Noname manuscript No.
(will be inserted by the editor)

Methods for Web Revisitation Prediction: Survey and Experimentation

George Papadakis · Ricardo Kawase · Eelco Herder · Wolfgang Nejdl

the date of receipt and acceptance should be inserted later

Abstract More than 45% of the pages that we visit on the Web are pages that we have visited before. Browsers support revisits with various tools, including bookmarks, history views and URL auto-completion. However, these tools only support revisits to a small number of frequently and recently visited pages. Several browser plugins and extensions have been proposed to better support the long tail of less frequently visited pages, using recommendation and prediction techniques. In this article, we present a systematic overview of revisitation prediction techniques, distinguishing them into two main types and several subtypes. We also explain how the individual prediction techniques can be combined into comprehensive revisitation workflows that achieve higher accuracy. We investigate the performance of the most important workflows and provide a statistical analysis of the factors that affect their predictive accuracy. Further, we provide an upper bound for the accuracy of revisitation prediction using an 'oracle' that discards non-revisited pages.

Keywords Web behavior · Navigation entropy · Revisitation prediction · Revisitation evaluation

G. Papadakis
Dept. of Informatics and Telecommunications, University of Athens
Panepistimiopolis, Ilissia, 15784 Athens, Greece
Tel.: +30 210-727-5207
Fax.: +30 210-727-5185
E-mail: gpapadis@di.uoa.gr

R. Kawase, E. Herder, W. Nejdl
Leibniz University of Hanover & L3S Research Center
Appelstr. 9a, 30167 Hanover, Germany
Tel.: +49 511 762-17713
Fax: +49 511 762-17779
E-mail: {kawase, herder, nejdl}@l3s.de

1 Introduction

A large portion of our activities on the Web involves Web sites and Web pages that we have visited before. We visit these pages for frequent tasks or routine behavior, such as following the latest news, communicating with friends and shopping online, but also for less frequent tasks or activities, such as planning the summer holidays or finding reference material for the annual tax declaration [1, 40, 51]. According to various studies, these frequent and less frequent activities account for 45.6% [40] to 81% [15] of all navigation on the Web. Browser features, such as bookmarks and URL auto-completion, and browser extensions, such as SmartFavourites [11], support the users' routine behavior and reoccurring tasks. They pro-actively identify and recommend the pages that are most likely to be revisited at a certain point in time, using revisitation prediction techniques that exploit recurrent navigation patterns.

In this article, we discuss and build upon existing prediction techniques for providing personalized revisitation support. We distinguish these techniques in two main types. The first type comprises *a-priori prediction techniques*, which estimate the probability of each page to be revisited in the next page request based on the recency and/or the frequency of its use. The second type encompasses *a-posterior prediction techniques*, which identify groups of pages that frequently co-occur within the same session. We elaborate on the characteristics of these main types and group the corresponding techniques into subtypes based on their common functionality. For example, the page associations captured by a-posterior techniques consider all evidence accumulated with the passage of time, even if they are outdated. A subtype of a-posterior prediction techniques, called *propagation drift methods*, takes into account the changes in user interests, tasks and routines by emphasizing recent activity in the prediction process.

A common aspect of the above-mentioned methods is that they rely on information that can be automatically extracted from the user's navigation history. This information includes:

- the sequence of page requests,
- the time they took place, and
- the sessions they belong to.

Another source of information about the user's navigational preferences involves details on the demographics, preferences, interests and background knowledge of the user [12]. These can be explicitly provided by the users themselves - for example by filling out a profile or by building a collection of bookmarks - or they can be derived from external sources, such as user activity in social media. However, evidence of this kind is rarely available in practice, as it either requires too much effort from the user, or might lead to privacy issues. For this reason, we limit our discussion to methods that exclusively exploit the user's navigation history.

Following the discussion on the different types of individual revisitation prediction techniques, we show how they can be aligned into comprehensive workflows, with each step targeting a different aspect of revisitation. The first step comprises *ranking methods*, which correspond to the a-priori prediction techniques. Ranking methods operate as scoring functions that associate each page with a numerical estimation of its probability to be re-accessed in the next page request. The second step encompasses *propagation methods*, which employ a-posterior prediction techniques to create weighted connections between pages that users revisit together. They determine the ranking of each Web page based on how often it is revisited together with the currently visited page - or based on a set of recently visited pages. The third step consists of propagation drift methods, which update the weights of page associations so that they reflect changes in the activities and interests of Web users. As our experimental results show, *revisitation workflows* that combine these three different types of revisitation techniques, achieve substantially higher predictive accuracy than the individual techniques.

To examine the actual performance of the individual methods and the revisitation workflows, we launched the Web History Repository initiative[1]. We managed to gather a large dataset of real data from individual users, who voluntarily contributed their navigation history, as recorded by their Web browsers. The collected data is completely anonymized, posing no threat to the privacy of the contributors and is freely available to any interested researcher, forming a valuable test-bed for the research community. Additionally, we experimented with another, smaller dataset that dates from 2005 [52]. A comparison of these two datasets, which stem from two different periods, provides some useful insights on how the evolution of the Web affects the navigational activity of its users.

In our experiments with both data collections, we measured the performance of the revisitation workflows based on their *effectiveness* and their *efficiency*. Effectiveness measures to what extent the revisited pages were covered in the top-N ranking positions, efficiency captures the computational cost in terms of the average time required for producing the list with the N best candidates for revisitation.

We also carried out a statistical analysis of our experimental results to obtain further insights into the main parameters that affect the performance of revisitation workflows. The outcomes show the high impact of the *entropy* in revisitation patterns: a low entropy indicates users that access a small number of Web pages on a regular basis, while a high entropy characterizes users who have a large set of pages that they revisit on an infrequent basis - the latter is inherently more difficult to predict (cf. Section 5.1.3). A second critical parameter is the composition of the *URL vocabulary*, the set of pages visited thus far. The continuously growing URL vocabulary can be divided into two subsets: the pages that have been or will be revisited by the user, and the set of pages that will never be revisited again. The latter set of *non-revisited pages* occupies the largest part of the URL vocabulary and should ideally be excluded from the prediction process in order to increase both effectiveness and efficiency. To quantify the effect of this phenomenon, we repeated our experimental evaluation using an *oracle*, an ideal classifier that is able to identify a-priori all pages that will never be revisited again after their first and only visit. In this way, we empirically estimated an upper bound for the performance of the main revisitation workflows and assessed how closely they approach it.

We conclude our experimental analysis with a discussion about the best revisitation workflow for each application, depending on its requirements and its available resources. We also explain why it is difficult to a-priori predict the best workflow for every user. Instead, we demonstrate that the workflow with the best average accuracy across both datasets achieves near-optimal performance for most users. Thus, it is suitable for applications that require high accuracy and can afford

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

a higher computational cost. Finally, we discuss the feasibility of approximating the functionality of the oracle.

To summarize, the contributions of this article are the following:

- We provide a systematic overview of revisitation prediction techniques, dividing them into a-priori and a-posterior techniques. Together with the propagation drift methods, they form three complementary types of techniques that can be combined into comprehensive revisitation workflows of high predictive accuracy.
- We present the results of an experimental study that examines the effectiveness and the efficiency of 25 revisitation workflows over two large, real-world datasets that in total comprise more than 2.6 million page requests from 205 distinct users. As the data collections differ in age by 7 years, the results of our analysis provide insights into the impact of the evolution of the Web on revisitation patterns.
- We perform a statistical analysis that identifies the parameters with the largest impact on the performance of a revisitation workflow. Apart from differences in entropy between individual users, the large set of non-revisited pages also affects effectiveness and efficiency to a significant extent.
- To estimate an upper bound of each workflow's performance, we cancel out the impact of non-revisited pages by using an oracle that identifies them with 100% accuracy.

The remainder of the paper is structured as follows. In Section 2, we discuss related work, which consists of studies that examine revisitation on the Web and tools that aim to facilitate it. In Section 3, we give a formal definition of the task of revisitation prediction and the data associated with it. In Section 4, we discuss the main types of revisitation prediction methods and show how they can be combined into comprehensive workflows. Section 5 presents the experimental results and statistical analysis of the prediction methods, while Section 6 concludes the paper and provides directions for future work.

2 Related Work

In this section, we elaborate on two major lines of research in revisitation: studies that investigate revisitation patterns in the navigational activity of Web users (Section 2.1) and tools that exploit these patterns in order to predict and facilitate revisitation (Section 2.2). Related work on revisitation prediction methods is discussed in Section 4, for easier comparison with methods that use similar techniques.

2.1 Revisitation Studies

One of the first studies on Web usage behavior was performed by Tauscher and Greenberg [49]. They quantified to what extent Web users carry out recurrent tasks on the Web and confirmed Catledge and Pitkow's [13] finding that following hyperlinks and clicking the back button are the most frequently used methods for re-accessing a Web page. In contrast, the temporally ordered history list is rarely used. They also coined the term *recurrence rate*, which expresses the probability that any page visit is a repeat of a previous visit. According to their estimations, the average recurrence rate for their participants amounted to 58%, while the reanalysis of the data from the Catledge and Pitkow study yielded a recurrence rate of 61%.

The same study also demonstrated that the URL vocabulary grows linearly with the number of page requests. Two important characteristics of revisited pages were also described: first, the probability for a page to be revisited decreases steeply with the number of page requests since the last visit to it; this implies that most page revisits involve pages that a user visited very *recently*. Second, the probability for a page to be revisited decreases steeply with its popularity ranking. As a result, there is a small number of highly *popular* pages that are visited very frequently.

Another long-term click-through study was carried out by Cockburn and McKenzie [15]. They observed that browsing is a *rapidly interactive* activity: the most common time interval between subsequent page visits is around 1 second, while time intervals of more than 10 seconds are rather scarce. They also analyzed bookmark collections, revealing that most users have or will have problems with their organization, due to their constantly increasing size.

More recently, Weinreich et al. [52] carried out a long-term study, in which they analyzed the interactions of 25 users with their Web browser during a period of four months and compared the results with the studies discussed above. They demonstrated that the introduction of new browser features had a dramatic impact on the way users navigate the Web. For example, tabbed browsing has been established as a useful alternative for hub-and-spoke navigation that replaces backtracking to a significant extent. Another major factor is the evolution of the Web from a repository of rather static hypermedia documents to a platform focused on interaction and transactions.

Based on user action logs and interviews, Obendorf et al. [40] distinguished revisits into those occurring within an hour (*short-term*), within a day (*medium-term*) and within a week or longer (*long-term*). Short-term revisits were observed to be primarily initiated through the back button and to involve portal pages and other navigational pages. Medium-term revisits mainly refers to pages that users visit on a regular basis, such as the portal page of search engines, news sites, shopping sites or reference sites. Browser tools that were commonly used for medium-term revisits are bookmarks and the URL auto-completion. For long-term revisits, browser support was of little use and users often had to repeat the search for the required page, or retrace their previous trails to the page.

Adar et al. [1] further investigated revisitation behavior, making use of a large user base that was collected via the Windows Live Toolbar. They found that short-term revisits are related to hub-and-spoke navigation while visiting reference or shopping sites or pages on which information is monitored. Medium-term revisits correspond to Web mail, forums, educational pages as well as browser home-pages. Long-term revisits involve pages found via search engines and usually pertain to weekend activities, such as going to the cinema. A subsequent study was carried out by Tyler and Teevan [51], based on a merged dataset of search engine logs, Web browser logs and a large-scale Web crawl that comprised several millions of users. The results confirmed earlier findings, advocating that within-session revisits typically aim at continuing work on a task or a routine behavior. In contrast, revisits that occur across different sessions (*cross-session revisits*) mainly involve re-evaluation (e.g., “Did I remember the information correctly?”, “Did something change?” or “Has something new been added?”).

The above observations were confirmed by Kumar et al. [32], who used a random sample of users drawn from Yahoo! toolbar logs to compare pageview categories for two kinds of revisits: *regular* revisits, which take place within a week, and *long-term* revisits, which occur within a week, but have not been repeated during the last 24 hours. They found that about 50% of Web navigation involves sites delivering content (news, multimedia, vertical content, games); around 35% of Web activity is dedicated to communication, mainly through social networks and email, and search accounts for 9% of Web navigation. Page revisits are rarely achieved with just one query; instead, after a query, users typically have to follow a trail of pages to reach the desired location.

2.2 Revisitation Support

Web browsers incorporate a versatile set of history mechanisms, including bookmarks, the URL auto-completion, the forward and back buttons and the history sidebar [35]. The use of multiple tabs can be considered as an implicit history mechanism, as well. However, as discussed in the previous section, the support offered by these tools is typically skewed toward a small set of frequently visited resources and therefore suboptimal [40].

For this reason, browsers like Mozilla Firefox[2] and Google Chrome[3] allow users to install extensions that offer improved and more dynamic revisitation support. Notable add-ons that are relevant to our work include – in no particular order – Delicious[4] (social bookmarking), Infoaxe[5], Hooeey (full-text history search), WebMynd[6] (history sidebar for search) and ThumbStrips (history visualization). In addition, many search engines currently offer personalized search, which greatly facilitates refinding [28].

Academic research has also delivered several alternative history mechanisms, including gesture navigation [15], dynamic bookmarks [48], a SmartBack button that recognizes waypoints [36], a browsable SearchBar organized around a hierarchy of past queries [39] and many types of history visualizations: lists, hierarchies, trees, graphs, 2d and 3d stacks and footprints. A comprehensive overview of these tools is provided in [35].

However, none of these approaches seems to consider all regularities that have been discovered by the aforementioned revisitation studies. Due to the power-law distribution of revisitations, their recommendations typically exhibit high levels of stability, biased towards popular and/or frequently accessed pages. The evolving traits of user’s navigational activity, though, introduce a clear trade-off between the stability and the usefulness of revisitation recommendations: the more stable the predictions offered by a revisitation support tool, the more likely it is that a Web user will anticipate its recommendations and make use of them; as a consequence,

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

[3] https://www.google.com/intl/en/chrome/browser

[4] http://www.delicious.com

[5] http://infoaxe.com

[6] http://www.webmynd.com

though, this results in a limited coverage of the visited pages, thus facilitating only the most common situations and needs. In contrast, dynamic recommendations can achieve a better balance between recency and frequency, and may exploit typical navigation patterns to provide recommendations for particular situations.

In an earlier study, we demonstrated the benefits of dynamic recommendations through the evaluation of a dynamic browser toolbar, called PivotBar [29]. The PivotBar resembles the common bookmark toolbar, containing favicons and links to the 10 pages that were most likely to be revisited. Unlike static toolbars, the content of the toolbar changed with each page visit and provided personalized suggestions specifically related to the currently visited page. The recommendations were generated using the best workflow of our framework (see Section 5.2.1). All computations took place on the client-side, exploiting the user history that is recorded in the browser's database.

The take-up and usability of the PivotBar was evaluated with 13 participants during a period of 10 days. On average, 12.1% (σ=7.3) of all revisits were initiated through the PivotBar. We also measured the number of '*blind hits*', situations in which the revisited page was displayed in the toolbar, but the user employed a different method for accessing the page. The average percentage of blind hits was 18.1% and the differences in blind hits between each user was strongly correlated with the individual user's take-up of the tool ($r = 0.92, p < 0.01$). This confirms that good recommendations are essential for the take-up of revisitation support tools.

In Section 4, we present a structured overview of revisitation prediction techniques that exploit most of the identified patterns. These techniques typically lie at the core of tools that provide dynamic short-cuts, page recommendations or other kinds of revisitation support.

3 Data Model & Problem Definition

Revisitation is the act of accessing a Web resource that has already been visited in the past. As explained above, this is a common practice among Web users, accounting for at least 45% of the overall Web activity [40] and up to 80% of the mobile Web activity [34]. Therefore, user experience can be significantly enhanced by supporting these recurrent activities. The task of predicting the subsequently revisited page is called *revisitation prediction.*

Any kind of evidence can be involved in this task. Potentially useful information ranges from content-based data, like the URL and the content of the accessed pages, to demographic information, such as the gender and the age of the user. To ensure the generality of our work, we do not rely on external information, such as the meta-data about Web users or their navigational activities. Instead, we consider methods that exclusively rely on the information that is inherent to the users' navigation on the Web, receiving as input the chronologically ordered set of visited pages, grouped into sessions.

More formally, the *navigation history* of a user is represented by $\mathbf{R} = \{r_1, r_2, \ldots, r_n\}$, where r_i is an individual page request. The page request r_i returns the page p_j that was retrieved during this visit, the time t_i the request took place as well as the corresponding serial number i, which indicates its relative position in the overall activity of the user – regardless of the corresponding session (cf. Definition 3); that is, the serial number of the chronologically first request is 1 and is incremented by 1 for each subsequent page visit. The set of all individual Web pages visited during $\mathbf{R}$ (the *URL vocabulary*) is denoted by $\mathbf{P} = \{p_1, p_2, \dots, p_m\}$.

Given that revisitation prediction techniques typically operate on the level of individual Web pages, they (re-)organize the information contained in $\mathbf{R}$ into navigational data per page. There are two structures that actually lie at the core of their functionality:

Definition 1 Given all page requests $\mathbf{R}$ of a user, the **request indices** of a page p_i ($\mathbf{I_{p_i}}$) is the set of the serial numbers of those requests in $\mathbf{R}$ that pertain to p_i.

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

The request indices are essential for revisitation prediction techniques that interpret users' navigational activity as a series of events, disregarding the time they actually took place. In contrast, time-based approaches rely exclusively on request timestamps. In the middle of these two extremes lie hybrid predictive methods, which incorporate both types of evidence into their functionality.

Another important aspect of a user's navigational activity is the notion of *session*, i.e., the collection of pages visited by a user in sequence during a specific period of time. There are two ways for defining a session: either on-line or off-line. In the former case, sessions are internally defined by browsers. For instance, Mozilla Firefox starts a new session when the user opens a new empty tab, a new empty window or after a certain time of inactivity; following links into a new window or a new tab is still considered part of the original session. The off-line identification of sessions is typically used for server-log and client-side analysis. The only information that is available in this context is the sequence of page requests and the time they took place. Therefore, sessions are usually defined as a continuous period of browsing. A common heuristic for identifying the start of a new session is a timeout of 25.5 minutes [13, 26, 40, 49]; a new session starts after the user remains inactive for 25.5 minutes. Following this practice, we define sessions as follows:

Definition 3 A **session** is a bag $\mathbf{S} = \{p_1, p_2, \dots, p_k\}$ of all pages visited by a user that follows links during a specific time period, placed in chronological order, from the earliest visit to the latest visit.

Note, though, that this definition does not distinguish between task-related and task-unrelated tabs and/or windows. All page requests falling in the same time frame are assigned to the same session without any exceptions.

Based on the above definitions, the problem that we address in this paper can be formally defined as follows:

Problem 1 (Revisitation Prediction) *Given the set of Web pages* $\mathbf{P}$ *that have been accessed during the past page requests of a user (*$\mathbf{R}$*), order them in such a way that the ranking position of the page* $p_i \in \mathbf{P}$ *that she will* re-access *in the next revisitation is the highest possible.*

The recommendations for the next revisited page can be presented to the user through any revisitation support tool, such as URL auto-completion. However, interface issues lie out of the scope of our work. Instead, we exclusively focus on the functionality of the predictive mechanisms that lie at the core of such tools. Note that these mechanisms do not aim at recommending not-visited, yet relevant Web pages; they merely facilitate the access to pages already viewed in the past.

4 Revisitation Prediction Methods

In this section, we give an overview of common and current methods for revisitation prediction on the Web. We separate them into two main categories: *a-priori prediction methods*, which rank pages based on the overall probability that they will be revisited, and *a-posterior prediction methods*, which rank pages based on the probability that they will be revisited in the current user context. The a-priori methods make use of evidence on how often and when pages have been visited, the a-posterior methods take into account how often pages are accessed together with the currently visited page or set of pages. In addition to these two main categories, we discuss so-called *propagation drift methods*, which aim to improve how propagation methods take changes in user habits and interests into account. As explained in the previous section, we only consider methods that exploit the users' navigation history and do not require any other external evidence.

Further, we analyze the complexity of the methods and discuss relevant literature on how they have been used in practice. Finally, we explain how the individual revisitation methods can be combined into revisitation workflows and briefly introduce the publicly available framework SUPRA, which implements all techniques discussed in this chapter.

4.1 A-priori Prediction Methods

Methods of this type aim to estimate the overall, prior probability that an already visited Web page will be re-accessed in the next page request. This overall probability is then used for ordering Web pages in such a way that higher ranks are assigned to pages that are more likely to be revisited. Therefore, we call the probability estimations *ranking scores* and the methods used to produce these estimations *ranking methods*.

Most of these ranking methods exploit the observation from Catledge and Pitkow [13] that revisits typically involve frequently and/or recently visited pages (see Section 2). Three types of ranking methods can be distinguished, based on the way the navigation history is modeled: *event-based* methods represent the users' history by the request indices (Definition 1), ignoring the time elapsed between any two requests; *time-based* methods make use of the exact timestamps of previous page visits (Definition 2); finally, there are *hybrid* methods that exploit a combination of both approaches.

4.1.1 Event-based Ranking Methods

These techniques interpret the navigational history of a user as a sequence of events. They rank the visited pages based on the number of times that they have been visited and/or the number of events (page requests) between the visits to these pages and the latest recorded event. This means that they do not exploit the actual time of the visits, only the request indices of a page in the navigation history. More formally:

Definition 4 An **event-based ranking method** is a function that takes as input a page $p_i \in \mathbf{P}$, its request indices $\mathbf{I_{p_i}} = \{i_1, i_2, \ldots, i_k\}$ together with the index n of the latest request r_n and produces as output a value $v_{p_i} \in [0,1]$[7] that is proportional to the probability of p_i being accessed at the next page request, r_{n+1} (the closer v_{p_i} is to 1, the higher is this probability).

The simplest methods of this category rely exclusively either on the recency or the frequency of page visits. The most common method that exploits recency is called *last recently used* (LRU). It simply orders visited pages in chronological order, with the top ranking given to the latest visited page. More formally, the ranking score assigned to p_i is derived from the following formula:

$$LRU(p_i, \mathbf{I_{p_i}}, n) = \frac{1}{n + 1 - max(\mathbf{I_{p_i}})}.$$

By contrast, the *most frequently used* (MFU) method exclusively considers the popularity, or rather the frequency of use, of a page. Essentially, it sorts the visited pages by the number of request indices (the number of visits to the page), in descending order. More formally:

$$MFU(p_i, \mathbf{I_{p_i}}, n) = \frac{n}{|\mathbf{I_{p_i}}|}.$$

More elaborate techniques combine the recency and frequency of use in their assessments. Most of these methods can be expressed as a *decay ranking model*, which was originally presented in [42]. This model ranks each Web page p_i after n requests based on the number of request indices, applying a decay function to give lower influence to page requests from the more distant past. The generic formula for decay ranking models is as follows:

$$DEC(p_i, \mathbf{I_{p_i}}, n) = \sum_{j \in \mathbf{I_{p_i}}} d(j, n), \tag{1}$$

where $d(j,n)$ is a *decay function* that takes as an input the index j of a request to p_i, together with the index of the current transaction n, and gives as output the contribution of this request to the total score of p_i. Every valid decay function should satisfy the following properties [16, 42]:

1. $\forall\ j \leq n$: $0 \leq d(j,n) \leq 1$,
2. $d(j,n) = 1 \rightarrow j = n$,
3. $\forall\ n' \geq n$: $d(j,n') \leq d(j,n)$ (a monotone non-increasing function).

There are three main families of decay functions that satisfy these properties: *polynomial* (PD), *exponential* (ED) and *logarithmic* (LD) decay. Together with LRU and MFU, they were all experimentally evaluated in [42]. The outcomes of the study showed that PD consistently outperforms all other event-based ranking methods, because it involves a smooth decay that harmonically balances the recency and

[7] As explained in Section 4.1.4, this is ensured through normalization.

the frequency of page visits. In contrast, ED favors recency against frequency, with a steep decay that is practically equivalent to LRU; the slow decay of LD favors frequency against recency to such an extent that it approximates MFU. For this reason, in the remainder of this article we exclusively consider PD for decay functions. For PD, the decay function $d(j,n)$ in Formula 1 takes the following form:

$$d(j,n) = \frac{1}{1+(n-j)^{\alpha}}, \tag{2}$$

where α denotes the *decay rate* of the polynomial decay function. Rates larger than 1 convey a steep decay, which puts more emphasis on recency, while rates close to 0 promote the frequency of use, with $d(j,n) \approx \frac{1}{2}$ for $\alpha \approx 0$. In the latter case, all page requests have a similar contribution to the ranking score of a page, regardless of their recency, and therefore the resulting ranking will be the same as for MFU.

4.1.2 Time-based Ranking Methods

Similar to event-based ranking methods, time-based methods take into account the frequency and/or recency of page visits. The difference is that time-based ranking methods rely on the request timestamps of a page and derive the ranking scores exclusively from them. Thus, the contribution of a page request r_i to the ranking score of a page p_j depends on the time it took place (t_i) and the time difference between t_i and the latest page request (t_n). More formally:

Definition 5 A **time-based ranking method** is a function that takes as input a page $p_i \in \mathbf{P}$ visited in $\mathbf{R}$, its request timestamps $\mathbf{T_{p_i}} = \{t_1, t_2, \ldots, t_k\}$, together with the time t_n of the latest request r_n and produces as output a value $v_{p_i} \in [0,1]$[8] that is proportional to the probability of p_i being accessed at the next page request, r_{n+1}.

Similar to event-based methods, time-based methods can be expressed as decay ranking models, with *polynomial decay* (PD) as the most applicable implementation.

Both event-based and time-based methods treat the user's navigation history as a continuous sequence. The difference is that, in contrast to event-based methods, time-based methods implicitly group user navigation into *sessions* (see Definition 3): due to the elapsed time during a period of inactivity between sessions, pages from earlier sessions receive a considerably lower ranking score than pages from the current session. For this reason, time-based methods put slightly more emphasis on within-session revisits, which typically aim at continuing work on a task. As a drawback, the time-based methods are sensitive to long periods of inactivity, such as weekend breaks.

4.1.3 Hybrid Ranking Methods

As explained in the previous section, event-based and time-based methods each have their advantages and disadvantages. In practice, many ranking methods are a *hybrid* of these two kinds, as they make use of both the request indices and the timestamps of the page visits. This allows for methods that exploit both the

[8] As explained in Section 4.1.4, this is ensured through normalization.

continuous sequence of page visits (as represented by the request indices) and the actual timestamp of the visits. The request index lends itself better for polynomial decay methods, but the timestamp can be exploited for capturing temporal patterns - apart from separating page visits into sessions, this allows for optimizing recommendations for pages that are usually visited at a particular time (such as checking news in the morning) or on a particular day (for example planning weekend activities).

More formally, hybrid ranking methods are defined as follows:

Definition 6 A **hybrid ranking method** is a function that takes as input a page $p_i \in \mathbf{P}$ visited in $\mathbf{R}$, its request indices $\mathbf{I_{p_i}}$ and its request timestamps $\mathbf{T_{p_i}}$ along with the time t_n and the index n of the latest page request, r_n. As output, it produces a value $v_{p_i} \in [0, 1]$[9] that is proportional to the probability of p_i being accessed at the next page request, r_{n+1}.

Frecency (FR), the default mechanism of Mozilla Firefox for page recommendations, is a hybrid method that uses a rather simple and intuitive time-based mechanism for ranking pages based on their recency and frequency of use. All page visits are placed in buckets according to their recency (less than 4, 14, 31 or 90 days, or older than 90 days). Visits in more recent buckets receive a higher weight. The ranking score for each page is then calculated as the sum of weights of all visits. FR is also partially event-based, as the ten most recent visits receive a weight according to how the visit was initiated (e.g., by following a link, typing a URL or clicking on a bookmark)[10].

The weighting of FR is based on natural time intervals, but the method does not exploit differences in user interests on different times of day, or days of the week - such as reading the news in the morning or planning a weekend trip [1, 31]. A straightforward and simple way to achieve this is to adapt MFU to only take visits from a particular time window into account. This may be achieved by separating each day in four buckets (morning, afternoon, evening, night) or to have two separate buckets for weekdays and the weekend. Even finer-grained buckets can be considered. However, this approach exclusively considers page visits from the current bucket, ignoring a large amount of pages that are visited regardless of the time of day (such as reference sites or search engines).

In this work, we consider a number of variations of *polynomial decay* (PD) that boosts page visits corresponding to the relevant time window by reducing the decay rate by 50%. The *hybrid day model* (HDM) boosts past page visits that took place on the same day of the week. The *hybrid hour model* (HHM) separates each day into buckets of one hour each and boosts the current hour. The *hybrid quarter model* (HQM) separates each day into four parts: morning (6am to 12pm), noon (12pm to 6pm), evening (6pm to 12am) and night (12am to 6am). All models have in common that they deliver higher ranking scores to pages that are visited on the same day of week or on the same part of day, but consider page visits from other buckets as well (α remains unmodified).

[9] As explained in Section 4.1.4, this is ensured through normalization.

[10] For more details, see `https://developer.mozilla.org/en/The_Places_frecency_algorithm`.

4.1.4 Complexity of Ranking Methods

As explained above, most of the ranking methods process and retain the entire navigation history of all Web pages in order to estimate their ranking scores. Thus, their overall space complexity is equal to $O(|\mathbf{P}|+|\mathbf{R}|)$. Their time requirements are determined by two procedures: *(i)* the process of iterating over all page requests in order to estimate the ranking scores, and *(ii)* the normalization process that iterates over all visited pages in order to restrict their ranking scores to the interval $[0, 1]$, by dividing it by the highest score. Therefore, their overall time complexity is equal to $O(|\mathbf{P}| + |\mathbf{R}|)$.

The only exceptions to these rules are LRU and MFU. They merely need to record a single counter for every visited page, thus having an overall space complexity of $O(|\mathbf{P}|)$. Similarly, their time complexity is significantly lower than the other ranking methods, as they need to iterate over the visited pages just once, i.e., $O(|\mathbf{P}|)$. In fact, the number of computations saved by LRU and MFU is equal to $|\mathbf{R}|-|\mathbf{P}|$. This means that the lower the ratio $|\mathbf{P}|/|\mathbf{R}|$, the fewer pages are never visited and the more computations are saved by LRU and MFU.

4.1.5 Applications of Ranking Methods

In the above sections, we elaborated on the basic functionality of ranking methods. In practice, tools that support revisitation (not only in the context of the Web) modify the functionality of ranking methods in order to adapt them to the application at hand. We already discussed the Frecency method, which is integrated into the Firefox browser and uses tailored buckets and corresponding weights. Another example is the CRF approach (CRF stands for Combined Recency and Frequency), a variation of the exponential decay ranking method that is crafted for cache management [33].

The Adaptive algorithm, which filters menu items in software like MS Office 2000, is basically an implementation of MFU that also takes into account recent periods, during which a menu item has not been used [8]. *SR&F* is a hybrid form of LRU and MFU, as it ranks the first n items with the former and the rest with the latter [21]. In [11], the authors propose a scoring mechanism based on a decay model that combines frequency with recency in order to predict the start of a new task - or navigation session.

Recently, the AccessRank algorithm was proposed in [22] for predicting revisitation of Web domains as well as window switching in desktop environments. AccessRank combines CRF with a custom weighting scheme that is based on the same rationale as our hybrid ranking methods: it promotes the contribution of past page requests that occurred on the same time of day or on the same day of the week as the latest page request. AccessRank also takes special care of the stability of the recommended pages, so that users can anticipate the ranking of the most important items - as explained in Section 2.2, this leads to improved support for common activities, though at the cost of a limited coverage of visited pages.

4.2 A-posterior Prediction Methods

The ranking methods, as discussed above, exploit in various ways that revisits focus on a small number of frequently or recently visited pages and do not take the users' current context into account. From the revisitation studies discussed in Section 2.1, we know that particularly long-term revisits involve reoccurring tasks or routine behavior, which often consists of trails of related pages that are revisited together [11, 22].

A-posterior prediction methods aim to identify groups of pages that typically co-occur within the same session - but not necessarily in the same order. Predictions are based on the 'evidence' formed by (a selection of) the pages already visited during the current session. As these methods propagate the probability of revisitation between co-occurring pages, we call them *propagation methods.* We formally define them as follows:

Definition 7 A **propagation method** consists of two functions: *(i)* the *update function*, which takes as input a session $\mathbf{S_i}$ and adjusts the degree of connection between its last visited page p_n and all other pages accessed during $\mathbf{S_i}$, and *(ii)* the *association function*, which receives as input a pair of visited pages, p_k and p_l, and produces as output a value $v_{kl} \in [0, 1]$ that is proportional to the probability of p_l being accessed immediately after p_k (the closer v_{kl} is to 1, the more likely the transition $p_k \rightarrow p_l$ is).

Essentially, all propagation methods maintain a data structure that captures the chronological patterns in the navigational activity of users. The most comprehensive representation of this data structure is a two-dimensional matrix, called *propagation matrix* (M). Its rows and its columns correspond to the Web pages that have been visited by the user so far ($\mathbf{P}$). Each cell in M stores the weight of the link between the corresponding pages - except for the diagonal cells, which are all set to 0, i.e., $\forall p_i \in \mathbf{P}$: $M(i,i) = 0$ (the reason for this convention is analyzed in Section 4.3). Given the propagation matrix M, the association function of Definition 7 can be seen as a function that simply returns the weight of the cell $M(i, j)$, when receiving the pair of pages p_i and p_j as input.

The individual propagation methods differ only in their update functions, which specify the weights that are stored in M. We distinguish the propagation methods into two subtypes, depending on the way they model the activity of a session: *order-preserving* methods take the order of requests into account, assuming that pages are typically revisited in the same order; *order-neutral* methods follow the assumption that revisits can occur in any order. We will discuss the corresponding update functions in the remainder of this section.

4.2.1 Order-Preserving Propagation Methods

The assumption behind order-preserving propagation methods is that when people revisit a set of pages for a particular purpose, they usually visit them in - more or less - the same order, which is inherent to the task that they carry out. This behavior can be modeled by a *transition matrix* (TM): for each pair of pages p_i and p_j, the corresponding value cell $TM(i, j)$ is proportional to the number of times that p_i has been visited *earlier than* (but not necessarily directly before) p_j

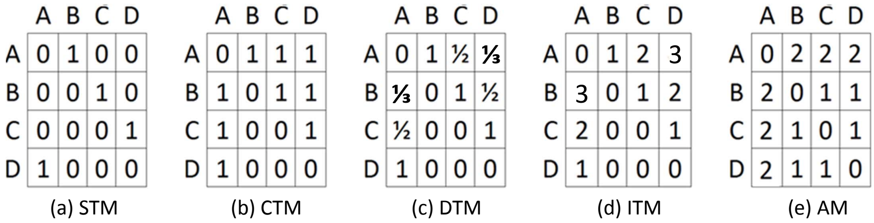
within the same session - and vice versa for $TM(j,i)$. As a consequence, TM is a non-symmetric matrix.

The difference between the various order-preserving propagation methods lies in the selection of the pages to take into account (varying from all pages visited in the session thus far to only the previous page) and how each page is weighted. We now illustrate the functionality of several variations of TM with a walk-through example that consists of 4 Web pages (A, B, C, D) and the session $\mathbf{S_1} : A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$. Based on how the weights of page associations are defined, we identify the following 4 types of TM:

1. The *simple transition matrix* (STM) is based on the assumption that page revisits tend to occur in the same strict order. Hence, it works as a first-order Markov model, incrementing the value of $TM(i,j)$ by one for each transition $p_i \rightarrow p_j$. As an example, consider Figure 1(a), which depicts the values of STM after the last transition of $\mathbf{S_1}$.
2. The *continuous transition matrix* (CTM) supports page revisits that take place in a similar order, but not necessarily in exactly the same sequence (e.g., $p_i \rightarrow p_j \rightarrow p_k$ and $p_i \rightarrow p_k$). Hence, it associates each Web page of a given session $\mathbf{S_i}$ with all the previously accessed ones. In our example in Figure 1(b), transition $D \rightarrow A$ causes A to be associated with all Web pages previously visited in the session, incrementing the corresponding cells by one.
3. The *decreasing transition matrix* (DTM) lies in the middle of STM and CTM, supporting evenly requests that occur in the same or in similar order. DTM associates each Web page p_j of a given session $\mathbf{S_i}$ with every page p_k that was previously accessed, but increments the value of $TM(j,k)$ with a decay parameter that is inversely proportional to the distance between them (the number of page requests between them). Continuing our example, Figure 1(c) depicts the entire DTM after the transition $D \rightarrow A$. Note that $TM(C,A)$ is incremented by 1/2 after $D \rightarrow A$, as C is two steps away from A in $\mathbf{S_1}$.
4. The *increasing transition matrix* (ITM) is the inverted version of DTM. Given a session $\mathbf{S_i}$, it creates stronger connections between pages that are more distant in $\mathbf{S_i}$, in an effort to identify the final destination of $\mathbf{S_i}$. By boosting the ranking score of the final page early enough, ITM tries to reduce the length of the trail of possibly irrelevant pages that the user visits before reaching its actual page of interest - as discussed in Section 2.1, users often have to search again for pages that were visited in the more distant past [40]. Hence, the value added to $TM(j,k)$ increases proportionally to the distance between pages p_j and p_k. The propagation matrix corresponding to session $\mathbf{S_1}$ is shown in Figure 1(d).

4.2.2 Order-Neutral Propagation Methods

In contrast to the order-preserving propagation methods, order-neutral methods assume that sets of pages that are regularly revisited together may not be revisited in the same order. For example, consider a returning task, such as planning the summer holidays: it may not matter whether one first books the flight and then the hotel, or the other way round. Therefore, pages that are visited in the course of a session $\mathbf{S_i}$ should be equally connected with each other, regardless of their relative position in $\mathbf{S_i}$ and the number of transitions between them.



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

(a) STM

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

(b) CTM

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & \frac{1}{2} & \frac{1}{3} \\ B & \frac{1}{3} & 0 & 1 & \frac{1}{2} \\ C & \frac{1}{2} & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 \end{array}$$

(c) DTM

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 2 & 3 \\ B & 3 & 0 & 1 & 2 \\ C & 2 & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 \end{array}$$

(d) ITM

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

(e) AM

Fig. 1 The values of several types of propagation matrices after the last page request of the session $\mathbf{S_1} : A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$.

The assumption of order-neutrality is modeled as an *association matrix* (AM), which is built by associating all pages visited in a single session with each other. This means that every Web page is connected not only with the pages preceding it, but also with those following it. As a result, AM is a symmetric matrix: $\forall p_i, p_j \in \mathbf{P} : AM(i,j) = AM(j,i)$. Continuing our example, Figure 1(e) depicts the AM corresponding to the session $\mathbf{S_1}$.

4.2.3 Propagation Drift Methods

The habits and interests of users change constantly over time, a phenomenon that is called *concept drift* [31]. This trait is supported by the ranking methods that consider recency: by gradually decaying the contribution of past page visits, they adapt to changes in the navigational patterns of the Web user. Propagation methods support concept drift only to a limited extent: the matrices continue to learn about (changes in) the user's interests and habits, but this will happen only slowly - in particular for users with a large navigation history. The reason is that the contribution of the most recent transitions is equal to that of the oldest ones. In order to better support sudden changes, we need techniques that make propagation matrices more dynamic and adaptive.

These techniques are called *propagation drift methods* and can be distinguished in two major subtypes: *decay-based methods*, which gradually reduce the contribution of past page transitions (similar to the decay ranking methods), and *window-based methods*, which use a sliding window of a particular size to discard older transitions [31]. Decay-based methods have the advantage that older transitions - which still might be useful at a later point - are not discarded; therefore, decay-based propagation methods are probably more precise than the window-based ones. However, the decay-based methods have a space and time complexity that is quadratic to the number of visited pages ($O(|\mathbf{P}|^2)$), as their decay function should be applied to all transitions between every pair of pages. For this reason, we limit our discussion to window-based drift methods.

The functionality of window-based methods is very basic: they simply identify the requests that fall outside the sliding window and remove their contribution to the weights in the propagation matrix. In this way, they add nothing to the space requirements of the propagation matrix, while their time complexity is linear with the number of page visits. In addition, window-based methods enhance the computational efficiency of propagation methods, as they reduce the number of (non-zero) page associations and, consequently, the number of calculations needed for the propagation of ranking scores.

We will consider two main types of window-based drift methods. First, *event-based* methods define the window size in terms of the number of retained requests. We define them formally as follows:

Definition 8 Given the page requests $\mathbf{R}$ of a user and a propagation matrix M, an **event-based drift method** with a window of size w updates the weights of the page connections stored in M so that they reflect the latest w page requests in $\mathbf{R}$.

In this paper, we will evaluate two event-based drift methods: *500-requests* (5HR) and *1000-requests* (TR). As their names suggest, they take into account the 500 and the 1,000 latest page transitions in $\mathbf{R}$.

Time-based drift methods base the window size on an actual time span. Therefore, they are more suitable for supporting tasks or activities that reoccur on a regular basis (for example hourly, daily, weekly and monthly) and not after a certain amount of page visits. We formally define them as follows:

Definition 9 Given the page requests $\mathbf{R}$ of a user and a propagation matrix M, a **time-based drift method** with a window of size t updates the weights of the page connections stored in M so that they reflect the page transitions of $\mathbf{R}$ that occurred within the latest t temporal units[11].

In this paper, we consider *day* (DM), *week* (WM) and *month drift methods* (MM). They update the underlying propagation matrix so that it maintains the transitions that took place within the latest day, week or month. These ‘natural time intervals’ are in line with the distinction between short-term, medium-term and long-term revisits in [40].

4.2.4 Complexity of Propagation Methods

The space requirements of the propagation methods are determined by the size of the propagation matrix M, in which the weights of page associations are stored. Given that each dimension corresponds to all pages visited so far, its complexity is equal to $O(|\mathbf{P}|^2)$. The time complexity of the association function is constant (i.e., $O(1)$), as it merely returns the value of a cell in M. For the update functions, the time complexity is equal to $O(|\mathbf{S_i}|)$, as in the worst case, they have to update the weight of the latest page in $\mathbf{S_i}$ with all others (the same applies to the propagation drift methods). The only exception to this rule is STM, in which updates have a constant time complexity, modifying the value of a single cell (i.e., $O(1)$).

After every page request, the propagation method calls the association function for every pair of visited page in order to retrieve the corresponding weights. Theoretically, this is a quadratic process (i.e., $O(|\mathbf{P}|^2)$) - regardless of the update function that produced M. In practice, though, its computational cost can be significantly restricted by using graphs for the implementation of propagation matrices. In this case, it suffices to call the association function only for the edges of the graph, which indicate the pages that co-occur within the same session; for all other pairs of pages, their degree of association is zero. Given the high sparsity of the resulting graphs, the efficiency of propagation methods is significantly

[11] A temporal unit is measured in milliseconds and expresses any time interval, ranging from seconds, minutes and hours to days, weeks and months.

enhanced. The actual degree of sparsity, though, depends on the underlying propagation method. Figure 1 demonstrates that, for the same session, STM yields the most sparse propagation matrix, while AM creates the largest number of links. All other methods lie between these two extremes, producing graphs of the same size.

4.2.5 Applications of Propagation Methods

Propagation methods aim to capture sequential patterns and co-occurring events and are widely used in the literature. For example, the authors in [43] use a CTM as a basis for their recommendation algorithms. Of the methods that we discussed, the simple transition matrix - which is essentially a first-order Markov model - is particularly popular [7, 18, 20, 46, 54, 55]. The STM was also employed in [9] for feeding transition frequencies as features into Support Vector Machines (SVM) in order to predict unseen patterns. Another hybrid Markov process was employed in [14] to take into account the creation, splitting and closing of browser tabs - which are assumed to constitute parallel tasks and subtasks. Markov models are also one of the three main components of the AccessRank algorithm [22] and lie at the core of the context-sensitive mechanisms of SmartFavorites [11], as well.

Association rules (AR) are an alternative, well-established method for effectively identifying revisits that regularly occur together [4, 6]. AR do not take the order in which they occur into account, but may involve rules that use several pages as their conditional element. Numerous variants of AR have been investigated, among others in [3, 24, 38, 45, 53]. Most of them coped with efficiency issues related to learning and updating the rules for larger navigation histories [5]. An alternative for AR, *Frequent Sequences* [26], exploits the set of transitions between Web pages that occurred within a session with a minimum amount of support. A more flexible form of the previous technique are *Frequent Generalized Sequences* [25], which make use of wildcards; experimental results suggest that plain Frequent Sequences perform better in revisitation prediction.

Of the propagation methods discussed earlier, the functionality of the association matrix comes closest to association rules. AM overcomes the disadvantage of AR that they require expensive updates of the rules after each page visit. Further, AM allows for recommending rare, non-obvious and serendipitous page trails that would never reach the minimum support level in AR - then again, AR can capture more complex and elaborate sequences than AM.

Drift methods are typically used in the context of recommender systems: in [37], the authors present a software assistant for scheduling meetings, which employs a time frame in order to adapt quickly to the changing preferences of the user. Another system for learning drifting user profiles through Web and e-mail activity is presented in [17]; NewsDude [10] incorporates concept drift in its news recommendation service. A personalized Web search engine that supports evolving user profiles was introduced in [47]. Of particular interest is also the integration of concept drift in recommender systems that are based on collaborative filtering [19, 30].

4.3 Creating Revisitation Workflows

In the preceding sections, we discussed three kinds of revisitation prediction techniques. *Ranking methods* estimate whether a page will be revisited based on the frequency and/or recency of previous visits. *Propagation methods* aim to identify pages that are typically revisited together - not necessarily in the same order. *Propagation drift methods* specifically address changes in habits and interests of users over time.

As these groups of methods capture complementary patterns in the user's navigation history, the predictive accuracy of their methods can be enhanced by combining them into *revisitation workflows*. For convenience of presentation, we separate the workflows into three steps, which correspond to the three kinds of revisitation prediction techniques.

A revisitation workflow can be created by selecting one or more methods from each category. Not all combinations are useful, though: drift methods, for example, cannot be used in isolation, as they are only useful when combined with a propagation method (as explained in Section 4.2.3). In the remainder of this paper, we will evaluate variations of the following three types of workflows:

- *One-step* workflows comprise single revisitation prediction techniques. These can be either ranking or propagation methods. We will refer to workflows of the former type as R and of the latter type as P.
- *Two-step* workflows involve methods from two different categories. They either combine ranking methods with propagation methods (R+P) or propagation methods with drift methods (P+D).
- *Three-step* workflows combine methods from each category and are represented by R+P+D.

In total, we consider five types of revisitation workflows: R, P, R+P, P+D and R+P+D. In each workflow, one can include multiple ranking, propagation or drift methods. For simplicity, though, we exclusively consider *flat* workflows, in which at most one method from each category is chosen. In the next section, we will evaluate the effectiveness and efficiency of a representative set of such workflows.

There are several ways to combine the individual methods into a two-step or three-step workflow. We have chosen the following approach for three-step flat workflows: after each page request, the selected ranking method estimates the ranking scores of all visited Web pages. Further, the corresponding weights in the propagation matrix are updated - these weights are then used for spreading the current ranking scores to co-occurring Web pages. This is done through a linear scheme that increments the *propagation score* of a page p_j as follows:

$$c_j = c_j + p(i,j) \cdot v_i, \text{ where}$$

- c_j is the propagation score of p_j.
- $p(i,j)$ is the *transition probability* from page p_i to page p_j, derived from $p(i,j) = \frac{M(i,j)}{\sum_{k=0}^{|\mathbf{P}|} M(i,k)}$, and
- v_i is the ranking score of p_i.

After the propagation method has completed its processing, the *overall score* of each Web page is calculated as the sum of its ranking score and its propagation score. Subsequently, the overall scores are normalized to the interval $[0,1]$ - these scores, in descending order, are used for sorting the pages based on the estimated probability that they will be revisited. As a final step, the selected drift method

Table 1 The major (sub)types of revisitation prediction techniques.

Ranking Methods	event-based	Least Recently Used (LRU) Most Frequently Used (MFU) Polynomial Decay (PD) Exponential Decay (ED) Logarithmic Decay (LD)
	hybrid	Frecency (FR) Hybrid Day Model (HDM) Hybrid Quarter Model (HQM) Hybrid Hour Model (HHM)
Propagation Methods	order-preserving	Simple Transition Matrix (STM) Continuous Transition Matrix (CTM) Decreasing Transition Matrix (DTM) Increasing Transition Matrix (ITM)
	order-neutral	Association Matrix (AM)
Propagation Drift Methods	event-based	500-requests (5HR) 1000-requests (TR)
	time-based	Day-model (DM) Week-model (WM) Month-model (MM)

adjusts the weight of page associations by applying the chosen sliding window - these associations will be used as propagation scores in the next step.

Note that this linear propagation scheme ensures that the higher the ranking score of a page, the more the pages associated with it are boosted and the more their ranking is upgraded. This functionality slightly resembles the processing of PageRank, but there is a fundamental difference between them: our approach propagates the ranking scores of each page only to its direct neighbors, whereas RageRank propagates the ranking scores to more distant nodes, performing multiple iterations until convergence.

Note also that we assume that all diagonal cells of a propagation matrix M - which represent transitions from one page to itself, for example caused by a 'refresh' - are set to 0 (i.e., $\forall p_i \in \mathbf{P} : M(i,i) = 0$). Preliminary experiments demonstrated that non-zero diagonal cells mainly cause that the scores of top-ranked pages are reinforced, which reduces the effect of boosting the overall score of co-occurring Web pages - the main goal of propagation methods.

The operation of two-step workflows is the same, skipping only the functionality of the missing step. Note that there is a clear trade-off between the predictive accuracy and the space and time complexity of these workflows. In Sections 5.2 and 5.3, we will demonstrate that adding a propagation method to a workflow increases substantially the accuracy at the cost of higher execution time and higher memory requirements. Therefore, when defining and implementing revisitation workflows, one should consider the time requirements as well as the resources that are available to their application.

In order to support developers in implementing revisitation workflows, and to encourage researchers to experiment with different revisitation workflows, we have publicly released the SUPRA framework, which is a Java implementation of the functionality described in this section[12] [41]. The framework contains implementa-

[12] SUPRA stands for "*SU*rfing *PR*ediction Fr*A*mework". The code is publicly available at `http://sourceforge.net/projects/supraproject`.

tions of the revisitation prediction methods discussed in this chapter (see Table 1) and methods for creating revisitation workflows, as discussed above. Methods from the same category expose the same interface, requiring the same form of input and producing the same form of output. Therefore, they can be used collectively and interchangeably, creating revisitation workflows of arbitrary complexity. The structure of the framework allows for a-priori estimating its requirements and performance. The set of revisitation methods can be extended with new mechanisms by ensuring that the implementation complies with Definitions 4 or 9.

5 Evaluation

In this section, we evaluate the effectiveness and efficiency of the revisitation prediction methods discussed in the previous section. Apart from the individual methods, we also investigate revisitation workflows that involve two or three methods. The effectiveness (predictive performance) is evaluated in two conditions: the 'normal' condition (Section 5.2.1), in which all visited pages are kept in the navigation history and the 'optimal' condition (Section 5.2.3), in which pages that will never be revisited again are removed from the user's navigation history. We compare and analyze the performance of the most common workflows with one another, identify the most accurate workflows and analyze the factors that influence the performance of the workflows (Section 5.2.2). We also compare the relative time efficiency of the most accurate workflows in Section 5.3. We end this section with a discussion of the findings and future directions in Section 5.4.

5.1 Preliminaries

In this section, we present the data collections that are used in our experiments, we explain the setup of our experimental study and we define the metrics that measure the effectiveness and the efficiency of revisitation workflows.

5.1.1 Datasets

For our experimental evaluation, we employed two large-scale, real-world datasets. Both of them are publicly available and can be used as benchmarks[13]. The main characteristics of both datasets are summarized in Table 2.

The smaller dataset, symbolized by $\mathcal{D}_{HT}$, was originally used in [40, 52]. It comprises the activity of 25 users (19 male and 6 female) with an average age of 30.5 - individual ages ranged from 24 to 52 years. The Web activity of all participants were logged for some period between August, 2004 and March, 2005. On average, their activity was recorded for a period of 104 consecutive days, with a minimum of 51 days and a maximum of 195 days. All 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, with half of them corresponding to revisits. The participants visited more than 65,000 distinct Web pages, of which only 28% were revisited. The page requests were grouped into 4,715 off-line sessions with an average session length of 29 page visits.

[13] `http://sourceforge.net/projects/supraproject`

Table 2 Descriptive statistics of the datasets employed in our experimental evaluation.

	$\mathcal{D}_{\mathbf{HT}}$	$\mathcal{D}_{\mathbf{WHR}}$
Users	25	180
Logging Period (days)	104.97 ± 32.41	104.59 ± 105.19
Page Requests	137,272	2,470,779
Revisitation Requests	69,631 (50.72%)	938,388 (37.98%)
Distinct Web Pages	67,641	1,532,391
Revisited Pages	19,380 (28.65%)	218,938 (14.29%)
Sessions	4,715	21,431
Av. Requests per Session	29.11	115.29
Av. Revisitation Entropy	7.41 ± 1.13	7.54 ± 1.67
Av. Navigation Entropy	9.71 ± 1.23	11.13 ± 1.86

The larger dataset, denoted by $\mathcal{D}_{WHR}$, was collected through the Web History Repository initiative. It contains the navigational activity of 180 users who contributed at least 1,000 page requests. We do not have any demographic information about the volunteers (such as age and gender), as they offered their data anonymously. The recorded navigational activity spans the time period between September, 2009 and June, 2011, with every user logged for 105 days, on average. The distribution of the logging period per user varies greatly, ranging from a few days to 14 months. In total, around 2.5 million page accesses were recorded, with more than one third of them constituting a revisit. In total, the users visited 1.5 million distinct Web pages, of which less than 15% were revisited. The page requests are partitioned in 21,431 on-line sessions with an average session length of 115 page visits.

In both datasets, the navigational activity is unevenly distributed among the individual users: some of them visited a few hundred distinct pages in the course of few and short sessions, while others visited tens of thousands of pages over much longer sessions. Yet, $\mathcal{D}_{HT}$ and $\mathcal{D}_{WHR}$ are quite different from one another in the main aspects of revisitation, reflecting remarkable changes in users' navigational activity in the about 6 years that lie between the periods in which the data for $\mathcal{D}_{HT}$ and $\mathcal{D}_{WHR}$ was collected: the revisitation rate in $\mathcal{D}_{WHR}$ is 25% lower than in $\mathcal{D}_{HT}$; the portion of revisited pages in $\mathcal{D}_{WHR}$ is even 50% lower. The two datasets are also substantially different in terms of *navigation entropy*, a measure that quantifies the diversity of a user's navigational activity with respect to the frequency of visits to the accessed pages: Table 2 indicates that, on average, every page request targets one of ($2^{9.71}\approx$) 838 candidate Web pages in $\mathcal{D}_{HT}$ and one of ($2^{11.13}\approx$) 2,241 possible destinations in $\mathcal{D}_{WHR}$. Nevertheless, both datasets have similar values for *revisitation entropy*, a measure that assesses the average diversity of a user's revisitation activity: Table 2 indicates that, on average, every revisit in $\mathcal{D}_{HT}$ targets one of ($2^{7.41}\approx$) 170 candidate Web pages, whereas a revisit in $\mathcal{D}_{WHR}$ targets one of ($2^{7.54}\approx$) 186 candidate pages. We explain the effect of these two measures in more detail in Section 5.2.

5.1.2 Setup

We simulated the navigational activity of each user independently from one another by 'replaying' their navigation history, repeating the procedure for each navigation prediction workflow listed in Table 1. For every page request, a ranked list of pages from the URL vocabulary was calculated using the current prediction method. If the page request was a revisit, the ranking position of the corresponding Web page was recorded for calculating the effectiveness scores - the measures are introduced after this section. If the visit involved a page not visited before, this page was added to the list of visited pages and the ranking of all pages was updated according to the selected revisitation workflow.

We carried out the experiments in two different settings: the *Full Search Space* (FSS), which adds all pages that a user visits to the URL vocabulary, and the *Optimized Search Space* (OSS), in which all pages that will not be revisited again are automatically ignored - for this, we use an 'oracle' that checks for each newly visited page whether it will be visited in the future or not. The results of the OSS settings allow us to explore the upper limit on the effectiveness and efficiency of each revisitation workflow. We will further elaborate on this setting in Section 5.2.3.

All methods were fully implemented in Java, version 1.7. We used the open-source library JUNG[14] for the implementation of graphs. For all workflows and individual techniques relying on PD, we set the decay rate (α in Formula 2) at 1.25; this configuration was experimentally verified to yield the highest predictive accuracy for PD [42]. All experiments were performed on a desktop machine with Intel i7, 32GB of RAM memory, running Linux (kernel version 2.6.38).

5.1.3 Evaluation Metrics

As explained above, during the simulation of each users' navigational activity, we recorded the ranking position of each revisited page. Based on the list of ranking positions for all revisits, we evaluate the *effectiveness* of the revisitation workflows with the following metrics:

- **Success Rate at 1** (S@1) expresses the percentage of revisits that involved the page that was ranked first by the revisitation workflow. S@1 scores are in the interval [0%, 100%], with higher values corresponding to higher predictive accuracy.
- **Success Rate at 10** (S@10) expresses the percentage of revisits that involved a page that was listed within the top-10 ranking positions. The reason for using S@10 is that users can only revisit one page - recommended or not - at a time and only consider a limited number of recommendations [27].

Another common metric for measuring the effectiveness of recommendation techniques is the mean reciprocal rank (MRR), the average of the inverse ranks of all recommended items. We did not use this metric for two reasons. First, the MRR also expresses differences in ranking in the long tail of lower-ranked recommendations. As revisitation tools usually only display a limited, small number of recommendations, we are more interested whether they are included in the top-n ranking positions. Second, the revisitation prediction task is more difficult for users with larger URL vocabularies; this implies that the average ranking of predictions early on in the simulation will be higher than at the end of the simulation, an effect that mainly affects lower-ranked pages.

Precision@k and Recall@k are two other common metrics that we did not use. P@k indicates the number of *correct* predictions in a set of k recommendations and

[14] `http://jung.sourceforge.net`

$R@k$ the number of *relevant* predictions in the set. As users will follow at most one recommendation at a time, the $P@k$ scores will be the same as $S@k$. Due to the evolving URL vocabulary, recall is an impractical measure, as it is impossible to determine the set of pages that are relevant for the user - except for the trivial case in which we assume that only the actually revisited page is the only relevant one, in which case also $R@k$ will be similar to $S@k$.

To measure the difference in effectiveness between FSS and OSS, we employ the metrics $\Delta S@1$ and $\Delta S@10$. Their values are always positive and express the maximum possible increase in $S@1$ and $S@10$ that can be achieved in the ideal settings of OSS. Formally, these metrics are defined as follows:

$$\Delta S@1 = \frac{S@1_m - S@1_a}{S@1_a} \cdot 100\% \text{ and } \Delta S@10 = \frac{S@10_m - S@10_a}{S@10_a} \cdot 100\%,$$

where $S@1_m$ ($S@10_m$) represents the maximum $S@1$ ($S@10$) for the selected workflow, while $S@1_a$ ($S@10_a$) stands for the workflow's actual value for $S@1$ ($S@10$). Note that $\Delta S@1$ and $\Delta S@10$ make sense only if $S@1_a$ and $S@10_a$ are non-zero.

To measure the *efficiency* of a revisitation workflow, we consider its *mean ranking time*, which is symbolized by $t_{\bar{rank}}$. It is measured in milliseconds and expresses the average time required by the selected workflow to update the ranking of the URL vocabulary after every new page request. The lower its value, the more efficient the corresponding workflow.

5.2 Effectiveness Experiments

All revisitation workflows have been applied to both datasets. Due to the high number of possible two-step and three-step workflows, we only consider those methods that involve the best-performing methods from the preceding step(s). For instance, R+P is only represented by workflows that use PD as ranking method. For estimating the effectiveness of workflows that only involve propagation methods P, we combine its workflows with a naive scoring approach: after each page request, the ranking score of all pages is 0, except for the currently visited page, of which the score is 1. In this way, only pages that are associated with the latest visited page are taken into account - in correspondence with Definition 7. The same configuration was employed for the workflows of category P+D. As baseline methods, we consider the basic revisitation techniques LRU and MFU, together with FR, which is widely used in practice, being integrated into one of the most popular Web browsers.

Section 5.2.1 presents our analysis under the FSS settings, Section 5.2.2 presents a statistical analysis of the main parameters that effect effectiveness, and Section 5.2.3 examines the upper limits on effectiveness under the OSS settings.

5.2.1 Full Search Space

The outcomes of our analysis over $\mathcal{D}_{HT}$ and $\mathcal{D}_{WHR}$ under the Full Search Space settings are presented in Tables 3 and 4. As we will discuss in more detail in Section 5.2.3, the predictive performance for $\mathcal{D}_{WHR}$ is lower than for $\mathcal{D}_{HT}$ because, on average, the methods had to work with a larger URL vocabulary.

Of the baseline methods, MFU and FR have the lowest performance for both datasets; LRU performs consistently better, which indicates that recency is a better

Table 3 Performance of the main revisitation workflows for $\mathcal{D}_{HT}$.

One-Step Workflows	**S@1**	**S@10**	**Two-Step Workflows**	**S@1**	**S@10**
PD	**20.33%**	**74.11%**	PD+AM	20.84%	76.13%
HDM	19.24%	71.31%	PD+CTM	21.46%	75.94%
HQM	19.23%	70.70%	PD+DTM	22.76%	80.88%
HHM	19.28%	70.13%	PD+ITM	20.69%	74.23%
			PD+STM	**24.63%**	**82.35%**
AM	19.07%	**59.13%**	AM+5HR	19.07%	56.28%
CTM	19.29%	45.29%	AM+TR	19.07%	57.80%
DTM	19.29%	50.11%	AM+DM	18.91%	51.70%
ITM	19.29%	40.09%	AM+WM	19.06%	55.48%
STM	**20.60%**	47.23%	AM+MM	**19.07%**	**58.18%**
Three-Step Workflows	**S@1**	**S@10**	**Baseline Methods**	**S@1**	**S@10**
PD+STM+5HR	24.58%	80.53%	LRU	**19.21%**	**71.26%**
PD+STM+TR	**24.59%**	**81.52%**	MFU	12.67%	32.19%
PD+STM+DM	23.93%	76.65%	FR	12.57%	32.33%
PD+STM+WM	24.47%	79.62%			
PD+STM+MM	**24.62%**	**81.57%**			

Table 4 Performance of the main revisitation workflows for $\mathcal{D}_{WHR}$.

One-Step Workflows	**S@1**	**S@10**	**Two-Step Workflows**	**S@1**	**S@10**
PD	**9.46%**	**44.34%**	PD+AM	9.69%	48.27%
HDM	9.27%	43.42%	PD+CTM	10.18%	50.94%
HQM	9.27%	43.24%	PD+DTM	10.57%	59.56%
HHM	9.43%	44.19%	PD+ITM	10.02%	47.55%
			PD+STM	**12.96%**	**60.68%**
AM	8.92%	**41.05%**	STM+5HR	13.58%	31.15%
CTM	9.83%	37.58%	STM+TR	**14.19%**	**35.64%**
DTM	9.75%	40.19%	STM+DM	10.61%	18.83%
ITM	9.83%	30.03%	STM+WM	11.49%	21.53%
STM	**14.45%**	39.73%	STM+MM	12.31%	23.82%
Three-Step Workflows	**S@1**	**S@10**	**Baseline Methods**	**S@1**	**S@10**
PD+STM+5HR	13.02%	56.00%	LRU	**8.97%**	**41.44%**
PD+STM+TR	**13.06%**	**57.98%**	MFU	8.39%	30.40%
PD+STM+DM	10.51%	44.27%	FR	8.79%	32.07%
PD+STM+WM	11.28%	44.77%			
PD+STM+MM	11.86%	46.43%			

predictor than frequency of use. The individual ranking methods in the one-step workflows all perform better than the baseline methods, which confirms the intuition behind PD and its variations that both recency and frequency should be taken into account in the ranking process. The performance of the hybrid ranking techniques HDM, HHM and HQM lies close to PD, but is still consistently lower. This is a sign that the time of day or the day of week only play a limited role in revisitation patterns.

Of the individual propagation methods in the one-step workflows, we observe that the order-preserving simple transition matrix STM consistently achieves the best performance in terms of $S@1$; the order-neutral association matrix AM consistently achieves best in terms of $S@10$. The lower performance of the more elaborate propagation methods, which take all pages visited in the current session into account, suggests that the latest visited page is the most important predictor - this is also supported by the observation that the decreasing transition model DTM is the best-performing of the more elaborate methods. Apparently, it does not make much difference whether one takes the order in which the pages are visited into account: STM appears to be better for revisitation tools that only provide one recommendation, but the order-neutral approach seems to be the best choice for tools that suggest more than one - however, particularly for the $\mathcal{D}_{WHR}$ dataset, the differences between the propagation methods are relatively small.

In terms of $S@1$, the performance of the propagation methods STM and AM is similar to the ranking method PD, but PD is superior in terms of $S@10$, scoring 26%±7 better in $\mathcal{D}_{HT}$ and 6%±4 better in $\mathcal{D}_{WHR}$. This indicates that if one has to select one single revisitation method, ranking methods are the better choice - in addition, ranking methods have significantly lower space and time requirements than propagation methods.

However, as we will discuss in more detail in Section 5.2.2, propagation methods capture different aspects of user navigation. For this reason, it comes not as a surprise that combining the two of them has a beneficial effect: in terms of $S@10$, the two-step workflow PD+STM achieves best performance with 82% in $\mathcal{D}_{HT}$ and 61% in $\mathcal{D}_{WHR}$ - it is interesting to observe that in the two-step workflow PD+STM outperforms the order-neutral PD+AM.

In contrast, combining propagation methods with drift methods, which impose sliding windows of different length on the propagation matrices (see Section 4.2.3), does not improve performance, as compared with propagation methods only. On the contrary, the scores in terms of $S@1$ and $S@10$ are consistently slightly lower than either STM or AM. The same yields for the three-step workflows that combine PD+STM with propagation drift methods: the performance is worse than that of PD+STM only.

The negative effect of drift methods seems to indicate that applying a sliding window, to better accommodate changes in user habits and interests, does more harm than good - the benefits of a larger navigation history are higher. However, it might well be that drift methods are useful for very active users with a large navigation history. To verify this, we calculated the Pearson correlation between $\Delta S@1$ and $\Delta S@10$ and the size of the navigation history of every user (here, $S@1_a$ and $S@10_a$ correspond to the performance of the individual propagation method, while $S@1_m$ and $S@10_m$ correspond to the two-step workflow). The resulting correlation coefficients were in the interval [-0.2,0.2] across both datasets, indicating little or no relation between the two. We applied the same analysis to the time-based drift methods, estimating the correlation of $\Delta S@1$ and $\Delta S@10$ with the logging period of every user, and got similar results. The same also applies to the effect of drift methods on three-step workflows. In summary, the results confirm the negative effect of limiting the navigation history and suggest that one should use as much evidence as one can get from the navigation history.

5.2.2 Statistical Analysis

In the previous section we discussed the performance of various revisitation workflows, averaged among all users. In this section, we investigate to what extent the

Table 5 Performance correlations between several pairs of revisitation workflows.

	FR	DM	PD	STM	AM	PD+STM	PD+AM	PD+TR
FR	–	.52	.51	.39	.59	.52	.56	.56
DM	.52	–	1.0	.38	.58	.85	.96	.87
PD	.51	1.0	–	.38	.57	.85	.96	.87
STM	.39	.38	.37	–	.82	.68	.39	.63
AM	.59	.58	.57	.82	–	.74	.64	.73
PD+STM	.52	.85	.85	.68	.74	–	.85	.98
PD+AM	.56	.96	.96	.39	.64	.85	–	.89
PD+TR	.56	.87	.87	.63	.73	.98	.89	–

performance varies per individual user and which user characteristics influence the performance of revisitation prediction methods. For the analysis, we take the S@10 scores for each individual user in $\mathcal{D}_{WHR}$ as a basis.

To start with, we inspected the distribution of performance scores between individual users. All predictive methods followed a non-skewed normal distribution, with standard deviations between 10.3 and 13.8. This indicates that the predictions are suitable for the average user. Moreover, the correlations between the method results for the users are significant with $p < .01$, albeit with R values varying from medium to very large effect sizes - see Table 5. FR and the individual propagation methods AM and STM correlate least with the other methods, with maximum values of R between .56 and .74. The PD-based methods (only PD or combined with a propagation or drift method) correlate with one another with R between .85 and .98, which suggests that the methods capture similar behavior for the users.

To investigate which revisit characteristics influence the success of a predictive method for individual users, we gathered several statistics of the users' revisit behavior: the revisit rate (the ratio between revisited pages and the total number of page visits), the average session length, the average revisit distance (the number of page requests between revisits of a particular page), the amount of backtracking and the entropy (unpredictability) within page visits and page revisits. Inspection of the correlations between these measures showed that they are largely unrelated, with no significant correlations between one another. The only significant correlation ($r = .70$) is between the average revisit distance and the entropy measures. In other words, users with low navigation and revisitation entropy mainly revisit recently used pages.

We used the user characteristics for finding explanations for variations in performance between users. First, we checked and did not find any effects of the users' daily activity or the size of their logs on prediction performance, which confirms that the experimental results were not influenced by differences in the sample sizes for individual users. We found that the above-mentioned revisit characteristics explained the differences in performance of the predictive methods only to a certain extent - see Table 6. The table shows that the navigation entropy (R between -.29 and -.74) and the revisitation entropy (R between -.40 and -.67) are the best predictors for method performance. Further, the correlation coefficients of a user characteristic with the different predictive methods follow similar patterns, with weak positive correlations for the revisit rate, moderate positive correlations for backtracking, and moderate to strong positive correlations for both revisit en-

Table 6 Correlations between user characteristics and prediction performance over $\mathcal{D}_{WHR}$.

	FR	DM	PD	STM	AM	PD+STM	PD+AM	PD+TR
Revisit Rate	.41	.12	.11	.42	.31	.27	.11	.22
Backtracking	.26	.43	.42	.49	.43	.42	.39	.40
Revisit Entropy	-.74	-.51	-.49	-.29	-.48	-.47	-.57	-.58
Navigation Entr.	-.67	-.52	-.51	-.40	-.52	-.55	-.59	-.66

tropy and page entropy. A linear regression model based on the user characteristics explained a modest 34% of the variance in prediction performance.

The impact of the navigation and revisitation entropy on prediction performance suggests that some gain may be achieved by cleaning the search space at the end of the tail of the navigation history's frequency distribution, by removing those pages that are visited only once and are never revisited. This will be investigated in the next section.

5.2.3 Optimized Search Space

The analyses presented in the previous sections were based on the Full Search Space, which involves the entire navigation history of the users. In practice, the power-law distribution of the number of visits to each page dictates that the vast majority of pages at the end of the tail (over 70%) are never revisited. If one would have prior knowledge which pages are likely not to be revisited again, they could be excluded in the revisitation prediction process. Excluding these pages would remove the noise caused by these irrelevant pages and boost both effectiveness and efficiency of revisitation workflows.

In this section, we examine the performance of all revisitation workflows under the ideal settings of OSS, in which a *revisitation oracle* discards with 100% accuracy the non-revisited Web pages right after the first (and only) access to them. We illustrate the effect of this oracle in view of the average values for ΔS@1 and ΔS@10 it yields per workflow category. They effect of the oracle on S@1 and S@10 is summarized in Figures 2(a) and (b).

A first observation is that the oracle has a similar, positive effect on the performance of all revisitation workflows, with relatively comparable values for ΔS@1 and ΔS@10. In other words, removing the irrelevant pages from the search space would make sense for all methods. The only exceptions - not displayed in Figure 2 - are FR and MFU, which do not improve significantly when combined with the oracle: $\Delta S@1 = 0.24\%$ and $\Delta S@10 = 0.94\%$ for $\mathcal{D}_{HT}$ and $\Delta S@1 = 0.44\%$ and $\Delta S@10 = 3.23\%$ for $\mathcal{D}_{WHR}$. This is not unexpected, as MFU and, to a somewhat lesser extent, FR are frequency-based, which implies that non-revisited pages rarely make it to the top ranking positions.

Comparing the average ΔS@1 and ΔS@10 among the two datasets, we observe that there is a positive correlation between them: the higher the ΔS value for a revisitation workflow in the $\mathcal{D}_{HT}$ dataset, the higher is its corresponding value for $\mathcal{D}_{WHR}$. In absolute values, both ΔS@1 and ΔS@10 are significantly higher for $\mathcal{D}_{WHR}$ in most of the cases. This may be explained by the increase in navigation activity between the two datasets - see Table 2. In the older dataset, $\mathcal{D}_{HT}$, the average number of pages visited per day was 52.29; in $\mathcal{D}_{WHR}$, collected six years later, users visited on average 131 pages per day. As the revisitation entropy and

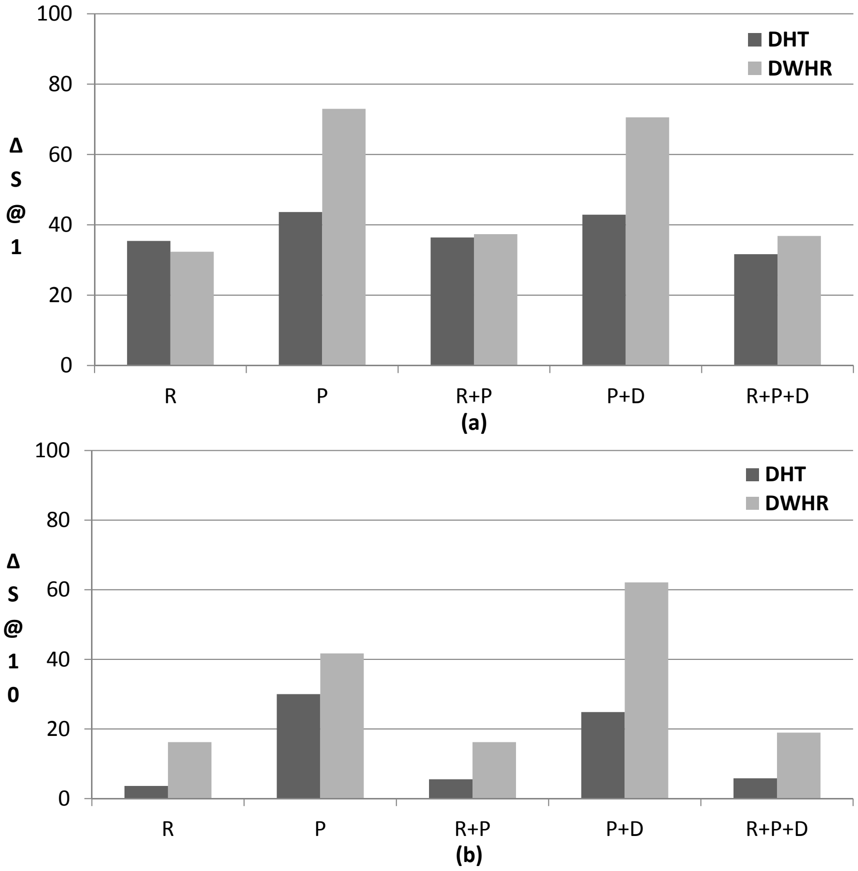


Fig. 2 Average ΔS@1 and ΔS@10 for every category of revisitation workflows in (a) $\mathcal{D}_{HT}$ and (b) $\mathcal{D}_{WHR}$. R denotes the ranking methods, P the propagation methods, and D the drift methods.

the navigation entropy remained stable, this means that the methods in $\mathcal{D}_{WHR}$ have to work with a larger URL vocabulary - and consequently with a far longer tail of pages that were visited only once. In absolute numbers, the oracle restricts the pool of candidate Web pages from 838 to 170 for $\mathcal{D}_{HT}$ and from 2,241 to 186 for $\mathcal{D}_{WHR}$.

Note also that ΔS@1 exceeds ΔS@10 in practically all cases, thus implying that the oracle is particularly useful when the goal is to provide the user with just one recommendation. There is a simple explanation for this: the original values of S@10 are substantially higher than those of S@1 across all revisitation workflows, making it harder to further improve them. Therefore, an equivalent increase in S@1 and S@10 is more recognizable for the former metric, due to its initial low values.

Another interesting observation is that both ΔS@1 and ΔS@10 are highest for those workflows that do not involve a ranking method - the categories P and P+D. This pattern is consistent across both datasets - but stronger for $\mathcal{D}_{WHR}$ - and indicates that the weighted page associations of propagation matrices convey

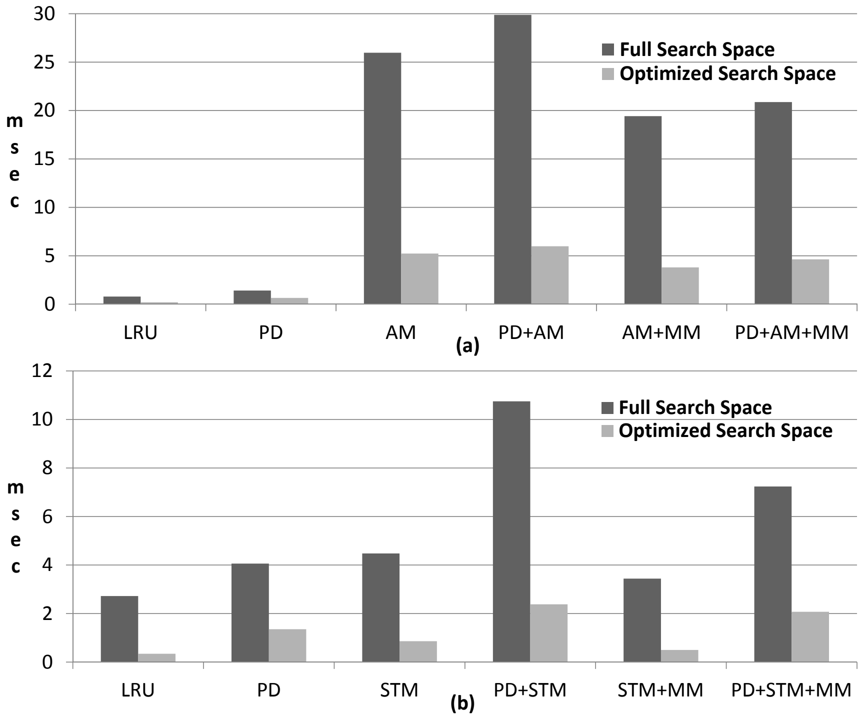


Fig. 3 Mean ranking time for the most accurate workflows over (a) $\mathcal{D}_{HT}$ and (b) $\mathcal{D}_{WHR}$, under both simulation settings. Note that the two diagrams use different scales.

high levels of noise in the FSS settings, which can be effectively cleaned by the revisitation oracle. In contrast, the lower $\Delta S@1$ and $\Delta S@10$ values for the categories R, R+P and R+P+D indicates that the cleansing effect of the revisitation oracle is moderate for workflows that involve a ranking method. The reason for this effect is probably that the functionality of ranking methods resembles that of the oracle: in essence, they distinguish the frequently revisited Web pages from those that are highly unlikely to be revisited.

Having outlined the theoretical benefits of the oracle, we discuss how we can put its functionality into practice in Section 5.4.

5.3 Efficiency Experiments

In Section 4, we already discussed the theoretical time complexity of revisitation workflows. In this section, we investigate how this translates into actual computation times. For this purpose, we assess the minimum ranking times for the most accurate workflows. To ensure that the computation times are comparable with one another and to reduce the influence of external parameters, we repeated the measurement of the ranking times for each workflow and for each individual user ten times, making use of the setup described in Section 5.1.2. We report the mean values of the results, separating the FSS and the OSS settings.

Figures 3(a) and (b) show the mean ranking times for the most effective revisitation workflows over $\mathcal{D}_{HT}$ and $\mathcal{D}_{WHR}$, measured in milliseconds. Note that the two scales in the diagrams are different from one another, partially due to the higher number of candidate Web pages for $\mathcal{D}_{WHR}$ - which we already observed and discussed in Section 5.2.3. A second reason for the different scales is that we selected AM as the propagation method for $\mathcal{D}_{HT}$ and STM for $\mathcal{D}_{WHR}$ - mainly for illustrative purposes, as both methods had comparable results in terms of effectiveness.

Comparing the FSS values of the results in (a) and (b), one can see that the differences in efficiency of the different revisitation workflows are similar in both datasets. The baseline method LRU is most efficient, closely followed by the ranking method PD.

In line with the higher computational complexity, the workflows with one of the propagation methods take significantly more time than the workflows without. However, also between the propagation methods a difference can be observed between the order-neutral AM and the order-preserving STM. The reason for this is that AM involves a significantly higher number of page associations than STM, as it connects all pages that co-occur within the same session, whereas STM only connects consecutively visited pages. Note that both AM and STM performed similarly in terms of effectiveness - see Section 5.2

The workflows R+P - which produced the best prediction performance - have the worst performance in terms of computational costs. This yields for both PD+AM over $\mathcal{D}_{HT}$ and PD+STM over $\mathcal{D}_{WHR}$. The reason for this is that the combined ranking and propagation methods require recomputation of all page associations stored in the propagation matrix; for propagation-only workflows it is sufficient to only update the associations of pages that are connected with the currently visited page p_n.

Applying a drift method to the workflow - we used the month model MM - clearly reduces the needed time for computing the rankings. The reason for this is simple: removing the outdated pages - in this case, pages that have not been visited during the past month - keeps the ranking lists and propagation matrices within limits. This positive effect in terms of computational efficiency comes at the price of slightly lower prediction performance - see Tables 3 and 4.

The same differences between the workflows can be observed for the efficiency results in the OSS condition. However, the most important observation is the dramatic reduction of the computation time for each workflow - varying from about 50% for LRU and PD over $\mathcal{D}_{HT}$ to about 80% for PD+STM over $\mathcal{D}_{WHR}$. Unsurprisingly, the improvement in efficiency is highest for the workflows that involve propagation methods, which have complexity $O(|\mathbf{P}|^2)$. It should be stressed that the improvement in computational efficiency of the *revisitation oracle*, which optimizes the search space by removing those pages that will never be revisited, is far higher than the cleansing effect from the drift methods, which remove outdated pages from the pool.

5.4 Discussion

In our study, we investigated the effectiveness (prediction performance) and the efficiency (computational performance) of revisitation workflows consisting of one, two or three complementary methods. The results indicate that the combination of ranking and propagation methods, (R+P), achieves the best results for the majority of users, with $S@10$ of up to 82% for the best-performing combination PD+STM for the $\mathcal{D}_{HT}$ dataset. However, effectiveness comes at a cost: the combination of ranking and propagation methods is computationally the most expensive workflow.

In Section 5.2.2, we showed that the performance of PD+STM for individual users followed a normal distribution, and that the performance of all methods for one user are positively correlated. Still, this does not imply that PD+STM would be the best method for all users - for 'non-average' users an alternative revisitation workflow might be beneficial. In order to find this out, we looked at the best-performing workflows for each individual user in $\mathcal{D}_{HT}$ and $\mathcal{D}_{WHR}$. In terms of $S@10$, PD+STM was the optimal choice for 54 (26%) of the users. However, all other workflows that performed best for more than two users (covering 73% of the participants) were of the category P+R; second was the hybrid hour model HHM combined with STM (8% of all users) and HQM+STM (6% of all users). As HHM and HQM are variations on PD, and given the similar performance of the PD-based ranking methods over both datasets, we can consider PD+STM to be a good choice for the majority of users.

The computational cost of the successful but computationally expensive workflow R+P revisitation workflows can be reduced by introducing propagation drift methods. These impose a sliding window on the navigational history and cancel out 'outdated interests and habits', do not improve predictive performance, but they keep the required computation time in limits. Therefore, for balancing effectiveness and efficiency, it is worth considering including a propagation drift method in workflows that involve propagation methods.

A further important method for reducing the high space requirements for propagation methods is to exploit the sparsity of the propagation matrices: in the navigation history, only a small number of all possible page transitions actually took place and the remaining cells have a value of zero. Therefore, in our implementation, we employed *propagation graphs* for modeling the propagation matrices. By doing so, we reduced the absolute differences in memory consumption to negligible values: all propagation matrices require less than 2GB of memory for up to 50,000 distinct visited pages, whereas the ranking methods occupy about 25% of this amount.

Finally, in Section 5.2.3, we showed the benefits in terms of both effectiveness and efficiency of a 'revisitation oracle' that optimizes the search space by removing pages that will not be visited again. However, we did not discuss how to approach the functionality of an oracle in practice. The oracle can be thought of as a binary classifier that categorizes every visited page as `revisited` or `non-revisited`. For this classification task, there are several types of evidence that the oracle can use. As we stated in the introduction, in this paper we limited the discussion to features that can be derived from the (anonymized) user's navigation history. In a preliminary experiment, we found that a classifier that makes use of the scores and rankings of individual pages - as well as of graph-based measures, including the node-degree and page-rank score - reached AUC values between 0.6 and 0.7, which is better than random, but still far from perfect.

However, other studies suggest that other types of evidence would be useful for this task. In [2, 50] it was shown that there is a relation between page content changes, the intent of visiting the page (for example, reference sites versus news site) and revisitation patterns. [40] showed that the function of a page within sites is an important indicator, as well: most sites have one or more portal pages that are revisited very frequently and a long tail of pages that are visited only once or twice. Further, [23] found that longer dwell times on a page can be indicative of its future utility. This is in line with the findings of [52], who also showed that within-page navigation - particularly scrolling - is an important indicator of user interest in the page content. Matching the semantics of the visited pages - keywords and entities extracted from its title and content - with the remainder of the user's navigation history for content-based classification and recommendation is a promising direction, as well [44]. This is supported by the findings of [1], who found a relation between the content of a site and how often it will be revisited: for instance, shopping and reference websites invoke fast revisits, whereas sites involving weekend activities are visited only infrequently. We see this area as a promising area for future research.

6 Conclusions

A significant amount of activities on the Web involves revisiting pages and sites that have been visited before. Particularly for revisits to pages that are used on an infrequent basis, improved support by (browser) history mechanisms is highly desirable. In order to develop such tools, it is important to be able to predict which pages will be revisited by users at a certain point in time. In this article, we investigated the predictive performance and computational efficiency of three different kinds of revisitation prediction methods: *ranking methods* (R), which estimate the overall, a-priori probability that a page will be visited, *propagation methods* (P), which base their predictions on groups of pages that typically co-occur in sessions, and *propagation drift methods* (D), which aim to reflect changes in user habits and interests by imposing a sliding window on the navigation history.

We evaluated several variations of these kinds of revisitation prediction methods, and combinations of them, over two real-world datasets. The results indicate that of the individual methods, the ranking method *polynomial decay* (PD) has the best predictive performance and moderate computational complexity. The best predictive results (with S@10 up to 80%) can be achieved by a *revisitation workflow* R+P that combines polynomial decay with the propagation method STM. However, PD+STM also has the highest computational costs. These costs can be reduced by imposing a sliding window through a propagation drift method, by employing more efficient graph-based representations of the sparse propagation matrices, or by removing pages that will not be revisited again using a 'revisitation oracle'.

The high predictive performance of the ranking and propagation methods confirms the importance of frequency and recency, as well as the current navigation context in the prediction process. Taking the time of day or the day of week into account does not improve - and in most cases slightly deteriorates - the results. Apparently, Web revisitation is only to a limited extent driven by the actual time of day. Similarly, taking changes in user habits and interests - as captured by the D methods - into account does not improve predictive performance either (but, as just discussed, they do improve the computational efficiency).

The performance of revisitation prediction workflows for individual users mainly depends on the entropy in the user's navigation and revisitation patterns: some users are more predictable than others - this observation is confirmed by the positive correlations between the performance of prediction methods between users. The best-performing workflow PD+SMT is a good choice for the majority of users.

A promising area of future research is the optimization of the navigation history, by recognizing pages that probably will not be revisited and removing them from the user's navigation history. In our studies we used an 'oracle' as an optimal classifier. The literature suggests that such a classifier can be achieved by taking additional features into account, including the dynamics and function of web pages, the user's dwell time and the content of the pages in the navigation history.

Appendix: Notations And Acronyms

In the following, we summarize the symbols used in this work:
5HR → the 500-requests drift method
AM → an association matrix (order-neutral propagation method)
AR → association rules
CTM → the continuous connectivity transition matrix (propagation method)
DM → the day-model drift method
DTM → the decreasing continuous connectivity transition matrix (propagation method)
ED → the exponential decay ranking method
FR → the Frecency ranking method
HDM → the hybrid day model (ranking method)
HHM → the hybrid hour model (ranking method)
HQM → the hybrid day quarter model (ranking method)
$\mathbf{I_{p_i}}$ → the request indices of a page p_i
ITM → the increasing continuous connectivity transition matrix (propagation method)
LD → the logarithmic decay ranking method
LRU → the last recently used ranking method
M → the propagation matrix
MFU → the most frequently used ranking method
MM → the month-model drift method
$\mathbf{P}$ → a set of Web pages
P → the category of one-step workflows that consist solely of a propagation method
P+D → the category of two-step workflows that combine a propagation method with a drift method
PD → the polynomial decay ranking method
$\mathbf{R}$ → a set of page requests corresponding to the navigational activity of a user
R → the category of one-step workflows that consist solely of a ranking method
R+P → the category of two-step workflows that combine a ranking method with a propagation method
R+P+D → the category of three-step workflows that combine a ranking method with a propagation and a drift method
S → a session
STM → the simple connectivity transition matrix (propagation method)

$\mathbf{T_{p_i}}$ → the request timestamps of a page p_i
TM → a transition matrix (order-preserving propagation method)
TR → the 1000-requests drift method
WM → the week-model drift method

References

1. Adar, E., Teevan, J., Dumais, S.T.: Large scale analysis of web revisitation patterns. In: Proceedings of the 26th Conference on Human Factors in Computing Systems, CHI 2008, 2008, Florence, Italy, April 5-10, 2008, pp. 1197–1206 (2008)
2. Adar, E., Teevan, J., Dumais, S.T.: Resonance on the web: web dynamics and revisitation patterns. In: Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009, pp. 1381–1390 (2009)
3. Adomavicius, G., Tuzhilin, A.: Using data mining methods to build customer profiles. IEEE Computer **34**(2), 74–82 (2001)
4. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993., pp. 207–216 (1993)
5. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile, pp. 487–499 (1994)
6. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the 11th International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan, pp. 3–14 (1995)
7. Albrecht, D.W., Zukerman, I., Nicholson, A.E.: Pre-sending documents on the WWW: A comparative study. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999, pp. 1274–1279 (1999)
8. Arcuri, M., Coon, T., Johnson, J., Manning, A., Van Tilburg, M.: Adaptive menus (2000). US Patent 6,121,968
9. Awad, M., Khan, L., Thuraisingham, B.M.: Predicting WWW surfing using multiple evidence combination. The VLDB Journal The International Journal on Very Large Data Bases **17**(3), 401–417 (2008)
10. Billsus, D., Pazzani, M.J.: A hybrid user model for news story classification. In: Proceedings of the 7th International Conference on User Modeling, UM 99, Banff, Canada, pp. 99–108 (1999)
11. Brank, J., Milic-Frayling, N., Frayling, A., Smyth, G.: Predictive algorithms for browser support of habitual user activities on the web. In: 2005 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2005), 19-22 September 2005, Compiegne, France, pp. 629–635 (2005)
12. Brusilovsky, P.: Adaptive hypermedia. User Modeling and User-Adapted Interaction **11**(1-2), 87–110 (2001)

13. Catledge, L.D., Pitkow, J.E.: Characterizing browsing strategies in the world-wide web. Computer Networks and ISDN Systems **27**(6), 1065–1073 (1995)
14. Chierichetti, F., Kumar, R., Tomkins, A.: Stochastic models for tabbed browsing. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010, pp. 241–250 (2010)
15. Cockburn, A., McKenzie, B.: What do web users do? an empirical analysis of web use. International Journal of Human-Computer Studies **54**(6), 903–922 (2001)
16. Cormode, G., Shkapenyuk, V., Srivastava, D., Xu, B.: Forward decay: A practical time decay model for streaming systems. In: Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China, pp. 138–149 (2009)
17. Crabtree, I.B., Soltysiak, S.J.: Identifying and tracking changing interests. Int. J. on Digital Libraries **2**(1), 38–53 (1998)
18. Deshpande, M., Karypis, G.: Selective markov models for predicting web page accesses. ACM Transactions on Internet Technology (TOIT) **4**(2), 163–184 (2004)
19. Ding, Y., Li, X.: Time weight collaborative filtering. In: Proceedings of the 14th ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005, pp. 485–492 (2005)
20. El-Sayed, M., Ruiz, C., Rundensteiner, E.A.: Fs-miner: efficient and incremental mining of frequent sequence patterns in web logs. In: 6th ACM CIKM International Workshop on Web Information and Data Management (WIDM 2004), Washington, DC, USA, November 12-13, 2004, pp. 128–135 (2004)
21. Findlater, L., McGrenere, J.: A comparison of static, adaptive, and adaptable menus. In: Proceedings of the 2004 Conference on Human Factors in Computing Systems, CHI 2004, Vienna, Austria, April 24 - 29, 2004, pp. 89–96 (2004)
22. Fitchett, S., Cockburn, A.: Accessrank: predicting what users will do next. In: Proceedings of the 2012 CHI Conference on Human Factors in Computing Systems, Austin, TX, USA - May 05 - 10, 2012, pp. 2239–2242 (2012)
23. Fox, S., Karnawat, K., Mydland, M., Dumais, S.T., White, T.: Evaluating implicit measures to improve web search. ACM Transactions on Information Systems (TOIS) **23**(2), 147–168 (2005)
24. Fu, X., Budzik, J., Hammond, K.J.: Mining navigation history for recommendation. In: Proceedings of the 5th International Conference on Intelligent User Interfaces, IUI 00, New Orleans, Louisiana, USA, pp. 106–112 (2000)
25. Gaul, W., Schmidt-Thieme, L.: Mining generalized association rules for sequential and path data. In: Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA, pp. 593–596 (2001)
26. Géry, M., Haddad, M.H.: Evaluation of web usage mining approaches for user's next request prediction. In: 5th ACM CIKM International Workshop on Web Information and Data Management (WIDM 2003), New Orleans, Louisiana, USA, November 7-8, 2003, pp. 74–81 (2003)
27. Hawking, D., Craswell, N., Bailey, P., Griffiths, K.: Measuring search engine quality. Information Retrieval **4**(1), 33–59 (2001)

28. Kawase, R., Papadakis, G., Herder, E., Nejdl, W.: The impact of bookmarks and annotations on refinding information. In: HT'10, Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, Toronto, Ontario, Canada, June 13-16, 2010, pp. 29–34 (2010)
29. Kawase, R., Papadakis, G., Herder, E., Nejdl, W.: Beyond the usual suspects: context-aware revisitation support. In: HT'11, Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia, Eindhoven, The Netherlands, June 6-9, 2011, pp. 27–36 (2011)
30. Koren, Y.: Collaborative filtering with temporal dynamics. Communications of the ACM **53**(4), 89–97 (2010)
31. Koychev, I., Schwab, I.: Adaptation to drifting user's interests. In: Proceedings of ECML Workshop: Machine Learning in New Information Age, Barcelona, Spain, pp. 39–46 (2000)
32. Kumar, R., Tomkins, A.: A characterization of online browsing behavior. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010, pp. 561–570 (2010)
33. Lee, D., Choi, J., Kim, J.H., Noh, S.H., Min, S.L., Cho, Y., Kim, C.S.: On the existence of a spectrum of policies that subsumes the least recently used (lru) and least frequently used (lfu) policies. In: Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Atlanta, Georgia, USA, pp. 134–143 (1999)
34. Lymberopoulos, D., Riva, O., Strauss, K., Mittal, A., Ntoulas, A.: Pocketweb: instant web browsing for mobile devices. In: Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2012, London, UK, March 3-7, 2012, pp. 1–12 (2012)
35. Mayer, M.: Web history tools and revisitation support: A survey of existing approaches and directions. Foundations and Trends in Human-Computer Interaction **2**(3), 173–278 (2009)
36. Milic-Frayling, N., Jones, R., Rodden, K., Smyth, G., Blackwell, A.F., Sommerer, R.: Smartback: supporting users in back navigation. In: Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004, pp. 63–71 (2004)
37. Mitchell, T.M., Caruana, R., Freitag, D., McDermott, J.P., Zabowski, D.: Experience with a learning personal assistant. Communications of the ACM **37**(7), 80–91 (1994)
38. Mobasher, B., Cooley, R., Srivastava, J.: Automatic personalization based on web usage mining. Communications of the ACM **43**(8), 142–151 (2000)
39. Morris, D., Morris, M.R., Venolia, G.: Searchbar: a search-centric web history for task resumption and information re-finding. In: Proceedings of the 2008 ACM CHI Conference on Human Factors in Computing Systems, Florence, Italy, April 5-10, 2008, pp. 1207–1216 (2008)
40. Obendorf, H., Weinreich, H., Herder, E., Mayer, M.: Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In: Proceedings of the 2007 ACM CHI Conference on Human Factors in Computing Systems, San Jose, California, USA, April 28 - May 3, 2007, pp. 597–606 (2007)
41. Papadakis, G., Kawase, R., Herder, E.: Client- and server-side revisitation prediction with SUPRA. In: 2nd International Conference on Web Intelligence,

Mining and Semantics, WIMS '12, Craiova, Romania, June 6-8, 2012, p. 14 (2012)

42. Papadakis, G., Niederée, C., Nejdl, W.: Decay-based ranking for social application content. In: WEBIST 2010, Proceedings of the 6th International Conference on Web Information Systems and Technologies, Volume 1, Valencia, Spain, April 7-10, 2010, pp. 276–281 (2010)
43. Parameswaran, A.G., Koutrika, G., Bercovitz, B., Garcia-Molina, H.: Recsplorer: recommendation algorithms based on precedence mining. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010, pp. 87–98 (2010)
44. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: P. Brusilovsky, A. Kobsa, W. Nejdl (eds.) The Adaptive Web, Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, pp. 325–341. Springer (2007)
45. Sandvig, J.J., Mobasher, B., Burke, R.D.: Robustness of collaborative recommendation based on association rule mining. In: Proceedings of the 2007 ACM Conference on Recommender Systems, RecSys 2007, Minneapolis, MN, USA, October 19-20, 2007, pp. 105–112 (2007)
46. Shani, G., Heckerman, D., Brafman, R.I.: An mdp-based recommender system. Journal of Machine Learning Research **6**, 1265–1295 (2005)
47. Sugiyama, K., Hatano, K., Yoshikawa, M.: Adaptive web search based on user profile constructed without any effort from users. In: Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004, pp. 675–684 (2004)
48. Takano, H., Winograd, T.: Dynamic bookmarks for the WWW. In: HYPERTEXT '98. Proceedings of the 9th ACM Conference on Hypertext and Hypermedia: Links, Objects, Time and Space - Structure in Hypermedia Systems, June 20-24, 1998, Pittsburgh, PA, USA, pp. 297–298 (1998)
49. Tauscher, L., Greenberg, S.: How people revisit web pages: empirical findings and implications for the design of history systems. International Journal of Human-Computer Studies **47**(1), 97–137 (1997)
50. Teevan, J., Dumais, S.T., Liebling, D.J.: A longitudinal study of how highlighting web content change affects people's web interactions. In: Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Atlanta, Georgia, USA, April 10-15, 2010, pp. 1353–1356 (2010)
51. Tyler, S.K., Teevan, J.: Large scale query log analysis of re-finding. In: Proceedings of the 3rd International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010, pp. 191–200 (2010)
52. Weinreich, H., Obendorf, H., Herder, E., Mayer, M.: Off the beaten tracks: exploring three aspects of web navigation. In: Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006, pp. 133–142 (2006)
53. Yang, H., Parthasarathy, S.: On the use of constrained associations for web log mining. In: O. Zaiane, J. Srivastava, M. Spiliopoulou, B. Masand (eds.) WEBKDD 2002 - Mining Web Data for Discovering Usage Patterns and Profiles, Lecture Notes in Computer Science, pp. 100–118. Springer (2003)

54. Yao, Y., Shi, L., Wang, Z.: A markov prediction model based on page hierarchical clustering. International Journal of Distributed Sensor Networks **5**(1), 89–89 (2009)
55. Zukerman, I., Albrecht, D.W., Nicholson, A.E.: Predicting users' requests on the www. In: Proceedings of the 7th International Conference on User Modeling, UM 99, Banff, Canada, pp. 275–284 (1999)

Author Biographies

George Papadakis is a Postdoctoral Researcher at the Management of Data, Information and Knowledge group (MaDgIK) of the University of Athens. Dr. Papadakis received his Diploma in Computer Engineering from the National Technical University of Athens (NTUA) and his Ph.D. degree in Computer Science from the Leibniz University of Hanover. Dr. Papadakis has worked at the NCSR Demokritos, the L3S Research Center, NTUA and the Research Center "Athena". His research interests include entity resolution and web data mining.

Ricardo Kawase works as a senior researcher at the L3S Research Center. His main research interests include Web personalization, contextualization, user modeling, personalization, technology-enhanced learning and crowdsourcing. His Ph.D. thesis at the Leibniz University of Hanover focused on the personalization and contextualization on the Web. He served as reviewer in several international conferences and has published over 50 scientific articles.

Eelco Herder works as a senior researcher at the L3S Research Center. His main research interests include Web personalization, user modeling, usability and HCI in general. His Ph.D. thesis at the University of Twente focused on the observation, modeling and prediction of user navigation on the Internet. He served as program chair for Hypertext 2014 and served as workshop, poster, publicity and local chair at various conferences, among which CHI 2012, UMAP 2010-2013 and Adaptive Hypermedia 2008.

Wolfgang Nejdl has been full professor of Computer Science at the University of Hannover since 1995. He was assistant professor in Vienna and associate professor at the RWTH Aachen. Prof. Nejdl heads the L3S Research Center as well as the Distributed Systems Institute / Knowledge Based Systems, and conducts research in the areas of search and information retrieval, information systems, semantic web technologies, peer-to-peer infrastructures, databases, technology-enhanced learning and artificial intelligence. He published more than 350 scientific articles and has been program chair, program committee and editorial board member of numerous international conferences and journals.