Advances in Keystroke Dynamics Techniques to Group User Sessions

Sebastián Sznur*‡, Sebastián García*

* Information Security, Faculty of Engineering, FASTA University, Gascon 3145 (B7600FNK) Mar del Plata, Argentina. Tel. / Fax (54223) 499-0400

‡ Corresponding Author; Tel: +54 223 9 426 7166, e-mail: szsebas@gmail.com

Abstract- Users identification by means of their keystroke pattern is an old and known technique. Several works had analysed and solved some of the most important issues in this area, but with the advances of technology, previous techniques have quickly become obsolete. Users as well as attackers spend a lot of time typing on their computers, locally and remotely. Their keystrokes leave a trace of patterns whose dynamism can be analysed and used to verify their identity. We propose to use unsupervised clustering algorithms to group user sessions together in order to correctly identify them. Furthermore, the verification of keystroke dynamics techniques has always been difficult because of the lack of a labelled free text dataset. To overcome this issue, we capture a large dataset of labelled keystrokes of more than two and a half million digraphs. Results show that having a keystroke dynamics dataset of user sessions and knowing the number of users who participated, it is possible to group the sessions of the same user, regardless of any previous knowledge about the users. A level of accuracy of at least 78% was achieved. If the sessions with fewer digraphs are discarded, a performance that exceeds 90% can be attained.

Keywords- Security; Keystroke Dynamics; Biometrics.

1. Introduction

Since the early days of telegraph communications in World War II, when Morse code was used to send and receive messages, each telegraph operator developed its own typing signature. This signature could be learnt from closely listening to the typing rhythm of the operators in order to identify them. It was even possible to precisely differentiate among operators. These techniques were the first rudimentary keystroke dynamics implementations, from which the current algorithms derive [11]. The typing pattern on a keyboard is currently considered a biometric behavioural characteristic that can be used to identify or authenticate users [16]. The study of this typing pattern is called keystroke dynamics.

When keystroke dynamics was first implemented in a computer environment, mainframes were the only computers available, so this technique first usage was to help identify users in the same computer. Specifically, the intention was to detect when the typing session of a logged user was used by another person. This problem is commonly known as masquerade detection. With personal computers and now with ubiquitous computing, the problem of masquerade detection has become obsolete and keystroke dynamics techniques shifted their focus mainly to user authentication and user authorization [17]. Authentication refers to the action of confirming the identity of a user trying to access the system, while authorization refers to the action of confirming if the already authenticated user is authorized to accomplish certain actions. However, a new approach to user authentication called continuous authentication was created and it refers to the ability of a system to constantly verify if the current user is the one who logged in. This concept is closely related to masquerade detection and benefits from its work.

Keystroke dynamics investigations can focus on the analysis of “fixed text” or “free text” [5]. In the former, all users type their samples employing the same text (such as passwords or short phrases), while in the latter, everything that is written is analysed which, although it makes it more attractive as it broadens the scope of applications, the disadvantage is that free text does not have fixed, easily extracted characteristics since what users are going to type cannot be obtained a priori.

Keystroke dynamics techniques had proven to be useful for hardening authorization and authentication processes [3]. To the best of our knowledge, it does not exist yet a successful attack against a well implemented keystroke dynamics security authentication program. Although not perfect, these techniques are growing in complexity and reliability and promise to be an important part of most secure systems in the near future.

The problem we approach in this paper is related to the issue of continuous authentication. We have a group of users that types freely, generating a large amount of keystrokes. Then, each user keystrokes are separated in groups called sessions. Finally, we face the problem of correctly deciding which typing sessions belongs to the same user. We call this problem users separation.

This type of analysis can be applied to the differentiation of a large group of users typing continuously or, also, to find out if all the keystrokes of one user alone belong to the same person. The last description corresponds to the former problem of continuous user authentication. User separation, then, refers to the ability to distinguish which typing session belongs to which user and it can be really useful within applications that receive text and that are prone to impersonation. These include:

- Webmail applications,
- Remote console sessions, such as SSH or telnet,
- Temporary abandoned console sessions,
- Stolen or temporary abandoned cellular phones,
- Chat applications.

In all of the above situations, user separation techniques can help mitigate the problem of detecting a masquerade or impersonation attack.

This paper has three goals. First, creating a new, labelled and large dataset of free text user sessions is available for everybody. Second, using this dataset verifies and validates the techniques employed in our previous papers [20] [21], where user sessions were grouped by means of their keystroke patterns. Here, “session” stands for a group of keystrokes belonging to the same user. Third and finally, publishing this dataset allows other researchers to compare their techniques.

Unfortunately, to the best of our knowledge there was no known public keystroke dataset of free text that could be used for our investigation. Consequently, our first task was to create a new, large, validated and labelled dataset of digraphs and trigraphs for several users. In our previous paper, we proposed some grouping techniques but it became difficult to verify the results. Our best verification technique consisted in separating the keystroke sessions in half and verifying that both parts got grouped together. Furthermore, we did not have labels because the dataset corresponded to real attacker keystrokes on a honeypot. Consequently, in this paper we needed to verify our techniques with a labelled dataset to obtain useful performance metrics [7].

This paper is organized as follows. In section 2 we review the state of the art. In Section 3 we describe our dataset. In Section 4 we present the proposed methods for evaluation. In Section 5 we describe each of the experiments done. In Section 6 final results are presented along with the corresponding performance metrics. Finally, in Section 7 we present our conclusions.

2. Background

In general terms, investigations have demonstrated that each person has their own typing pattern which, to a greater or lesser extent, can be used to differentiate them.

Some investigations focus on the extraction of characteristics while others directly center on the classification or identification of users. Regarding the former, it has been analysed which patterns must be taken into account in the extraction as well as the possible problems that may arise [8]. Furthermore, genetic algorithms have been implemented to select the most relevant characteristics, thus, significantly reducing classification errors [18]. Among those characteristics, performance improvements have been found with different combined times (for example, the times of depressing and releasing of a key) [4] [15].

The size of the sample and groups of samples are important [15] [13]. Some investigations based on authentication [6] concluded that the user ID and password must have a minimum of eight characters. The size of the group of samples varies depending on the method employed. An excellent performance using Bayesian classification has been reported [19] but carefully considering the size of the group of samples. In the case of a reduced group, the K-Means method is advisable. Although neural networks have proved their good performance, they require a great quantity of samples to operate successfully [1].

According to some publications, rhythm in writing varies through time [16], and there are a number of factors that can alter it: physical, mental and environmental conditions. However, once the user becomes familiar with their writing, the characteristics turn more stable [14]. Another investigation also highlights this relationship [4], which could more easily help classify experienced attackers who operate with a frequent series of commands.

In order to minimize the impact of the conditions that may alter user typing factors, it is suggested that although the total time spent in writing a paragraph can vary, the relations among internal times are maintained [5]. Consequently, the information entered was divided in trigraphs (that is, a cluster of three letters), sorted by time and the distance among samples was calculated. This system has succeeded in identifying and authenticating users. It must be noted that these characteristics do not derive from each individual user, but from the distances that measure the relation between samples; they are not values that depend on a single sample but on a pair of them.

The majority of investigations focus on the analysis of words or short phrases which can be categorized as “fixed text”, where all persons write the same text. Only a few investigations center on “free text” [2].

One of the first researches carried out on free text was able to distinguish the style of four users [10]. A deeper investigation can be found in [9] which, following the methodology employed in [5], applies it to the analysis of free text by only considering the times of key depressing. The above has demonstrated that just a few lines are enough to identify and classify users with a high level of accuracy. Following this course of investigation, it has been proved that only digraphs and trigraphs can be employed [3].

The second goal of the present research is to apply and verify our previous techniques [21] with the new dataset. We can summarize our previous paper as follows. We applied different unsupervised methods to group user sessions together. These methods needed to compute the distance between two sessions in order to cluster them. We applied three different distance measure concepts (A-distance, R-distance and Weighted-distance) based on digraphs and trigraphs, ending with a total of six distance measures. Two clustering algorithms were applied, an adapted K-means and an adapted Subtractive Clustering. Finally, the performance evaluation was carried out with four ad-hoc metrics: Success Rate, Proximity, Within and P. The most important problem of our previous paper was the lack of a large and labelled dataset that could be used to verify the results and techniques. The creation of such a dataset was included in our further work.

Based on the results of the investigation, the following steps were:

- Obtain a large group of samples that allow investigations on the analysis of free text. As an available online dataset could not be found, it became an ideal opportunity to create one that could be used in multiple investigations.
- Perform for the first time a group of user sessions with a set of labelled data and verify that this group coincides with the original data classification.

3. Dataset

The previous section analysed the state of the art of free text keystroke dynamics analysis and free text keystroke dynamics dataset areas. This section describes the main features and design decisions of our dataset and the details of the data capture phase.

The primary motivation for obtaining a labelled dataset of free text keystrokes was the need to verify your previous detection algorithms. In the absence of labels, our previous method relied upon ad-hoc performance metrics. Results were promising, but no significant performance metrics could be obtained. To complete our work, we decided to design and create a new labelled and large dataset of free text keystrokes.

It is not easy to obtain keystrokes from users, especially free text keystrokes. The most common problems to obtain such a dataset are: First, users may not want to share their keystrokes because of privacy concerns. Second, users should type freely and not under supervision, because their actions and mood should be as real as possible. Consequently, we had to find some way to obtain a large number of keystrokes of real users typing freely, without undermining their privacy.

With these restrictions in mind, the required features of the dataset were designed:

- Have a quite large number of users.
- The user has to type during their everyday computer work, regardless of the program, operating system and date.
- Have a large number of keystrokes for each user. Preferably for several days.
- Have the digraphs and trigraphs for each user.
- Respect the privacy of users. There should be no way to find out what the user has typed from the post analysis of the keystrokes.
- The user should not share their computer during the capture of the keystrokes.

To accomplish these design goals, a new and public keylogger utility was developed. In the following subsection we describe our keylogger program.

3.1. KEasyLogger

A new keylogger application, called KEasyLogger, was developed to capture user keystrokes. Its aim is to present a friendly graphic user interface that allows the user to completely control the capture process. The user needs to trust this project and this keylogger in order to voluntarily cooperate. The KEasyLogger application has no malicious intentions and was designed to respect the privacy of the user and to only capture the user's own keys.

The keylogger was designed according to the following requirements:

- It should execute both on Linux and Windows operating systems.
- It should capture all the keystrokes typed in the graphical window system.
- The user should be able to see the complete text recorded at any moment in a meaningful and clear way.
- The user should be able to delete any keystrokes and words at any moment.
- The user should be able to see the state of the keylogger at any moment.
- The user should be able to change the state of the keylogger at any moment.
- The user should be able to generate and store the digraphs and trigraphs at will.
- The user should only send the digraphs and trigraphs to us by hand.
- We should not be able to recover the original text from the digraphs and trigraphs.

At first, the keylogger was mostly distributed among friends and co-workers, because trust was important to obtain useful keystrokes. After several weeks, the keylogger was also given to everyone that wanted to cooperate with the project.

The KEasyLogger application allows us to obtain very useful keystrokes. Users can capture their everyday work including mail typing, chat sessions, web browsing, programs development, administration tasks and document writing. Most of the users forget about the keylogger after some time and we can capture the inner characteristics of their keystroke behaviour. The keylogger was successfully implemented and it gave us a very large number of real keystrokes from real users.

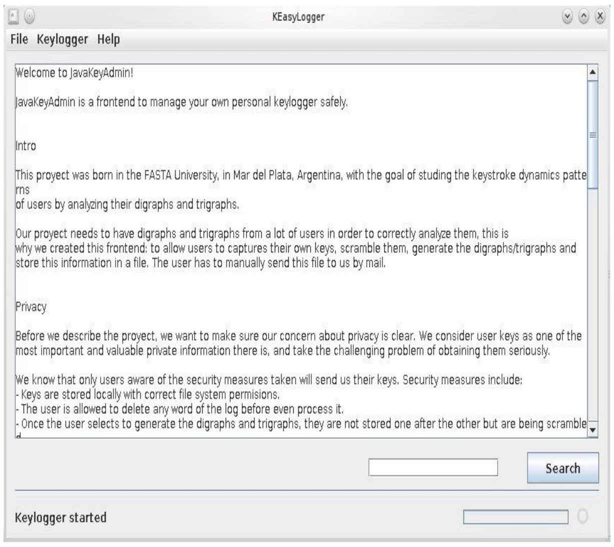
The KEasyLogger is composed of three main subsystems:

- The KEasyLogger.jar file, which is the java GUI frontend.
- The logkeys file, which is the original C++ keylogger.
- The keasylogger-analyser.py file, which is the python log text processor and analyser.

It was important for the KEasyLogger to be useful and visually simple. Figure 1 shows a screenshot of the main frontend, where there are only three menus. Figure 2 shows the File menu, where the user can choose to see or delete content. When a log file is shown, the original text is recreated from the keystrokes. The user can edit the captured keystrokes and can control the keylogger.

As maintaining privacy was paramount for obtaining more users, the application randomly sorted the digraphs and trigraphs from the keystrokes before storing them on disk. Also, the application never connected to internet directly, so the user had to manually send us the text file with the information. Therefore, it was impossible for us to recover the original text from the digraphs and trigraphs received. The keylogger can be downloaded from:

http://www.szsebas.com.ar/keasylogger.



Welcome to JavaKeyAdmin!

JavaKeyAdmin is a frontend to manage your own personal keylogger safely.

Intro

This proyect was born in the FASTA University, in Mar del Plata, Argentina, with the goal of studing the keystroke dynamics patte
rns
of users by analyzing their digraphs and trigraphs.

Our proyect needs to have digraphs and trigraphs from a lot of users in order to correctly analyze them, this is
why we created this frontend: to allow users to captures their own keys, scramble them, generate the digraphs/trigraphs and
store this information in a file. The user has to manually send this file to us by mail.

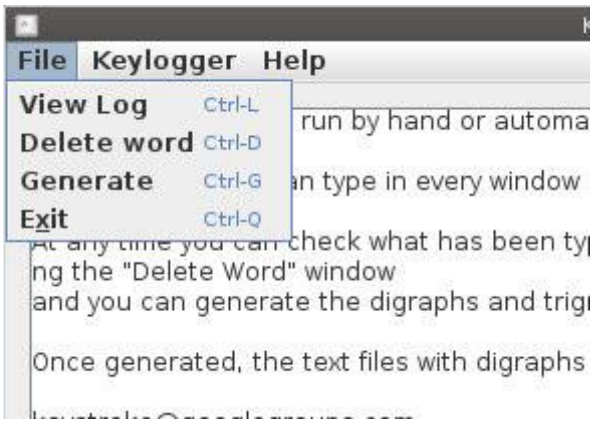
Privacy

Before we describe the proyect, we want to make sure our concern about privacy is clear. We consider user keys as one of the
most important and valuable private information there is, and take the challenging problem of obtaining them seriously.

We know that only users aware of the security measures taken will send us their keys. Security measures include:
- Keys are stored locally with correct file system permisions.
- The user is allowed to delete any word of the log before even process it.
- Once the user selects to generate the digraphs and trigraphs, they are not stored one after the other but are being scramble

Keylogger started

Fig. 1. KEasyLogger frontend



run by hand or automa
an type in every window
check what has been ty
ng the "Delete Word" window
and you can generate the digraphs and trig
Once generated, the text files with digraphs

Fig. 2. KEasyLogger File Menu

3.2. *Dataset Characteristics and publication*

After some time, we managed to obtain a large dataset of real user keystrokes. The main characteristics are:

- 17 unique different users.
- 379 sessions. Each session last one day.
- 2,726,203 total digraphs.
- 2,553,494 total trigraphs.

The dataset was made public in at http://www.szsebas.com.ar/keasylogger.

This is, to the best of our knowledge, the larger public keystroke labelled dataset available to date.

4. Proposed Method

In the previous section we have described the dataset and its main characteristics. In this section we propose a method to analyse the keystroke dataset. The main idea is to use unsupervised algorithms on the data in order to seek natural grouping patterns. Our hypothesis is that every user has their own behavioural pattern which can be used to group user keystrokes together. Our proposal includes modified versions of common clustering algorithms and new distance measures.

Our method is composed of four phases. First, we prepare our dataset by separating keystrokes into sessions, which constitute the minimal comparison unit among users. Second, we compute a set of distance measures among all sessions. These distances were created according to our past experiences in grouping user keystrokes. Third, we apply several modified clustering algorithms to identify natural session groups. Finally, we evaluate the algorithms using the ground-truth labels in the dataset.

In subsection 4.1 we present the dataset preparation for our analysis. In subsection 4.2 we present the dataset distances used to compare the sessions against each other. In subsection 4.3 we specify the algorithms that we use in order to cluster the sessions and obtain the alleged users. In subsection 4.4 we evaluate the quality of the formed clusters to compare and select the best distance measures and clustering algorithm.

4.1. Dataset preparation

Upon receiving all the keystrokes from the users, we needed to pre-process them in order to create our dataset.

To test different hypothesis the dataset was separated into different groups which are described as follows:

- J-1, 5 users only. Users that have the similar amount of sessions (15 to 20).
- J-2, 10 users only. Limited to 18 sessions each. Sessions sorted by the number of digraphs.
- J-3, 10 users only. Limited to 18 sessions each. No sorting.
- J-4, 16 users only. Users have between 1 and 10 sessions. Sessions sorted by the number of digraphs.
- J-5, 16 users only. Users have between 1 and 10 sessions. No sorting.
- J-6, 16 users only. Users have between 1 and 2 sessions. Sessions sorted by the number of digraphs.
- J-7, 10 users only. Users have between 15 and 79 sessions.
- J-8, all the users with all sessions.

The intention was to test the influence of different factors: the number of users and the difference between the number of sessions of each user and the number of digraphs in each session. In this way, we can get a clearer idea of the minimum requirements needed for the system to function. The prepared data was also stored in a database and some of it was dynamically calculated when the tests were performed.

4.2. Distance Measures

In the previous subsection we described the pre-processing steps in our dataset. At this point, the dataset is a collection of user sessions, each of them having digraphs and trigraphs, with their respective time between key presses. However, the clustering algorithms proposed in the next subsection need a proper way to compute the distance between two sessions. In this section, we focus on analysing and creating new distance measures between sessions.

A distance measure is the distance between two sessions. This distance should be computed using the information on each session, that is, digraphs and trigraphs. Defining distance is not an easy task and it determines the way in which the algorithms will behave. Its purpose is to define the degree of similarity of the typing pattern between two sessions. If such distance is 0, it means that both sessions have the same typing pattern. The further the distance, the more different the typing pattern. The best distance would be that which provides a closer distance measure to those sessions belonging to the same user and a more distant one for different user sessions.

The base distances were defined by [5], but they can be combined in different ways (by adding or multiplying them) in order to obtain better results.

In our previous investigation, some of these combinations had been tested [20] [21].

Below is a list of the distances employed. The first six refer to distances created following the paper [5].

From the seventh onwards, they are combinations of the previously described distances. Tests using other combinations alternating different mathematical operators and base distances were carried out, but produced worse results.

- I-1, R-distance digraphs
- I-2, R-distance trigraphs
- I-3, Cumulative-R
- I-4, A-distance digraphs
- I-5, A-distance trigraphs

- I-6, Cumulative-A
- I-7, A-distance digraphs + R-distance digraphs
- I-8, A-distance digraphs + A-distance trigraphs
- I-9, digraphs-A + trigraphs-A + digraphs-R
- I-10, Ponderada-A
- I-11, A-distance digraphs * R-distance digraphs
- I-12, A-distance digraphs + R-distance digraphs * R-distance digraphs
- I-13, A-distance digraphs * A-distance digraphs + R-distance digraphs
- I-14, 2 * A-distance digraphs * R-distance digraphs

The basic idea behind R-distance was introduced in [5] and used in [9]. Given an array V of K elements, a simple measure of the degree of disorder (or, simply, the disorder) of V with regard to its ordered counterpart V' can be computed as the sum of the distances between the position of each element in V and the position of the same element in V'. Given an array of K elements, it is convenient to normalize its disorder by dividing it by the value of the maximum disorder of an array of K elements. In this way, it is possible to compare the disorder of arrays of different size. After this normalization, the disorder of any array V falls between 0 (if V is ordered) and 1 (if V is in reverse order).

R-distance fails to discriminate between the typing samples of two typists that have very similar typing rhythms, even if one of the typists is much faster than the other one. Unlike R-distance, A-distance only considers the absolute value of the typing speed of each pair of identical n-graphs in the two samples under comparison.

Let GS1;d1 and GS2;d2 be the same n-graph occurring in two typing samples S1 and S2, with durations d1 and d2, respectively. The A-distance between S1 and S2 with regard to the n-graphs they share and for a certain value of t is then defined as:

A(S1, S2) = 1 - (number of similar n-graphs between S1 and S2)/(total number of n-graphs shared by S1 and S2)

Where we say that two digraphs are similar if $1 < \max(d1; d2)/\min(d1; d2) \leq t$ for some constant t greater than 1.

It is possible to compute a cumulative absolute distance between two typing samples with regard to n-graphs of different length:

$A_{n,m}(S1; S2) = A(S1, S2)_n + A(S1, S2)_m M/N$

for M m-graphs and N n-graphs shared by S1 and S2 and N > M. We compute Cumulative-R and Cumulative-A using digraphs and trigraphs.

These distances were previously used to compare sessions and performed very well [5] [21]. We decided to carry out an exhaustive search for their combinations and use them in a clustering algorithm, which has not been done before by other researchers.

4.3. *Clustering Algorithms*

The proposed clustering method had to be adapted in order to be meaningful in our context. We decided to use for the context of clustering, features that were good solving other situations. Since there was no classification method that allows using as distance a feature that emerges from the comparison between observations, we had to adapt the K-means algorithm to use a proper distance measure between sessions.

The general idea is to take all sessions, group them and then verify whether those automatically generated and unsupervised groups coincide with the users of the original sessions. As a result, we combined a method that worked well with the problem of identification (comparing a user against all users) with a grouping method (comparing all users against all). Then, sessions were grouped by means of a binary comparison measure. The distances proposed by [5], and consequently, those calculated in this investigation, return an intrinsic value to the comparison between two sessions. An isolated session cannot be obtained; all values arise from the comparison between two (and only two) sessions. As no grouping method of these characteristics could be found, an existent grouping method was adapted. K-Means was selected because of its simplicity and effectiveness and it follows these steps:

1. K sessions are randomly selected so as to obtain the initial cluster centers

2. The distance of each session to each cluster center is calculated
3. Which cluster belongs to each data (session) is determined
4. New cluster centers are determined
5. If there is no variation with respect to step 4, it is interrupted
6. It goes back to step 2

The proposed adaptation was made in step 2. While the original method tends to use the Euclidean distance, we employ one of our metrics used to compare sessions. For the center of each cluster in step 4, one of the sessions had always to be chosen, since an average point that represents the center cannot be calculated (the data components cannot be averaged nor the distance of this point to the sessions can be calculated). The chosen session was the one whose sum of the values of the distances with the rest of the sessions was the lowest.

A distinctive feature of the original method is that results can differ significantly if there is a "bad" selection of the initial cluster centers. That is the reason why throughout the years, the selection of the initial cluster centers has not been left randomly and a smart method has been adopted. We proposed two techniques that showed far better results than random initialization. The first one consists of randomly selecting the first center. Then, for each instance we multiply its distances to the rest of the cluster centers. The instance with the larger value is picked as next cluster centroid. This is done to favour the most far away instances. In this way, the chosen cluster centers are distant from the already picked center, which at first favours the creation of homogenous, equidistant groups. The second pursues the same concept but instead of multiplying the distance to the centers, the minimum distance is picked. In summary:

- K-Means
- Adapted K-Means 0 (K-0)
- Adapted K-Means 1 (K-1)
- Adapted K-Means 2 (K-2)

K-Means-type represents the different ways to initialize K-means.

K-0: Random initialization of centers.

K-1: The first center is random. Then, for each instance we multiply its distances to the rest of the cluster centers. The instance with the larger value is picked as next cluster centroid.

K-2: The first center is random. Then, for each instance we compute the minimum distance to the rest of the cluster centers. The maximum of these distances is picked as next cluster center.

4.4. Evaluation Techniques

There are different ways to verify whether the created clusters coincide with those which were to be created. In our case, we must confirm if the sessions have been grouped per user. Since unsupervised learning methods have been used, the created clusters lack labels; consequently, it cannot be ascertained whether the sessions fell in the "correct" cluster or not and besides, the True Positive Rate or False Positive Rate cannot be employed. On the contrary, other functions which measure how good the groups were formed must be followed.

Following [12], the functions employed to measure the performance of the different tests were: Purity, NMI and RI.

Purity is a simple and transparent evaluation measure. To compute purity, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N. Formally:

$$Purity(\Omega, C) = \frac{1}{N}\sum_k max_j \left|\omega_k \cap c_j\right| \qquad (1)$$

where $\Omega = \omega_1, \omega_1, \ldots, \omega_k$ is the set of clusters and $C = c_1, c_2, \ldots, c_j$ is the set of classes. We interpret ω_k as the set of sessions in ω_k and c_j as the set of sessions in c_j. High purity is easy to achieve when the number of clusters is large – in particular, purity is 1 if each session gets its own cluster. Thus, we cannot use purity to trade off the quality of the clustering against the number of clusters. A measure that allows us to make this trade-off is normalized mutual information.

Normalized Mutual Information (NMI) is a measure that allows to trade off the quality of the clustering against the number of clusters. The function measures the amount of information by which our knowledge about the classes increases when we are told what the clusters are.

$$NMI(\Omega, C) = \frac{I(\Omega, C)}{[H(\Omega)+H(C)]/2} \qquad (2)$$

where I is mutual information:

$$I(\Omega, C) = \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} log \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|}$$

and H is entropy:

$$H(\Omega) = \frac{|\omega_k|}{N} \log \frac{|c_j|}{N}$$

An alternative to this information-theoretic interpretation of clustering is to view it as a series of decisions, one for each of the N(N-1)=2 pairs of sessions in the collection. We want to assign two sessions to the same cluster if and only if they are similar. The Rand Index (RI) measures the percentage of decisions that are correct. A true positive (TP) decision assigns two similar sessions to the same cluster, a true negative (TN) decision assigns two dissimilar sessions to different clusters. There are two types of errors we can commit. A false positive (FP) decision assigns two dissimilar sessions to the same cluster. A false negative (FN) decision assigns two similar sessions to different clusters.

$$RI = \frac{TP+TN}{TP+FP+TN+FN} \qquad (3)$$

5. Experiments

Given the few investigations published on the analysis of keystroke dynamics in free text and the novelty of implementing comparative characteristics for clustering, we decided to carry out a thorough analysis of all the possible combinations.

5.1. *Choosing distances*

We started carrying out a basic comparison regarding which distance performed better. Taking basic distances (I-1 to I-6) a large variety of distances combined with different mathematical operators can be obtained, for instance, I-1 + I-4 is a new distance which has, at the time of the clustering, different results if compared to an analysis of I-1 and I-4 separately. A complete search was performed using the combination of two or three distances with the mathematical operations of addition, subtraction, multiplication and division. The ones which performed better (according to evaluation functions) in an average of 10 groups were I-7 to I-14. I-10 was obtained and introduced in the experiments because of its good results in previous investigations.

The same group, J-5, was used in all the tests since it was quite heterogeneous and a half-way point with regards to the rest of the groups. K-0 was employed as a grouping method.

Having the 14 distances selected, a comparative analysis of all distances was performed resulting in the average of grouping 20 times each of the 8 groups. As shown in table 1, the best distances so far are number 4 (A-distance digraphs), 12 (A-distance digraphs + R-distance digraphs * R-distance digraphs) and 7 (A-distance digraphs + R-distance digraphs). In table 2 we also compared the distance which performed better in each of the groups. Results vary according to the group, even though the general performance of distances 4, 7 and 12 is good (if they are not the bests, they are among the first best ones), each group seems to have its “favourite” distance.

Table 1. Comparison of distances for all groups

i	Purity	NMI	RI	FI
4	0.706	0.790	0.926	0.684
12	0.704	0.790	0.923	0.685
7	0.701	0.792	0.926	0.685
13	0.698	0.791	0.922	0.689
9	0.694	0.786	0.920	0.682
14	0.692	0.779	0.920	0.669
11	0.688	0.776	0.919	0.666
8	0.678	0.767	0.918	0.662
1	0.669	0.751	0.913	0.642
5	0.621	0.693	0.900	0.571
2	0.588	0.633	0.884	0.508
10	0.587	0.696	0.851	0.624
6	0.476	0.586	0.803	0.481
3	0.386	0.449	0.710	0.385

5.2. *Comparing groups*

In order to understand how the data to be analysed should be formed, which conditions are better and what must be avoided, a comparative analysis of groups J-1 to J-8 was carried out: a comparison among groups with an average of performance in each of the 14 distances (Table 3), a comparison among groups with the best 3 distances obtained from the previous step (Tables 4, 5 and 6). The average results of 20 groups for each distance are shown.

Table 2. Best distance for each group.

i	j	Purity	NMI	RI	FI
7	1	0.863	0.908	0.944	0.901
7	2	0.742	0.838	0.935	0.780
7	6	0.721	0.905	0.964	0.622
7	4	0.716	0.849	0.949	0.724
7	5	0.672	0.795	0.941	0.652
7	3	0.660	0.744	0.914	0.670
7	7	0.635	0.650	0.870	0.624
7	8	0.589	0.642	0.886	0.499

Table 3. Comparison among groups, all distances.

i	j	Purity	NMI	RI	FI
A	1	0.751	0.763	0.877	0.790
A	6	0.706	0.884	0.951	0.560
A	4	0.681	0.806	0.927	0.680
A	2	0.661	0.757	0.900	0.695
A	3	0.594	0.665	0.876	0.600
A	5	0.591	0.700	0.902	0.534
A	7	0.542	0.539	0.826	0.532
A	8	0.520	0.568	0.852	0.457

Table 4. Comparison among groups. Distance 12

i	j	Purity	NMI	RI	FI
12	1	0.802	0.860	0.915	0.865
12	6	0.738	0.901	0.965	0.607
12	4	0.737	0.854	0.951	0.737
12	2	0.730	0.829	0.934	0.762
12	5	0.704	0.805	0.948	0.670
12	3	0.683	0.759	0.915	0.693
12	7	0.621	0.640	0.866	0.612
12	8	0.606	0.654	0.888	0.520

Table 5. Comparison among groups. Distance 4

i	j	Purity	NMI	RI	FI
4	1	0.802	0.822	0.906	0.836
4	4	0.756	0.863	0.956	0.755
4	2	0.744	0.830	0.934	0.770
4	6	0.741	0.903	0.964	0.611
4	5	0.673	0.784	0.940	0.632
4	3	0.665	0.742	0.913	0.665
4	7	0.577	0.577	0.858	0.553
4	8	0.558	0.621	0.880	0.491

Table 6. Comparison among groups. Distance 7

i	j	Purity	NMI	RI	FI
7	1	0.863	0.908	0.944	0.901
7	2	0.742	0.838	0.935	0.780
7	6	0.721	0.905	0.964	0.622
7	4	0.716	0.849	0.949	0.724
7	5	0.672	0.795	0.941	0.652
7	3	0.660	0.744	0.914	0.670
7	7	0.635	0.650	0.870	0.624
7	8	0.589	0.642	0.886	0.499

The first conclusion is probably the most expected: a small group like J-1 (5 users) with a large and similar amount of data (all users had between 15 and 20 sessions) is the best option. These results greatly improve (even up to 50%) from those where no previous data analysis is performed. The fact that groups 2, 4 and 6 have the following positions in all the cases demonstrates another expected behaviour: what is important is the number of digraphs in a session and the more of them, the better. The generalized bad performance of 7 and 8 with respect to the rest gives us a remarkable and not that trivial note: it is preferable to discard sessions so as to equal the number of sessions of each user rather than considering all of them. The results obtained from 4, 5 and 6 are always better although they have to evaluate a 60% more of users.

5.3. Comparing k-means

Table 7 first shows a general comparison among the different ways of initializing k-means, taking into account all groups and distances I-4, I-7 and I-12.

Table 7. K-means comparisons for distances I-4, I-7 and I-12

Purity	NMI	RI	FI	K-Means
0.763	0.822	0.932	0.785	2

0.704	0.791	0.925	0.684	0
0.656	0.722	0.902	0.652	1

At first sight, K-2 has the best performance. In order to carry out a more detailed analysis tables 8, 9 and 10 have been elaborated, where the best average performance for each type of k-means and each group is shown, as well as with which distance the performance was carried out. 20 groupings for each distance were performed, which were then averaged to obtain the representative value for each distance and the one that best performed was chosen.

Table 8. Best distance for each group, using K-0

i	j	Purity	NMI	RI	FI
7	1	0.863	0.908	0.944	0.901
11	2	0.747	0.847	0.936	0.788
9	3	0.693	0.773	0.918	0.713
4	4	0.756	0.863	0.956	0.755
12	5	0.704	0.805	0.948	0.670
8	6	0.751	0.912	0.966	0.649
7	7	0.635	0.650	0.870	0.624
12	8	0.606	0.654	0.888	0.520

Table 9. Best distance for each group, using K-1

i	j	Purity	NMI	RI	FI
12	1	0.878	0.933	0.946	0.941
7	2	0.807	0.887	0.950	0.835
8	3	0.669	0.736	0.899	0.685
14	4	0.782	0.891	0.952	0.824
12	5	0.692	0.793	0.938	0.678
2	6	0.789	0.923	0.971	0.691
9	7	0.533	0.498	0.825	0.539
9	8	0.543	0.507	0.864	0.494

Table 10. Best distance for each group, using K-2

i	j	Purity	NMI	RI	FI
7	1	0.982	0.990	0.992	0.991
7	2	0.836	0.932	0.960	0.901
7	3	0.787	0.856	0.945	0.818
7	4	0.855	0.940	0.974	0.913
7	5	0.770	0.863	0.959	0.784
13	6	0.941	0.978	0.991	0.906
7	7	0.721	0.726	0.886	0.722
7	8	0.725	0.733	0.910	0.692

By observing the tables, one can notice about the utilization of K-2:

- Its performance improved considerably.
- The best distance was unified: in all the cases was I-7, except for J-6. However, in this case, I-7 had the 4th position and a performance superior to 0.90 in all the functions.

6. Final Results

The distance with the best performance was I-7: Adistance digraphs + Rdistance digraphs. Table 11 shows the average value of all functions for each group (from now onwards called "performance").

Groups J-1, J-2, J-4 and J-6 had a performance above 90%, which results in the fact that no matter the number of users to be grouped, sessions with a high number of digraphs should be preferred. The performance does not decrease much when users are added but when sessions contain few digraphs.

Likewise in the case of J-8, where all sessions with all users are considered, the performance exceeds 78%.

Table 11. Final results for each group using I-7 and K-2

J	Average Performance
1	0.988
2	0.909
3	0.863
4	0.923
5	0.864
6	0.955
7	0.778
8	0.789

7. Conclusions

The initial objective of the investigation was to group (by means of unsupervised methods) user sessions writing in free text through the analysis of user typing patterns.

A group of labelled data belonging to 17 users was obtained and published, which allowed the validation of previously used methods and which could be utilized to compare further investigations.

A comparison measure that had achieved good results in the identification of users was employed and then adapted to fit the case of grouping. Several alternatives and combinations of that measure were tested to find the best one.

A classic grouping method was adapted to be used with a relational method and improvements to enhance its performance were defined.

It is concluded that having a keystroke dynamics dataset of user sessions and knowing the number of users who participated, it is possible to a great extent, to group the sessions of the same user, regardless of any previous knowledge about the users. A level of accuracy of at least 78% was achieved. If the sessions with fewer digraphs are discarded, a performance that exceeds 90% can be attained, and in the case of fewer users, it can reach 100%.

In summary:

- User sessions can be grouped according to their free text typing pattern.
- The best feature to group sessions is Adistance digraphs + Rdistance digraphs.
- The K-means adapted algorithm generates useful results and is a good algorithm to employ.
- The K-means adapted algorithm with selected initial centres is the best algorithm.
- Sessions with more keywords can be better grouped.
- With new users added to the grouping problem, the performance of the method decreases a little.

8. Future Work

Many questions still need to be answered or expanded. Now that we have a trustable data set and we know that the technique developed works, further experiments could be made.

If this method could be implemented within continuous monitoring software and what could happen if we applied the same distances to a supervised method are the questions that we are planning to answer in future work.

References

[1] Obaidat M. S. and Sadoun B. Verification of computer users using keystroke dynamics. IEEE COMPUTER SOCIETY. 1997.

[2] Ahmed I and Traore A.A. Biometric recognition based on free-text keystroke dynamics. Cybernetics, IEEE Transactions on, 99, 2013. doi: 10.1109/TCYB.2013.2257745.

[3] Warwick A. and Alsultan K. Keystroke dynamics authentication: A survey of free-text methods. IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 4, No 1, 2013.

[4] Aráujo L. C. F., Lizárraga M. G. et al. Autentificación personal por dinámica de tecleo basada en lógica difusa. IEEE COMPUTER SOCIETY, 2005.

[5] Gunetti D. Picardi C. Bergadano, F. User authentication through keystroke dynamics. ACM Transactions on Information and System Security, 5(4):367–397, 2002. doi: 10.1145/581271.581272.

[6] Cheng-Huang J. and Shiuhpyng S. et al. Keystroke statistical learning model for web authentication. Proceedings of the 2nd ACM symposium on Information, computer and communications security. Singapore, ACM, 2007.

[7] Devijver P. A. and Kittler J. Pattern Recognition: A Statistical Approach. Prentice-Hall, Londres, 1982.

[8] Hart P. E., Stork D. G. and Duda R. O. Pattern classification. New York: Wiley, 2001.

[9] Picardi C. and Gunetti, D. Keystroke analysis of free text. ACM Transactions on Information and System Security, 8(3):312–347, 2005. doi: 10.1145/1085126.1085129.

[10] Shepherd S. J. Continuous authentication by analysis of keyboard typing characteristics. IEEE COMPUTER SOCIETY, 1995.

[11] Narendra K. S. and Parthsarathy K. Identification and control of dynamical system using neural networks. IEENN, 1(1):4–27, 1990.

[12] Prabhakar R. M., Christopher D. and Hinrich S. Introduction to Information Retrieval. Cambridge: Cambridge University Press. Cambridge Books Online, 2012.

[13] Paeckock A, Ke X. et al. Typing patterns: A key to user identification. IEEE COMPUTER SOCIETY, 2004.

[14] Mroczkowski P. Identity verification using keyboard statistics. Linkoping University, Electronic Press, 2004.

[15] Liang V., Chambers J., Mackenzie C. and Robinson, J. Computer user verification using login string keystroke dynamics. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 28(2):236–241, 1998. doi: 10.1109/3468.661150.

[16] Kacholia V. and Pandit S. Biometric authentication using random distributions (bioart). 2004. shashankpandit.com.

[17] Mark S. Information Security: Principles and Practice, 2nd edition. Wiley, 2011.

[18] Yu Enzhe and Cho Sungzoon. Biometrics-based password identity verification: Some practical issues and solutions. 2003. Http://dmlab.snu.ac.kr.

[19] Cho Tai-Hoon. Pattern classification methods for keystroke analysis. SICE-ICASE International Joint Conference, 2006.

[21] Sorondo G., Garcıa S., Meschino G. J., Zamonsky Pedernera G. and Sznur S. Revisiting clustering methods to their application on keystroke dynamics for intruder classification. In IEEE Workshop on Biometric Measurements and Systems for Security and Medical Applications, volume 9, pages 36–40, 2010. Taranto, Italia.