ の表から得ることができ，そのセルの値 (r,c) は次のように定義されている．

$$Cell_{r,c} = \begin{cases} 0, & |r-c| \leqq 1 \\ \beta_{max(r,c)-1} - \beta_{min(r,c)}, & otherwise \end{cases} \quad (1)$$

β とは，正規分布の面積を a 個分に等しく分割する値である．例えば文字種類 3（abc）のとき分割点は 2 つであり，それぞれ β_1 = -0.43，β_2 = 0.43 となる．よって (a,b) = 0，(a,c) = 0.43 となる．

■挿入，削除コスト

挿入コストと削除コストは前節にて考察したように，ユーザがイベントの挙動が起こる速さが異なっても同一イベントしたいか，異なるイベントと認識したいかによって異なる．そこで挿入コストと削除コストはユーザが指定できるパラメータとする．削除コストと挿入コストを共に 0 とすれば，数字を無視した検索，「ドアの開閉」イベントの場合開閉スピードを考慮しない検索が出来ると考えられる．

以上のように計算される編集距離を用いて，図 2 で示した実際の時系列 A をクエリとし，削除・挿入コストを 0 とした B，C それぞれのコストは 6.07 と 4.85 となる．また，A をクエリとし，削除・挿入コストを 1 とした B，C それぞれのコストは 70.07 と 46.85 となる．よって図 2 のデータのうち，状況差を考慮せず単に「ドアの開閉」というイベントを検出したい場合は，削除・挿入コストを 0 として，閾値を 6.07 以下に，状況差を考慮して，クエリと同様な挙動をする「ドアの開閉」というイベントを検出したい場合は，削除・挿入コストを 1 として，閾値を 46.85 ～ 70.07 の間にとればよい．

5　まとめ

我々は，センサデータからのイベント検索を高速におこなうために Suffix Array を適用し，SAX の距離定義に基づく編集コストを定義することによって類似部分イベント検索を実現した．

今後は，今回の実験を更に追及することに加え，他のイベント検索でも同様に実験と検証を行っていく．

参考文献

[1] 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.

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

[3] 山下達雄: “Suffix Array を用いたフルテキスト類似用例検索,” In IPSJ SIG Notes, September 1997.