社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－CSEC－34 (57)
2006／7／21

キーストロークデータを対象としたコンテクスト抽出手法の提案

石川 宗寿† 岡 瑞起†,†† 加藤 和彦†

† 筑波大学大学院システム情報研究科 〒305-0006 茨城県つくば市 天王台 1-1-1
†† 日本学術振興会 〒102-0082 東京都千代田区一番町 6

あらまし 我々は、プライバシーを保全するためにキーの文字情報を利用しない、キーストロークによる常時認証をするという研究を行っている。この常時認証を高精度で行うには、キーストロークのコンテクストを抽出し、同じコンテクスト同士で比較を行う必要がある。本研究で言うコンテクストとは、例えばワープロで文書を書いている・webブラウジングをしている、といったキーボードの使用状態をいう。ここで、文字情報を利用しないキーストローク認証の場合、コンテクストの抽出に使用可能な情報はキーイベントのタイミング情報のみになる。その問題を解決するために、我々は異なる幅の時間間隔を同時に考慮するというdt Vectorize 手法を提案し、実験を行った。その結果、時間間隔を固定した場合に比べ、抽出制度が平均で約7%～13%向上した。

キーワード キーストローク、コンテクスト、認証、ベクトル化、情報拡張

Extracting Contexts From Keystroke Data

Munetoshi ISHIKAWA†, Mizuki OKA†,††, and Kazuhiro KATO†

† Graduate School of Systems and Information Engineering, University of Tsukuba Ten-noudai 1-1-1, Tsukuba-shi, 305-0006, Japan
†† Japan Society for thePromotion of Science Ichibancho 6, Chiyoda-ku, 102-0082 Japan

Abstract We have implemented a verification system that continuously verify current users using keystroke dynamics. With a privacy protection motive , our system does not use characters associated with keystrokes but only uses timings for verification. For an accurate verification, it is essential for our system to be aware of user's contexts so that the system can compare the same kind of contexts. In this paper, we propose "dt Vectorizing" method as a solution to achieve an accurate verification. Our methods looks at different time interval to characterize keystroke sequences. Our experimental results showed an increase in verification rate by 7-13 % on average.

Key words keystroke,context,authentication,vectroize,parameter-extending

1. まえがき

現在ユーザ認証として、最も広く使用されている方法にパスワード認証がある。しかしパスワードによる認証は、パスワードが流出したり、ユーザがシステムを利用可能な状態で放置した場合は無防備になるという問題がある。そのため、より安全なユーザ認証を行うためには、システムが稼動している間、常に認証し続ける必要がある。そこで我々は、常時認証する手段の一つとしてユーザのキーストロークを監視する手法を研究し、それを用いて不正使用を検知するシステムを開発している

キーストロークによる常時認証行うには、キーの入力情報を常に取得し続ける必要がある。これは、メーラーやワープロソフトといった、通常のアプリケーションソフトウェアを利用する際にも、何を打鍵したかという情報を取得することを意味する。他にも、ユーザがブラウザを使用し、web上で何か別のパスワードを入力しているかもしれない。これでは、プライバシーの点で問題がある。それを解決する方法として我々が行っている方法は、「何のキーを押したか」というキーの種類の情報を取得せずに、「キーをいつ押して、いつ離したか」というタイミングの情報のみを用いて行おうというものである。

ここで、キーストロークによる常時認証を高精度にするには、キーストロークのコンテクストを抽出し、コンテクストに応じた学習や特徴比較を行う必要がある。例えば、webブラウジングをしているときは、webブラウジングの学習データと特徴比較をしなくてはならない。もしwebブラウジングをしている時に、それをワープロによる文書作成の学習データと特徴比較をしてしまうと、認証の精度が低下する。また学習において、webブラウジングをしているときのキーストロークを、コー

ディングをしているものとして学習してしまうと以降の特徴比較のときに悪影響が出る。

このコンテクストを抽出する方法としては、現在フォーカスのあたっているプロセス名を取得したり、キーボードから入力されている内容を用いて抽出する方法が考えられる。しかし、我々の研究している手法においては、プライバシーを保護することを目的とし、取得する情報はキーストロークのタイミングのみということを前提としているため、タイミングの情報のみでコンテクストを抽出したい。

タイミング情報から取得できる最も特徴的な情報のひとつに、ある時間間隔における頻度がある。そこで本研究では、キーイベントの起こる頻度を用いて、キーストロークのコンテクストを抽出することを目的とする。その手順として、まずキーストロークのタイミング情報からキーイベントの頻度を算出し、その統計情報を取得する。そして、コンテクスト毎に統計情報の特徴を抽出し、それを学習を使用して蓄積し、それと抽出対象のキーストロークの特徴を比較することにより、コンテクストの抽出を行う。

ここでキーイベントの頻度とは(キーイベントの数)/(時間間隔)というスカラーである。その頻度は、同一のコンテクスト内でも変化の度合いが大きく、違うコンテクスト間でもとり得る値の範囲が重なることが多いという特徴がある。時間間隔を1secとした場合の例を示すと、webブラウジング時のキーイベントの頻度はおよそ0～5(*events/sec*)であり、プログラミング時の頻度は0～6(*events/sec*)程度といったようにほぼ変わらない。その状況で、コンテクストが不明である頻度0～5(*events/sec*)のキーイベント列が渡されても、そのときのコンテクストがwebブラウジング、プログラミングのどちらであるかを抽出するのは難しい。従って、頻度をある値に固定した場合は、頻度の統計情報の特徴を学習させても、コンテクスト抽出の精度向上は難しい。

そこで我々は、異なる幅の時間間隔を同時に考慮するというdt Vectorize手法を提案する。このdt Vectorize手法では、スカラーである時間間隔をベクトルに拡張することにより、同時に複数の時間間隔を考慮する。それによって、頻度の次元が拡張され、各コンテクストのとる頻度の値が重なる可能性を減らすことができ、結果としてコンテクスト抽出の精度が向上できる。本論文ではこのdt Vectorize手法の有効性を示す。

2. コンテクスト抽出の概要

提案手法について述べる前に、本研究で用いたコンテクスト抽出方法の概要について説明する。本研究で用いたコンテクストの抽出方法は、キーストロークのタイミング情報の特徴を学習し、学習した特徴と抽出対象のキーストロークのタイミング情報の特徴を比較するというものである。抽出対象のコンテクストは、その特徴に最も近い学習データのコンテクストであると抽出する。

この方法の流れを大きく3つのフェイズに分けると、学習フェイズ、テストフェイズ、およびフィードバックフェイズに分けられる。以降この流れの概略を述べる。

学習フェイズは、入力されたデータからコンテクストの特徴を抽出し、それを学習モデルとして蓄積するフェイズである。ここでは以下のようなことを行う。

- キーロガー等を用いて、学習に使用するキーストロークのタイミングデータを取得する。
- 取得したデータに、"そのデータにおけるコンテクスト"を示すラベルを付ける。これを学習データとする。(図1の1)
- 同一のコンテクストにおけるデータの特徴を抽出する。
- それらをコンテクスト毎に蓄積しする。これを学習モデルとする。(2)

テストフェイズは、抽出対象のデータの特徴と学習モデルを比較し、そのデータのコンテクストを抽出するフェイズである。ここでは以下のようなことを行う。

- キーロガー等を用いて、抽出を行いたいキーストロークのタイミングデータを取得する。(3)
- テストデータの特徴と各学習モデルを比較する。この比較の機能を持った関数をスコア関数とする。(4)
- (4)の結果、テストデータと特徴が最も似ている学習モデルのコンテクストを、テストデータのコンテクストとする。この解となるコンテクストを返す関数を、識別関数とする。(5)

フィードバックフェイズは、テストフェイズの結果を受けて学習モデルを修正するというフェイズである。ここでは以下のようなことを行う。

- (5)によって抽出されたテストデータを、そのコンテクストの学習データに加え、特徴を再抽出する。(6)
- 再抽出された特徴を新しいモデルとして、学習モデルを更新する。

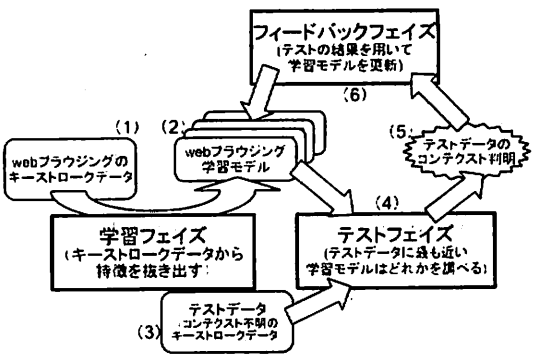

図1 コンテクスト抽出の概要

これら3つのフェイズの中で、本研究で提案する手法は学習フェイズに関する手法である。次節では提案手法について述べる。

3. 提案手法

3.1 dt Vectorize手法

我々が提案する手法の基本方針は、コンテクストの抽出に使用する属性を増やすことである。その目的は、データから特徴抽出を行う際に、高次元空間にマッピングをすることである。

こうすることで、コンテクストを抽出しやすくなる。その理由を、図2を用いて説明する。この図には既知のコンテクストA・Bの学習データと、コンテクストが不明なテストデータCがあり、それぞれ属性 x_1 と x_2 を持つとする。そして、データCはコンテクストA・Bのどちらかに一方に属することが既知であり、そのどちらに属するかを抽出をしたい場合を考える

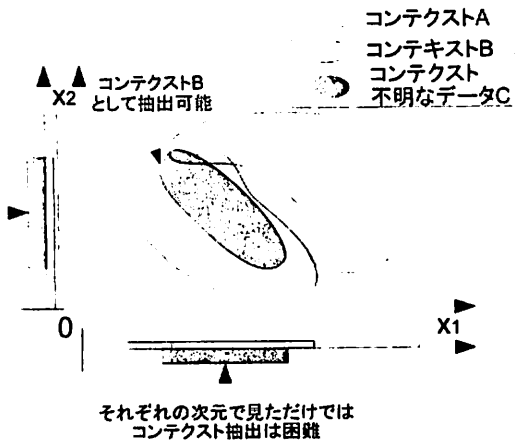

図2 属性の高次元化

最初に、観測可能な属性が x_1 のみ、つまり図2の横軸しか見えないとする。この状況下では、Cのとる x_1 の範囲がAとBの両方にほぼ重なっているため、CがAとBのどちらに属するかを抽出することは困難である。次に、属性 x_1 の他に x_2 も観測可能であるとする。この時に、x_1 と x_2 をそれぞれ独立して見たままでは、x_1 と x_2 の両方において、Cの属性がとる範囲はAとBの範囲にほぼ重なっているため、やはり抽出が困難となる。しかし、各コンテクストのとる属性の範囲を x_1-x_2 の空間にマッピングすると、この場合はCはBに属するということが抽出できる

このように属性の種類が増えると、それぞれが独立した情報を持つ場合は、それらの情報を統合することで抽出の精度を上げることができる。しかしながら、本研究で行うコンテクスト抽出では、使用する情報をキーストロークのタイミング情報に限るため、キーイベントの発生頻度という属性以外は取得することが困難である。そこで我々の提案する手法では、頻度で用いる時間間隔を1つの値に固定せずに、複数の値をとることを許すことで、それぞれの時間間隔における頻度属性を、別の種類の属性と扱う。それによって、抽出に使用する空間を高次元化し、抽出の精度を高めることができる。この手法は本来スカラーである時間間隔を、ベクトルとして扱うことで実現できることから、我々はこれをdt Vectorize 手法と名付けた

このdt Vectorize 手法の概要を図示すると、図3のようになる。本手法ではこの図のように、時間軸に様々な大きさの区間を設定し、その中のキーイベントを数え、区間の大きさで割ることによって頻度を求めている。この手順を式を用いて解説する

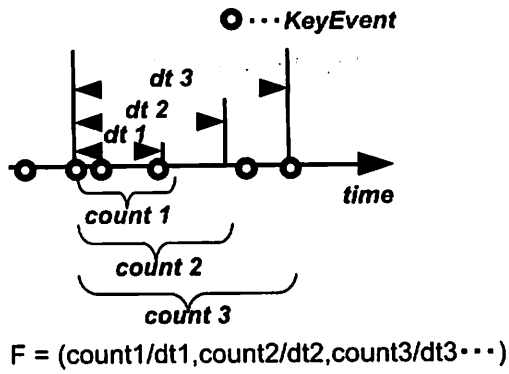

図3 dt Vectorize

まず、本手法では複数の時間間隔 dt_i を集めたベクトル $\mathbf{dt}$ を用いるので、これを以下のように定義する

$$\mathbf{dt} = (\mathbf{dt_1 dt_2 \cdots dt_n})^t$$

ここで、時間間隔を dt_i とした場合の時刻 t における頻度を $f_i(t)$ とすると、それらを集めた頻度ベクトル F は以下のようになる

$$F(t) = (f_1(t) f_2(t) \cdots f_n(t))^t$$

ここで一般的な頻度の定義は、

$$(\text{頻度}) = \frac{(\text{イベント数})}{(\text{時間間隔})}$$

であるので、$f_i(t)$ と dt_i の関係式は以下のようになる

$$f_i(t) = \frac{\int_t^{t-dt_i} g(x)dt_i}{dt_i}$$

ただし $g(x) = \begin{cases} 1 & \text{Exists a KeyEvent at time } x \\ 0 & \text{else} \end{cases}$

ここで $g(x)$ は、時刻 x において、キーイベントが存在するときに1を返す関数であるから、上式の分子は x から $x+dt$ までのキーイベント数という意味になる

本手法では新たに、dt_i と dt_{i-1} の間隔を設定しなくてはならない。仮に計算コストを考慮しないとすると、$\mathbf{dt}$ ベクトルは、各ベクトル要素が稠密値でベクトル長が無限であるのが、情報量の点からみると理想である。しかし、実際にコンピュータで計算を行うには、$\mathbf{dt}$ の要素 dt_i と dt_{i-1} 間のステップ値を設定する必要がある。またベクトル長も有限にし、計算を打ち切る必要もある。ここで、実際のキーストロークデータを用いて作成した、図4のグラフを参照されたい。これはワープロソフトで文書作成をしている時のキーストロークデータで、ある時刻 t における dt と頻度 F の関係を示すグラフの一例である。横軸は dt の添え字 i を表し、縦軸は F の値を示している。このグラフでは $dt_i = i(sec)$、つまり dt_i と dt_{i-1} の間隔 $1(sec)$ とした。このグラフで見るとわかるように dt が小さい値の場合、F の変化が激しく、dt の値が増えるとともに、F は収束してい

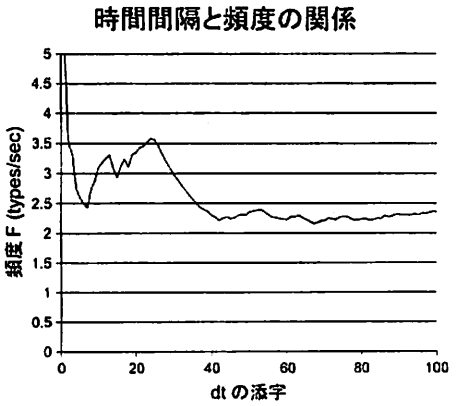

図 4 $dt_i = i(sec)$ の場合の例

く、そこで本手法では、dt_i を定数 a、b を用いて $dt_i = b \cdot a^i$ と決めた。このように dt_i を設定することで、F の変化が激しい dt の小さい間は dt_i と dt_{i+1} の間隔が小さく、dt が大きくなり F が収束していくと間隔が大きくなるようにできる。図5に、図4と同じデータで、$dt_i = 1000 \cdot 1.2^i$ と定めた時のグラフを示す。

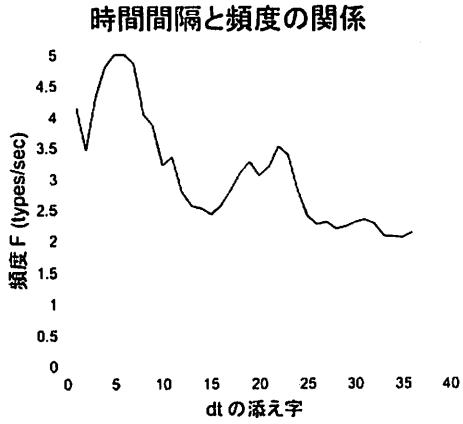

図 5 $dt_i = 1000 \cdot 1.2^i(sec)$ の場合の例

また、dt のベクトル長は、頻度 F が収束したところで打ち切るようにした。図5での dt ベクトル長は36とした。

この dt Vectorize 手法のアルゴリズムを図6に示す。

3.2 dt 固定手法

dt Vectorize 手法を使用せずにある dt を定める場合は、以下の手順により頻度を算出する。

（1） 頻度を計算する時区間 dt を決定する

（2） 抽出対象となるキーイベント集合を dt 毎に分割する

（3） 分割された集合毎に $\frac{\text{キーイベント数}}{dt}$ を算出する

この手法を dt Vectorize 手法と区別して、dt 固定手法という

```
dt Vectorize method
function freq_calc:
input: start, step, end, data_{1...datasize};                    //
output: freq_{(1....1...)};                                      //
var
    n, m, i, j, k : integer;                    //i,j:ループ変数,
    dt : array of integer;                                       :
begin
                                                  //dt ベクトル作成
    n := 1;
    while start < end
        dt_n := start;
        start := start * step;
        n := n + 1;
    end
                                                  //頻度計算
    freq_{(1...n...1)} := 0                                      //
    for i := 1 to n - 1 do begin                                 //
        for j := 1 to datasize do begin
            for k := 1 to datasize do begin
                if dt_i > data_j - data_k then
                    freq_{(i,j)} := freq_{(i,j)} + 1;
                    freq_{(i,k)} := freq_{(i,k)} - 1;
                end
            end
        end
        freq_{(i,...,1)} := freq_{(i,...,)}/dt_i;                //
    end
end
```

図 6 dt vectorize のアルゴリズム

これを図示すると、図7のようになる

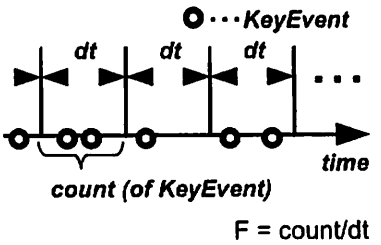

図 7 dt 固定手法

dt 固定手法では、何らかの方法で dt を決定する必要がある。以下に、主な方法のうち2つを挙げる

（1） ヒューリスティックス的に解決

（2） 学習過程で最適な dt を、再計算を行うことで探す

しかし、これらの方法で dt を決定するのには以下のような問題がある。まず(1)を用いて dt を決定した場合は、導かれた dt が最適である保証がないという問題がある。また、良好

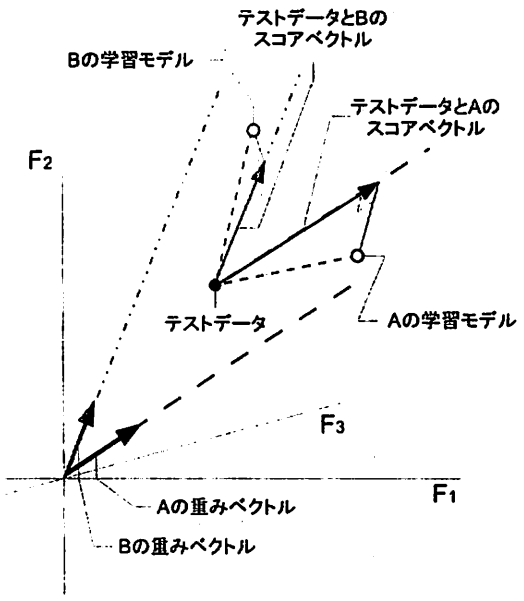

図 8　dt Scoring

な dt を探索する時間も、人手による試行錯誤が多くなるために、長くなってしまうという問題もある。さらに、学習を重ねるに従って最適な dt の値が変化するという可能性があるにも関わらず、それにあわせて dt を変化させには手動でしか変えることがことができない点も問題である。

また、(1)(2) の両方に共通する問題として、決定したいコンテクストごとに最適な dt が違う場合は、対象データがどのコンテクストに分類されるかを決定する際、統一的な指標がなくなるため、工夫が必要となる点が挙げられる。ここで、dt を全コンテクスト間で統一する方法もあるが、そうした場合は、各コンテクストから見た dt は最適でなくなるので、抽出の精度が落ちるという問題がある

これらの点で dt Vectorize 手法は有利であり、これについても精度を向上させる要因になると考えられる。本研究では (2) の方法との比較を行った

3.3　dt の重み付け

dt 固定手法における dt は 1 次元スカラーであるのに対し、dt Vectorize 手法の dt は多次元ベクトルであるため、それらの統計情報を取得した場合に次元の違いが存在する。ここで dt 固定手法と dt Vectorize 手法の性能比較を行うためには、2 手法間で、スコア関数・識別関数を統一する必要があるため、これらの次元の違いを吸収しなければならない。また dt Vectorize 手法において、頻度 F を高次元化したままでは、抽出の役に立たない情報や、ノイズとして抽出に悪影響を及ぼす情報も含むことも考えられる

我々はこれらの問題を解決するために、有用な dt_i データに重み付けを行い、dt_i の識別結果のスコアを累積することで、次元の違いを吸収する方法を用いた。この内、重み付けは学習フェイズで行い、スコアの累積はテストフェイズで行う。まず、重み付けの手順を以下に示す。

（1） dt の各要素 dt_i に対して、dt 固定手法を適用し学習を行う。

（2） (1) で使用した学習データをテストデータとみなして、各 dt_i でスコア関数をかける。

（3） (2) の抽出精度に応じて、各 dt_i に重み付けをする。

（4） (3) を各学習データコンテクスト毎に分けて行う。

（5） 割り振られた重みを、コンテクスト間で大きさが等しくなるように正規化する。

(3) で使用したり重み付けには、以下の式を使用した。

$$weight = \frac{S_{true}}{S_{all}} \cdot \frac{O_{true}}{O_{all}}$$

（1） 各 dt_i に関して、テストデータをスコア関数にかけ、それぞれのコンテクストに対するスコアを求める。

（2） (1) のスコアに、各 dt_i の重みを掛ける。

（3） (2) のスコアの総和をコンテクスト毎に算出する。

ここで S_{all} と O_{all} はそれぞれ学習データと同一のコンテクストデータの総数と、異なるコンテクストデータ総数である。そして S_{true} と O_{true} はそれぞれ正しく同一・異なると抽出されたデータ数である。この式は、完全な抽出ができた場合に重さ 1 となる。また、同一・異なるコンテクストの抽出の内、どちらか一方でも抽出精度が低い場合は、重みが小さくなるように決められている。これは、同一のコンテクストが正しく抽出されない場合に重みが低くなるのは無論であるが、1 つのコンテクストが他のコンテクストテストデータを誤抽出して、独占してしまうのを防ぐ意味を持つ。また、(5) で重みを正規化することで各コンテクスト間で異なる重みを使用しても、コンテクスト間で公平な抽出を実現している。ここでは正規化の基準として、ベクトル 2-ノルムを統一させた。

次にスコアの累積を行う手順を示す。この手順を図にしたものを 8 に示す。この図では、コンテクスト A・B の学習データとテストデータがある。また、ここでのスコア関数は、各属性軸に関して学習データとテストデータの距離返すものであるとする。その場合、各コンテクスト・テストデータ間の差ベクトルと、各々の重みベクトルの内積をとればよい。即ち、この図におけるスコアベクトルの大きさが、スコア累積の結果となる。ここでは、スコア関数はデータ間距離としてあるので、このスコアベクトルの大きさが小さいほうに、テストデータは抽出される

この重み付けを行うことで、dt Vectorize 手法に、dt 固定手法のスコア関数を適用することができる。これは多くのスカラー要素用のスコア関数を、dt Vectorize 手法に使用することができることを意味する。即ち、この手法とスコア関数はそれぞれ独立しており、スコア関数を自由に取り替えることができる

表 1 取得データ

UP・DOWN	キーストローク No.	イベントタイム (*msec*)
DOWN	1	129142931
DOWN	2	129142973
UP	1	129143030
UP	2	129143096
DOWN	3	129143204
UP	3	129143281
DOWN	4	129143332
UP	4	129143429
DOWN	5	129143829
DOWN	6	129143906
⋮	⋮	⋮

表 2 整形データ

イベントタイム (*msec*)
129142931
129142973
129143204
129143332
129143829
129143906
⋮

表 3 dt ベクトル

i	dt_i (*msec*)	F_i の同 (*types/sec*)
1	961	0.0061
2	1157	0.0054
3	1388	0.0051
4	1666	0.0047
5	2000	0.0046
6	2400	0.0044
7	2880	0.0043
⋮	⋮	⋮

4. 評価実験

4.1 実験方法

実験に必要なデータを採集する方法として、実験協力者が各々普段使用している windows マシンに、キーロガーを入れてもらった。このとき使用したキーロガーは、実験協力者の一人に C#言語で作成してもらったものであり、文字情報を取得せずに、何時キーイベントが起きたかを記録するものである。実験協力者は 6 人で、キーロガーを用いて一週間のキーストロークの記録を行った。その際、何をしたかを毎日書いてもらった。また、この時記録したデータは、以下の 3 種である。

- キーロガーを起動してから何回目のストロークであるか
- キーを押したのか離したのか
- ストロークが起きた時刻 (マシンを起動してからの経過時間)

このデータの例を表 1 に示す。

表 4 重みベクトル

i	文書作成	ブラウジング	プログラミング
1	0.4937	0.5357	0.3583
2	0.5049	0.5322	0.6607
3	0.4385	0.4705	0.4566
4	0.3695	0.3538	0.3702
5	0.2888	0.2348	0.2528
6	0.2014	0.1364	0.1398
7	0.1708	0.0745	0.0695
8	0.1099	0.0621	0.0356
9	0.0576	0.0124	0.000
10	0.0419	0.000	0.000
11	0.0315	0.000	0.000
12	0.0262	0.000	0.000
13	0.0210	0.000	0.000
14	0.0052	0.000	0.000

このうち今回使用したデータは、キーが押された時の時刻のみを取り出して整形したものである。その例を表 2 に示す。

次に、整形したデータを 120〜180(sec) 毎に分割する。分割する基準は 120〜180(sec) の間で、最も長くキーイベントが起きなかった時刻に決めた。これを 1 つの window とし、コンテクストの抽出はこの window を単位として行う。そしてこの window を用いて、dt vectorize 手法と dt 固定手法を、それぞれ使用した場合に分けて実験を行った。

ここでの学習モデルは、各 dt_i 毎に頻度の平均 μ と分散 σ を求め、その 2 つのベクトルによって表した。また、頻度は正規分布に従うと仮定して分布関数をつくり、それをスコア関数としたテストデータのスコアは、そのスコア関数に頻度平均 x を代入することで求めた。以下にこのときの正規分布の関数を示す。

$$score = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

また、今回のコンテクスト識別対象は、文書作成・web ブラウジング・プログラミングのみに絞った。本実験では取得したデータ中で、これらのコンテクストに確実に当てはまると判ったもののみ使用した。ラベル付けを行ったのは合計約 12 万ストロークで、各コンテクストの内訳は文書作成:ブラウジング:プログラミング = 76%:13%:13% となった。これらを更に実験 10 回分に分け、一回の実験で約 1 万 2 千ストロークを用いた。その中で 20% を学習データと使用して、残りをテストデータとして使用した。計算に使用したマシンは dell の Precision 650(CPU:Intel(R) Xeon(TM) CPU 3.20GHz Memory:3.6GB)Fedora Core 4 を使用し、実験は matlab 7.0 R14 上で行った。

4.2 結　果

まず、各コンテクストごとのスコア基準ベクトルは、表 4 の示される値になった。これを見ると文書作成は比較的長い dt に重みがつき、ブラウジングは短い dt に重みがついた。特徴的なものがコーディングで、約 1sec の付近で強く重みがつき、他の dt では重みが小さかった。

次に抽出精度データを図 9 と図 10 に示す。これらのグラフで

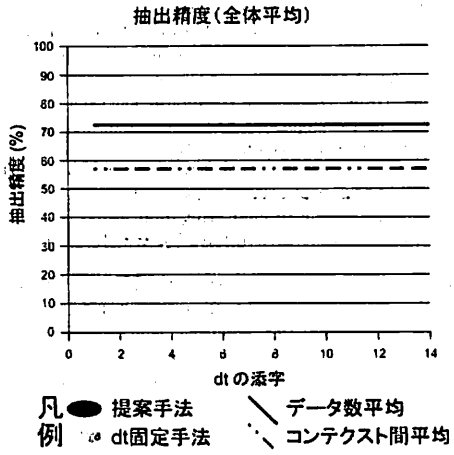

図 9　抽出精度（平均）

表 5　実験結果（実行速度）

Matlab 起動終了時間	dt Vectorize	dt 固定	(内 Matlab)
起動終了時間込み	28.757	6.587	6.250
起動終了時間抜き	22.502	0.337	--

は、横軸が dt の添字 i、縦軸が抽出精度を示す．dt Vectorize 手法は十分すべての dt の値を使用するため、便宜上横軸に平行なグラフを書いた．また図 9 におけるデータ数平均とは、全テストデータの中で抽出に成功したのはどのくらいかを示し、コンテクスト間平均とは、各コンテクスト間でデータ数を正規化した場合の抽出精度を示す．各コンテクストそれぞれの抽出精度は、図 10 に示した．

dt Vectorize 手法のデータ数平均とコンテクスト間平均の抽出精度は、それぞれ 72.61% と 57.09% となった．一方で dt 固定手法に関して、データ数平均では $i = 14(dt \fallingdotseq 10sec)$ のとき最高値 65.31% であり、コンテクスト平均は $i = 11(dt \fallingdotseq 6sec)$ で 44.23% という結果になった．よって、データ数平均で約 7%、コンテクスト間平均で約 13% 抽出精度が向上したといえる．

コンテクスト別に見てみると、dt 固定手法の場合は、文書作成とプログラミングの抽出精度はトレードオフの関係になっていて、ブラウジングに関しては一定して抽出精度が低かった．一方で dt Vectorize 手法は dt 固定手法と比較すると、プログラミングでの抽出制度は平均して 12% ほど落ちたが、ブラウジングに関しては平均で約 32% の改善と、劇的な向上が見られた．

各手法の実行速度に関しては、表を参照されたい．Matlab の起動時間は共通時間であるから、各手法の実際の実行時間は表 1.2 の下段となる．これで見ると、dt Vectorize 手法は dt 固定手法の 67 倍時間がかかっている．しかし、学習データ数 n に対するオーダーは両手法ともに $O(n)$ であるので、抽出制度向上に対しては妥当であるといえる．

4.3 考　察

まず、dt 固定手法について考察する．文書作成は大きい dt で見ると、他のコンテクストより頻度が高くなりやすいため一定以上の dt になると抽出精度が比較的高い状態で安定したと思われる．プログラミングについては、大きい dt で見てしまうとウェブによる参照や考える時間が入るため、小さい dt の方が抽出精度が高かったと考えられる．ブラウジングは、ブラウザを開いていたとしても、ウェブコンテンツの種類に応じて行うキーボード入力が異なるため、dt を固定した場合は、抽出精度が上がらなかったと考えられる．

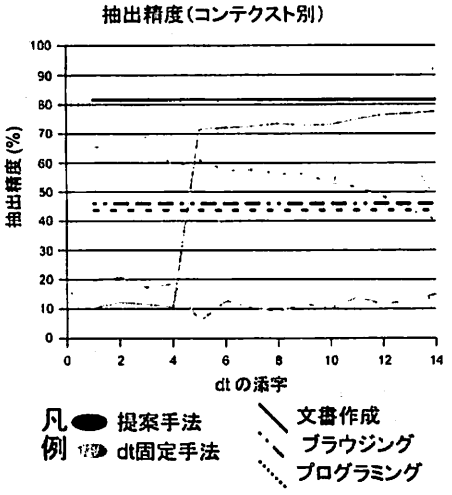

図 10　抽出精度（コンテクスト別）

次に、dt Vectorize 手法について考察する．dt Vectorize 手法では、プログラミングと文書作成が各々識別しやすい dt を使用したので、両方とも一定の抽出精度を保つことができた．ブラウジングについては、dt を固定した場合、どの dt の値を使っても抽出精度は上がらなかったが、複数の dt を同時に見ることが結果につながったと思われる．ただし、この手法では作成する頻度ベクトルが膨大となるため、dt 固定手法に対して 67 倍も時間がかかった．

5. 関連研究

我々の知る限りでは、キーボードから取得できる情報のみでコンテクストを抽出する研究はされていない．しかしながら、デスクワーク等のキーボードを使用する状況で、コンテクストを抽出する場合に、情報の一部としてキーストロークを利用している研究はある．例を挙げると、永口 充らによる研究 [1] では、ユーザの忙しさを推定するために、キーボード・マウスの使用状況や、カメラやマイクから取得できる情報を用いている．この研究の目的は、ユーザ支援の提案を忙しい時に行わないようにすることであり、本研究とは目的が大きく異なる．

また、キーストロークによる認証を行っている研究に Francesco Bergadano らの研究 [5] がある．これは、自由文脈によるキーストローク認証の研究である．ここで用いられている方法は、キー 3 つの組み合わせとそれを押すのにかかった

時間を学習しておき、テスト入力と学習データを比較することによって認証を行う方法である。しかし、テスト入力されるものは、文章を書き写している時のキーストロークであることを前提としている。そのため、これを常時認証で利用するには、コンテクストの抽出が必要となる。

6. まとめ

本研究では、キーストローク認証を高精度で行うために、キーストロークのコンテクストを抽出することを目的とした。その際の制約として、キーストロークのタイミング情報しか使わないとした。それによって起こる問題として、使用できる情報の種類が限られる点が挙げられた。我々はその問題を解決するために、情報の種類を拡張するというdt Vectorize手法を提案した。この手法は重み付けを行うことで、拡張前に使用していたスコア関数・識別関数を使用することも可能である。そしてdt Vectorize手法の有効性を示すために、6人1週間分のキーストロークデータを用いて実験を行った。その結果、この手法を用いない場合と比べて計算時間が約67倍となったが、抽出制度がデータ数平均で約7%、コンテクスト間平均で約13%向上した。これによってdt Vectorize手法の有効性が示せたと言える。

また、今後取り組むべき課題としては大きく以下の2点が挙げられる。

（1） スコア関数・識別関数を変更して抽出精度の向上を計る。

（2） フィードバックフェイズを作成する

(1)について、今回はスコア関数に正規分布関数を使用しただけであったが、統計情報だけではなく、動的情報を用いた関数を適用すれば、抽出率が向上する可能性がある。(2)のフィードバックフェイズの作成にあたっては、特に、誤抽出可能性の考慮とインクリメンタル性の2点について注意しなければならない。誤抽出をした場合、それがフィードバックをして学習データを書き換えると、抽出精度が低下して、悪循環に陥る可能性がある。そのため、フィードバックさせるデータとさせないデータを区分する際に、工夫が必要となる。また、フィードバックを行う時のコストが、過去のフィードバック回数に関わらず一定である必要がある。仮にフィードバック回数に応じてコストがかかると、抽出の速度が徐々に低下する。よって、過去の計算結果を再利用するか、追記を行うことで実行できるフィードバックアルゴリズムの考案が必要となる。今後は以上の取り組みを行いたい。

謝辞

筑波大学システム情報工学研究科 池嶋俊氏には、実験に使用したキーロガーを提供してして頂きました。ここに深く感謝いたします。

文献

[1] 水口 充, 竹内 友則, 倉本 到, 渋谷 雄, 辻野 嘉宏: デスクリークにおける忙しさの自動推定, ヒューマンインタフェース学会論文誌, Vol. 6, No. 1, pp. 69-74, 2004

[2] Anind Dey and G.D. Abowd. Towards a Better Understanding of Context and Contex-Awareness. Technical report. GeorgiaTech, 1998

[3] Salber, D., A. Dey and G. D. Abowd, The Context Toolkit: Aiding the Development of Context-Enabled Applications. In proceedings of CHI'99, pp. 434-441. Pittsburgh, 1999

[4] R. Gaines, W. Lisowski, S. Press, and N. Shapiro, Authentication by keystroke timing: some prelimary results, Rand Report R-256-NSF, Rand Corporation, 1980

[5] FRANCESCO BERGADANO, DANIELE GUNETTI, and CLAUDIA PICARDI, User Authentication through Keystroke Dynamics, 1, ACM Transactions on Information and System Security, Vol.5, No.4, pp.367-397, 2002

[6] Saleh Bleha, Charles Slivinsky, and Bassam Hussien, Computer-Access Security Systems Using Keystroke Dynamics, IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol.12, No.12, 1990

[7] http://www.biopassword.com/