論文 / 著書情報
Article / Book Information

Title	Relationship Extraction Methods Based on Co-occurrence in Web Pages and Files
Author	Qiang Song, Yousuke Watanabe, Haruo Yokota
Citation(English)	Proceeding of The 13th International Conference on Information Integration and Web-based Applications & Services, , , pp. 82 - 89
Issue date	2011, 12
Copyright	Copyright (c) 2011 Association for Computing Machinery
Note	This is the definitive version http://www.iiwas.org/conferences/iiwas2011/, http://dx.doi.org/10.1145/2095536.2095552
Set statement	(c) ACM, 2011. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Proceeding of The 13th International Conference on Information Integration and Web-based Applications & Services, , pp. 82 - 89, http://dx.doi.org/10.1145/2095536.2095552

Relationship Extraction Methods Based on Co-occurrence in Web Pages and Files

Qiang SONG[*]
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku
Tokyo, Japan
soukyou@de.cs.titech.ac.jp

Yousuke WATANABE[†]
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku
Tokyo, Japan
watanabe@de.cs.titech.ac.jp

Haruo YOKOTA[‡]
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku
Tokyo, Japan
yokota@cs.titech.ac.jp

ABSTRACT

Every day, information on the Web becomes increasingly enriched. Web access is now very useful in many aspects of daily life, particularly for writing documents and programs. In fact, it has become quite usual to edit files while referring to information on the Web. During the file-editing process, we usually visit so many Web pages that we cannot remember all of the relevant ones. Later, if we want to revisit the same Web pages to modify some part of a file, it can be very hard to track down the Web pages originally referred to. In this paper, we propose methods for finding relationships between files and Web pages based on the co-occurrence of data in Web-access logs and file-access logs. These relationships are very useful for revisiting Web pages related to target files. To analyze co-occurrence in these two types of access logs, there are two approaches for merging the logs, involving a trade-off between accuracy and execution time. We call them the Pre-Merge and Post-Merge methods, and we have evaluated these two methods using actual access logs.

Categories and Subject Descriptors

E.5 [**Files**]: Sorting/searching; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*; H.2.8 [**Information Systems**]: Database ManagementDatabase Application[Data Mining]

General Terms

Experimentation

[*]Department of Computer Science, Graduate School of Information Science and Engineering

[†]Global Scientific Information and Computing Center.

[‡]Department of Computer Science, Graduate School of Information Science and Engineering

iiWAS2011, 5-7 December, 2011, Ho Chi Minh City, Vietnam..

Keywords

file searching, Web page, relationship between Web pages and files, co-occurrence frequency

1. INTRODUCTION

The Internet is sufficiently established that information on the Web is significantly enriched every day. This means that the information on Web pages has become increasingly useful in daily life. It has therefore become very common for us to refer to information on the Web, particularly when writing documents or programs.

For example, suppose that a user has edited a program file while referring to Web pages describing the program-language specifications or the algorithm description, and the user tries later to modify the program during debugging. It can be very hard to find not only the Web pages but also the program files themselves, because there may be many files as well as many Web pages.

Traditional desktop search engines [8, 5, 6] and Web search engines are useful for finding an individual file or group of Web pages if the user can remember some related keywords. However, if the user cannot recall appropriate keywords, it will be rather difficult to find the file or the Web pages. For files that include only programs, experimental data, and figures, for example, it becomes even more difficult, because these files have no text corresponding to the keywords. Moreover, current desktop search engines and Web search engines cannot provide information about any relationships between the file and the Web pages.

We have previously proposed methods to search for files containing no keywords [17, 13, 14, 15, 19] by analyzing file-access logs, based on the concept that related files are frequently accessed at the same time. We have also evaluated the effectiveness of these methods. However, although they can search for files, they cannot extract information about any relationships between files and Web pages. Therefore, we cannot apply these methods directly to the scenario described above.

In this paper, we propose new methods for extracting relationships between groups of files and Web pages, both of which involve the work of the same user. The proposed methods output URLs for files and Web pages that co-occur in the file logs and Web-access logs, in the form of a virtual folder. This means that the virtual folder will contains URLs for files and Web pages that are related to the same "task". Here, the term "task" stands for a work unit with a unifying context. For example, the task "writing thesis"

might refer to the image files, table files, and LaTeX files that were created while writing a thesis, together with some Web pages viewed when identifying related publications.

To analyze co-occurrence in the two types of access logs, file access and Web access, there are two approaches to merging the logs. One method merges them first, whereas the other analyzes each log separately and finds matches between them. We name them the **Pre-Merge** and **Post-Merge** methods, respectively. There is a trade-off between them involving accuracy and execution time. We have implemented the two methods and compared their performance using actual access logs.

The remainder of this paper is organized as follows. In Section 2, we explain how the two proposed methods, Pre-Merge and Post-Merge, work. In Section 3, we compare these methods by reporting on experiments using actual access logs. Related work is presented in Section 4. Section 5 contains conclusions and proposals for future work.

2. PROPOSED METHODS

We first outline our goal and approach, and then describe the logs used in this approach. We then propose two methods, Pre-Merge and Post-Merge, for extracting information about the relevant groups of files and Web pages from the logs.

2.1 Goal and Approach

Our goal is to search for correlated files and Web pages belonging to the same task. When a user edits specific files and visits specific Web pages more frequently during this task, the relationship between these files and Web pages is considered significant. Based on this assumption, we propose to use file-access logs and Web-page-access logs to analyze the relationship between the files and Web pages referred to.

We have previously proposed methods for using file-access logs to analyze relationships between files to search for related files in the absence of keywords [17, 14, 13, 15, 19]. We analyzed the timing overlaps between file accesses from the file-access logs, and combined this with a full-text search engine [17]. We then proposed a method for preparing a virtual folder containing related file data [14]. We also applied a data-mining method based on frequent-item sets [9] to identify combinations of files that appear frequently in the file-access logs [13], combined with an overlap method [15] and file-operation (rename, move, and copy) logs [19]. However, these methods cannot treat combinations of files and Web pages.

To extract the relationships between files and Web pages, we need to handle two types of logs, namely file-access logs and Web-access logs. The file-access logs can be derived either from a file server or by using dedicated tools. There are two types of Web-access logs, namely browser logs and proxy logs. The pros and cons of these logs will be discussed in the next section.

In this paper, we propose two methods for utilizing the contents of the file-access logs and Web-access logs. The approaches of these two methods are as follows.

- **Pre-Merge** method: first integrate the file-access logs and Web-access logs using timestamp information, and then extract frequent-item sets from the integrated logs.
- **Post-Merge** method: first apply frequent-item set extraction to each of the individual logs and then associate both frequent-item sets based on the timestamp information.

As an internal process in both methods, we adopt the FI method (Frequent Itemsets discovery from file-access logs) proposed in [13], where access logs are converted into transactions based on a specified duration, and frequent-item sets are extracted from the transactions. The details of these methods will be explained in Sections 2.3 and 2.4, respectively.

2.2 Access Logs

Two types of access logs are used in the proposed methods, namely file-access logs containing the reading and writing history of files, and Web-access logs that record the Web-page viewing history.

2.2.1 File-access Logs

The history of file accesses is recorded in an access-log file in the file server. Some file servers, such as Samba, have a built-in function for maintaining the file-access log that can be used in this approach [17]. In this paper, we use a tool called FaccLog [3] that operates on a file server to maintain more detailed logs, particularly for experimental evaluations. The information written in the log comprises the user's connection time, action, user name, server IP, client IP, and file path.

2.2.2 Web-access Logs

Web-access logs can be obtained in two ways. We can obtain them from HTTP proxy servers (squid [7]), or they can be obtained directly from a user's browser, such as Firefox [4], by using add-ons such as Boomtango [1]. The differences between the two types of logs are as follows.

Advantage of proxy logs

- It is easy to collect Web-access logs without considering the different types of browsers, by simply recording all activities of all machines inside the target local area network that use the proxy.

Disadvantages of proxy logs

- There are many irrelevant records within the logs. For example, when a user loads a single page, all image files inside the page are recorded separately. Proxy logs therefore tend to be much larger than browser logs.
- Because a browser has a cache to improve the response time for frequently visited Web pages, multiple accesses to the same Web page over a short duration are not retained in the proxy logs. This reduces the accuracy of the logs.

Advantage of browser logs

- This records Web pages that are visited directly via user actions, and the log files are relatively small.

Disadvantages of browser logs

- To collect Web-access logs, we need to visit all target client machines.

- The log-recording mechanisms of different browsers vary. For example, only the most recent access to a Web page is saved in the log file in Chrome [2], while Firefox [4] records all accesses for each page.

We compared the effects of using proxy and browser logs by experiment, as reported below.

2.2.3 *Log Filtering*

Because a log file may contain many irrelevant records, we develop separate filters for file-access logs and Web-access logs to remove unnecessary information.

For file-access logs, we prepare in advance a list of file extensions that indicate file types, to filter out unnecessary file accesses such as accesses to system files. The logs also contain many records of accesses to folders. Because these accesses are not used in our current approach, we also remove them. (We may use them in future work.) If there are many file accesses within a very short duration (several per second), these accesses can be considered automatic accesses generated by software, because a human user could not access so many files so quickly. Therefore, if more than N records are accessed in a second (in our experiments, N was four), we remove them from the log file. In this way, many irrelevant access-log entries, such as file scans by antivirus software, can be removed.

On the other hand, for the Web-access logs, we have to consider the behavior of users. It is usual to traverse several Web pages by clicking links to reach a useful Web page, even when using a Web search engine. The viewing duration for these stopover Web pages tends to be short until the target Web page is reached. Therefore, if a user moves quickly to another Web page (within three seconds in our experiments), we treat the Web page as a stopover, and remove the reference from the logs.

We also remove logs of visits to specific Web sites. For example, users tend to access a Web mailbox such as Gmail frequently, independently of the current task. These types of Web access are irrelevant to finding relationships with file accesses. We therefore exclude these logs using a site exclusion list. In this list, we also exclude logs of social networking service sites such as Twitter and Facebook because the accesses on these sites hardly have any relationship with the current task at the most time.

In proxy logs, accesses to images within a Web page are also recorded. Since these files, used to construct parts of a Web page, are not a target in this approach and can be found by the URL of Web page later, we remove logs accessing these image files based on their file extensions: .jpg/.JPG/, .png/.PNG/, or .jpeg/.JPEG.

We delete logs for HTTP response status codes other than 200, which is the standard response for successful HTTP requests.

2.3 Pre-Merge Method

In the Pre-Merge method, we first merge the file-access logs and Web-access logs, and then extract frequent-item sets from the integrated log. Its process flow is shown on the left side of Figure 1. We describe each step below. Log filtering is performed by the procedure described in Section 2.2.3.

2.3.1 *Log Merging*

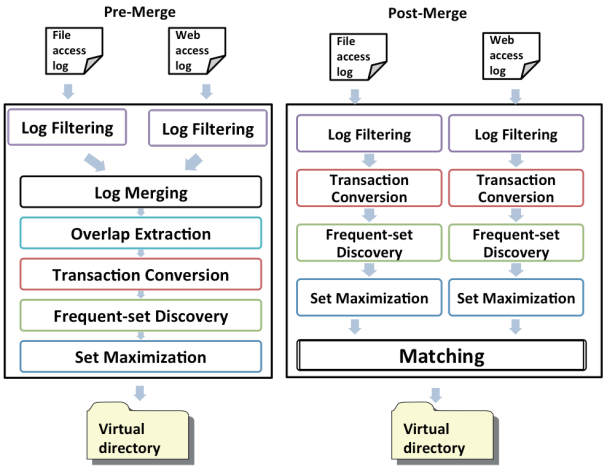

Figure 1: Processing Flow for Pre-Merge (Left) and Post-Merge (Right)

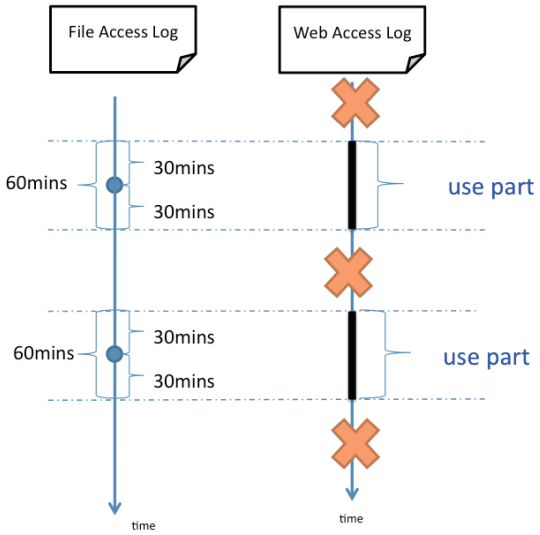

Figure 2: Extracting the Overlap between Two Log Files

We first unify the formats of the two types of log files to enable them to be merged, then use the timestamp information to sort them in time order.

2.3.2 *Overlap Extraction*

It is of course very common for users to browse Web pages independently of editing a file. Because the goal of this research is to extract the relationship between files and Web pages, we extract the overlapped part of the two logs in terms of access time. We remove Web-access logs outside a particular period (30 minutes in our experiments) before and after a file access has occurred 2.

2.3.3 *Transaction Conversion*

To enable application of a transaction-based algorithm for mining associations between items [9], we convert the integrated access logs into a sequence of transactions containing log records appearing within a predefined period (parameter

$TransactionTime[s]$).

For example, if the integrated logs contain the records "1:02:04 w.docx", "1:05:28 www.example.com", "1:40:39 y.txt", and "2:04:06 z.pptx", and the parameter $TransactionTime[s]$ is set to 3600[s], the converted transactions are transaction$_1$ = "2.docx, www.example.com, y.txt" and transaction$_2$ = "z.pptx".

2.3.4 Frequent-set Discovery

In this step, we discover frequently appearing sets from transactions by using the Apriori algorithm [9]. We extract combinations of the same file or Web page occurring over minimum times predefined by the parameter ($MinSupport$). This frequent-access-set discovery eliminates accidental simultaneous-access combinations that are not in fact related.

2.3.5 Set Maximization

The previous step tends to generate a large number of small sets containing the same elements. We merge the small sets into a maximized one. For example, if the frequent-access-set discovery generates $\{a, b, c, d\}$ and $\{a, b, c, e\}$, the files or Web pages $\{a, b, c, d, e\}$ are probably related to the same task, in spite of there being no set $\{a, b, c, d, e\}$ in the frequent-access-set discovery results.

To merge similar small sets into a larger set, we use the *Dice* coefficient, defined by the following expression:

$$\frac{2|A \cap B|}{|A| + |B|} \geq Dice\ threshold$$

.

If the Dice coefficient between two item sets, A and B, is equal to or greater than a predefined threshold (parameter *Dice threshold*), the sets A and B are combined into a larger set.

2.4 Post-Merge Method

In the Post-Merge method, we first apply the Apriori algorithm [9] to the file-access log and Web-access log individually, and then merge the results of the two frequent-item sets. The right part of Figure 1 shows the process flow for this method. Because the sets obtained by Log Filtering (2.2.3), Transaction Conversion (2.3.3), Frequent-set Discovery (2.3.4), and Set Maximization (2.3.5) are the same as those in the Pre-Merge method, we need only explain the Matching step in this section.

2.4.1 Matching

After the frequent-access-set discovery for each log file, we obtain the frequent-item sets for both files and Web pages. Because we assume that files and Web pages for the same task are accessed during the same time period in a transaction, we combine file sets with Web-page sets based on the degree of overlap in access times for each element.

If the number of element pairs of overlapped transactions is greater than or equal to a predefined parameter ($Overlap$), we consider the two frequent-access sets to have a strong relationship, and combine them. The matching condition for a frequent-access set of files f_p and a frequent-access set of Web pages w_q is defined as follows:

$$|TranIDs(f_p) \cap TranIDs(w_q)| \geq Overlap$$

.

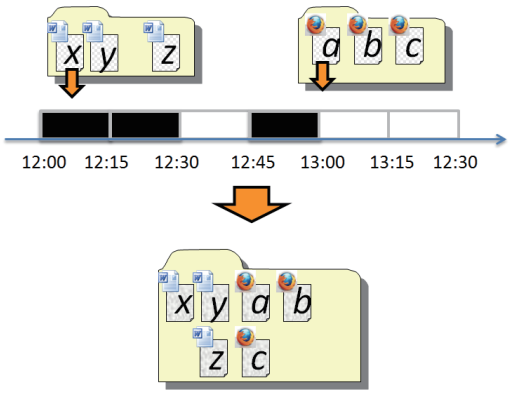

Figure 3: Matching in Post-Merge

Here, $TranIDs(x)$ means the set of transaction IDs containing x.

For example, suppose that we obtain a frequent-item set $\{x, y, z\}$ for files, and $\{a, b, c\}$ for Web pages, as shown in Figure 3. When the parameter value for $Overlap$ is three and that for $TransactionTime$ is 15 minutes, and the file x and Web page a are both accessed in the three transaction-time periods "12:00–12:15", "12:15–12: 30", and "12:45–13: 00" (corresponding to the box colored black in Figure 3), these two sets are combined.

3. EXPERIMENTS

We performed three experiments to find the optimal parameter values for the Pre-Merge and Post-Merge methods, and compared these two methods in terms of accuracy and execution time.

3.1 Environment

We used the following environment for the experiments:

- OS: Linux CentOS 4.3
- CPU: Dual Core AMD Opteron(tm) Processor 280 *4
- Memory: 16 GB (4 GB *4)

3.2 Data

The file-access logs used in these experiments were derived from the file server in our research laboratory using a tool called Facclog. For the Web-access logs, the proxy logs in the HTTP proxy server operating in our laboratory were recorded, and the browser logs were from the Firefox browsers installed on each client. We used the access logs of two distinctive users in these experiments. The details are shown in Table 1.

We prepared three datasets, which were various combinations of the users and Web-access log types, as follows:

- Dataset1: File-access logs + Browser logs for User1
- Dataset2: File-access logs + Proxy logs for User1
- Dataset3: File-access logs + Proxy logs for User2

The reason for preparing these three datasets was that Dataset1 and Dataset2 are for the same user, the difference being that they combine the same file-access logs with different types

Table 1: Data

User	Term	Log	Size (byte)	File/Web page numbers
User1	2010/09/24 - 2010/11/21	File	23,624,947	472
		Browser	2,036,315	1731
		Proxy	52,332,311	3939
User2	2010/10/1 - 2010/10/31	File	42,450,684	506
		Proxy	68,546,676	3036

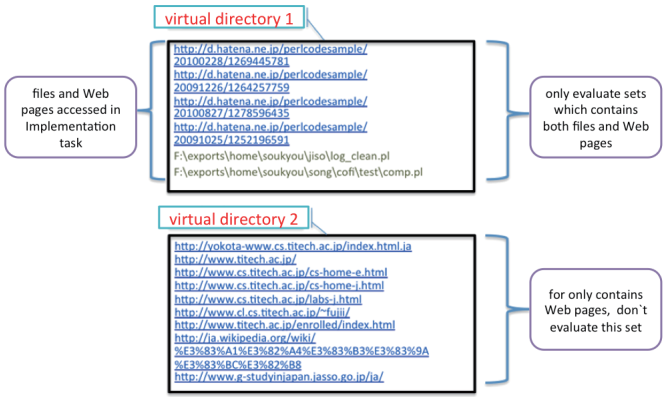

Figure 4: Examples of Virtual Folders

of Web-access logs. We aimed to identify the better type of Web-access log by comparing results for Dataset1 and Dataset2. On the other hand, Dataset2 and Dataset3 are from different users using the same type of Web-access log. We aimed to find differences between users by comparing the results for Dataset2 and Dataset3.

To evaluate our experimental results, we created correct Answer Sets by hand for four tasks. The details are shown in Table 2.

3.3 Output Examples

We describe examples of the output from our proposed methods in this section. As illustrated in Figure 4, we use *virtual folders* to show the results. Here, a virtual folder corresponds to a task. Two virtual folders are depicted in Figure 4.

Virtual Folder1 contains two Perl files and four Web-page URLs belonging to an implementation task for User1. The desktop search engines could find the two Perl files but were unable to present the four Web pages to the users at the same time. Our methods were able to identify both files and Web pages at the same time.

Virtual Folder2 contains only Web-page URLs. The reason for this kind of output is that, when a user browses the Web, the user will occasionally touch a file. Therefore, these records will not be deleted in the log-filtering step. However, in the frequent-set-discovery step, the records for files are removed, leaving only frequently accessed records of Web pages. Such sets are the results of Web browsing by a user. The purpose of this paper is the extraction of relationships between files and Web pages, but we would not use such result sets in experiment.

3.4 Evaluation

We use $Precision$, $Recall$, and $F-measure$ [10] to evaluate the results. The definitions of these metrics are:

$$Precision = \frac{|Results \cap Examinees|}{|Results|} \quad (1)$$

$$Recall = \frac{|Results \cap Examinees|}{|Examinees|} \quad (2)$$

$$F{-}measure = \frac{2 * Precision * Recall}{Precision + Recall}. \quad (3)$$

We only evaluate result sets that refer to both files and Web pages. With reference to Figure 4, we do not evaluate Virtual Folder2 because it refers to no files. If the items of the correct answer set are distributed across more than one result set, we compare each result set with the answer set and choose the one with the highest value for Recall.

3.5 Experimental Results

3.5.1 Experiment 1

The goal of Experiment 1 was to determine the optimal parameter values for the two methods. We changed the parameter values individually to derive the highest F-measure value. The results are shown in Table 3. The order of the parameters listed in the Parameters column of the table is "TransactionTime – MinSupport – Threshold value for Dice – Overlap (only used in Post-Merge)".

We found that the optimal parameter values for the two methods are different. Between different users, because of differences in their behavior, the optimal parameters are also different.

For $TransactionTime$, too small a value will limit the opportunities for related items to combine with each other. Conversely, too large a value will lead to results containing irrelevant items. We found that the optimal setting was 1800[s]for the Pre-Merge method with User1, it was 3600[s] for Post-Merge with User1, it was 3600[s] for Pre-Merge with User2, and it was 5400[s] for Post-Merge with User2.

For the *threshold value for Dice*, we found it differed significantly for different users. Except for User2's optimal setting being 0.7, the optimal threshold value is between 0.1 and 0.4. Setting too high a threshold value will limit set unification and will lead eventually to low Recall values.

For *Overlap* in the Post-Merge method, values of 3 or 4 are optimal in all cases. If we set a value of up to 6, the frequent sets for files and Web pages will no longer unite, meaning that we cannot get a final result that refers to both files and Web pages.

3.5.2 Experiment 2

In Experiment 2, we used the optimal parameter values derived in Experiment 1 to compare the accuracy of the Pre-Merge and Post-Merge methods. The results are depicted as graphs in Figure 5.

The results indicate that Pre-Merge performs significantly better than Post-Merge on all Precision, Recall, and F-measure metrics. We consider that there are two reasons for this.

Firstly, from the nature of Web pages, there are only a few Web page sets that appear frequently. For example, suppose that files "a, b, and c" and Web page "w" belong

Table 2: Answer Sets

User	Set	Content	Extensions	Files	Web page numbers	Total numbers
User1	S_1	Implementation of the core part	java, pptx	24	1	25
	S_2	Implementation of the filter	pl, txt	8	14	22
	S_3	Seminar	pdf, pptx	2	7	9
User2	S_4	Academic management	bmp, pdf, xlsx doc, docx, pptx	14	4	18

Table 3: Optimal Parameter Values

Dataset	Method	Parameter	Answer set	Precision	Recall	F-measure	Execution time[s]
Dataset1	Pre-Merge	1800-2-0.1	S_1	0.61	0.76	0.68	
			S_2	1.00	0.86	0.93	
			S_3	1.00	0.67	0.80	
			average	0.87	0.76	0.80	35.31
	Post-Merge	3600-2-0.4-3	S_1	0.56	0.76	0.64	
			S_2	0.21	0.13	0.16	
			S_3	0.00	0.00	0.00	
			average	0.26	0.30	0.27	18.85
Dataset2	Pre-Merge	1800-2-0.3	S_1	0.00	0.00	0.00	
			S_2	1.00	0.36	0.53	
			S_3	0.80	0.44	0.57	
			average	0.60	0.27	0.37	38.79
	Post-Merge	3600-2-0.4-3	S_1	0.56	0.76	0.64	
			S_2	0.21	0.13	0.16	
			S_3	0.00	0.00	0.00	
			average	0.26	0.29	0.27	22.81
Dataset3	Pre-Merge	3600-2-0.3	S_1	0.80	0.22	0.35	
			average	0.80	0.22	0.35	67.38
	Post-Merge	5400-2-0.7-4	S_1	0.10	0.17	0.13	
			average	0.10	0.17	0.13	24.38

to the same task, and that every time a user edits file "a", "b", or "c", page "w" is accessed. Under this assumption, the frequent-access set $\{a, b, c, w\}$ will be derived in the Pre-Merge method. However, for Post-Merge, the frequent sets for files and Web pages are calculated separately, so we will derive only $\{a, b, c\}$ because no other Web pages are accessed together with "w". In other words, "w" will not be extracted into any frequent set before matching, because it is a singleton with respect to Web accesses. That is, because the frequent sets for Web pages tend to be small in the Post-Merge method, its accuracy will be worse. The same phenomenon is occurred in the case that a file is a singleton in the unified frequent-access set.

Secondly, in the matching step of Post-Merge, it is possible that some unrelated items are matched into the final result. This also decreases the accuracy of the Post-Merge method. It would appear to be necessary to improve the matching process in future.

In addition, the period of collected logs used in these experiments was about 1–2 months. When we apply the two methods to longer periods of log data in the future, we would expect that the accuracy differences will be larger.

Comparing the results for Dataset1 and Dataset2, the use of browser logs performs better than the use of proxy logs on all Precision, Recall, and F-measure metrics. We consider that there are also two reasons for this.

Firstly, the proxy logs contain too many irrelevant logs such as records of advertising sites and loading records for material embedded in a Web page. These decrease the Precision values for the proxy logs directly.

Secondly, because of the cache function in browsers, the browser will not load the same Web page from Web servers again until the cache expires. Therefore, the access history for the same Web pages within a short period may be found in the browser logs, but not in the proxy logs. This decreases the Recall values for the proxy logs.

3.5.3 Experiment 3

Experiment 3 uses the optimal parameter values from Experiment 1 to compare the two methods in terms of execution time. The results are shown in Figure 6.

For Dataset1, the Pre-Merge method takes 1.87 times longer than the Post-Merge method. This is 1.70 times longer for Dataset2, and even reaches 2.76 for Dataset3. We consider that there are two reasons for this.

Firstly, the frequent-access-set discovery step is the most time-consuming step for both methods, and its execution time depends on the size of the logs. Because Pre-Merge merges both logs first, the size of the logs becomes larger before calculating the frequent sets.

Secondly, when Pre-Merge combines the two logs, it is necessary to sort the two logs in order of timestamps. This is also time consuming.

We should expect that, when we apply the proposed methods to larger log datasets in future, the differences in exe-

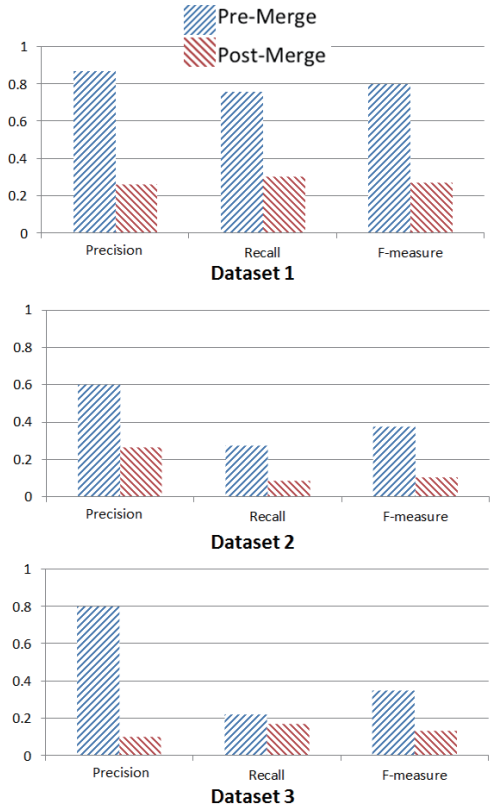

Figure 5: Comparison of the Two Methods in Terms of Accuracy

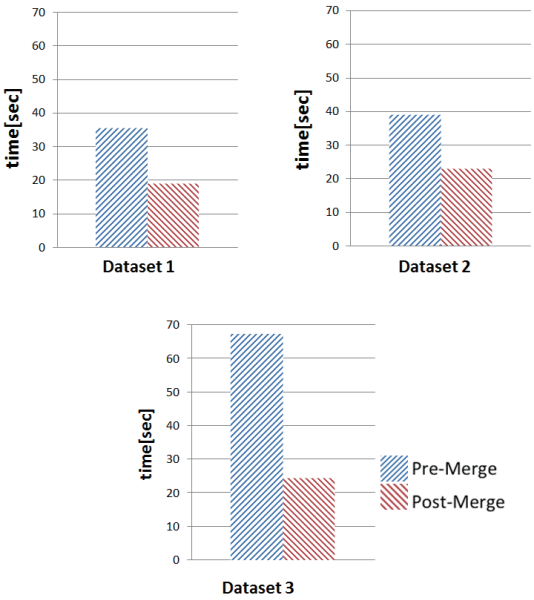

Figure 6: Comparison of the Two Methods in Terms of Execution Time

cution time will be even larger. Therefore, improving the processing efficiency in Pre-Merge is a big problem to be solved.

4. RELATED WORK

There are some popular desktop search engines, such as Windows Search [8], Google Desktop [5], and Spotlight [6] in Mac OS X. However, because these desktop search engines all use textual information inside files for keyword searching, there is the problem that they cannot find nontext-based files such as image and video files. Another problem is that they all require the user to input appropriate keywords, whereas files associated with the same piece of work do not always use the same keywords. For this reason, using traditional desktop search engines to search for all files belonging to a task without missing some of them is very difficult, particularly when the scale of the task is large.

FRIDAL [17, 18] is a system for solving the problem of desktop search engines finding files containing no textual information. FRIDAL calculates the relationship between files using information such as overlap counts, overlap durations, and differences in open times in the file-access logs. It then utilizes the relationships to provide related files to add to the results of a traditional desktop search engine. Therefore, it is capable of providing related files that do not contain the given keyword directly.

The approach of FRIDAL is expanded in the Clustering using Overlap of file-use (CO) method [14], the FI [13] method, and the COFI [15] method. The goal of each of these three methods is to list files associated with the same task in a virtual folder, even though the files may be distributed across multiple directories in the file system. In other words, the goal of these methods is to group files by task.

The CO method discovers files whose times of use overlap. It then uses hierarchical clustering to group the files into virtual folders. The CO method can find files distributed across the file system. However, the problem is that even files belonging to the same task may not overlap. Thus, CO cannot find files whose access times do not overlap.

The FI method is an improved version of the CO method. It converts file accesses into a sequence of transactions defined over a certain period, and then finds the frequently appearing combinations of files. It also outputs its results into virtual folders.

The COFI method is a method based on both the CO and FI methods. It first derives the relationships between file accesses from file-access logs using the concept of a transaction. It then calculates the relationships between the transactions using overlap information, similarly to the CO method.

Based on an assumption that a set of Rename, Move, and Copy (RMC) operations tends to initiate a new task, the SUGOI [19] system detects two types of tasks, FI tasks and RMC tasks, from file-access logs. An FI task corresponds to a group of files frequently accessed together; an RMC task is generated by RMC operations and then constructs a graph of intertask relationships based on the influence of RMC operations and the similarity between tasks. Experiments using actual file-access logs indicate that the SUGOI system significantly improves search results of the COFI methods.

Connections [16] is a file search system that uses contextual information with the aim of enhancing full-text search results. Connections generates a relational graph of files based on traces of filesystem calls. To discover relationships, it splits access logs into multiple relation windows and identifies the input and output files in each window by

considering which operation is performed on the files. Connections then creates links with weight 1 from input files to output files or increases the weight of existing links. It uses a Basic-BFS (Breadth First Search) algorithm to propagate the weights of keyword-containing files to keyword-lacking files to expand and reorder the results generated by a full-text search engine.

Semantic Desktop Search [11] extracts the attributes from each file, then search files using the attributes. However, it is necessary to write programs for each type of files to extract the attributes. Our methods are independent of file formats.

Because the targets of these above-mentioned methods are file accesses only, with no consideration of Web access, the main contribution of this paper is the combined treatment of file-access logs and Web-access logs to derive relationships between them.

OreDesk [12] is a tool that can search both files and Web pages browsed in the past, as a type of time machine. OreDesk records the behavior of specific applications by installing plug-ins in them. OreDesk then shows its results on a timeline. For example, by reviewing a previous timeline, users can find those Web pages that were browsed when creating a Word file. However, the problem is that, if the user cannot remember when the Word file was created, the user will have to check the whole timeline. A significant difference between OreDesk and our methods is that our methods can classify files and Web pages for a task automatically, by calculating access frequencies.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed two methods for extracting groups of files and Web pages used in the same task into a virtual folder that provides a means of revisiting related files and Web pages. We analyzed two types of access logs, file-access logs and Web-access logs, deriving relationships between files and Web pages by applying a data mining technique. We named the methods **Pre-Merge** and **Post-Merge**, where the former method first merges the two types of logs and then finds the relationships, whereas the latter method first analyzes each type of log separately and then finds matches between them.

We evaluated the two methods using actual access logs from our laboratory. In the experiments, we first determined optimal parameter values for each method, and then compared their accuracy, Precision, Recall, and F-measure. We also compared the execution times for the two methods, using the optimal parameter values. The results of the experiments indicate that there is a trade-off between accuracy and execution time for these methods. Pre-Merge performs better than Post-Merge for all of the Precision, Recall, and F-measure metrics. However, Pre-Merge takes from 1.70 to 2.76 times longer than Post-Merge. For the size of log files used in the experiments, the difference in accuracy is more significant than the execution times, and we conclude that the Pre-Merge method is preferable. However, for larger log files, the differences in execution time will increase and may become a serious problem. We therefore plan to evaluate the methods using longer-period logs in the future.

In future work, we also plan to refine the methods to improve accuracy, to reduce execution time, to determine optimal parameter values automatically from user behavior, and to experiment with larger log files.

6. ACKNOWLEDGMENTS

Part of this research was supported by MEXT via a Grant-in-Aid for Scientific Research #22240005.

7. REFERENCES

[1] Boomtango. http://www.boomtango.com/.
[2] Chrome. http://www.google.com/chrome/.
[3] Facclog. http://www2s.biglobe.ne.jp/~masa-nak/fal_down.htm.
[4] Firefox. http://mozilla.jp/firefox/.
[5] Google desktop. http://desktop.google.com/.
[6] Mac os x spotlight. http://www.apple.com/server/macosx/features/spotlight-server.html.
[7] Squid cache. http://www.squid-cache.org/.
[8] Windows search. http://www.microsoft.com/windows/products/winfamily/desktopsearch/default.mspx.
[9] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *Proc. ACM SIGMOD*, pages 207–216, 1993.
[10] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.
[11] A. Chirita, R. Gavriloaie, S. Ghita, W. Nejdl, and R. Paiu. Activity based metadata for semantic desktop search. *Proc. Second European Semantic Web Conference(ESWC 2005)*, pages 439–454, 2005.
[12] R. Ohsawa, K. Takashio, and H. Tokuda. Oredesk: A tool for retrieving data history based on user operations. *IEEE International Symposium on Multimedia (ISM 2006)*, pages 762–765.
[13] K. Otagiri, Y. Watanabe, and H. Yokota. Access-log analysis for virtual directory creation to restore files used in user's works. *Information Processing Society of Japan*, 2009.
[14] K. Otagiri, Y. Watanabe, and H. Yokota. Access-log based virtual directory creation to restore user's works. In *DEIM Forum 2009*, 2009.
[15] K. Otagiri, Y. Watanabe, and H. Yokota. Virtual directory creation considering access-times in frequent accessed file sets. *DEIM Forum 2010*, 2010.
[16] C. A. N. Soules and G. R. Ganger. Connections: using context to enhance file search. *Proc. of SOSP'05*, pages 119–132, 2005.
[17] T. Watanabe, T. Kobayashi, and H. Yokota. A method for searching keyword-lacking files based on interfile relationships. In *OTM '08: Proceedings of the OTM Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems*, pages 14–15, Berlin, Heidelberg, 2008. Springer-Verlag.
[18] T. Watanabe, T. Kobayashi, and H. Yokota. Searching keyword-lacking files based on latent interfile relationship. *Software and Data Technologies*, pages 236–244, 2010.7.
[19] Y. Wu, K. Otagiri, Y. Watanabe, and H. Yokota. A file search method based on intertask relationships derived from access frequency and rmc operations on files. *22nd International Conference on Database and Expert Systems Applications (DEXA 2011)*, pages 364–378, 2011.