論文 / 著書情報
Article / Book Information

Title	A File Search Method Based on Intertask Relationships Derived from Access Frequency and RMC Operations on Files
Author	Yi Wu, Kenichi Otagiri, Yousuke Watanabe, Haruo Yokota
Journal/Book name	Lecture Notes in Computer Science, Vol. 2011, No. 6860/2011, pp. 364-378
発行日 / Issue date	2011, 8
DOI	10.1007/978-3-642-23088-2_27
権利情報 / Copyright	The original publication is available at www.springerlink.com.
Note	このファイルは著者（最終）版です。 This file is author (final) version.

A File Search Method based on Intertask Relationships Derived from Access Frequency and RMC Operations on Files

Yi Wu[1]**, Kenichi Otagiri[2], Yousuke Watanabe[3], and Haruo Yokota[1]

[1] Department of Computer Science, Tokyo Institute of Technology
[2] Cowbell Engineering Corporation
[3] Global Scientific Information and Computing Center, Tokyo Institute of Technology
{goi,otagiri,watanabe}@de.cs.titech.ac.jp,yokota@cs.titech.ac.jp
http://yokota-www.cs.titech.ac.jp

Abstract. The tremendous growth in the number of files stored in filesystems makes it increasingly difficult to find desired files. Traditional keyword-based search engines are incapable of retrieving files that do not include keywords. To tackle this problem, we use file-access logs to derive intertask relationships for file search. Our observations are that 1) files related to the same task are frequently used together, and 2) a set of Rename, Move, and Copy (RMC) operations tends to initiate a new task. We have implemented a system named SUGOI, which detects two types of task, 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. In utilizing detected tasks and intertask relationships, our system expands the search results of a keyword-based search engine. Experiments using actual file-access logs indicate that the proposed approach significantly improves search results.

Keywords: file-access logs, desktop search, full-text search, task mining

1 Introduction

The explosion in the volume of information that people handle has been accompanied by daily increases in the number of files stored in filesystems. Many of these are unstructured files, such as images, diagrams, and numerical-data files, which do not contain any appropriate text that can be used for search. As these files do not include the target keywords, traditional text-based desktop keyword search engines are not useful for finding them.

A number of systems have been developed for desktop keyword search, such as Google Desktop Search[3], Microsoft's Windows Desktop Search[5], and Spotlight on Mac OS X[1]. These systems all develop indexes for high-speed search,

** The author is currently with NTT DATA Corporation.

and some use a thesaurus and meta-information such as the file name, creation time, and file type, to improve search performance. However, it is still difficult to search for files that do not include text.

Recently, some desktop search systems, such as FRIDAL[9, 10] and Connections[8], have been proposed to tackle the problem of the association between files based on the co-occurrence information derived from file-access logs. These systems try to capture the relationships between files based on users' access patterns. These approaches are effective in searching for files that do not contain text. However, sometimes the searches return irrelevant files or fail to return relevant files because infrequent but important operations for files could not be found and temporary simultaneous access of files influences the results from these approaches.

In this paper, to improve the accuracy of search results based on file-access logs, we focus on a task where a user accesses multiple files to accomplish his/her goal. In other words, most files accumulated in a filesystem must be related to individual tasks. We refer to a group of files that are used to accomplish a particular goal as a "task." Thus, in this paper, the term "task" represents a logical unit of files to be processed in a computer system, such as collating some experimental data, writing a report or article, or preparing presentation slides.

Based on the concept of task, we propose a file search system named *SUGOI*—Search by Utilizing Groups Of Interrelated files in a task—which consists of two parts: a task mining component and a file search component. The task mining component extracts tasks and discovers the interrelation between tasks from file-access logs. The file search component incorporates the task mining results within the search results of a traditional desktop search engine to achieve an accurate keyword-based file search.

We observed that 1) files related to the same task are frequently used together, and 2) a set of Rename, Move and Copy (*RMC*) operations tends to initiate a new task. Therefore, the task mining component of SUGOI extracts two types of task: *FI* (Frequent Itemset) tasks and *RMC* tasks. We then propose formulas to combine FI and RMC tasks.

We also consider the graph of intertask relationships for retrieving related files. Assuming that the greater the number of identical files accessed during different tasks, the stronger is the interrelation between these tasks, we build similarity links between these tasks. In addition, we generate RMC links between tasks, assuming that users tend to RMC files for reuse in related tasks. For example, as researchers tend to use the same graph of experimental results in both their articles and their presentation slides, they may copy the file of the graph from the article folder to the presentation folder. In this case, the copy operation indicates a strong relationship between the tasks of writing the article and of preparing the presentation. To represent such intertask relationships, the weight of an RMC link is computed considering the type of the operation and its direction. We propose a formula to handle both types of links.

After the tasks and intertask relationships have been extracted, the file search component of SUGOI expands the search results of a traditional keyword-based

search engine using the mined tasks and intertask relationships, and ranks the search results. We evaluate the proposed system using actual file-access logs. The experimental results indicate that the proposed approach significantly improves recall and F-measure.

The remainder of this paper is organized as follows. Section 2 presents related research on desktop search. Section 3 gives a detailed explanation of SUGOI and Section 4 describes the experimental results and investigation. Section 5 concludes this paper and indicates future work.

2 Related Work

With the great increase in the capacity of storage devices, the volume of all types of data is steadily increasing, with most being file-based and unstructured. Users require more effective and simpler mechanisms for managing vast numbers of files, such as locating desired files easily from files scattered across different directories. Traditional content-based search tools cannot deal with files that do not include the query keywords, and supported file formats are restricted. For the purpose of enhancing full-text search, much research has been done using file-access logs. This section describes three previous studies that used file-access logs for file search.

As mentioned in Section 1, Connections [8] 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. By contrast, we utilize file-access logs to group files into tasks and identify the interrelations between tasks rather than the relationships between files.

FRIDAL[9, 10] is another system for searching keyword-lacking files by using the interfile relationship. FRIDAL exploits open/close logs in order to derive the duration for which the file is used. Assuming that files used at the same time have some relevance to each other, FRIDAL calculates the interfile relationship by using the co-occurrence data between files in access logs. It finds keyword-lacking files by using a Basic-BFS-like method. By contrast, SUGOI emphasizes tasks and only uses access patterns that occur frequently. Furthermore, we take RMC operations into account in calculating the weight of semantic links between tasks.

The iMecho[2] system performs task mining similar to SUGOI, building three types of associations: Content-based Associations (CA), Explicit Activity-based Associations (EAA), and Implicit Activity-based Associations (IAA). It reranks the results of a full-text search engine, using a random walk algorithm based on

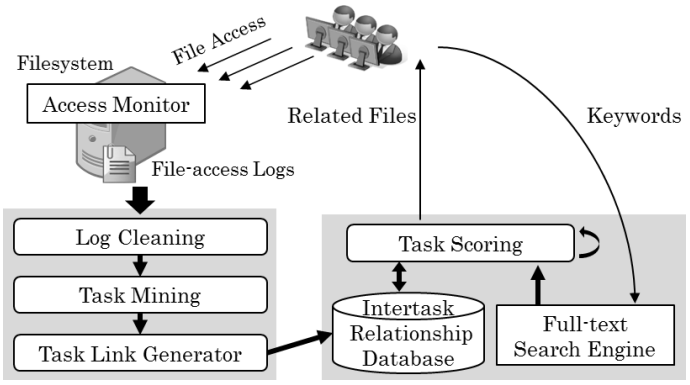

Fig. 1. Overview of SUGOI

PageRank[7]. The purpose of the task mining in iMecho is to generate IAA links between files; it does not consider the relationship between tasks. In addition, the ranking score used in iMecho is defined as the product of a full-text search result and a link analysis result, so iMecho does not extend the result of a full-text search to include non-text files. By contrast, SUGOI can extend the result of a full-text search by task mining and detecting the relevance between tasks.

3 Proposed Approach

We propose an approach for searching files by introducing the concept of "tasks." Before the search process, we cluster related files as a task and discover the correlation between tasks by exploiting file-access logs. We then expand the results from a traditional keyword-search engine using the tasks and intertask relationships. We have implemented a prototype system using the proposed approach and named it "Search by Utilizing Groups Of Interrelated files in a task" or "SUGOI" for short. An overview of SUGOI is illustrated in Fig. 1. To trace user access as file-access logs, SUGOI places an access monitor on the filesystem. The file-access logs should include access time, information to identify the client, the path of the target file, and the operation on it. After cleaning the log, the system extracts the semantic file groups as tasks, and builds weighted links between tasks. Following a user's search request, SUGOI searches for files by combining the context of tasks with the traditional full-text search results to calculate the task scores. The details of the proposed methods (apart from log cleaning) are described in the following subsections. Log cleaning is described in Section 4.1, because the process corresponds to the monitoring tool used in the experiments.

3.1 Task Mining

The purpose of task mining is to group those files that are related to the same task. We extract two types of task: FI tasks and RMC tasks. After the task

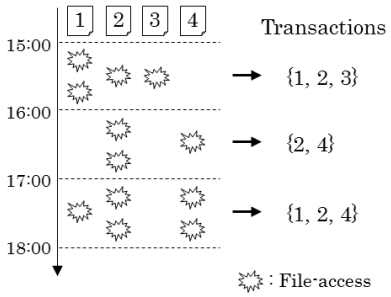

Fig. 2. Extraction of Transactions (TransactionTime = 3600 [s])

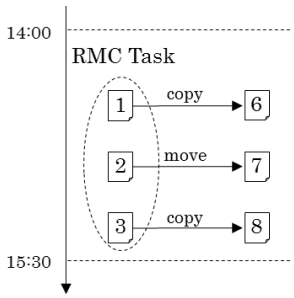

Fig. 3. Extraction of an RMC Task (RMCTaskTime = 1800 [s])

mining processes, a number of files accessed by a user belong to at least one task.

FI Task Mining The FI task, or Frequent Itemset task, is constructed from files accessed concurrently. The point is that files related to the same task tend to be accessed frequently within short time periods of each other. For example, when a user writes an article, he/she edits a TeX file as well as EPS files containing charts and produces a PDF file. With this characteristic, we propose a method to semantically group files as FI tasks by mining the access patterns that occur frequently and simultaneously.

We first convert the file-access logs into a set of transactions (Fig. 2) by splitting the logs into several transactions with a certain duration ($TransactionTime$). To discover the frequent patterns, we apply an existing algorithm, Eclat[11], to the set of transactions. Eclat is one of the best-known algorithms for mining frequent itemsets. It finds combinations of items whose occurrence frequency is greater than some minimum support value. In this study, we determine the minimum support count ($MinSuppCnt$) to extract frequent itemsets because file-access logs last for a long period but common tasks last for a fixed period of time and the number of occurrences of each individual file should be small. Finally, we extract the maximally frequent itemsets as FI tasks.

RMC Task Mining Assuming that the files RMCed together within a short time period are related to a task done in the past, we extract such file groups as an RMC task. To mine RMC tasks, we split access logs into multiple time windows with the same length of time and only extract the files that are the target of RMC operations (Fig. 3).

3.2 Intertask Relationship

This subsection presents our proposed formulas for weighting the similarity links and RMC links of tasks and calculating the relevance between tasks by considering the overlap and RMC operations between tasks. In addition, we consider reducing the weight of RMC links because the content of files created by RMC operations can differ from the original by modification.

Similarity Links Analysis Assuming that tasks that are strongly interrelated use a number of common files, we weight the similarity links between tasks based on the number of duplicated files in each task. The weight of a similarity link from task i to task j, which is expressed as $sim(t_i \rightarrow t_j)$, is given by Equation (1).

$$sim(t_i \rightarrow t_j) = \frac{|t_i \cap t_j|}{|t_i|} \tag{1}$$

Here, t_i and t_j represent the file sets of task i and task j, respectively.

RMC Links Analysis Supposing that a user RMCes files contained in related task to reuse in other tasks, RMC operations can express a strong relationship between tasks. We build RMC links of tasks by detecting an RMC operation between different tasks. To calculate the weight of RMC links, we introduce the element $rmc_f(f_m \rightarrow f_n)$, which presents the weight from file m to file n caused by an RMC operation.

$$rmc_f(f_m \rightarrow f_n) = \begin{cases} \alpha_1 \text{ if } f_m \text{ was renamed as } f_n, \\ \alpha_2 \text{ if } f_m \text{ was renamed from } f_n, \\ \beta_1 \text{ if } f_m \text{ was moved to } f_n, \\ \beta_2 \text{ if } f_m \text{ was moved from } f_n, \\ \gamma_1 \text{ if } f_m \text{ was copied to } f_n, \\ \gamma_2 \text{ if } f_m \text{ was copied from } f_n, \\ 0 \quad \text{otherwise.} \end{cases} \tag{2}$$

According to Equation (2), we assign rmc_f a constant value specified by the parameters (α_1, α_2, β_1, β_2, γ_1, γ_2). However, the degree of relevance of each file will be attenuated because the content of files coming from RMC operations probably differs from that of the original files by repeated edits.

In addition to rmc_f, we propose formulas to take into consideration the factors that cause reduced relevance. Thus, we define the reduction formulas using the elapsed time (Equation (3)), the sum of frequency of write operation

handled to file f_m and file f_n (Equation (4)), and the sum of the sizes of the changes to file f_m and file f_n after RMC operations performed (Equation (5)).

$$T(f_m, f_n) = \Delta_{time}(f_m, f_n)^{-\tau} \tag{3}$$

$$E(f_m, f_n) = \Delta_{edit}(f_m, f_n)^{-\epsilon} \tag{4}$$

$$S(f_m, f_n) = \Delta_{size}(f_m, f_n)^{-\sigma} \tag{5}$$

Here, τ, ϵ, σ are parameters. Considering the circumstances mentioned above, we calculate the weight of an RMC link by using rmc_f and the reduction functions (Equation (6)).

$$rmc(t_i \rightarrow t_j) = \sum_{(f_m, f_n) \in (t_i, t_j)} rmc_f(f_m \rightarrow f_n) * T(f_m, f_n) * E(f_m, f_n) * S(f_m, f_n) \tag{6}$$

Intertask Relevance To calculate the degree of association between tasks, we adopt the weight of similarity links and RMC links and use the parameter $\theta (0 \leq \theta \leq 1)$ to control which element to emphasize. The proposed formula is given by Equation (7).

$$R(t_i \rightarrow t_j) = \theta * sim_t(t_i \rightarrow t_j) + (1 - \theta) * rmc_t(t_i \rightarrow t_j) \tag{7}$$

3.3 Keyword-based File Search

In addition to files that include keywords, SUGOI can find files contained in tasks related to the keywords by combining the context and interrelation of tasks. This subsection describes the procedure for keyword-based search.

STEP 1: Identify tasks containing the keyword-containing files. The initial relevance score of a task to the given keywords is assigned using the file score given by the existing full-text search engine (Equation (8)).

$$score^0(q, t_i) = \sum_{f_m \in t_i} score_f(q, f_m) \tag{8}$$

Here, $score^0(q, t_i)$ denotes the initial score of task t_i to query q, and $score_f(q, f_m)$ denotes the relevance score of file f_m to the query. According to Equation (8), we use the summation of the file score to define the task score so that the score of the tasks that do not contain the keyword-containing files should be zero.

STEP 2: In this paper, we adopt a reflexive method based on Basic-BSF, which is used in Connections[8] and FRIDAL[9, 10], to find tasks that contain related files. To find the files that do not contain keywords, we iterate the calculation of the relevance score for all tasks K times for translating the relevance ($score^0(q, t_i)$) to other tasks for emphaisizing tasks that linked with

many related tasks and giving score to tasks that do not contain keyword-containing files. At the k-th ($1 \leq k \leq K$) calculation, the relevance score of task t_i is given by Equation (9).

$$score^k(q, t_i) = score^{k-1}(q, t_i) + \sum_{t_j \in InLink(t_i)} score^{k-1}(q, t_j) * R(t_j \rightarrow t_i) \quad (9)$$

Here, $InLink(t_i)$ expresses the set of tasks whose links point to task t_i.

STEP 3: Normalize the relevance score for all t_i and output files contained in tasks that satisfy $score^K(q, t_i) > TH_{score}$ as results, where TH_{score} is a threshold parameter.

4 Experiments

4.1 Experimental Environment

To verify the efficiency of SUGOI, we conduct evaluation experiments by using actual file-access logs gathered from a shared filesystem (Windows Server 2003 SP2, NTFS) used by our research group. To monitor the file access to the server, we use a tool named FAccLog[6]. FAcclog records the logs by monitoring access to the OS and the LAN adapter. Logging can be done almost in real time and includes read, write, create, delete, and rename operations. The full path of the target file is also included, but in the case of a rename, the path is recorded as "path before rename $\gg$ path after rename". Since the raw logs have noise and some necessary information is lacking, we apply the log-cleaning process before the main mining process.

The logs created by machine access mostly generate an incorrect task mining result, deriving groups of files with little reference to each other. Such kinds of access often come from background processes such as virus scanning, extracting indexes by a desktop search engine or making backups. In addition, in many cases, a lot of file access will occur in a short period. Therefore, we use two thresholds (TH_{min} and TH_{sec}) to eliminate the background machine access from logs. If the number of accesses occurring in a one minute/second range is larger than TH_{min}/TH_{sec}, all of the log entries are ignored. We also prepare a filter for detecting machine access by file extension.

The raw logs only distinguish rename operations; they do not move and copy. To find move operations, we treat entries whose directory part of the path changed after the rename as move operations. To detect copy operations, we look for a pattern of a create log entry occurring after a read log entry with the same filename, and treat it as a copy operation.

The implementation of SUGOI uses an existing full-text search engine named HyperEstraier[4]. In addition to plain text and HTML files, we can optionally search PDF, DOC, DOCX, XLS, XLSX, PPT, and PPTX.

Table 1. Experimental Datasets

Dataset	# log entries	# files	# files available for full-text search	# relevant files
A	3591	201	137	70
B	2808	416	113	25
C	3424	318	276	32
D	5911	764	311	84
E	8203	642	335	244
F	5123	3422	1152	227
G	13102	3258	338	73

4.2 Experimental Setup

The purpose of the experiments was to verify the effect of utilizing file-access logs for file search. We used datasets with keywords and lists of relevant files provided by seven testers. The system was evaluated by comparing the relevant files except the files that were deleted from the filesystem. A summary of the experimental datasets is given in Table 1. These datasets were gathered from April to July 2010. There are more than 2 GB of raw data and the summary is of the logs after log cleaning.

The parameters below were fixed during the experiments. In log cleaning, $TH_{min} = 30$, $TH_{sec} = 5$. To calculate the weight of RMC links, $(\tau, \epsilon, \sigma) = (0, 0, 0)$. In keyword-based search, $TH_{score} = 0$, $K = 3$.

4.3 Evaluation of Task Mining

To expand the full-text search results, we detect two types of tasks and derive the intertask relationships from file-access logs. As files contained in the same task are identified with each other, the results of task mining should be important. We first set up these experiments to acquire appropriate values of parameters used for task mining. The other parameters were fixed as follows: $(\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2) = (1.0, 1.0, 1.0, 1.0, 1.0, 1.0)$, $\theta = 0.5$.

Parameter Tuning for FI Task Mining To determine the value of $TransactionTime$ and $MinSuppCnt$, we only used FI tasks, and experiments were performed in the combinations of $TransactionTime = \{900, 1800, 3600, 5400, 7200\}$ [s] and $MinSuppCnt = \{2, 3\}$.

The averages of the F-measure are depicted in Fig. 4. The $MinSuppCnt = 2$ case performed better than the $MinSuppCnt = 3$ case, with the value of $TransactionTime$ held constant. Increasing $MinSuppCnt$ caused the number of files belonging to tasks to decrease; this indicates that related files that do not include keywords cannot be found by either tasks or intertask relationships. In addition, we notice a trend of the F-measure decreasing when the

Table 2. Average F-measure (FI Task Mining)

TransactionTime [s]	900	1800	3600	5400	7200
MinSuppCnt=2	0.543	0.513	0.507	0.470	0.506
MinSuppCnt=3	0.452	0.432	0.450	0.411	0.453

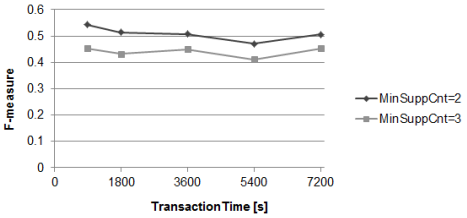

Fig. 4. Experimental Results for FI Task Mining

$TransactionTime$ increases. This is because, as $TransactionTime$ increases, more related files are put into the same transaction, and the number of files in a set of transactions will be less than $MinSuppCnt$ because the number of accesses is only counted once during the $TransactionTime$-long interval. We omit the details of the experimental results for each dataset because of space limitations. Analysis of the results indicates that $TransactionTime = 3600$ accomplished the highest F-measure on datasets A, B and E, while $TransactionTime = 900$ showed the best performance on datasets C, D, F and G. Differences in work patterns can be inferred from these results. Based on the results, we used a different $TransactionTime$ in subsequent experiments.

Parameter Tuning for RMC Task Mining $RMCTaskTime$ is the parameter used in extracting RMC tasks. To determine the value, in addition to FI tasks, we use the RMC tasks extracted in each case of $RMCTaskTime = \{60, 180, 300, 600, 1800\}$ [s] to perform file search.

As shown in Table 3, the average recall increased as $RMCTaskTime$ increased. However, the average precision decreased slightly, because a long $RMCTaskTime$ potentially brings unrelated files into the same RMC tasks. As the RMC operations were not handled very frequently, the impact was small and caused little change in the F-measure. In subsequent experiments, $RMCTaskTime$ was set to 60 s.

4.4 Evaluation of Intertask Relationships

Relationships between tasks are derived from similarity links and RMC links. In this section, we conduct experiments to determine the appropriate values of

Table 3. Average F-measure (RMC Task Mining)

RMCTaskTime [s]	60	180	300	600	1800
Precision	0.776	0.758	0.762	0.762	0.756
Recall	0.669	0.674	0.673	0.675	0.684
F-measure	0.684	0.681	0.679	0.684	0.684

Table 4. Setups for Comparison of RMC Operations

	Rename		Move		Copy	
Setup	α_1	α_2	β_1	β_2	γ_1	γ_2
non-RMC	0	0	0	0	0	0
Rename	1	1	0	0	0	0
Move	0	0	1	1	0	0
Copy	0	0	0	0	1	1
RMC	1	1	1	1	1	1

parameters used in the formulas defined for calculating the weights of links and the degree of intertask relevance.

Experiments for RMC Links RMC links, an aspect used for determining intertask relevance, are weighted using Equation (6), which considers the type and direction between files, by introducing rmc_f. rmc_f is defined by Equation (2) and the six parameters (α_1, α_2, β_1, β_2, γ_1, γ_2). To compare RMC operations, we set these parameters according to the configuration given in Table 4. As the results depicted in Fig. 5 show, using the rename and move operations had little effect on expanding the results of traditional file search. One reason is that renamed and moved files were ignored because files that do not exist in the filesystem were outside the evaluation targets. Another reason is that only 2% of log entries are created from rename and move operations. In contrast with rename and move, the copy operation enhanced the value of recall and the F-measure of full-text search results. Hence the result which utilized RMC performed the best F-measure, we set $(\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2) = (1.0, 1.0, 1.0, 1.0, 1.0, 1.0)$ in subsequent experiments.

Experiment with θ In Equation (7), we use θ to adjust the relative emphasis on similarity links and RMC links when calculating the degree of intertask relevance. To investigate the proper value of θ, we average the 11-point average precision with $\theta = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

As shown in Fig. 6, better-ranking results were obtained when $\theta > 0.0$ than when $\theta = 0.0$, although the precise value of θ was not so important. Because

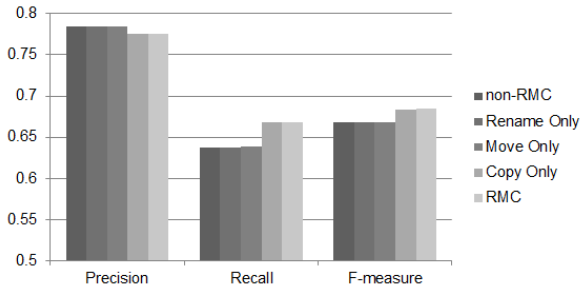

Fig. 5. Comparison of Experimental Results on RMC Links

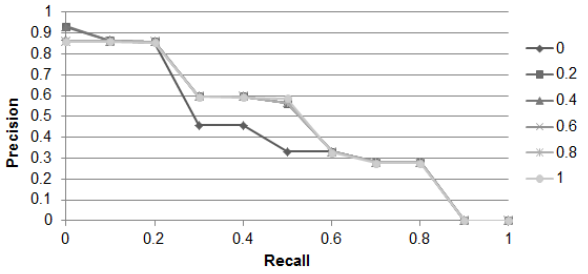

Fig. 6. Experimental Results for θ

tasks that have more links to other tasks tend to obtain a higher score under the proposed method, even if the degree of intertask relevance changes, tasks of this nature would be prioritized. $\theta = 0.0$ led to a poor performance because using RMC links only was not sufficient to expand the full-text results to other tasks. We also note that $\theta = \{0.2, 0.4\}$ got the highest 11-point average precisions (Table 5).

4.5 Evaluation of SUGOI

To improve the results of an existing keyword-based search engine, we group files related to the same tasks and derive intertask relationships from access frequency and RMC operations on files. We propose methods for mining FI tasks and RMC tasks. To expand the full-text search results, we use similarity links and RMC links to determine the relevance between tasks. To investigate the effect of task mining and the association links, we use the setup shown in Table 6 to compare SUGOI with an existing full-text search engine[4].

The experimental results are given in Table 7 and Fig.7. All configurations of the proposed system SUGOI are better according to the F-measure than the existing full-text search engine. FI tasks and similarity links increased the average of recall from 0.273 to 0.5, while precision increased from 0.795 to 0.820

Table 5. 11-point Average Precision

θ	0.0	0.2	0.4	0.6	0.8	1.0
Average of 11-point Average Precision	0.430	**0.482**	**0.482**	0.476	0.476	0.475

Table 6. Setup of SUGOI

Setup	Task Type	Intertask Relational Links
SUGOI 1	FI Task Only	Similarity Links Only
SUGOI 2	FI Task Only	Similarity Links + RMC Links
SUGOI 3	FI Task + RMC Task	Similarity Links Only
SUGOI 4	FI Task + RMC Task	Similarity Links + RMC Links

(SUGOI 1). In general, there is a trade-off between precision and recall. However, FI tasks consist of groups of files that are frequently accessed together and the average size of FI tasks is small, so few nonrelevant files are mingled in FI tasks.

In comparing SUGOI 1 with SUGOI 2 and SUGOI 3 with SUGOI 4, without RMC tasks, the effect of RMC links was small. An analysis of the mining results showed that only about 9% of log entries were RMC entries and most RMCed files were not contained in FI tasks because of low access frequency. Therefore, it was difficult to find new FI tasks using RMC links.

In addition to FI tasks and similarity, using RMC tasks and RMC links resulted in a certain improvement in recall and F-measure (SUGOI 3, 4). The reason is that files that do not include the keywords were found because they are related to tasks with files that do include the keywords. Thus, the precision average decreased slightly because RMC tasks are extracted from files RMCed together and, in contrast with FI tasks, extraneous files are easily mingled in RMC tasks.

From Table. 7, it is clear that SUGOI 4 achieved the highest recall and F-measure. The result confirmed that our proposed methods for task mining and deriving the intertask relationships are significantly effective for file search.

4.6 Summary of Experiments

In this section, we inspected the characteristics of the parameters and confirmed the validity of SUGOI by using actual file-access logs. The following conclusions were obtained.

1. A lower minimal support count ($MinSuppCnt = 2$) generated results with a higher F-measure. This means that if a combination of file accesses occurs more than twice in a set of transactions, these files are likely to have relevance to each other.

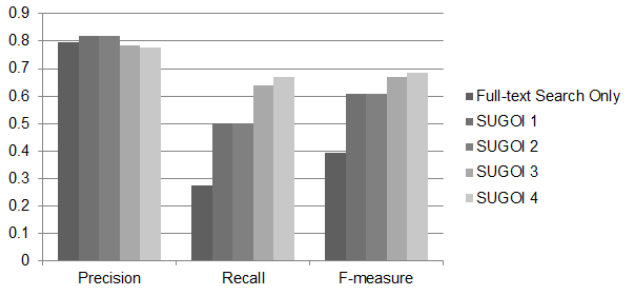

Fig. 7. Experimental Results of SUGOI

Table 7. Experimental Results of SUGOI

Setup	Precision	Recall	F-measure
Traditional Full-text Search	0.795	0.273	0.392
SUGOI 1	0.820	0.500	0.606
SUGOI 2	0.820	0.500	0.606
SUGOI 3	0.784	0.638	0.668
SUGOI 4	0.776	0.669	0.684

2. In the experiment on RMC task mining, we observed the trend that increasing $RMCTaskTime$ decreases the precision and increases the recall. However, the effect is small because users do not perform RMC operations frequently.
3. Copy operations, which can be done without making changes to the original file, were the most effective in RMC.
4. By investigating θ, a parameter used in Equation (7), we found that emphasizing the RMC links is effective, but the similarity links were also essential. SUGOI generated the best-ranking results when $\theta = \{0.2, 0.4\}$.
5. SUGOI conspicuously improves the average recall and F-measure over traditional full-text search results. The recall rise represented an increase of 0.396, while the increase in the F-measure was 0.292.

5 Conclusion and Future Work

As the volume of data stored in filesystems is increasing rapidly, and a large proportion of this is file-based unstructured data such as multimedia files, many files cannot be found using traditional full-text search engines. In this paper, we proposed a method for searching for such files by introducing the concept of tasks and intertask relationships derived from file-access logs. The main contributions of this paper are summarized as follows.

1. Task mining methods for two types of task: FI tasks, consisting of files frequently accessed together, and RMC tasks, consisting of files that were renamed/moved/copied (RMCed) simultaneously.
2. Methods for deriving association links from file-access logs by considering the similarity and RMC operations between tasks to generate a graph of intertask relationships.
3. A search method incorporating the task mining results into a full-text search to accomplish an accurate keyword-based file search.
4. Experimental results using actual file-access logs, which demonstrated that the proposed approach significantly improves search results.

As future work, we plan to conduct evaluation experiments using larger file-access logs. Contriving measures to choose the parameters automatically is also important for practical use. We also want to refine the proposed methods of task mining and the formulas for indicating intertask relevance.

Acknowledgments

This research was supported in part by a MEXT Grant-in-Aid for Scientific Research on Priority Areas (#201013017) and a JSPS Grant-in-Aid for Scientific Research (A) (#22240005).

References

1. Apple Inc.: Spotlight, http://www.apple.com/macosx/what-is-macosx/spotlight.html
2. Chen, J., Guo, H., Wu, W., Wang, W.: iMecho: an associative memory based desktop search system. In: CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management. pp. 731–740. ACM, New York, NY, USA (2009)
3. Google: Google Desktop, http://desktop.google.com
4. Hirabayashi, M.: Hyper Estraier, http://fallabs.com/hyperestraier/
5. Microsoft Corporation: Windows Search, http://www.microsoft.com/windows/products/winfamily/desktopsearch/
6. Daikoku Net: FAccLog, http://www2s.biglobe.ne.jp/~masa-nak/fal_down.htm
7. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Tech. rep., Stanford University (1998)
8. Soules, C.A.N., Ganger, G.R.: Connections: using context to enhance file search. SIGOPS Oper. Syst. Rev. 39(5), 119–132 (2005)
9. Watanabe, T., Kobayashi, T., Yokota, H.: A Method for Searching Keyword-Lacking Files Based on Interfile Relationships. In: OTM '08. pp. 14–15. Springer-Verlag, Berlin, Heidelberg (2008)
10. Watanabe, T., Kobayashi, T., Yokota, H.: Searching Keyword-lacking Files Based on Latent Interfile relationship. In: Software and Data Technologies. pp. 236–244 (2010)
11. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New Algorithms for Fast Discovery of Association Rules. In: KDD-97 Proceedings. pp. 283–286 (1997)