論文 / 著書情報
Article / Book Information

論題(和文)	頻出ファイル集合のアクセス時間を考慮した仮想ディレクトリ生成手法
Title(English)	Virtual Directory Creation Considering Access-times in Frequent Accessed File Sets
著者(和文)	小田切健一, 渡辺陽介, 横田治夫
Authors(English)	Kenichi Otagiri, Yousuke Watanabe, Haruo Yokota
出典(和文)	DEIM Forum 2010 F9-2, , ,
Citation(English)	DEIM Forum 2010 F9-2, , ,
発行日 / Pub. date	2010, 3

頻出ファイル集合のアクセス時間を考慮した仮想ディレクトリ生成手法 (O)

小田切健一† 渡辺 陽介†† 横田 治夫†,††

† 東京工業大学大学院情報理工学研究科計算工学専攻 〒 152-8552 東京都目黒区大岡山 2-12-1
†† 東京工業大学学術国際情報センター 〒 152-8552 東京都目黒区大岡山 2-12-1
E-mail: †{otagiri,watanabe}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし 近年、ストレージが進歩し、使用したファイル全てを保存しておくことが現実的になった。しかし、大量のファイルの中から過去の作業で用いたファイル群を発見するのは実際には困難である。特に大規模な作業はファイル数が多く複数ディレクトリにまたがっていることも多い。我々の研究グループではアクセスログの情報を利用し、複数ディレクトリに分散している同一作業で使用されたファイル群を発見し、仮想的なディレクトリとしてユーザに提示する研究を行っている。実際の一つの作業は、関係が密なファイル群が複数組み合わさって成り立っている。そのため、一つの作業全体のファイルを発見するには、近い時間に頻繁に使われるファイル群を発見するだけでなく、発見された複数のファイル群を一つに結合する必要がある。本稿では、発見されたファイル群に含まれるファイルの使用時間に着目する。アクセスログに対して頻出集合発見を行い、頻出するファイルの集合を発見した後、集合内の各ファイルの使用時間が重複している集合同士を結合することにより仮想ディレクトリの生成を行う。本稿では手法の詳しい解説と実データを用いた比較実験・考察を行う。

キーワード ファイル管理, アクセス履歴, アプリオリアルゴリズム, クラスタリング, 情報爆発, 仮想ディレクトリ, ファイル間関連度, 同一作業指数

Virtual Directory Creation Considering Access-times in Frequent Accessed File Sets

Kenichi OTAGIRI†, Yousuke WATANABE††, and Haruo YOKOTA†,††

† Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
E-mail: †{otagiri,watanabe}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

Abstract Due to the progress of storage technologies, it becomes easy to store all of used files. But it is difficult to discover the files that related to the specific task from a large amount of files. These files sometimes disperse over many directories. Our research group has been developing a system which rediscovers the files used in the same task and provides virtual directories including these files. A task includes some file groups whose files are used frequently and simultaneously. We make groups of files that are used frequently and simultaneously. Then combine file groups that belongs to the same task to create virtual directories. This paper proposes a new method that makes groups by apriori algorithm and combines file groups by overlaps of use-times of their files. We explain about the proposed method and experiment using real data.

Key words file management, access log, apriori algorithm, clustering, information explosion, virtual directory, relationship between file, same work index

1. はじめに

近年のストレージの増加により、データを全てストレージに保存しておくことが容易になった。保存したデータを再利用し、次の作業を効率的に行いたいという要求が高まっている。しかし、膨大なファイル群を適切に管理する手法が確立されておらず、本来再利用できるファイルを再利用できていない事が多い。過去の作業のデータを再利用するには、再利用に必要な全てのファイルを発見することが必要である。例えば過去に作成したプログラムを再利用しようとしても、そのプログラムに必要なデータファイルやドキュメントといったものも同時に発見できなければ十分に活用することが出来ない。そのため、過去の作業において使用したファイル群を発見できる手法が必要となる。デスクトップ検索 [1] [2] などが普及してきているが、キーワード検索ではテキストを含まないファイルを発見することが出来ない。また、同じ作業であっても同じキーワードを含んでいるとは限らないため、デスクトップ検索のキーワード検索だけで過去の作業のファイル群を発見することは困難である。そのため、ストレージに保存された大量のファイル群から過去の作業で使用したファイル群を発見する手法が必要となっている。

我々の研究グループでは、一つの作業でありながら複数のディレクトリに分散してしまったファイル群を発見し、そのファイル群を一つの仮想ディレクトリとしてユーザに提示する手法の研究を行っている [3] [4]。ユーザは仮想ディレクトリを見ることで分散してしまった同一作業のファイル群を発見することができる。同じ作業に用いられたファイル群の発見には、ファイルサーバのアクセスログのみを用いる。アクセスログのみを使用しファイルの中身を使用しないため、以下のような利点がある。

- テキストを含まないファイルでも発見できる。
- ファイルサーバを用いている多くの組織で必要最小限の変更で利用できる。
- クライアント PC に特殊な設定をする必要がない。
- ファイルの中身をみる必要がないので、パーミッション等の問題がない。

我々は今までの CO 法 [3] と FI 法 [4] という二つの手法を提案した。CO 法は Precision を重視した手法である。ファイルの使用時間が重複するファイルの組み合わせを発見し、その重複度合いをファイル間の関係の強さ（同一作業指数）とする。その情報を元に階層的クラスタリングを適用し、いくつかのファイルクラスタを得る。各々のクラスタを仮想ディレクトリとしてユーザに提示する。使用時間が確実に重複するファイルを発見するので、比較的誤ったファイルが出にくい。しかし実際には同じ作業で使用するファイルであっても使用時間が多少ずれていて重複がない場合が多い。

その問題に対処するために FI 法という手法を提案した。FI 法ではログファイルを一定期間ごとに区切り、各々の区間をトランザクションとする。同じトランザクションに入っていれば関係があるとすることで、使用時間が重複しないが使用時間が近いファイルを発見可能にする。しかし、無関係なファイルも多く入ってしまうので、頻出集合の発見を行って一定回数以上出現する組み合わせのみを抽出する。その後、一定以上重複がある頻出集合を結合することで仮想ディレクトリを生成する。これにより、FI 法は Recall を向上することに成功した。しかし、FI 法であっても全てのファイルを発見できるわけではなく、本来一つの仮想ディレクトリになるべきものが複数の仮想ディレクトリに分割されてしまう事がわかった。同じ仕事に属する頻出集合同士に全く要素の重複が無いことがあり、そういった頻出集合同士が孤立してしまうためである。

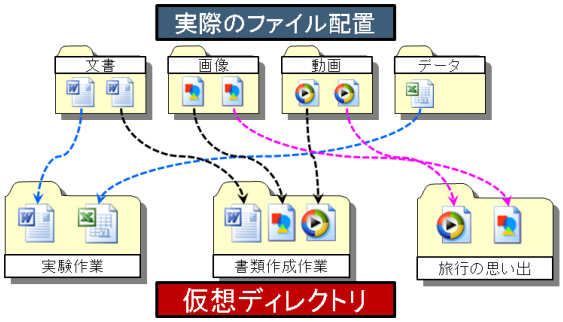

図 1　仮想ディレクトリ

本稿では、頻出集合内のファイルの使用時間に着目した仮想ディレクトリ生成手法を提案する。ログファイルを区切りトランザクションにし、頻出集合発見を行った後、頻出集合内の各ファイルの使用時間が重複している集合同士を結合をすることにより仮想ディレクトリを生成を行う。これにより、発見された頻出集合同士に要素としての重複がない場合でも、使用時間を元に同じ作業であるか判定できるようになる。この改良は Recall が優れている FI 法のクラスタリングステップに時間的情報を用いる CO 法の手法を取り入れたことになる。同じ作業であっても FI 法では孤立していた頻出集合が、この手法により結合されると考えられる。本論文では実データを用いて評価実験を行い、CO 法・FI 法との比較実験を通して提案手法の効果の考察を行う。

本論文の構成は以下の通りである。2 節ではこれまでの仮想ディレクトリ生成手法について解説する。3 節では今回提案する COFI 法について解説する。4 節では実データを用いた評価実験を行う。5 節では関連論文の紹介を行う。6 節ではまとめと今後の課題について述べる。

2. これまでの仮想ディレクトリ生成手法

本研究では複数のディレクトリに分散した同一作業のファイル群を発見し、それらを仮想ディレクトリとしてユーザに提示することを目的としている。例として図 1 のようにユーザが、文書・画像・動画・データの各々のディレクトリを作成してファイルを管理していたとする。ファイルは一見整理されているようだが、実際には実験作業・書類作成作業といった各作業は複数のディレクトリに分散している。数ヵ月後ユーザがファイルを再利用しようとするが、配置関係を忘れてしまっているのでディスク内を探索して複数のディレクトリを見つけなければならない。

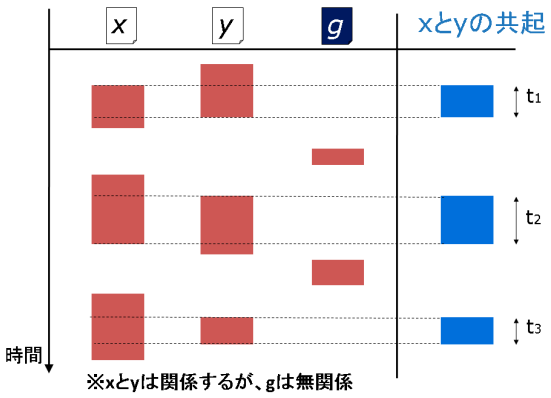

図 2　CO 法の要素間の関係発見

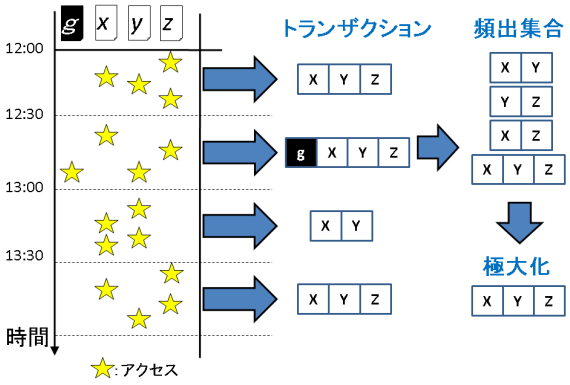

図 3　FI 法の要素間の関係発見

本研究ではこういったファイル群を発見し、図 1 の下部のように一作業を一ディレクトリとする仮想ディレクトリを生成する。どのファイルが同じ作業で用いられているかという情報は、ファイルサーバのアクセスログに記録された各ファイルのアクセス時間から推測を行う。ファイルのコンテンツではなくアクセス時間を用いるので、どのような種類のファイルであっても発見できるという利点がある。ファイルサーバのログファイルには OPEN/CLOSE の時刻が記録されているものと、アクセスのタイミングのみが記録されているものがある。例として、前者には Samba [5] のログ等があり、後者としては Windows Server 2003 の監査機能を用いて取得したアクセスログ等がある。CO 法で用いているファイル使用時間の推測では前者のログが必要だが、FI 法ではどちらのログでも使用可能である。

我々は、今までに二つの仮想ディレクトリ生成手法を提案してきた。それらの概要を解説する。

2.1　CO　　法

ファイル使用時間の重複に基づくクラスタリング法 (CO 法：Clustering using Overlap of file-use) [3] について解説する。「アクセスが重複するファイル同士は同一作業に用いられている可能性が高い」という考え方に基づいて提案した手法である。例として図 2 のようなファイル x,y,g があった場合、使用時間の重複に注目すると x と y は関係しているが g は無関係であると分かる。CO 法ではこの関係を元にファイルをノードとしてエッジの重みを重複時間とした重み付きネットワークを生成し、階層的クラスタリングを適用することで結びつきの強いファイル集合を発見し、仮想ディレクトリとして出力をする。

ファイル使用時間が比較的厳密に推測され、使用時間が本当に重複していたファイルのみ発見されるため理論的には Precision が高くなる。しかし、実際には同じ作業で用いられていても使用時間が重複しないファイルもあり、Recall の面で問題がある。

2.2　FI　　法

アプリオリアルゴリズムによる頻出ファイル集合発見法（FI 法：Frequent Itemsets discovery of file-use) [4] について解説をする。CO 法では、使用時間が重複するファイルの関係を発見していた。しかし、実際の作業では関連するファイルの使用が前後することがあり、近い時間に使用していても使用時間が重複しないケースがあった。そのため、CO 法では発見できるファイルが少ないという問題があった。FI 法はログファイルを一定時間ごとに区切り、その区間で使用があったファイルをトランザクションとする（図 3)。これにより、多少使用時間がずれたファイルも同じトランザクションに入り、使用時間が重複しないファイルも発見できる。しかし、これでは偶然近い時間に触ったファイルなどの無関係なファイルが入りやすい。そこで、トランザクションに対して頻出集合発見を行い、頻出する組み合わせのみを発見する。これにより、図 3 の偶然触った無関係なファイル g のようなものが除去される。さらに、頻出集合を極大化して、小さな集合を除去する。その後、集合内の要素が類似する頻出集合同士を結合して大きなファイル集合を生成し、それを仮想ディレクトリとして出力する。

ファイルの使用時間を荒く見ることで多少使用時間がずれたファイル同士も発見することができ、CO 法に対して Recall を向上させることができる。しかし、類似する頻出集合同士を結合する際要素の重複を見ているので、ある作業「ファイル A,B,C,D,E,F」が二つの頻出集合「A,B,C」「D,E,F」として発見された場合、この二つを結合することができないという問題がある。

3.　提案手法 COFI 法

今回新しく提案する COFI 法（Clustering using Overlap of file-use time for Frequent Itemsets）の提案動機と具体的な処理ステップについて解説する。

3.1　動　　機

同じ作業のファイルは同時に使用されると考え、ログファイルをトランザクション化し頻出集合発見を行ってみると、発見される頻出集合は作業の一部分であることが分かる。数十〜数百ものファイルを扱う作業において、ユーザが全てのファイルを同時に扱うことはほぼない。そのため、発見された頻出集合は作業の断片であると考え、頻出集合同士を組み合わせて一つの作業を復元する必要がある。例として、「ファイル A,B,C,D と「ファイル B,C,D,E」のような頻出集合が発見されれば、これらは内容がほぼ同一なので同じ作業ではないかと予測が出来る。FI 法ではこのように要素の重複がある頻出集合同士を結

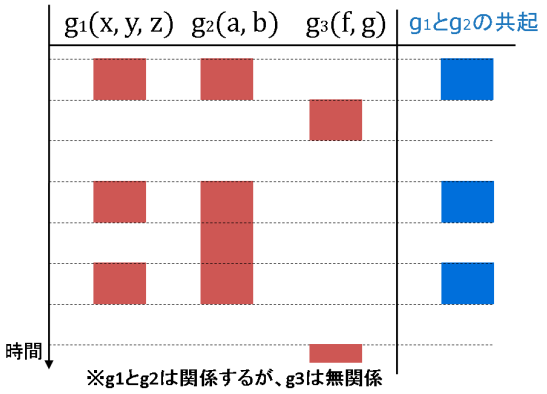

図 4　COFI 法の集合内ファイルの使用時間の重複図

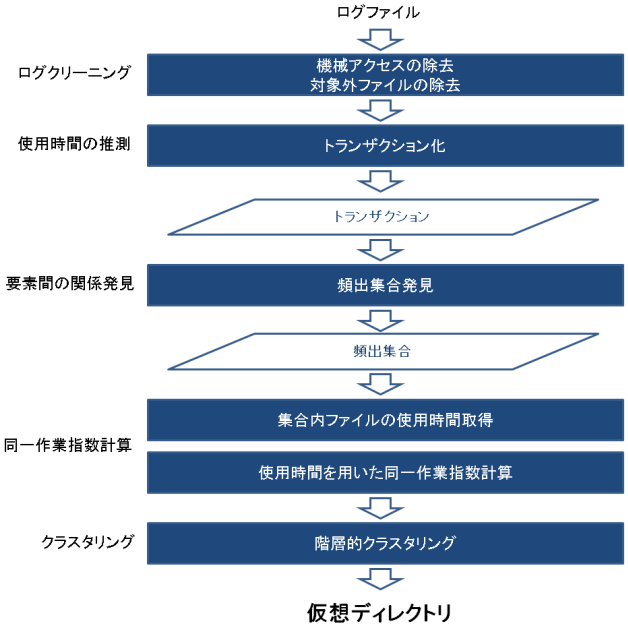

図 5　仮想ディレクトリ生成手順

合して仮想ディレクトリを生成していた。しかし、「ファイル a,b,x,y,z」という作業が「a,b」と「x,y,z」という二つの頻出集合として発見されてしまうと、要素を見ただけでは同じ作業であるか判別することが出来ない。a,b が図表作成作業、x,y,z が文書作成作業、のように小さな作業が組み合わさって一つの作業を形成している場合に起こってしまう。しかし、同じ作業に属しているので「a,b」と「x,y,z」が全く異なる時間に使われていたとは考えにくい。「a,b,x,y,z」全てが同時に使われた事がなくとも、図表を文書に取り入れる際などに「a と x」「b と z」といった組み合わせで同時に使われていたと考えられる。そこで、「a,b のいずれかを使用していた時間」「x,y,z のいずれかを使用して時間」に着目する。すると図 4 のように両方の使用していた時間が重複しており、これを用いて要素の重複がない「a,b」と「x,y,z」を統合することが出来る。このアイディアは CO 法のファイルの使用時間の重複を用いる考え方に基づいている。

3.2　処 理 手 順

各ステップの処理内容を表した図 5 に沿って解説する。我々の提案手法は「ログクリーニング」「ログデータからファイルの使用時間の推測」「ファイル要素間の関係発見」「同一作業指数計算」「クラスタリング」というステップからなっている。

3.2.1　ログクリーニング

ログファイルには人のアクセスのほかに、バックアップ・自動プレビュー・ファイル検索等のユーザが意図しないアクセスが記録されている。ユーザが意図しないアクセスが含まれると手法の適応時に正しい結果が得られない。そこで、この段階でユーザが意図しないと思われるファイルアクセスを除去する。今回は過去の報告 [4] で利用した一秒間に 4 ファイル以上の高頻度アクセスを機械アクセスとして除去する処理を行う。

また、本手法が対象としているのは、ユーザが作成・編集するファイルであり、実行ファイルや中間ファイルは対象としない。ここでフォルダ名やファイル名により不要なファイルを判別し、それらへのアクセスを除去する。

3.2.2　ファイル使用時間の推測

アクセスログに記録されている OPEN/CLOSE は、ファイルをすぐ閉じてしまうアプリケーション等の影響で必ずしもファイル使用時間に対応しない。また、厳密に使用時間を見て同時に使用したファイルを発見すると、極一部のファイルしか発見されない。前のファイルを閉じてから次のファイルを使うなど、ユーザがファイルを同時に使わない場合があるためである。

COFI 法ではログファイルをトランザクションという一定期間ごとに分割し、各トランザクションの中でアクセスがあったファイルを「そのトランザクションの中でずっと使用していた」と推測することにしている。これにより、多少使用時間がずれるファイルも同じ時間に使用していたと扱うことができる。一つのトランザクションの期間は TransactionTime というパラメータで決定する。これは非常に荒い使用時間の推測に対応する。例として、「ファイル A: 12:13:18, 12:29:09, 15:56:09」「ファイル B: 12:05:00, 12:19:00, 19:27:08」というアクセスを考える。ここで TransactionTime を 30 分とすると、この処理は次のように使用時間を推測することに対応する。「ファイル A: 12:00:00〜12:29:59, 15:30:00〜15:59:59」「ファイル B: 12:00:00〜12:29:59, 19:00:00〜19:29:59」

TransactionTime を大きくすることでより離れた時間で使用したファイルを発見可能になるが、より無関係ファイルも含みやすくなる。この処理は FI 法と共通で図 3 の左部分に対応する。

3.2.3　要素間の関係発見

トランザクション化によりファイル使用時間を推測すると、非常に多くのファイルが同一期間に使用されたことになる。使用時間がずれたファイルも発見できるが、偶然使用したファイルなども多く含まれてしまう。そこで、アプリオリアルゴリズム [6] による頻出集合の発見を行うことで、頻繁に同一期間内で使用されるファイルの組み合わせを発見する。ユーザが意図的に同時に使用するファイルと偶然同時に使用したファイルで

は、その頻度に大きな差があるため、適切なサポート値を設定することで偶然による組み合わせを除去できると考えられる。サポート値は MinSup というパラメータを用いる。このパラメータを大きくすると、頻繁に出現した組み合わせのみを発見するため精度が上がる。小さくすると偶然使用したファイルが発見されやすくなり、精度が下がる。実際には低頻度でしか使用しないファイルが多いため、あまり高い MinSup を設定すると頻出集合が出現しなくなってしまうということが分かっている。この処理は FI 法と共通で図 3 の右部分に対応する。

この処理により頻繁に近い時間に使用したファイル群を発見できる。このファイル群に偶然同時に使用したファイルは入りにくく、信頼性は高い。しかし、作業で使用したファイル全てを同時に触ることは少なく、ここで発見された集合は「書類 X 作成時に必要な図表を作成する際に使用したファイル群」「書類 X 本体」といったより小さな単位であることが多い。そこで、次のステップでこのような集合同士を結合し、ユーザにとって有益な「書類 X 作成」という作業を発見する。

3.2.4 頻出集合内のファイルの使用時間取得

FI 法では単純に要素の重複がある頻出集合同士を結合していた。しかし、COFI 法では頻出集合内のファイルの使用時間の重複がある集合同士を結合する。頻出集合内の各ファイルの使用時間を調べ、その和集合を頻出集合に付加する。実際には使用時間はトランザクションになっているので、頻出集合内の各ファイルが出現するトランザクション ID を調べ、その和集合を頻出集合に付与する。例として、頻出集合（A,B,C）があり各ファイルが出現したトランザクション ID を「A:1,2,6,7,9」「B:1,2,4,5,6,7」「C:2,3,7,8,9」とする。この時、頻出集合には「1,2,3,4,5,6,7,8,9」というトランザクション ID が付与される。以下、集合 A のトランザクション ID を $TranIDs(A)$ と表記する。

3.2.5 同一作業指数の計算

二つの頻出集合が同じ作業に属している可能性を「同一作業指数」と呼ぶ。その同一作業指数を COFI 法では頻出集合内のファイルの使用時間の類似度を用いて定義する。図 4 の g_1 と g_2 の共起に着目した計算になっている。g_1, g_2 を集合としたとき、dice 係数を用いて以下のように定義される。

$$\text{同一作業指数}\ (g_1, g_2) = dice(TranIDs(g_1), TranIDs(g_2))$$

$$= \frac{2|TranIDs(g_1) \cap TranIDs(g_2)|}{|TranIDs(g_1)| + |TranIDs(g_2)|}$$

これにより、使用時間が近いファイル群を持つ頻出集合同士が結合される。

FI 法では同一作業指数を頻出集合内の要素の重複度合いで定義していた。CO 法では同一作業指数をファイルの使用時間の重複度合いで定義していた。今回の COFI 法では、FI 法の頻出集合を使い、同一作業指数は CO 法的な使用時間の重複を用いている。

3.2.6 クラスタリング

本手法ではクラスタの数などがあらかじめ不明でもクラスタリングを行うことが出来る階層的クラスタリング [7] という手法を使う。前ステップで定義した同一作業指数が最も大きい頻出集合同士を結合することを複数回繰り返す。同一作業指数が最も大きい組み合わせの同一作業指数が Threshold 以下になったところで処理を終了する。Threshold を小さくすることで同一作業指数の小さな頻出集合同士も結合され、Recall が高くなる。Threshold を大きくすることで同一作業指数の大きな頻出集合同士のみが結合され、Precision が高くなる。

同一作業指数の定義は異なるが、CO 法や FI 法でも階層的クラスタリングの手順は同様である。

表 1　実験データ

ユーザ	期間	ログサイズ (byte)
ユーザ 1	2008/04/08〜2009/11/26	347,850,468
ユーザ 2	2008/08/07〜2009/11/25	1,110,478,322
ユーザ 3	2008/06/09〜2009/12/11	769,136,350
ユーザ 4	2008/06/10〜2009/10/23	1,895,819,901

表 2　正解セット

	ユーザ	内容	代表的拡張子	ファイル数
セット S_1	ユーザ 1	論文作成	pptx, png, tex	79
セット S_2	ユーザ 1	講義課題	docx, java, png, pptx	84
セット S_3	ユーザ 1	講義課題	java, pptx, tex	34
セット S_4	ユーザ 2	論文作成	eps, pdf, pptx	32
セット S_5	ユーザ 2	CGI 作成	html, pl, pm	33
セット S_6	ユーザ 2	発表資料	pdf, docx, txt	19
セット S_7	ユーザ 3	講義課題	pptx, java, txt	5
セット S_8	ユーザ 4	実験データ	pptx, wmf, xlsx	24

4. 実　　験

実際のログデータとユーザに作成してもらった正解セットを用いて、CO 法・FI 法・COFI 法の比較実験を行った。

4.1 実験データ

実験には Samba [5] のアクセスログを使用した。各ユーザのログデータの期間とサイズを表 1 に示す。Samba のログには OPEN/CLOSE が記録されており、CO 法・FI 法・COFI 法全てのファイル使用時間推測手法で利用可能である。

4.2 正解セット

本研究の目的は複数のディレクトリに分散したファイル群の発見である。そのため、正解セットとしても複数ディレクトリに分散したファイルを使用した。詳細は表 2 の通りである。正解セットには画像や数値データといったテキスト検索では発見できないファイル群も含まれている。

4.3 評 価 方 法

正解セット内の 2 ディレクトリ以上に属するファイル群を発見できた仮想ディレクトリの Precision・Recall・F 値を評価する。正解セット内のファイル群を含んだ仮想ディレクトリが存在しても、そのファイル群が全て元々一つのディレクトリに属している場合は評価しない。例として、プログラムディレクトリとレポートディレクトリが存在する正解セットの場合、プログラムのみを含んだ仮想ディレクトリは評価対象外で、プログラムとレポートの両方を含んだ仮想ディレクトリが評価対象

表 3　実験結果：ユーザ 1

手法	パラメータ	正解セット	Precision	Recall	F-measure
CO	0.7	S_1	0.368	0.177	0.239
		S_2	0.951	0.464	0.624
		S_3	1.000	0.088	0.162
		平均	0.779	0.243	0.342
FI	3600-2-0.1	S_1	0.263	0.607	0.378
		S_2	0.829	0.405	0.544
		S_3	0.389	0.411	0.383
		平均	0.484	0.475	0.432
COFI	3600-2-0.3	S_1	0.260	0.557	0.355
		S_2	0.486	0.429	0.456
		S_3	0.458	0.324	0.379
		平均	0.402	0.436	0.397

表 4　実験結果：ユーザ 2

手法	パラメータ	正解セット	Precision	Recall	F-measure
CO	0.6	S_4	0.141	0.656	0.232
		S_5	1.000	0.333	0.500
		S_6	0.081	0.632	0.143
		平均	0.407	0.540	0.292
FI	3600-2-0.3	S_4	0.941	0.500	0.653
		S_5	1.000	0.212	0.350
		S_6	1.000	0.263	0.417
		平均	0.980	0.325	0.473
COFI	3600-2-0.6	S_4	0.652	0.469	0.545
		S_5	0.813	0.394	0.531
		S_6	0.889	0.421	0.571
		平均	0.785	0.428	0.549

となる。複数の仮想ディレクトリにプログラムとレポートが含まれる場合は、F 値が最大の仮想ディレクトリを選択して評価する。

4.4　最適パラメータの探索方法

現在提案している手法ではユーザごとに最適パラメータが異なると考えられる。そこで、各手法の最適パラメータ探索はユーザごとに行った。各々のパラメータの組み合わせの中で平均 F 値が最大になるものを選択する。CO 法において探索したパラメータ範囲は以下の通りである。Threshold は階層的クラスタリング時のスレッショルドの値である。

$$Threshold = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$$

FI 法と COFI 法において探索したパラメータ範囲は以下の通りである。TransactionTime は 1 つのトランザクションの期間である。MinSup は頻出集合発見の際の最低サポートである。Threshold は階層的クラスタリング時のスレッショルドの値である。

$$TransactionTime = [1800, 3600, 7200][sec]$$

$$MinSup = [2, 3, 4]$$

$$Threshold = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$$

表 5　実験結果：ユーザ 3

手法	パラメータ	正解セット	Precision	Recall	F-measure
CO	0.3	S_7	0.600	0.600	0.600
FI	1800-3-0.1	S_7	1.00	0.600	0.750
COFI	1800-3-0.1	S_7	1.00	0.600	0.750

表 6　実験結果：ユーザ 4

手法	パラメータ	正解セット	Precision	Recall	F-measure
CO	0.4	S_8	0.571	0.167	0.259
FI	7200-3-0.4	S_8	0.625	0.208	0.313
COFI	7200-2-0.4	S_8	0.875	0.292	0.438

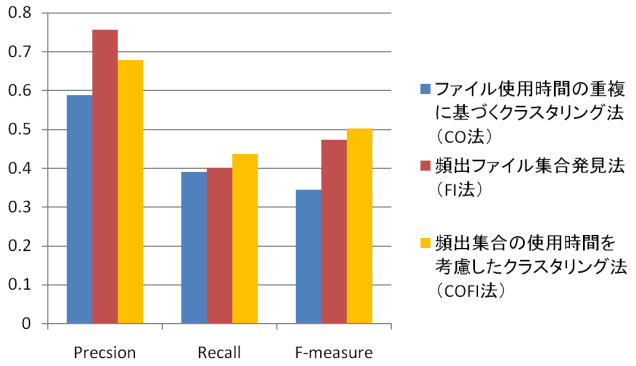

図 6　実験結果：全正解セット平均

4.5　実験結果

各ユーザの実験結果を表 3,4,5,6 に示す。ユーザ 1 とユーザ 2 は正解セットが複数あるため、表には正解セットごとの Precisoin,Recall,F 値と平均値を示した。CO 法のパラメータは、階層的クラスタリングの Threshold を表記している。FI 法と COFI 法のパラメータは、TransactionTime-MinSup-Threshold の順に表記している。さらに、全正解セットにおける平均 Precision, 平均 Recall, 平均 F 値の比較図を図 6 に示す。

4.6　考　　察

4.6.1　パラメータの考察

CO 法であってもユーザごとに最適なスレッショルドの値が異なることが分かる。また FI 法や COFI 法についても、ユーザごとにパラメータがかなり異なっている。これはユーザの作業スタイルやファイルの種類により最適なパラメータが異なる事を示している。

各ユーザ内で FI 法と COFI 法を比較するとかなり最適パラメータが似通っていることが分かる。ユーザ 4 を除き、特に TransactionTime と MinSup が同一である。FI 法と提案する COFI 法ではクラスタリングの処理が異なるが、それ以前の処理のパラメータである TransactionTime と MinSup は余り影響を受けないこと示している。

4.6.2　性 能 比 較

図 6 の全正解セットの平均に着目すると、今回提案した COFI 法が今までに提案した 2 手法に対して高い F 値を出していることが分かる。結論として、COFI 法は FI 法に対して Precision は少し落ちたものの狙い通り Recall を改善出来たといえる。

各ユーザに着目すると、FI 法や COFI 法に対して CO 法の

Recall が低いことが分かる。ユーザ 2 では Recall が高く見えるが、これは正解セット S_4 と S_6 のファイル群が同じ仮想ディレクトリに入ってしまうなど無関係なものが一つの仮想ディレクトリに入りすぎた結果である。そのため Recall が高いものの Precision が非常に悪くなっている。

各ユーザの FI 法と COFI 法を比較すると、ユーザ 1 では悪化し、ユーザ 2,4 では改善し、ユーザ 3 では変化がないことが分かる。ユーザ 3 の結果は正解セットが小さすぎたせいだと考えられる。ユーザ 2,4 では Recall が上昇しており、COFI 法の狙い通りに FI 法では分割されたままだった頻出集合が結合された結果だと考えられる。ユーザ 1 の悪化については、次のような事が考えられる。

ユーザ 1 の正解セットには講義課題が含まれている。同じ時期に複数の課題が集中するため、別々の講義課題が極めて近い時間に行われる。頻出集合発見時には低頻度の組み合わせが除去されるため、正しいファイル集合が見つかる。課題 A と課題 B の頻出集合同士は近い時間に行われていても、要素としては重複がない。従来の FI 法では要素の重複のある頻出集合同士が結合されるため、違う課題の頻出同士が結合される事は無い。しかし、COFI 法では使用時間の近さで結合するため、課題 A と課題 B に要素の重複がなくとも使用時間が重なっているため結合されてしまう。これにより Precision が下がってしまった。本来 Recall は FI 法より高かったのだが、高い Recall を出すときの Precision が悪いので最大 F 値で選択したところ FI 法より低い Recall が最適値になってしまっている。

このように、COFI 法は Recall の向上を実現できる場合もあるが、同時に複数の作業を行ったログでは悪化するということが言える。

5. 関連研究

ファイルを検索するシステムとしてはデスクトップ検索 [1] [2] が有名である。しかし、デスクトップ検索システムはファイルに含まれるテキストにするキーワード検索を用いるため、テキストが含まれないファイルは探すことができないという問題がある。また、同じ作業に属するファイルであってもソースコード・レポート・実験データ等で含まれるテキストは異なるため共通のキーワードで全種類のファイルを探し出すことは難しい。

デスクトップ検索システムの改良として提案された手法として渡部らの FRIDAL [8] がある。この手法ではキーワードでテキストを含むファイルを発見した後、テキスト検索で発見したファイル群に更にそれらと関連する別のファイル群も検索結果に含めて結果出力する。この関連ファイルの発見は、ファイルを同時に使用したという情報を使っている。これによりキーワードを含まないファイルであっても、キーワードで間接的に検索可能になる。CO 法はこのシステムの同時に使用したファイル群を発見する仕組みを利用している。

Soules らによる Connections [9] も、同様のデスクトップ検索の拡張である。テキスト検索の結果ファイル群と関連するファイル群も出力に含めることでキーワードを含まないファイルを検索可能にする。しかし、ファイル同士の関連を発見する方法は FRIDAL とは異なり、使用時間の厳密な重複ではなく「それらのファイルが N 秒以内にアクセスされたか」といった尺度を使用している。これは FI 法のトランザクションの考え方に近い。ただし、ある 2 ファイルが N 秒以内にアクセスされたか見ているだけで、FI 法のように 3 ファイル以上の組み合わせを発見しているわけではない。

また、井ノ口らの研究 [10] では、ファイルアクセスの順序やアクセスの密度に着目した尺度を提案している。これらの方式も仮想ディレクトリ生成に応用可能であると考えられる。

大澤らによる俺デスク [11] では、ブラウザや Word の動作を記録し、その操作履歴をタイムライン表示することでユーザが「Word で企画書を作成した時に参照していた Web ページ」といった物を検索できるようにしている。イベントの共起をタイムラインで表示するが、本研究のように自動で同時に使ったファイルを集約する機能がない点が異なる。

ファイル検索ではなくファイル整理という観点に立った研究として、Gifford らによる Semantic File Systems [12] がある。このシステムは各ファイルから属性を抽出し、その属性を用いて管理をするファイルシステムを提案している。しかし、属性を抽出するためにファイルタイプごとに属性抽出プログラムを書く必要があり、ファイル形式に依存しない本研究と異なる。

暦本は Time Machine Computing [13] を提案している。ファイルは全てデスクトップに置かれ時間とともに消えていくシステムを提案している。ファイルを探したい場合は、過去のデスクトップに遡ってそのときデスクトップに置かれていたファイル群と一緒に発見できる。時間の概念を利用している点では本研究と近いが、ユーザが明示的にファイルをデスクトップに置く必要がある点と、本研究のディレクトリのような明確な分類を行わない点が異なっている。

また、アクセスログのマイニングとしては、Web アクセスログを利用したリコメンデーション [14] などが関連する。これらの成果を利用しファイルのリコメンデーションができれば有用である。しかし、Web アクセスログでは性質も異なり、そのまま応用するのは難しいと考えられる。

6. まとめと今後の課題

本論文では、アクセスログを用いて複数ディレクトリに分散した同一作業に属するファイル群を発見する新しい手法を提案した。アクセスログを用いて近い時間に頻繁に使われるファイル集合を発見すると、それらは一つの作業内のよく使われる組み合わせに過ぎず、一つの作業全体のファイルを発見するためにはそのファイル集合を結合する必要があった。しかし、同じ作業に属していても発見されたファイル集合同士に要素として重複がないことがあり、これまでの要素の重複があるファイル集合同士を結合する手法だけでは不十分であった。そこで、本論文では発見されたファイル集合内の各ファイルの使用時間に着目してファイル集合同士を結合する仮想ディレクトリ生成手法を提案した。実データを用いた実験を通して、実際に Recall が向上することを確認した。しかし、同時に複数の作業を行うユーザでは結果が悪化することがわかり、この手法が有効な作

業タイプと有効でない作業タイプがあることが分かった。

今後の課題として、同時に複数の作業を行うユーザに対応できる手法の提案、より多様な正解セットによる評価、パラメータの自動設定方法、計算量・メモリ削減のための頻出集合発見の近似アルゴリズム採用などがあげられる。

謝　　辞

本研究の一部は文部科学省科学研究費補助金特定領域研究 (#21013017) の助成により行われた。

文　　献

[1] Google デスクトップ. http://desktop.google.co.jp/.

[2] Windows デスクトップサーチ. http://www.microsoft.com/japan/windows/desktopsearch/default.mspx.

[3] 小田切健一, 渡辺陽介, 横田治夫. アクセス履歴を用いたユーザの作業に対応する仮想ディレクトリの生成, 2009. 第 1 回データ工学と情報マネジメントに関するフォーラム（DEIM2009）.

[4] 小田切健一, 渡辺陽介, 横田治夫. ユーザ作業を反映する仮想ディレクトリ生成のためのアクセス履歴解析手法, 2009. 第 148 回データベースシステム・第 95 回 情報学基礎 合同研究発表会.

[5] *Samba*. http://us3.samba.org/samba/.

[6] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pp. 207–216, New York, NY, USA, 1993. ACM.

[7] 新納浩幸. R で学ぶクラスタ解析. オーム社, 2007.

[8] 渡部徹太郎, 小林隆志, 横田治夫. キーワード非含有ファイルを検索可能とするファイル間関連度を用いた検索手法の評価, 2008. 第 19 回データ工学ワークショップ (DEWS2008).

[9] Craig A. N. Soules and Gregory R. Ganger. Connections: using context to enhance file search. *SIGOPS Oper. Syst. Rev.*, Vol. 39, No. 5, pp. 119–132, 2005.

[10] 井ノ口伸人, 吉川正俊. アクセス履歴を考慮したファイル間の関連度を用いたデスクトップ検索 (履歴応用, 夏のデータベースワークショップ dbws 2006). 情報処理学会研究報告. データベース・システム研究会報告, Vol. 2006, No. 77, pp. 141–146, 20060712.

[11] 大澤亮, 高汐一紀, 徳田英幸. 俺デスク:ユーザ操作履歴に基づく情報想起支援ツール, 2006. 第 47 回プログラミング・シンポジウム報告集，pp.15-21.

[12] David K. Gifford, Pierre Jouvelot, Mark A. Sheldon, James W. O'Toole. *Semantic File Systems*, 1991. 13th ACM Symposium on Operating Systems Principles.

[13] Jun Rekimoto. Time-machine computing: a time-centric approach for the information environment. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology*, pp. 45–54, New York, NY, USA, 1999. ACM.

[14] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective perssonalization based on association rule discovery from web usage data. In *Proc. 3rd Intl. Workshop on Web information and data management*, 2001.