USER PROFILING BASED ON MOUSE MOVEMENT

Lorand Dali
Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel: +386 1 477 3144;
e-mail: lorand.dali@ijs.si

ABSTRACT

The paper presents an approach to user profiling based on the user's mouse activity. The hypothesis which we try to verify in this work is that everybody uses the mouse in a specific way, and therefore a user model can be learned from the mouse activity. The aim of the user model is to recognize who a user is, given the way he uses the mouse. The data, collected from 10 users, consists of Windows events which were fired as a result of mouse activity. The described user profiling could be applied in security systems and for personalization.

1 INTRODUCTION

Authentication is a very important service for the security of a computer system. Many authentication methods such as passwords, fingerprints, iris recognition, face recognition, voice recognition have been used. We propose an authentication method based on mouse activity. The work builds on the assumption that usage of the mouse is specific to individual users. The advantage in using the mouse for authentication is that data is plentiful and cheap to collect and analyse. Also, mouse movement is harder to fake than a password.

The remainder of the paper is organized as follows. Section 2 describes the data collection and preprocessing, Section 3 talks about the experiments and the evaluation, and Section 4 draws the conclusions.

2 DATA COLLECTION AND PREPROCESSING

The data analyzed consists of events triggered by mouse usage on a Windows system. To help collect this data, ten users agreed to track their mouse over the time period of about a week. The users (4 female and 6 male) will be refered to with the fictional names of: Ana, Brian, Claudio, Dorina, Elsa, Flavia, Gerard, Holger, Iain and Jeffrey. During the mouse tracking, each event triggered by the mouse was recorded together with the following attributes:

- **Event Type.** Possible event types are: Move, LeftButtonUp, LeftButtonDown, RightButtonUp, RightButtonDown and MouseWheel.
- **Mouse Position.** X and Y coordinates of the mouse position on the screen at the moment when the event was triggered
- **Timestamp.** The time (in milliseconds) when the mouse event occurred.

Thus the raw data of about 800 000 events per user was collected. In what follows, the preprocessing steps applied to this data will be described.

2.1 Dividing into Gestures

By gesture I mean a sequence of events which happen close to eachother in time. A gesture ends when the user makes a break longer than one second between two successive mouse events. For each user we obtain a number of gestures somewhere between 5000 and 10 000.

2.2 Annotation of High Level Events

After segmenting the data into gestures, we annotate each gesture with higher level events such as: left click, right click, double click, movement, scroll, drag and drop. The higher level events are semantically more meaningful. The annotation is done based on a few simple rules.

Table 1 Annotation Rules

Annotation	Rule
LeftClick	LeftButtonDown → LeftButtonUp
RightClick	RightButtonDown → RightButtonUp
DoubleClick	LeftClick → LeftClick
Movement	Move → Move → … → Move
Scroll	MouseWheel → … → MouseWheel
DragNDrop	LeftButtonDown → Movement → LeftButtonUp

2.3 Approximating the Path of Mouse Movement by Line Segments

Each Movement as well as DragNDrop event is composed of a sequence of mouse moves (i.e. a sequence of points on the screen). The movement path made of a sequence of points is approximated by a line segments. An important observation is that linear regression, the usual way of fitting a line to a set of points cannot be used in this case because we have the points as a sequence in time, not as a set, and because of this the direction of the line we fit is important. In absence of a standard method to approximate the path by line segments, a simple algorithm was implemented.

```
function APPROX-PATH(points)
returns: lines
lines ← []
for i in [1,3,5,…n] do
   line ← MakeLine(points[i], points[i+1])
   lines.push(line)
end for
while not done do
   forall angle in angles(lines) do
      if angle < 30° do
         MakeLine(angle.prevLine,
                  angle.nextLine)
      end if
   end forall
end while
```

First between each pair of successive points we draw a line, then as long as there are angles smaller than 30° we join the two lines together into a single line. Figure 1 shows an example of a segment approximation of a path. The red dots show mouse positions on the path.

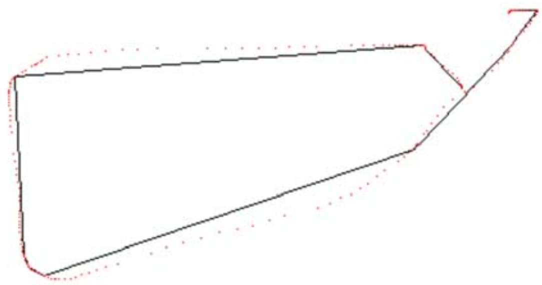

Figure 1 Line segment approximation of a path

Every line segment thus obtained can be described by three parameters: the length, the direction (i.e the angle between the segment and the OX axis), and the speed with which the user moved the mouse on that segment. Only the lines with length at least 20 pixels are taken into account.

2.4 Discretizing Line Segment Parameters

Each of the segment parameters (length, angle and speed) is a continuous value which is discretized. The angle is discretized into 12 values, each of the 12 values covering an angle of 30°, thus we obtain angles of (0° - 30°, 30° - 60° etc.). The segment lengths are discretized into 5 values (very short, short, medium, long, very long). The speed values are also discretized into 5 (very slow, slow, medium, long, very long). The thresholds for discretizing length and speed values are obtained by looking at the entire training data. For instance the threshold for very short is the length of the line which is longer than 20% of the lines.

3 EXPERIMENTS

The experiments try to find out ways of recognizing the users. This means that given some mouse events, can we determine to which user these events belong? We look at this problem as a multi-class classification problem, in our case we have 10 classes (the 10 users).
The data is divided into training data and test data. The training data consists of 70% of the gestures from each user. The test data is made of the remainder of 30% gestures from each user. The gestures are taken ordered by time and the ones from the test data are the gestures generated last.

3.1 Mouse Activity Maps

Because for each event the position is known where the mouse was when the event was triggered, we can draw activity maps to discover the areas with most activity.
Figure 2 and Figure 3 show move maps of two users, Flavia and Ana. The differences can be noticed quite easily. Flavia has a lot of movement on the right of the screen, while Ana uses the bottom part more and the right almost at all. Interesting features like the task bar, window title bar, start button, minimize and close buttons can be recognized. In a discussion with Ana she explained that she has widgets which she very rarely uses on the right part of the screen; this is why this part appears more white than the rest. Flavia also confirmed that her task bar and start button are on the right instead of at the bottom.

3.2 Mouse Movement Models

As described in the section about data preprocessing, the mouse movement is approximated by a sequence of segments. Each segment has a length, an angle (direction) and a speed. Based on these three features and of the time-dependency of the segments, user models can be built.
A user model is a vector whose entries are probabilities of the segment attributes taking certain values. For example, an entry could be the probability that for the user Flavia a segment has the length 'very short' and the speed 'fast'. Another entry could be the probability that the angle of a segment is between 30° and 60°. The complete list of features whose probabilities are computed is:

- length
- angle
- speed
- length + angle
- length + speed
- angle + speed
- length + angle + speed

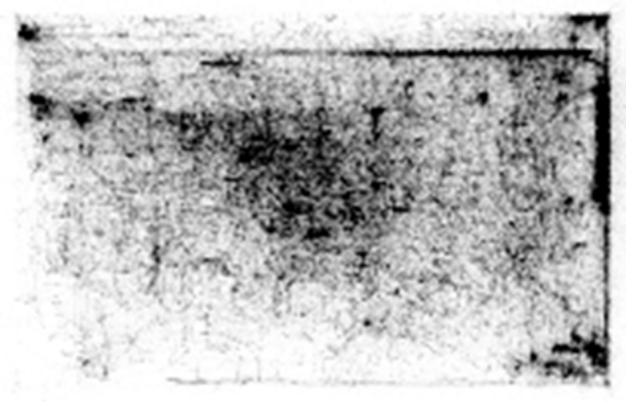

Figure 2 Flavia's Move map

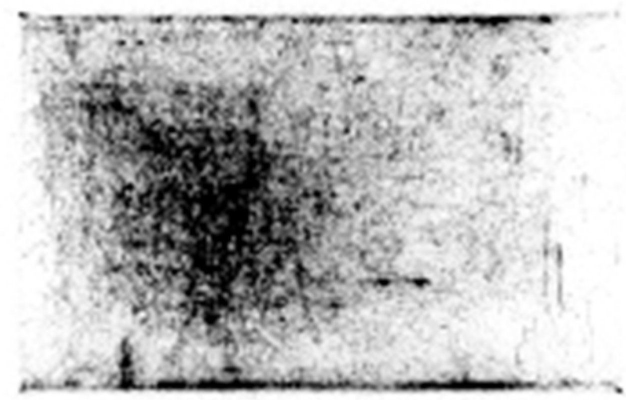

Figure 3 Ana's Move map

Moreover, the values of the previous segments are also taken into account. For instance we could have as a feature the probability that the current segment length is 'long' and the previous segment length is 'short'. For length and for speed the values of up to 5 previus segments are taken into account, for angle up to 3, for length+angle, length+speed and for angle+speed only the previous 2 and for length+angle+speed no previous segment is taken into account as that would make the feature vectors very sparse.

Having a user model expressed as a vector, we can compute the distance to other users. By finding the closest user to each user the directed graph in Figure 4 is obtained. There are two connected components, one of which has mostly male users (white nodes). In the other connected component the ratio between male users and female users is equal. An observation to make is that if user B is the closest to user A it is not true in general that also user A is closest to user B.

The experiments consist of computing a model from the training data for each user. Then, from the test data of each user we compute several test models. The test model is classified by finding the training model closest to it. From the test data of each user 100 sequences of segments are sampled. For each of these samples is classified and then the accuracy is computed.

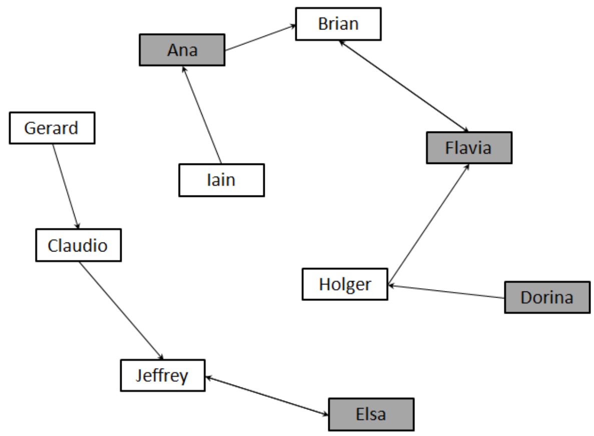

Figure 4 All users and the smallest distances between them

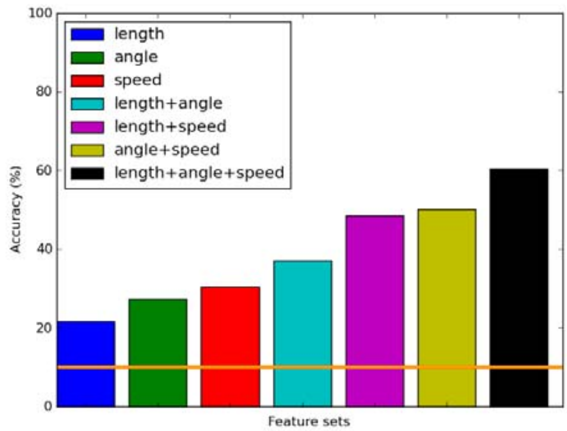

Figure 5 Accuracy of different feature sets

We try to find out which features are most helpful for the classification. In Figure 5 the accuracy of models based on seven different time independent feature sets are shown. The model which takes into account only the length of a single segment performs worst. It has an accuracy of about 20%. The best accuracy of about 60% can be obtained by considering the joint probability of length, angle and speed. For each of the test samples a sequence of 100 segments was used.

Having found that all three attributes of a segment have to be used for accurate classification, two important questions remain still open. How many segments per sample are enough? Can we improve the accuracy by taking time dependency into account?

To find the answers to the first question we have varied the number of sequences in a sample from 10 to 1000. Two feature sets are considered: one of length+angle+speed without N-grams, and the other taking into account all features with N-grams. Figure 6 shows that the model which does not take N-grams into account has an accuracy

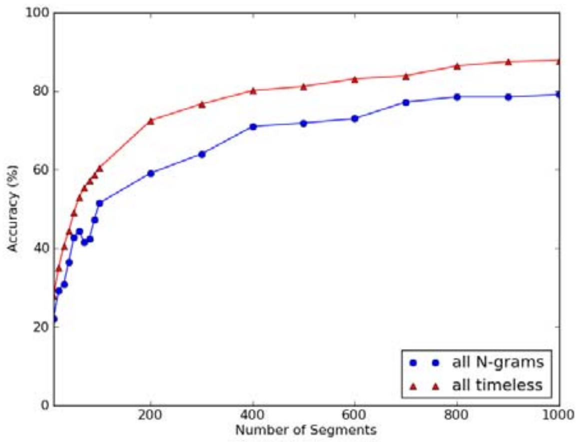

Figure 6 Increase of accuracy with the number of segments in the test models

of about about 10% better than the other. Another thing we notice is that the accuracy increases as the number of segments increase. Until around 200 the accuracy increases fast after which it increases at quite a small rate.

We have also noticed that the classification accuracy varies a lot from one user to another. For instance the accuracy of classifying data from Gerard at 400 segments is 93% while for Flavia only 70%.

4 CONCLUSIONS

We have presented a couple of methods for analyzing data obtained from mouse events produced by the activity of 10 users. We have foused mainly on move events. The experimental results show that the user which produced given mouse data can be determined with high accuracy. For this, all parameters of segments (length, angle, speed) should be taken into account and at least 200 segments are necessary to determine the correct user reliably. Surprisingly time features did not help in the classification but ‘confused’ it instead.

For the future, we plan to extend the user models with other features aside from movement. Also segments of smaller length could prove to be important, and a finer grained discretisation of length, angle and speed values might be necessary.