# Client- and Server-side Revisitation Prediction with SUPRA

George Papadakis[$,◇], Ricardo Kawase[◇], Eelco Herder[◇]

[$] National Technical University of Athens, Greece gpapadis@mail.ntua.gr
[◇] L3S Research Center, Hanover, Germany {papadakis, kawase, herder}@l3s.de

## ABSTRACT

Users of collaborative applications as well as individual users in their private environment return to previously visited Web pages for various reasons; apart from pages visited due to backtracking, they typically have a number of favorite or important pages that they monitor or tasks that reoccur on an infrequent basis. In this paper, we introduce a library of methods that facilitate revisitation through the effective prediction of the next page request. It is based on a generic framework that inherently incorporates contextual information, handling uniformly both server- and the client-side applications. Unlike other existing approaches, the methods it encompasses are real-time, since they do not rely on training data or machine learning algorithms. We evaluate them over two large, real-world datasets, with the outcomes suggesting a significant improvement over methods typically used in this context. We have also made our implementation and data publicly available, thus encouraging other researchers to use it as a benchmark and to extend it with new techniques for supporting user's navigational activity.

## Categories and Subject Descriptors

H.5.4 [**Hypertext/Hypermedia**]: User Issues, Navigation

## General Terms

Algorithms, Experimentation, Measurement

## Keywords

Web behavior, Revisitation Prediction, Contextual Support

## 1. INTRODUCTION

Nowadays, millions of people browse the Web every second, navigating from site to site and producing massive amounts of navigational log data. These data have the intrinsic potential to provide a solid basis for understanding individual user's behavior. User modeling is the first step towards this direction, laying the foundation for recommendation and prediction techniques that facilitate users' surfing activity.

More than a decade ago, Nielsen claimed that, rather than investing time and energy on trying to predict individual user's needs, it would be more fruitful to enhance the overall system design[1]. In contrast to his assertion, we share the vision of the adaptive hypermedia community, supporting the idea that "one size does not fit all" [9]. Much has changed since Nielsen's declaration, with the majority of contemporary systems (especially web-based ones) incorporating recommendation mechanisms to suggest resources (e.g., web pages, files or products) to their users according to an underlying prediction model.

Indeed, many applications can benefit from effective methods of user modeling, such as Web search, where predictive models have improved the ranking of search engine results [8]. Navigational information is actually considered more important than text keywords, since relevant web pages are typically re-ranked according to the distribution of visits over them. Hence, the more accurate the predictive models are, the better search results they yield.

Similarly, individual users can benefit to a large extent from methods predicting and recommending their next page request. Both in their working and in their personal environment, they usually have to handle repetitive but infrequent tasks, revisiting pages after a considerable amount of time [10]. Although users typically employ bookmarks to facilitate such activities, the usability of their bookmark declines rapidly with the constant increase of its size [10]. There is, therefore, a great need for new methods that facilitate users' revisitation activity. This applies not only to client-side settings, but to server-side ones, as well.

In this paper, we introduce a generic surfing prediction library that serves this need through a set of methods aligned in three layers. The first one — called *ranking layer* — comprises a set of functions that, based on click data, sort a set of web pages in descending order of their likelihood to be accessed in the immediate future. The second tier — called *propagation layer* — conveys methods that enhance the ranking ones with contextual information from past surfing activity in the form of frequently co-occurring resources. The third tier — called *drift layer* — encompasses methods that adapt the associations captured by the propagation layer to the continuously changing interests of the underlying user(s). The abstraction that lies at the core of our framework ensures its generality, allowing for the uni-

[1]http://www.useit.com/alertbox/981004.html

form treatment of both aspects of the next-page prediction problem: the server-side and the client-side one. Further, our framework constitutes a real-time solution for predicting navigational activity (i.e., it involves no training phase of machine learning algorithms), that is simple to implement and integrate into a user interface.

Special care has been taken to make our library extensible, so that adding new or improving existing methods in any tier is a straightforward procedure. This is ensured by the transparency of the strictly defined interfaces of each layer, which are analytically described in Section 4. We have also made public both the implementation and the data used in this paper under the *SUPRA*[2] project of SourceForge[3]. 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.

In summary, the main contributions of this paper are the following:

- We introduce a generic, real-time library that is suitable for predicting the next page request uniformly for both server-side and client-side applications.

- We extensively evaluate the methods of our library through a thorough experimental study, involving two voluminous, real-world datasets. The results verify the effectiveness and the accuracy of their predictions.

- We have made publicly available the implementation of our library as well as our experimental data. Through our formalizations, we provide succinct guidelines to other researches and encourage them to experiment with our framework and to extend it with novel methods.

The rest of this paper is organized as follows: in Section 2 we discuss related work, while in Section 3 we formally define the problems we are tackling. Section 4 contains a detailed description of the library we introduce, and Section 5 elaborates on the experimental evaluation. We wrap up our work with final remarks and future plans in Section 6.

## 2. RELATED WORK

Several past works have explored surfing behaviors with respect to users' revisitation activity. Although they vary in their estimations, they all recognize that revisitation constitutes a large part of the Web activity. Herder, for instance, quantifies it to 50% of the overall Web traffic [17], while Cockburn and McKenzie approximate it to 80% [10]. They also noted that bookmarks, the most popular tool for supporting revisitation, invariably involve managing and organizational problems, due to the increasing size of their collections. Novel approaches are, thus, needed to facilitate this dominant web activity.

We distinguish works relevant to revisitation in two broad categories: *revisitation analysis*, which includes studies that explore patterns in this activity, and *revisitation prediction*, which involves works introducing new prediction models for facilitating revisitation.

[2]*SUPRA* stands for "SUrfing Prediction fRAmework"
[3]http://sourceforge.net/projects/supraproject/

### *Revisitation Analysis.*

Tauscher and Greenberg describe in [26] two important patterns of revisitation: first, most page revisits pertain to pages accessed very recently (i.e., the probability for a page to be revisited decreases steeply with the number of page requests since the last visit). Second, there is a small number of highly popular pages that are visited very frequently (i.e., the probability for a page to be revisited decreases steeply with its popularity ranking).

Revisitation behavior has been distinguished by Obendorf et al. into three different categories: short-term (backtrack or undo), medium-term (re-utilize or observe), and long-term revisits (rediscover) [21]. The authors argue that the back button is the most commonly used tool for short-term revisit. Medium-term revisits are facilitated through the automatic URL completion function, which is activated when typing the page address directly into the address bar. However, long-term revisits, which involve a broad range of pages accessed on a less frequent basis, are poorly supported: users often do not remember the exact address, and ironically browsers do not 'remember' it either. Adar et al. further argue in [1] that short-term revisits involve hub-and-spoke navigation, visiting shopping or reference sites or pages on which information was monitored. Medium-term revisits pertain to popular home pages, Web mail, forums, educational pages and browser homepages, whereas long-term ones involve the use of search engines with respect to weekend activities (e.g., going to the cinema).

Though analyzing the phenomenon of revisitation in depth and providing significant insights into it, none of these works aims at predicting it.

### *Revisitation Prediction.*

The problem of the next-page prediction has been extensively studied in the literature. The method that has prevailed in this field (at least in terms of popularity), is the Association Rules Mining. More specifically, *association rules* ($AR$) is a well established method for effectively identifying related resources without taking into account their order of appearance (i.e., pages that are typically visited together, in the same session, but not necessarily in the same order) [3, 4]. They are ideal for recommending resources related to another one, as they disregard the ordering relation between items. For this reason, numerous works have investigated the functionality of AR variations [2, 20, 13, 18, 25]. For example, a recent work by Kazienko explores indirect AR for web recommendations, involving resources that are not "hardly" connected as in typical AR [18].

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. Second, they fail to recommend rarely visited, non-obvious and serendipitous items, since such resources never reach the minimum support limit. Third, disregarding the order of itemsets invariably leads to loss of information about the frequency of different patterns that involve the same resources (i.e., the 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 models like Markov models are particularly popular [28, 5, 27, 11, 12, 6]. Slightly different from these models are sequence mining techniques that do not take into account the strict order between items [4, 24, 23]. A comparison of such techniques with AR was conducted by Géry and Haddad [15]. The authors evaluated AR against *Frequent Sequences*, which can be considered equivalent to association rule mining over temporal data sets, and *Frequent Generalized Sequences*, which constitute a more flexible form of the previous technique, involving wildcards [14]. The results suggest that the plain Frequent Sequence Mining performs better in revisitation prediction. Nevertheless, all these methods still suffer from the inability to predict and recommend *unseen items* (i.e., those not included in the training set).

With the aim of introducing a prediction method that is equally effective with unseen data, Awad et al. combined the Markov model with Support Vector Machines (SVM) under Dempster's rule [6]. They experimentally compared their hybrid model with the individual methods comprising it and with AR, verifying the superiority of their approach, especially when domain knowledge is incorporated into it. Although this is a considerable step towards a method with better generalization capabilities, it is far from being practical; it requires a separate SVM classifier for each one of the available resources and involves a considerably high training time. In fact, their experimental study involved 5,430 classifiers and 26.3 hours of training for a single dataset.

In a more recent work by Parameswaran et al. [23], the authors coin *precedence mining* and build a suite of recommendation algorithms based on it. They model users' history as a set of items that have co-occurred in the past without considering their order of appearance and predict the set of items that are most likely to follow, regardless of their actual order (i.e., not necessarily in the next action of the user). Though quite interesting, they explicitly stress that their approach is not crafted for the next-page prediction problem.

In contrast to existing works, we introduce a *real-time* framework for the next-page prediction problem that requires no training set, thus being able to predict and recommend unseen items. Moreover, it is inherently capable of incorporating contextual information in a generic way, uniformly handling both the server-side and the client-side versions of this particular problem.

Closer to our work are the predictive algorithms presented in [7]. In essence, they constitute scoring schemes that aim at a-priori identifying the next-page request. They are compatible with the first layer of our framework, but they are applied in a different context, as the authors partition user's navigational activity into "Web trails" and primarily aim at detecting the next recurring one. In addition, they disregard associations between pages (second layer of *SUPRA*) and the drift in user's interests (third layer of *SUPRA*).

## 3. PROBLEM DEFINITION

At the core of this work lie the notions of *web site* and *web page*. The former is considered as a Web domain (e.g., www.l3s.de), which comprises a set of web pages (e.g., www.l3s.de/people). Based on this definitions, we formalize the two aspects of the *next-page prediction problem* that we are tackling: the server-side and the client-side one.

The server-side next-page prediction problem aims at detecting which of the visited web pages of a web site will be accessed in the next request, judging exclusively from the past navigational activity of its entire user base. More formally, it is defined as follows:

**Problem** 1 (SERVER-SIDE NEXT-PAGE PREDICTION). *Given the set of web pages $\mathbf{P_s} = \{p_1, p_2, ...\}$ of a web site $s$, which have been accessed during the past $n$ page requests, $\mathbf{R_s} = \{r_1, r_2, \ldots, r_n\}$, order them in such a way that the ranking position of the page $p_i$ that will be* re-accessed *in the next request $r_{n+1}$ is the highest possible.*

The client-side next-page prediction problem copes with identifying which web pages, among those visited by a specific user in the past, will be revisited in her next request. Formally, it is defined as follows:

**Problem** 2 (CLIENT-SIDE NEXT-PAGE PREDICTION). *Given the collection of web pages $\mathbf{P_u} = \{p_1, p_2, ...\}$ that have been visited by a user $u$ during her past $n$ page requests $\mathbf{R_u} = \{r_1, r_2, \ldots, r_n\}$, order them so that the ranking position of the page $p_i$ that she will* revisit *in her next request $r_{n+1}$ is the highest possible.*

There are two fundamental differences between these two problems:

1. Problem 1 involves a set of separate users, with different behavior and interests, who are possibly accessing the given web site at the same time (i.e., their sessions are overlapping). In contrast, Problem 2 entails the navigation activity of a single user, whose interests change with the time, but probably not as dramatically as in the case of a web site's user base. From this aspect, it seems easier to identify patterns in the user-based scenario.

2. The information space of Problem 1 covers solely the web pages of the web site at hand, excluding any resources that lie out of its borders. On the other hand, the predictions of Problem 2 are not confined to a specific set of pages; they can potentially involve the whole Web. Hence, the distinct page transitions involved in the activity of a web site are probably fewer than those in the activity of a user, making Problem 1 a seemingly easier scenario than Problem 2.

Note, however, that the second difference does not imply that the set of pages visited by a user is necessarily larger than that of a web site. Neither does it mean that a web site has a confined set of resources; depending on its traffic, nature and scope, their number can increase much faster than the pages visited by a single user. As an example, consider the intranet of a multinational company with pages added on a daily bases by its employees in order to share knowledge with their colleagues all over the world.

Our focus in the following is not to compare the peculiarities of these two aspects of the next-page prediction problem. Instead, we aim at proposing a generic methodology that is capable of uniformly handling both problems through the abstraction of the access history of each web page and the contextual information it entails. Its functionality is real-time, in the sense that it does not involve the burden of training data and of configuring machine learning algorithms.

| **Ranking Methods** | | | **Propagation M.** | | **Drift Methods** | |
|---|---|---|---|---|---|---|
| *event-based* | *time-based* | *hybrid* | *order-preserving* | *order-neutral* | *event-based* | *time-based* |
| *FR* | *PD* | *HDM* | *CTM* | *AM* | 5*HR* | *DM* |
| | | *HQM* | *DTM* | | *TR* | *WM* |
| | | *HHM* | *ITM* | | | *MM* |
| | | | *STM* | | | |

**Table 1: Summary of all the methods of the *SUPRA* framework.**

## 4. APPROACH

The definitions of Problems 1 and 2 stress that methods coping with them should facilitate the access to pages used in the past, rather than trying to suggest not-visited, yet relevant ones. In this work, we present a library of methods that deal with these problems by ranking visited web pages according to their likelihood of being revisited in the next request; the higher this likelihood is, the higher is the ranking of the corresponding web page. This ranking is updated after each page request, and the higher the ranking position of the subsequently accessed page, the better the prediction. This is in line with the intuition behind ranking search engines' query results: as noted by Hawking et al. [16], the lower the ranking of the desired resource, the better the performance of the search engine.

In more detail, our library consists of three tiers of methods. The first one involves usage-based *ranking methods* that estimate for each web page the likelihood that it will be accessed in the next request. They derive their estimation from evidence drawn from the surfing history of a web site or user, such as the recency and/or the frequency of accesses to each page. The second layer covers *propagation methods*; they capture repetitiveness in the navigational activity of a web site or user and identify groups of 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*; they employ a sliding time frame to periodically discard outdated page associations, thus adapting the propagation methods to the volatile interests of the user(s). In conjunction, the three layers of our framework form a comprehensive approach that tackles all aspects of the revisitation activity.

In the following, we elaborate on the different types of methods conveyed by each layer and illustrate their functionality with concrete techniques. In summary, they are presented in Table 1 along with the techniques of the *SUPRA* framework that belong to them. Their implementation is freely available through SourceForge[4], thus encouraging other researchers to experiment with our framework and extend it with new mechanisms. Special care has been taken to make this a straightforward procedure through the strict formalization of each layer's transparent interface that is presented in the following; any implementation complying with Definitions 3 and 7 can be easily integrated in our library. Note also that the real-world data employed in our experiments have been publicly released, so that they can be used as a benchmark for prediction algorithms.

[4]See `http://sourceforge.net`.

### 4.1 Ranking Methods

The aim of ranking methods is to provide each web page with a numerical estimation of the likelihood that it will be accessed in the next transaction. In essence, they work as follows: after each page request, the selected ranking method goes through all visited web pages, estimates their value and then sorts them in descending order of their expected value. The estimation is based on the access history of each web page, which is represented either by the timestamps or by the indices of the related requests. The abstract form of these two types of evidence captures the navigational activity of web sites and users in a uniform way, accommodating both Problems 1 and 2. They are formally defined as follows:

**Definition** 1 (Request Indices). *Given all page requests* $\mathbf{R}$ *of a system (Problem 1) or a user (Problem 2), the* request indices *of a page* $p_i$ *— denoted by* $\mathbf{I_{p_i}}$ *— is the set of the serial numbers of those requests in* $\mathbf{R}$ *that pertain to* $p_i$*. The serial number of the chronologically first request is 1 and is incremented by 1 for each of the subsequent page visits.*

**Definition** 2 (Request Timestamps). *Given all page requests* $\mathbf{R}$ *of a system or a user, the* request timestamps of a page $p_i$ *— denoted by* $\mathbf{T_{p_i}}$ *— is the set of timestamps of those requests in* $\mathbf{R}$ *that pertain to* $p_i$*.*

Depending on which type of evidence a ranking method builts on, we distinguish three categories of such methods: the event-based, the time-based and the hybrid ones. They are analytically presented in the following.

#### 4.1.1 Event-based

The ranking methods of this category take as input the request indices of a page and draw its ranking value from the evidence contained in them. These methods interpret page visits as a sequence of events and exclusively take into account their relative position; to estimate the contribution of an individual visit to the total ranking value of the corresponding page, they solely consider the number of events that have elapsed since its occurrence (i.e., without taking the actual time of occurrence into account). More formally, an event-based ranking method is defined as follows:

**Definition** 3 (Event-based Ranking Method). *An* event-based ranking method *is a function that takes as input the request indices* $\mathbf{I_{p_i}} = \{i_1, i_2, \ldots, i_k\}$ *of a page* $p_i$ *together with the index of the latest request* $i_n$ *of a web site or user, and produces as output a value* $v_{p_i} \in [0, 1]$ *that is proportional to the likelihood of* $p_i$ *being accessed at the next page request* $r_{n+1}$ *(i.e., the closer* $v_{p_i}$ *is to* 1*, the higher is this likelihood).*

This family of methods are represented in our framework by the *decay ranking functions*, which estimate the value of a web page as a comprehensive combination of the frequency and the recency of its visits [22]. In more detail, the value of a web page $p_i$ after $i_n$ requests of a web site or user is derived from the following formula:

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

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 transaction $i_n$; its output quantifies the contribution of this request to the total value of $p_i$.

The main types of decay functions were experimentally compared in [22], with the *polynomial* ones consistently outperforming the *exponential* and the *logarithmic* ones. The reason is that their smooth decay balances harmonically the recency and the degree of usage of web pages. In contrast, exponential functions convey a steep decay that puts more emphasis on the recency of usage, whereas the logarithmic functions excessively promote 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:

$$d(i_j, i_n) = \frac{1}{1 + (i_n - i_j)^\alpha}. \qquad (2)$$

In the following, we exclusively consider polynomial decay functions, denoted by $PD$ for short.

### *4.1.2 Time-based Ranking Methods*

This family of ranking functions receives as input the request timestamps of a page and estimates its ranking value based exclusively on them. In more detail, the contribution of each request to the total ranking value depends on the actual time the respective page visits took place and the time that has elapsed ever since (i.e., the time of the latest request). More formally, a time-based ranking method is defined as follows:

**Definition** 4 (TIME-BASED RANKING METHOD). *A* time-based ranking method *is a function that takes as* input *the request timestamps* $\mathbf{T_{p_i}} = \{t_1, t_2, \ldots, t_k\}$ *of a page* $p_i$ *together with the time of the latest request* $t_n$ *of the underlying user or web site, and produces as* output *a ranking value* $v_{p_i} \in [0, 1]$ *that is proportional to the likelihood of* $p_i$ *being accessed at the next page request* $r_{n+1}$.

This type of ranking methods are represented in our framework by the *Frecency* method — symbolized by $FR$ in the following — that is integrated in Mozilla Firefox[5] for page recommendations. In essence, the total value it assigns to a page is equal to the sum of the "bonuses" associated with its individual visits. Each "bonus" is proportional to the recency of the corresponding page request; those 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), but this is out of the scope of Definition 4[6]. Thus, $FR$ puts more emphasis on the frequency of the use of a Web page, discounting to some extent the contribution of old visits.

### *4.1.3 Hybrid*

Apparently, the event-based ranking methods cannot capture temporal patterns that occur periodically in the navigational activity of a user or system. For example, in the client-side environment, it is common for users to visit the same news sites early in the morning, before starting their actual work. Similarly, in the server-side environment, it is common for the employees of a company to consult the wiki page containing the topics of a fixed, weekly meeting, before actually attending it. On the other hand, time-based ranking methods cannot capture recent event patterns; imagine the users of a web site that shift their focus from page A to page B within a short period of time (e.g., few hours); $FR$ will continue to rank the former page higher than the latter one until the accesses of B exceed those of A. $PD$, though, will be much faster in adapting its recommendations to the new trend.

To accommodate such behaviors, hybrid ranking methods combine the functionality of time-based techniques with that of the event-based ones: they derive the ranking value of a page from the number of events and the time that has elapsed, after its accesses. More formally, they are defined as follows:

**Definition** 5 (HYBRID RANKING METHOD). *A* hybrid ranking method *is a function that takes as input the request indices* $\mathbf{I_{p_i}}$ *and the request timestamps* $\mathbf{T_{p_i}}$ *of a page* $p_i$ *along with the time* $t_n$ *and the index* $i_n$ *of the latest request. As output, it produces a value* $v_{p_i} \in [0, 1]$ *that is proportional to the likelihood of* $p_i$ *being accessed at the next page request* $r_{n+1}$.

In our framework, this type of ranking methods is represented by techniques that rely on the polynomial decay model and enhance it by adjusting its decay rate according to temporal evidence. More specifically, we consider the *hybrid day model* ($HDM$), the *hybrid day quarter model* ($HQM$) and the *hybrid hour model* ($HHM$), which double the exponent $\alpha$ of the polynomial decay function (cf. Formula 2) for the past requests that took place on the same day of the week, on the same quarter of the day[7] and on the same hour of the day with the latest request, respectively. The exponent $\alpha$ remains unmodified for the rest of the visits, thus increasing the contribution of the requests that share some temporal information with the latest page visit.

**Normalization.** Before presenting the propagation methods, it is worth noting that we employ normalization in order to restrict the output of all ranking methods in the interval $[0, 1]$ independently of their functionality. To this the end, we divide the ranking value of all pages with the currently largest one after each iteration.

## 4.2 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 associated ones. In this way, the highest the value of a web page is, 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 *session*: a collection of pages visited by the same

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

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

[7]We split the hours of a day in four equal intervals and consider the following quarters of a day: morning (6am to 12am), noon (12am to 6pm), afternoon (6pm to 12 pm) and night (from 12pm to 6am).

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 0 & 0 \\ B & 0 & 0 & 1 & 0 \\ C & 0 & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 \end{array}$$

(a)

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 1 & 1 \\ C & 1 & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 \end{array}$$

(b)

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & \frac{1}{2} & \frac{1}{4} \\ B & \frac{1}{4} & 0 & 1 & \frac{1}{2} \\ C & \frac{1}{2} & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 \end{array}$$

(c)

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 2 & 4 \\ B & 4 & 0 & 1 & 2 \\ C & 2 & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 \end{array}$$

(d)

**Figure 1: The values of several types of Transition Matrices after the last page request of the session: $S_1 : A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$. a) corresponds to $STM$, b) to $CTM$, c) to $DTM$ and d) to $ITM$.**

user in sequence, during a specific period of time. On the client-side, sessions are internally defined by browsers, taking into account the time and the tabs through which pages are visited. On the server-side, though, it is not clear which pages belong to the same session. Following the literature, we define them as follows:

**Definition** 6 (Server-side Session). *A* server-side session $\mathbf{S} = \{p_1, p_2, \ldots, p_k\}$ *is a bag containing in chronological order all pages $p_i$ visited by a specific IP address for a time period of up to 25.5 minutes [15, 26].*

Based on Definition 6, propagation methods can be defined as follows:

**Definition** 7 (Propagation Method). *A* propagation method *is a function that takes as input the last page $p_i$ of a session $\mathbf{S}$ and defines appropriately the degree of connection between $p_i$ and all other pages visited during $\mathbf{S}$. Thus, for each pair of 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 the transition $X \rightarrow Y$ is).*

In this work, we distinguish two families of propagation methods: the *order-preserving* ones, which take into account the order of the transactions within a session, and the *order-neutral* ones, which disregard this order. For the former case, we consider transition matrices, while for the latter we examine association matrices. We elaborate on them in the following subsections.

### *4.2.1 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 the page requests of a session (ordered by time), they build the associations between pages according to their sequence of appearance; that is, each page is connected only with the pages preceding it. To capture these chronological patterns in the navigational activities of systems and users, we introduce the transition matrix.

In more detail, a *transition matrix* ($TM$) is a two dimensional structure with its rows and columns representing the set of all visited web pages of a system ($\mathbf{P_s}$ of Problem 1) or user ($\mathbf{P_u}$ of Problem 2). Each cell $TM(x, y)$ expresses the number of times that a user visited page *$y$ directly after $x$*. Given that a transition matrix respects the order of accesses within a session, it is not a symmetrical one, i.e., 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 (i.e., $\forall x \; TM(x, x) = 0$). This is because there is no point in associating a page with itself. In case a retrieved web page is re-visited by the same user or another one in the subsequent request, its ranking does not need to be boosted by the propagation method, as it is already high enough, due to the value assigned to it by the ranking method.

In the following, we introduce 4 different techniques for correlating web pages according to the past navigational activity in the context of a $TM$. 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 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* ($STM$). The rationale behind this approach is the expectation that requests tend to occur in the same strict order. Thus, the frequencies it defines work exactly as a first-order Markov model; for each transition $x \rightarrow y$, only the value of the cell $TM(x, y)$ is incremented by one. Figure 1(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* ($CTM$). 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; that is, 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 (cf. Figure 1(b)).

3. *Decreasing Continuous Connectivity Transition Matrix* ($DTM$). This strategy operates in a similar way as the previous one, but increments the cells of $TM$ by a decay parameter that represents the distance in terms of number of transitions intervening between the corresponding web pages. Therefore, this form of $TM$ lies in the middle of $STM$ and $DTM$, 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 1(c) depicts the whole $TM$ after the transition $D \rightarrow A$.

4. *Increasing Continuous Connectivity Transition Matrix* ($ITM$). This is the inverted version of the previous strategy. Instead of decreasing the value added to

$$\begin{array}{c|cccc} & A & B & C & D \\ A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 1 & 1 \\ C & 1 & 1 & 0 & 1 \\ D & 1 & 1 & 1 & 0 \end{array}$$

**Figure 2: The values of the Association Matrix after the last page request of the session: $S_1 : A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$.**

$TM(x, y)$, it increases it proportionally to the distance of pages $x$ and $y$. 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 $TM$ produced by this rule after transition $D \rightarrow A$ is presented in Figure 1(d).

It is worth noting that $STM$ was also employed in Awad et al. [6], but its frequencies were merely used as features to a classification algorithm. Moreover, $CTM$ was also introduced in Parameswaran et al. [23] as the means of providing the frequencies of the probabilistic analysis lying at the core of precedence mining.

#### 4.2.2 Order-Neutral Propagation Methods

In contrast to order-preserving methods, the order-neutral ones are based on the idea that the temporal order of transactions 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*, symbolized as $AM$; similar to $TM$, it is a matrix whose rows and columns correspond to the already visited set of web pages $\mathbf{P_s}$ or $\mathbf{P_u}$. The difference is that $AM$ is built simply by associating all pages visited in a single session with each other; 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. Continuing our example with session $S_1$, the resulting $AM$ has all non-diagonal cells equal to one, as all resources were accessed during the same session (Figure 2).

#### 4.2.3 Combining Ranking with Propagation Methods

To combine a raking method with a propagation technique, 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 $TM(x, y)$ or $AM(x, y)$, we increment the value $v_y$ assigned to page $y$ by the ranking method 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 $p(x \rightarrow y) = \frac{TM(x,y)}{\sum_i^{in} TM(x,i)}$ or $p(x \rightarrow y) = \frac{AM(x,y)}{\sum_i^{in} AM(x,i)}$, and
- $v_x$ is the value of $x$ estimated by the ranking method.

### 4.3 Drift Methods

Unlike ranking methods, propagation ones do not support the drift in the focus of user's interests; the connections stored in their matrix remain static and can possibly become outdated, as they cannot adapt to the constantly changing habits and interests of users. In the literature, two main approaches that support *concept drift* have been proposed: the first one relies on decay functions to gradually reduce the weight of page associations, while the second one comprises the window-based methods [19]. Based on a sliding window, the latter retain the most recent associations and discard the old ones. Due to their simplicity, the third layer of our framework exclusively encompasses methods of the latter type to enable the dynamic nature of the propagation methods.

Depending on the way the size of their window is specified, *window-based drift methods* are distinguished into *event-based* and *time-based* ones. The former specify their window with respect to the size of a batch of considered requests and are formally defined as follows:

**Definition** 8 (EVENT-BASED DRIFT METHODS). *Given the page requests* $\mathbf{R}$ *of a user or web site, 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 latest* $n$ *page requests of* $\mathbf{R}$.

Time-based drift methods specify their window with respect to the considered period of time and are formally as follows:

**Definition** 9 (TIME-BASED DRIFT METHODS). *Given the page requests* $\mathbf{R}$ *of a user or web site, 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* $\mathbf{R}$ *that occurred in the latest* $t$ *temporal units (e.g., days or weeks).*

In this work, we examine the performance of two methods of the former type and three of the latter. More specifically, the event-based drift methods we consider are the *500-requests* — denoted by $5HR$ — and the *1,000-requests* — symbolized by $TR$. As their name suggests, they take into account the five hundred and the one thousand most recent page requests of the input $R$, respectively. From the time-based drift methods, we consider the *Day-*, the *Week-* and the *Month-model*, which update the underlying propagation matrix so that it maintains the associations of the last day, the last week and the last month, respectively. They are symbolized by $DM$, $WM$ and $MM$, respectively.

## 5. EVALUATION

### 5.1 Datasets

The goal of our experimental study is threefold: (i) to identify the best performing method from each layer, (ii) to compare the performance between the layers, and (iii) to

|  | $\mathbf{D_{wiki}}$ | $\mathbf{D_{cms}}$ | $\mathbf{D_{users}}$ |
|---|---|---|---|
| Users | 1,742 | 24,614 | 25 |
| Page Requests | 35,223 | 359,211 | 137,272 |
| Web Pages | 693 | 4,880 | 67,641 |
| Sessions | 4,783 | 79,794 | 4,552 |
| Session Size | 7.36 | 4.50 | 28.28 |
| Entropy | 7.17 | 7.31 | 7.41 ± 1.13 |
| Start Date | 15.09.2008 | 31.01.2008 | 01.08.2004 |
| End Date | 15.02.2010 | 08.03.2010 | 31.03.2005 |
| Elapsed Days | 518 | 746 | 104.97 ± 32.41 |
| Backtracking | 32.74% | 31.00% | 19.41% ± 7.45 |
| Revisitation Rate | 98.03% | 98.64% | 48.97% ± 11.33 |

**Table 2: The data sets used in our experimental evaluation.**

compare the performance of each layer in the two different application areas: the server-side and the client-side one. To this end, we thoroughly evaluate our framework over three voluminous, real-world datasets: two for server-side and one for client-side applications. Their technical characteristics are presented in Table 2.

The *server-side datasets* comprise the navigational activity of two distinct systems: the internal wiki of the L3S Research Center (denoted as $D_{wiki}$ from now), and the content management system of the European IP project OKKAM[8] (referred to as $D_{cms}$ in the following). They entail completely heterogeneous information spaces that are suitable for examining the performance of our approach in diverse settings: $D_{wiki}$ comprises a small set of web pages that are used to a small extent (68 page requests per day), whereas $D_{cms}$ contains 7 times more pages that have been accessed to a considerably higher degree (482 requests per day). Note that, in the absence of any other evidence in the logs providing these datasets, we exclusively relied on the time and the IP of the page requests in order to estimate the number of users and to divide the page requests into sessions according to Definition 6.

Quite diverse is the *client-side dataset*, as well, which is symbolized as $D_{users}$ in the following. It consists of the activity of a participant pool with 25 individuals, 19 male and 6 female. Their average age is 30.5, ranging from 24 to 52 years. The participants were logged for some period between August, 2004 and March, 2005. The average time span of the actual logging periods was 104 days, with a minimum of 51 days and a maximum of 195 days. Participants were logged in their usual contexts — 17 at their workplace, 4 both at home and at work, and 4 just at home. In total, 137,737 page requests were recorded during the logging period. They are unevenly distributed over the 25 users, as some users visited few hundred distinct pages in the course of few and short sessions, while some others visited tens of thousands of pages over much longer sessions.

## 5.2 Setup

In the course of our experimental evaluation, we simulated the navigational activity of each system and user independently of the others. After each page request, the ranking of all visited pages was updated, and, in case the next access was a revisitation, the position of the corresponding web resource was recorded. From these ranking positions, we derived the following measures to evaluate the performance of each prediction method:

1. *Success at 1* ($S$@1). It denotes the portion of requests that involved a revisited web page placed at the highest ranking place by the prediction method. Thus, it provides evidence for the accuracy of a prediction method in identifying the next revisited page. The higher this percentage is, the better is the performance of the method.

2. *Success at 10* ($S$@10). It stands for the percentage of requests pertaining to a revisited page that is ranked in one of the top 10 positions. Thus, it expresses the actual usability of the prediction method, as users typically have a look only at the first 10 pages presented to them (just like they do with web search engine results [16]). Similar to $S$@1, the higher its value is, the better the performance of the method is.

3. *Average Ranking Position* ($ARP$). It represents the position a revisited page is found on average in the ranking list that the prediction method produces. Thus, it provides an estimation of the overall performance of a prediction method, considering the performance over all the revisitations in the navigational history of a system or user, instead of just the top ranked ones. The lower its value is, the better is the performance of the prediction algorithm.

On the whole, the combination of these three metrics provides a comprehensive estimation of the effectiveness of a prediction algorithm in identifying the subsequently revisited pages. They cover both the predictions that are indeed useful for users ($S$@1, $S$@10) as well as their overall performance when considering all revisitations ($ARP$).

## 5.3 Results

In the following, we present and analyze the performance of the methods of each layer in a distinct section. Note that we present the outcomes for each server-side dataset separately, whereas, for the client-side dataset, we show the mean and the standard-deviation of each measure across all users.

### *5.3.1 Ranking layer*

Before commenting on the actual performances, it is worth noting that $PD$ can (and sometimes needs to) be calibrated in order to find the best performing exponent $\alpha$ of the Formula 2. The reason is that $\alpha$ determines the steepness of the decay model (i.e., the balance between recency and popularity); values closer to 0.0 favor the latter, while values closer to 2.0 emphasize recency. In our experimental study, though, we solely considered $\alpha = 1.25$, since this was found to have the best performance in [22]. The same configuration was also employed for $HDM$, $HQM$ and $HHM$, all of which are based on $PD$.

The exact performance of each method of the first layer is presented in Table 3. We can easily observe the following pattern: $PD$ consistently achieves the best value for all three metrics, in contrast to $FR$, which exhibits the worst one in all cases. The reason is probably the different balance between frequency and recency that the two methods entail: $FR$ favors the former over the latter, whereas $PD$ puts more emphasis on recency. In fact, there is a steeper

[8]www.okkam.org

| | | FR | PD | HDM | HQM | HHM |
|---|---|---|---|---|---|---|
| $D_{wiki}$ | **ARP** | 42.37 | 18.72 | 19.05 | 19.17 | 18.78 |
| | **S@1** (%) | 5.74 | 32.58 | 32.26 | 32.59 | 32.59 |
| | **S@10** (%) | 35.57 | 70.82 | 70.52 | 70.44 | 70.78 |
| $D_{cms}$ | **ARP** | 85.73 | 39.46 | 40.36 | 40.74 | 39.64 |
| | **S@1** (%) | 23.24 | 34.34 | 34.01 | 33.64 | 34.22 |
| | **S@10** (%) | 40.49 | 59.82 | 59.77 | 59.61 | 59.76 |
| $D_{users}$ | $\overline{\textbf{ARP}}$ | 290.17 | 52.59 | 74.54 | 74.35 | 64.78 |
| | $\sigma_{\textbf{ARP}}$ | 347.44 | 50.19 | 67.71 | 66.83 | 57.15 |
| | $\overline{\textbf{S@1}}$ (%) | 12.57 | 20.33 | 19.24 | 19.23 | 19.28 |
| | $\sigma_{\textbf{S@1}}$ | 7.72 | 7.54 | 7.51 | 7.48 | 7.48 |
| | $\overline{\textbf{S@10}}$ (%) | 32.33 | 74.11 | 71.31 | 70.70 | 70.13 |
| | $\sigma_{\textbf{S@10}}$ | 10.99 | 7.65 | 6.68 | 6.83 | 7.24 |

**Table 3: Performance of the first layer's ranking methods over the server-side data sets, $D_{wiki}$ and $D_{cms}$, along with their average performance over the client-side one, $D_{users}$.**

| | | ATM | CTM | DTM | ITM | STM |
|---|---|---|---|---|---|---|
| $D_{wiki}$ | **ARP** | 12.38 | 12.97 | 11.39 | 15.26 | 11.19 |
| | **S@1** (%) | 32.81 | 32.53 | 32.72 | 32.15 | 31.15 |
| | **S@10** (%) | 74.48 | 73.92 | 76.26 | 72.17 | 77.50 |
| $D_{cms}$ | **ARP** | 29.16 | 33.88 | 25.43 | 40.93 | 25.72 |
| | **S@1** (%) | 34.67 | 34.33 | 34.54 | 34.05 | 34.42 |
| | **S@10** (%) | 62.56 | 61.86 | 66.32 | 60.13 | 68.69 |
| $D_{users}$ | $\overline{\textbf{ARP}}$ | 28.08 | 31.54 | 24.83 | 42.04 | 29.71 |
| | $\sigma_{\textbf{ARP}}$ | 22.60 | 25.76 | 19.86 | 38.36 | 27.69 |
| | $\overline{\textbf{S@1}}$ (%) | 20.84 | 21.46 | 22.76 | 20.69 | 24.63 |
| | $\sigma_{\textbf{S@1}}$ | 7.21 | 7.11 | 6.92 | 7.21 | 6.45 |
| | $\overline{\textbf{S@10}}$ (%) | 76.13 | 75.94 | 80.88 | 74.23 | 82.35 |
| | $\sigma_{\textbf{S@10}}$ | 7.96 | 7.47 | 5.72 | 7.52 | 5.19 |

**Table 4: Performance of the second layer's propagation methods over the server-side data sets, $D_{wiki}$ and $D_{cms}$, along with their average performance over the client-side one, $D_{users}$. In all cases, $PD$ was selected from the first layer.**

decay for the contribution of each page request for $\alpha > 1$. The hybrid models lie very close to $PD$ and, as expected, the finer the granularity of their core time interval is, the smaller their difference from $PD$ is. Especially for the server-side datasets, the performance of $HHM$ is almost identical with that of $PD$. On the client-side, the same pattern is followed, but the differences between $PD$ and the hybrid models are larger — in favor of the former. This obviously implies that the temporal patterns lying at the core of these models are rather weak, especially in the case of individual users.

Comparing the performances of the server-side datasets with the client-side ones, we can notice that $ARP$ is significantly lower for $D_{wiki}$ and $D_{cms}$ than for $D_{users}$, across all ranking methods. This is probably caused by the higher entropy of the client-side navigational activity; as noted in Table 3, entropy rises from 7.17 for $D_{wiki}$ to 7.31 for $D_{cms}$ and to 7.41 (on average) for $D_{users}$. This means that each revisit can refer to 144 distinct pages in the case of $D_{wiki}$, to 159 for $D_{cms}$ and to 170 for $D_{users}$. Note, though, that entropy is even higher for some users, taking a maximum value of 9.90, which corresponds to 956 candidate, distinct, revisited pages.

Regarding $S$@1, we can notice that it is higher for server-side datasets for most of the ranking methods. To identify the reason, we estimated the extent of *backtracking*; that is, how often do two consecutive transactions pertain to the same page. It was found to be relatively high for $D_{wiki}$ and $D_{cms}$, taking values over 30% for both of them, while being significantly lower for $D_{users}$, amounting to less than 20% on average. On the other hand, $S$@10 exhibits similar values in $D_{wiki}$ and $D_{users}$, with $D_{cms}$ yielding significantly lower values. This phenomenon is directly related to the search space of each dataset; the more pages have been visited in a system or by a user, the harder is to rank the subsequently revisited page in one of the top 10 positions. Indeed, $D_{wiki}$ involves the smallest search space with 695 distinct pages, followed by $D_{users}$ with 2,706 distinct pages per user (on average), while $D_{cms}$ constitutes the largest web site, encompassing 4,880 pages.

### 5.3.2 *Propagation layer*

To estimate the performance of the second layer, we combined its propagation methods with the best performing ranking one (i.e., $PD$). The results are presented in Table 4. Looking into its numbers, we can observe that $DTM$ and $STM$ consistently exhibit the best performance with respect to all metrics, both for the server- and the client-side datasets. $ITM$, on the other hand, consistently exhibits the worst performance in most of the cases. $ATM$ and $CTM$ follow in close distance from $DTM$ and $STM$, with $ATM$ achieving the best values for $S$@1 over $D_{wiki}$ and $D_{cms}$ and the second best $ARP$ for $D_{users}$. In general, though, the differences between these four algorithms are rather minor, with the deviation between the maximum and the minimum value being lower than 10% in most of the cases.

It is interesting to compare again the server-side with the client-side performance. We can easily notice that the patterns exhibited in the first layer apply only to $S$@1 and $S$@10. In fact, the former remains virtually unchanged, despite the addition of the second layer, whereas the latter is increased in both cases to the same extent. $ARP$, though, exhibits a slightly different evolution; although it continues to take the lowest value for $D_{wiki}$, the substantial difference between $D_{cms}$ and $D_{users}$ is practically extinguished in favor of the latter. In fact, some propagation methods exhibit better $ARP$ for $D_{users}$ than for $D_{cms}$. This indisputably implies that propagation methods manage to negate the impact of entropy through the page-to-page associations recorded in the navigational activity of users and systems.

Comparing the performance of the propagation methods with plain $PD$, we can notice that $ARP$ values are extensively improved (by up to 50%), $S$@1 remains practically stable, while $S$@10 is increased by 10% on average. Note that these patterns apply equally to client- and server-side datasets. They, thus, lead us to the safe conclusion that propagation methods are not suitable for boosting the ranking of the next revisited page alone; instead, they spread ranking values in such a way that the aggregate ranking of the most frequently revisited pages is higher.

### 5.3.3 *Drift layer*

To estimate the performance of the third layer, we first had to identify the best combination of propagation methods with $PD$. As mentioned above, $CTM$ and $STM$ achieve similar performance in all settings, outperforming all other

| | | **5HR** | **TR** | **DM** | **WM** | **MM** |
|---|---|---|---|---|---|---|
| $D_{wiki}$ | **ARP** | 17.20 | 15.80 | 18.71 | 18.69 | 18.68 |
| | **S@1** (%) | 32.11 | 32.41 | 32.39 | 31.98 | 31.87 |
| | **S@10** (%) | 75.84 | 76.88 | 70.83 | 70.88 | 70.93 |
| $D_{cms}$ | **ARP** | 37.69 | 36.05 | 39.45 | 39.37 | 39.17 |
| | **S@1** (%) | 34.41 | 34.60 | 30.08 | 30.45 | 32.12 |
| | **S@10** (%) | 66.32 | 67.68 | 60.11 | 60.44 | 60.97 |
| $D_{users}$ | $\mathbf{A\bar{R}P}$ | 40.78 | 45.18 | 51.82 | 49.05 | 41.66 |
| | $\sigma_{\mathbf{ARP}}$ | 47.36 | 49.53 | 49.74 | 48.15 | 43.00 |
| | $\mathbf{S\bar{@}1}$ (%) | 24.58 | 24.59 | 23.93 | 24.47 | 24.62 |
| | $\sigma_{\mathbf{S@1}}$ | 6.57 | 6.42 | 6.46 | 6.55 | 6.48 |
| | $\mathbf{S\bar{@}10}$ (%) | 80.53 | 81.52 | 76.65 | 79.62 | 81.57 |
| | $\sigma_{\mathbf{S@10}}$ | 6.09 | 5.67 | 6.71 | 5.76 | 5.27 |

**Table 5: Performance of the third layer's drift methods over the server-side data sets, $D_{wiki}$ and $D_{cms}$, along with their average performance over the client-side one, $D_{users}$. In all cases, the same methods were selected from the first and the second layer: $PD$ and $STM$, respectively.**

techniques of the second layer. Among them, we opted for $STM$, given that it consistently has the highest $S$@10 value, while the differences with respect to the other metrics are negligible.

The performance of the drift methods in combination with $PD$ and $STM$ is presented in Table 5. Comparing them with $PD$+$STM$, we can easily notice that their performance is significantly lower in all cases. In fact, they have a considerable impact on predictive performance, moving it closer to that of a plain ranking method. The scarcer the update of the propagation matrix is, the lower is actually the impact on the benefits of $STM$; that is, $TR$ outperforms $5HR$ in most of the cases and so does $MM$ in comparison to $DM$ and $WM$. It is also interesting to notice that event-based drift methods outperform the time-based ones. These patterns apply equally to the server- and the client-side datasets, indicating that in both cases there is no significant change in the associations between revisited pages. For the server-side applications, this can be explained by the closed world effect: the rate of creating new pages was rather low in both systems, thus restricting the choice of users to a limited set of pages. Note, though, that the higher the number of distinct pages is, the more restricted the impact of drift methods is. For the client-side dataset, we cannot draw safe conclusions, due to the limited period of activity logged by our subjects. Therefore, further experimentation is needed over individual users whose recorded activity spans a longer period of time. Similarly, investigation over server-side systems with a significantly higher diversity of web pages is necessary.

## 5.4 Discussion

In our experimental analysis, we compared various algorithms for predicting revisited pages both on the server- and the client-side. These algorithms exploit the following characteristics of revisits:

- Typically, they are focused on pages visited very recently.
- The higher the number of revisits is, the more repetitive is the behavior in terms of frequent transitions between pages.
- There is a small group of pages that is visited very frequently.
- There are temporal patterns in navigational activity of systems of users, in terms of repeated revisits during the same day, the same part of day or the same hour.
- The drift in the interest of the underlying user(s).

It turns out that, even though a small set of frequently accessed pages covers the majority of revisits, popularity alone does not suffice for predicting accurately the next revisited page. This explains the poor performance of $FR$. On the contrary, the recency effect plays a major role in prediction algorithms. Indeed, backtracking is more common than revisiting popular sites, thus ensuring much higher performance for $PD$.

Another important factor in predicting revisits are the behavioral patterns and the contextual information encoded in them (i.e., transitions between pages that co-occur frequently, but not necessarily in the same order). For this reason, the combination of $PD$ with the propagation methods (especially $CTM$ and $STM$) improves significantly upon the list of most recently used pages. The differences, though, become smaller together with the increase of the recency effect (i.e., higher backtracking).

Another interesting observation is that, despite the different assumptions that lie behind the predictions schemes we are considering, there is a strong correlation between their performances per system and user. To put it differently, the better the performance of the best-performing method for one user or system is, the higher are the scores of the other prediction schemes for this user or system ($r > .87$ for all $PD$-based methods). This can be attributed to the inherent characteristics of the navigational activity of systems and users.

Surprisingly, temporal aspects of revisiting behavior — be it hourly, weekly or monthly patterns — do not improve prediction performance. Our evaluation results showed poor performance for both the drift layer and the hybrid ranking methods. These results contradict our everyday intuition that web usage involves routine patterns centered around, for example, working weeks, lunch breaks and spare time. A likely explanation is that our web usage behavior does not follow any temporal patterns — at least not to an extent that is needed for reliable predictions. It may also be the case that these temporal patterns can only be reliably exploited after a longer time period than the one considered in our experiments. One indication for the latter explanation is that for some users the addition of drift methods led to a slight improvement.

In general, we can conclude that the most important factor to be taken into account for predicting revisiting behavior is the recency effect. However, the performance of the predictions is greatly improved when the ranking takes page popularity into account, as well. Another major improvement can be obtained by combining the a-priori page rankings with propagation methods that exploit conditional patterns in user navigation.

## 6. CONCLUSIONS

In this paper, we presented $SUPRA$, a generic, real-time surfing prediction library that aims at facilitating users' revisitations in a uniform way for both server-side and client-side applications. To this end, it encompasses a set of prediction methods that are distinguished into three layers:

1. the *ranking layer*, which orders resources according to their likelihood of being accessed in the immediate future,
2. the *propagation layer*, which captures contextual information from navigation patterns and related resources, and
3. the *drift layer*, which exploits temporal aspects of revisits.

Our extensive experimental evaluation with two large, real-world datasets verified the benefits of combining methods of the different categories in order to effectively predict the next revisited page.

An additional goal of our work is to enable other researchers to experiment with our library, to extend it with novel methods and to integrate it into their applications. Thus, we have made the implementation publicly available through SourceForge. Having specified a set of minimal requirements for the ranking, propagation and drift methods, it is easy to plug into our framework any new algorithm that follows the formal definitions of Section 4. Moreover, we have released our real-world dataset, too, with the aim of offering to the research community a benchmark for comparing similar algorithms on an equal basis.

In the future, we plan to focus on enhancing the efficiency of our methods and to integrate our library into intelligent user interfaces. We actually intend to develop a browser tool that allows users to interact with the output of our methods, recommending a collection of URLs that are expected to be the next destination of the user. Special care will be taken to ensure the serendipity of the recommendations, so that they include not only the obvious resources, but also web pages that are usually overlooked between the head and the long tail. Users will also have the option to choose the prediction method that best fits their navigational behavior.

## 7. ACKNOWLEDGMENTS

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

## 8. REFERENCES

[1] E. Adar, J. Teevan, and S. T. Dumais. Large scale analysis of web revisitation patterns. In *CHI*, pages 1197–1206, 2008.

[2] G. Adomavicius and A. Tuzhilin. Using data mining methods to build customer profiles. *IEEE Computer*, 34(2):74–82, 2001.

[3] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216, 1993.

[4] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, pages 3–14, 1995.

[5] D. W. Albrecht, I. Zukerman, and A. E. Nicholson. Pre-sending documents on the www: A comparative study. In *IJCAI*, pages 1274–1279, 1999.

[6] M. Awad, L. Khan, and B. M. Thuraisingham. Predicting www surfing using multiple evidence combination. *VLDB J.*, 17(3):401–417, 2008.

[7] J. Brank, N. Milic-Frayling, A. Frayling, and G. Smyth. Predictive algorithms for browser support of habitual user activities on the web. In *Web Intelligence*, pages 629–635, 2005.

[8] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[9] P. Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–110, 2001.

[10] A. Cockburn and B. J. McKenzie. What do web users do? an empirical analysis of web use. *Int. J. Hum.-Comput. Stud.*, 54(6):903–922, 2001.

[11] M. Deshpande and G. Karypis. Selective markov models for predicting web page accesses. *ACM Trans. Internet Techn.*, 4(2):163–184, 2004.

[12] M. El-Sayed, C. Ruiz, and E. A. Rundensteiner. Fs-miner: efficient and incremental mining of frequent sequence patterns in web logs. In *WIDM*, pages 128–135, 2004.

[13] X. Fu, J. Budzik, and K. J. Hammond. Mining navigation history for recommendation. In *IUI*, pages 106–112, 2000.

[14] W. Gaul and L. Schmidt-Thieme. Mining generalized association rules for sequential and path data. In *ICDM*, pages 593–596, 2001.

[15] M. Géry and M. H. Haddad. Evaluation of web usage mining approaches for user's next request prediction. In *WIDM*, pages 74–81, 2003.

[16] D. Hawking, N. Craswell, P. Bailey, and K. Griffiths. Measuring search engine quality. *Inf. Retr.*, 4(1):33–59, 2001.

[17] E. Herder. Characterizations of user web revisit behavior. In *LWA*, pages 32–37, 2005.

[18] P. Kazienko. Mining indirect association rules for web recommendation. *Applied Mathematics and Computer Science*, 19(1):165–186, 2009.

[19] I. Koychev and I. Schwab. Adaptation to drifting user's interests. In *ECML Workshop: Machine Learning in New Information Age*, pages 39–46, 2000.

[20] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.

[21] H. Obendorf, H. Weinreich, E. Herder, and M. Mayer. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In *CHI*, pages 597–606, 2007.

[22] G. Papadakis, C. Niederee, and W. Nejdl. Decay-based ranking for social application content. In *WEBIST*, pages 276–282, 2010.

[23] A. G. Parameswaran, G. Koutrika, B. Bercovitz, and H. Garcia-Molina. Recsplorer: recommendation algorithms based on precedence mining. In *SIGMOD*, pages 87–98, 2010.

[24] J. Pei, J. Han, and W. Wang. Mining sequential patterns with constraints in large databases. In *CIKM*, pages 18–25, 2002.

[25] J. J. Sandvig, B. Mobasher, and R. Burke. Robustness of collaborative recommendation based on association rule mining. In *RecSys*, pages 105–112, 2007.

[26] L. Tauscher and S. Greenberg. 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.

[27] Y. Yao, L. Shi, and Z. Wang. A markov prediction model based on page hierarchical clustering. *Int. J. Distrib. Sen. Netw.*, 5(1):89–89, 2009.

[28] I. Zukerman, D. W. Albrecht, and A. E. Nicholson. Predicting users' requests on the www. In *UM*, pages 275–284, 1999.