CHALMERS

# Identifying Users Based on Use of Input Devices

*Master of Science Thesis in the Programme Computer Science: Algorithms, Languages and Logic*

ERIK ALEXANDERSSON
GIANFRANCO ALONGI

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, June 2009

Identifying Users Based on Use of Input Devices

ERIK ALEXANDERSSON
GIANFRANCO ALONGI



Examiner: PETER DYBJER

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

[Cover:
MRI image of the human brain, mouse and keyboard and self organizing maps produced from test-subject interaction with data collection page.]

Department of Computer Science and Engineering
Göteborg, Sweden June 2009

## Abstract

The need for other security measures besides secret keywords has highlighted the possibility to base a biometric security system on input device patterns. This report explores the feasability of using mouse and keyboard biometry in an online security system. After examining prior work in this area, an online test environment was set up where subjects could remotely complete experiment sessions by interacting with a web-site.

Artificial Neural Networks (ANN), k-Nearest Neighbor (k-NN) and Self Organizing Maps (SOM) were combined into profiles which predicted sample validity. A final system accuracy of 0.92% False Accept Rate (FAR) and 43.16% False Reject Rate (FRR) was achieved. The best profile in the system achieved a FAR of 0.8% and a FRR of 0.0%. When one of the 19 subjects was browsing the online test environment, the system could with an accuracy of 54.1% correctly identify the user. Examining the results, it is clear that data of a higher quality is needed and alternative collection methods should be explored. It seems feasible to recognize a registered user solely based on keyboard, mouse and browser variables.



**Keywords:** User Identification, Biometric Security, Online, ANN, k-NN, SOM.



## Sammanfattning

Behovet av andra säkerhetsmekanismer utöver lösenord har banat väg för möjligheten att basera ett biometriskt säkerhetessystem på användandet av inmatningsenheter. Den här rapporten utforskar möjligheten att använda mus- och tangentbords-biometri i ett online-baserat säkerhetssystem. Efter en granskning av tidigare arbete inom området så skapades en testmiljö där användare genomförde experimentsessioner genom att interagera med en webbplats.

Ett system kombinerade Artificiella Neurala Nätverk (ANN), k-Nearest Neighbor (k-NN) och Self Organizing Maps (SOM) till profiler som bestämde giltigheten i ett sampel. Det slutgiltiga systemet uppnådde en säkerhet på 0.92% False Accept Rate (FAR) och 43.16% False Reject Rate (FRR). Den bästa profilen i systemet uppnådde en säkerhet på 0.8% FAR och 0.0% FRR. Systemet kunde även förutsäga vilken utav de 19 användarna som besökte webbplatsen med en säkerhet på 54.1%. När man granskar resultaten så är det tydligt att det behövs högre kvalitet på den insamlade datan. Andra metoder för insamling behöver också undersökas. Slutligtvis verkar det möjligt att känna igen en registrerad användare enbart utifrån mus, tangentbord och variabler från webbläsare.



**Nyckelord:** Användaridentifiering, Biometrisk säkerhet, Online, ANN, k-NN, SOM.

# Preface

This report explores the feasibility of identifying users with the use of information gathered from user interaction with a web page. The web page used in the experiment was written in PHP and the code used to gather data from users was written in JavaScript using jQuery [1]. Ruby was used in the analysis and classification of data.

The Master's Thesis proposal was presented to the authors by Nils Svangård at ICE House, who also worked as supervisor on the Master's Thesis. Chalmers University of Technology accepted the proposal as a Master's Thesis in Computer Science and Engineering, and Peter Dybjer was appointed examiner for the Master's Thesis.

Thanks to Peter Dybjer for the excellent work as examiner. Additional thanks to Harald Hammarström for taking time to discuss different approaches as well as general ideas. We would also like to thank all test persons that were willing to help us by using the test page and Nils Svangård for the supplied servers.

# Contents

# 1 Introduction

## 1.1 Background

Learning user behavior is of interest for security reasons, for example to prevent the illegitimate use of an account and hinder session hijacking.

The problem of user identification based on other means than known secret keywords is called *biometric security* [2] and utilizes techniques such as voice recognition, iris analysis and finger print analysis. These techniques are difficult and expensive to use since they require additional specialized hardware other than mouse and keyboard; thus these techniques do not to adapt well for widespread E-commerce.

Recent computer security reports [3–6] highlight the increased interest in session security based on inputs from peripheral units and evaluate the proposed security models based on these techniques. Due to the increased amount of spyware and other digital attacks, there is now an interest from the E-commerce to present stronger security to their users where the session is not only bound to the user's secret keywords, but also bound to their behavior.

The *main problem* to overcome for a behavior-based online security system is to successfully learn user behaviors (profiles) and correctly map these profiles to a known registered user.

The learning of *user behavior* requires large amounts of collected data which can be analyzed to find biometric features which are fed to the security system. Collecting data is one of the difficult tasks which requires access to mouse-motion/keyboard-typing data. The other more problematic challenge is to analyze and extract patterns in the data which can be used for user-identification purposes. Deriving qualitative conclusions regarding the visitors from such extensive data is even harder, and thus for such a task, machine learning is particularly well suited.

Assuming that each user interacts almost uniquely with a web-page and also handles input devices differently; a system incorporating machine-learning algorithms should learn these individuals features and recognize the user. Some features are hardware sensitive, such as mouse movements and keyboard typing [7]. Other features are more software dependent, such as browser variables. Hopefully some features are clean from external influence and depend solely on the user psychology.

An important measure used in biometric security systems are the False Accept Rate (FAR), False Reject Rate (FRR) and Equal Error Rate (EER). These are the per-say standard measures by which biometric security systems are compared and evaluated. The FAR of a biometric system is how likely an illegitimate user is to be identified as a legitimate user, while FRR is how likely an legitimate user is to be identified as an illegitimate user. If a system has several

settings which can be regulated and thus producing several different FAR and FRR, the systems EER is the situation where the settings produce equal FAR and FRR.

Roughly, given two systems A and B which have equal FAR and FRR, lowering system A's FAR while keeping system B intact will make system A more *secure*, while lowering system B's FRR and keeping system A intact would make system B more *user friendly*.

If the system correctly identifies users, it can be used to identify multiple accounts. Thus, not only creating strong session security for each individual user; protecting them from session hijacking, but also creating a strong fraud security for E-commerce by flagging accounts as duplicates if a user registers several accounts. Such a user identification system could also be used to detect cross-site users by cross referencing profiles.

## 1.2 Purpose

The proposed solution to the problem is to try to produce a system which identifies web page visitors by means of collected mouse movements, keyboard typing patterns, navigational patterns[1] and session variables. In order to prevent unauthorized access, the system should accurately predict if two unique visits were from the same individual. After performing literature studies on prior/related work, suitable algorithms for each part of the user features will be used together in a combined system to distinguish between users. The system should be able to correctly answer the question: Does the user logged in as X behave as user X?. The data for the tests will be collected from an experimental web page, where the user interacts with the mouse and keyboard. The users actions will be stored in a log file that should be easily parsed in order to extract features such as typing speed, mouse movement speed, etc. Which pages the user browses will also be stored in a web log. From the web log the users navigation paths will be extracted and analyzed.

To learn the users behaviors, the algorithms should be of the class machine learning/artificial intelligence. There are however no specific preference on which algorithms to use, as long as the system can be brought into production and work efficiently.

## 1.3 Delimitation

After extensive search for related work it seemed as though there was no previously published work on combining mouse movements, keyboard typing and navigation patterns for user identification. However, extensive material was available on the possible subsystems incorporated within such a joint construct. This material covered mouse movement analysis [5,8–15], keyboard typing analysis [4, 6, 7, 16–23] and web navigation patterns extraction [24–35] separately in each report. The focus was therefore put on the use of mouse movement data,

[1] Which was later-on discarded.

keyboard typing data, navigation patterns (called click paths) and session variables as raw data for user identification.

The report does not cover the impact of identity-obfuscating techniques such as User-Data spoofing [36], Onion routing [37] and other proxy-based [38] methods. The users of these E-commerce systems will likely agree to activating the system in exchange for a more robust security. From what we have read and found [7, 16, 23], it seems hard to find a good (efficient and accurate) solution to the author attribution problem[2], even if given a large corpus. Since this work is aimed at the E-commerce arena, it is not likely that extensive amounts of texts will be available.

## 1.4 Prior work

### 1.4.1 Keyboard-biometric

The features used in [7] were mostly based on averages and standard deviations on keystrokes and on transition times between keystroke pairs such as digraphs[3] The transitions between keystrokes were measured in two ways: from the release of the first key until the second key is pressed and from the first keystroke to the second keystroke. A total of 239 features were identified and stored in a feature vector which was given to a Nearest Neighbor classifier using Euclidean distance. This system performed well when presented with long enrollment samples from each user. Given that E-commerce does not involve such extensive enrollment samples, this might not be a good choice for identifying visitors, since the sampled texts are too short and too sporadic. The results were also tightly coupled with the hardware in question during enrollment. Accuracy of user and sample matching under optimal conditions was greater than 98%.

In a study by [6], the data collection was performed with a higher accuracy than most other studies. Keystroke collection was performed on Operating System level (OS-level), with timing errors of down to $10 - 15$ ms. From this data, histograms of the keystroke durations (ignoring shift, alt and other special keys) were produced for each user. Features extracted were named "dwell"[4] and "flight"[5]. A k-Nearest Neighbor (k-NN) classification was performed on the feature vectors (histograms) and results showed a correct classification rate of user and existing profiles at 73%.

A more abstract analysis was performed in [20]; namely the "rhythm"[6]. The typing "rhythm" is independent of total time needed to enter a particular text, as long as the timing relation between typed letters is preserved. Di-grams were analyzed and from this analysis, the "rhythm" was extracted. No quantitative results were presented. The classification was performed through sorting and measuring the difference between the training set and the new rhythm.

[2] The problem of identifying authors based on text.
[3] 2-(di) letter sequences.
[4] Corresponded to keystroke duration.
[5] Corresponded to pause times between keystrokes.
[6] Corresponded to the time it took to write specific di-grams (two letter long sequences).

Fuzzy logic was used in [17] where samples were fuzzified and stored as templates. Interval length was considered and each template was classified into a fuzzy variable using Gaussian distribution as the membership testing function. A predetermined threshold value determined acceptance or rejection if membership of new fuzzified input data differentiated too much. The average final EER of this system was 20%.

Tri-grams[7] were timed and processed with the "A" measure in [4]. The duration of trigraphs referred to the elapsed time between the first key pressed and the third key pressed. Two arrays of trigraphs were then sorted according to the duration of trigraphs and hence a distance in terms of degree of disorder between these two arrays was counted as the sum of the number of disorder of each trigraph in the array. A Clustering Based Keystroke Authentication Algorithm (CKAA) was used to train and authenticate within clusters. FRR at 0% and FAR at 0.045%, taking 19s for authentication/classification of 15 users, might not scale well for E-commerce (hundreds of simultaneous users).

In a study by [18], intra key pressing times and key press durations were used as features. A fixed string was entered several times during which the features were extracted. Support Vector Machine (SVM) was the classifier and Particle Swarm Optimization (PSO) tuned the SVM parameters (which features to use). Reduction from full set of features to minimal set reduced the set with up to 77.59%. This also reduced the Classification Error to 1.57%. This approach could be too memory heavy and process demanding when using one PSO with an assigned SVM for each unique user when the user base is that of a large E-commerce site.

In [22] several Decision Trees (DT) were created with commercial software in parallel and by partitioning the input vectors amongst the DTs, the accuracy of each tree could be increased. The input vectors were constructed from key press and key release times from a fixed sequence of 32 letters typed in by each user. The Monte Carlo (MC) method was used to generate more data since the collected data was insufficient. Wavelet transforms were used to obtain the training subsets from the partitioned sets. Results of 9.62% FRR and 0.88% FAR were obtained by rigorous testing.

Duration of and interval between key presses were the selected features for use in [19]. A feature vector was created for each user and clustering using K-Means [39] was performed on which classification was possible using some kind of Euclidean distance. The novel idea introduced in [19] was the notion of Moving-/Growing-/Fixed- window samples. This window could be regarded as the memory with which the algorithm adapted to new data which was seen. Depending on the type of window, the window might change dynamically with success on authentication. Measured values for EER: Moving window 3.8%, Growing window 3.8%, Fixed window 4.8%.

k-NN, Artificial Neural Networks (ANN) and Bayesian classification were com-

[7] Three letter long sequences.

pared in [21] on the task of identifying users with keystroke timing as input data. The conclusion was that k-NN was suitable during initial classification (when little input is available) but one should switch over to Bayesian classification when the amount of data has increased. k-NN had an error rate of 2% when using small amounts of data, but increased to 4.7% as the amount of data increased. Bayesian classification had an error rate of 3.7% for the small initial data set and 3.3% for the increased data set. The ANN approach performed well with a moderate accuracy, but it would become too computationally cumbersome for larger inputs. Error rate was 2.3% for the small data set and 4.0% for the larger data set.

### 1.4.2 Mouse-biometric

Mouse acceleration can be used as a distinguishable feature to identify users. In [8] acceleration was one of the features which proved to be functional to identify users using signatures which were written with the help of the mouse. Geometric average mean was resistant to mismatching when comparing the features. The use of a dynamic database was a performance improvement over using a static database. 7% FRR and 4% FAR.

In [9], speed, deviation and angle were used as biometric features from mouse movements. The amount of sampled mouse points were reduced by focusing on dense regions only. [9] concluded that more research had to be done and that a Hidden Markov Model maybe could be used to increase accuracy.

Mouse movements were extracted in [3] and feature vectors were created on which nearest neighbor with Euclidean distance was used. Results varied between $66 - 75\%$ accuracy with high error rates when classifying. A base set of 206 samples from 10 users was generated by pressing 25 randomized buttons. One of the main ideas in [3] was to use Dichotomy Model Transformation on the values.

The raw feature set for mouse identification in [10] was derived from the following sample points: mouse curve length in pixels and in points, time to complete a curve (milliseconds) and average velocity (pixels/ms). From this set of raw data, the following was computed

1. The average and standard deviation of these click durations
2. The average and standard deviation of these transition times
3. The average and standard deviation of the curvature measurements
4. The average and standard deviation of the transition velocities
5. The average and standard deviation of the transition accelerations

For classification, 1) - 5) constituted a feature vector. Given a new mouse trail with extracted data; classification against the known feature vector was computed with k-NN, with a result of 92% accuracy.

Similar results were shown in [5] as previously reported in [10]. This enforced the belief that the features used in [10] were reasonable.

Instead of focusing on mouse curvature and transition velocities [11] used mouse actions as features to classify users. ANN were introduced as the classifiers. Designating a small and fast ANN to each user, the input to the ANN was a feature distribution set consisting of

1. Distribution of Mouse actions: Mouse movement, Drag and drop, Point and click
2. Distribution of Traveled distance
3. Distribution of Time elapsed on each action

According to [11] other features could also be used. The produced ANN gave confidence rates, degree of similarity which were used to classify users. Results show that a FAR of 2.4% could be lowered to 0.4% and FRR of 2.4%. A Lower FAR often results in higher FRR. The concept Mean Time To Alarm (MTTA) was introduced, this is the time the system needed to detect a deviation.

In [15], mouse motion generated signatures were sampled and vectorized. A Genetic Algorithm (GA) was used to create forged signatures which were similar to the original vectorized data. These forged signatures were given to an ANN which was trained to reject the forged copies and only accept the real signature set. Thus the ANN would recognize false signatures as well as valid. The system had an overall FAR of 8.5% and an overall FRR of 38.6%.

"Curve straightness" was measured in [12] by sampling and analyzing mouse curves using a Cubic B-Spline. Other more advanced features could also be extracted. Using this new measure (and others as well), the mouse curves were classified into certain types. These classes were then used to construct a profile consisting of a histogram which represented the particular users mouse curve classification frequencies. A new curve could then be analyzed, classified and tested against the users distribution profile using Euclidean distance between the profiles. Error rates ranged from 1% to 44%.

The raw data used for re-authentication in [13] consisted of

1. Mouse "wheeling"
2. Mouse single-click
3. Mouse double-click
4. Mouse movements

Angle, Speed and Distance were calculated from the mouse movement set and used as classification features. A commercial Decision Tree algorithm (C5.0) was used for learning and classifying. Results produced showed a FAR of 0.43% and FRR 1.75%. [13] also stated that anomaly detection would fail to detect an attacker who did not use the mouse at all if the legitimate user seldom used the mouse.

### 1.4.3 Click path analysis

From [28] it became clear that in the action of mining web patterns, it is better to regard the events as intervals instead of points. Intervals have $(starttime, endtime)$ and thus hold more data than just a time index T. One could also use stochastic algorithms for navigation pattern extraction/modeling.

In [25], an ant colony model was used to construct a user-specific prediction graph with the users pheromone trails. This graph could be used to predict the user's next page visit *given a specific trail*. This model proved to have accuracy of 65% to 87% in predicting the next page when using sessions with lengths of up to 8 pages.

Another ant algorithm was used in [27]: The Leader Ant Algorithm (LAA), which focused on partitioning the web log data into clusters of similar clusters. Similarity was based on IP, session variables and times. Results were determined by measuring cluster tightness and intra cluster distances (Davies-Bouldin index). When testing LAA against Linearized Fuzzy C-Medoids (LFCMdd) and Variant of Fuzzy C-Medoids (VFCM), the results were as follows LAA := $0.38 - 0.88$, LFCMdd := $0.64 - 0.94$ and VFCM := $0.67 - 0.98$, measured in Davies-Bouldin index, lower is better.

The Active Ant Colony Clustering algorithm ($A^2C^2$) [40] used for incremental Web Usage Mining (WUM) mined web logs for features which were used for representing the objects as vectors. The algorithm was adaptive and handled new data which was added later on. The F-measure[8] of the proposed $A^2C^2$ algorithm ranged from 0.875 to 0.922 with number of sessions from 200 to 6000 thus making $A^2C^2$ near perfect for this range of sessions.

Another ant algorithm implementation AntClust (AC) [34] modeled ants as combinations of data-set objects and templates. The templates controlled the behavior of the ants throughout the simulation where ants meet and interact. Ants clustered into nests which became the clustering of sessions. AC was tested with both artificial and real data sets against K-Means and AntClass. AntClust performed well, and did not rely on a predefined guess as K-Means. The rate of which the AntClust algorithm produced the expected clustering ranged from 54% to 95%.

Genetic Algorithms (GA) also seemed to work within the field of navigation pattern analysis. The work in [29] was based on automatic discovery of the sequential accesses from web log data files by the use of genetic algorithms. Web log data was migrated to a database and redundant data were removed. A specialized GA was used where chromosomes represented page access sequences (sessions). An extremely high mutation rate and low crossover rate was used. Only individuals with a sufficiently high fitness were passed on to the result queue, others were sent to next generation. Highest fitness obtained was 0.8. Fitness was measured as support*similarity-rate.

[8]The F-measure relates each clusters intra distances to the clustering inter distances (F-measure of 1.0 is perfect).

On the topic of stochastic modeling, one may also choose a pure Markovian Model for this purpose, [31] modeled the users web navigation path as a Markovian process, and tried to predict the next page from this. No results were presented.

One of the pure clustering approaches [24] clustered the log data into web access patterns and used a combined system of Feed Forward Neural Network (FFNN) and PSO. Web logs were mined for URLs and access times, features were fuzzified and sessions were transformed into a fuzzy vector. The fuzzy vectors were used as input to an Learning Vector Quantization (LVQ) - 2 layer FFNN, PSO was then applied for parameter tuning for the LVQ. Clustering results were measured in the Davies-Bouldin index and proved good (0.496) [lower is better].

Fuzzy logic were used in [30] to construct behavior patterns, but presented no qualitative prediction or accuracy analysis.

Another rule extracting algorithm [35] mined user navigation patterns as precedents and sequents through click path analysis. A Frequent Transition Matrix (FTM) was created from web log data, pruned and processed into a Frequent Behavior Path tree (FBP-tree). From the FBP-tree, rules were calculated in the form PathIn $\rightarrow$ PathOut, where PathIn and PathOut were sequences of pages. The rules were probabilistic in the sense that if you traversed all the pages in PathIn, you would likely traverse the pages in PathOut with a high probability. No consideration to chronological order of page visits was made. Thus the PathIn was a conjunction of preconditions and PathOut was a conjunction of results. All algorithms were custom-made, but in validation, Advanced Log Analyzer was used. From experiments, rules with as high probability as 99.2% could be obtained, that is, if PathIn was traversed, with 99.2% probability PathOut was visited as well.

By considering page access and time spent on each page, the User Access Matrix (UAM) and Preferred Navigation Tree (PNT) algorithms presented in [33] seemed to learn and predict the visitors habits with an accuracy of 100% to 70% given paths of 50 to 220 pages. This approach seemed very successful.

N-gram analysis of page visits using Temporal information [26] yielded a next page prediction of approximately 55%.

## 1.5 Report layout

Chapter 2 describes how the data was collected, how the features were extracted from the data, which algorithms were used and how the complete system should work. Chapter 3 presents the results and Chapter 4 the conclusions and discussion on the results. The appendices contains additional information and results for the interested reader and should therefore be thought of as a complement to the report.

1. Appendix A contains detailed tables with results which are already spec-

ified in the report.

2. Appendix B contain supplementary images for the interested reader searching for more clarification.

3. Appendix C roughly explains algorithms used in the report, with references to more detailed explanations.

4. Appendix D is a short list of frequent abbreviations with their respective meaning.

# 2 Collection to Classification

## 2.1 Data acquisition

A data collection experiment was set up to be able to acquire the amount of data needed for feature extraction and algorithm application. The experiment was performed as follows.

1. A group of people were contacted and asked to perform at least 10 data collection experiments each by visiting a particular page (discussed in *The data collection page*). Each data collection experiment consisted of 10 iterations and took approximately $5 - 12$ minutes to complete. Thus once a subject had completed all his/her 10 (or more) experiments, the subject would have provided 100 raw data files (also called post-backs).
2. Subjects were told to use the same computer for each data collection session if possible. This was done to increase the consistency of the data since mouse and keyboard features have shown to be very hardware dependent [3, 7]. However not all subjects did this.
3. The subjects could access the data collection page during a whole week, thus enticing them to perform the data collections whenever they wanted to within that week. This was done to increase the chances of the subjects not loosing interest too much as well as letting the subjects establish their own pattern.
4. Once the collection experiment was finished, the collected data was analyzed. All errors and anomalies were corrected. This will be described in Section 2.1.3 (*Error correction*).

### 2.1.1 The data collection page

The data was collected from users interactions with an experimental web page. At the start of each session, the user was asked to enter his/her name and email as seen in Figure B.3, this data was coupled to a randomly generated alphanumeric session-string (hereafter called the session-ID) which persisted throughout the ten data collection cycles. The (hopefully unique) random session-ID was later on used together with the name and email to validate the classifications.

By coupling each generated data set to a session-ID, it was possible to couple web log entries to a specific IP even though the web log entries defined one IP and the post-back defined another.

The user was asked to repeat four steps 10 times to gather enough data from each user

1. **Text copying with the keyboard**. The first step was to re-write a short text that was taken randomly from The Project Gutenberg E-Book

The Adventures of Sherlock Holmes by Sir Arthur Conan Doyle [41] as seen in Figure B.4. This step was aimed at collecting keyboard data from the user, such as keystroke times, di/tri -gram times and distribution of special keys, etc.

2. **Number marking with the mouse**. In the second step, the user had to move the mouse over several numbers in order, as seen in Figure B.5, this was solely for mouse data acquisition, and the user was not allowed to proceed before this step was finished.
3. **Image viewing**. The third step was purely optional, and the subjects could choose to look at randomly selected images belonging to the previously selected category. For the first iteration, these images were selected as one image from each category. The intention of this step was to keep users longer on a page which exhibited images of interest. Thus, the view time would be bound to the interest. Figure B.6 shows a snapshot of this step.
4. **Category clicking**. The last step as seen in Figure B.7 presented the user with several category links. When the user clicked an image, the user was sent to the next page where the same steps were repeated, but with the exception that all images that were presented on the page came from the sole selected category.

The images shown were an attempt at avoiding the problem resulting from the controlled test environment since there was no easy way to acquire the user interest in such a predefined web-page hierarchy. The authors had to figure out how to capture such a feature as user interest in a particular subject. Naturally, users visit pages which fit their needs or is of interest in some way or another. It was also claimed that the time spent on the pages could be used as a user interest measure for that particular page [25, 30, 31, 33].

The assumption was made that the controlled test environment would render some of the algorithms useless depending on the view time [40]. It was also hypothesized that the controlled environment made the subjects bored and thus there was rather quickly no interest whatsoever. This lack of interest has also been stated by the subjects themselves after ending the experiment. This result together with the fact that all the algorithms for web navigation mining were not aimed at user recognition but aimed at interest path extraction or clustering made it hard to use or implement the Click Path data (which was qualitatively and quantitatively poor to begin with).

An attempt was made to perform user identification based on a Markov model, but the results proved to be poor. Thus it was chosen not to use or implement the Click Path data as part of the recognition.

### 2.1.2 Data is sent back to the server

As the user clicked a category in the fourth step, the gathered mouse, keyboard and session variables were sent back to the main server by means of a HTTP-POST. This was performed each time a click began a new cycle (set of four

tasks) and so the stored log of times and actions was kept small on the client side.

All data was sent in verbatim and was immediately parsed and stored into a specific session-ID catalog where all the data for a particular session was stored in separate files named by time instance. These files contained the raw data from the post-back. An example file could look like this

```
fN6tRwu0Z1Y8COKW3JMkWpHs0himOV
SWE
SWEDEN
129.16.195.51
Netscape Mozilla/5.0 (X11; U; ..... )
1280
800
24
2
null
Tue Feb 24 2009 11:54:41 GMT+0100 (CET)
8,547,353,1849
8,539,360,2037
9,[object HTMLTextAreaElement],2
```

where the first column's numbers corresponded to codes who's descriptions can be found in Table A.1.

### 2.1.3 Error correction

The error correction of the data was performed as follows

1. **Backwards time correction**. Entries in the raw log file which contained time entries in reverse order were corrected by setting the incorrect time instance as the latest correct time entry and adding the (currently seen) average time difference between all entries so far. Thus the average time difference between entries was kept unchanged. It is not known why this type of errors occured.

$$T_{i+1} = T_i + \frac{\sum_{j=1}^{i}(T_j - T_{j-1})}{i} \tag{2.1}$$

2. **Same time correction**. Entries in the raw log file where the mouse had performed two or more move actions during a very small time instance were considered erroneous. The correction was performed by adding one time instance to the following entry which reported the same time, increasing an accumulation counter and the accumulated value to subsequent non

erroneous entries.

$$\textbf{if } T_{i+1} \textbf{ equals } T_i \textbf{ let } T_{i+1} = T_i + 1$$
$$accumulated = accumulated + 1$$

All the following time entries which did not have any errors were then increased with the accumulated value.

$$\textbf{if } T_{i+1} \textbf{ does not equal } T_i \textbf{ let } T_{i+1} = T_{i+1} + accumulated$$

3. **Missing code/time correction**. Due to unknown problems, the interaction between the user and the information retrieving script caused some erroneous entries in the raw logs. Examples are entries without codes (missing 8 in second entry below)

   ```
   8,543,357,1987
   ,544,356,1976
   8,540,359,2024
   ```

   or entries without timestamps (missing number after last ',' in second entry below)

   ```
   8,543,357,1987
   8,544,356,
   8,540,359,2024
   ```

   Such entries were discarded since correcting such missing data would lead to creating (forging) data.
4. **Missing or more than expected entries in the web log**. As each session consisted of 10 iterations, 10 pages should be recorded in the web log as visited by the user. Due to what could possibly be explained as browser differences or the refreshing of the same page, some logs recorded more than 10 page visits for a user and some had less than 10. Such logs were modified to only contain 10 page visits. If one or more entries were missing, a prior entry was duplicated.

## 2.2 Feature extraction

When the user had completed a session (10 iterations with 4 steps in each iteration), the stored log files were automatically parsed by a program written in Ruby. The program extracted features for mouse, keyboard and stored the browser variables. For features that involved time durations, the mean and standard deviations were calculated. Click Path features such as navigation paths were extracted from the HTTP log on the server. The features were stored in four separate files named accordingly.

EMAIL.mouse(NUMBER + EXPERIMENT*10)
EMAIL.keyboard(NUMBER + EXPERIMENT*10)

EMAIL.session(NUMBER + EXPERIMENT*10)
EMAIL.weblogEXPERIMENT

Where $EMAIL$ is from the initial registration, $NUMBER$ is the particular iteration data number $n \in [0, 9]$, and $EXPERIMENT$ was the particular experiment iteration number $n \in [0, 9]$ (each person performed 10 data collection experiments). Only one web log file was created for each experiment since it was created upon termination of the collection cycles. The web log file held the mined Click Paths specific to that session-ID.

### 2.2.1 Extraction step

The features extracted from the mouse, keyboard, web log and session data can be seen in Tables 2.1, 2.2, 2.3, 2.4. Features extracted were chosen with support in the reports which used them for their particular algorithms. Thus we used the features and tested how well they worked for the proposed techniques and algorithms taken under consideration.

| Number | Feature | Description |
|---|---|---|
| 0 | Move move distr | % of all mouse actions that were mouse movements |
| 1 | Left click distr | % of all mouse actions that were left clicks |
| 2 | Middle click distr | % of all mouse actions that were middle clicks |
| 3 | Right click distr | % of all mouse actions that were right clicks |
| 4 | Double click distr | % of all mouse actions that were double clicks |
| 5 | Mouse over distr | % of all mouse actions that were mouse overs |
| 6 | Mouse-wheel distr | % of all mouse actions that were mouse wheel actions |
| 7 | M Stroke time | Mean time used for a mouse stroke (ms) |
| 8 | M Stroke length | Mean length in pixels for a mouse stroke |
| 9 | M Mouse-wheel scroll steps | Mean mouse wheel steps in each mouse-wheel-stroke |
| 10 | M Time on Left Click | Mean time used for each left click (ms) |
| 11 | M Time on Middle Click | Mean time used for each middle click (ms) |
| 12 | M Time on Right Click | Mean time used for each right click (ms) |
| 13 | Mean stroke speed | As the name implies (pixels/ms) |
| 14 | StD stroke time | Standard deviation time for the stroke times (ms) |
| 15 | StD stroke lengths | Standard deviation of the stroke lengths (pixels) |
| 16 | StD Mouse-wheel scroll steps | Standard deviation of the mouse wheel scrolls (amount) |
| 17 | StD time on Left Click | Standard deviation time for left clicks (ms) |
| 18 | StD time on Middle Click | Standard deviation time for middle clicks (ms) |
| 19 | StD time on Right Click | Standard deviation time for right clicks (ms) |
| 20 | StD time on stroke speed | Standard deviation time for mouse stroke speed |
| 21 | Mouse vector data (avg speed) | Average mouse movement speed in each direction |
| 22 | Mouse vector data (num moves) | Distr of mouse moves in each direction |

Table 2.1: The extracted and calculated mouse data from each user mouse data file.

| Feature number | Feature |
|---|---|
| 0 | Mean Key-press time |
| 1 | Mean pause time between keys |
| 2 | Mean Di-gram time |
| 3 | Mean Tri-gram time |
| 4 | Mean time between key presses |
| 5-24 | Mean common Di-gram times |
| 25-34 | Mean common Tri-gram times |
| 35 | Std-Deviation Key-press time |
| 36 | Std-Deviation pause time between keys |
| 37 | Std-Deviation Di-gram time |
| 38 | Std-Deviation Tri-gram time |
| 39 | Std-Deviation time between key presses |
| 40-59 | Std-Deviation common Di-gram |
| 60-69 | Std-Deviation common Tri-gram |
| 70 | Arrow-Up distr |
| 71 | Arrow-Down distr |
| 72 | Arrow-Left distr |
| 73 | Arrow-Right distr |
| 74 | Esc distr |
| 75 | Tab distr |
| 76 | Shift distr |
| 77 | Ctrl distr |
| 78 | Alt distr |
| 79 | Space distr |
| 80 | Del distr |
| 81 | Backspace distr |
| 82 | Enter distr |

Table 2.2: The extracted and calculated keyboard data from each user keyboard data file.

| Variable/Feature |
|---|
| IP Address |
| User Agent String |
| Category distributions |
| Category navigation history |
| Category transition matrix (occurrences) |
| Category transition matrix (probabilities) |
| Category viewing times |
| Fictional HTML page transitions |

Table 2.3: Features and variables extracted from the web log.

| Number | Variable |
|---|---|
| 1 | Country code |
| 2 | Country name |
| 3 | IP Address |
| 4 | User Agent String |
| 5 | Screen width |
| 6 | Screen height |
| 7 | Color depth |
| 8 | History length |
| 9 | Visit time and date |

Table 2.4: Session variables stored in .session files.

### 2.2.2 Mouse feature extraction

The mouse movements stored in the raw logs consisted of the mouse movement action code 8, the final x and y coordinate on the screen, and the corresponding millisecond (in reference to data capture start) the movement was performed. Thus, mouse movement data could be as follows.

```
8,548,353,1833
8,547,353,1849
8,546,353,1870
```

To extract mouse strokes $S$ from this data, a maximal pause interval between time values was chosen. This time interval would separate clusters of mouse strokes. The chosen interval value was

$$T_{mouse-stroke} = 280 \text{ ms} \tag{2.2}$$

This value was chosen based on the result of empirical testing. The empirical testing was performed by the authors iteratively generating fixed amount of mouse strokes, parsing the generated data and evaluating how near the parsed system estimated the amount of mouse strokes to the actual (by the authors perceived) amount of mouse strokes. Thus, a mouse stroke $S$ was defined as a chronologically ordered set of points in $(x, y)$ in $\mathcal{R}^2$ such that no time difference between each pair of *consecutive* points was greater than $T_{mouse-stroke}$.

$$S = \{p_i | p_i = (x_i, y_i) \wedge |Time(p_i) - Time(p_{i+1})| < T_{mouse-stroke}\} \tag{2.3}$$

The size of a mouse stroke $S$ was then the amount of points $p_i \in S$. Mouse strokes with less than 4 consecutive points were ignored [10].

$$\text{size of S } = |S| \tag{2.4}$$

The length of the mouse stroke $S$ was defined as the arc length of the curve.

$$\text{length of S} = \sum_{i=2}^{|S|} |p_i - p_{i+1}| \tag{2.5}$$

Likewise, the time Time($S$) of the mouse stroke $S$ was defined as the difference in time between the last and the first points in $S$

$$Time(S) = Time(p_{|S|}) - Time(p_1) \tag{2.6}$$

Based on the defined measures; mean stroke times, mean stroke lengths and standard deviations for mouse stroke times and stroke lengths were computed.

Despite the already mentioned features, two other feature vector were also extracted from the mouse movement data. These were called Mouse Vector Data, listed as 21 and 22 in Table 2.1. Each element in the vectors constitutes one of 64 angles in which average speed and movement distributions were recorded. For each movement into a direction, the system collected the movement speed in that direction and increased a movement counter. Elements in the vectors which represented that particular angle were then updated, a neighboring set of size 8 were also updated on each side with a smoothing function. A visual example of these vectors can be seen in Figures B.8 and B.9.

### 2.2.3 Keyboard data feature extraction

The raw data file contained key-codes indicating which key was pressed or released and at which time instance in relation to the beginning of the data capture that the event occurred. As for the mouse feature extraction, an interval dividing sets of key-presses was defined.

$$T_{key-press} = 2000 \text{ ms} \tag{2.7}$$

The pause between two keystrokes $k_i$ and $k_{i+1}$ was defined as the time between the release time of $k_i$ and the down press event time of $k_{i+1}$ [6, 18, 19, 22].

$$Pause(k_i, k_{i+1}) = Time(Press, k_{i+1}) - Time(Release, k_i) \tag{2.8}$$

From the way the pause is defined, it is possible to find negative pauses, this implies overlapping between $k_i$ and $k_{i+1}$ [6, 7, 18].

Thus, key-presses could now be clustered together into keystroke-sets $K$ consisting of an ordered set of key-codes $k$ such that no two consecutive $k_i$'s had a pause greater than $T_{key-press}$.

$$K = \{k_i || Pause(k_i, k_{i+1})| < T_{key-press}\} \tag{2.9}$$

These keystroke sets could then be used to analyze di-gram and tri-gram times [4, 7, 20]. A set of the 10 most common tri-grams and the 20 most common di-grams [42] were selected and their corresponding mean times and standard deviations were computed as well. The time of an N-gram was defined as the down press event time of the first key in the sequence and the release event time of the last key in the sequence. This was equal to the time of the keystroke-set $K$ consisting of the di-/tri-grams.

$$Time(K) = Time(Release, k_{|K|}) - Time(Press, k_1) \tag{2.10}$$

The typing speed could now be defined as the mean pause time. The smaller the mean pause time, the faster the typist.

Using the above defined measurements, the mean pause, mean general di-gram and tri-gram times, mean specific di-gram and tri-gram times and their corresponding standard deviations were computed [7, 22].

The distribution of the key-presses on the set of special keys was also recorded. The set of special keys was the set of { *Arrow-Up, Arrow-Down, Arrow-Left, Arrow-Right, Esc, Tab, Shift, Ctrl, Alt, Space, Del, Backspace, Enter*} [7]

### 2.2.4 Click Path features

The transitions between categories and view-times (for how long a visitor stayed on a page in ms) for each page were extracted from the web log. The category transitions were mapped to a fictional URL structure by the simple algorithm seen in Algorithm 1.

The chosen mapping generated from Algorithm 1 from category transitions to URL pages produced a spheroidal Deterministic Finite Automaton (DFA) where each new click on a category link delved deeper into that category, and each click on a new category link performed a jump to the most shallow level of that new category. This can be visualized for 2 categories A and B as seen in Figure 2.1.

The previously described URL mapped set of pages was stored together with auxiliary data that was intended to be used with other algorithms (which was previously explained to not be used).

### 2.2.5 Session data features

The extracted session data variables are listed in Table 2.4. This data was extracted by means of the jQuery [1] script on the page used for the data collection experiment. Note that no previous work based on session variables was found.

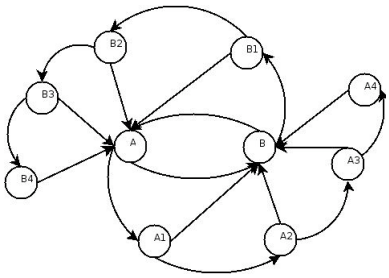


Figure 2.1: This example has a maximal depth of 4 but could be repeated for any depth. Note that $A$ and $B$ represent the first page for each respective category, and that $A1...A4$, $B1...B4$ represents the delving deeper into each category. Please note that this image does not comply with the formal definition of a DFA since edge labels, initial state and accepting states are not defined.

## 2.3 Comments on the data

Beforehand, it was suspected that the data acquired would be inferior in quality to the data used by the authors of the prior works. Much of the data in the prior work reports were created in a controlled environment with full access to OS timings, thus their reported times for the events would have less erroneous timing values and less noise.

It was also known that many of the reports used a repetitive and simple task for enrollment sample acquisition. This improved their quality of their data whilst our data was generated freely (within some limits) by our subjects in time and space.

It was therefore expected beforehand that the results would be worse than each independent result of the prior works which presented FARs down to 0.045% [4] and FRRs down to 0% [4].

The acquired data was partitioned as shown in Figure 2.2. The data set marked (A) was solely used for training and validation on each component, while the data set marked as (B) was used solely for final validation. Thus the validation of the system was performed on data from the same source, but which had never been used before.

## 2.4 Initial selection

Algorithms and methods which seemed feasible were selected from prior works and implemented in Ruby which was the requested language of choice from ICE House AB. During the time of implementation, a basic set of data was generated by the authors themselves as well as others. A total of six subjects had then supplied a set of 10 iterations each (corresponding to one full experiment per person), this was the basic testing-corpus on which the rough trial-evaluations were performed.

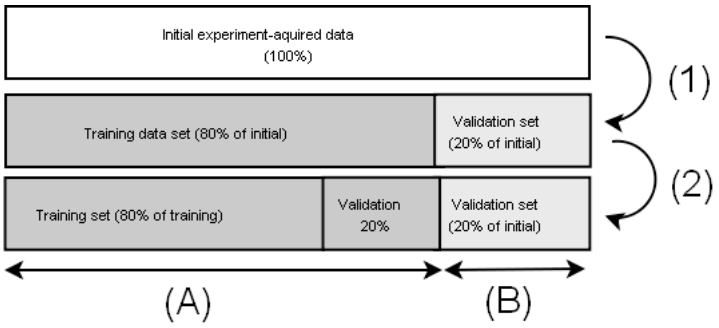


Figure 2.2: All the acquired, processed and feature-extracted data was partitioned into a main validation set which was used to validate the final system (1) and a training set which would be used to train and validate the internal components of the final system. The training set (dark gray) was then additionally split into a training set and a validation set (2). This was done so that the classifiers of the system would use the same training and validation set.

The initially implemented algorithms[1] together with the type of data they were tested with can be seen in Table A.2.

The basic testing-corpus was divided into two sets, one training set consisting of 80% and a validation set consisting of 20%. In this early testing phase it was concluded that ANN, ID3 and k-NN worked well for both Keyboard and Mouse data. It was also concluded that most of the Click Path algorithms should be left out due to being ill-defined and containing errors, also too many of the algorithms were simply clustering algorithms, something that was known in before-hand, but the authors felt that the clustering could maybe still be used in some way.

What remained after the initial selection were the algorithms seen in Table A.3. The chosen algorithms were selected based on the fact that they performed well enough (more correct classifications than expected amount of classifications based on pure chance) and had strong support in the reports.

The algorithms qualifying from the initial selection were then re-tested with the acquired data from the data collection experiment. This will be described in Section 2.5(*Additional testing*).

[1]It shall be said that numerous other Click Path directed algorithms were intended to be implemented but the reports in which they were stated either contained numerous errors or the algorithms were insufficiently described.

## 2.5 Additional testing

As the collection experiment was finished, 19 users had provided 10 experiments each. This data was corrected and partitioned into a 80% training and 20% validation set. A computational server was acquired for the algorithms to run on, during which the ANNs and SOMs were trained and the ID3's were created. The features previously chosen for each algorithm were also retested as well, since more data was available and this could change the outcome.

Let $\mathcal{I}$ denote a set of algorithm implementations where each implementation $I_i \in \mathcal{I}$ was of type ANN, SOM or ID3.

$$\mathcal{I} = \{I_1, ...I_u\} \tag{2.11}$$

Each implementation $I_i$ was positively trained for a particular class $i \in [1, u]$ (class was the same as a user) and negatively trained for the other classes. The measure of accuracy for $\mathcal{I}$ was both the FAR and the FRR for these implementations. Given a set of implementations of one particular algorithm and one set of validation samples $S$ where each $s_i \in S$ is denoted $(X, f(X))$, such that $X$ was the feature vector and $f(X)$ was the class of the vector $X$, the FAR and FRR for this particular validation set $S$ were calculated as follows.

$$S = \{(X, f(X))_1, ..., (X, f(X))_l\} \tag{2.12}$$

$$FR^2((X, f(X)) = \begin{cases} 1 & \text{if } Answer(I_{f(X)}) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

$$FRR(S) = \frac{\sum_{s \in S} FR(s)}{|S|} \tag{2.14}$$

$$FAR((X, f(X)) = \frac{|\{I_k | k \neq f(X) \text{ and } Answer(I_k) = 1\}|}{u - 1} \tag{2.15}$$

Thus, given a training set $S$ consisting of $l$ 2-tuples of samples $(X, f(X))$

The accuracy for an algorithm implementation for this set $S$ was defined as the total FAR and FRR

$$Accuracy(S, \mathcal{I}) = (\sum_{s \in S} FAR(s), FRR(S)) \tag{2.16}$$

This was the measure by which the additional testing was based to further select which algorithms were suitable based on the new (larger) data set. Running the additional tests, with features selected with support from the reports, the final set of suitable algorithms can be seen in Table 2.5.

[2]False Reject

| Algorithm | Keyboard | Mouse | Session | Click Path |
|---|---|---|---|---|
| Artificial Neural Networks | X | | | |
| K-Nearest Neighbor | X | X | | |
| Self Organizing Maps | | | X | |

Table 2.5: The final set of algorithms. As can be seen, no algorithm was present for Click Path data since discussions and results led the authors to exclude the low grade and erroneous Web log data. A cross (X) in the field of Keyboard, Mouse, Session or Click Path indicates that the algorithm would be used for that type of data.

The ANN implementation for Mouse data was excluded since repeated testing with different ANN configurations could not map the profiles correctly (or within acceptable bounds). Finally, more time could not be wasted on forcing good results out of a bad algorithm-data match and the authors had to conclude that ANN could not handle the mouse data as specified by the reports, given this particular setting. Initially the set of data was small, and the initially good results could be attributed to the low amount of classes and the low amount of samples.

ID3 was discarded since the results were very poor. Most ID3-trees created could not decide whether or not to accept or reject a sample. This can probably be attributed to the fact that ID3 is most suitable for binary decision problems with discrete values and does not handle real valued problems well [43]. In the reports that use decision trees, the commercial decision tree algorithm C5.0 was used which is superior to C4.5 [44] which in turn is an extension of the ID3 algorithm.

The final selection of algorithms also excluded the Click Path algorithms completely after trying a final Markov Model with smoothing unsuccessfully.

Note that the Self Organizing Map (SOM) algorithm was employed before the calculation servers were acquired. The basis for the SOM was that some kind of system for Session variable data had to be found/created. Luckily, a computer-security report focusing on analyzing digital attacks had employed the SOM with impressive results [45]. With this in mind, a SOM class was implemented and used to learn Session profiles.

## 2.6 Learning and classification

Once proper algorithms had been selected based on the initial and additional testing, the ANNs and SOMs were trained, the k-NN algorithm was used for keyboard and mouse. The results are presented with corresponding comments in Tables A.4, A.5, A.6 and A.7. For the final classification system, a voting system based on ANN was created. This will be described in Section 2.7.

### 2.6.1 Keyboard

The keyboard data was used by both the k-NN classifier and the ANN classifier. The best result for the k-NN classifier (K-value 3) when applied to the keyboard data had FAR 0.0239 and FRR 0.4309 can be seen in Table A.4. The number of features used by the particular result were $0, 1, 35, 36$. Their corresponding feature names can be found in Table 2.2.

The best result for the ANN was acquired by using the features numbered $70-82$ yielding a FRR of 0.3586 and a FAR of 0.4128 as seen in Table A.5.

### 2.6.2 Mouse

The mouse data was used solely by the k-NN classifier with the best result, FRR of 0.5526 and FAR of 0.0307 (K-value 9) as seen in Table A.6. The number of features used by the particular result were $7, 8, 10, 11, 12, 14, 15, 21, 22$. Their corresponding feature names can be found in Table 2.1.

### 2.6.3 Session

The SOM implemented for the session data used a vector of length 5 as models from the absolute value of the hashed User-Agent string of the browser, screen width, screen height, and the hour and minutes of the day at which the visit was made. These features are numbered $4, 5, 6, 7, 9$ in Table 2.4. The results for the SOM were a FRR of 0.3322 and a FAR of 0.1080. The hashing of the User-Agent string was made with the Ruby string hash function.

The IP was not used since there was a risk for over-fitting if this feature had been used.

The session data was used by the implemented SOM which produced the results seen in Table A.7.

## 2.7 Voting for acceptance

Once the algorithms had been trained for their respective data, each user **U** had a collection of classifiers $\{C_1, ..., C_k\}$ which constituted the profile $P$ of the user. Some kind of majority vote system had now to be implemented in order for a profile to work properly once a new sample $(X, f(X))$ was presented. It was also necessary to allow majority voting with bias towards the classifier with the best precision. In the most basic form, a profile $P$ would have accepted a sample $(X, f(X))$ as belonging to the profile if the majority of the classifiers classified the sample positively (classified the sample as belonging to the pro-

file). This was the basic way in which a profile would either reject or accept a sample. Let $Vote(C_i, X)$ denote the vote of the classifier $C_i$ for the sample $X$. The acceptance of the profile $P$ for the training sample $(X, f(X))$ can then be expressed as

$$Accept(P, (X, f(X))) = \begin{cases} + & \text{if majority of } C_i \in P \text{ vote } + \\ - & \text{otherwise} \end{cases}$$

However, some users provided more distinct keyboard data than mouse data, while others had more unique patterns in their generated mouse data. This was exploited by simulating bias in the classification system so that the most correct classifier would have the strongest vote when necessary. Some kind of learner which would learn the classifiers results for the input data had to be implemented. A simple 4 input, 2 hidden and 1 output ANN was used for this purpose where it was presented the results from the classifiers and the correct result for such a vote. This ANN was called the Meta-ANN and was responsible for whether a profile $P$ would accept the sample $(X, f(X))$. Thus, the Meta-ANN would have each classifier on an input node and output the final vote $+$ or $-$. The acceptance for a profile $P$ and the sample $(X, f(X))$ would now be expressed as

$$Accept(P, (X, f(X))) = MetaAnn(Vote(C_1), Vote(C_2), Vote(C_3), Vote(C_4)) \tag{2.17}$$

Each profile would then have their own personal Meta-ANN which was trained with the output resulting from the classifiers voting on a sample. Figure 2.3 shows a schematic view of the voting mechanism.

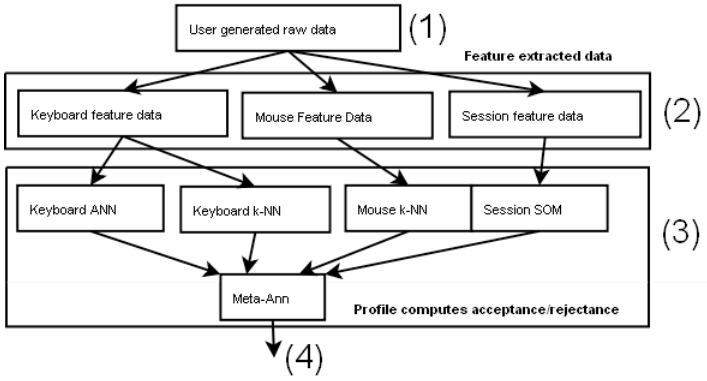


Figure 2.3: Acquired raw user data (1) is processed and features are extracted (2) which are given to the profile classifiers (3) (including Meta-ANN) which votes for acceptance of the raw data sample (4).

## 2.8 Complete system

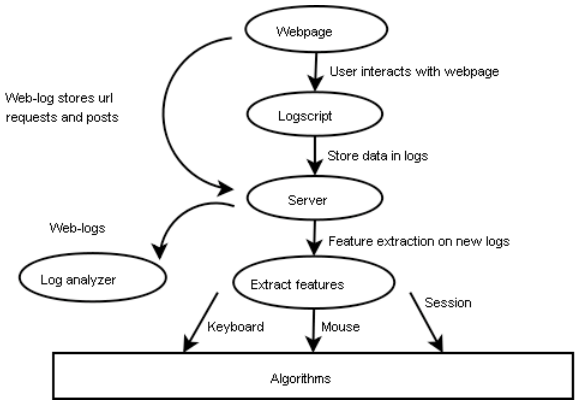


Figure 2.4: From the user interaction with the web page to extracted features are sent to algorithms.

The steps from user interaction with the test web page to the extracted features getting fed into the algorithms can be viewed in Figure 2.4. As previously stated, Click Paths from web log were not used in any algorithm in the system. The results from the algorithms were saved in a similar fashion as the log files, which can be seen below. Each user profile consisted of one ANN for the selected keyboard features, one SOM for the selected session features and a Meta-ANN which was used for voting. As nothing was trained or created for k-NN, all feature files for mouse and keyboard were stored in order to run k-NN when testing a new sample.

EMAIL.keyboard_ann
EMAIL.session_som
EMAIL.meta_ann

Those three files together with the stored log files for mouse and keyboard, should be used to predict if a new sample is accepted or rejected. A perfect system should for all user profiles accept the matching samples from the user and reject all samples coming from other users.

Figure B.1 shows a schematic view of how the system would work if deployed in a real life scenario. As the system should add a level of security by re-authenticating the users of a web page, each user needs a profile that should be created when the user creates a new account on the web page. When a user $A$ is logging in as $U$, session data will be stored and checked against the stored user $U$'s profile. If it's a match, the user can continue its session and the profile of $U$ is updated with the new session data. If it's a mismatch, the system should raise suspicion that $A$ is not $U$. The system should work in silence and user $A$ should be unaware of its presence.

The evaluation of the complete system worked similarly to how the deployed system should work. Instead of having a web page where users trying to log in as a specific user, the data from the validation set noted as (B) in Figure 2.2 was used to create login attempts on all users. Figure 2.5 shows how a sample from the validation set got tested. For all 19 user profiles, all samples in the validation set were tested against the user profile. With 20 samples from each user, a total of 380 login attempts were tested on each user profile. As 20 of those samples were the users own samples they should have been accepted as coming from the user and any rejection of the user's own samples would be counted towards FRR. The remaining 360 samples should have been rejected as not coming from the user and would be counted towards FAR if accepted by the user profile. With 380 login attempts for each user, a grand total of 7220 login attempts were evaluated.

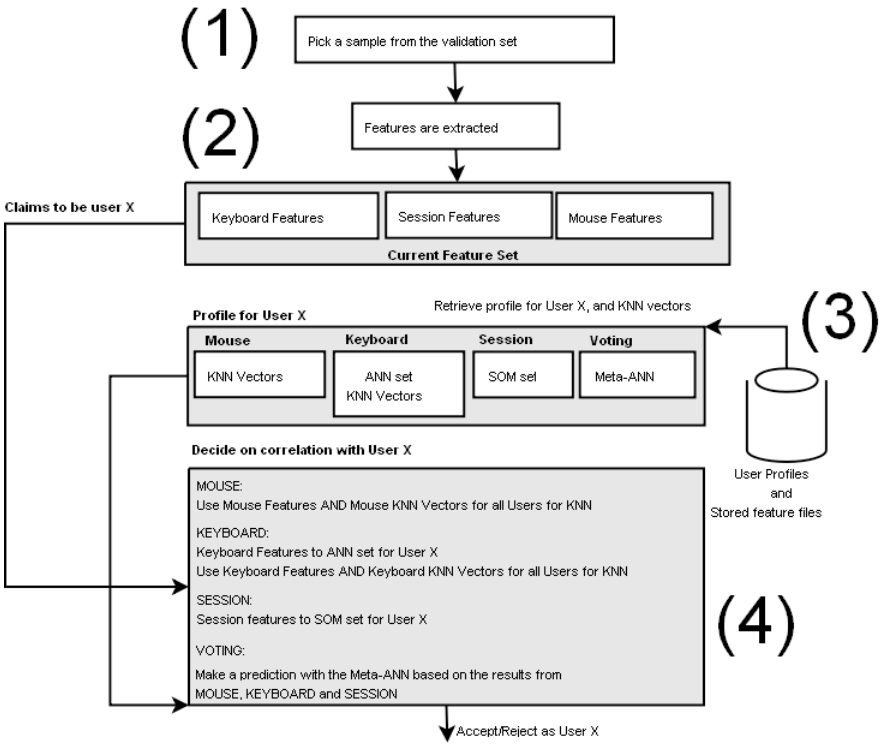


Figure 2.5: Validation system.

# 3 Results

A number of final systems were validated several times using different Keyboard-ANN configurations where the size of the negative training set was varied between 5.6% of the original negative set to 100% of the original negative set. The variation of this fraction parameter was introduced to slack the bias towards rejection. Introducing too many negative samples in comparison to positive samples could result in a system biased towards rejection since rejection would easily result in a low error during ANN training. The raw Keyboard-ANN results for each fraction setting can be seen in Table A.5.

The final results for the system can be seen in Table 3.1 where different voting techniques as well as different fractions where used for the Keyboard-ANN implementation. The results were based on the 20% of untouched initially acquired data which constituted the validation set.

| Voting mechanism | Keyboard-ANN Fraction (%) | FRR (%) | FAR (%) |
|---|---|---|---|
| Meta-ANN | 100 | 43.42 | 0.99 |
| Meta-ANN | 89.6 | 43.16 | 0.92 |
| Meta-ANN | 44.8 | 45.79 | 0.89 |
| Meta-ANN | 22.4 | 43.95 | 0.79 |
| Meta-ANN | 11.2 | 47.11 | 0.86 |
| Meta-ANN | 5.6 | 46.58 | 0.99 |
| Majority Voting | 100 | 55.53 | 0.95 |
| Majority Voting | 89.6 | 53.95 | 0.67 |
| Meta-ANN (SOM,Keyboard k-NN) | Not used | 65.53 | 1.05 |
| Meta-ANN (No SOM) | 100 | 48.95 | 1.11 |

Table 3.1: Based on 7220 samples in total, 6840 should be rejected by the correct profiles and 380 should be accepted by the correct profiles.

Majority-Voting without additional modifications was also tried on the 100% and 89.6% Keyboard-ANN sets without improvement contra Meta-ANN. This was to be expected and no further tests were made with the unmodified basic Majority-Voting technique. The results in Table 3.1 conclude that the Meta-ANN was better than plain Majority-Voting.

Judging by the plot in Figure 3.2, it is possible to be mislead and believe that only SOM and keyboard k-NN ought to perform better than the final system which uses ANNs trained with 89.6%. However this is not the case. The point marked as Meta-ANN (SOM and Keyboard k-NN) in the same plot (upper left corner) visualizes such a combined system with Meta-ANN and shows that the distance to origo is 0.66 which is a decrease in performance.

The FAR and FRR values for each individual profile can be seen in Figure 3.4 and Table A.8. Profiles 4 and 17 had the best individual results with FAR

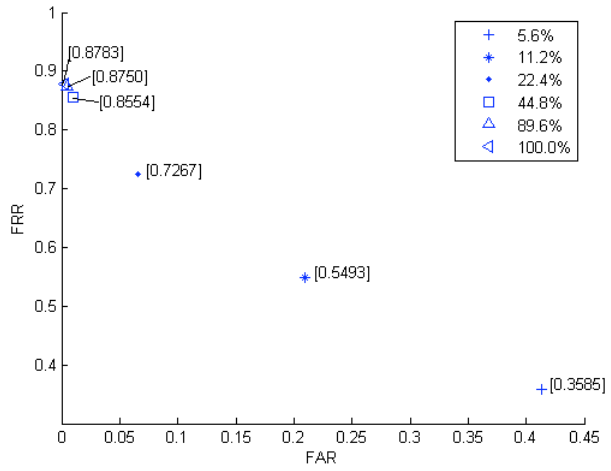


Figure 3.1: The Keyboard-ANN implementation trained with differing fractions of the negative set. The respective implementations are plotted based on their FAR and FRR values. Closer to origo is better, distance to origo can be seen within the brackets [ ].

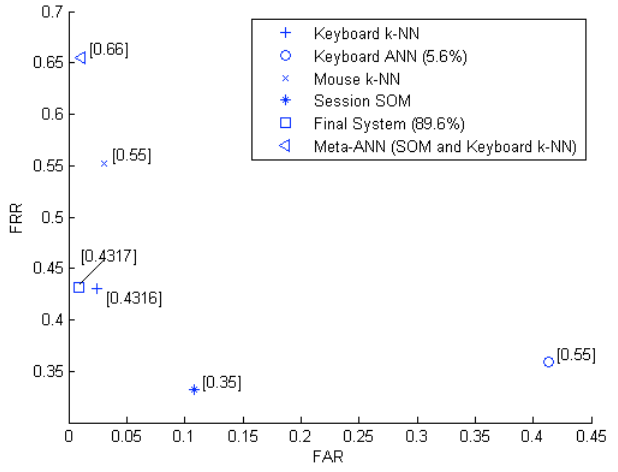


Figure 3.2: Image showing the best final system together with each classifier. The 5.6% keyboard ANN classifier is shown since it had the best accuracy amongst the keyboard ANN fractions. Note that the final system uses the 89.6% keyboard ANN fraction. Closer to origo is better, distance to origo can be seen within the brackets [ ].

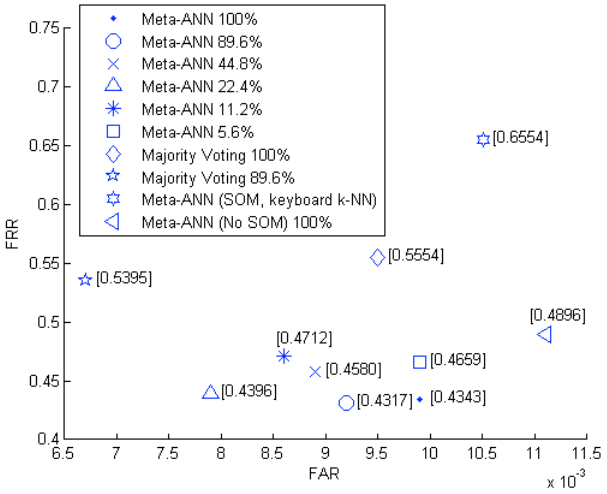


Figure 3.3: Image showing the accuracy of the various systems based on the differing ANN negative training set fractions. The Majority Vote classification is also plotted (based on 100% and 89.6%). Closer to origo is better, distance to origo can be seen within the brackets [ ].

between 0.8% to 1.9% and both with FRR of 0.0%. The worst individual profiles were profiles 6 and 18, which rejected everything.

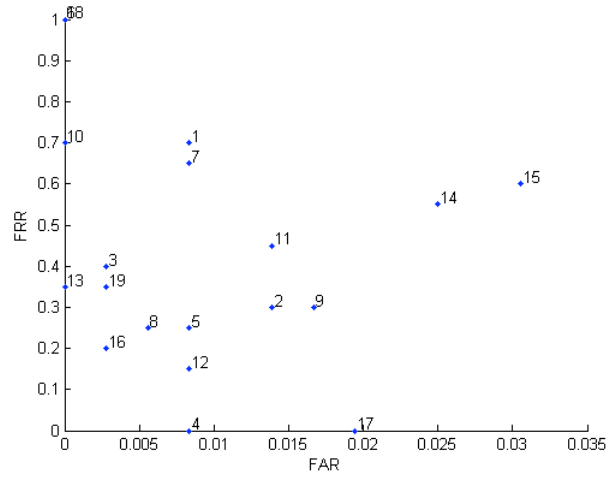


Figure 3.4: Image showing the accuracy of each profile for the best system (based on 89.6%). Closer to origo is better (note that the axis are not equal).

An alternative validation method which was explored was to subject the system, using the keyboard ANNs which were trained using 89.6% of the negative training set, to one sample and record which profiles were the ones to accept the sample as their own. Such a validation would be to guess which registered (if previously registered) user is currently browsing a website. This validation method proved to be very successful. An accuracy as high as 54.1% was observed. The statistics for this alternative validation method can be seen in Table A.10. For each sample, there were three cases considered.

1. *The correct profile solely accepted the sample.* This was the best case since all other profiles rejected the sample as belonging to them. In the alternative validation, there were 195 such correct classifications out of 380 samples.
2. *The correct profile accepted (together with others) the sample.* This was the next best case, since the correct profile also accepted the sample. A natural question is then; how many other profiles accepted the sample at the same time as the correct profile? The statistics shows that maximally (only) one other profile accepted the sample at the same time as the true profile. A total of 21 such cases were observed out of 380 samples.
3. *The correct profile did not accept the sample.* Such cases were ignored.

# 4 Conclusions

The best system was the one using Meta-ANN as a final voting technique where the the keyboard ANNs were trained with 89.6% of the negative training set as seen in Table 3.1. The high FRR (43.16%) can be attributed to the low grade of the collected data as a result from the several parts mentioned below.

1. **High number of negative samples in training** The proportion of negative samples was a factor 18 in comparison to the positive samples (64 positive and 1152 negative). This could impact the training of ANN to bias rejection and reduce the Root Mean Square Error (RMSE) drastically when training by favoring negative output classification. This hypothesis was tested as previously seen in Table 3.1 and the conlusion was that the negative sample size did not affect the final results noticeably.
2. **Data collection through a web page instead of a local application and local hardware.** Since the data collection was performed remotely through a web page using jQuery [1], the collected data was distorted in time (and space) by differing hardware configurations and browser capabilities. This caused the time resolution to differ on slower and faster machines. Differing hardware such as mouse and keyboards could also affect the data in space by noisy mouse movements. Had a fixed computer been used by all the subjects during the data collection, the hardware-specific distortions could maybe have been neglected since it would contribute equally for all the subjects.
3. **Corrupted data which had to be adjusted.** The collected data contained many entries which had to be corrected manually. Some entries had to be removed since correcting said data would involve direct data-forgery. This introduced the possibility that there might even be many errors which stayed undiscovered. Errors could be involuntary mouse-movements (gliding mouse pointer), involuntary double-clicks, etc. Other errors could be sporadic but still frequent enough to modify the already weak data sufficiently so as to make pattern recognition difficult. Still all the corrected data was used since data acquisition for this type of data was very time-expensive.
4. **No static enrollment samples from mouse.** Prior work reports utilized static enrollment samples from each user. Such enrollment samples were based on fixed mouse trajectories or mouse written signatures. The work in this report has not been subject to the same luxury since it was meant to be as real as possible. Based on the idea of a free E-commerce site, the mouse and keyboard interactions were to be as little restricted as possible (within reasonable bounds)[1]. It was also the case that the measures in the reports had been naturally selected to work well with the static enrollment sample scenario.

[1] Mouse data still had to be acquired somehow

5. **No static enrollment samples from keyboard.** As for the mouse data, prior work reports utilized static enrollment samples from each user. Such enrollment samples were based on text input in the form of username and password or copying a long fixed text. In this manner, N-gram statistics were successfully captured and could be used by proper measures to identify users. However, static enrollment samples were not used in this master thesis.

When analyzing the results from the best profiles mentioned in Chapter 3, some interesting occurrences were noted.

1. **For the best profiles, the SOM always accepted the samples from the correct profile.** But this alone was not enough for the profile to accept the sample. Whenever the profile accepted a sample, either the keyboard or mouse k-NN accepted that sample as well.
2. **For the best profiles, the keyboard ANN never accepted any samples as belonging to the profile.**

A number of questions arise naturally after performing such an analysis with the mentioned findings. These questions are posed together with their likely explanations below.

1. *Why was SOM always 1 for the best profiles when the sample indeed was from the profile?*
   One possible explanation could be that many of the subjects chose to perform their experiments in batches, thus performing several experiments one after another during a short period of time. Indeed, this was the case. Interviews and log reviews suggest that many of the subjects actually displayed this kind of behavior. Thus, many of the time stamps recorded in the session files were indeed close in time. Another contributing factor for the SOM's high accuracy could be that many of the User-Agent strings were indeed unique; thus resulting in a set of unique[2] hash values. However, even though the User-Agent strings were unique, some of them had a short Levenshtein Distance to others as can be seen in Table A.9. The User-Agent strings often differed greatly (different OS or Browsers) or little (different browser versions but same OS, etc).
   A first plan was to extract OS information, Browser Information and Version; however it was found that there is no real standard for the User-Agent string. It is also possible to forge User-Agent strings and send whatever data one would like to. Locking the implementation to a certain type of User-Agent formatting would be unwise given that the known set of User-Agent strings is tremendous[3].
   Analysis of the various access times for the experiments also reveal that many of the access times occur around the same time (between 17:10 to

[2] The absolute value reduced the uniqueness mapping
[3] At 2009-05-05, [46] reported 4094 different User-Agent strings

23:50), as can be seen in Figure B.2. There was a logical explanation to this; many of our friends work or study, thus there was often only spare time to perform such forced experiments in the evening. Such a clustered access time pattern reveals that there is no simple way to perform a polychotomy evaluation. Which could have contributed to the bad results of the SOM.

2. *Why did the Meta-ANN not accept the SOM vote as a veto?*
   For all profiles, the Meta-ANN would not accept the SOM's answer as definite when the SOM accepted the sample. During training of the Meta-ANN, the SOM made too many False Accept mistakes which forced the Meta-ANN to inhibit of the SOM input neuron. This can also be seen in Table A.7 where SOM achieved a FAR value of 10.8% during evaluation of the algorithm. Since the keyboard-ANN almost always rejected a sample, any one of the keyboard- or mouse-k-NN would have to accept the sample in order for the profile to accept it.
3. *Why did the keyboard-ANN perform so badly?*
   With a high FRR (35.9−87.8% during evaluation, Table A.5), the keyboard-ANN rejected almost every sample during validation. Even though it lacked support in prior works, the distribution of special keys were chosen as the features for keyboard-ANN due to the fact that it outperformed any other feature combination mentioned in prior works, such as N-gram times. The distribution of special keys remained unchanged even if the timing values were incorrect. This was not true for other features that rely heavily on the correctness of the timing values, such as typing speed and N-gram times. If a static text would have been used in the data collection as well as getting the correct timing information, other feature combinations might have been possible and the keyboard-ANN performance could have been improved.
   According to [43], ANN suits problems where data can be noisy and complex. The Backpropagation algorithm for training the ANN should also be a good choice for our setting. The data should be represented by attribute-value pairs such as our features on keyboard. The training data can contain noise and errors, which should cover the problem of timing errors in the data collection. It also handles all kind of input values, either discrete- or real-valued. In addition, a multilayer network like the one used, should also be able to separate the data set with non-linear expressions. Seeing as both ANN and the Backpropagation algorithm should work well for our setting, the most likely explanation to the poor performance of the keyboard-ANN would be that the features from the profiles were too similar, the samples were too scattered or the samples were incosistent, such that they could not be separated.

Given that the alternative validation method proved to be so accurate (54.1%), it seems feasible to construct a system which recognizes registered users as they visit a web-page without logging in. It would then be quite simple to transfer collected profile data from one website where a user $A$ has registered a profile

and apply that profile to other websites. The profile of user $A$ could then be used to find user $A$ on other sites. This could lead to a method where one could map people's internet-habits by tracking them across web-sites.

If one would have access to several profiles from different websites, one could also compare the profiles in the hope of finding the same user on several websites. Such practice could increase the accuracy by which targeted advertisement is used and could also help in finding multiple accounts from one user.

Yet one other possible field of study could be to analyze the keyboard and behavioral patterns of certain demographic profiles, thus one could maybe say with a certain certainty whether the user visiting a web-page is male/female, child/adult and so forth.

## 4.1 Future work

Future work would emphasize improvements to performance, mainly to reduce the high FRR. Improvements to algorithms could also be made, such as extending k-NN to weighted k-NN. The user supplied data for authentication could be used to improve the stored profile with more and newer data. The systems should then be analyzed to check whether or not the performance improves over time as more data is gathered from each user. Implementing the system into a live E-commerce web-site would be needed to be done as well, in order to evaluate how well the system would work in a real-life scenario.

# Bibliography


[1] Resig John and the jQuery Team. *jQuery JavaScript Library*. `www.jquery.com`.

[2] M. Faundez-Zanuy. Biometric security technology. *IEEE Aerospace and Electronic Systems Magazine*, 21(6):15–26, 2006.

[3] Swati Bharati, Rumeiz Haseem, Raheel Khan, Mark Ritzmann, and Alex Wong. Biometric authentication system using the dichotomy model. 2008.

[4] J. Hu, D. Gingrich, and A. Sentosa. A k-nearest neighbor approach for user authentication through biometric keystroke dynamics. pages 1556–1560, 2008.

[5] Nkem Ajufor, Antony Amalraj, Rafael Diaz, Mohammed Islam, and Michael Lampe. Refinement of a mouse movement biometric system. *Proceedings of Student-Faculty Research Day, CSIS, Pace University, May 2nd, 2008*, 2008.

[6] M. Rybnik, M. Tabedzki, and K. Saeed. A keystroke dynamics based system for user identification. pages 225–230, 2008.

[7] M. Villani, C. Tappert, G. Ngo, J. Simone, H. St. Fort, and Sung-Hyuk Cha. Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions. volume 2006, 2006.

[8] A. F. Syukri, E. Okamoto, and M. Mambo. A user identification system using signature written with mouse. *the Proceedings of the Third Australasian Conference on Information Security and Privacy (ACISP)*, pages 403–414, 1998.

[9] Shivani Hashia. Authentication by mouse movements. 2004.

[10] Adam Weiss, Anil Ramapanicker, Pranav Shah, Shinese Noble, and Larry Immohr. Mouse movements biometric identification: A feasibility study. *Proceedings of Student/Faculty Research Day, CSIS, Pace University, May 4th, 2007*, 2007.

[11] A. A. E. Ahmed and I. Traore. A new biometrie technology based on mouse dynamics. *IEEE Transactions on Dependable and Secure Computing*, 4(3):165–179, 2007.

[12] D. A. Schulz. Mouse curve biometrics. 2006.

[13] M. Pusara and C. E. Brodley. User re-authentication via mouse movements. pages 1–8, 2004.

[14] R. Atterer, M. Wnuk, and A. Schmidt. Knowing the user's every move: User activity tracking for website usability evaluation and implicit interaction. pages 203–212, 2006.

[15] R. A. J. Everitt and P. W. McOwan. Java-based internet biometric authentication system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1166–1172, 2003.

[16] D. V. Khmelev and F. J. Tweedie. Using markov chains for identification of writers. *Literary and Linguistic Computing*, 16(4):229–307, 2002.

[17] Z. Jin, A. B. J. Teoh, S. O. Thian, and C. Tee. Typing dynamics biometric authentication through fuzzy logic. volume 4, 2008.

[18] G. L. F. B. G. Azevedo, G. D. C. Cavalcanti, and E. C. B. Carvalho Filho. An approach to feature selection for keystroke dynamics systems based on pso and feature weighting. pages 3577–3584, 2008.

[19] P. Kang, Seong seob Hwang, and S. Cho. *Continual retraining of keystroke dynamics based authenticator*, volume 4642 LNCS. 2007.

[20] K. N. Shiv Subramaniam, S. Raj Bharath, and S. Ravinder. Improved authentication mechanism using keystroke analysis. pages 258–261, 2007.

[21] Tai-Hoon Cho. Pattern classification methods for keystroke analysis. pages 3812–3815, 2006.

[22] Y. Sheng, V. V. Phoha, and S. M. Rovnyak. A parallel decision tree-based method for user authentication based on keystroke patterns. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(4):826–833, 2005.

[23] M. Koppel and J. Schier. Exploiting stylistic idiosyncrasies for authorship attribution. *Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis*, pages 69–72, 2003.

[24] R. Wu. Clustering web access patterns based on hybrid approach. volume 1, pages 52–56, 2008.

[25] H. Ling, Y. Liu, and S. Yang. An ant colony model for dynamic mining of users interest navigation patterns. pages 281–283, 2008.

[26] V. S. Tseng, K. W. Lin, and Jeng-Chuan Chang. Prediction of user navigation patterns by mining the temporal web usage evolution. *Soft Computing*, 12(2):157–163, 2008.

[27] N. Labroche. Learning web users profiles with relational clustering algorithms. volume WS-07-08, pages 54–64, 2007.

[28] Shin-Yi Wu and Yen-Liang Chen. Mining nonambiguous temporal patterns for interval-based events. *IEEE Transactions on Knowledge and Data Engineering*, 19(6):742–758, 2007.

[29] E. Tug, M. Sakiroglu, and A. Arslan. Automatic discovery of the sequential accesses from web log data files via a genetic algorithm. *Knowledge-Based Systems*, 19(3):180–186, 2006.

[30] B. Zhou, S. C. Hui, and A. C. M. Fong. Discovering and visualizing temporal-based web access behavior. volume 2005, pages 297–300, 2005.

[31] J. J. Chen, J. Gao, J. Hu, and B. S. Liao. A hierarchical markovian mining approach for favorite navigation patterns. volume 3381, pages 92–95, 2005.

[32] A. Scherbina and S. Kuznetsov. Clustering of web sessions using levenshtein metric. volume 3275, pages 127–133, 2004.

[33] D. Xing and J. Shen. Efficient data mining for web navigation patterns. *Information and Software Technology*, 46(1):55–63, 2004.

[34] N. Labroche, N. Monmarché, and G. Venturini. *AntClust: Ant clustering and web usage mining*, volume 2723. 2003.

[35] E. Menasalvas, S. Millán, J. M. Peña, M. Hadjimichael, and O. Marbán. Subsessions: A granular approach to click path analysis. *International Journal of Intelligent Systems*, 19(7):619–637, 2004.

[36] Chris Pederick. *User Agent Switcher*. `http://chrispederick.com/work/user-agent-switcher/`, 2009-05-07.

[37] The Tor Project Inc. *Tor: anonymity online*. `http://www.torproject.org/`, 2009-05-07.

[38] *The Proxy Authority*. http://proxy.org/, 2009-05-07.

[39] G. A. Wilkin and H. Xiuzhen. K-means clustering algorithms: Implementation and comparison. pages 133–136, 2007.

[40] J. Shen, Y. Lin, and Z. Chen. Incremental web usage mining based on active ant colony clustering. *Wuhan University Journal of Natural Sciences*, 11(5):1081–1085, 2006.

[41] Sir Doyle, Arthur Conan. *The Adventures of Sherlock Holmes*. Project Gutenberg, 1999.

[42] Chalmers University of Technology. *Mono-, Bi and Trigram Frequency for English*. `http://www.cs.chalmers.se/Cs/Grundutb/Kurser/krypto/en\_stat.html`.

[43] M. Mitchell Tom. *Machine Learning*. McGraw-Hill, 1997.

[44] Ross Quinlan. *Is C5.0 Better Than C4.5?* `http://www.rulequest.com/see5-comparison.html`, 2009-05-19.

[45] H. Oh and K. Chae. Real-time intrusion detection system based on self-organized maps and feature correlations. volume 2, pages 1154–1158, 2008.

[46] Andreas Staeding. *List of User-Agents (Spiders, Robots, Crawler, Browser)*. `http://www.user-agents.org/`.

[47] T. Kohonen. *Self-Organizing Maps , Third Edition*. Springer, 2001.

[48] Tuevo Kohonen and Timo Honkela. *Kohonen network*. `http://www.scholarpedia.org`, 2009-05-08.

[49] M. Wahde. *Biologically Inspired Optimization Methods An Introduction*. WIT Press, 2008.

# A Tables

| Code | Format | Description |
|---|---|---|
| 0 | A | Key down |
| 1 | A | Key up |
| 2 | B | Mouse-down left |
| 3 | B | Mouse-up left |
| 4 | B | Mouse-down middle |
| 5 | B | Mouse-up middle |
| 6 | B | Mouse-down right |
| 7 | B | Mouse-up right |
| 8 | B | Mouse move |
| 9 | C | Mouse over |
| 10 | B | Double-click left |
| 11 | B | Double-click middle |
| 12 | B | Double-click right |
| 13 | D | Mouse-wheel |

| Format | Includes |
|---|---|
| A | Character code |
| B | X,Y position |
| C | HTML object that mouse moved over |
| D | X,Y position, direction of scroll |

Table A.1: Description of actions stored in the log files.

| Algorithm | Keyboard | Mouse | Session | Click Path |
|---|---|---|---|---|
| Artificial Neural Networks | X | X | | |
| K-Nearest Neighbor | X | X | | |
| ID3[a] | X | X | X | |
| Antclust | | | | X |
| Particle Swarm | | | X | |
| GA | | | | X |
| Subsession analysis | | | | X |
| Ant colony model | | | | X |
| Efficient mining | | | | X |
| K-Means | | | | X |
| Self Organizing Maps[b] | | | X | |
| Web Navigation Markov Model [c] | | | | X |

Table A.2: The implemented algorithms and for which kind of data they where tested. A cross (X) in the field of Keyboard, Mouse, Session or Click Path indicates that the algorithm was tested for that type of data.

[a] Report used C5.0
[b] Added later
[c] Added later

| Algorithm | Keyboard | Mouse | Session | Click Path |
|---|---|---|---|---|
| Artificial Neural Networks | X | X | | |
| K-Nearest Neighbor | X | X | | |
| ID3[a] | X | X | | |
| Self Organizing Maps[b] | | | X | |
| Web Navigation Markov Model [c] | | | | X |

Table A.3: The selected algorithms, based on prior work and own tested data. A cross (X) in the field of Keyboard, Mouse, Session or Click Path indicates that the algorithm would be used for that type of data.

[a]Report used C5.0
[b]Added later
[c]Added later

| K-value | FRR | FAR |
|---|---|---|
| 1 | 0.4572 | 0.0254 |
| 2 | 0.5033 | 0.0280 |
| 3 | 0.4309 | 0.0239 |
| 4 | 0.4507 | 0.0250 |
| 5 | 0.4605 | 0.0256 |
| 6 | 0.4671 | 0.0260 |
| 7 | 0.4901 | 0.0272 |
| 8 | 0.5033 | 0.0280 |
| 9 | 0.4934 | 0.0274 |
| 10 | 0.5000 | 0.0278 |
| 11 | 0.5132 | 0.0285 |

Table A.4: k-NN on keyboard data based on 304 samples using features numbered $0, 1, 35, 36$ as seen in Table 2.2

| Fraction of negative set | FRR | FAR | $\sqrt{FAR^2 + FRR^2}$ | Comment |
|---|---|---|---|---|
| 0.056 | 0.3586 | 0.4128 | 0.3585 | Equal amount positive and negative |
| 0.112 | 0.5493 | 0.2098 | 0.5493 | Double amount negative contra positive |
| 0.224 | 0.7237 | 0.0656 | 0.7267 | |
| 0.448 | 0.8553 | 0.0095 | 0.8554 | |
| 0.896 | 0.875 | 0.0044 | 0.8750 | |
| 1.0 | 0.8783 | 0.0026 | 0.8783 | |

Table A.5: ANN on keyboard data using features numbered $70 - 82$ as seen in Table 2.2. Each ANN consisted of 13 input, 13 hidden and 1 output neuron trained with $2 * 10^6$ iterations with a stop-error at 0.01. Each row corresponds to the results of the implementation based on differing sizes of negative values (Fraction).

| K-value | FRR | FAR |
|---|---|---|
| 1 | 0.5954 | 0.0331 |
| 2 | 0.6118 | 0.0340 |
| 3 | 0.5789 | 0.0322 |
| 4 | 0.5954 | 0.0331 |
| 5 | 0.5789 | 0.0322 |
| 6 | 0.5789 | 0.0322 |
| 7 | 0.5559 | 0.0309 |
| 8 | 0.5691 | 0.0316 |
| 9 | 0.5526 | 0.0307 |
| 10 | 0.5592 | 0.0311 |
| 11 | 0.5526 | 0.0307 |

Table A.6: k-NN on mouse data using features numbered 7,8,10,11,12,14, 15,21,22 as seen in Table 2.1

| FRR | FAR |
|---|---|
| 0.3322 | 0.1080 |

Table A.7: SOM on session data using 304 samples with features numbered $4, 5, 6, 7, 9$ as seen in Table 2.4. Each SOM consisted of $25^2$ neurons trained for 1000 iterations each.

| Profile | FRR | FAR |
|---|---|---|
| 1 | 0.7000 | 0.0083 |
| 2 | 0.3000 | 0.0139 |
| 3 | 0.4000 | 0.0028 |
| 4 | 0 | 0.0083 |
| 5 | 0.2500 | 0.0083 |
| 6 | 1.0000 | 0 |
| 7 | 0.6500 | 0.0083 |
| 8 | 0.2500 | 0.0056 |
| 9 | 0.3000 | 0.0167 |
| 10 | 0.7000 | 0 |
| 11 | 0.4500 | 0.0139 |
| 12 | 0.1500 | 0.0083 |
| 13 | 0.3500 | 0 |
| 14 | 0.5500 | 0.0250 |
| 15 | 0.6000 | 0.0306 |
| 16 | 0.2000 | 0.0028 |
| 17 | 0 | 0.0194 |
| 18 | 1.0000 | 0 |
| 19 | 0.3500 | 0.0028 |

Table A.8: Individual profile results with FRR, FAR and fitness values.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 88 | | | | | | | | | | | | | | | |
| 3 | 79 | 129 | | | | | | | | | | | | | | |
| 4 | 33 | 87 | 47 | | | | | | | | | | | | | |
| 5 | 33 | 87 | 49 | 2 | | | | | | | | | | | | |
| 6 | 84 | 110 | 99 | 81 | 81 | | | | | | | | | | | |
| 7 | 30 | 102 | 71 | 42 | 43 | 91 | | | | | | | | | | |
| 8 | 54 | 104 | 26 | 21 | 23 | 87 | 55 | | | | | | | | | |
| 9 | 34 | 87 | 53 | 6 | 4 | 79 | 47 | 27 | | | | | | | | |
| 10 | 29 | 86 | 53 | 6 | 4 | 83 | 47 | 27 | 8 | | | | | | | |
| 11 | 31 | 86 | 53 | 6 | 4 | 83 | 47 | 27 | 8 | 2 | | | | | | |
| 12 | 88 | 116 | 100 | 85 | 85 | 14 | 95 | 88 | 84 | 87 | 87 | | | | | |
| 13 | 25 | 82 | 76 | 35 | 36 | 86 | 40 | 52 | 36 | 32 | 34 | 90 | | | | |
| 14 | 88 | 1 | 130 | 87 | 87 | 110 | 102 | 104 | 87 | 86 | 86 | 117 | 82 | | | |
| 15 | 26 | 101 | 75 | 46 | 47 | 91 | 4 | 59 | 51 | 43 | 45 | 95 | 36 | 101 | | |
| 16 | 40 | 89 | 77 | 30 | 30 | 60 | 60 | 51 | 27 | 26 | 28 | 65 | 41 | 89 | 56 | |
| 17 | 84 | 110 | 98 | 83 | 83 | 4 | 91 | 88 | 81 | 81 | 81 | 18 | 84 | 110 | 91 | 58 |
| 18 | 18 | 80 | 74 | 27 | 28 | 87 | 33 | 48 | 30 | 24 | 26 | 91 | 14 | 80 | 29 | 35 | 85 |
| ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |

Table A.9: Levenshtein distance between unique User-Agent strings recorded in session files. The ID is an assigned number only to keep track of which unique User-Agent string is being tested.

| Case | Observed |
|---|---|
| Correct classifications where correct profile is single guess | 195.0 |
| Correct classifications where correct profile is in guess set | 21.0 |
| Average amount of other classifiers in guess set | 1.0 |
| Maximal amount of other classifiers in guess sets | 1 |
| Minimal amount of other classifiers in guess sets | 1 |

Table A.10: The recorded results of the alternative validation method based on the same untouched validation data as for the final validation system.

# B Figures

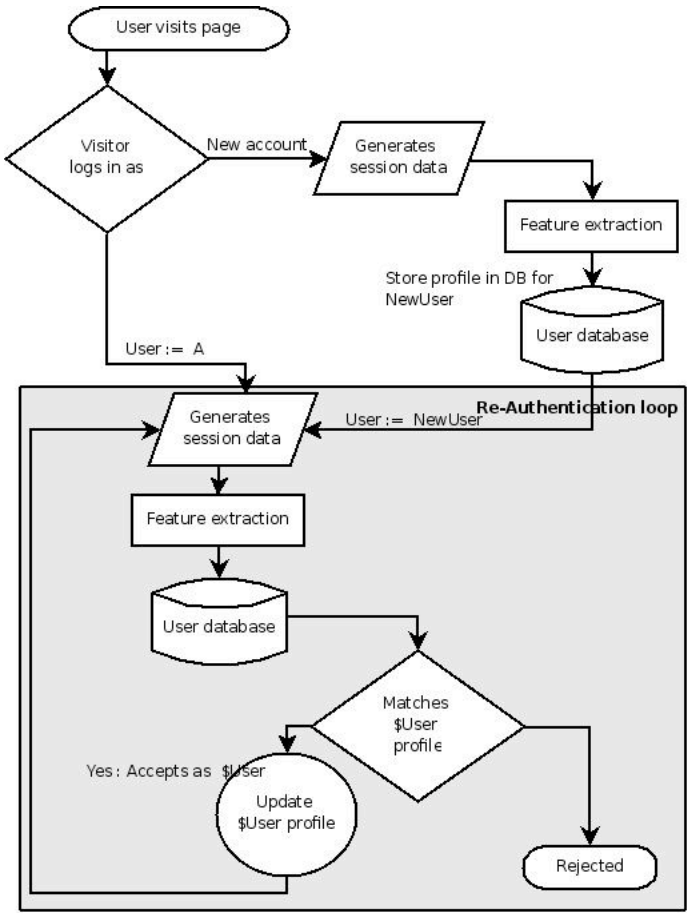


Figure B.1: How the system is supposed to work once deployed into a real life scenario.

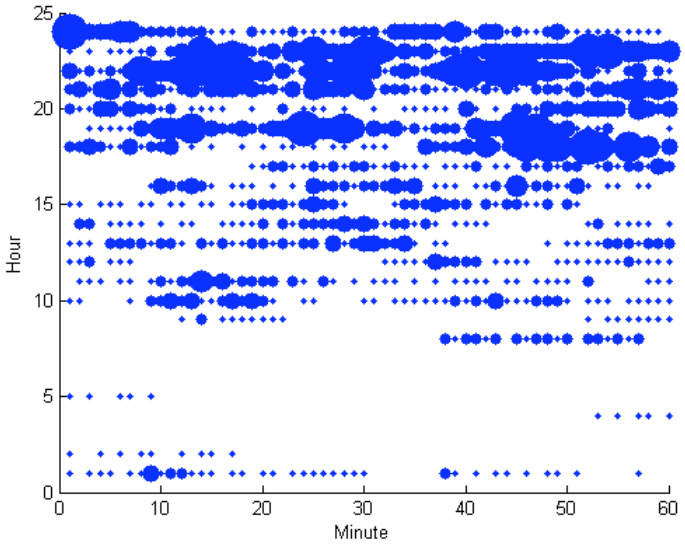


Figure B.2: The amount of access times plotted against the hour of the day and the minute.

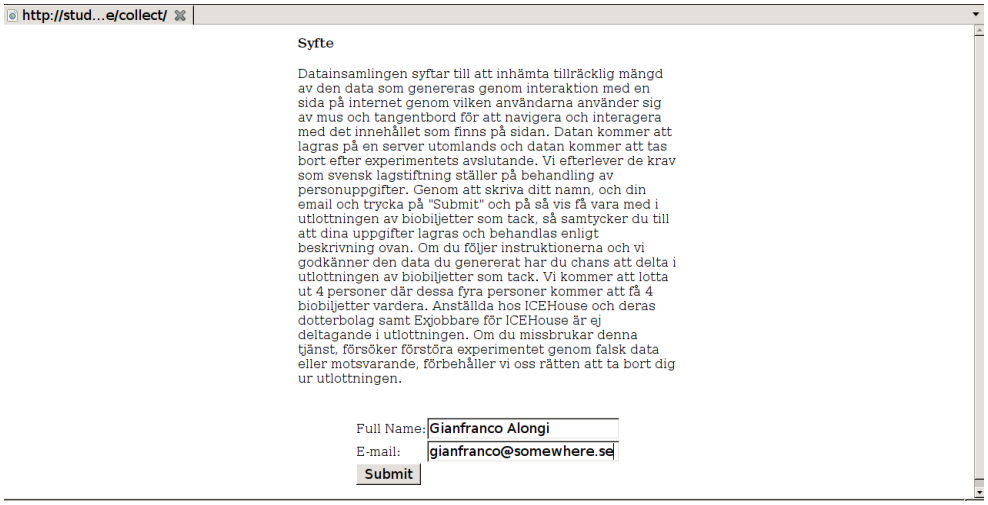


Figure B.3: A screenshot of the login page with the disclaimer text. Users entered their name, email and proceeded to the collection steps.

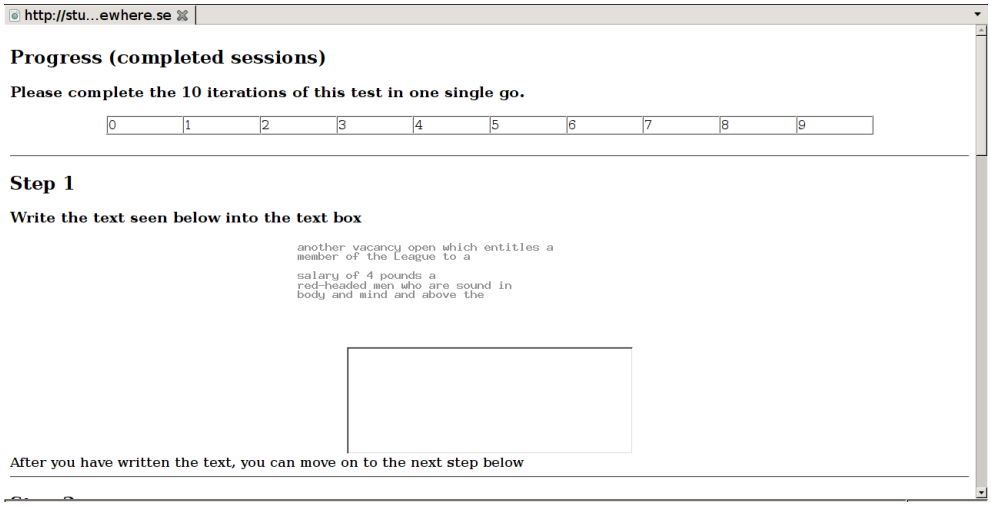


Figure B.4: A screenshot of step 1 in the collection experiment. This step involved re-writing the text as seen in the image "*another vacancy open which entitles a member of the League to a salary of 4 pounds a red-headed men who are sound in body and mind and above the*".

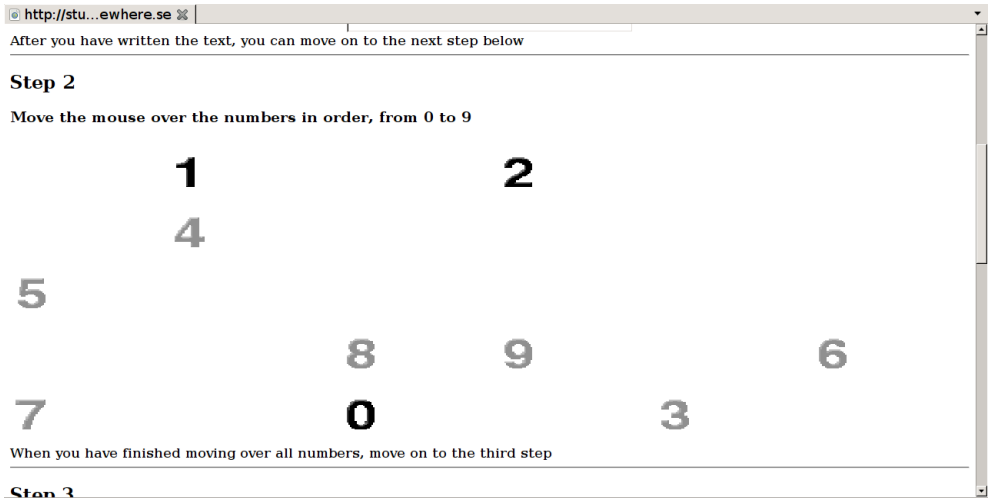


Figure B.5: A screenshot of step 2 in the collection experiment. The user was forced to move the mouse over the numbers in order, to proceed to the next step. The numbers seen as black in the image have been moved over by the mouse and the grey numbers have not been moved over yet.

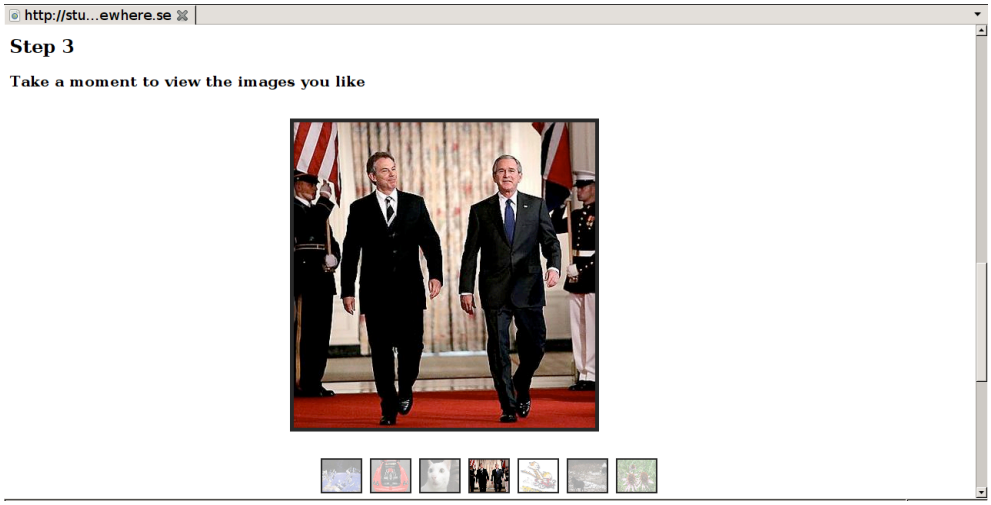


Figure B.6: Step 3 showed a sequence of images from some category which the user had previously selected. The current image seen by the subject was the enlarged one (former president of the United States of America George W Bush with former British prime minister Tony Blair). A jQuery plugin slid between the images seen in the bottom.

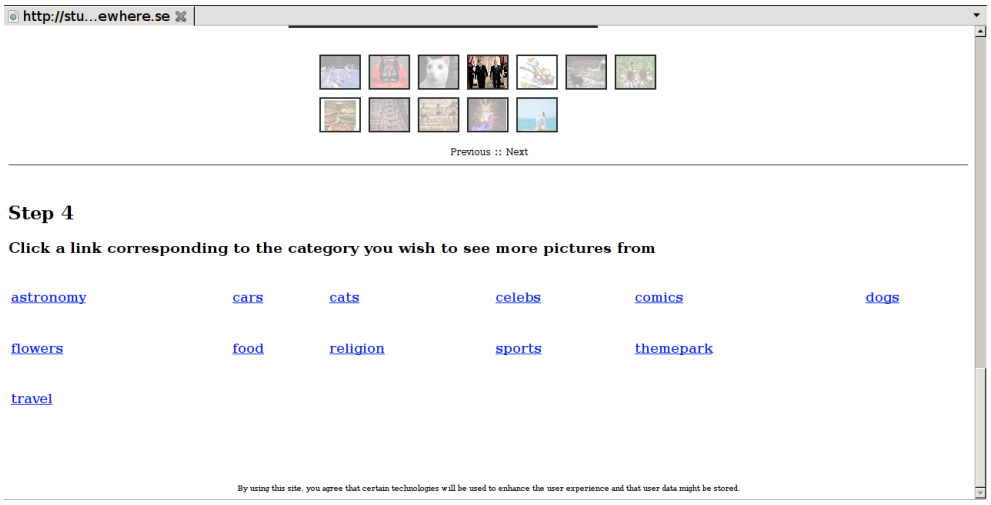


Figure B.7: A screenshot of step 4 of the data collection. The user could choose any category he/she preferred. Images from this category would be shown in step 3 in the next iteration of the data collection.

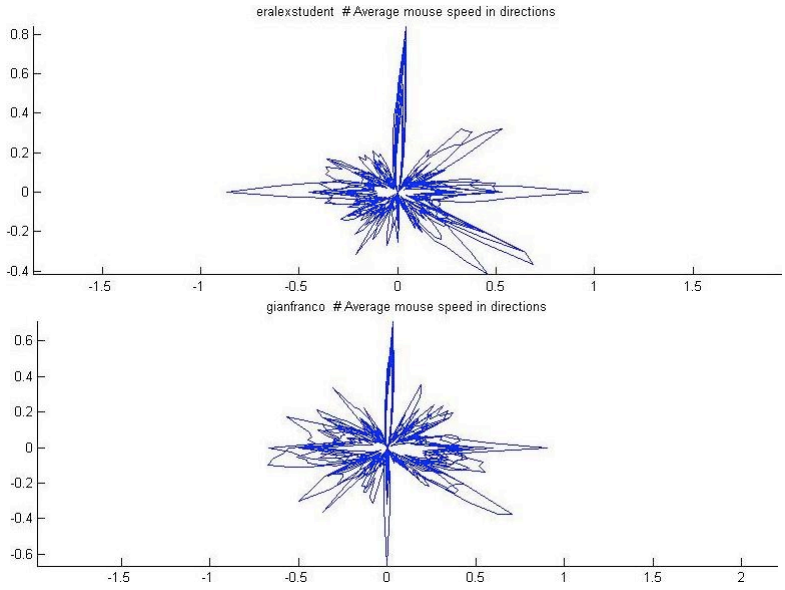


Figure B.8: 2 visual examples of Mouse Vector data features from the authors with average movement speeds in 64 directions. The images shows 10 Mouse Vector features each, extracted from 10 iterations.

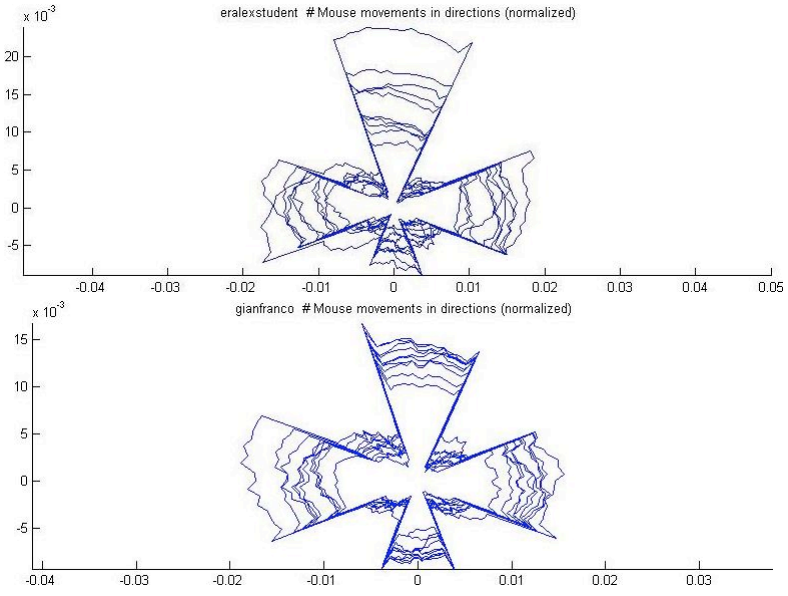


Figure B.9: 2 visual examples of Mouse Vector data features from the authors with total number of mouse moves in 64 directions. The images shows 10 Mouse Vector features each, extracted from 10 iterations.

# C Algorithms

## C.1 URL mapping

**Input**: Transitions between categories
**Output**: Mapping of viewed pages
First mapped page is automatically "/index.html"
Initially DEPTH = 0
**foreach** *Transition from category* $C_i$ *to* $C_{i+1}$ **do**
  **if** $C_{i+1} == C_i$ **then**
    DEPTH = DEPTH + 1
    mapped page becomes "/$C_{i+1}$/DEPTH.html"
  **else**
    DEPTH = 0
    mapped page becomes "/$C_{i+1}$/index.html"
  **end**
**end**

**Algorithm 1**: Transition to URL mapping

## C.2 k-NN

The k-NN algorithm is an example of instance-based supervised learning algorithm which forms no hypothesis upon execution. Classification is performed for a certain $m$-dimensional vector based on the $k$ nearest $m$-dimensional samples in the training set.

The inductive bias of the k-NN algorithm is that each sample will be most similar to other samples of it's own class. This inductive bias seems reasonable for the user classification setup by assumption that a user's generated data lies within a fixed personal distribution.

However, the inductive bias makes the k-NN algorithm suffer from "the curse of dimensionality". There is no way for the algorithm to know which attributes are irrelevant and each attribute carry equal weight. This could be slightly avoided by attribute normalization, or adding weights to each attribute.

**Input**: Training set $X$, Sample $x_c$ to be classified
**Output**: Classification of sample $x_c$
**foreach** *Training sample* $(x, f(x)) \in X$*, where x is the feature vector and f(x) is the class of x.* **do**
  Calculate standard Euclidean distance to $x_c$ from all vectors in training set, and decide most common class amongst the K nearest instances.
**end**

**Algorithm 2**: Short description of k-NN algorithm. A more extensive description can be found in [43].

## C.3 Self Organizing Map

The Self Organizing Map (SOM) is a competitive unsupervised learning algorithm which converts nonlinear statistical relationships between high-dimensional data into simple geometric relationships. A self-organized similarity graph of the input data is created by topologically clustering similar objects. The SOM algorithm is partially based on the LVQ [47] algorithm and models the synaptic plasticity of the brain.

**Input**: Set $\mathcal{V}$ of n-dimensional vectors $R^n$ with labels $\mathcal{L}$
**Output**: Kohonen Neural Network trained on the input set
Initialize the lattice of $n$-dimensional vectors $R^n$ (model) bound to each neuron.
$t \leftarrow 0$
**for** *K iterations* **do**
  $t \leftarrow t + 1$
  **foreach** *vector v in the set* $\mathcal{V}$ **do**
    $m_c \leftarrow \underset{i}{argmin}\{||v - m_j||\} : m_j$ are models of neurons
    Update $m_c$ and it's neighbors $m_i \in Neigborhood(m_c)$
    $m_c \leftarrow m_c + \alpha(t)[v - m_c]$
    **foreach** $m_i \in Neighborhood(m_c)$ **do**
      $m_i \leftarrow m_i + \alpha(t)h_{ci}(t)[v - m_i]$
    **end**
  **end**
**end**
**foreach** *vector v in the set* $\mathcal{V}$ **do**
  Find the most similar neuron and label it with the label of $v$
**end**

**Algorithm 3**: A brief description of the SOM algorithm as presented in [47, 48]. Where $\alpha(t)$ is a scalar valued "*learning factor*" which should decrease over time. The Gaussian function $h_{ci}(t) = e^{-\frac{(||m_c - m_j||)^2}{2\sigma(t)^2}}$ is the smoothing kernel. $\sigma(t)$ is the width of the kernel.

## C.4 ANN

ANN is another instance-based learning method that origins from biological learning systems, especially those with neurons connected in a network. An Artificial Neural Network mimics biological neural networks with neurons (or nodes) that are connected to other nodes by links which serves to propagate a signal between the nodes. When calculating the output from a node, the weighted sum of its input links are sent through an activation function. Link weights are often set by the use of the supervised learning algorithm Backpropagation explained in Algorithm 5.

The inductive bias of Backpropagation algorithm lies in the continous hypothesis space that is created by the $n$-dimensional hypothesis space from the $n$ link weights. The continous hypothesis space leads to a well-defined error gradient which can be efficiently searched using gradient descent search for the best hypothesis. The inductive bias can be roughly characterized as a smooth interpolation between data points.

**Input**: An Artificial Neural Network and an input vector.
**Output**: The output from the network.

1. The first layer in the network is the input layer which simply transmits the values from the input vector to the hidden layers (there can be several hidden layers).
2. A simple operation is performed in each neuron in the hidden layers and the signal propagates until it reaches the output layer which gives the output of the network.
3. Each link between neurons in the hidden layers has a weight which should be set by using a training algorithm, such as Backpropagation.

**Algorithm 4**: A brief description on how an ANN computes the output for a given input. [49] gives a more thorough description.

**Input**: A training set $\mathcal{T}$ with input vectors and desired output.
**Output**: An ANN with weights trained according to training set.

1. Initialize the network with chosen layers and set neuron weights to a small random vale.
2. In each epoch, generate a permutation $\mathcal{T}_p$ of the training set.
   **foreach** $t \in \mathcal{T}_p$ **do**
   Calculate output for $t$ and the error $e$ between the output and the desired output. Update the weights of the neurons according to equations found in [49].
   **end**
3. After a set amount of epochs, compute the RMSE over the whole training set.
4. Repeat 2 and 3 until the error has dropped below a desired level.

**Algorithm 5**: Backpropagation algorithm for setting neuron weights in ANN.

# D Abbreviations

| Abbreviation | Meaning |
|---|---|
| ANN | Artificial Neural Network |
| k-NN | K-Nearest Neighbor |
| SOM | Self Organizing Map (Kohonen Neural Network) |
| FRR | False Reject Rate |
| FAR | False Accept Rate |
| RMSE | Root Mean Square Error |

Table D.1: Common abbreviations and their respective meaning.