Enhancing Web Revisitation by Contextual Keywords

Tangjian Deng, Liang Zhao, and Ling Feng

Tsinghua National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
{dtj08,jing-zhao11}@mails.tsinghua.edu.cn, fengling@tsinghua.edu.cn

Abstract. Web revisitation is a common behavior supported by many web history tools. Taking advantages of access context (like time, location, concurrent activity), context-based search of previously accessed web pages is also being investigated, due to the fact that context under which information is accessed tends to be more easily to remember than content. To mimic users' memory recall, we present a way to automatically capture user's access context from user's concurrent computer programs, and manage it in a probabilistic context tree for each accessed web page in a life cycle. An algorithm for contextual keyword search of accessed web pages, together with a revisitation feedback mechanism, are also given. We evaluate the proposed method on synthetic data and through a 6-week user study. The comparisons of revisit precision and recall show our method outperforms the existing contextual search method *YouPivot*. In the user study, our method can also work as effectively as popular methods (like bookmark, browse history) in recall rate (over 90%), while with less average time cost (16.25 seconds) than that (38.66 seconds) of those methods to complete a web revisitation task.

Keywords: Web revisitation, context memory, contextual keyword search, revisitation feedback.

1 Introduction

Web Revisitation Support. The web is playing a significant role in people's daily activities in delivering information to one's fingertips. Among the common web behavior, revisitation of previously browsed web pages constitutes an important web access portion [5,2,21]. According to [17], over 58% of web pages accessed by 23 users within a 6-week period were revisited ones. The analysis of a 1-year web search by 114 users also revealed that around 40% of queries belong to revisitation requests [19]. To support web revisitation, a number of web history techniques and tools have been developed [14,1,10,22,3].

Bookmark. Manual bookmark of favorite web pages embedded in web browser is a traditional way to enable users to re-locate the visited pages. Users can also mark a specific part within a visited web page by the *Landmark* tool [13]. The *SearchBar* [15] allows users to organize their search keywords and clicked pages under different topics for easy navigation.

F. Daniel, P. Dolog, and Q. Li (Eds.): ICWE 2013, LNCS 7977, pp. 323–337, 2013.

History Tools. Web browsers maintain users' accessed URLs according to visit time (*e.g., today, yesterday, last week, etc.*). *Google Web History*[1] keeps users' search keywords and clicked pages, then puts them into different categories like image, news, normal page, and so on. Users can either navigate or search the history by keywords from accessed page titles/contents. The *Contextual Web History* [22] improves the visual appearance of web browser history by combining thumbnails of web sites and snippets of contents, assisting users to easily browse or search the history by time. The dynamic browser toolbar [11] can further recommend relevant visited pages according to the currently viewed page.

Re:Search Engine. The *Re:Search* system [18] can support simultaneous finding and re-finding on the web. When a user's query is similar to a previous query, *Re:Search* obtains the current results from an existing search engine, and fetches relevant previously viewed results from its cache. The newly available results are then merged with the previously viewed results to create a list that supports intuitive re-finding and contains new information.

Contextual Search. Access context is also exploited for web revisitation, due to the fact that context under which information is accessed sometimes is more easily to remember than content itself [4,20]. [9] developed a *YouPivot* system, which allows users to search the context they remember, so that users can see what was going on under that context. User's web activities are logged via the Chrome Extension and sent to *YouPivot*, which also pulls LastFM and Twitter data and retrieves calendar data via public ICS files. Also, the user can time-mark a moment worth remembering, and provide a description on it for contextual recall in *YouPivot*. [6,7] also developed a *ReFinder* system to allow users to manually annotate such access context as time, location, and concurrent activity for the visited web pages or local files, with which the users can pose structured re-find requests to the previously accessed web pages or files.

YouPivot and *ReFinder* are two closely related work of this study, with the following differences.

- Instead of prompting users to manually annotate access context as done by *ReFinder*, this paper proposes a method to automatically capture the access context through users' computer programs running before and after the access event. Historical access context is managed in probabilistic context trees, each linked to an accessed web page URL. Later users can revisit the web pages by contextual keywords. This is different from the *ReFinder* system which only supports structured context-based re-search.
- Unlike *YouPivot* which keeps context persistently, this study considers progressive evolution of historical access context trees, since along with human memory fade, the past context for recall will also degrade gradually. Revisitation requests can thus be more realistically interpreted and fast executed.
- To tailor to different users' revisitation habits, we incorporate a feedback mechanism during users' web revisitation to dynamically adjust historical context memory and relevant result ranking. This is not discussed in either *YouPivot* or *ReFinder*.

[1] `http://www.google.com/history`

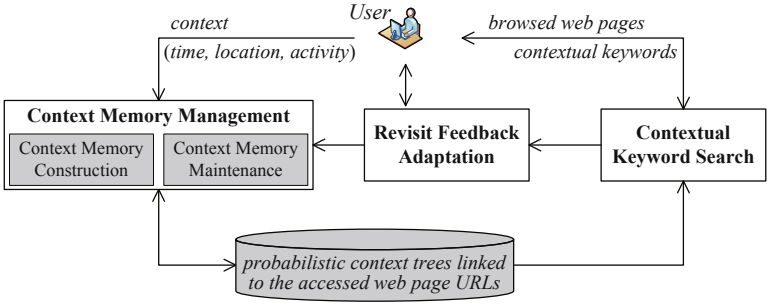


Fig. 1. Our web revisitation framework

Our Work. Fig. 1 outlines the basic idea and framework of our contextual keywords based web revisitation, consisting of three components: *context memory management, contextual keyword search,* and *revisit feedback adaptation.*

1) *Context Memory Management.* It captures and represents each access context as a *probabilistic context tree*, bounded with the corresponding accessed web page URL. The tree is comprised of contextual keywords, inferred automatically from user's running computer programs. The probability assigned to each keyword node reflects how likely the user will use the keywords for later revisit. The probabilistic context trees are organized as a context memory, evolving along with the elapsing time. The keyword nodes are measured by retention strengths, which will decrease as they age, leading to the circumstance that the contextual keywords may become from specific to general. Meanwhile, the keyword nodes' probabilities will decrease gradually due to memory decay.

2) *Contextual Keyword Search.* Since the mappings between web pages and probabilistic context trees have been built, we take the contextual keywords as input, and produce a list of web pages as output. For a contextual keyword search, we find out all the match trees with non-zero probabilities, where the trees are ranked based on their probabilities from high to low. Then the visited web pages linked by the trees are returned straightforwardly.

3) *Revisit Feedback Adaptation.* From the user's actions of web revisitation by contextual keywords, the revisited web pages and the revisit conditions (the match contexts) are recorded as user's revisit feedbacks, which are then used to guide the adjustments on context memory construction and maintenance.

We evaluate our approach by conducting two sets of experiments with synthetic data and through a 6-week user study. In the synthetic data experiment, we simulate the contextual search method *YouPivot* and use it as a baseline. The comparisons of revisit precision and recall show our method outperforms *YouPivot*, as our method can adapt to the user's revisitation habit. In the user study, the revisit recall rate of our revisit prototype is over 90%. On average, 16.25 seconds are needed to complete a web revisitation task with our method and 38.66 seconds with popular methods like bookmark, browse history, search engines, *etc.* The experimental results show that our prototype provides a

complementary effective solution in facilitating user's web revisitation through contextual keywords.

The rest of the paper is organized as follows. We address context memory construction and maintenance in Section 2. We present contextual keyword search in Section 3, and describe revisit feedback adaptation in Section 4. We evaluate our approach in Section 5 and conclude the paper in Section 6.

2 Context Memory Management

Context memory management component performs two tasks, *i.e.*, construction of context memory, followed by dynamic maintenance of the context memory.

2.1 Context Memory Construction

Three kinds of user's access context, *i.e.*, *access time, access location*, and *concurrent activity*, are considered in this study. *Access time* is determinate. *Access location* is obtained based on the IP address of user's computing device or his/her possible GPS information if available. We infer user's *concurrent activity* from his/her computer programs running before and after the page access as follows.

We continuously monitors the change of the current focus program window during the user's interacting with his/her computer. Each focus object held by the program window can be either a web page or not like a word document, a friend with whom the user is chatting online, *etc.* A focus window possesses a start time, an end time, and a focus time length.

Definition 1. *The user's computer activities is a sequence of focus windows, denoted as* $\mathcal{O} = \langle O_1, O_2, \ldots \rangle$*, where* O_i *(*$i \geq 1$*) denotes a quadruple (*t_{begin}*,* t_{end}*,* t_{focus}*, object) representing the start time, the end time, the focus time length and the focus object respectively. For any* $1 \leq i < j$*,*

(1) $O_i(t_{begin}) \leq O_j(t_{begin})$*;*

(2) $O_i(t_{focus}) \leq Length(O_i(t_{end}) - O_i(t_{begin}))$*;*

(3) if $O_i(object) = O_j(object)$*, then* $Length(O_j(t_{begin}) - O_i(t_{end})) > \tau_{gap}$*.*

Table 1. An example of the user's computer activities

No.	Start Time	End Time	Focus Time (sec)	Window Object
1	*2012/8/9 10:04:24*	*2012/8/9 10:09:47*	*96*	*MSN - Lily*
2	*2012/8/9 10:05:38*	*2012/8/9 10:16:38*	*128*	*eBay - jeans*
3	*2012/8/9 10:06:07*	*2012/8/9 10:06:20*	*13*	*eBay - shoes*
4	*2012/8/9 10:06:51*	*2012/8/9 10:13:03*	*199*	*eBay - shirt*
...	...	...	...	...

If the objects of any two focus windows are the same and the time gap between them is less than a threshold τ_{gap} (10 minutes), they will be merged together as one window. Table 1 demonstrates an example of the user's computer activities. In this study, if the focus time length of a browsed web page is greater than a threshold $\tau_{wf} = 30$ seconds, we consider it as a to-be-revisited web page. Note that τ_{wf} will adjust based on user's revisit feedback.

Definition 2. *Let w_p be a to-be-revisited web page, a time window $\mathcal{T}_W$ $(w_p(t_{begin}) - \Delta t_b, w_p(t_{end}) + \Delta t_e)$, a threshold of focus time length τ_{cf}. For every $O_c \in \mathcal{O}$ (the sequence of focus windows) overlaps with $\mathcal{T}_W$, namely, $O_c(t_{begin}) < (w_p(t_{end}) + \Delta t_e)$ and $O_c(t_{end}) > (w_p(t_{begin}) - \Delta t_b)$, if $O_c(t_{focus}) \geq \tau_{cf}$, then O_c is considered as an **associated context** for w_p.*

The associated contexts for w_p depend on the 3 parameters Δt_b, Δt_e and τ_{cf}. Initially, we set $\Delta t_b = \Delta t_e = 10$ minutes, $\tau_{cf} = 90$ seconds. They will adjust to the user's revisit feedbacks, and the details are described in Section 4.

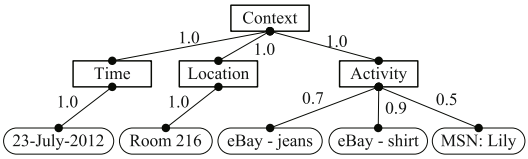


Fig. 2. Example of a probabilistic context tree

Consider the uncertain characteristic of user's memory, the obtained associated contexts are formulated into a probabilistic context tree. An example of context tree is shown in Fig. 2. Besides access time and location, each context extracted from user computer activities forms a leaf node of the context tree. The edge linking a child node to a parent node in the context tree has a probability in [0, 1]. It reflects the likelihood that the child node is used as a contextual recall cue. For simplicity, all the edges are set as probability 1.0, except the ones linking the activity leaf nodes and their parent nodes. We use the association probability between an activity context O_c and the to-be-revisited web page w_p as the probability of the edge linking O_c and its parent node. To compute the association probability, we consider four features: 1) the focus time length of O_c; 2) the appearing times of O_c; 3) the time distance between O_c and w_p; and 4) the content similarity between O_c and w_p, $sim(O_c, w_p) = \frac{TermCount(O_c \wedge w_p)}{TermCount(O_c)}$. We normalize the four values within the same context tree, denoted as f_1, f_2, f_3 and f_4, respectively, where $0 \leq f_i \leq 1$ $(i = 1, 2, 3, 4)$. Taking the four features into account, the initial association probability pr_0 between O_c and w_p is computed:

$$pr_0 = (f_1 + (1 - f_2) + (1 - f_3) + f_4)/4 \tag{1}$$

Intuitively, the longer the focus time length of the context and the more similar the context to the web page, the larger association probability between them,

while the appearing times of the context and the time distance between the context and the web page lead to the opposite case.

2.2 Context Memory Maintenance

Probabilistic context trees in the context memory evolves dynamically in life cycles to reflect the gradual degradation of human's context memorization as well as the contextual keywords that human users will use for recall. For each leaf node in a probabilistic context tree, both its value and its association probability will progressively decay with time.

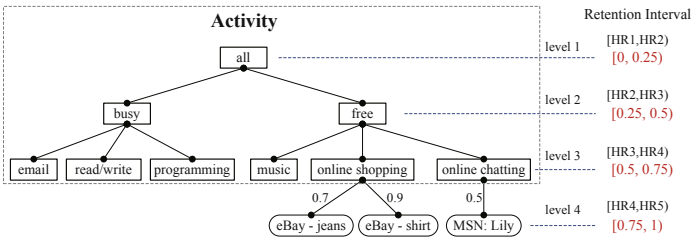


Fig. 3. An abstraction hierarchy of activity context

Taking the *actitivity* context type for example, we build an n-leveled abstraction hierarchy H, where lower-leveled activity values are more precise than upper-leveled ones, as shown in Fig. 3. The captured "*eBay-jeans*" activity value is initially located at the bottom level n. It will climb upwards along H, and finally disappear when reaching the top level 1. In order to quantitatively measure such a context value degradation process, we introduce and compute the *retention strength* $R \in [0, 1]$ of a context value v, based on which we determine its locating level in the hierarchy. Assume each hierarchical level has a reference retention interval $[HR_i, HR_{i+1})$ $(1 \leq i \leq n)$, which is evenly distributed in [0, 1] in this study. If v's retention strength R falls into $[HR_i, HR_{i+1})$, then the context value belongs to level i. According to the psychology studies [16], we define the retention strength of a context value in its context hierarchy as a function of the exponential in the square root of time.

For context value v captured and assigned web page association probability pr_0. It initially situates at the bottom level n of its n-leveled context hierarchy H. Let $[HR_i, HR_{i+1})$ be the retention interval of the i-th hierarchical level in H. At v's age t, the **retention strength** of v is defined as:

$$R(t) = r_0 \cdot e^{-\lambda\sqrt{t}} \tag{2}$$

where λ is the context memory decay rate, and r_0 is the initial retention strength of v, computed as: $r_0 = HR_n + (HR_{n+1} - HR_n) \cdot pr_0 \quad \in [HR_n, HR_{n+1})$.

Intuitively, the larger v's web page association probability pr_0 is, the better the context value v is remembered. Thus, pr_0 affects r_0 positively.

Assume after age $t_{01} \in [T_{min}, T_{max}]$, v will start to degrade from the bottom level n to its upper level n-1, where

$$t_{01} = T_{min} + (T_{max} - T_{min}) \cdot pr_0 \tag{3}$$

and T_{min} and T_{max} are user-dependent, and initially set to 14 days and 21 days in this study. The revisitation feedback mechanism will adjust the two settings based on user's revisitation requests and result ranking.

Putting HR_n and t_{01} into Equation 2, we get

$$\lambda = \frac{1}{\sqrt{t_{01}}} \ln \frac{r_0}{HR_n} = \frac{1}{\sqrt{T_{min} + (T_{max} - T_{min}) \cdot pr_0}} \ln \frac{r_0}{HR_n} \tag{4}$$

With r_0 and λ, we can compute the retention strength $R(t)$ of v at age t according to Equation 2. If it falls in the range of $[HR_i, HR_{i+1})$, then context value v degraded to the i-th level of H.

In a similar fashion, the web page association probability pr_0 of context value v also evolves along with the elapsing time.

$$Pr(t) = pr_0 \cdot e^{-\lambda\sqrt{t}} \tag{5}$$

After each degradation computation, the decay rate λ is adjusted to $\lambda = \frac{1}{\sqrt{T_{min}+(T_{max}-T_{min})\cdot Pr(t)}} \ln \frac{r_0}{HR_n}$.

3 Contextual Keyword Search

Given a set of contextual keywords as a user's revisit request $Q = \{k_1, k_2, \ldots, k_n\}$, we evaluate it over the context memory, and returns a ranked list of the match probabilistic context trees with their linked web pages as the final result. Evaluation of Q proceeds in two steps. Nodes containing contextual keywords are first identified. Their contribution to the final tree ranking is then computed.

We apply the Dewey encoding scheme to probabilistic context trees based on [8,23,12]. In our probabilistic context trees, the Dewey number of the root is actually the tree id. For each node v in a probabilistic context tree, we build an index according to its keywords. The mapping between the node and its all probabilities (from the root to it) is also kept, denoted as $\varphi : v \rightarrow PrLink$, *e.g.*, the node *"eBay - jeans"* $\rightarrow \langle 1.0,\ 0.7 \rangle$ as shown in Fig. 2. Through scanning the keyword inverted node lists, the match nodes are identified. The relationship between a node v and its local probabilistic keyword distributions *dist w.r.t.* Q is maintained, denoted as $\mu : v \rightarrow dist$. Assume $PNode \rightsquigarrow \{CNode_1, CNode_2, \ldots, CNode_m\}$, where $CNode_i$ is a relevant child node of $PNode$ against Q, $1 \leq i \leq m$. The computation of $PNode$'s keyword distributions is as follows: for each $1 \leq i \leq m$, $\mu : CNode_i \rightarrow dist$ is promoted by multiplying $Pr_{rev}(PNode \rightsquigarrow CNode_i)$ (the probability of the edge) and the part $0 \rightarrow \mu_0$ is

set to $1 - Pr_{rev}(PNode \rightsquigarrow CNode_i)$. Then $\mu : PNode \rightarrow dist$ is merged using a set of bitwise **OR** operations with $\mu : CNode_i \rightarrow dist$, *i.e.*, multiply $\mu : PNode$ with each part of $\mu : CNode_i$ and then add the product to the corresponding part (based on the bitwise **OR** operation) of $\mu : PNode$.

Since the tree id can be easily got from the Dewey code of a node, based on the match nodes, we can easily get the match context trees, whose probabilities of matching the revisit request Q can be computed accordingly. Assume there are m match nodes v_1, ..., v_m in a context tree, p be the lowest common ancestor of the m match nodes. We compute, promote and merge the keyword distributions of the m match nodes into that of the p node, denoted as $\mathcal{S} = \{v_1, ..., v_m\} \rightarrow \mathcal{S} = \{p\}$. During the promotion process, especially, for $\forall v \in \mathcal{S}$, if $\nexists c \in \mathcal{S}$ satisfying that c is a descendent of v, then v can be promoted to its parent v_p. The keyword distributions of v_p are created or updated at the same time. Then v is removed from $\mathcal{S}$, while v_p is put into $\mathcal{S}$ if $v_p \notin \mathcal{S}$. Let $Pr(path_p)$ be the product of the probabilities from the root to p, $p.\mu_{2^n-1}$ be the distribution $(2^n - 1) \rightarrow \mu_{2^n-1}$, then the probability that the context tree matches Q equals $Pr(path_p) \cdot p.\mu_{2^n-1}$.

Algorithm 1. Context-Based Revisit Algorithm

input:
a revisit request $Q = \{k_1, ..., k_n\}$ and a set of probabilistic context trees
output:
a ranked list of context trees $\mathcal{R}$ that match request Q

load node list $L = \{L_i\}$, $1 \leq i \leq n$, $\varphi : v \rightarrow PrLink$, determine the match context trees $T = \{ct_1, ct_2, ...\}$ based on L, create and update $\mu : v \rightarrow dist$;
for each $ct \in T$ **do**
  divide the match nodes of ct into $\mathcal{S} = \{\mathcal{S}_i\}$, $1 \leq i \leq \mathcal{L}$, $\mathcal{S}.num = \sum |\mathcal{S}_i|$;
  for $i = \mathcal{L}$; $i \geq 1$; $--i$ **do**
    if $|\mathcal{S}_i| = 0$ **then**
      continue;
    for each $v \in \mathcal{S}_i$ **do**
      if $\mathcal{S}.num = 1$ **then**
        $ct.prob =$ ComputeProb($v \rightarrow dist$, $v \rightarrow PrLink$);
        if $ct.prob > 0$ **then**
          insert ct into $\mathcal{R}$ according to $ct.prob$;
        $i = 0$; **break**;
      let p be the parent of v;
      if $p \notin \mathcal{S}_{i-1}$ **then**
        create $p \rightarrow PrLink$ and $p \rightarrow dist$, put p into $\mathcal{S}_{i-1}$;
      else
        update $p \rightarrow PrLink$ and $p \rightarrow dist$, decrease $\mathcal{S}.num$ by 1;
      remove v from $\mathcal{S}_i$;
return $\mathcal{R}$;

Example 1. Consider the context tree shown in Fig. 2 w.r.t. a revisit request {eBay, jeans}. Two match nodes "eBay - jeans" and "eBay - shirt" are got. The

first node with distributions {'11'→1, '10'→0, '01'→0, '00'→0} and the second node with distributions {'11'→0, '10'→1, '01'→0, '00'→0} need to promote to their parent node "Activity", whose distributions is computed as {'11'→0.7, '10'→0.27, '01'→0, '00'→0.03}. Hence, the probability of the context tree matching the revisit request is $1.0 \cdot 0.7 = 0.7$.

The algorithm for finding the match context trees and computing their probabilities is illustrated in Algorithm 1. It scans the keyword inverted node lists once and determines the elementary match context trees based on the match nodes. To compute the probability of a match context tree, it firstly divides the match nodes into different sets based on their hierarchical levels, and then promotes and merges the nodes one by one starting from the lowest level set. It stops the promotion process if there remains only one node, which is no other than the lowest common ancestor of the match nodes. The match context tree will be inserted into the result list at the right position if its probability *w.r.t.* the revisit request is larger than zero.

Complexity Analysis: (1) *Time.* Identifying the match context trees T (Line 1) takes $O(n \cdot |T|)$. Dividing the match nodes into different sets (Line 3) takes $O(\sum_{i=1}^{\mathcal{L}} |\mathcal{S}_i|)$. Promoting and merging the match nodes (Line 4 - 18) takes $O(\mathcal{L} \cdot \sum_{i=1}^{\mathcal{L}} |\mathcal{S}_i|)$. Thus, the total time cost is $O(n \cdot |T|) + O(|T| \cdot \mathcal{L} \cdot \sum_{i=1}^{\mathcal{L}} |\mathcal{S}_i|)$ $= O(|T| \cdot \mathcal{L} \cdot \sum_{i=1}^{\mathcal{L}} |\mathcal{S}_i|)$. Clearly, it depends on the number of match context trees, the depth of the context tree and the number of the match nodes. (2) *Space.* In computing the rankings of the match context trees, we need to store the match nodes temporally. So the additional space cost is $\sum_{i=1}^{\mathcal{L}} \beta \cdot |\mathcal{S}_i| \cdot T$, where β is the cost for storing a match node, T is the number of match context trees and $|\mathcal{S}_i|$ is the number of match nodes in the i^{th} level of a context tree.

4 Revisit Feedback Adaptation

As the outcome of context memory construction and maintenance will directly impact the actions of the user's web revisitation by contextual keywords, the user's revisit feedbacks should be taken into account in the on-going management of context memory, so as to provide more suitable contexts for the user to search.

The user's revisit feedbacks, denoted as $\mathcal{F} = (\mathcal{W}_R, \mathcal{C}_R)$, are comprised of a set of true revisit web pages $\mathcal{W}_R$ and a set of the corresponding revisit conditions (match contexts) $\mathcal{C}_R$, obtained from the user's revisit actions. Based on $\mathcal{F}$, some useful information depicting the user's revisit habit can be got, *e.g.*, the focus time lengths of the revisited web pages and the match contexts, as well as the time distance between them, *etc.* Besides, if the match contexts are at the second lowest hierarchical level, their current ages are recorded.

For convenient reference, we denote $\mathcal{H} = (W_F, C_F, \Delta T_B, \Delta T_E, D\tau)$ as the parameters characterizing the user's web revisitation habit, where W_F is the set of the focus time lengths of $\mathcal{W}_R$, C_F is the set of the focus time lengths of $\mathcal{C}_R$, ΔT_B and ΔT_E are the sets of the lengths of $(\mathcal{W}_R(t_{begin}) - \mathcal{C}_R(t_{end}))$ and $(\mathcal{W}_R(t_{end}) - \mathcal{C}_R(t_{begin}))$ respectively, and $D\tau$ is the set of ages at which $\mathcal{C}_R$ decay from the bottom level to the next upper level.

Assumption 1. *The parameters W_F, C_F, ΔT_B and ΔT_E of the user's revisit habit satisfy a normal distribution separately, namely, $W_F \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $C_F \sim \mathcal{N}(\mu_2, \sigma_2^2)$, $\Delta T_B \sim \mathcal{N}(\mu_3, \sigma_3^2)$, and $\Delta T_E \sim \mathcal{N}(\mu_4, \sigma_4^2)$.*

Upon the user's revisit actions, the mean values and standard deviations $W_F(\mu_1, \sigma_2)$, $C_F(\mu_2, \sigma_2)$, $\Delta T_B(\mu_3, \sigma_3)$, $\Delta T_E(\mu_4, \sigma_4)$ and the minimum and maximum values T_{min}, T_{max} of $D\tau$ will be updated accordingly.

Adjustments. The revisit feedback adaptation over context memory management refers to the adjustments of several key parameters: (1) $\tau_{wf} = \mu_1 - 2\sigma_1$; (2) $\tau_{cf} = \mu_2 - 2\sigma_2$; (3) $\Delta t_b = \mu_3 + 2\sigma_3$; (4) $\Delta t_e = \mu_4 + 2\sigma_4$; (5) $T_{min} = \min\{D\tau\}$; and (6) $T_{max} = \max\{D\tau\}$. The objectives of these adjustments are to capture the to-be-revisited web pages and the to-be-employed contexts as far as possible, and to maintain the context memory at more appropriate decay rates.

Reinforcement. Because the recall actions can often refresh the user's memory, during the evolution process, certain parts of the context memory are reinforced due to the user's revisit actions. Thus, we set the current time as the new born time of the involving contexts, and reset the decay rate $\lambda = \frac{1}{\sqrt{T_{min} + (T_{max} - T_{min}) \cdot pr_0}} \ln \frac{r_0}{HR_n}$ based on Equation 4.

5 Evaluation

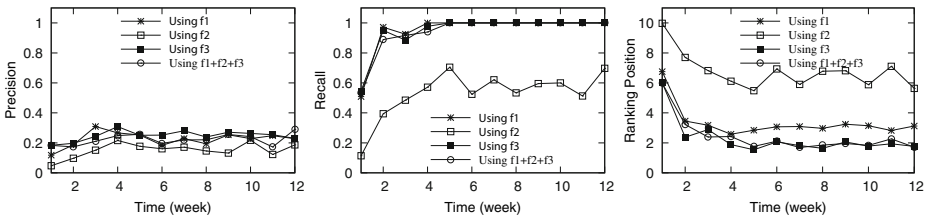
To evaluate our approach, we implemented a prototype called $\mathcal{R}$E$\mathcal{V}$isit and conducted two sets of experiments: with synthetic data and through a user study. The two experiments focus on two measurements: 1) revisit result quality (precision, recall and ranking); and 2) revisit response time. $\mathcal{R}$E$\mathcal{V}$isit is implemented in *C#*. The first experiment with synthetic data is conducted on a PC with 2.2 GHz Intel Core 2 Duo CPU, and 2 GB memory on Windows 7 OS.

5.1 Experiment on Synthetic Data

Design: We first build two extra components: 1) *data simulator*, to simulate the generation of a user's computer activities; and 2) *user simulator*, to simulate the user's memory over the generated data and the user's revisit actions, acts as a "real user". *Data simulator* generates 3-month data comprising of 7 activity types: *email, browsing, programming, read/write, music, online shopping* and *online chatting*. It separately generate a set of phrases (2 to 5 words) for each activity type. For a data item, the time span ($t_{end} - t_{begin}$) is a random value from 30 seconds to 15 minutes, the focus time (t_{focus}) is from 5 seconds to half of the time span, the activity type is randomly selected and the keywords are generated from the corresponding set of phrases, where the keywords' repetition rate is 3%, and the data type is also set randomly to be web page or not. In total, 27,824 data items are generated, where the web pages occupy about 57%.

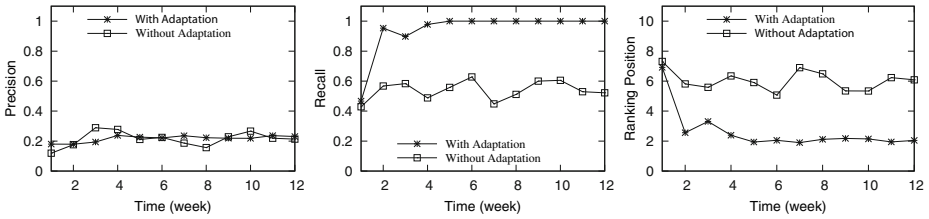
For each generated web page, if $t_{focus} \geq 30$ seconds, it will be stored as a candidate revisit target by *user simulator*, which will identify a set of associated contexts based on its own $\Delta t_b = 5$ minutes, $\Delta t_e = 15$ minutes, $\tau_{cf} = 60$

seconds, $T_{min} = 10$ days and $T_{max} = 28$ days. Every *period* (7 days), *user simulator* will randomly select a part of the the candidate revisit targets as the true to-be-revisited web pages. For each of them, it chooses 2 keywords from the corresponding stored contexts with higher association probabilities as a revisit request. Then the revisit requests are submitted to $\mathcal{R}$E$\mathcal{V}$isit. Meanwhile, $\mathcal{R}$E$\mathcal{V}$isit identifies each generated web page, and then build a probabilistic context tree for the possible to-be-revisited one, based on its own $\Delta t_b = \Delta t_e = 10$ minutes, $\tau_{cf} = 90$ seconds, $T_{min} = 14$ days and $T_{max} = 21$ days. Every *period*, $\mathcal{R}$E$\mathcal{V}$isit processes the revisit requests from *user simulator*, and then the relevant parameters are updated according to the revisit feedbacks.



(a) Average precision (b) Average recall (c) Average rank position

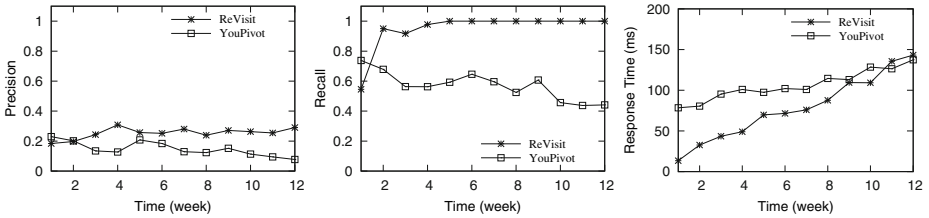
Fig. 4. Results on synthetic data using different features in memory construction



(a) Average precision (b) Average recall (c) Average rank position

Fig. 5. Results on synthetic data with/without revisit feedback adaptation

Results: The average precision, recall and ranking position of the revisit results are studied under different settings of context memory management, including applying various features in computing the association probabilities between the contexts and the targets, and with or without revisit feedback adaptation. Clearly, the adopted features will impact the revisit result quality, as demonstrated in Fig. 4. Note that the influence of the f_4 feature is not shown since the content similarity is very minimal in the synthetic data. Consider the feedback adaptation, as $\mathcal{R}$E$\mathcal{V}$isit at first does not grasp the revisit habit of the "real user" and probably not capture the most associated contexts for later revisit,



(a) Average precision (b) Average recall (c) Average response time

Fig. 6. Comparison results between $\mathcal{R}$E$\mathcal{V}$isit and *YouPivot* on synthetic data

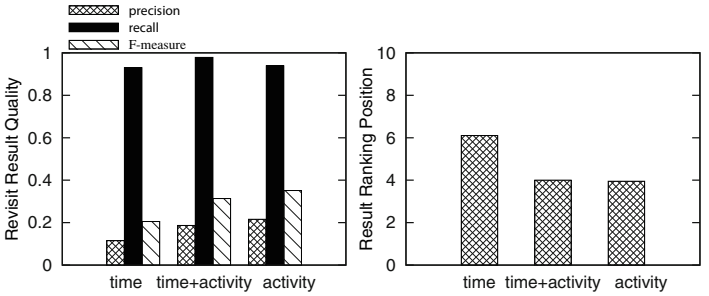
the result quality is not so good, shown in Fig. 5. As time goes by, since $\mathcal{R}$E$\mathcal{V}$isit adapts to the revisit habit, the revisit quality become better, especially that the recall rate almost keeps at 100%.

We also simulate the contextual search method *YouPivot* (as we did not get the source code) and use it as a baseline. Fig. 6 shows the comparisons between our method and *YouPivot* on revisit result quality and response time. The ability of feedback adaptation clearly makes our method to achieve higher precision and recall than that of *YouPivot* as time goes by. On the other side, as the search space (context memory) of our method is smaller than that (all data) of *YouPivot*, it takes less time cost for our method than *YouPivot* to perform revisit actions. As the generated data is continuously growing, more and more possible to-be-revisited web pages are recorded, and the context memory size becomes larger and larger, so it takes more time to do re-finding.

5.2 User Study

Set-Up: A 6-week user study was conducted to investigate the performance of $\mathcal{R}$E$\mathcal{V}$isit in real case, with 16 participants (8 male and 8 female, aged between 21 and 37), whose computers were installed with $\mathcal{R}$E$\mathcal{V}$isit. During that period, participants were asked to freely re-find the previously visited web pages with $\mathcal{R}$E$\mathcal{V}$isit, which kept the re-finding details automatically. For comparison, they were also asked to re-find the same web pages by popular methods like browsing or searching history list, using search engine, bookmark, *etc.* and meanwhile record the details in the re-finding process, such as revisit method, input-keywords, the response time, *etc.* The user study with $\mathcal{R}$E$\mathcal{V}$isit gathered 864 web revisitation records in total, 54 records per participant in average.

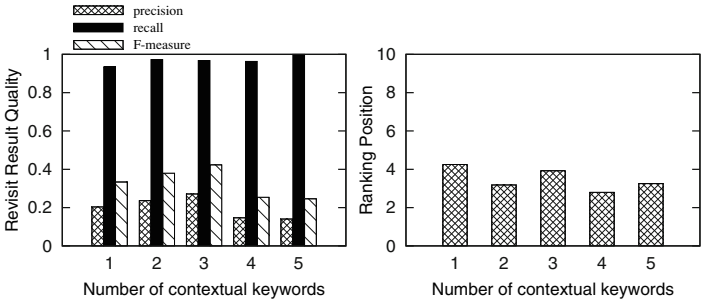
Results: The participants used time, time + activity, activity as re-finding contextual cues at the percentage of 3.97%, 6.29% and 89.74%, respectively. It indicates that users tend to remember activity more often than time. Note that place was not referred to in the user study, the reason is probably that participants did the experiment with IP-fixed computers. Fig. 7 shows the result quality of $\mathcal{R}$E$\mathcal{V}$isit by using different contexts as revisit requests. Since users tend to forget time more easily than activity, using time only to revisit works poorer than the other cases. While activity works quite well as re-finding cues, since richer



(a) result quality (b) ranking position

Fig. 7. The result quality of using *time*, *time+activity* and *activity* with $\mathcal{R}$E$\mathcal{V}$isit

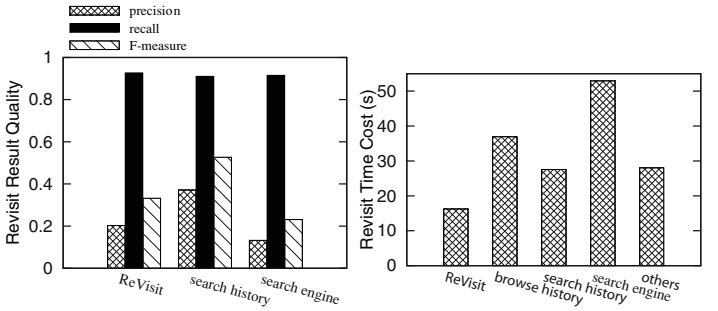
information makes it more easily to be remembered and distinguished. For the ranking positions of the revisited targets, time-only also works poorer than the other two cases. Participants were more likely to use less than 3 contextual keywords in a revisit request, where 1-keyword occupies 55.87%, 2-keyword occupies 31.28%, 3-keyword occupies 8.66%, and the remainder is occupied by more than 3 keywords. The result quality of using different number of contextual keywords with $\mathcal{R}$E$\mathcal{V}$isit is shown in Fig. 8. It is interesting to discover that the precision is not proportional to keyword number. The reason is that the 4- and 5-keyword revisit requests often contain time, which can not play well in revisitation.



(a) result quality (b) ranking position

Fig. 8. The result quality of using different number of keywords with $\mathcal{R}$E$\mathcal{V}$isit

The overall performance comparison between $\mathcal{R}$E$\mathcal{V}$isit and popular methods is shown in Fig. 9. The precision rate of $\mathcal{R}$E$\mathcal{V}$isit is lower than that of searching history. It is mainly because $\mathcal{R}$E$\mathcal{V}$isit supports general matching, and participants tended to revisit by general contextual keywords like music, shopping, chatting and so on, and thus the result list returned by $\mathcal{R}$E$\mathcal{V}$isit was sometimes longer. The average time cost for a revisit request shows that $\mathcal{R}$E$\mathcal{V}$isit outperforms the traditional methods. The reason includes several aspects. The long history list



(a) result quality (b) time cost

Fig. 9. Performance comparison between $\mathcal{R}$E$\mathcal{V}$isit and popular methods

often required users to spend a bit more time to re-locate the desire targets, and participants sometimes even gave up browsing when they could not find the target after several minutes. Searching history needs exact match, and participants had to try a few more times if they could not remember the keywords very well. While the query results and their rankings are frequently updated within the search engine, participants sometimes felt difficult to get the targets.

6 Conclusion

In this work, we propose a method to automatically construct an adaptive and evolutive context memory based on user's concurrent computer programs, supporting user's web revisitation by contextual keywords. Access context is formulated as a probabilistic context tree for each possible to-be-revisited web page. Context memory evolves as the elapsing time and adjusts according to the user's revisit feedbacks. The proposed method is evaluated by an experiment on synthetic data and a 6-week user study. Our experimental results show that it can adapt to the user's revisit habit, and contextual keywords based web revisitation offers another simple yet effective solution since it is closer to the way that human recalls information by context. As future work, we would like to explore the more appropriate and precise contextual keywords for the user when constructing probabilistic context trees, since the contextual keywords contained in the context trees directly influence the user's web revisitation action. Also, the preferable contexts would be explored based on the user's revisit activities.

Acknowledgments. The work is supported by National Natural Science Foundation of China (60773156, 61073004), Chinese Major State Basic Research Development 973 Program (2011C B302203-2), Important National Science & Technology Specific Program (2011ZX0 1042-001-002-2), and research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

References

1. Abrams, D., Baecker, R., Chignell, M.: Information archiving with bookmarks: personal webspace construction and organization. In: CHI, pp. 41–48 (1998)
2. Adar, E., Teevan, J., Dumais, S.T.: Large scale analysis of web revisitation patterns. In: CHI, pp. 1197–1206 (2008)
3. Capra, R., Perez-Quinones, M.A.: Using web search engines to find and refind information. IEEE Computer 38(10), 36–42 (2005)
4. Chen, Y., Jones, G.: Integrating memory context into personal information refinding. In: The 2nd Symposium on Future Directions in Info. Access (2008)
5. Cockburn, A., Greenberg, S., Jones, S., Mckenzie, B., Moyle, M.: Improving web page revisitation: analysis, design and evaluation. IT & Society 1(3), 159–183 (2003)
6. Deng, T., Zhao, L., Feng, L., Xue, W.: Information re-finding by context: a brain memory inspired approach. In: CIKM, pp. 1553–1558 (2011)
7. Deng, T., Zhao, L., Wang, H., Liu, Q., Feng, L.: Refinder: a context-based information re-finding system. IEEE TKDE (August 14, 2012) (preprint)
8. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: ranked keyword search over xml documents. In: SIGMOD, pp. 16–27 (2003)
9. Hailpern, J., Jitkoff, N., Warr, A., Karahalios, K., Sesek, R., Shkrob, N.: Youpivot: improving recall with contextual search. In: CHI, pp. 1521–1530 (2011)
10. Jones, W., Bruce, H., Dumais, S.: Keeping found things found on the web. In: CIKM, pp. 119–126 (2001)
11. Kawase, R., Papadakis, G., Herder, E., Nejdl, W.: Beyond the usual suspects: context-aware revisitation support. In: ACM Conference on Hypertext and Hypermedia, pp. 27–36 (2011)
12. Li, J., Liu, C., Zhou, R., Wang, W.: Top-k keyword search over probabilistic xml data. In: ICDE, pp. 673–684 (2011)
13. MacKay, B., Kellar, M., Watters, C.: An evaluation of landmarks for re-finding information on the web. In: CHI 2005 Extended Abstracts, pp. 1609–1612 (2005)
14. Mayer, M.: Web history tools and revisitation support: a survey of existing approaches and directions. Foundations and Trends in HCI 2(3), 173–278 (2009)
15. Morris, D., Morris, M.R., Venolia, G.: Searchbar: a search-centric web history for task resumption and information re-finding. In: CHI, pp. 1207–1216 (2008)
16. Rubin, D.C., Wenzel, A.E.: One hundred years of forgetting: a quantitative description of retention. Psychological Review 103(4), 734–760 (1996)
17. 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, 97–137 (1997)
18. Teevan, J.: The re:search engine: simultaneous support for finding and re-finding. In: UIST, pp. 23–32 (2007)
19. Teevan, J., Adar, E., Jones, R., Potts, M.: Information re-retrieval: repeat queries in yahoo's logs. In: SIGIR, pp. 151–158 (2007)
20. Tulving, E.: What is episodic memory? Current Directions in Psychological Science 2(3), 67–70 (1993)
21. Tyler, S., Teevan, J.: Large scale query log analysis of re-finding. In: WSDM, pp. 191–200 (2010)
22. Won, S.S., Jin, J., Hong, J.I.: Contextual web history: using visual and contextual cues to improve web browser history. In: CHI, pp. 1457–1466 (2009)
23. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest lcas in xml databases. In: SIGMOD, pp. 527–538 (2005)