# Searching Desktop Files Based on Access Logs

Yukun Li, Xiyan Zhao, Yingyuan Xiao, and Xiaoye Wang

Key Laboratory of Intelligence Computing and Novel Software Technology, Tianjin Key Laboratory of Computer Vision and System, Ministry of Education
Tianjin University of Technology, 300384, Tianjin, China
{liyukun,yyxiao,wangxy}@tjut.edu.cn, zhaoxiyan322@sina.com

**Abstract.** People often meet trouble in searching a desktop file when they can not remember exact words of its filename. In this paper, we firstly propose an algorithm to generate access logs by monitoring desktop operations and implement a prototype. By running it in several computers of selected participants we collected a data set of access logs. Then we propose a graph model to represent personal desktop files and their relationships, and highlight two file relationships(content relationship and time relationship) to help users search desktop files. Based on the graph model, we propose a desktop search method, and the experimental results show the feasibility and effectiveness of our methods.

## 1 Introduction

When people want to re-find a desktop file and can not remember its location, they often choose desktop search tools to do it. Because most desktop search tools are based on keyword search technology, people often meet trouble in searching a desktop file when forgetting exact words of the filename. Because of the limitation of human memory, it is unreasonable to ask each person to exactly remember words of every filename of desktop. For example, if a user wants to search the file "An draft on dataspace framework.pdf" with existing desktop search tools, he/she has to remember one or some words of set {"draft", "dataspace", "framework"}. Because most existing desktop search tools do not distinct accessed files from a great number of system files, they often work at low performance. This paper focuses on helping people efficiently search desktop file when they lose memory about exact file information.

### 1.1 Related Work

Chirita and Nejdl [1] proposed to connect semantically related desktop items by exploiting analysis information about sequences of accesses. Peery et al. [2] presented a multi-dimension query method in personal dataspace, which individually grades each dimension(content, structure and metadata), then combines the three dimension scores into a meaningful unified score. All the works above do not refer to how to get the access logs and how to search desktop files based

W. Meng et al. (Eds.): DASFAA 2013, Part II, LNCS 7826, pp. 267–274, 2013.

on them. In [3], an idea about identifying personal tasks based on user's operations was proposed and demonstrated. In [4], a method on querying personal file based on user's working context was proposed. Some researchers of database area studied about managing personal data set, and the work involves personal dataspace model [5,6], pay-as-you-go integration [7,8], index [9] and query. Some interesting prototypes were developed like iMemes [10], Semex [7], MyLifeBit [11], HyStack [12] and so on. The works listed above didn't efficiently solve the problem on how to search desktop file based on access logs.

### 1.2 Contribution Summary

The contributions of this paper can be summarized as below:(1) Propose an algorithm to generate access logs by monitoring user's operations on desktop and implement a prototype system, and by running it on several computers of selected participants we collect a data set of access logs from eight persons.(2)Propose a graph model to represent personal desktop files and their relationships, and highlight two file relationships(content relationship and time relationship) to help users re-find personal desktop files, furthermore propose a desktop search method based on the graph model.

The rest is organized as follows: In section 2, we describe our desktop search method. Section 3 is about experiments. Section 4 concludes this paper.

## 2 Searching Methods Based on Access Logs

As most accesses to desktop files are re-finding [13], we propose that (1)it should be enough for desktop search to scan only the accessed files, (2)the accessed files can be identified by monitoring desktop operations and (3)the access logs can provide additional methods about desktop search.

### 2.1 Generating Access Logs

We propose to generate user access logs by monitoring the recently-accessed folder of operating system like Windows XP, and take a 3-ary tuple {*OperationTime, OperatedFileName, OperatedDirectory*}to represent the schema. The steps include: (1) If a change of the latest accessed desktop file is detected, a new access record will be generated, and the attributes OperationTime, OperatedFileName and OperatedDirectory can be identified through APIs provided by operating system; (2) If the latest accessed file is not involved in the log table, it means the user is accessing a new file. By this method we developed a prototype and collected logs of eight persons about one year.

Table 1 shows a part of an author's access logs, which includes 8 records and refers to 5 different desktop files. Except "A proposal for applying an award.doc", all the files are related to the activity "submitting to DASFAA 2013", although some filenames are not similar, like "figure1.vsd" and "submission to DASFAA

2012". When the user wants to re-find "figure1.vsd" and only remembers it relates to the activity "submitting to DASFAA 2013", instead of remembering the filename "figure1.vsd", it will be difficult for user to do by existing desktop search tools. But if the time relation between the two files is highlighted, which will provide the user additional ways for searching "figure1.vsd". Therefore besides content similarity, we propose to highlight time relationship to help users search desktop files more efficiently.

**Table 1.** Overview of a part of a user's access logs

| No | User | File name | Access time |
|---|---|---|---|
| 1 | U1 | Submission to DASFAA 2012.tex | 2012-10-01 14:00 |
| 2 | U1 | figure1.vsd | 2012-10-01 14:02 |
| 3 | U1 | Experimental data for DASFAA submission.xls | 2012-10-01 14:05 |
| 4 | U1 | Comments from a Coauthor.doc | 2012-10-01 14:20 |
| 5 | U1 | A proposal for applying an award.doc | 2012-10-01 14:30 |
| 6 | U1 | Submission to DASFAA 2012.tex | 2012-10-01 14:35 |
| 7 | U1 | figure1.vsd | 2012-10-01 15:30 |
| 8 | U1 | Experimental data for DASFAA submission.xls | 2012-10-01 15:40 |

### 2.2 Desktop File Graph Model and Construction

We propose a graph model to describe desktop files and their relationships, and name it *DFG(Desktop File Graph)*. A *DFG* is described as *G(F,R,n)*, where $F$ is a set of desktop files accessed by user, $n$ is the number of files in $F$, and $R$ is a set of file relationships. Based on the observations mentioned in section 2.1, we take the following two relationships into consideration: *content similarity(Co)* and *time relationship(Ti)*, where *Co* means the similarity of two files in content, and *Ti* means the possibility that two files are accessed together. The *DFG* model provides an additional method for users to search desktop files. How to identify the relationships is the key problem. In this section, we propose methods to identify the two relationships.

As to content relationship, we propose to take filename similarity to approximately represent the content relationship of two files. For each file, we take a set of tokens included in the filename to denote its content. By computing the similarity of token sets of two files, we can work out the content similarity of them. In our work we take the formula 1 to compute Jaccard similarity [14] of the two token sets of the files $F_i$ and $F_j$, and regard it as the content relationship of the two files.

$$Co(F_i, F_j) = \frac{|F_i.S_{token} \cap F_j.S_{token}|}{|F_i.S_{token} \cup F_j.S_{token}|} \tag{1}$$

In formula 1, $F_i.S_{token}$ means the token set of file $F_i$, and $F_j.S_{token}$ means the token set of file $F_j$. If $Sim(F_i, F_j)$ is bigger than 0, we add an edge between $F_i$

and $F_j$ in the graph to denote the content relationship, and the weight of the edge ($F_i$,$F_j$) equals to the value of $Co(F_i, F_j)$.

As to the time relationship, Our algorithm is based on the position below: If two files are often accessed at the same time, we think they have time relation. The hard problem is how to decide "at the same time". In this work we propose to take "accessed sequentially" to approximately evaluate "accessed at the same time". For example, let A and B be two files, the more times they are accessed sequentially, the closer time relationship they have.

How to compute the time relationship is a challenging problem. Firstly, to two given files, the times they are accessed sequentially is dynamic; Secondly, it needs a method to increasingly update the time relation value based on its existing value. We propose a simple method to compute it as formula 2.

$$Ti_n(F_i, F_j) = \frac{Ti_{n-1}(F_i, F_j) + 1}{2} \tag{2}$$

In formula 2, $Ti_{n-1}(F_i, F_j)$ means the existing time relation value between the two files denoted by $F_i$ and $F_j$, and the initial value $Ti_0(F_i, F_j)$ is 0. When a new sequential access to $F_i$ and $F_j$ is found during monitoring user accesses, their time relation will be updated based on formula 2. The new value will be bigger than the old one, and its maximum value will not exceed 1. For example, when their first sequential access is found, $Ti_1(F_i, F_j)$ = (0+1)/2 =0.5, and when the second sequential access is found, $Ti_2(F_i, F_j)$ = (0.5+1)/2 =0.75.

Algorithm 1 shows the process of constructing desktop file graph. It supposes there exists a desktop file graph $G_s$, and shows how the file set and the two file relationship sets will be updated when a new access to a desktop file is found.

### 2.3 Searching Method

we propose a simple interface to perform the graph-based search, whose format is "$keyword_1, keyword_2, ..., keyword_n \backslash [C|T]$", where $keyword_i$ is a keyword user input, C and T are options which are set by users when they plan to search desktop files, where C means searching based on content relationship and T means searching based on time relationship. For example, "database, index \ C" means searching the files including keywords "database" and "index" based on content relationship, "database, index \ T" means searching the files whose filename includes keywords "database" and "index" based on time relationship. Based on the input keywords, we take Jaccard [14] method to compute the similarity between the input keywords($In.S_{keywords}$) and each file's token set($F_i.S_{token}, 1 \leq i \leq n$) by formula 3, and get a n-ary vector $V_s$ as the primary results, which is taken to generate final results based on the desktop file graph.

$$V_s(i) = \frac{|F_i.S_{token} \cap In.S_{keywords}|}{|F_i.S_{token} \cup In.S_{keywords}|}. \tag{3}$$

Assume the desktop file space is a graph $G(F, Co, Ti, n)$, where $F$ is the set of desktop files accessed by user, $n$ is the number of files in $F$, $Co$ is the edge set

**Algorithm 1.** Constructing desktop file graph

**Input:** A new accessed file $f$ and a graph $Gs(F, Co, Ti, m, n)$, where $F$ is a set of files, Co is the content relation set, Ti is the time relation set, $n$ is the total number of files, and $m$ is the ID number of the file accessed last time.
**Output:** An updated graph $Gs(F, Co, Ti, m, n)$.

**procedure** *Constructing Desktop File Graph*$(f, Gs(F, Co, Ti, m, n))$
  **if** $f \in F$ **then**
    find the ID number of $f$ in $Gs.F$ and store it into k
  **else**
    add a new file $Gs.F_{n+1}$
    n = n + 1, k = n
    **for** (int i = 1, i $\leq n$, i++) **do**
      $S_{co} = |Gs.F_i.Tokens \bigcap Gs.F_k.Tokens|$ / $|Gs.F_i.Tokens \bigcup Gs.F_k.Tokens|$
      **if** $S_{co} > 0$ **then**
        $\mathrm{Co}(Gs.F_i, Gs.F_k) = S_{co}$
      **end if**
    **end for**
  **end if**
  $Ti(Gs.F_m, Gs.F_k) = (Ti(Gs.F_m, Gs.F_k) + 1)/2$
**end procedure**

of content relationship, $Ti$ is the edge set of time relationship. In our method, we imagine $G(F, Co, Ti, n)$ as two virtual graphs $Gc(F, Co, n)$ and $Gt(F, Ti, n)$, and take two $n \times n$ adjacency matrixes to present them, where the nondiagonal element $a_{ij}$ is the weight of the edge from vertex $i$ to vertex $j$, and the diagonal element $a_{ii}$ is set 1 here. Let M be the adjacency matrixes of selected graph view ($Gc(F, Co, n)$ *or* $Gt(F, Ti, n)$), based on $V_s$ we can compute the result file set by the formula $V_r = V_s \times M$, and the result $V_r$ is a n-ary vector. Based on $V_r$, we can compute the final result $Rs$ by the formula $Rs = \{F_i | V_r(i) \neq 0, 0 \leq i \leq n\}$. Naturally, based on the values of $V_r$, the searching results can be ranked easily.

## 3 Experiments

Table 2 shows the participants' attributes(age, sex and position) and data sets. The parameters of data set include time length of data collection(Time), access times, accessed files, re-access times, and the ratio of re-access times to access times(Re-accessRatio). From the table we can discover most operations of desktop are re-accesses.

### 3.1 Experimental Design

We create a benchmark with the help of the participants. To the best of our knowledge there is no existing benchmark on evaluating desktop re-finding methods. Based on the number of the files a user wants to search, we classify the searching cases into two categories: single file search and multiple file search.

**Table 2.** Overview the statistics on access log collection

| *User* | Age | Sex | Position | Time (day) | Access Times | Accessed Files | Re-access Times | Re-accessRatio (%) |
|---|---|---|---|---|---|---|---|---|
| U1 | 26 | Female | Master | 351 | 9514 | 1836 | 7678 | 80.70 |
| U2 | 25 | Male | Master | 351 | 5994 | 2291 | 3703 | 61.78 |
| U3 | 27 | Female | Master | 223 | 1005 | 393 | 612 | 60.90 |
| U4 | 36 | Male | PhD | 355 | 7320 | 1894 | 5426 | 74.13 |
| U5 | 25 | Male | Master | 354 | 15040 | 3829 | 11211 | 74.54 |
| U6 | 29 | Male | PhD | 183 | 3021 | 813 | 2208 | 73.09 |
| U7 | 22 | Female | Undergraduate | 213 | 6064 | 1522 | 4542 | 74.90 |
| U8 | 23 | Female | Undergraduate | 233 | 6587 | 1755 | 4832 | 73.36 |

Single file search means relocating a specific file, and multiple file search means searching multiple files. We ask each participant to design some searching cases according to their searching experience, and give the correct answer for each search based on what they want to find. We let each user $U_i$ design 10 search samples respectively for single file search and multiple file search, and ask them to give a file or a file set to every search sample as right answer.

We take the popular measures recall,precision and F-score [15] to evaluate our methods. Because we have not found existing work about helping users search desktop files based on monitoring user access logs, and desktop search tools are popular ways for users to re-find desktop files, we select two popular desktop tools MS desktop search and Google desktop search engine as baseline to evaluate our method. To each search, we perform it with different methods and take top-k files returned as the final results, and set $k = 30$ in our experiments. By comparing the final results with the benchmark for each search sample, we can compute the recall, precision and F-score of each search, then we can work out the average value of recall, precision and F-score of each method.

### 3.2 Experimental Results

Figure 1 illustrates the advantages of our method: (1)Either to single file search or to multiple file search, our access log-based method's F-score is the best; (2) The recall of our method equals to 1 approximately, which is much better than other tools; (3) Precisions of all methods are not high, which is in accord with our expects because there exist some unrelated files whose names share some same words. Totally our method has better precision than other desktop search tools.

Like desktop search engine, our method also has two types of cost: off-line cost and online cost. (1)As to online cost, MS desktop search tool shows the lowest performance, it always takes several minutes to handle a search and returns a great number of files which often include many system files. Our log-based method and Google Desktop search engine show a better online performance, especially the log-based method's average response time is less than one second,

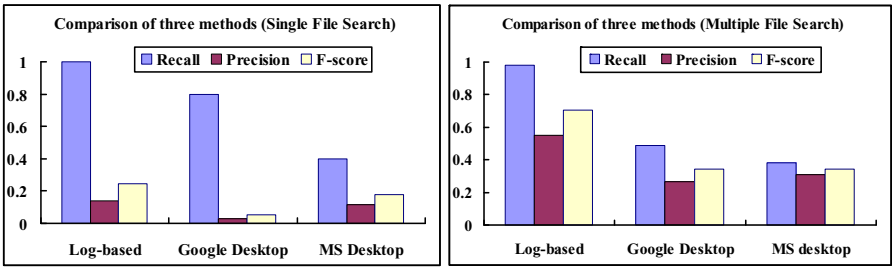


**Fig. 1.** Comparison of log-based method, Google desktop and MS desktop

which can satisfy most users' needs. (2)As to off-line cost, the cost of our method is much lower than the selected desktop search tools. Take google desktop search for example, it always takes several hours to build the initial index in some cases, and the update of index is also delayed much more, which sometimes results in search failure. To MS desktop search, it has little additional cost for updating. Totally, our access log-based method's performance is comprehensively better than other desktop search tools, and can satisfy users' requirements.

We also have the following observations in experiments. (1)Sometimes users do not name a desktop file according to its content for some reasons like "download it from a web site and keep its original filename", "get it from other persons", and so on; (2)People archive personal desktop files with folders according to different rules. For example, some folders are created based on user activities, like "Submission to DASFAA 2013", which includes the files related to the submission to DASFAA 2013, and sometimes based on the file categories, like "dataspace paper", which includes the papers related to dataspace topic.(3)Access logs provide users additional facets to search desktop files like access frequency, access time, operation types and so on. The observations discover some interesting research topics and we will study them in the future.

## 4 Conclusions

In this paper, we firstly propose a method to generate access logs by monitoring users' operations on desktop and build a data set of access logs of eight persons. Then we propose a desktop search method based on access logs. The experimental results show the effectiveness of our method.

**Acknowledgments.** This research was supported by the Natural Science Foundation of China under grant number 61170027, 61170174; Natural Science Foundation of Tianjin under grant number 11JCYBJC26700.

## References

1. Chirita, P.-A., Nejdl, W.: Analyzing User Behavior to Rank Desktop Items. In: Crestani, F., Ferragina, P., Sanderson, M. (eds.) SPIRE 2006. LNCS, vol. 4209, pp. 86–97. Springer, Heidelberg (2006)
2. Peery, C., Wang, W., Marian, A., Nguyen, T.D.: Multi-Dimensional Search for Personal Information Management Systems. In: 11th International Conference on Extending Database Technology, pp. 464–475. ACM Press, Nantes (2008)
3. Li, Y., Zhang, X., Meng, X.: Exploring Desktop Resources Based on User Activity Analysis. In: 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, p. 700. ACM Press, Geneva (2010)
4. Li, Y., Meng, X.: Supporting Context-based Query in Personal DataSpace. In: 18th ACM Conference on Information and Knowledge Management, pp. 2–6. ACM Press, Hong Kong (2009)
5. Franklin, M.J., Halevy, A.Y., Maier, D.: From databases to dataspaces: A new abstraction for information management. SIGMOD Record (SIGMOD) 34(4), 27–33 (2005)
6. Dittrich, J.P., Antonio, M., Salles, V.: iDM:A unified and versatile data model for personal dataspace management. In: 32nd International Conference on Very Large Data Bases, pp. 367–378. ACM Press, Seoul (2006)
7. Dong, X., Halevy, A.: A platform for personal information management and integration. In: 2nd Biennial Conference on Innovative Data Systems Research, pp. 119–130. Online Proceedings, Asilomar (2005)
8. Dong, X., Halevy, A., Yu, C.: Data integration with uncertainty. In: The 33rd International Conference on Very Large Data Bases, pp. 687–698. ACM Press, Vienna (2007)
9. Dong, X., Halevy, A.: Indexing dataspaces. In: The ACM SIGMOD International Conference on Management of Data, pp. 43–54. ACM Press, Beijing (2007)
10. Blunschi, L., Dittrich, J.P., Girard, O.R., Karakashian, S.K., Salles, M.A.V.: A Dataspace Odyssey: The iMeMex Personal Dataspace Management System. In: 3rd Biennial Conference on Innovative Data Systems Research, pp. 114–119. Online Proceedings, Asilomar (2007)
11. Gemmell, J., Bell, G., Lueder, R., Drucker, S.M., Wong, C.: MyLifeBits: fulfilling the Memex vision. In: 10th ACM International Conference on Multimedia, pp. 235–238. ACM Press, Juan les Pins (2002)
12. Karger, D.R., Bakshi, K., Huynh, D., Quan, D., Sinha, V.: Haystack: A customizable general-purpose information management tool for end users of semistructured data. In: 2nd Biennial Conference on Innovative Data Systems Research, pp. 13–26. Online Proceedings, Asilomar (2005)
13. Elsweiler, D., Baillte, M., Ruthven, I.: Exploring Memory in Email Refinding. ACM Transactions on Information Systems(TOIS) 26(4), Article No.21 (2008)
14. Bayardo, R.J., Ma, Y., Srikant, R.: Scaling up all pairs similarity search. In: 16th International Conference on World Wide Web, pp. 131–140. ACM Press, Banff (2007)
15. Raghavan, V.V., Bollmann, P., Jung, G.S.: Retrieval System Evaluation Using Recall and Precision: Problems and Answers. In: 12th International Conference on Research and Development in Information Retrieval, pp. 59–68. ACM Press, Cambridge (1989)