Dealing with Context Ambiguity in Context-Based Information Re-finding

Tangjian Deng, Liang Zhao, and Ling Feng

Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
{dtj08,jingzhao-11}@mails.tsinghua.edu.cn, fengling@tsinghua.edu.cn

Abstract. Inspired by human memory that context can often play as important recall cues, access context under which information is previously accessed is being exploited to assist users in information re-finding, *e.g.*, allow users to search their accessed web pages by context. Due to the nature that human memory will decay and could be impacted by interference which will lead to context misremembering, users may refer to unreliable contextual cues in context-based re-finding. In this paper, we focus on how to re-find users' desired results under the circumstance of users' misremembering of precise contextual retrieval cues. To this end, we first categorize three kinds of ambiguity in context-based information recall queries, which are *context degradation, context confusion,* and *context error*, and organize user's access context in associated probabilistic context trees. We then propose an approximate matching approach to deal with users' re-finding requests (formed as contextual keywords) carrying possible ambiguity by taking advantages of context associations, which are extracted guided by human memory's decay and interference characteristics. Experimental results on both synthetic and real data confirm the effectiveness of our solution that can enable users to achieve a good performance in ambiguous context-based information re-finding.

Keywords: Ambiguous context, context association, information re-finding.

1 Introduction

Re-finding previously accessed information is a common behavior in people's daily lives [22,23]. To support web revisitation, a number of web history techniques and tools like bookmarks and history tools are developed [16]. Considering that context under which information was accessed is sometimes more easily to remember than data content itself [3,25], access context like *time, location, concurrent activity, etc.* has also been exploited to enhance information re-finding [11,5,6,4]. "*Re-find the African sweet recipe I saw during a trip to Africa in 1998*" is a context-based information re-finding example. In the literature, [11] developed a *YouPivot* system, which allows users to search through their digital history through the context they remember. The *ReFinder* system [5,6] leverages human's natural recall characteristics and builds a query-by-context model over a context memory snapshot, linking to the previously accessed information.

R. Meersman et al. (Eds.): OTM 2013, LNCS 8185, pp. 685–702, 2013.

So far, context-based re-finding efforts in the literature target at precise answers, which are only concerned with the data matching contextual keywords exactly. The ambiguity of human memory about context as re-finding cues, however, is not considered. In fact, human memory exhibits life-cycle degradation nature, and human's recall process may also easily be influenced by various interferences [24]. For instance, newly access events may better be recalled than old ones. The ability to recall a certain piece of information from memory could be impaired by either the newly learned or previously learned information [13,26]. Some nontarget items that are similar to the target ones may compete with the target ones as potential responses in memory recall [20,15]. This may lead to some mismemorized and ambiguous contextual keywords that are used as re-finding requests by users. To illustrate, let's see a real case.

Lily wants to buy a sofa. After comparing many similar styles of sofas for a while, she finally selects the one which she saw at 2013/4/17 when listening to Adele's songs on the Internet, for that one had a good discount. Unfortunately, she cannot re-locate it any more on the web by simple "sofa" keyword, since she forgets any useful information about it, such as detailed title, name of the online shop, etc. What Lily can remember is that she was listening to some music someday in April when coming across that sofa. Finally Lily has to recall that sofa web page through ambiguous contextual keywords (`in-April, listen-to-music`), rather than the exact ones (`2013/4/17, listen-to-Adele-music`).

Due to human memory decay and interference nature, ambiguous contextual keywords may be used in information recall requests. We categorize such context ambiguity into the following three kinds.

- *Context Degradation.* Owing to human memory decay, contextual keywords for searching degrade to a more general level, *e.g.*, instead of giving the exact date `2013/4/17`, the user may only recall via an approximate date `in-April`.
- *Context Confusion.* Due to memory interference and lots of information access events, access context mixes up, *e.g.*, `in-April` is mixed with `in-March`. This happens frequently when the events occur temporally close by. Also, access context may be interfered with each other due to context similarity, *e.g.*, the user may input `listen-to-Ying-music` as the concurrent activity rather than the correct one `listen-to-Adele-music`, since both Ying's and Adele's songs are listened frequently by the user. This may also happen due to the similarity in the accessed information contents themselves.
- *Context Error.* Unlike context confusion, the user may take some non-access-context wrongly as an access context for information re-finding. For example, the use may input `in-April` to query the information which she accessed in March, while no information access happened in April at all.

From the re-finding system perspective, it is not easy to get users' desired results, given such ambiguous context-based re-finding requests, which would often result in plenty of irrelevant results or null results. It would be quite frustrating for users, as unlike information finding, information re-finding is a more directed process, where users have already seen the information before, and can recognize the popped-up target [1]. For the existing context-based re-finding systems, this

usually means that the user is forced to try repeatedly with alternative context values until it finally matches the desired data. If the user fails in recalling the access context, then even this solution is infeasible. The aim of this paper is to support users' ambiguous queries for context-based information recall. To the best of our knowledge, this is the first attempt in the literature to deal with ambiguous context-based information re-finding requests. The main contributions of this paper lie in the following three aspects.

- We propose an approximate contextual search solution to deal with users' re-finding requests with ambiguous contextual retrieval cues.
- Guided by human memory decay and interference characteristics, we build three types of context associations between user's contextual retrieval cues: neighborhood associations, similarity associations, and inference associations, to facilitate ambiguous context-based information re-finding.
- We implement an approximate matching algorithm by utilizing the constructed context associations, and conduct experiments on both synthetic and real data to evaluate its effectiveness.

Our experimental results show that approximate matching can achieve a much better recall rate (about 90% and 80% in synthetic and real data experiments respectively) than exact or partial matching (below 60% in both synthetic and real data experiments) under the situation of query requests carrying mistaken information, and it acts similarly to or better than the other two matching methods for precision rate. Whereas, approximate matching takes a bit more time than exact matching.

The remainder of the paper is organized as follows. We review some closely related work in Section 2. We then present a general framework for flexible context-based information re-finding in Section 3. Two important components (*i.e., associated context memory construction* and *ambiguous contextual keyword search*) are addressed in Section 4 and 5, respectively. We evaluate the performance of our solution in Section 6, and conclude the paper in Section 7.

2 Related Work

Context-Based Re-finding. There have been recent studies on context-based information re-finding, incorporating contextual memory cues [3,9]. Dumais *et al.* [8] developed a system *Stuff I've Seen* to facilitate personal information reuse by building index for what a person has seen, and using some cues like file-type, access date, and author for filtering and sorting results. Soules and Ganger [21] presented a file search tool that combines content-based search with contextual information (temporal relationships between files) which is gathered from user activities (system file calls). There are two steps in using this file search tool: first locates files through content-based search and second, extends those results with contextually related files. Chen *et al.* [2] also built a desktop search tool that exploits semantic associations among files, mining from contents such as similar-to relationship and users' such operations as jump-to, copy-from, same-task, and so

on. Hailpern *et al.* [11] developed a *YouPivot* system, which keeps and visualizes access context and visited web pages, and allows users to search the context they remember, so that users can see what was going on under that context. Deng *el al.* [5,6] developed a *ReFinder* system to allow users to manually annotate such access context as time, location, and activity for the visited web pages or local files, with which users can pose structured re-finding requests to previously accessed web pages or files. Besides, an approach for web revisitation by context is also presented in [4], which automatically captures user's access context and manages it in a probabilistic context tree for each accessed web page. These existing methods did not consider the context ambiguity in users' re-finding requests and the issue of approximate matching in context-based information re-finding.

Approximate Querying. The techniques of approximate matching in information retrieval and database querying have been explored extensively in the literature [17,18,14,12,7,19]. Commonly, in order to equip with vague retrieval capabilities, these methods employ a distance or similarity function to compare data objects. For example, the predefined distances between values of the same database domain, the edit distance between two strings, and the cosine similarity between two vectors, *etc.*, are frequently adopted. Two data objects are considered to be similar if their similarity score is greater than a predefined threshold value, and the similar objects are considered as the relevant results for the query request. In addition, a priori knowledge about synonyms sometimes is also utilized in finding relevant data. Basically, the approximate matching techniques of these existing work take two aspects into account in performing data comparison: 1) content-based matching, *i.e.*, focus on the data values; and 2) structure-based matching, *i.e.*, focus on the structure in which data is represented, *e.g.*, path, tree, graph, *etc.* Differing from these existing work, our work exploits context associations in approximate matching in order to refrain from the influences of users' misremembering on contextual retrieval cues.

3 The Framework

Fig. 1 outlines the basic idea and framework of our approach to process users' ambiguous context-based re-finding requests. It is comprised of two major components, namely, *associated context memory construction* and *ambiguous contextual keyword search*.

– *Associated context memory construction.* Contextual cues for re-finding are managed in a set of probabilistic context trees (context memory) which link to accessed target items. As shown in Fig. 2, three kinds of user's access context, *i.e.*, access time, access location, and concurrent activity, are considered. Based on human memory interference and misremembering characteristics, we construct three types of context associations between probabilistic context trees' keyword nodes: Neighborhood Associations (NA), Similarity Associations (SA) and Inference Associations (IA). NAs represent that the two

associated nodes are neighbors according to their occurring time. SAs are constructed based on the similarity between the two nodes' contextual keywords. IAs refer to associations that one node is inferred by the other.

– *Ambiguous contextual keyword search.* Given a set of associated probabilistic context trees and an ambiguous query request consists of a set of contextual keywords, our idea is to exploit the associations between context trees' keyword nodes and apply them in getting the matched context trees, which include the ones containing all the query keywords (exact matching) and the ones associating with all the query keywords (approximate matching). A contextual keyword is considered to be associated by a context tree if there exists an association between one node of the tree and one node of another tree which exactly contains the keyword. Since the mappings between accessed information and probabilistic context trees have been built, we can easily return the final results to users.

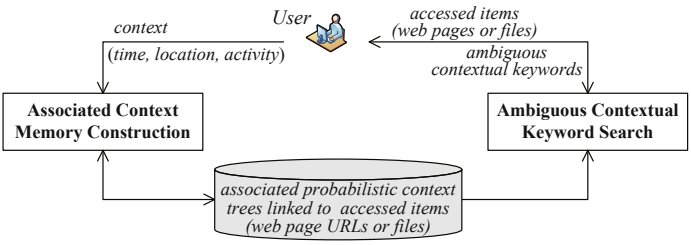


Fig. 1. The framework for ambiguous context-based information re-finding

4 Associated Context Memory Construction

In this section, we present the construction of context associations between probabilistic context trees.

4.1 Probabilistic Context Tree

For an accessed item (web page or file), the access context like time, location and activity is organized in a probabilistic context tree linking to the target item. The details of how to construct probabilistic context trees are addressed in our previous work [4]. Access time is determinate. Access location is obtained based on the IP address of user's computing device or his/her possible GPS information. Concurrent activity is inferred from user's computer applications running before and after the target item access. It is depended on a sliding time window that encompasses the related contexts, which are represented as contextual keywords extracted from focused windows's title. An example of probabilistic context tree is demonstrated in Fig. 2. The leaf nodes attached with contextual keywords are

also called keyword nodes, each of which is assigned by a probability from 0 to 1 reflecting the potential that the user will refer to the keywords for re-finding, The probability is set based on 4 factors: 1) the context's focused time length; 2) the context's frequency; 3) the time distance between the context and the access target; and 4)the content similarity between the context and the access target, where the longer the context's focused time length and the more similar the context to the target, the larger the probability between them, while the context's frequency and the time distance between them lead to the opposite case. Note that the hierarchies of time, location and activity at the middle level of the context tree are used to process generalized queries.

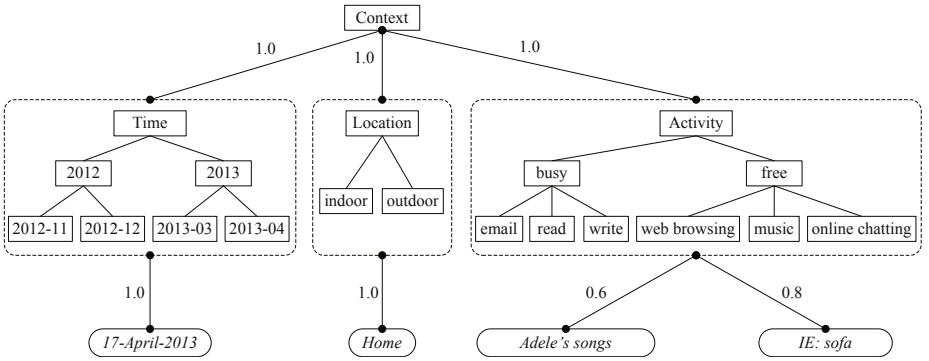


Fig. 2. Example of a probabilistic context tree

4.2 Building Context Associations

As mentioned before, in this work we consider three types of context associations between probabilistic context trees, the details of which are described as follows.

Neighborhood Associations (NA). The related contextual cues of the to-be-revisited data items are organized and formed as a sequence of probabilistic context trees $\langle ct_1, ct_2, \ldots \rangle$. For every $i \geq 1$, ct_i that links to an accessed data item has a start time t_{start} and an end time t_{end}.

Definition 1. *Let ct_i, ct_j be two probabilistic context trees, where ct_i occurs before ct_j ($ct_i(t_{start}) < ct_j(t_{start})$). If $Length(ct_j(t_{start}) - ct_i(t_{end})) \leq \tau_d$ (a predefined time length threshold), then ct_i and ct_j are neighbors to each other.*

Note that if $ct_j(t_{start})$ is before $ct_i(t_{end})$, $Length(ct_j(t_{start}) - ct_i(t_{end}))$ would be negative. Here we set $\tau_d = 30$ min. The contextual information of two neighbor trees may be remembered confusedly by the user. For example, assume ct_i and ct_j are two context trees linking to the data items d_i and d_j respectively. k_i and

k_j are two contextual keywords of two neighbors ct_i and ct_j separately, and the user may remember that d_i is linked by k_j or d_j is linked by k_i.

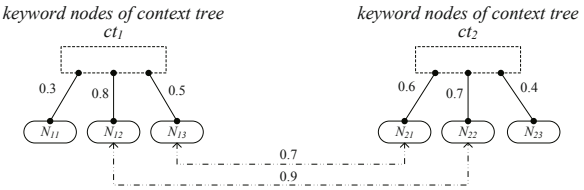


Fig. 3. Neighborhood associations between contextual keyword nodes

To overcome such problem, we construct NAs for context trees. Assume two context tree ct_1 and ct_2 are neighbors, where ct_1 has three keyword nodes N_{11}, N_{12} and N_{13}, and ct_2 also has three keyword nodes N_{21}, N_{22} and N_{23}. We do not need to construct NAs for all keyword nodes (otherwise there will be $3 \cdot 3 = 9$ NAs in Fig. 3 in total), since some of which would not be referred to by users in most cases. We should construct NAs for the ones that will be utilized by users with the most potential, *e.g.*, the top two keyword nodes with the biggest probabilities. Moreover, we can divide the four involving keyword nodes into two pairs and build only one NA for each pair, as shown in Fig. 3. Based on NAs, the user can get ct_2 through ct_1's keyword nodes and vice versa. Consider that some keyword nodes may be associated more closely, while some less closely, we define an association strength for each NA.

Definition 2. *Let N_i and N_j be two keyword nodes of two neighbor context trees respectively, $N_i(t_{start})$ and $N_j(t_{start})$ denote the start time of the corresponding context, τ_d is a time length threshold used in Definition 1, the association strength between N_i and N_j is computed as:*

$$NA\_Strength(N_i, N_j) = 1.0 - \frac{|N_i(t_{start}) - N_j(t_{start})|}{2\tau_d} \tag{1}$$

$NA\_Strength(N_i, N_j)$ indicates the closer the two keyword nodes, the larger their association strength. Taking the example of Fig. 3, assume $|N_{12}(t_{start}) - N_{22}(t_{start})| = 6$ min, $|N_{13}(t_{start}) - N_{21}(t_{start})| = 18$ min, then $NA\_Strength(N_{12}, N_{22}) = 1.0 - \frac{6}{2 \cdot 30} = 0.9$, $NA\_Strength(N_{13}, N_{21}) = 1.0 - \frac{18}{2 \cdot 30} = 0.7$.

Similarity Associations (SA). To mimic the phenomenon that the similar items may compete with each other in human memory recall, we construct SAs for the keyword nodes whose contextual keywords are similar to a certain extent. Like NAs, we do not build SAs for each keyword node, but rather choose the ones with the biggest probabilities (*e.g.*, the top two ones) to construct their SAs. We first define a function to measure the similarity of two keyword nodes.

Definition 3. *Let $TermCount(N)$ be a function computing the number of contextual keyword terms of a keyword node N. For N_i and N_j, their similarity, denoted as $C\_Sim(N_i, N_j)$, is computed as:*

$$C\_Sim(N_i, N_j) = \frac{TermCount(N_i \cap N_j)}{TermCount(N_i \cup N_j)} \tag{2}$$

Clearly, $0 \leq C\_Sim(N_i, N_j) \leq 1$. If the keywords of N_i and N_j are completely different, then $C\_Sim(N_i, N_j) = 0$. While at the other extreme, if their contextual keyword terms are exactly the same, then $C\_Sim(N_i, N_j) = 1$.

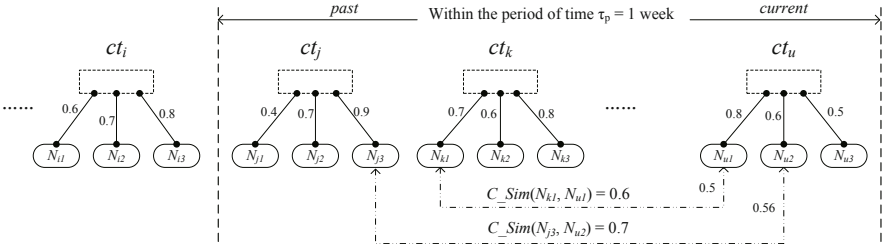


Fig. 4. Similarity associations between contextual keyword nodes

For a newly emerging context tree, it's keyword nodes may be associated by one or more keyword nodes of previous context trees due to their similarities. In this work we focus on the context trees over a period of time (denoted as τ_p) in building SAs for a given keyword node.

Definition 4. *Let N_u and N_j be two keyword nodes of two context trees respectively, satisfying $0 \leq |N_u(t_{start}) - N_j(t_{start})| \leq \tau_p$ and $\theta_{sim} \leq C\_Sim(N_u, N_j) < 1$, the association strength between N_u and N_j is computed as:*

$$SA\_Strength(N_u, N_j) = (1 - \frac{|N_u(t_{start}) - N_j(t_{start})|}{2\tau_p}) \cdot C\_Sim(N_u, N_j) \tag{3}$$

Consider the circumstance that the closer the two similar keyword nodes, the more likely they will be associated, we set τ_p as the past week ($\tau_p = 1$ week), as shown in Fig. 4. For a current keyword node N_u and a previous keyword node N_j within τ_p, if their similarity $\theta_{sim} \leq C\_Sim(N_u, N_j) < 1$ (where θ_{sim} is a predefined a threshold value, here we set $\theta_{sim} = 0.5$), then an SA will be built for N_u and N_j. Note that there is no need to build association between N_u and N_j if $C\_Sim(N_u, N_j) = 1$. Taking the example of Fig. 4, assume $|N_{u2}(t_{start}) - N_{j3}(t_{start})| = 4$ days, $|N_{u1}(t_{start}) - N_{k1}(t_{start})| = 1$ day, $C\_Sim(N_{u2}, N_{j3}) = 0.7$, $C\_Sim(N_{u1}, N_{k1}) = 0.6$, then $SA\_Strength(N_{u2}, N_{j3}) = (1 - \frac{4}{2 \cdot 7}) \cdot 0.7 = 0.5$, $SA\_Strength(N_{u1}, N_{k1}) = (1 - \frac{1}{2 \cdot 7}) \cdot 0.6 = 0.56$.

Inference Associations (IA). Some contextual cues that did not appear actually are inferred to exist owing to users' past experiences. It often occurs in the cases that the involving target items share some common features. Assume there are two target items T_i and T_j linked by context trees ct_i and ct_j respectively, where N_i is one of ct_i's keyword nodes. Because of the similarity of T_i and T_j, N_i is considered as one of ct_j's keyword nodes by the user, *i.e.*, the user would refer to the contextual keywords of N_i in re-finding the target T_j. Towards such circumstances, we particularly construct IAs for the involving context trees. For example, we would build a virtual keyword node N_j with null value for ct_j, and N_i is then linked to N_j by an IA. In this case, we say N_j is inferred by N_i.

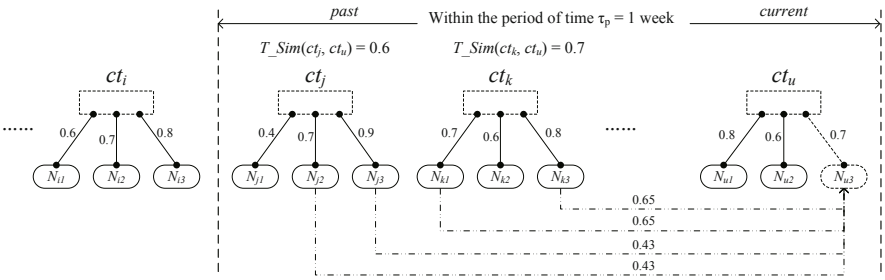


Fig. 5. Inference associations between contextual keyword nodes

For a newly emerging context tree, we first need to identify the context trees to which the one should be linked by IAs. Like SAs, we focus on the context trees over the period of time τ_p in building IAs. For a previous context tree ct_j within τ_p and a current context tree ct_u, we denote $T\_Sim(ct_u, ct_j)$ as the similarity of their linking target items, which can be computed based on Formula 2. If $T\_Sim(ct_u, ct_j) \geq \theta_{sim}$, IAs should be built between ct_u and ct_j. Hence, for the current context tree ct_u, we can obtain a set of context trees $TS = \{ct_j | T\_Sim(ct_u, ct_j) \geq \theta_{sim}$ and $|ct_u(t_{start}) - ct_j(t_{start})| \leq \tau_p\}$. The next step is to build a virtual keyword node N_u for ct_u, and its probability in the context tree denoted as $Pr(N_u)$ is computed as:

$$Pr(N_u) = \max\{T\_Sim(ct_u, ct_j) | ct_j \in TS\} \tag{4}$$

For each element of TS, we choose its two keyword nodes with the biggest probabilities to construct their IAs with the virtual keyword node. We also define an association strength for each IA.

Definition 5. *Let N_u be the virtual keyword node of a context tree ct_u, N_j be a keyword node of a context tree ct_j, the association strength of IA between N_u and N_j is computed as:*

$$IA\_Strength(N_u, N_j) = (1 - \frac{|ct_u(t_{start}) - ct_j(t_{start})|}{2\tau_p}) \cdot T\_Sim(ct_u, ct_j) \tag{5}$$

An example of IAs is demonstrated in Fig. 5, where ct_u is the current context tree, N_{u3} is a virtual keyword node, $|ct_u(t_{start}) - ct_j(t_{start})| = 4$ days, $|ct_u(t_{start}) - ct_k(t_{start})| = 1$ day, $T\_Sim(ct_u, ct_j) = 0.6$, $T\_Sim(ct_u, ct_k) = 0.7$, then $IA\_Strength(N_{u3}, N_{j2}) = IA\_Strength(N_{u3}, N_{j3}) = (1-\frac{4}{2\cdot 7})\cdot 0.6 = 0.43$, and $IA\_Strength(N_{u3}, N_{k1}) = IA\_Strength(N_{u3}, N_{k3}) = (1-\frac{1}{2\cdot 7})\cdot 0.7 = 0.65$.

5 Ambiguous Contextual Keyword Search

A context-based re-finding request $Q = \{k_1, k_2, \ldots, k_n\}$ is a set of contextual keywords, and the query target $\mathcal{C}_T = \{ct_1, ct_2, \ldots\}$ is a set of probabilistic context trees. We first construct index for probabilistic context trees utilizing Dewey encoding [10,27]. In our context trees, the Dewey number of the root is the tree id. For each keyword node in a context tree, we build an index according to its keywords, and the mapping between the node and its probability in the context tree is also kept. Besides, we build an index for each context association. Note that NAs and SAs are bidirectional, while IAs are directional.

5.1 Probabilistic Context Tree Matching

Given Q and $\mathcal{C}_T$, the keyword nodes that contain at least one keyword of Q are called matched keyword nodes, which can be identified through scanning the keyword inverted node lists. Since the tree id can be easily got from the Dewey code of a keyword node, based on the matched keyword nodes, we can easily obtain the exactly matched context trees that encompass every keyword of Q.

For a matched context tree ct, we need to compute its probability of matching Q, denoted as $Prob(ct, Q)$. Assume there are m matched keyword nodes v_1, ..., v_m in ct, each of which contains a set of contextual keywords. The probability of v_i in the context tree is denoted as $Pr(v_i)$, $1 \leq i \leq m$. As there are n keywords in Q, we use a set of real numbers $U(v_i) = \{\mu_{2^n-1}, \mu_{2^n-2}, \ldots, \mu_0\}$ (the maximum size is 2^n) to represent the probabilities of v_i matching Q, *i.e.*, μ_{2^n-1} means the probability that v_i contains all the keywords in Q, and μ_{2^n-i} (where $i = 1, 2, \ldots, n$) means the probability that v_i only contains k_i, *etc.* Hence, we get m sets of probabilities, which should be merged together in the way that the subscript numbers use bitwise OR operation, while the probabilities use multiplication and add operations. An example of computing the probability of a context tree matching Q is shown in Fig. 6, where $Q = \{k_1, k_2, k_3\}$, the matched keyword nodes are v_1, v_2, v_3 and v_4, and their probabilities in the context tree are $Pr(v_1) = 0.5$, $Pr(v_2) = 0.6$, $Pr(v_3) = 0.7$ and $Pr(v_4) = 0.8$. The four sets of probabilities $U(v_1)$, $U(v_2)$, $U(v_3)$ and $U(v_4)$ are merged together ($\biguplus$ denotes the merge operation), and the final μ_{2^n-1} (0.68) is the probability of the context tree matching Q.

5.2 Approximate Search

If the query request Q contains ambiguous information, exact matching will not work well. To eliminate the possible incorrectness of a query request Q with n

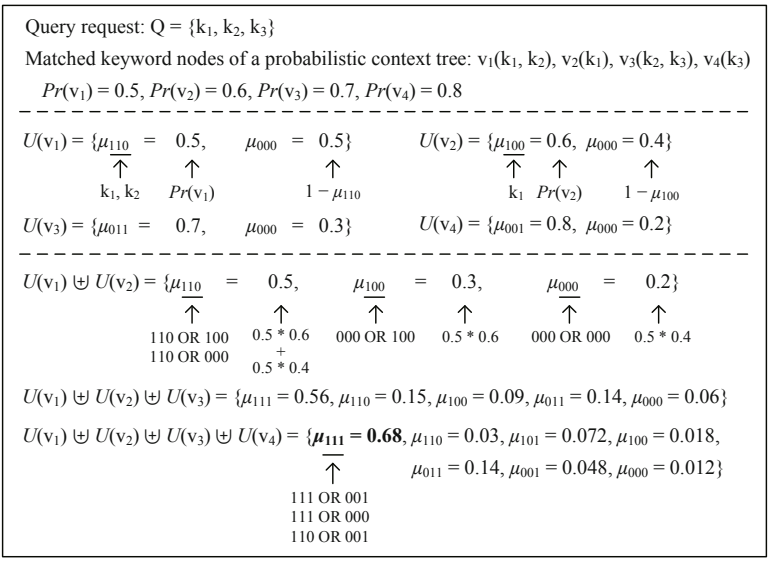


Fig. 6. Example of computing the probability of a context tree matching Q

(≥ 2) keywords, we can adopt partial matching approach [19], *i.e.*, remove each keyword in turn, using the remaining $n-1$ keywords in creating multiple re-finding queries, and retrieve the results of each modified query via exact matching method. Since we do not know which keyword in Q is incorrect, partial matching is a good choice to some degree. However, it will inevitably bring many irrelevant results and meanwhile take more time to execute the queries. Moreover, it is based on the assumption that only one keyword of the query request Q is wrong. Thus, we address approximate matching in the following.

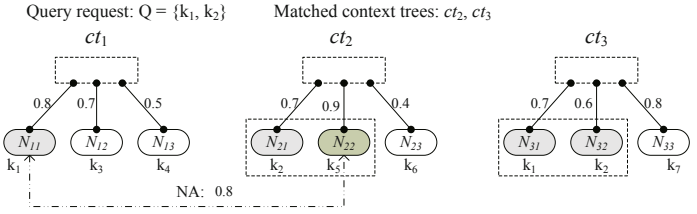


Fig. 7. Example of approximate matching

Generally, users' referring to unreliable contextual retrieval cues is caused by memory interference. For example, to re-find a target item, assume $Q = \{k_1, k_2\}$ is one of the correct query requests. However, the user imposes a wrong query $Q' = \{k_1, k_2'\}$. We consider that there should be some associations between k_2

Algorithm 1. Approximate Contextual Search Algorithm

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

```
1: load node list L = {L_i} based on Q, 1 ≤ i ≤ n;
2: for i = 1; i ≤ n; ++i do
3:    L_tmp ← L_i;
4:    for each v ∈ L_tmp do
5:       load v's associated nodes A;
6:       for each a ∈ A do
7:          if a ∉ L_i then
8:             update Pr(a) by multiplying the association strength;
9:             add a to L_i;
10: determine the matched context trees T = {ct_1, ct_2, ...} based on L;
11: for each ct ∈ T do
12:    let M be the set of matched and associated keyword nodes of ct;
13:    for each v_i ∈ M do
14:       compute the probabilities of v_i matching Q, U(v_i) = {μ_{2^n-1}, μ_{2^n-2}, ..., μ_0};
15:    merge all the U(v_i) together, U ← ⊎_{i=1}^{|M|} U(v_i);
16:    Prob(ct, Q) ← U.μ_{2^n-1};
17:    if Prob(ct, Q) > 0 then
18:       insert ct into R_T according to Prob(ct, Q);
19: return R_T;
```

and k'_2. We do not modify the query request Q. We consider the context trees that contain or associate every keyword of Q as matched context trees, where a contextual keyword is deemed to be associated by a context tree if there exists at least an association between one node of the tree and one node of another tree which exactly contains that keyword. Firstly, we will use each keyword of Q to get the matched keyword nodes, based on which the associated keyword nodes can be obtained passingly. Then the matched context trees (exact or approximate matching) can be determined by all the retrieved keyword nodes. Fig. 7 demonstrates an example of approximate matching against a query request $Q = \{k_1, k_2\}$. Four matched keyword nodes N_{11}, N_{21}, N_{31} and N_{32}, and one associated keyword node N_{22} (through the NA with N_{11}), are obtained. Based on these keyword nodes, two matched context trees ct_2 and ct_3 are got. Particularly, when computing the probability of a context tree ct matching Q, the association strengths are multiplied to the involving keyword nodes' probabilities. For example, we multiply $NA\_Strength(N_{11}, N_{12})$ with $Pr(N_{22})$ before computing $Prob(ct_2, Q)$. In the end, $Prob(ct_2, Q) = 0.504$, while $Prob(ct_3, Q) = 0.42$. Thus, ct_2 should be ranked before ct_3 in the result list.

The detailed procedure of approximate matching is illustrated in Algorithm 1. It scans keyword inverted node lists once (Line 1), and add them with the associated nodes (Line 2 - 9). Based on the matched and associated keyword nodes, the approximately matched context trees can be determined (Line 10).

To compute the probability of a matched context tree, it first compute the probabilities of each involving node v_i matching Q, and then merge them together (Line 12 - 15). The matched context tree ct will be inserted into the result list at the right position if its probability *w.r.t.* Q is larger than zero (Line 16 - 18).

Complexity Analysis: Adding associated keyword nodes (Line 2 - 9) takes $O(\sum_{i=1}^{n} |L_i| \cdot |A|)$. Identifying the matched context trees T (Line 10) takes $O(n \cdot |T|)$. Computing and merging the probabilities of a matched context tree's involving keyword nodes matching Q (Line 12 - 15) takes $O(\prod_{i=1}^{|M|} |U(v_i)|)$. Thus, the total time cost is $O(\sum_{i=1}^{n} |L_i| \cdot |A|) + O(n \cdot |T|) + O(|T| \cdot \prod_{i=1}^{|M|} |U(v_i)|) = O(\sum_{i=1}^{n} |L_i| \cdot |A|) + O(|T| \cdot \prod_{i=1}^{|M|} |U(v_i)|)$, depending on the number of matched and associated keyword nodes, and the number of matched context trees.

6 Experiments

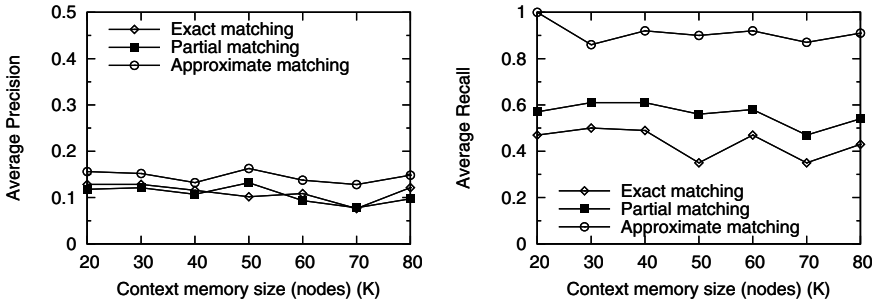
In this section, we conduct a set of experiments to demonstrate the effectiveness of our solution for approximate contextual search in both synthetic and real-world datasets, concentrating on two measurements: 1) query quality (precision and recall) and 2) query response time. The experiments are conducted on a PC with 2.2 GHz Intel Core 2 Duo CPU, and 2 GB memory on Windows 7 OS.

6.1 Experiments on Synthetic Data

Setting. We generate 3-month data to simulate a user's computer activities. Each data item contains 6 attributes: start time, end time, focus time length, keywords, data type (web page or not) and activity type, which are set randomly, where its time span (end time - start time) is a random value from 30 seconds to 15 min, and its focus time length is from 5 seconds to half of its time span. There are 7 activity types, and we separately generate a set of phrases for each of which, where each phrase contains 3 to 7 words. In generating data items, the keywords are randomly selected from the corresponding set of phrases based on the chosen activity type. In the end, 27,667 data items are generated, based on which we build a probabilistic context tree for each web page type data item whose focus time length is not less than 60 seconds. We get 9,954 context trees with 86,648 keyword nodes in total.

For generating query requests, we first randomly select a part of the web page type data whose focus time length is not less than 60 seconds as the true to-be-revisited targets. For each of them, we then choose 2 to 5 keywords as a query request, complying with one of the following requirements: 1) the keywords are around the target; and 2) the keywords are within the past week taking the target as reference. For the former, we are prone to choose the keywords from the data items whose focus time length is longer than others. For the latter, the similarity and inference associations are taken into account, *e.g.*, the keywords which are similar to the former's candidate keywords will be considered. In total, we separately generate 100 queries for the 2- to 6-keyword query requests.

Results. The first experiment focuses on query precision and recall rates under different sizes of context memory (measured by the total number of keyword nodes in context trees). The performances of contextual keyword search with exact matching, partial matching and approximate matching are studied, and the comparison results are shown in Fig. 8. Because of the low frequencies of contextual keywords in the generated synthetic dataset, the number of returned results for a query request will not change observably along with the varying of context memory size, *i.e.*, the context memory size does not impact the average precision and recall rates very much. Since the query requests are with some flaws, the approximate matching outperforms the exact and partial matching in both average precision and recall rates.

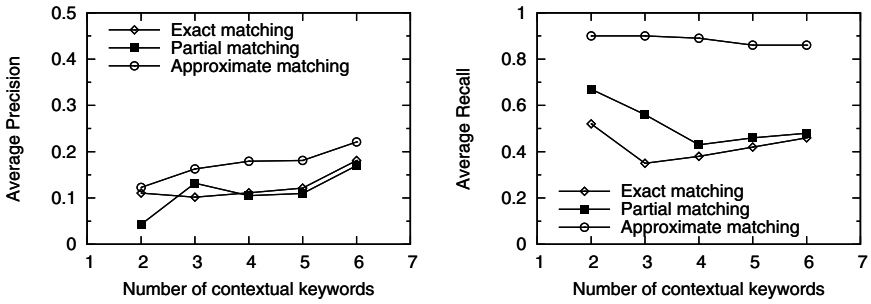


(a) Average precision (b) Average recall

Fig. 8. Varying context memory size (nodes) on synthetic data, $|Q| = 3$

The second experiment also concerns on query precision and recall rates through varying the number of contextual keywords in a query request. Fig. 9 demonstrates the comparison results between approximate matching and exact/partial matching. Clearly, the approximate matching can achieve much better results than the other two matching methods. As with more contextual keywords, more query conditions are imposed, the precision rate get improved along with the increase of the number of contextual keywords from 2 to 6. Whereas, more contextual keywords will lead to a higher potential of carrying mistaken information, making the recall rates of exact and partial matching decrease. While the approximate matching approach is not impacted, the recall rate of which keeps at high level, over 85%.

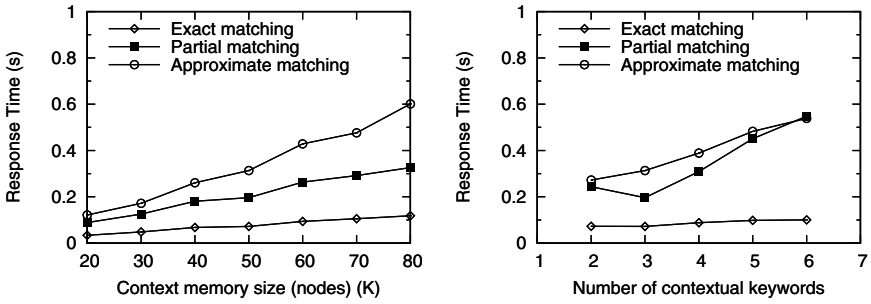
The query response time under different sizes of context memory and different number of contextual keywords in a query request is also studied, as illustrated in Fig. 10. The three matching methods take more time to do re-finding along with the growth of context memory size. However, increasing the number of contextual keywords hardly affects the response time of exact matching, while the response time of the other two methods grows along with the keyword number. This is contributed to the factor that the exact matching method always chooses the



(a) Average precision (b) Average recall

Fig. 9. Varying $|Q|$ on synthetic data, the number of nodes = 50K

shortest node list ($|Q|$ lists in total) to start determining the matched context trees and the increase of $|Q|$ just incurs a bit more time, while the number of matched context trees decreases, which requires less time to compute their probabilities. Of the three methods, exact matching takes the least response time. This is due to the reason that both approximate and partial matching have to deal with much more matched (associated) keyword nodes for a query request than exact matching.

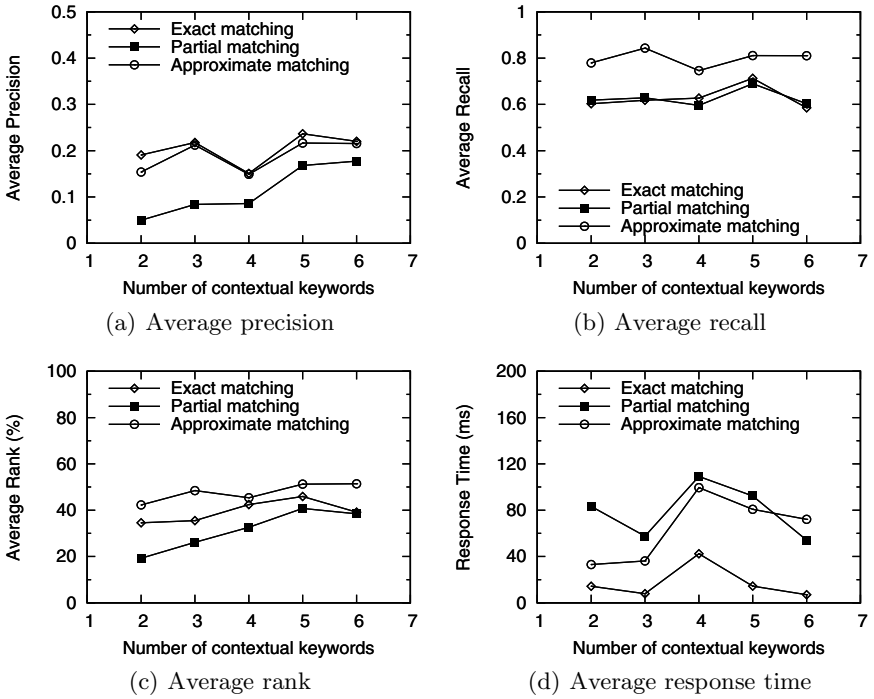


(a) $|Q| = 3$ (b) number of nodes = 50K

Fig. 10. Query response time for varying context memory size (nodes) and $|Q|$ on synthetic data

6.2 Experiments on Real Data

Setting. To examine the performance of our solution on real data, we collected computer activity data of 8 participants (graduate students) over 6 weeks. The participants' running computer applications were continuously monitored, and related information of focused windows like window title, application type, the start and end time of being focused, *etc.*, were kept automatically. The browsed



(a) Average precision (b) Average recall

(c) Average rank (d) Average response time

Fig. 11. Performance on real data for varying $|Q|$

web pages were considered as to-be-revisited targets, and we build a probabilistic context tree for each of them. The context tree's keyword nodes are inferred from participant's computer applications running before and after the web page access. In total, we collected 1,530 context trees (linking to browsed web pages) with 28,256 keyword nodes. For each participant's data, we separately select 20 queries of 2- to 6-contextual keyword as query requests.

Results. We study query quality (including precision, recall and rank) and query response time of our solution on the 8 collected real datasets separately. The average results are demonstrated in Fig. 11. For precision rate, both exact and approximate matching methods act much better than partial matching method. It is because that the latter retrieves more irrelevant results. For recall rate, approximate matching outperforms the other two methods clearly, since approximate matching is able to be immune to the impacts of query requests containing mistaken information in most cases. To measure results' ranking, we define $rank = true\_result\_ranking\_position/number\_of\_returned\_results \cdot 100\%$. Considering results' ranking positions are determined by their probabilities of matching Q, and approximate matched results involving association strengths could not get good rankings in many cases, both exact and partial matching perform better than approximate matching for average rank. For response time, exact matching still outperforms the other two methods.

7 Conclusion

In this work, we propose an approximate matching approach for contextual keyword-based information re-finding. It is aiming at eliminating the negative influences of users' misremembering on contextual retrieval cues which are organized in probabilistic context trees. Based upon the observation of human memory interference and the characteristics of users' misremembering in memory recall, we construct three types of context associations between probabilistic context trees: neighborhood associations, similarity associations and inference associations. The approximate matching algorithm takes advantages of such context associations to deal with users' unreliable re-finding requests, and the matched or associated results will be returned. The proposed solution is evaluated on both synthetic and real-world datasets. Under the situation of some contextual keywords of users' query requests containing mistaken information, our experimental results show that approximate matching performs much better than exact and partial matching in re-finding users' desired results.

Acknowledgement. The work is supported by National Natural Science Foundation of China (60773156, 61073004), Chinese Major State Basic Research Development 973 Program (2011C B302203-2), Tsinghua University Initiative Scientific Research Program, and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

References

1. Capra, R., Perez-Quinones, M.A.: Using web search engines to find and refind information. IEEE Computer 38(10), 36–42 (2005)
2. Chen, J., Guo, H., Wu, W., Wang, W.: iMecho: an associative memory based desktop search system. In: CIKM, pp. 731–740 (2009)
3. Chen, Y., Jones, G.: Integrating memory context into personal information refinding. In: The 2nd Symposium on Future Directions in Info. Access (2008)
4. Deng, T., Zhao, L., Feng, L.: Enhancing web revisitation by contextual keywords. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 323–337. Springer, Heidelberg (2013)
5. Deng, T., Zhao, L., Feng, L., Xue, W.: Information re-finding by context: a brain memory inspired approach. In: CIKM, pp. 1553–1558 (2011)
6. Deng, T., Zhao, L., Wang, H., Liu, Q., Feng, L.: ReFinder: a context-based information re-finding system. IEEE TKDE (PrePrint, August 14, 2012)
7. Dorneles, C.F., Goncalves, R., dos Santos Mello, R.: Approximate data instance matching: a survey. Knowledge and Information Systems 27(1), 1–21 (2011)
8. Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., Robbins, D.C.: Stuff i've seen: a system for personal information retrieval and re-use. In: SIGIR, pp. 72–79 (2003)
9. Fuller, M., Kelly, L., Jones, G.: Applying contextual memory cues for retrieval from personal information archives. In: PIM, Workshop at CHI (2008)
10. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: ranked keyword search over xml documents. In: SIGMOD, pp. 16–27 (2003)

11. Hailpern, J., Jitkoff, N., Warr, A., Karahalios, K., Sesek, R., Shkrob, N.: Youpivot: improving recall with contextual search. In: CHI, pp. 1521–1530 (2011)
12. Ji, S., Li, G., Li, C., Feng, J.: Efficient interactive fuzzy keyword search. In: WWW, pp. 371–380 (2009)
13. Jonides, J., Nee, D.E.: Brain mechanisms of proactive interference in working memory. Neuroscience 139(1), 181–193 (2006)
14. Kailing, K., Kriegel, H.-P., Schönauer, S., Seidl, T.: Efficient similarity search for hierarchical data in large databases. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 676–693. Springer, Heidelberg (2004)
15. Lustig, C., Hasher, L.: Implicit memory is not immune to interference. Psychological Bulletin 127(5), 629–650 (2001)
16. 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)
17. Motro, A.: Vague: a user interface to relational databases that permits vague queries. ACM TOIS 6(3), 187–214 (1988)
18. Navarro, G.: A guided tour of approximate string matching. ACM Comput. Surv. 33(9), 31–88 (2001)
19. Qumsiyeh, R., Pera, M.S., Ng, Y.: Generating exact and ranked partially matched answers to questions in advertisements. In: VLDB, pp. 217–228 (2012)
20. Robinson, J.A., Swanson, K.L.: Autobiographical memory: The next phase. Applied Cognitive Psychology 4(4), 321–335 (1990)
21. Soules, C.A.N., Ganger, G.R.: Connections: using context to enhance file search. In: SOSP, pp. 119–132 (2005)
22. Tauscher, L., Greenberg, S.: How people revisit web pages: empirical findings and implications for the design of history systems. Int'l J. of Human Computer Studies 47, 97–137 (1997)
23. Teevan, J., Adar, E., Jones, R., Potts, M.A.S.: Information re-retrieval: repeat queries in yahoo's logs. In: SIGIR, pp. 151–158 (2007)
24. Tomlinson, T.D., Huber, D.E., Rieth, C.A., Davelaar, E.J.: An interference account of cue-independent forgetting in the no-think paradigm. Proceedings of the National Academy of Sciences 106(37), 15588–15593 (2009)
25. Tulving, E.: What is episodic memory? Current Directions in Psychological Science 2(3), 67–70 (1993)
26. Wohldmann, E., Healy, A., Bourne, L.: A mental practice superiority effect: Less retroactive interference and more transfer than physical practice. Journal of Experimental Psychology: Learning, Memory, and Cognition 34(4), 823–833 (2008)
27. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest lcas in xml databases. In: SIGMOD, pp. 527–538 (2005)