FileSearchCube: A File Grouping Tool Combining Multiple Types of Interfile-Relationships

Yousuke Watanabe[1], Kenichi Otagiri[2], and Haruo Yokota[1,2]

[1] Global Scientific Information and Computing Center
Tokyo Institute of Technology
2–12–1 Ookayama, Meguro-ku, Tokyo, 152-8552 Japan
watanabe@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp
[2] Graduate School of Information Science and Engineering
Tokyo Institute of Technology
otagiri@de.cs.titech.ac.jp

Abstract. Files in computers are increasing in number, so we require file management tools to find target files and to classify large groups of files. Our research group has been developing a system that provides virtual directories made up of related files. Many methods to extract inter-file relationships are available, such as word frequency, access co-occurrence, and so on. In practice, users need to select and combine multiple types of inter-file relationships to form groups of files they want. For this purpose, we propose FileSearchCube, a tool for file group analysis that introduces the data-cube concept. FileSearchCube provides cube operations for a set of files, and helps users to find relevant file groups.

1 Introduction

Files stored in computer systems are increasing in number [1]. We have various types of files, including plain text, RTF documents, images, videos and so on. Even if the files are organized by folder, it is difficult to find target files in a large set of files. To solve this problem, we often use desktop search [8, 11]. Most desktop search tools are keyword-based full-text search. They extract keywords from file paths, metadata of files (tags) and file contents. They also create an index for extracted keywords. When a user enters keywords, the tool returns a list of files associated with the keywords. Desktop search is useful when a target file includes common keywords.

Many other situations arise, however, in which desktop search tools cannot deliver satisfactory results. For example, when we need multiple files associated with different keywords, we have to choose multiple keywords and submit multiple search requests. Especially, keyword information for non-text files is limited (only file paths may be available); it is difficult for desktop search tools to find them. The difficulty arises because of limitations in keyword-based search techniques.

When we want to derive multiple target files, we would like to obtain, at one time, multiple files that have logical relationships. To achieve this, a good approach is to analyze interfile relationships and group related files. When such logical file relationships are discovered, desktop search tools can expand search results more easily. And when we have forgotten files used in previous work, we can find them again by traversing file relationships. Examples of such relationships include common keywords, modified at similar times, having similar file paths, concurrently accessed by the same user, and so on. Types of interfile relationships abound, and various scoring measures for them are proposed. Which measures should we use? Can we use them together? For more precise and flexible search requirements, we need to combine multiple measures of interfile relationships to create relevant file groups. Even if files are not appropriately separated by the keyword-based grouping method, we may do that by combining other measures, such as access co-occurrence. Although file grouping tools supporting a single type of interfile relationship already exist, there are no schemes to uniformly combine multiple measures.

We propose a file grouping scheme to combine multiple interfile relationships. Its model is based on the data cube [4], which is generally used in OLAP. We call it FileSearchCube. Our scheme supports functions for generating relevant file groups: choosing appropriate relationship measures, changing granularity of grouping, selecting groups based on multiple conditions. A dimension in FileSearchCube is a list of clusters constructed from one scoring measure of interfile relationship. FileSearchCube consists of multiple lists of clusters constructed from multiple measures. A user applies cube operations to the cube, and gets the sub-cube that the user wants. Cube operations are slice and roll-up/drill-down. Slice is an operation to cut the cube based on multiple conditions for dimensions. Roll-up/drill-down are operations to change the granularity of groupings for each dimension. Our scheme analyzes a large number of files more easily than previous systems. We have developed a prototype system and include an experimental evaluation in this paper.

The remainder of this paper is organized as follows: Section 2 describes related work. Section 3 introduces interfile relationships used in this paper. Section 4 presents the proposed file-grouping framework using the data cube. Section 5 describes our prototype system. Section 7 shows experiment results. And finally the Section 8 summarizes the paper and mentions future research issues.

2 Related Work

We first introduce search engines integrating keywords and other information. Google desktop search [8] provides a timeline view to display files that are modified and where in the search result they are modified. FRIDAL[7] integrates keyword search and access co-occurrences recorded in log data of file servers. Although this technique can find more files than conventional keyword search engines, the purpose is not to group files.

Semantic File System [3] provides a logical view based on metadata attached in file attributes. In this system, files belong to virtual directories defined by multiple conditions for a file's metadata. Windows Vista provides a search folders function. These systems use only file attributes. They do not consider interfile relationships, such as path similarity and access co-occurrences.

Faceted search [5, 6] explores objects with step-by-step selections. In each step, the system presents a set of selected objects and candidates of the object's attributes to give a condition for the next selection. Faceted search is similar to our scheme; it does not, however, consider interfile relationships.

Linking File Systems (LiFS)[2] can handle interfile relationships such as reference, copy/original and dependency. This system supports links to describe file relationships. Our scheme does not consider complex graph structures, but takes into account binary interfile relationships computed as numerical values.

3 Interfile Relationships

This section introduces similarity measures for interfile relationships. Of course, there are many similarity measures not described below. It is possible to introduce additional measures into FileSearchCube. We assume only that a measure returns a numeric value corresponding to an interfile relationship. In the remaining part, we assume that a similarity value between file x and file y is normalized in $[0, 1]$. The number 1 is the strongest relationship, and 0 is the weakest.

3.1 Text Similarity

Text similarity is used in document clustering. When many common keywords are contained in files, the files are strongly related to each other. Here, we assume a vector model. We extract keywords from file x and y; we then create word vectors v_x, v_y. Similarity between x and y is computed by a cosine value of two vectors. $Sim_{text}(x, y) = \frac{v_x \cdot v_y}{||v_x|| ||v_y||}$. For simplicity, we do not consider metadata (tag) information and file name to compute text similarity in this paper. We regard similarity between non-text files and other files as 0.

Although applicable only to text data, an advantage of text similarity is that the value is computed from file contents. Unlike text similarity, the three measures described below do not handle file contents. A disadvantage of this measure is that it is impossible to compute similarity for non-text files.

3.2 Modified Time Similarity

If two files are modified in close time instants, they may be processed together. This measure computes the similarity score based on the last time the two files were modified. Similarity score is computed by the following formula. $Sim_{mtime}(x, y) = 1 - \frac{|mtime(x) - mtime(y)|}{max_{a,b \in F}(|mtime(a) - mtime(b)|)}$, $mtime(f)$ expresses the last modified time of file f. The denominator in the formula is to normalize score ($\in [0, 1]$).

The advantage of modified time similarity is that it is applicable to all files. However, it is difficult to associate modified files with read-only files (their modified times are not changed).

3.3 Path Similarity

This measure regards files to be strongly related if they are located in the same folder or ancestor-descendant folders. In this paper, we use edit distance to compute path similarity. Edit distance is the distance between two strings. It is the number of operations (insert, delete, replace) to get the same string from one to another. Path similarity is computed by the following formula: $Sim_{path}(x,y) = 1 - \frac{editDistance(x,y)}{max_{a,b \in F}(editDistance(a,b))}$.

Path similarity works well when a user religiously organizes files and folders. On the other hand, when a user does not pay attention to file organization (e.g., many files are located in the same folder), this measure cannot distinguish between related files and irrelevant files.

3.4 Access Co-occurrence Similarity

When a user accesses two files at the same time, these files are processed together. If access co-occurrences are frequently observed, such files are strongly related. We can obtain access information from log data produced by file servers. [7] uses log data from a Samba server [10]. This log contains user id, file path, open/close time, and so on. In this paper, co-occurrence similarity is computed by total length of the co-occurrence time. $Sim_{cooccur}(x,y) = \frac{\Sigma t(x,y)}{max_{a,b \in F}(\Sigma t(a,b))}$, $t(x,y)$ expresses the set of co-occurrence times of files x and y.

Unlike modified time similarity, co-occurrence similarity can associate modified files with read-only files. Because of limited log data, however, it is difficult to derive precise similarity for files rarely used and files newly created.

As described above, various measures for interfile relationships have been proposed. They have advantages and disadvantages, and the effectiveness of these measures depends on user behavior and tendency. Thus, we have to choose and combine appropriate measures case by case.

4 FileSearchCube

FileSearchCube generates file groups by integrating multiple measures of interfile relationships (Figure 1). Since its data model is based on the data cube, it consists of dimensions and hierarchies. And it supports cube operations such as slice/dice and roll-up/drill-down. Although the conventional data cube targets the presentation of aggregation results rather than groups of items, FileSearchCube aims to discover logical groups of files.

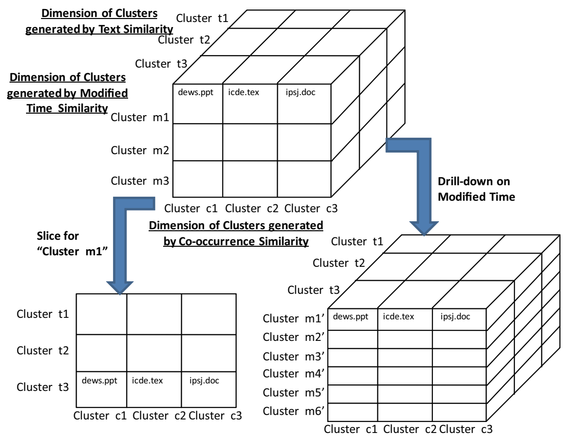

Dimension of Clusters generated by Text Similarity
Cluster t1
Cluster t2
Cluster t3
Dimension of Clusters generated by Modified Time Similarity
dews.ppt
icde.tex
ipsj.doc
Cluster m1
Cluster m2
Cluster m3
Cluster c1 Cluster c2 Cluster c3
Dimension of Clusters generated by Co-occurrence Similarity
Drill-down on Modified Time
Slice for "Cluster m1"
Cluster t1
Cluster t2
Cluster t3
Cluster m1'
Cluster m2'
Cluster m3'
Cluster m4'
Cluster m5'
Cluster m6'

Fig. 1. FileSearchCube

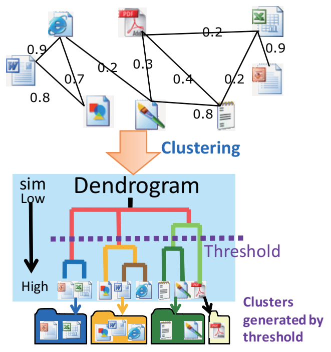


Fig. 2. Dendrogram generated by hierarchical clustering

4.1 Dimension

In FileSearchCube, a dimension is a list of file clusters generated by one similarity measure. Here, we use similarity measures introduced in Section 3. Conventional clustering is used in the tool. We summarize a clustering method below. Figure 1 presents a cube consisting of three dimensions. For instance, a file *dews.ppt* belongs to a cluster $t3$ generated by text similarity, $m1$ by modified time similarity, and $c1$ by co-occurrence similarity. We can use auxiliary dimensions generated by file information such as file name (dictionary order), file size, and file types (file extensions). A user groups files by combining clustering results from multiple measures and file information.

Clustering Method: By applying the formulas shown in Section 3 to N files, we can get an $N \times N$ similarity matrix. The clustering method receives this matrix as input. In this paper, we use conventional hierarchical clustering [4]. The method first makes a set of minimum clusters composed of each element; it then merges clusters that have high similarity values. It repeats merging, and finally produces one large cluster concatenating all clusters. A process in which the clustering method merges clusters is expressed as a dendrogram. Figure 2 is a dendrogram, which is the clustering result of 8 files. Clusters merged near the root of the dendrogram are regarded as having weak relationships. We can cut a dendrogram based on a threshold value and divide it into subtrees. Each pair of clusters within a subtree has larger similarities than the threshold value. A threshold value controls granularity of clusters. When a threshold value is near the root of the dendrogram, rough clusters are generated. When a threshold value is near the leaves of the dendrogram, fine clusters are generated. This property contributes to adjusting the granularity of groups in the data cube.

4.2 Hierarchy

In the conventional data cube, a concept hierarchy is used to change aggregation units in more abstracted/concreted layers. FileSearchCube also uses hierarchy information to group files. Hierarchies for dimensions corresponding to file attribute information are similar to those in the conventional method. They are defined in advance. For example, with the file extension type dimension, lower (concrete) layers are ".doc", ".txt", ".png", ".jpg" and higher (abstract) layers are "document","image". Hierarchies for dimensions corresponding to clustering results are substituted by dendrograms of clustering results. As described in Section 4.1, a dendrogram expresses the progress of merging clusters. Roll-up/drill-down cube operations change the granularity of clusters using dendrograms.

4.3 Cube Operations

Slice and Dice: Slice is a cube operation, which clips a sub-cube according to specified conditions. When conditions are defined on two or more dimensions, the operation is called dice. In this system, a user is also able to clip file groups by slicing the data cube. Beyond specifying a cluster label directly, conditions can also specify "cluster containing file A". Figure 1 is an example of slice. It clips a sub-cube with conditions that a label of modified time cluster equals $m1$.
Roll-up and Drill-down: Roll-up is an operation to regroup items by more abstract layers in the concept hierarchy. For example, roll-up changes monthly groups to yearly groups. On the other hand, drill-down is an operation to regroup items by more concrete layers. In FileSearchCube, dimensions of file attributes support conventional style roll-up/drill-down. Roll-up/drill-down for dimensions of clustering results are interpreted as an operation changing granularity of clusters. A threshold to cut a dendrogram controls granularity. When a dendrogram is cut near the root, the operation generates rough clusters. Figure 1 is an example of drill-down on a clustering result of modified time similarity.
Pivot: Pivot is an operation to change the order of dimensions and direction of axes. This is the same operation used in the conventional data cube.

5 Prototype System

A prototype system of our proposed scheme is written in Ruby (ActiveScriptRuby) and R. The architecture of the system is shown in Figure 3. The system consists of a similarity extractor, cluster generator, cube controller, and the Cube UI.

- **Similarity Extractor:** The similarity extractor obtains interfile relationships between files in a file system. The current implementation supports the text similarity, modified time similarity, path similarity and co-occurrence similarity introduced in Section 3. To compute text similarity, we use a similarity search facility provided by Hyperestraier [9], a full-text search engine. Hyperestraier has file convertor xdoc2txt, so beyond handling plain text, it

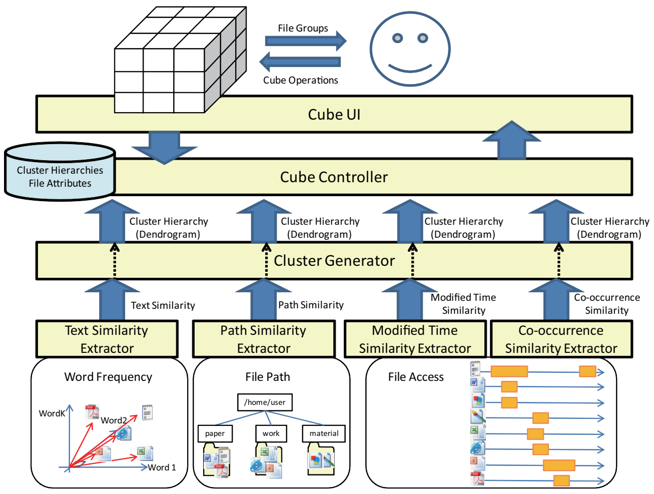


Fig. 3. System architecture

can also handle .doc, .xls, .ppt, .pdf and other common file types. Path similarity and modified time similarity are computed by scanning a file system directly. Access co-occurrences are extracted from log data recorded by a Samba file server [10]. Log data entries each contain the timestamp of file open/close, user name and file path.

– **Cluster Generator:** The cluster generator applies the hierarchical agglomerative clustering method for each similarity measure. It derives a dendrogram corresponding to the cluster merge process. This part is achieved using the hclust function in R.
– **Cube Controller:** The cube controller manages dendrograms produced by the cluster generator. It integrates clustering results and produces a multi dimensional cube. When it receives roll-up/drill-down requests from the Cube UI, it cuts dendrograms with a relevant threshold value.
– **Cube UI:** The Cube UI provides a cube-style interface and cube operations to users. The interface is based on Microsoft Excel. Since Excel has a named pivot table function similar to data cube, the Cube UI uses the Win32 OLE library and outputs Excel files that include cube data. Excel does not support Roll-up/drill-down operations, so they are performed in a separate window.

6 Examples

We show examples of the file grouping process using FileSearchCube (Figure 4). The first example is a requirement that “drill down text similarity composes groups by combining access co-occurrence similarity.”

1. In Figure 4, Snapshot1 shows 10 clusters generated by the clustering method based on text similarity. Clusters of other measures are not divided (only one cluster in each dimension). A cluster text06 consists of seven files that have common keywords.
2. Snapshot2 presents a cube divided into 10 clusters on co-occurrence similarity dimension. Cluster text06 is divided into two small clusters.
3. A user who wants to divide clusters more precisely performs a drill-down operation. Snapshot3 shows a cube divided into 20 clusters on co-occurrence similarity dimension. Cluster text06 is now divided into three groups (3 files, 3 files and 2 files). Files in each group are strongly related.

The second requirement is "find files modified in the same period in which files in the group of the first requirement are modified."

1. Snapshot4 shows the same cube in Snapshot 3. Here, we suppose base files belong to both cluster text06 and cluster cooccur02. The requirement wants to get files modified in the same period in which base files are modified.
2. To get the target group, drill-down operation on the dimension of modified time similarity is required. Snapshot5 is a cube divided into 10 clusters on the dimension of modified time similarity. Since the base files belong to cluster mtime01, we have to search target files in mtime01.
3. Snapshot6 represents a cube sliced by a condition mtime01 on the modified time dimension. Each cell of the cube includes files modified in the same period (mtime01).

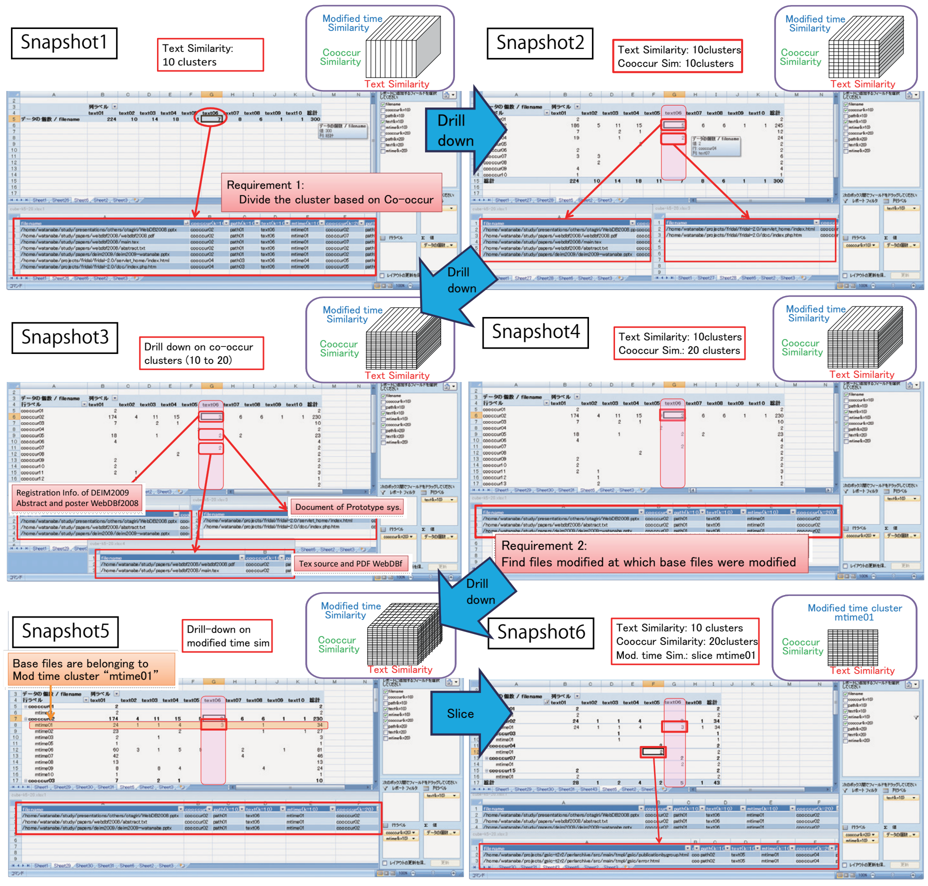
7 Experimental Evaluation

To evaluate effectiveness of FileSearchCube, we experimented using our prototype system. As experiment data, we prepared 300 files from an author's home directory. We chose files with the file extensions shown in 1. Full-text search engine Hyperestraier can extract keywords from 200 files (the remaining 100 files are images or binary data). The Samba log (Aug. 7 2008 - Dec. 24 2008) contains 834 access co-occurrences with 135 files.

To build an answer set, we manually labeled 300 files. Table 2 shows 39 labels for 300 files. The label "Research paper" contains tex source and presentation files. "Event" contains calls for workshop papers and member lists. "Software development" contains manuals and text-based configuration files, and icon images. "Office document" contains application forms for business trips; some are created from the same template.

7.1 Number of Generated Groups

The number of groups generated by one or more measures is shown in Table 3. The first column represents the combination of measures. The second column is the number of groups. The number of clusters generated in each dimension



Snapshot1
Text Similarity: 10 clusters
Modified time Similarity
Coocur Similarity
Text Similarity
Requirement 1: Divide the cluster based on Co-occur
Drill down
Snapshot2
Text Similarity: 10clusters
Coocur Sim: 10clusters
Drill down
Snapshot3
Drill down on co-occur clusters (10 to 20)
Registration Info. of DEIM2009 Abstract and poster WebDBf2008
Document of Prototype sys.
Tex source and PDF WebDBf
Snapshot4
Text Similarity: 10clusters
Coocur Sim.: 20 clusters
Requirement 2: Find files modified at which base files were modified
Drill down
Snapshot5
Drill-down on modified time sim
Base files are belonging to Mod time cluster "mtime01"
Slice
Snapshot6
Text Similarity: 10 clusters
Coocur Similarity: 20clusters
Mod. time Sim.: slice mtime01
Modified time cluster mtime01

Fig. 4. Examples of file grouping

(single measure) is fixed at 20. Files not extracted by keywords and files not recorded by any co-occurrence are grouped into the unknown group. The third column is the average number of files in one group. The fourth column is the average/maximum entropy among all groups. When a group consists of files with the same label, then the entropy of the group becomes 0. We first explain the differences in groups generated by each measure, then show combinations of two or more measures.

Single Measure

– **Text:** Text similarity generates clusters that have the highest value of average entropy among four measures. This means they are mixtures of various labels. There are interesting clusters: a cluster integrating Research paper 1 and 2 and a cluster integrating Equipment management 1 and 2. Office

Table 1. Extensions of experiment files

.JPG .PNG .ai .bib .css .csv .doc .docm .docx .eml .eps .htm .html .ico .jpg .pdf .png .ppt .pptx .rtf .tex .txt .xls .xlsx

Table 2. Answer set

Category of labels	**#files in the sets**
Research paper 1–2	1, 29
Event 1–3	3, 8, 13
Software development 1–5	16, 24, 5, 49, 54
Meeting 1–4	6, 9, 4, 3
Equipment management 1–4	4, 3, 9, 6
Paper review 1–7	2, 2, 1, 2, 3, 3, 3
Office document 1–12	3, 4, 2, 2, 5, 3, 4, 1, 1, 1, 2, 3
etc 1–2	5, 2

Table 3. Experiment result (20 clusters in each dimension)

Measure	#groups	Average #file	Entropy(avg/max)
Sim_{text}	20	15.0	0.369 / 1.105
Sim_{mtime}	20	15.0	0.382 / 0.920
Sim_{path}	20	15.0	0.174 / 1.291
$Sim_{cooccur}$	20	15.0	0.064 / 1.270
$Sim_{text} \times Sim_{cooccur}$	47	6.38	0.137 / 1.080
$Sim_{mtime} \times Sim_{cooccur}$	48	6.25	0.112 / 1.270
$Sim_{text} \times Sim_{path}$	57	5.26	0.112 / 1.045
$Sim_{mtime} \times Sim_{path}$	63	4.76	0.092 / 0.700
$Sim_{text} \times Sim_{mtime}$	75	4.00	0.091 / 0.678
$Sim_{path} \times Sim_{cooccur}$	42	7.14	0.080 / 1.302
$Sim_{text} \times Sim_{path} \times Sim_{cooccur}$	83	3.61	0.065 / 1.073
$Sim_{mtime} \times Sim_{path} \times Sim_{cooccur}$	84	3.57	0.061 / 0.728
$Sim_{text} \times Sim_{mtime} \times Sim_{cooccur}$	100	3.00	0.053 / 0.678
$Sim_{text} \times Sim_{mtime} \times Sim_{path}$	107	2.80	0.036 / 0.602
$Sim_{text} \times Sim_{mtime} \times Sim_{path} \times Sim_{cooccur}$	126	2.38	0.022 / 0.602

documents created by the same template are grouped into one cluster. A copied file and its original file become the same cluster.

– **Modified time:** Modified time similarity generates clusters classified roughly, but they are still mixtures of various labels. We think the number of clusters for this measure is too small (#cluster=20).
– **Path:** Path similarity generates clusters reflecting folder hierarchy in a physical file system. Files located in brother folders are not divided, even if we change the granularity of clustering. The author tends to put office documents in similar locations, so they are not divided.
– **Co-occurrence:** Co-occurrence similarity generates clusters with low entropy. For "Software development 1," it generates a cluster including manual and configuration files. For "Research paper 2," it generates a cluster that includes tex source and PDF files. For "Paper review 6," a paper to review and present comments for the paper are grouped into the same cluster. There are, however, many files in which there are no co-occurrences in log data; they are included in the unknown group.

Combination of Multiple Measures
As shown in Table 3, all combinations of two measures can generate smaller groups than a single measure. Single cases contain files not classified by text similarity and co-occurrence similarity. Combinations of multiple measures, however,

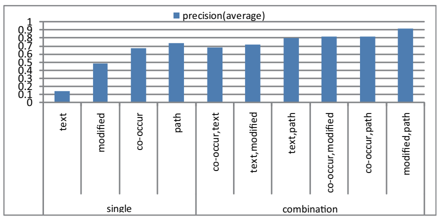


Fig. 5. Precision of single measure and combinations in type 1

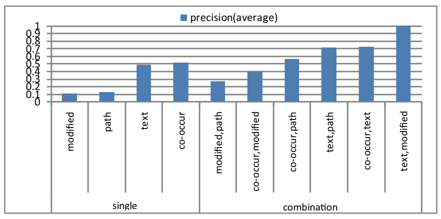


Fig. 6. Precision of single measure and combinations in type 2

can classify them into relevant groups, because path and modified time similarity cover all files, unlike text and co-occurrence similarity. Groups generated by a combination of co-occurrence and path similarities have the lowest average entropy among all combinations of two measures. However, this combination marks the highest maximum entropy. This implies that generated groups are skewed. Combinations of three or more measures generated more small groups. Some combinations generated over 100 groups. However, we have only 39 labels in the data set. We need a larger data set to evaluate 100 groups correctly. Evaluation with large data sets is a future research issue.

7.2 Effectiveness of Combining Multiple Measures

We consider two scenarios to evaluate the efficiency of our scheme.
Type 1: *"Group files used in the same work."*
We often access files in different folders during the same work. It is hard, however, to remember all files used in the work. This type of requirement is typical in file search. For evaluation, we use a set of files labeled "Software development1" (16 files). Figure 5 shows the result of this experiment. The vertical axis represents average precision of generated groups including files labeled "Software development1." If at least one common file exists in both a cluster C and an answer set A, we calculate precision of C by the following formula: $Precision_A(C) = \frac{|A \cap C|}{|C|}$
In single cases, the measure of path similarity is the best among four measures. Text similarity is the worst, because text information is not useful to determine which files were used in the same work. In combinations of two measures, combining path and modified time similarity is the best. And we can see that all combinations of multiple measures mark higher precision than when they are used separately.
Type 2: *"Group files created from the same template."*
When we write a new document for a paper procedure, we want to search old documents written for the same procedure. As a test data set, we used files labeled Paper review 5, 6, and 7 in Table 2 (9 files). They include reviewer's comments of the same journal, so these files are text data written in the same format. The name of the journal may be a common keyword. Results are shown in

Figure 6. In a single case, co-occurrence similarity is the best. In combinations of two measures, however, combining text and modified time similarity is the best. Since content information is important in this type of requirement, combinations that include text similarity improve.

This experiment shows the effectiveness of combining multiple measures. We think the best combination may change according to a user's tendencies and requirements. Without FileSearchCube, users must manually check which combination is the best. FileSearchCube can simplify the task.

8 Conclusion

This paper proposes FileSearchCube, which provides a multidimensional cube to analyze a large number of files. It generates file groups using multiple measures of interfile relationships. To achieve roll-up/drill-down operations, FileSearchCube uses results of the hierarchical clustering method. The generated groups are useful in summarizing logical structure of file systems and expanding the results of desktop search. We implemented a prototype system and evaluated it with experiments.

Several future research issues remain. The first is to integrate more similarity measures, such as image similarity. FileSearchCube can treat a measure that returns a numeric value corresponding to an interfile relationship. The second is detailed experiments for more users and more files. We will also evaluate the Cube UI with the focus on improved usability.

References

1. N. Agrawal, W. J. Bolosky, J. R. Douceur and J. R. Lorch. "A Five-Year Study of File-System Metadata," ACM Trans. Storage, Vol. 3, No. 3, p. 9, 2007.
2. A. Ames, C. Maltzahn, N. Bobb, E. L. Miller, S. A. Brandt, A. Neeman, A. Hiatt and D. Tuteja. "Richer File System Metadata Using Links and Attributes," Proc. IEEE MSST, pp. 49–60, 2005.
3. D.K.Gifford, P.Jouvelot, M.A.Sheldoon, J.W.O 'Toole,Jr. "Semantic File Systems," Proc. ACM SOSP, pp. 16–25, 1991.
4. J. Han and M. Kamber. "Data Mining: Concepts and Techniques" Morgan Kaufmann, 2006.
5. G. Smith, M. Czerwinski, B. Meyers, D. C. R. , G. G. Robertson and D. S. Tan. "FacetMap: A Scalable Search and Browse Visualization Visualization and Computer Graphics," IEEE TVCG, Vol. 12, No. 5, pp. 797–804, 2006.
6. K. Yee, K. Swearingen, K. Li and M. Hearst. "Faceted Metadata for Image Search and Browsing," Proc. CHI, pp. 401-408, 2003.
7. T. Watanabe, T. Kobayashi, H. Yokota. "A Method for Searching Keyword-lacking Files Based on Interfile Relationships" Proc. of CoopIS, Springer LNCS 5333, pp. 14-15, 2008. (poster)
8. Google Desktop. http://desktop.google.com/ja/features.html
9. Hyper Estraier. http://hyperestraier.sourceforge.net/index.ja.html
10. Samba, http://us3.samba.org/samba/
11. Windows Search. http://www.microsoft.com/windows/products/winfamily/desktopsearch/default.mspx