# A Layered Approach to Revisitation Prediction

George Papadakis[1,2], Ricardo Kawase[2], Eelco Herder[2], and Claudia Niederée[2]

[1] ICCS, National Technical Unversity of Athens, Greece
gpapadis@mail.ntua.gr
[2] L3S Research Center, Leibniz University of Hanover, Germany
{surname}@l3s.de

**Abstract.** Web browser users return to Web pages for various reasons. Apart from pages visited due to backtracking, they typically have a number of favorite/important pages that they monitor or tasks that reoccur on an infrequent basis. In this paper, we introduce the architecture of a system that facilitates revisitations through the effective prediction of the next page request. It consists of three layers, each dealing with a specific aspect of revisitation patterns: the first one estimates the value of each page by balancing the recency and the frequency of its requests; the second one captures the contextual regularities in users' navigational activity in order to promote related pages, and the third one dynamically adapts the page associations of the second layer to the constant drift in the interests of users. For each layer, we introduce several methods, and evaluate them over a large, real-world dataset. The outcomes of our experimental evaluation suggest a significant improvement over other methods typically used in this context.

## 1 Introduction

Revisitation is the act of accessing again a previously visited Web page. As such, it constitutes a major part of the entire Web activity: Web users usually have to handle repetitive but infrequent tasks, revisiting pages after a considerable amount of time [4]. This was verified by most past works that explored users' surfing behaviors; Herder [10], for instance, quantifies it to 50% of the overall Web traffic, while Cockburn and McKenzie [4] approximate it to 80%. As a result, individuals have been found to waste 15% of their overall browsing time in their effort to find information they have accessed in the past [18]. They can benefit, therefore, to a large extent from browser-based methods that predict and facilitate their next revisitation request.

The most popular tools for client-side revisitation are bookmarks [4,10] and search engines [18,19]. The former, though, had their popularity significantly declined in favor of the latter, as they involve serious managing and organizational problems: the size of bookmark collections constantly increases with time, thus reducing their usability [4]. Search engines, on the other hand, are becoming the dominant tool for supporting revisitation, with about 40% of all queries pertaining to *re-finding*; that is, the process of using the same or a similar query to re-locate a previously visited Web page. However, the use of search engines is impractical, as it requires the memorization of a usually hard-to-remember combination of keywords [11]. There is, therefore, a great need for new methods that predict and facilitate users' revisitation activity.

In this paper, we introduce a system architecture that encompasses a set of methods aligned in three tiers. Each layer captures specific patterns in the navigational activity of a user, in order to effectively predict her next revisitation: the first one, the *ranking layer*, comprises functions that rank visited resources according to their likelihood of being (re-)accessed in the immediate future. The second one, called *propagation layer*, enhances the ranking methods with techniques that encapsulate contextual patterns in the behavior of a user; that is, it identifies groups of pages visited together during the same session - in the same or different order - and boosts their ranking values accordingly. Finally, the third tier, the *drift layer*, conveys methods that adapt the patterns captured by the propagation layer to the changing nature of the interests of the user. On the whole, our framework constitutes a comprehensive method for revisitation prediction, that covers all its aspects (i.e.., frequency and recency of page requests, contextual patterns and concept drift), while being easy to implement and integrate into a user interface.

Special care has been taken to make our framework extensible, so that adding new methods or improving existing ones, in any of its three tiers, is a straightforward procedure. This is indeed ensured by the transparency of the strictly defined interfaces described in Section 3. We have also made public both the implementation and the data used in this paper under the *SUPRA*[1] project of SourceForge[2]. Thus, we provide a common benchmark for new algorithms in this area, and encourage other researchers to experiment with our library and extend it with improved or novel techniques.

To summarize, the main contributions of this paper are the following:

- We introduce a layered architecture for a system that effectively addresses the next revisitation prediction problem. It consists of three tiers, each tackling a specific aspect of the problem.
- We coin several methods for each layer, based on the navigational and the time patterns of individual user's activity.
- We evaluate the methods of our library through a thorough experimental study that involves a voluminous, real-world dataset. The results verify its superiority over well-established methods for this problem.

The rest of this paper is organized as follows: in Section 2 we discuss related work, while in Section 3 we formally define the problem we are tackling and elaborate on the architecture of our system. Section 4 analyses our thorough evaluation study, and Section 5 wraps up our work with final remarks and future plans.

[1] SUPRA stands for "SUrfing Prediction fRAmework".

[2] See `http://sourceforge.net/projects/supraproject`.

## 2 Related Work

A problem more general than the *next revisitation prediction* has has been extensively studied in the literature: the *next page prediction* problem. The method that has prevailed in this field, at least in terms of popularity, is Association Rules Mining. In more detail, *association rules* (**AR**) effectively identify related resources without taking into account their order of appearance (e.g., pages that are typically visited together, in the same session, but not necessarily in the same order) [1,2]. This feature turns them ideal for recommending resources related to a particular site. Numerous works have investigated the functionality of different variations of AR [7,12,16]. For example, a recent work by Kazienko [12] explores indirect AR for Web recommendations, involving resources that are not “hardly” connected as in typical AR.

However, AR suffer from a variety of drawbacks: first, they rely on the most frequent patterns identified in the training set, thus misclassifying new patterns that are not included in it (e.g., patterns stemming from concept drift). Second, they fail to recommend rarely visited, and, thus, non-obvious and serendipitous items, since such resources never reach the minimum support limit[3]. Third, they disregard the order of itemsets, and cannot distinguish between different patterns that involve the same resources (i.e., an itemset $I_1 = \{1, 2, 3\}$ is treated equally with all its 6 permutations).

To overcome this last problem, *sequential patterns* have also been employed in the context of prediction methods. Among them, state-based methods, like Markov models, are particularly popular [20,6,3]. Sequence mining techniques constitute a variation of this approach, in the sense that they do not consider the strict order between items [2,15]. A comparison of these techniques with AR was conducted by Géry and Haddad [8], with the outcomes of their evaluation suggesting that Frequent Sequence Mining has the best performance. Nevertheless, all these methods still suffer from the inability to predict/recommend unseen items (i.e., not included in the training set, typically due to concept drift).

With the aim of introducing a prediction method that is equally effective with unseen data, Awad et al. [3] combined the Markov model with Support Vector Machines (SVM) under Dempster’s rule. Their experimental evaluation verified the superiority of their hybrid model over AR, especially when domain knowledge is incorporated into it. However, their method is quite impractical: it requires a different SVM classifier for each one of the available resources and, thus, entails an excessively high training time.

In the following sections, we propose a layered system architecture for revisitation prediction that overcomes the shortcomings of existing works, while taking their advantages into account. To this end, the first layer incorporates techniques that estimate the value of visited pages from the frequency and the recency of their requests. The second layer, on the other hand, captures the

[3] The problem of identifying rare but important associations has been tackled through the multiple minimum support method. This technique, however, has not yet been applied in the context of the next page prediction problem.

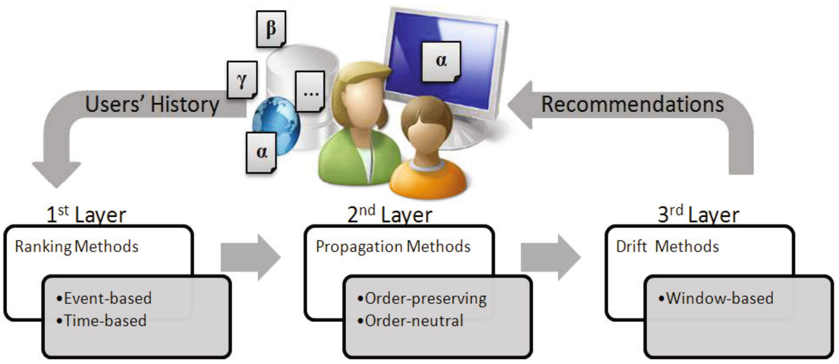

**Fig. 1.** The layered architecture of our revisitation prediction system

connections between pages visited during the same session, either by considering or by ignoring their order of access. Its novelty lies in its ability to identify new associations on-the-fly and to incorporate them dynamically into its data structure. To discard the connections that are outdated due to the drift in the interests of a user, we introduce another layer encompassing a window-based drift method; it re-adjusts the associations between pages after a certain period of time, so that they reflect the latest patterns in the user's activity.

## 3 Approach

The problem we are tackling in this paper consists of the task of identifying which Web page, among those visited by a specific user in the past, will be revisited in her next page request. More formally, we define it as follows:

**Problem Statement.** *Given the collection of Web pages, $P_u = \{p_1, p_2, ...\}$, that have been visited by a user, $u$, during her past $n$ page requests, $R_u = \{r_1, r_2, \ldots, r_n\}$, order them accordingly, so that the ranking of the page $p_i$ that she will revisit in her next request, $r_{n+1}$, is the highest possible.*

The above definition stresses that the goal is to facilitate the access to pages that have already been visited in the past, rather than trying to recommend not-visited but relevant ones. To serve this goal, we present a collection of methods that produce a ranking of all visited Web pages; the more likely a Web page is to be accessed in the next request, the higher its ranking. The ranked list of pages is updated after each page visit, and the higher the ranking of the subsequently accessed page, the better the prediction. This is in line with the intuition behind the ranking of search engines' results to keyword queries: users typically consult only the top 10 results, and the higher the ranking of the desired resource, the better the performance of the search engine [9].

Figure 1 depicts the architecture of our system, that encompasses three tiers of methods. The first one entails *ranking methods*, which estimate for each Web page the likelihood that it will be accessed in the next request. Their estimation is derived from patterns in the surfing history of the underlying user, namely the recency and the frequency of accesses to each page. The second layer covers *propagation methods*; these are techniques that capture repetitiveness in the navigational activity of the underlying user and identify contextual associations between pages that are typically visited together (i.e., in the same session, but not necessarily in the same order). Depending on the degree of connectivity between the associated Web pages, their values (assigned by the ranking methods) are then propagated to each other. The third layer contains window-based *drift methods*, which adapt the associations encapsulated by the propagation methods to the volatile interests of the user. They employ a sliding time frame (e.g., of a day or a week) that periodically discards the connections that took place out of its borders. On the whole, these three layers provide a comprehensive framework that tackles all aspects of the revisitation activity.

In the following, we present and analyze several techniques for each layer. Their implementation is already freely available through the *SUPRA* project of SourceForge. In this way, we encourage other researchers to experiment with them and to extend our library with new methods for every layer. Special care has been taken to make this a straightforward procedure, by providing clear guidelines through the formalization of the methods that are presented in the following sections. Any implementation complying with the minimal requirements for a ranking, a propagation and a drift method, as described in Definitions 1 to 8, can be easily integrated in our library. It is also worth noting that the real-world data employed in our experiments have also been publicly released through the *Web History Repository* project[4], so that they can be used as a general benchmark for prediction algorithms, independently of our framework.

## 3.1 Ranking Methods

As mentioned above, the aim of a ranking method is to provide for each Web page a numerical estimation of the likelihood that it will be accessed in the next request. All pages are then sorted in descending order of their value, with the aim of placing the next revisited page to the highest possible ranking. After each page visit, the value of all pages changes, and the ranked list is updated. The reason is that the numerical estimation of each page is derived by contrasting the latest page visit with all (or part) of the past requests to that particular page; depending on the way the page's access history is handled, we distinguish two kinds of ranking functions: the event- and the time-based ones.

**Time-based Ranking Methods.** This family of ranking functions relies on the time the requests to a page occurred, in order to estimate its value. That is, the contribution of each request to the total value of the corresponding page depends on the actual time the respective page visits took place and the time that has elapsed ever since. Thus, the input of these methods principally comprises the request timestamps of each page:

[4] See `http://webhistoryproject.blogspot.com`

**Definition 1.** *Given the page requests $R_u$ of a user $u$, the* ***request timestamps of a page*** *$p_i$, $T_{p_i}$, is the set of timestamps of those requests in $R_u$ that pertain to $p_i$.*

A time-based ranking method can be now defined as follows:

**Definition 2.** *A* ***time-based ranking method*** *is a function that takes as input the request timestamps $T_{p_i} = \{t_1, t_2, \dots, t_k\}$ of a page $p_i$ together with the time of the latest request, $t_n$, of the given user $u$, and produces as output a value for $p_i$, $v_{p_i} \in [0,1]$, that is proportional to the likelihood that it will be accessed at the next page request, $r_{n+1}$ (i.e., the closer $v_{p_i}$ is to 1, the higher this likelihood).*

In our system, we selected *Frecency* (**FR**) as representative of this kind of ranking methods. The reason is that it is integrated in one of the most popular Web browsers, namely Mozilla Firefox[5]. In essence, it places more emphasis on the frequency of the use of a Web page, and discounts only to some extent the influence of the very old visits. In more detail, the total ranking value of each page is equal to the sum of the values assigned to each of its requests; the value of a single page visit is called bonus and its size is proportional to its recency: requests occurring within the last four days take the highest bonus, whereas requests that are older than 90 days take the lowest one. In addition, FR considers the type of access, i.e., whether the URL of the page was typed, clicked upon or selected from the bookmarks collection. This is, however, out of the scope of our definition[6]. Note that to restrict the ranking values of Frecency in the interval $[0, 1]$, our implementation normalizes the value of each page with the globally largest page value.

**Event-based Ranking Methods.** In contrast with the previous category, event-based ranking methods interpret page visits as a sequence of events, and exclusively take into account their relative position. That is, they disregard the actual time of each request, and consider only the number of events that have elapsed since it occurred, in order to estimate its contribution to the total value of the corresponding page. Thus, this family of methods represents the access history of a page by the indices of the related requests:

**Definition 3.** *Given the page requests $R_u$ of a user $u$, the* ***request indices of a page*** *$p_i$, $I_{p_i}$, is the set of the serial numbers of those requests in $R_u$ that pertain to $p_i$. The serial number of the chronologically first request is 1 and is incremented by 1 for each subsequent page visit.*

Given this definition, an event-based ranking method is defined as follows:

**Definition 4.** *An* ***event-based ranking method*** *is a function that takes as input the request indices $I_{p_i} = \{i_1, i_2, \dots, i_k\}$ of a page $p_i$ together with the*

[5] See `http://www.mozilla.com/en-US/firefox`

[6] See `https://developer.mozilla.org/en/The_Places_frecency_algorithm` for more details.

*index of the latest request, $i_n$, of a user $u$, and produces as output the value of $p_i$, $v_{p_i} \in [0, 1]$, that is proportional to the likelihood that $p_i$ will be accessed at the next page request, $r_{n+1}$ (i.e., the closer $v_{p_i}$ is to $1$, the higher this likelihood).*

As an illustration of the this kind of methods, we consider the *decay ranking model* that was introduced by Papadakis et al. in [14]. According to this model, the value of a Web page $p_i$ after $i_n$ visits is derived from the following formula:

$$DEC(p_i, I_{p_i}, i_n) = \sum_{j=1}^{|I_{p_i}|} d(i_j, i_n),$$

where $d(i_j, i_n)$ is a *decay function* that takes as an input the index $i_j$ of a request to $p_i$ together with the index of the current page vist, $i_n$, and gives as output the contribution of this request to the total value of $p_i$.

According to Cormode et al. [5], every *valid decay function* should satisfy the following properties:

1. $d(i_j, i_n) = 1$ when $i_j = i_n$
2. $0 \leq d(i_j, i_n) \leq 1 \;\; \forall i_j \in [0, i_n]$
3. $d$ is monotone non-increasing as $n$ increases:

$$i'_n \geq i_n \rightarrow d(i_j, i'_n) \leq d(i_j, i_n) \;\; \forall i_j \in [0, i_n].$$

Among the valid decay function families, the *Polynomial Decay* (**PD**) functions were found by Papadakis et al. [14] to outperform both the *exponential* and the *logarithmic ones*. The reason is that their smooth decay balances harmonically the recency and the frequency of page revisits; in contrast, exponential functions convey a steep decay that puts more emphasis on recency, whereas the logarithmic functions promote excessively frequency, due to their overly slow decay. The actual value of a polynomial decay function with exponent $\alpha$ for a page $p_i$ at the $i_j - th$ request out of $i_n$, in total, accesses is given from the following formula:

$$d(i_j, i_n) = \frac{1}{1 + (i_n - i_j)^{\alpha}}. \tag{1}$$

The main difference between Polynomial Decay and Frecency, apart from the evidence they take into account, is the balance they achieve between frequency and recency. Frecency favors the former over the latter, thus constituting a mere improved version of the Most Frequently Used caching algorithm. On the other hand, Polynomial Decay achieves a better balance between these two metrics, while being more flexible, as well. In fact, it can be adapted to the behavioral patterns of the underlying user, employing the value of $\alpha$ in Formula 1 as a fine-tuning parameter; the larger its value ($1 << \alpha$), the higher the effect of recency on the overall value of a page (due to the steeper the decay of the contribution of a page request), and vice versa. Thus, Polynomial Decay can adjust its performance to different kinds of users.

## 3.2 Propagation Methods

Unlike ranking methods that produce an ordering of Web pages, propagation methods aim at capturing contextual information through the detection of patterns in the surfing activity of users. They identify those pages that are commonly visited within the same session and associate them with each other. The "links" created by these methods are then combined with a ranking method, so that the value of a Web page is propagated to its relevant ones. In this way, the higher the value of a Web page, the more the pages associated with it are boosted and the more their ranking is upgraded.

At the core of the associations between resources lies the notion of the session, which can be formally defined as follows:

**Definition 5.** *A **session** $S$ is the bag of all pages $p_i$ visited by a user $u$ in the same browser tab for a time period of up to 25.5 minutes ([8,17]), placed in chronological order, from the earliest to the latest one: $S = \{p_1, p_2, \ldots, p_k\}$.*

Based on Definition 5, propagation methods can be defined as follows:

**Definition 6.** *A **propagation method** is a function that takes as input the last requested page, $p_i$, within a session, $S$, and defines appropriately the degree of connection between $p_i$ and all the other pages visited during $S$. Hence, given two pages, $X$ and $Y$, it returns a value, $v_{XY} \in [0, 1]$, that is proportional to the likelihood of $Y$ being accessed immediately after $X$ (i.e., the closer $v_{XY}$ is to 1, the more likely this transition is).*

In this work, we distinguish two families of propagation methods: the *order-preserving* ones, which take into account the order of the page requests within a session, and the *order-neutral* ones that disregard this order. For the former case, we consider transition matrices, whereas for the latter we examine association matrices.

**Order-Preserving Propagation Methods.** This category of propagation methods relies on the idea that Web pages are typically accessed in the same or similar order. Hence, given a session that contains a series of page requests ordered by time, they build the associations between pages according to this ordering: each page is connected only with the pages that precede it. To capture these transitions that form chronological patterns in the navigational activity of users, we introduce the transition matrix (**TM**).

In more detail, a TM is a two dimensional structure with its rows and columns representing the Web pages $P$ visited so far by the given user $u$ (Problem Statement); each cell $TM(x, y)$ expresses the number of times that a user visited page $y$ *after* $x$. Given that a transition matrix respects the order of accesses within a session, it is not a symmetrical one: the value of $TM(x, y)$ is not necessarily equal to that of $TM(y, x)$. Moreover, its diagonal cells are all equal to 0: $\forall x \;\; TM(x, x) = 0$. This is because there is no point in associating a page with itself; in case a requested Web page is revisited in the subsequent request,

$$
\underset{(a)}{\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 0 & 0 \\ B & 0 & 0 & 1 & 0 \\ C & 0 & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 \end{array}}
\quad
\underset{(b)}{\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 1 & 1 \\ C & 1 & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 \end{array}}
\quad
\underset{(c)}{\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & \frac{1}{2} & \frac{1}{4} \\ B & \frac{1}{4} & 0 & 1 & \frac{1}{2} \\ C & \frac{1}{2} & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 \end{array}}
\quad
\underset{(d)}{\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 2 & 4 \\ B & 4 & 0 & 1 & 2 \\ C & 2 & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 \end{array}}
\quad
\underset{(e)}{\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 1 & 1 \\ C & 1 & 1 & 0 & 1 \\ D & 1 & 1 & 1 & 0 \end{array}}
$$

**Fig. 2.** The values of several types of the Transition and the Association Matrices after the last page request of the session: $S_1 : A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$. a) corresponds to SM, b) to CM, c) to DM, d) to IM, and e) to AM.

its ranking will be high enough due to the value assigned to it by the ranking method and does not need to be boosted by the propagation method.

In the following, we introduce 4 different techniques for correlating Web pages according to the past navigational activity in the context of a transition matrix. They are intuitively illustrated through a simple walkthrough example. Given a set of 4 Web pages - $A, B, C, D$ - and the following set of requests during a given session, $S_1 : A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$, we can associate these pages in four different ways (taking into account the order of the accesses):

1. **Simple Connectivity Transition Matrix (SM)**. For each transition $x \rightarrow y$ in the given session, only the value of the cell $TM(x, y)$ is incremented by one. The frequencies defined by this rule work exactly as a first-order Markov model. The rationale behind this approach is, thus, the expectation that requests tend to occur in the same strict order. Figure 2(a) depicts the values of the transition matrix according to this rule after the last transition of the given session, $D \rightarrow A$.
2. **Continuous Connectivity Transition Matrix (CM)**. Each Web page visited within the current session is associated with all the previously accessed pages. In this way, it can effectively support requests that take place in a similar order, i.e., in the same direction, but not necessarily in the same sequence (e.g., $X \rightarrow Z \rightarrow Y$ and $X \rightarrow Y$). In our example, $A$ is associated with all other Web pages after transition $D \rightarrow A$, incrementing the corresponding cells by one (Figure 2(b)).
3. **Decreasing Continuous Connectivity Transition Matrix (DM)**. This strategy operates in a similar way as the previous one with the difference that it increments the cells of TM by a decay parameter representing the distance (i.e., number of transitions) that intervenes between the corresponding Web pages. Therefore, this form of transition matrix lies in the middle of SM and DM, supporting evenly requests that occur either in the same or in similar order. In our example, $TM(C, A)$ is incremented by 1/2 after $D \rightarrow A$, since page $C$ is two steps away from the page $A$. Figure 2(c) depicts the whole DM after the transition $D \rightarrow A$.
4. **Increasing Continuous Connectivity Transition Matrix (IM)**. This is the inverted version of the previous strategy. Instead of decreasing the value added to $TM(x, y)$ according to the distance of pages $x$ and $y$, it increases it

proportionally. Hence, it results in stronger connections between pages that are more distant, in an effort to identify the final destination of the given session. By boosting its value early enough, it can significantly restrict the number of irrelevant pages that the user visits before reaching its actual page of interest. The matrix produced by this rule after the last transition of our example is presented in Figure 2(d).

It is worth noting that SM is also used in Awad et al. [3], but its frequencies are merely used as features to a classification algorithm. In addition, CM is also employed in Parameswaran et al. [15] as the means of providing the frequencies of the probabilistic analysis that precedence mining involves.

**Order-Neutral Propagation Methods.** In contrast to the order preserving methods, the order-neutral ones are based on the idea that the temporal order of page visits within a session is not important; pages that are visited in the course of the same session should be equally connected with each other, regardless of their order and the number of transitions that intervene between them. The rationale behind this idea is that users may visit a group of pages $X, Y, Z$ on a regular basis, but not necessarily in that order.

To model this idea, we introduce the association matrix (**AM**); similar to TM, AM is a matrix whose rows and columns are the Web pages $P$ visited so far by the given user. The difference is that AM is built simply by associating all pages visited in a single session with each other; i.e., each Web page is connected not only with the pages preceding it, but also with those following it. Thus, an AM is always a symmetrical matrix with all its diagonal cells equal to 0 ($\forall x \;\; AM(x,x) = 0$). Given the session $S_1$ of the above example, the resulting AM has all non-diagonal cells equal to one, as all resources were accessed during this session (Figure 2(e)).

**Combining Ranking with Propagation Methods.** To combine a ranking method with one of the propagation techniques, we employ a simple, linear scheme: following the $i_n$-th page request, the value of all pages is (re)computed, according to the selected ranking method. Then, for each non-zero cell of the matrix at hand ($TM(x,y)$ or $AM(x,y)$), we increment the value assigned to page $y$ by the ranking method, $v_y$, as follows:

$$v_y += p(x \rightarrow y) \cdot v_x, \text{ where}$$

- $p(x \rightarrow y)$ is the transition probability from page $x$ to page $y$, estimated by $p(x \rightarrow y) = \frac{TM(x,y)}{\sum_i^{i_n} TM(x,i)}$ (or $p(x \rightarrow y) = \frac{AM(x,y)}{\sum_i^{i_n} AM(x,i)}$), and
- $v_x$ is the value of $x$ estimated by the ranking method.

## 3.3 Drift Methods

Unlike the ranking methods, the propagation ones encompass no inherent support for drift in the focus of user's interests: the connections stored in their data structure (i.e., matrix) remain static, and, thus, cannot adapt to the constantly changing habits and interests of users. In the literature, two main approaches that support *concept drift* have been proposed: first, the decay functions, like the polynomial one, and, second, the window-based methods [13]. The latter take their name from the sliding window they employ in order to keep the most recent evidence and ignore the rest.

To enable the dynamic nature of the propagation methods, we introduce in our system a third layer that consists of a *window-based drift method*, operating on the data structure of the second layer. Depending on the way the size of the window is specified, we distinguish between *event-based* and *time-based* drift methods; the former define the window with respect to the size of a batch of requests, whereas the latter with respect to a period of time. More formally, the time-based drift methods are defined as follows:

**Definition 7.** *Given the page requests $R_u$ of a user $u$, the matrix $m$ of a propagation method and a time period $t$, a* ***time-based drift method*** *updates the connections stored in $m$ so that they reflect the page requests of $R_u$ that occurred in the latest $t$ temporal units (e.g., days or weeks).*

Similarly, the event-based drift methods are defined as follows:

**Definition 8.** *Given the page requests $R_u$ of a user $u$, the matrix $m$ of a propagation method and a number of requests $n$, an* ***event-based drift method*** *updates the connections stored in $m$ so that they reflect the page requests of $R_u$ that occurred in the latest $n$ page requests.*

Due to the temporal, periodic patterns we identified in the large, real-world data set we have at our disposal, we considered only time-based drift methods. Thus, in our experimental study we examine the **Day-**, the **Week-** and the **Month-model**. As their name suggests, they update the matrix of the underlying propagation method so that it maintains the associations of the last day, the last week and the last month, respectively.

## 4 Evaluation

**Data Set.** To thoroughly evaluate our approach, we employed a real-world, voluminous data set that was gathered through the Web History Repository project. It comprises the navigational activity of 200 users, logged in the time period between 30/09/2010 and 11/01/2011. In total, more than 580,000 page requests were recorded. They are not, though, evenly distributed over the participants; characteristically, there are 100 users with less than 1,000 requests (with a minimum of 200), and 18 users with more than 10,000 requests (with a maximum of 16,570). The distribution of the logging period per user varies greatly, as well, ranging from 1 to 278 days. On average, though, each user issued almost 3,000 page requests, in a time period of 38 days. One third of them constituted a revisit, thus producing a revisitation rate that is a bit lower than the estimation

**Table 1.** Technical characteristics of the WHR data set we employed

| | | | |
|---|---|---|---|
| Users | 200 | Av. Requests per User | 2,914 |
| Page Requests | 582,853 | Av. Web Pages per User | 1,905 |
| Web Pages | 381,066 | Av. Revisits per User | 1,009 |
| Sessions | 91,300 | Av. Sessions per User | 457 |
| Revisits | 201,787 | Av. Requests per Session | 126.40 |
| Revisitation Rate | 34.62% | Av. Days per User | 37.72 |

of Herder et al. [10] in 2005. Regarding the demographics (e.g., age and sex) of the participants, we do not have any relevant data at our disposal, since the volunteers contributed their navigational history anonymously. Note also that the sessions in the data set are set transparently by the browser, not necessarily according to the time criterion of Definition 5. The technical characteristics of our data set are summarized in Table 1.

**Setup.** In the course of our experimental evaluation, we simulated the navigational activity of each user independently of that of the others: her page requests were sorted in ascending order of time, and the simulation proceeded by one request at a time, starting from the earliest and moving to the latest one. A ranked list of the so-far-visited pages was maintained per user, and, after each page request, the ranking of all pages was updated, according to our prediction methods; if the next page request was not a revisitation, the new page was added to the list. Otherwise, the position of the corresponding Web resource was recorded. Based on the ranking positions we collected, we evaluate the performance of the prediction algorithms in terms of the following metrics:

(i) **Success Rate at 1 (S@1)** denotes the portion of revisitations that pertained to the top ranked Web page. The higher this percentage, the better the performance of the prediction method.

(ii) **Success Rate at 10 (S@10)** stands for the percentage of revisitation requests pertaining to a page that is ranked in some of the top 10 positions. Similar to S@1, the higher its value, the better the performance of the method. This metric expresses the actual usability of a prediction algorithm, as users typically have a look only at the first 10 pages presented to them, just like they do with Web search engine results [9].

(iii) **Average Ranking Position (ARP)** represents the place a revisited page is found on average in the ranking list of the method at hand. Thus, it provides an estimation of the overall performance of a prediction algorithm, considering the ranking position of all revisitations, and not just the top ranked ones. The lower its value, the better the performance of the prediction method.

On the whole, the combination of these three metrics provides a comprehensive estimation of the effectiveness of a prediction algorithm: it considers both its practical recommendations (S@1, S@10) and its performance over all revisitations (ARP).

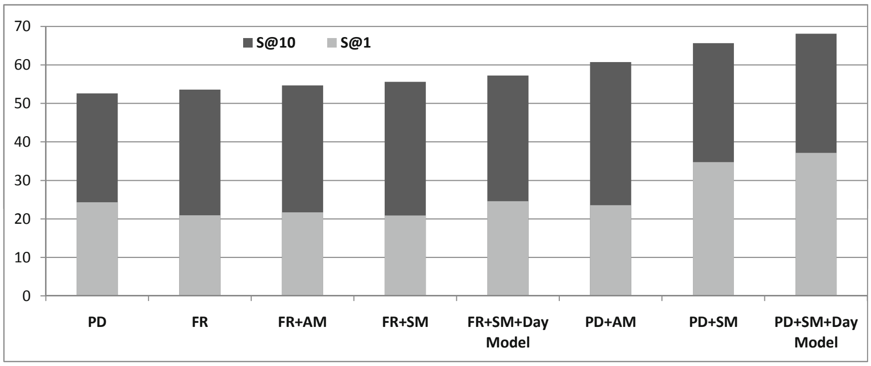

**Fig. 3.** Performance in percentage (%) with respect to S@1 and S@10 for the selected prediction methods. The methods are placed in ascending order of S@10 from left to right, with the right-most one achieving the optimal performance.

**Results Analysis.** As baseline approaches for the evaluation of our system, we consider the individual ranking methods of Section 3, i.e., FR and PD; they have already been proposed in the literature, and the former actually constitutes the state-of-the-art method, as it is integrated in a popular browser (Mozilla Firefox), and is widely used. The goal is, thus, to verify that combining existing ranking methods with the propagation and the drift ones enhances their performance to a significant extent and results in more accurate predictions.

In total, our framework accommodates (2 ranking methods $\times$ 5 propagation methods $\times$ 3 drift methods =)30 combinations of prediction methods. Due to lack of space and for the sake of readability, we will consider only 6 of them, in addition to the baseline ones: the best of combination of each ranking method with an order-neutral and an order-preserving propagation method, as well as the best combination of ranking and propagation methods with the day-model drift method. The criterion for choosing the best combination of ranking and propagation methods was their performance for S@10, the most indicative metric for the usability of a revisitation prediction method. In this aspect, both ranking functions maximized their performance when combined with SM. On the other hand, the selection of the drift method was determined by the characteristics of our data set, which does not cover the long-term navigational activity of the participants (apart from a couple of users); rather, it contains their short- or their mid-term activity (i.e., few days and weeks, respectively). Nevertheless, our evaluation enables us to examine the contribution of each layer to the overall performance, independently of that of the others[7].

The results of the evaluation for the first two metrics are presented in Figure 3, while Figure 4 depicts the performance of the selected methods for ARP. In

[7] Note that there is not point of examining the performance of ranking methods in combination with the drift ones alone, since the latter apply only to the data structure of the propagation methods.

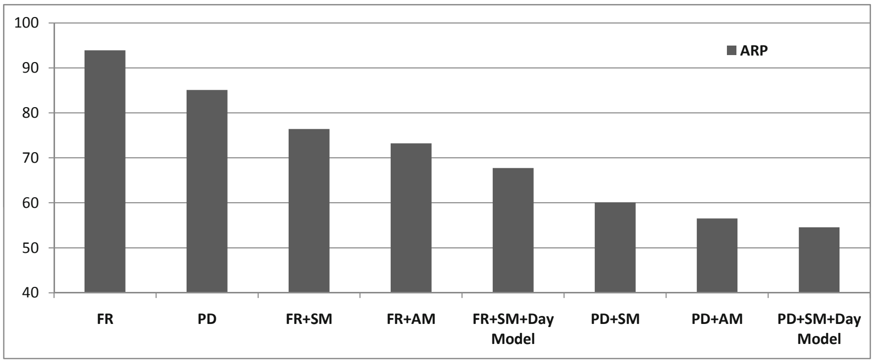

**Fig. 4.** Performance with respect to ARP for the selected prediction methods. The methods are placed in ascending order of performance from left to right, with the right-most one achieving the optimal performance.

both cases, the methods are ordered from left to right in ascending order of performance, so that the right-most method exhibits the best performance.

In the case of S@1 and S@10, we can easily notice that the performance of ranking methods is substantially improved by both the other levels of the system. Actually, the best performances are achieved when all three layers are employed in conjunction. This is particularly true for PD and its combinations with AM, SM and the Day Model, with the complete method (PD+SM+Day Model) improving PD by 52.84% and 29.41% with respect to the S@1 and S@10, respectively. FR, on the other hand, is improved to a lower extent by the three-layered model by 6.8% and 17.41%, respectively. All improvements of the second layer over the first and the third over the two other layers, were found to be statistically significant ($p < 0.05$), with the exception of FR+AM over FR.

Regarding ARP, we notice the same patterns: the more layers are employed, the better the overall performance of our system. It is worth noting that the degree of improvement for FR is much higher in this case: it ranges from 18.66% for FR+SM to 27.87% for FR+SM+Day Model. For PD, the improvements fluctuate between 29.38% for PD+SM and 35.88% PD+SM+Day Model. Again, all improvements of one layer over the underlying ones were found to be statistically significant ($p < 0.05$).

It is worth noting that the order-preserving propagation methods outperform the order-neutral ones for S@1 and S@10, while having a lower performance with respect to ARP. The reason is that, unlike SM, AM does not identify the most relevant page(s) to the currently accessed one; rather, it uniformly associates each page with all other pages visited during the same sessions. Thus, AM evenly distributes the value of the most recently visited page among all relevant pages, leading to higher ARP, whereas SM merely boosts the value and the ranking of the most likely next pages. The success rate gets, therefore, substantially higher, but ARP is not improved at a lower rate.

## 5 Conclusions

In this paper, we presented a layered architecture for a system that facilitates the revisitation activity of users, through accurate predictions. It consists of three tiers, each addressing a particular aspect of this phenomenon: the recency and frequency of patterns of revisitations, their contextual patterns as well as the ever-changing interests of a user. Our thorough experimental evaluation verified that each layer conveys significant improvements over the prevalent method for this task (i.e., Mozilla Firefox's Frecency). On the whole, our system predicts the next revisited page in 37% of the cases, while its top-10 recommendations contain the desired page in 68% of the cases. Given that different users receive the optimal recommendations for different methods, in the future we plan to investigate ways of inferring a priori the optimal combination of methods for each user. In addition, we intend to examine whether our system is applicable in the context of server-side recommendations, as well (e.g., in the case of the intranet of a large company).

## Acknowledgments

This research has been co-funded by the European Commission within the eContentplus targeted project OpenScout, grant ECP 2008 EDU 428016.

## References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: SIGMOD Conference, pp. 207–216 (1993)
2. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE, pp. 3–14 (1995)
3. Awad, M., Khan, L., Thuraisingham, B.M.: Predicting www surfing using multiple evidence combination. VLDB J. 17(3), 401–417 (2008)
4. Cockburn, A., McKenzie, B.J.: What do web users do? an empirical analysis of web use. Int. J. Hum.-Comput. Stud. 54(6), 903–922 (2001)
5. Cormode, G., Shkapenyuk, V., Srivastava, D., Xu, B.: Forward decay: A practical time decay model for streaming systems. In: ICDE, pp. 138–149 (2009)
6. Deshpande, M., Karypis, G.: Selective markov models for predicting web page accesses. ACM Trans. Internet Techn. 4(2), 163–184 (2004)
7. Fu, X., Budzik, J., Hammond, K.J.: Mining navigation history for recommendation. In: IUI, pp. 106–112 (2000)
8. Géry, M., Haddad, M.H.: Evaluation of web usage mining approaches for user's next request prediction. In: WIDM, pp. 74–81 (2003)
9. Hawking, D., Craswell, N., Bailey, P., Griffiths, K.: Measuring search engine quality. Inf. Retr. 4(1), 33–59 (2001)
10. Herder, E.: Characterizations of user web revisit behavior. In: LWA, pp. 32–37 (2005)
11. Kawase, R., Papadakis, G., Herder, E., Nejdl, W.: The impact of bookmarks and annotations on refinding information. In: HT, pp. 29–34 (2010)
12. Kazienko, P.: Mining indirect association rules for web recommendation. Applied Mathematics and Computer Science 19(1), 165–186 (2009)

13. Koychev, I., Schwab, I.: Adaptation to drifting user's interests. In: ECML Workshop: Machine Learning in New Information Age, Citeseer, pp. 39–46 (2000)
14. Papadakis, G., Niederee, C., Nejdl, W.: Decay-based ranking for social application content. In: WEBIST, pp. 276–282 (2010)
15. Parameswaran, A.G., Koutrika, G., Bercovitz, B., Garcia-Molina, H.: Recsplorer: recommendation algorithms based on precedence mining. In: SIGMOD, pp. 87–98 (2010)
16. Sandvig, J.J., Mobasher, B., Burke, R.: Robustness of collaborative recommendation based on association rule mining. In: RecSys, pp. 105–112 (2007)
17. Tauscher, L., Greenberg, S.: How people revisit web pages: empirical findings and implications for the design of history systems. Int. J. Hum.-Comput. Stud. 47(1), 97–137 (1997)
18. Teevan, J., Adar, E., Jones, R., Potts, M.A.S.: Information re-retrieval: repeat queries in yahoo's logs. In: SIGIR, pp. 151–158 (2007)
19. Tyler, S.K., Teevan, J.: Large scale query log analysis of re-finding. In: WSDM, pp. 191–200 (2010)
20. Yao, Y., Shi, L., Wang, Z.: A markov prediction model based on page hierarchical clustering. Int. J. Distrib. Sen. Netw. 5(1), 89–89 (2009)