A closed sets based learning classifier for implicit authentication in web browsing

Diyé Dia, Giacomo Kahn, Fabien Labernia, Yannick Loiseau, Olivier Raynaud

A closed sets based learning classifier for implicit authentication in web browsing

Diyé Dia[a], Giacomo Kahn[b], Fabien Labernia[c], Yannick Loiseau[b], Olivier Raynaud[b]

[a] *ARISTARC : Liberté 6, Dakar, Sénégal*
diye.die at aristac-conseils.com
[b] *Clermont Auvergne University, LIMOS Laboratory, 63178 Aubière Cedex,*
kahngi, loiseau, raynaud at isima.fr
[c] *Paris-Dauphine University, LAMSADE Laboratory, 75775 Paris Cedex 16,*
fabien.labernia at dauphine.fr

Abstract

Faced with both identity theft and the theft of means of authentication, users of digital services are starting to look rather suspiciously at online systems. To increase access security it is necessary to introduce some new factor of implicit authentication such as user behavior analysis. A behavior is made up of a series of observable actions and taken as a whole, the most frequent of these actions amount to habit. The challenge is to detect identity theft as quickly as possible and, reciprocally, to validate a legitimate identity for as long as possible. To take up this challenge, we introduce in this paper a closed set-based learning classifier. This classifier is inspired by classification in concept lattices from positive and negative examples and several works on emerging patterns. We also rely on the *tf-idf* parameter used in the context of information retrieval. We propose three heuristics named H^c_{tf-idf}, H^c_{sup} and H^c_{supMin} to select closed patterns for each class to be described. To compute performance of our models we have followed an experimental protocol described in a previous study which had the same purpose. Then, we compared the results from our own dataset of web navigation connection logs of $3,000$ users over a six-month period with the heuristic H_{sup} introduced in this study. Moreover, to strengthen our analysis, we have designed and set up one model based on the naive Bayes classifier to be used as a reference statistical tool.

Keywords: Machine learning, Classifier, Closed set, Emerging Patterns, Implicit authentication

1. Introduction

In order to achieve productivity gains, companies encourage their customers to access their services via the Internet. It is accepted that on-line services are more immediate and user-friendly than accessing these services via a brick and mortar agency, which involves going there and, more often than not, waiting around (cf. [8]). Nevertheless, access to on-line services does pose security problems. Certain services provide access to sensitive data such as banking data, for which it is absolutely essential to authenticate the users concerned. However, identity theft is becoming more and more common (cf. [17]). We can distinguish two paradigms for increasing access security. The first one consists of making access protocols stronger by relying, for example, on external devices for transmitting access codes that are supplementary to the login/password pair. Nevertheless, these processes are detrimental to the user-friendliness and usability of the services. The number of transactions abandoned before reaching the end of the process is increasing and exchange volumes are decreasing. The second paradigm consists on the contrary to simplify the identification processes in order to increase the exchange volumes. By way of examples, we can mention single-click payment (cf. [8] and [3]) or using RFID chips for contact-less payments. Where these two paradigms meet is where we find implicit means of authentication.

A means of authentication is a process that makes it possible to ensure that the identity declared in the event of access is indeed the user's identity. Traditionally, a user authenticates himself or herself by providing proof of identity (cf. [9]). This process is called explicit authentication. In contrast, implicit authentication does not require anything from the user but instead studies his or her behavior, the trace left by the user's actions, and then either does or does not validate the declared identity. Thus, an authentication system is *implicit* if it does not require the user to take actions towards his or her authentification (cf. Table 1 and [23] for a survey).

An implicit means of authentication cannot replace traditional means of authentication as it is necessary for the user to have access to his or her service so that the person's behavior may be studied and their identity can either be validated or rejected. However, with efficient implicit means of authentication, it would be possible to avoid stronger authentication modes (such as chip cards and PIN numbers), which are detrimental to the usability of services.

Feature	*Capturing Method*	*Implicit/Explicit*	*Spoofing Threats*
Passcode	Keyboard	Explicit	Keyloggers, Shoulder Surfing
Token	Hardware device	Mainly explicit, implicit possible	None
Face & Iris	Camera	Both	Picture of the legitimate user
Speech	Microphone	Both	Recording of the user's voice
Keystroke	Keyboard	Implicit, explicit possible	Typing imitation (difficult)
Location	GPS	Implicit	Informed strangers
Network	Software protocol	Implicit	Informed strangers

Table 1: Comparison of different authentication methods (extract from [23]).

The challenge is to detect identity theft as quickly as possible and, reciprocally, to validate a legitimate identity for as long as possible. To take up this challenge, we propose to design a classifier. The purpose of this classifier is to guess the corresponding user for a given anonymous behavior. Through this tool, we will be able to authenticate a user as follows: if the classifier correlates the right user behind the current behavior then the user is authenticated, otherwise, the user may be rejected.

This contribution is organized as follows: in Section 2 we discuss the state-of-the-art concerning implicit authentication and user profiles in web browsing, as well as general techniques of classification that use closed sets. In Section 3 we describe the formal frameworks and in Section 4 we present our new learning model for implicit authentication of web users. Bayes model is described in Section 5. Section 6 discusses experimental results. More precisely, in this section we compare performance of methods described above and we analyze all of our results. Finally, in Section 7 we sum up our results.

2. Related work

Classification with closed sets is a subject that has been developped, in particular in concept lattices. Specifically, from the work of V.K. Finn in [4] describing a plausible reasoning in JSM type systems, the authors of [14] and [13] design and formalize a classifier in concept lattices from positive and negative examples. But, by using this binary classifier to our context with n classes (where n stands for the number of users of a given system) we could classify as non-identifiable most of the anonymous behaviors.

In 2007, the authors of [19] defined *emerging patterns* as patterns appearing frequently on the objects in a single class, but being harder to find in objects belonging to other classes (cf.[6] and [18] for surveys on emerging patterns). However, a difficulty remains in selecting the most efficient emerging patterns among the large number of patterns. Intuitively, the authors of [19] used the support measure to define and extract emerging patterns.

In our context of security, implicit authentication systems were studied quickly for mobile phones. In [21], the authors studied behavior based on variables specific to smart phones such as calls, SMSs, browsing between applications, location, and the time of day. Experiments were conducted based on the data for 50 users over a period of 12 days. The data were gathered using an application installed by users who were volunteers. The user profiles were built up from how frequently positive or negative events occurred and the location. Within this context, a positive event is an event consistent with the information gathered upstream. By way of an example, calling a number which is in the phones directory is a positive event. The results of this study show that based on ten or so actions, you can detect fraudulent use of a smartphone with an accuracy of 95%. In a quite different context, the authors of [25] relied on a Bayesian classification in order to associate a behavior class with each video streaming user. The dataset is simulated and consists of $1,000$ users over 100 days. The variables taken into account are the quality of the flow, the type of program, the duration of the session, the type of user, and the popularity of the video. The results are not accurate enough for our needs, because the proposed model has an accuracy rate of 50%.

The particular context of implicit authentication for web browsing was studied in [26], [7], [12] and [1]. In [26], the author adopted the domain name, the number of pages viewed, the session start time and its duration as characteristic variables. The dataset, which was gathered by a service provider, consisted of 300 first connections by $2,798$ users over a period of

12 months. The user profiles consisted of patterns of size 1. The author compares several pattern selection approaches such as the support and the relative support approaches. The study shows that for small, anonymous behavioral patterns (involving up to twenty or so sites visited), the most effective models are still traditional classification models such as decision trees. On the other hand, whenever anonymous behavior exceeds 70 or so sites, the support and relative support-based classification models are more accurate. The study conducted in [1] states that the size of the dataset remains a determining parameter. Their study, conducted on 10 users over a one-month period, did not enable them to build a significant model for distinguishing users. The authors also concluded that no variable taken individually enables a user to be authenticated. Drawing inspiration from a study conducted in [26], the authors of [10] studied several techniques for analyzing a user who holds a dynamic IP address, based on behavioral models. The compared methods are seeking motives, the nearest neighbors technique, and the multinomial Bayesian classifier. The dataset consisted of DNS requests from $3,600$ users over a two-month period. In this study, only the most significant variables and the most popular host names were considered. The accuracy rate for the models proposed were satisfactory.

From our point of view, the study carried out in [26] is the most detailed and accurate. For this reason, we faithfully reproduce here his experimental protocol on our own dataset and we compare performance of our classification heuristics to his specific models.

3. Formal framework

We call a session a set of visited websites by a given user u_i with i in $[1, n]$ and n the number of users. The size of sessions is fixed[1]. The learning database of each user u_i takes the form of a list of sessions denoted $\mathcal{S}_{u_i}$ and is built from log data[2]. More precisely, thanks to a timestamp associated to each visited web site by a given user, we are able to build its set of sessions by taking into account the natural order of the visits. But, after this last step, each session is then considered as a non ordered set of websites. Thus, inside a session the natural order is not taken into account. We call $\mathcal{S} = \bigcup_i \mathcal{S}_{u_i}$ the whole list of sessions of the database. We call X_{u_i} the set of websites visited

[1] In our experimental protocol the best results are obtained with 10. But this choice remains subjective and the size should be a natural parameter of the study. For this reason we give experimental results with sessions of size 5 and 20 in Appendix.

[2] Cf. Section 6

at least once by user u_i and we call $X = \bigcup_i X_{u_i}$ the whole set of websites. Fig. 1 shows our running example.

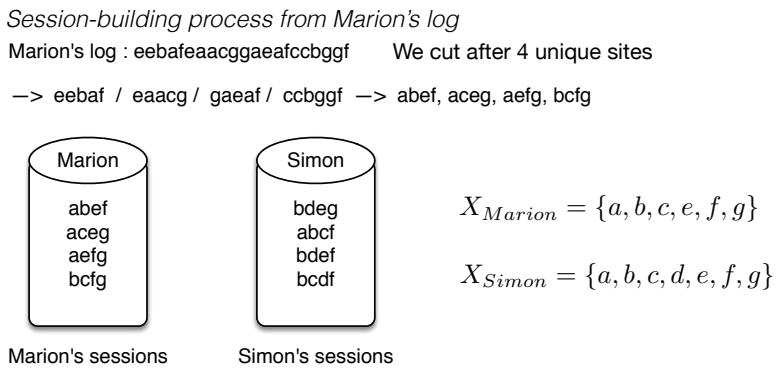

Figure 1: A learning database with two users, Marion and Simon. The set X of visited web site is $\{a, b, c, d, e, f, g\}$. From Marion's logs, we show the building process of her sessions. Each session has to be a set of different websites of fixed size. Here, the size of a session is 4 instead of 10. Each database has 4 sessions. $\mathcal{S} = \mathcal{S}_{Marion} \cup \mathcal{S}_{Simon}$. For short, in the following we write $abef$ instead of $\{a, b, e, f\}$. $\mathcal{S}_{Marion} = \{abef, aceg, aefg, bcfg\}$ and $\mathcal{S}_{Simon} = \{bdeg, abcf, bdef, bcdf\}$.

Definition 1 (k-pattern). *Let X be a set of websites and $\mathcal{S}$ be a list of sessions on X. A subset P of X of size k is called a k-pattern. A session S in $\mathcal{S}$ is said to contain a k-pattern P if $P \subseteq S$.*

Let S be a set, we note $|S|$ the size of S.

Definition 2 (Supports and relative support). *Let $\mathcal{S}$ be a list of sessions and P a pattern. We define the overall support of a pattern P as the percentage of sessions in $\mathcal{S}$ containing P:*

$$support_{\mathcal{S}}(P) = \frac{|\{S \in \mathcal{S} \mid P \subseteq S\}|}{|\mathcal{S}|}$$

By extension the support of a pattern P in the list of sessions $\mathcal{S}_{u_i}$ of a user u_i is :

$$support_{\mathcal{S}_{u_i}}(P) = \frac{|\{S \in \mathcal{S}_{u_i} \mid P \subseteq S\}|}{|\mathcal{S}_{u_i}|}$$

For a given user u_i, the relative strength of a pattern is :

$$relSupport_{\mathcal{S}_{u_i}\mathcal{S}}(P) = \frac{support_{S_{u_i}}(P)}{support_{\mathcal{S}}(P)}$$

The *within-user support* measures the strength of a pattern in behavioral description of a given user. The *relative support* mitigates the within-user support measure by considering the overall support of the pattern.

Example 1 (From our running example). *Let* $P_1 = \{abe\}$ *and* $P_2 = \{af\}$ *be two patterns. We have:*

$$support_{\mathcal{S}}(P_1) = 1/8 \quad support_{\mathcal{S}}(P_2) = 3/8$$

$$support_{\mathcal{S}_{Marion}}(P_1) = 1/4 \quad support_{\mathcal{S}_{Marion}}(P_2) = 1/2$$

$$relSupport_{\mathcal{S}_{Marion}\mathcal{S}}(P_1) = (1/4)/(1/8) = 2$$
$$relSupport_{\mathcal{S}_{Marion}\mathcal{S}}(P_2) = (1/2)/(3/8) = 4/3$$

The tf-idf measure is a numerical statistic that is intended to reflect how relevant a word is to a document in a corpus. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the whole corpus ([20]). In our context, a word becomes a pattern, a document becomes a set of sessions $\mathcal{S}_{u_i}$ of a given user and the corpus becomes the set $\mathcal{S}$ of all sessions.

Definition 3 (tf-idf). *Let* P *be a pattern, let* U *be a set of users and* $U_p \subseteq U$ *such that* $\forall u_i \in U_p$, $support_{S_{u_i}}(P) \neq 0$. *Let* $\mathcal{S}_{u_i}$ *be the list of sessions of a user* u_i *and* $\mathcal{S}$ *a set of sessions. The normalized term frequency denoted* $tf(P)$ *is equal to the within-user support* $support_{S_{u_i}}(P)$ *and the inverse document frequency denoted* $idf(P)$ *is equal to log* $(|U|/|U_P|)$.

$$tf\text{–}idf_{u_i}(P) = tf \times idf_{u_i}(P) = support_{S_{u_i}}(P) \times log\left(\frac{|U|}{|U_P|}\right)$$

Example 2 (From our running example). *Let us consider the list of Marion's sessions and Simon's sessions. We have two users, thus* $|U| = 2$. *Let the patterns* $P_1 = aeg$ *and* $P_2 = bf$, *thus* $|U_{aeg}| = 1$ *since the pattern* aeg *appears only in Marion's sessions and* $|U_{bf}| = 2$ *since the pattern appears in both sets of sessions. We have:*

- $tf\text{–}idf_{Mar}(aeg) = support_{S_{Mar}}(aeg) \times log\ 2 = 1/2$
- $tf\text{–}idf_{Mar}(bf) = support_{S_{Mar}}(bf) \times log\ 1 = 0$

Properties of the tf-idf measure are discussed below (see Property 1).

Definition 4 (Closure system). *Let $\mathcal{S}$ be a list of sessions on the set X of websites. We call closure system and we denote by $\mathcal{S}^c$ the closure under intersection of S to which we add the set X.*

Definition 5 (Closure operator). *Let X be a set. A map $C\colon 2^X \mapsto 2^X$ is a closure operator on X if for all sets A and B in X we have: $A \subseteq C(A)$, $A \subseteq B \Longrightarrow C(A) \subseteq C(B)$ and $C(C(A)) = C(A)$.*

Theorem 1. *Let $\mathcal{S}^c$ be a closure system on a set X. Then the map $C_{\mathcal{S}^c}$ defined on 2^X by $\forall A \in 2^X$, $C_{\mathcal{S}^c}(A) = \bigcap\{S \in \mathcal{S}^c \mid A \subseteq S\}$ is a closure operator on X.*

Definition 6 (Closed pattern). *Let $\mathcal{S}^c$ be a closure system on a set X and $C_{\mathcal{S}^c}$ its corresponding closure operator. Let $P \subseteq X$ be a pattern (i.e. a set of visited sites), we say that P is a closed pattern if $C_{\mathcal{S}^c}(P) = P$.*

This definition is equivalent to the definition of a formal concept of the formal context $\mathcal{K} = (S, X, I)$ where S (the sessions) is the set of objects, X (the websites) the set of attributes and I a binary relation between S and X. If $s \in S$ and $x \in X$, $(s, x) \in I$ is read "session s contains the web site x". For a framework about formal contexts as a representation of closure systems, see [5].

In Fig. 2 we give a graphic representation of the lattice of closed patterns from Marion's set of sessions. The lattice of closed patterns from Simon's set of sessions is given in Appendix (see. Fig. A.10).

Property 1. *Let P be a pattern and $\mathcal{S}_{u_i}$ be the list of sessions of a given user u_i. We have:*

$$tf\text{–}idf_{u_i}(P) \leq tf\text{–}idf_{u_i}(C_{\mathcal{S}^c_{u_i}}(P))$$

Proof: Since $|U_{C_{\mathcal{S}^c_{u_i}}(P)}| \leq |U_P|$ and straightforward from the definition, the tf-idf measure is maximized by the closed patterns.

Remark: Unlike the support measure, the tf-idf does not admit a monotonic behavior with inclusion set operator. Indeed, from the running example, one can check that $tf \times idf_{Mar}(a) = 3.log(2/2) = 0$, $tf \times idf_{Mar}(ae) = 3.log(2/1) = 3$ and that $tf \times idf_{Mar}(aef) = 2.log(2/1) = 2$.

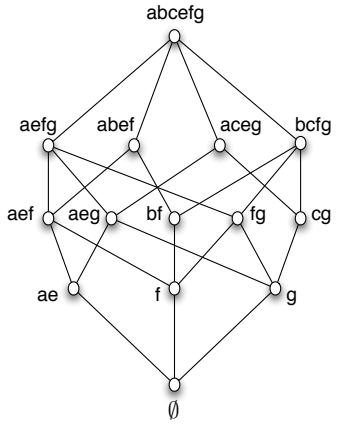

Figure 2: The Hasse diagram of the closure system $\mathcal{S}^c_{Marion}$. The set aeg is a closed 3-pattern of Marion, and f is a closed 1-pattern.

4. Models for a learning closed-sets based classifier

The approach we have chosen to implement our classifier is based on three points:

1. *Closed patterns selection*: From a dataset of web browsing logs we compute a set of closed patterns for each user. We call those patterns *own patterns* for a particular user. The measure used to select the patterns and the size of these patterns may vary. See Def. 7 and Fig. 3 for a graphic representation.

2. *User profiles computation*: We compose a vector space that is common to all users from the closed patterns. Then we compute a vector profile for each user. All users are embedded in a common space with a similarity function.

3. *Identification step*: Thanks to the *profiles* computed in step 2, we are able to provide an identification for anonymous sessions by searching the nearest neighbor in the vector space. We then compute confusion matrices and we provide accuracy of the models.

In the remainder of this section, we will develop each of the aforementioned steps.

4.1. Own patterns selection

Definition 7 (Own patterns). *Let H_{cond} be a heuristic that selects closed patterns under condition cond for a given user u_i. Then the set of patterns selected by H_{cond} for u_i is called the set of* own patterns *of u_i and denoted $\mathcal{P}_{u_i}$.*

The first and most important step of our model, called *own patterns selection* is to calculate the set of own patterns for each user u_i. This set of patterns is denoted $\mathcal{P}_{u_i} = \{P_{i,1}, P_{i,2}, ..., P_{i,p}\}$ (See Fig. 3). Our experimental results show that the value of p should belongs to the interval $[100 - 140]$ to obtain the best performance. In [26], each class is described by a set of 1-pattern with the best *support* or the best *relative support*. We call these approaches H_{sup} and $H_{relSupport}$. The aim of our study is to show that it could be more efficient to select closed k-patterns. But, the number of closed patterns should be strong, thus we compare three heuristics (H^c_{tf-idf}, H^c_{sup} and H^c_{supMin}) to select the p closed k-patterns of each user. For each heuristic, closed patterns are computed thanks to the Charm algorithm ([28]) provided by the Coron platform ([24]).

The main principles of the different heuristics are summarized below:

1. H_{sup} and $H_{relSupport}$: *p 1-patterns with the largest support/relative support values;*
2. H^c_{tf-idf}: *p closed k-patterns with the largest tf-idf values;*
3. H^c_{sup}: *p closed k-patterns with the largest support values;*
4. H^c_{supMin}: *p closed k-patterns with the largest support and minimal by inclusion set operator.*

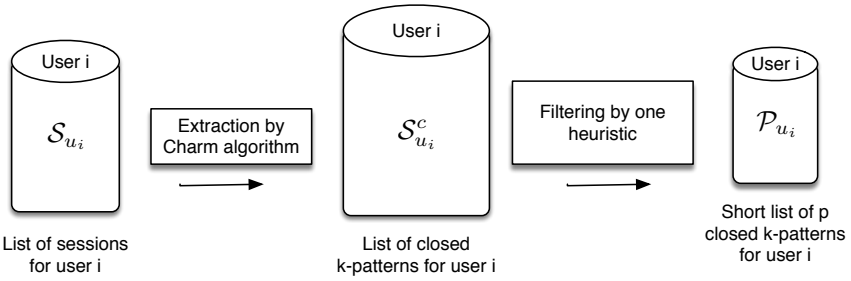

Figure 3: Closed k-patterns selection thanks to Charm algorithm and filtering by one of the heuristics H^c_{tf-idf}, H^c_{sup} and H^c_{supMin}).

Specifically, heuristic H^c_{tf-idf} is a simplified process based on the *tf-idf* measure to select the closed own patterns (cf. Algorithm 1). As H_{sup}, the Process H^c_{sup} use the *support* measure to select the closed patterns. We will be able to compare their performance to show that closed patterns are more representative of a behavior. As previous processes, H^c_{supMin} (cf. Algorithm 2) use the *support* measure to select the closed patterns but only retain patterns which are minimal by inclusion set operator. This way, no pattern could be included in another one. Comparative performance of the heuristics are given in Fig. 6.

Algorithm 1: H^c_{tf-idf}: p closed k-patterns with the largest *tf-idf* values.

Data:
$\mathcal{C}_{u_i}$: the set of closed patterns of user u_i from Charm;
p : the number of selected own patterns.
Result: $\mathcal{P}_{u_i}$: the set of p own patterns of user u_i;
begin
  Compute the tf–idf for each pattern from Charm;
  Sort the list of patterns in descending order according to the tf–idf value;
  Return the top p patterns;

Algorithm 2: H^c_{supMin} : p closed k-patterns with the largest *support* and minimal values by inclusion set operator.

Data:
$\mathcal{C}_{u_i}$: the set of closed patterns of user u_i from Charm;
(by decreasing order of *support*);
p : the number of selected own patterns.
Result: $\mathcal{P}_{u_i}$: the set of p own patterns of user u_i;
begin
  $\mathcal{P}_{u_i} \leftarrow \emptyset$;
  while $(|\mathcal{P}_{u_i}| < p)$ **do**
    Take next $C \in \mathcal{C}_{u_i}$;
    if $(\forall P \in \mathcal{P}_{u_i},\ P \not\subseteq C)$ **then**
      $\mathcal{P}_{u_i} \leftarrow \mathcal{P}_{u_i} \cup \{C\}$;
  Return $\mathcal{P}_{u_i}$;

Example 3 (From our running example). *The value of p is fixed at* 4 *(instead of* 10 *in the experimental protocol). We give Marion's own patterns for the four approaches. Marion's closed patterns can be found in Fig. 2.*

We have:

1. H_{sup}*:* 4 *1-patterns: own patterns are taken in* $\{a, b, c, e, f, g\}$*. The support function is max for each* P *in* $\{a, e, f, g\}$*. Thus* $\mathcal{P}_{Marion} = \{a, e, f, g\}$*.*
2. H^c_{tf-idf}*:* 4 *k-patterns with the best tf-idf. We give the first closed patterns and the corresponding tf-idf values together in pairs:* $(ae, 3/4), (aeg, 1/2), (aef, 1/2), (fg, 1/2), (cg, 1/2), ...$ *Since* $p = 4$*, we can choose* $\mathcal{P}_{Marion} = \{ae, aef, aeg, fg\}$*.*
3. H^c_{sup}*:* 4 *k-patterns with the best support. Some execution of this process may compute* $\mathcal{P}_{Marion} = \{ae, f, g, aef\}$*.*
4. H^c_{supMin}*:* 4 *k-patterns with the best support and minimal by inclusion. Algorithm 2 gives only* 3 *k-patterns.* $\mathcal{P}_{Marion} = \{ae, f, g\}$*. Indeed, other closed k-patterns are not minimal.*

4.2. User profiles computation

We define and we denote $\mathcal{P}_{all} = \bigcup_i \mathcal{P}_{u_i}$ the set of all own patterns for all users. The set $\mathcal{P}_{all}$ allows us to define a common space in which all users could be embedded. More formally, $\mathcal{P}_{all}$ defines a vector space $\mathcal{V}$ of size $all = |\mathcal{P}_{all}|$ where a given user u_i is represented as a vector $V_{u_i} = \langle m_{i,1}, m_{i,2}, ..., m_{i,all} \rangle$ (cf. Fig. 4). In the following we will refer to V_{u_i} as the profile vector of the user u_i.

Figure 4: The set of own patterns denoted $\mathcal{P}_{all} = \bigcup_i \mathcal{P}_{u_i}$ defines a vector space $\mathcal{V}$.

The second step of our model, called *user profile computation*, is to compute, for each user u_i, a numerical value for each component $m_{i,j}$ of the vector V_{u_i}. i is the user id, $j \in [1, all]$ is a pattern *id* and m stands for a given *measure*. In our model, we use three measures: the *support*, the *relative support* and the *tf-idf*. We have respectively:

$$m_{i,j} = support_{\mathcal{S}_{u_i}}(P_j), \quad m_{i,j} = relSupport_{\mathcal{S}_{u_i}\mathcal{S}}(P_j) \quad \text{or} \quad m_{i,j} = tf\text{–}idf_{u_i}(P_j)$$

Note that even if the choice of this measure can be seen as a parameter of the model, experimental results confirm that best accuracies are obtained when the measure is the same as that used to select own patterns. Specifically, for heuristics H_{sup}, H^c_{sup} and H^c_{supMin} we will use the *support* measure, and for Heuristic H^c_{tf-idf} (resp. $H_{relSupport}$) we will use the *tf-idf* measure (resp. the *relative support* measure).

Example 4. *For our running example, suppose that we have applied* H^c_{sup} *heuristic to compute the own patterns of Marion and Simon and a third given user Jean.* $\mathcal{P}_{Marion} = \{ae, aef, aeg, fg\}$, $\mathcal{P}_{Simon} = \{b, bf, bd, bcf\}$ *and* $\mathcal{P}_{Jean} = \{ag, aef, bcf, fg\}$. *The size of the vector space* $\mathcal{V}$ *is equal to* 10. *We give in Fig. 5 profile vectors of Marion and Simon computed with the support measure.*

V_{Marion}

3/4	3/4	3/4	1/2	1/2	1/2	0	1/4	1/2	1/2
ae	f	g	aef	b	bf	bd	bcf	ag	fg

V_{Simon}

0	3/4	1/4	0	1	3/4	3/4	1/2	0	0
ae	f	g	aef	b	bf	bd	bcf	ag	fg

Figure 5: Profile vectors for Marion and Simon with the *support* measure. The size of the space vector $\mathcal{V}$ is 10, and not 12 since Jean shares 2 own patterns with Marion and Simon (namely aef and bcf).

4.3. Identification step

Identification step is to guess the user corresponding to an anonymous set of sessions. Note that all the sessions of a given anonymous set are naturally coming from a same user. Then, for each anonymous set of sessions we have to build a test profile and to find the nearest user profile defined during the learning step.

4.3.1. Test sessions

Performance of our models are calculated on anonymous sets of sessions of growing size, specifically from 1 session to 35 sessions. The more sessions we have in an anonymous set, the better the classification will be.

4.3.2. Building test profile

Let $\mathcal{S}$ be the set of all sessions from the data test set. Let $\mathcal{S}_{u_t}$ be an anonymous set of sessions from user u_t, and $V_{u_t} = \langle m_{t,1}, m_{t,2}, ..., m_{t,all} \rangle$ its corresponding profile vector. As we have done it on the learning step, we use three measures: the *support*, the *relative support* and the *tf-idf*. We have respectively:

$$m_{t,i} = support_{\mathcal{S}_{u_t}}(P_i), \quad m_{t,i} = relSupport_{\mathcal{S}_{u_t}\mathcal{S}}(P_i) \quad \text{or} \quad m_{t,i} = tf\text{–}idf_{u_t}(P_i)$$

4.3.3. Distance functions

Let $V_{u_i} = \langle m_{i,1}, ..., m_{i,all} \rangle$ and $V_{u_t} = \langle m_{t,1}, ..., m_{t,all} \rangle$ be two profiles. We denote $Dis_{Euclidean}(V_{u_i}, V_{u_t})$ and $Sim_{Cosine}(V_{u_i}, V_{u_t})$ the Euclidean distance and the cosine similarity between two vectors V_{u_i} and V_{u_t} respectively, where:

$$Dis_{Euclidean}(V_{u_i}, V_{u_t}) = \sqrt{\sum_j (m_{t,j} - m_{i,j})^2}$$

$$Sim_{Cosine}(V_{u_i}, V_{u_t}) = \frac{\sum_j (m_{t,j} \times m_{i,j})}{\sqrt{\sum_j (m_{t,j})^2 \times \sum_j (m_{i,j})^2}}$$

Note: Formally an *Euclidean* distance function has to be used in a linear independent space. Even if the vector space $\mathcal{V}$ is not exactly linearly independent, the results of our experimental tests look interesting (cf. Fig. 8).

We propose to test performance of two other similarity functions: the *Kulczynski* measure introduced in 1927 and the *Dice* similarity introduced independently by L.R Dice in 1945 and T. Sorensen in 1948. Both measures are statistical tools used to compare two vectors in $\mathcal{R}^n$.

$$Sim_{Kulczynski}(V_{u_i}, Vu_t) = \frac{\sum_j \min(m_{t,j}, m_{i,j})}{\sum_j |m_{t,j} - m_{i,j}|}$$

$$Sim_{Dice}(V_{u_i}, Vu_t) = \frac{2 \times \sum_j (m_{t,j} \times m_{i,j})}{\sum_j (m_{t,j})^2 + \sum_j (m_{i,j})^2}$$

Comparative performance of the two latter similarity functions are given in Fig.8.

5. Bayesian models

By keeping the formal framework described in the previous section we call $\mathcal{S} = \bigcup_i \mathcal{S}_{u_i}$ the whole set of sessions of the database, $\mathcal{S}_{u_i}$ being the list of sessions of a given user u_i. By this way, a set of n users defines a class partition of the learning database. A Bayesian classifier states that a given anonymous session of size l, $S = \{s_1, s_2, ..., s_l\}$ (each s_i is a web site in our context), is assigned to a user i if and only if for all j in $[1, n]$, $i \neq j$ we have $P(u_i|S) > P(u_j|S)$.

From the Bayes's theorem :

$$P(u_i|S) = \frac{P(S|u_i)P(u_i)}{P(S)}$$

5.1. Traditional Bayes classifier

In practice, there is interest only in the term $P(S|u_i)$ of that fraction. Under the independence assumptions, the conditional distribution over the class variable is:

$$P(S|u_i) = P(s_1|u_i) \times P(s_2|u_i) \times ... \times P(s_l|u_i)$$

For all k in $[1, l]$, $P(s_k|u_i)$ is calculated from the learning database. A visited web site s_k is not a continuous value then we can define $P(s_k|u_i)$ as the number of sessions of u_i containing the visited web site s_k divided by the total number of u_i's sessions.

5.2. The smoothed Bayes classifier

The major drawback with the traditional Bayes classifier is that a descriptor s_k of an anonymous session, which never occurs in the learning dataset of a given class, will produce a probability equal to zero. To avoid this problem, we have applied the Laplace smoothing, also called add-one smoothing, which consists of adding one to the support of each descriptor appearing in the learning dataset of the given class. This way, the missing descriptor will have a support of $1/m$ (m being the number of sessions) and all other descriptors will see their support incremented by one. Thus, the frequency-based probability estimate will never be zero. When referring with the Bayes classifier, we are discussing the classifier on which we have applied the Laplace smoothing.

6. Experimental results

Our dataset comes from Blaise Pascal University proxy servers. It consists of 17×10^6 lines of connection logs from more than $3,000$ users and contains the user ID, the timestamp and a domain name for each line. We applied two types of filters on the domain names: blacklist filters and HTTP-request based filters. We used some lists[3] of domain names to remove all domains regarded as advertising. We also filtered the data by the status code obtained after a simple HTTP request on the domain name. After those steps, we still have 4.10^6 lines. We divide the file between the 3000 users to obtain the class files. The studies were conducted on the 150 users with the highest number of requests. Table 2 gives the detailed statistics for this dataset[4].

68.481 sessions	Mean	Minimum	Maximum	Std dev
# sessions/users	456	209	1909	318

Table 2: Descriptive statistics of the used dataset: number of sessions per user on the set of 150 users with the higher number of requests.

6.1. Experimental protocol: a description

Algorithm 3 describes our experimental protocol. The first loop on line 2 sets the size of the anonymous set of sessions to be classified to each attempt. For example, if $S = 1$, the classifier has to find the user behind one anonymous session, if $S = 10$, the classifier has to find the corresponding user of a set of 10 sessions. It is more difficult when the size of S is smaller. The loop on line 8 computes the specific patterns of each user and establishes the profiles vector. The loop on line 11 computes the vector's components for each user. The nested loops on lines 14 and 15 classify test data and compute the accuracy rate.

[3] `http://winhelp2002.mvps.org/hosts.htm` and `https://pgl.yoyo.org/as`.
[4] This dataset is available at `https://fc.isima.fr/~kahngi/cez13.zip`.

Algorithm 3: Experiment procedure

Data: $\bigcup_i \mathcal{S}_{u_i}$: all sessions from n users; p the number of own patterns;
x: the number of executions to smooth results;
Result: The mean accuracy of selected models;

```
1  begin
2      for (S = {1, 3, 5, 10, 20, 30, 35}) do
3          for (k = 1, ..., x) do
4              for (each u_i, i = 1, ..., n) do
5                  S'_{u_i} ← 2/3 of S_{u_i} to form the learning set;
6                  T_{u_i} ← 1/3 of S_{u_i} to form the testing set;
7              P^k_all ← ∅ (the global profile vector for each k);
8              for (each u_i, i = 1, ..., N) do
9                  Compute the p own patterns P^k_{u_i} on S'_{u_i};
10                 P^k_all ← P^k_all ∪ P^k_{u_i};
11             for (each u_i, i = 1, ..., n) do
12                 Compute the vector V^k_{u_i} with support, relative support
                   or tf.idf;
13             Initialize to 0 the confusion matrix M^k of the execution k;
14             for (each u_i, i = 1, ..., n) do
15                 while (T_{u_i} ≠ ∅) do
16                     Take S sessions from T_{u_i} to compute V^k_T;
17                     u_a ← max(simil(V^k_{u_i}, V^k_T)) or min(dist(V^k_{u_i}, V^k_T));
18                     M^k[u_i][u_a] ← M^k[u_i][u_a] + 1;
19         Compute the mean accuracy of M^k[][] on k;
```

6.2. Comparative performance of H^c_{tf-idf}, H^c_{sup} and H^c_{supMin}

We have followed the protocol described above by executing our three heuristics on our dataset of 150 users (cf. Table 2) with anonymous sets of sessions of growing size, from one sessions to 35 sessions. The number of own patterns per user is fixed to 140 and their maximal size to 7. Each result is smoothed to 10 executions. Following Fig. 6 shows comparative performance of the heuristics and our naive Bayes classifier.

We are able to present four significant results. Firstly, for the three heuristics other than the Bayes classifier, models reach an accuracy close to

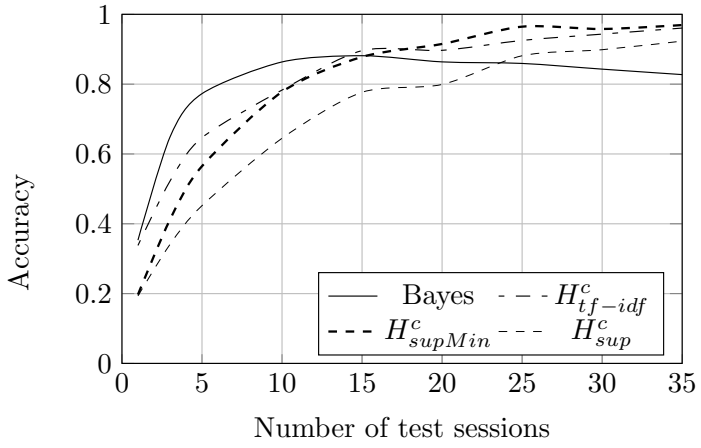

Figure 6: Comparative performance of H^c_{tf-idf}, H^c_{sup}, H^c_{supMin} and Bayes on the dataset of 150 users. The number of own patterns per user is fixed to 140 and their maximal size to 7. Measured values are smoothed on 10 executions.

100% if the size of the anonymous set of sessions is beyond 25. Secondly, Heuristic H^c_{tf-idf} and our Bayes classifier are able to recognize one user among 150 with only one session (10 visited websites) with an accuracy close to 35%. The accuracy reaches 50% (resp. 60%) with an anonymous set of 3 sessions for H^c_{tf-idf} (resp. for the Bayes model). Thirdly, performance of Heuristic H^c_{sup} is significantly lower than the others. Finally, when the number of sessions in the anonymous set is around 15, the Bayes classifier reaches a plateau, and the different heuristics except H^c_{sup} have an accuracy of 90%.

6.3. Comparative performance with H_{sup} and $H_{relSupport}$

In [26], the author compares, in particular, two methods called H_{sup} and $H_{relSupport}$ to select own 1-patterns. H_{sup} (resp. $H_{relSupport}$) selects the patterns with the best *support* (resp. the best *relative support*) and uses it as the numerical value for each component of the profile vector. In both cases, the own patterns are size 1. In order to compare performance of the H^c_{sup} and H^c_{tf-idf} models with H_{sup} and $H_{relSupport}$ we have accurately replicated the experimental protocol given in [26] on our own dataset. The results are given in Table 3 and Table 7 for a zoom.

# Users	# Sessions	Bayes	H_{Sup}	H^c_{sup}	$H_{relSupport}$	H^c_{tf-idf}
2	1	0.88	0.83	0.82	0.95(0.65)	**0.96** (0.63)
	10	0.97	**1**	**1**	0.95	**1**
	20	1	1	1	1	1
	30	1	1	1	1	1
10	1	0.71	0.55	0.53	**0.83**(0.64)	0.62
	10	**0.99**	0.88	0.88	0.94	0.97
	20	0.91	**1**	**1**	0.98	**1**
	30	0.96	**1**	**1**	**1**	**1**
50	1	0.47	0.31	0.30	**0.53**(0.85)	0.39
	10	0.90	0.89	0.88	0.87	**0.92**
	20	0.91	0.94	0.94	0.93	**0.99**
	30	0.88	0.98	0.98	0.98	**1**
100	1	0.40	0.25	0.24	**0.43**	0.33
	10	**0.88**	0.83	0.82	0.84	**0.88**
	20	0.88	0.95	0.95	0.92	**0.98**
	30	0.87	0.98	0.98	0.95	**1**
150	1	0.35	0.21	0.20	**0.39**	0.28
	10	**0.85**	0.80	0.78	0.79	**0.85**
	20	0.86	0.94	0.93	0.90	**0.98**
	30	0.84	0.98	0.97	0.92	**1**

Table 3: On the left we find the number of users and the size of the anonymous set of sessions. Sessions are of size 10. Measured accuracy rate are smoothed on 10 executions. The highest values are in bold. To compare fairly the different approaches we applied the best parameters to each one. This way, we used the *cosinus* similarity for H_{Sup}, H^c_{Sup} and $H_{relSupport}$ and the *Kulczynski* similarity for H^c_{tf-idf}. All methods computes own patterns of size one (i.d. 1-patterns).

In Table 3 we compare three classes of methods: the smoothed Bayes classifier as a statistical approach, H_{Sup} or H^c_{sup} which rely on the *support* measure to obtain a formal description of each user, and $H_{relSupport}$ or H^c_{tf-idf} which rely on the *relative support* or the *tf-idf* as a relative measure. From the *relative support* or the *tf-idf* our classifier is able to compute a description of each user, which sets it apart from others.

From Table 3 we would like to highlight several key points. Firstly, with 2, 50, 100 or 150 users, the results of the classifier are not so far from each other for a given classifier. We conclude that it is possible to identify one user among a large amount of users by learning its visited websites. Secondly, regardless of the method, from an anonymous set of 10 sessions

or more (in other words beyond 100 visited websites), the accuracy of the classifier is beyond 80%, even with 150 users.

The challenge is then to identify a user from the smallest possible number of sessions. In the case we have only one session to identify the given user, the Bayes method and H^c_{tf-idf} give very similar results that are significantly better than results from H_{Sup} or H^c_{sup}. $H_{relSupport}$ over performs all methods in that case with an accuracy of 39% with 150 users.

Finally, one can observe that results obtained with a relative measure as the *relative support* or the *tf-idf* are significantly better than results obtained with the *support*. Moreover, select own patterns with the best support is not less efficient than select frequent closed own patterns since H_{Sup} or H^c_{sup} give exactly the same results. Comparative performance of H_{sup} and $H_{relSupport}$ on one side and H_{Sup}, H^c_{sup} and H^c_{supMin} one another side, are given in Appendix (see. Fig.B.11 and C.12).

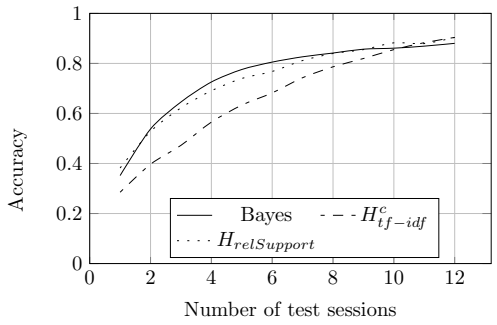

Figure 7: Comparative performance of H^c_{tf-idf}, $H_{relSupport}$ and Bayes on the dataset of 150 users. The number of own patterns per user is fixed to 140 and their size to 1. Measured values are smoothed on 10 executions.

6.4. Distribution of own patterns according to size

From the own patterns of each user we compute the set $\mathcal{P}_{all}$ as the set of all own patterns used to define the profile vector. In order to understand the impact of the chosen heuristic on the selected own patterns, Table 4 provides the distribution of $\mathcal{P}_{all}$ according to size.

	1	2	3	4	5	6	7
H^c_{tf-idf}	8.7%	25.8%	30.3%	20.8%	9.7%	3.5%	1.2%
H^c_{sup}	11.7%	33.6%	33.1%	15.5%	4.1%	0.8%	1.2%
H^c_{supMin}	87.6%	9.5%	1.8%	0.5%	0.16%	0.02%	0.01%

Table 4: For each heuristic, the distribution of the own patterns according to size. As an example, with Heuristic H^c_{Sup}, 33.1% of the patterns are of size 2. The number of own patterns per user is fixed to 140 and their maximum size to 7. Measured values are smoothed on 30 executions.

As we can observe in the Table 4, the distribution may vary considerably depending on the method used. Heuristic H^c_{tf-idf} produces a nice normal distribution centered at 3 for a maximum proportion of 30.3%. Heuristic H^c_{Sup} produces a more compact normal distribution with 66.7% of own patterns centered within the interval $[2-3]$.

Finally, Heuristic H^c_{supMin}, which selects closed own patterns which are minimal by inclusion set operator, produces an unbalanced distribution with 87.6% of own patterns of size one and 9.5% of size two. All these results are obtains with a number of own patterns per user equal to 140. If this number falls to 50 the distribution makes a shift to patterns of lower size (see Table 5).

	1	2	3	4	5	6	7
H^c_{tf-idf}	13.8%	32.6%	29.81%	15.8%	5.7%	1.6%	0.5%
H^c_{sup}	21.8%	44%	27%	6.3%	0.5%	0.1%	0.01%
H^c_{supMin}	96%	3%	0.48%	0.08%	0.04%	0.005%	0.003%

Table 5: For each heuristic, the distribution of the own patterns according to size. The number of own patterns per user is fixed to 50 and their maximum size to 7. Measured values are smoothed on 30 executions.

6.5. Comparative performance of distance or similarity functions

We show in Fig. 8 and 9 the effect of the similarity function on model accuracy rate. *Cosine* and *Dice* similarity measures have same accuracy rate and rate of the *Euclidean* distance are always lower than the others.

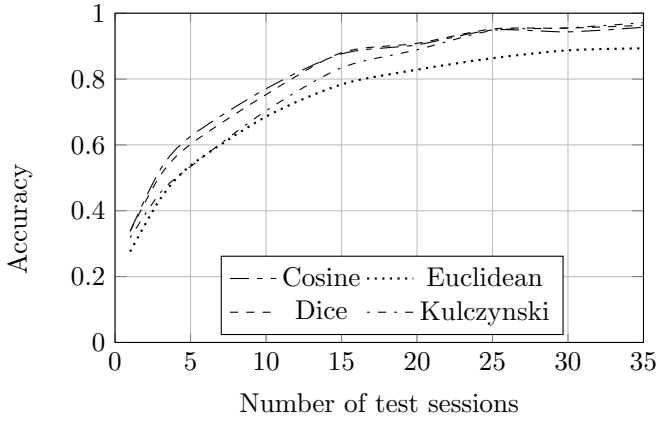

Figure 8: Comparative performance of H^c_{tf-idf} with *Cosine* similarity, *Dice* similarity, *Kulczynski* similarity and the *Euclidean* distance. The number of sessions of the anonymous set is on the X-axis and accuracy rate on the Y-axis. Number of users is equal to 150. Measured values are smoothed on 10 executions.

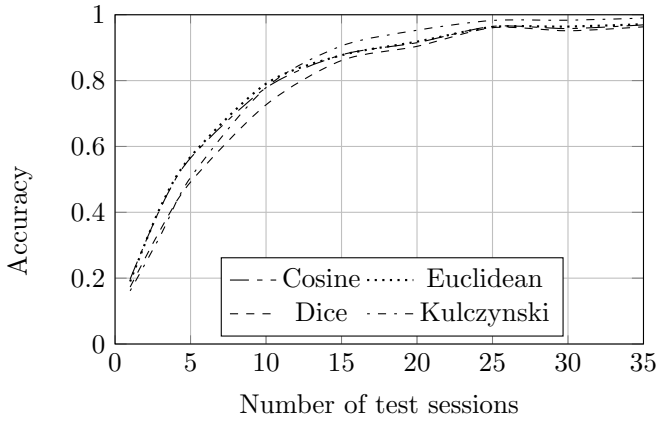

Figure 9: Comparative performance of H^c_{supMin} with *Cosine* similarity, *Dice* similarity, *Kulczynski* similarity and the *Euclidean* distance. The number of sessions of the anonymous set is on the X-axis and accuracy rate on the Y-axis. Number of users is equal to 150. Measured values are smoothed on 10 executions.

7. Analysis and conclusion

In this study, we addressed the issue of authenticating a user based on its behavior in a system. In our experimental context, a behavior would consist of a list of websites visited by each user, but the model could be extended to a set of functions to be executed on a given system. For this purpose we have compiled a large dataset of 17.10^6 connection logs for $3,000$ users. This dataset has been cleaned and filtered, and we have designed a set of $68,481$ sessions for 150 users. To identify each user from the others we have introduced a closed set-based learning classifier which can be described as a combination of the binary classifier studied in [14] and classifiers based on emerging patterns described in [19] and [6]. To select the best patterns from the huge amount of closed patterns extracted from the dataset of each user, we compared several approaches based on the *support*, the *relative support*, and the *tf-idf* measures. To complete the study, we have designed a smoothed Bayes classifier and we compared the set of models by reproducing the experimental protocol described in [26] on our own dataset.

As explained in the introduction, the challenge is for example to detect identity theft on a system used by numerous users as quickly as possible. This way, there are two main parameters. The first lies in the number of users of the system and the second lies in the number of sessions[5] available to the classifier to identify one user amongst the others. The lower the number of sessions, the more difficult the problem is. Our experimental results show that even with 150 users all heuristics have an accuracy rate beyond 80% if the anonymous set contains 10 or more sessions (see Table 3). Specifically, in that case Heuristic H^c_{tf-idf} is the most efficient and reaches 100% with 20 sessions or more. If the anonymous set of sessions contains between one and ten sessions, heuristics based on relative measures such as the *relative support*, and the *tf-idf* have much better accuracy rate than the *support* measure. In particular, Heuristic $H_{relSupport}$ has an accuracy rate of 40% to identify one user among 150 with only one session. The smoothed Bayes classifier has very good accuracy rate even with small anonymous set of sessions but reaches a plateau with an accuracy of 85% for around 15 sessions in the anonymous set (see Fig. 6).

We recall that the design of our models depends on some others parameters such as the number of own patterns for each user, the maximal size of these own patterns, the size of a session and the distance/similarity function.

[5] In our study a session contains 10 unique visited websites.

All theses parameters can be set with our classifier.

Number of own patterns for each user. In the present study, we have shown two major effects of increasing the number of own patterns to describe each user. Firstly, if this number is smaller than 50 and if the anonymous set of sessions contains only one session, then a significative number of these sessions cannot be classified. Indeed, to classify a session (correctly or not) the classifier needs to find at least one match between the profile vector (seen as a list of patterns), and this anonymous session (See Fig. D.13 in Appendix). Otherwise the vector is null. Secondly, as seen in Table 4 and 5 the distribution of own patterns according to size depends on their number. The lesser the number of own patterns, the higher the proportion of small size own patterns is.

Maximum size of own patterns. We have shown that we were able to identify a user among 150 by extracting patterns of size one with $H_{relSupport}$ or H^c_{tf-idf}. However, an intuitive idea could be that with own patterns of different sizes (between 1 and 7 for example) heuristics could extract some very specific own patterns for each users and this way should be more efficient. It is partially true. Specifically, when the anonymous set of sessions is beyond 10 sessions H^c_{tf-idf} reaches an accuracy rate of 80% (see Fig. 6) with patterns of different sizes (see Table 4) and is better than $H_{relSupport}$ with own patterns of size one. However, with few sessions in the anonymous set, we can see in Fig. E.14 that increasing the maximum size slightly reduces the accuracy. The possibility to choose the size of our patterns can give some elements of semantic analysis, a feature that is desired in a fair number of applications in the industry, such as profiling for users or behaviours.

Size of a session. As mentioned in [26] and other studies, we obtained the best accuracy rate with sessions of 10 visited websites. There is no doubt that the size of a session needs to be calibrated to the potential size of the own patterns to be extracted. Indeed, the building-session process arbitrary cuts the log after each 10 unique websites and this way may lose and spread some valuable patterns across two different sessions.

Similarity and distance functions. Finally, we compared performance of different similarity functions and showed that for our study, *Cosine* and *Dice* similarities present the best accuracy rate for all heuristics (see Fig. 9). The *Euclidean* distance has average rate for Heuristic H_{supMin} and significantly worse accuracy rate for H^c_{tf-idf} (see Fig. 8).

In the near term, our intention is to supplement the model by studying some other own patterns selection criteria. Here, we can cite the *stability* measure introduced in [11] and evaluated in [2] and [15]. In particular, in a recent paper [16], Kuznetsov and Makhalova compare interestingness measures of closed patterns including stability, some probabilistic estimates of stability and robustness. It would be interesting to compare the accuracy rate obtained by some heuristics based on these measures with the accuracy rate obtained in the present study.

From a longer-term perspective, our goal is to extract some **ordered** patterns. Indeed, in the present study, a session has been interpreted as a set of visited sites and extracted own patterns are non **ordered** closed-patterns. It seems intuitive that the order relation of visited websites should provide more information to guess the user of a given anonymous set of sessions. An ordered session is called *a sequence* in the literature and several efficient algorithms have been designed specifically for their extraction (see Algorithms GSP in [22] and Spade in [27]). However, the number of sequences which can be extracted from a given context is very large and, as for the present study, some heuristics will have to be designed to select the more relevant patterns.

Acknowledgments

This research was partially supported by the European Union's "*Fonds Européen de Développement Régional (FEDER)*" program and the Auvergne-Rhône-Alpes region's call for projects S3-DIS4 dedicated to numerical traceability.

Bibliography

[1] Abramson, M. and Aha, D. W. (2013). User authentication from web browsing behavior. *Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference*, pages 268–273.

[2] Buzmakov, A., Kuznetsov, S. O., and Napoli, A. (2014). On evaluating interestingness measures for closed itemsets. In *7th European Starting AI Researcher Symposium (STAIRS 2014)*, volume 264, pages 71–80.

[3] Filson, D. (2004). The impact of e-commerce strategies on firm value: Lessons from Amazon.com and its early competitors. *The Journal of Business*, 77(S2):S135–S154.

[4] Finn, V. (1991). Plausible reasonning in systems of jsm-type. *Itogi Nauki Tekh., Inform*, 15:54 –101.

[5] Ganter, B. and Wille, R. (1999). *Formal concept analysis - mathematical foundations*. Springer.

[6] Garca-Borroto, M., Martnez-Trinidad, J., and Carrasco-Ochoa, J. (2014). A survey of emerging patterns for supervised classification. *Artificial Intelligence Review*, 42(4):705–721.

[7] Goel, S., Hofman, J. M., and Sirer, M. I. (2012). Who does what on the web: A large-scale study of browsing behavior. In *ICWSM*.

[8] Guvence-Rodoper, C. I., Benbasat, I., and Cenfetelli, R. T. (2008). Adoption of B2B Exchanges: Effects of IT-Mediated Website Services, Website Functionality, Benefits, and Costs. *ICIS 2008 Proceedings*.

[9] He, R., Yuan, M., Hu, J., Zhang, H., Kan, Z., and Ma, J. (2003). A novel service-oriented AAA architecture. 3:2833–2837.

[10] Herrmann, D., Banse, C., and Federrath, H. (2013). Behavior-based tracking: Exploiting characteristic patterns in DNS traffic. *Computer & Security*, pages 1–17.

[11] Klimushkin, M., Obiedkov, S. A., and Roth, C. (2010). Approaches to the selection of relevant concepts in the case of noisy data. In *Formal Concept Analysis, 8th International Conference, ICFCA 2010, Agadir, Morocco, March 15-18, 2010. Proceedings*, pages 255–266.

[12] Kumar, R. and Tomkins, A. (2010). A characterization of online browsing behavior. In *Proceedings of the 19th international conference on World wide web*, pages 561–570. ACM.

[13] Kuznetsov, S. O. (2004a). Complexity of learning in concept lattices from positive and negative examples. *Discrete Applied Mathematics*, 142(1-3):111–125.

[14] Kuznetsov, S. O. (2004b). Machine learning and formal concept analysis. In *P. Eklund (Ed.), proceedings Second International Conference on Formal Concept Analysis, Lecture note in Artificial Intelligence, 2961*, pages 287–312.

[15] Kuznetsov, S. O. (2007). On stability of a formal concept. *Ann. Math. Artif. Intell.*, 49(1-4):101–115.

[16] Kuznetsov, S. O. and Makhalova, T. P. (2018). On interestingness measures of formal concepts. *Inf. Sci.*, 442-443:202–219.

[17] Lagier, F. (2013). Cybercriminalité : 120.000 victimes d'usurpation d'identité chaque année en france. *Le populaire du centre (in French)*.

[18] Novak, P. K., Lavrač, N., and Webb, G. I. (2009). Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *The Journal of Machine Learning Research*, 10:377–403.

[19] Ramamohanarao, K. and Fan, H. (2007). Patterns based classifiers. *World Wide Web*, 10(1):71–83.

[20] Salton, G. (1989). *Automatic text processing: The transformation, analysis and retrieval of information by computer*. Addison Wesley.

[21] Shi, E., Niu, Y., Jakobsson, M., and Chow, R. (2010). Implicit authentication through learning user behavior. In *Information Security - 13th International Conference, ISC 2010, Boca Raton, FL, USA, October 25-28, 2010, Revised Selected Papers*, pages 99–113.

[22] Srikant, R. and Agrawal, R. (1996). *Mining sequential patterns: Generalizations and performance improvements*. Springer.

[23] Stockinger, T. (2011). Implicit authentication on mobile devices. In *The Media Informatics Advanced Seminar on Ubiquitous Computing*.

[24] Szathmary, L. (2006). *Symbolic Data Mining Methods with the Coron Platform*. PhD Thesis in Computer Science, University Henri Poincaré – Nancy 1, France.

[25] Ullah, I., Bonnet, G., Doyen, G., and Gati, D. (2011). Un classifieur du comportement des utilisateurs dans les applications pair--pair de streaming vido. *CFIP 2011 - Colloque Francophone sur l'Ingénierie des Protocoles (in French)*.

[26] Yang, Y. C. (2010). Web user behavioral profiling for user identification. *Decision Support Systems*, 49(3):261–271.

[27] Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2):31–60.

[28] Zaki, M. J. and Hsiao, C. (2002). Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on knowledge and data engineering*, 17(4):462–478.

Appendix A. Lattice of closure system

Let S_{Simon} be the set $\{bdeg, abcf, bdef, bcdf\}$ of simon's sessions. For our running example, the size of each session is 4 instead of 10 in our experimental protocol. We give in Fig. A.10 the lattice of the closure system $\mathcal{S}^c_{Simon}$.

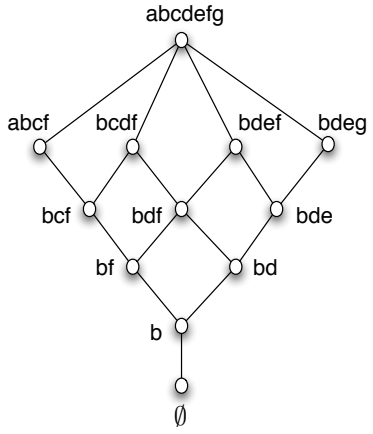

Figure A.10: The Hasse diagram of the closure system $\mathcal{S}^c_{Simon}$. The set bcf is a closed 3-pattern of Simon, and b is a closed 1-pattern.

Appendix B. Performance comparison between H_{sup} and $H_{relSupport}$

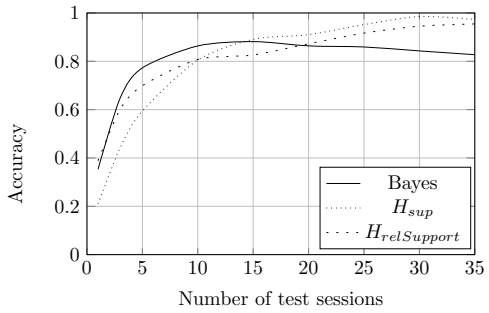

Figure B.11: Comparative performance of H_{sup} and $H_{relSupport}$. The number of sessions of the anonymous set is on the X-axis and accuracy rate on the Y-axis. Number of users is equal to 150. Measured values are smoothed on 10 executions.

Appendix C. Performance comparison H_{sup}, H^c_{sup} and H^c_{supMin}

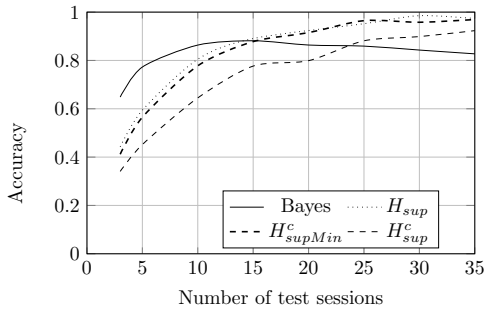

Figure C.12: Comparative performance of H_{sup}, H^c_{sup} and H^c_{supMin}. The number of sessions of the anonymous set is on the X-axis and accuracy rate on the Y-axis. Number of users is equal to 150. Measured values are smoothed on 10 executions.

Appendix D. Number of own patterns per user

The number of own patterns to be selected for describing each user is an important parameter of our study. As an example, we have shown with Table 4 and Table 5 that this parameter had an effect on the size of the selected patterns. In the following diagram given in Fig. D.13 we give the rate of anonymous set of sessions which they were able to be classify by the classifier. However, the problem especially occurs with anonymous sets of only one or two sessions. As an example, with one session and less than 20

own patterns, the model classifies less than 60% of the anonymous set of sessions. On the contrary, with more than 50 own patterns per user, the model classifies 100% of the anonymous sets of three sessions or more.

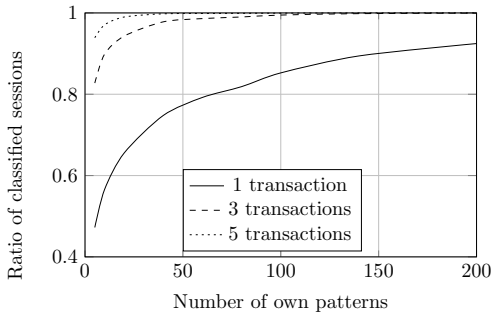

Figure D.13: The number of own patterns per user is on the X-axis and classified anonymous sets of sessions rate on the Y-axis. Number of users is equal to 150. Measured values are smoothed on 10 executions.

Appendix E. Maximal size of own patterns

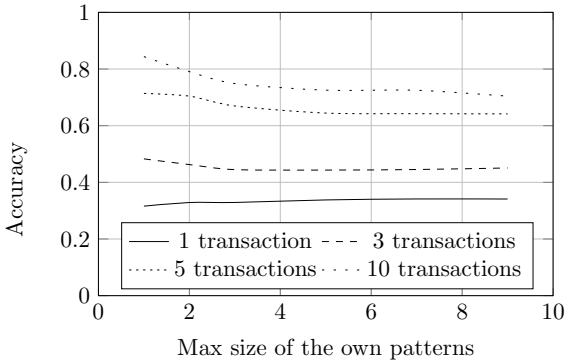

Figure E.14: The maximal size of the own patterns is on the X-axis and accuracy rate on the Y-axis. Number of users is equal to 150. Measured values are smoothed on 10 executions.

Appendix F. The size of sessions

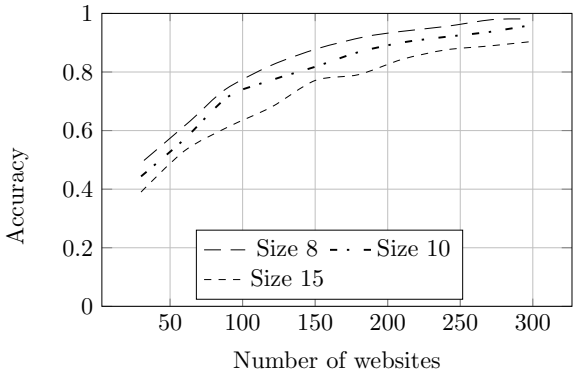

Figure F.15: Accuracy rate of the Heuristic H^c_{tf-idf} with 140 closed own patterns and their maximum size fixed to 7. For a given number of visited websites in the anonymous set to be classify, the accuracy rate increases slightly if the size of the sessions decreases.