ファイル名抽象化に基づくアクセスログからのワークフロー抽出とファイル推薦手法の評価

宋　強† 川端　貴幸†† 伊藤　史朗†† 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
†† キヤノン（株）ソフトウェア応用技術開発センター　〒 146-8501 東京都大田区下丸子 3-30-2
††† 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: †{soukyou,watanabe}@de.cs.titech.ac.jp, ††{kawabata.takayuki,ito.fumiaki}@canon.co.jp, †††yokota@cs.titech.ac.jp

あらまし　情報の増加により，大量のファイル群から必要なものを探すコストが増大している．我々の研究グループでは，ファイルアクセスログを解析することで，頻繁に発生するファイル群へのアクセス系列をワークフローとして抽出し，利用者の必要なファイルをワークフローに基づいて推薦するシステムを研究・開発してきた．アクセスログ中には，同一の目的や手順で操作されたにも関わらず，個々のファイル名の一致しないアクセス系列が多く出力される．本稿では，そのような完全一致しないアクセス系列群を抽象化し，一つのワークフローとして抽出・活用するための手法を扱う．複数の抽象化方式と，抽象化されたワークフローを用いた推薦候補の選出方式を提案し，比較評価を行う．
キーワード　[ファイル推薦，抽象ファイル，抽象タスク，抽象ワークフロー，ログ解析]

Qiang SONG†, Takayuki KAWABATA††, Fumiaki ITOH††, 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
†† Canon Inc. Applied Software Technology Development Center, 3-30-2 Simomaruko, Ota-ku, Tokyo 146-8501, Japan
††† Global Scientific Information and Computing Center, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
E-mail: †{soukyou,watanabe}@de.cs.titech.ac.jp, ††{kawabata.takayuki,ito.fumiaki}@canon.co.jp, †††yokota@cs.titech.ac.jp

1. はじめに

近年，情報の増加により，ファイルシステムにおけるファイル数が劇的に増加している．大量のファイル群から必要なものを探すコストが増大している．調査によると，ユーザのオフィスにおける作業時間の約四分の一が「検索」という行為に使われている [1]．ユーザを文書の作成など価値ある知的創造性の作業に専念させるために，検索にかかる時間や労力を削減することが重要になる．

また，ユーザの探す対象は文書など情報源としてのファイルだけではない．仕事を達成するための手順などのノウハウとしての情報も探している．我々は，通常人手で経験的に整理するワークフローを，システムがユーザの操作履歴から価値あるワークフローを自動的にマイニングし，それを基にユーザに操作の推薦をすることで，ユーザをより知的創造性の作業に専念させられると考えている．

既存の関連文書の推薦の技術として，協調フィルタリング [2] が知られている．協調フィルタリングとは，ユーザの今後の行動を，過去の行動や嗜好が似ている他のユーザの情報をもとに予測する方法である．ファイル推薦システムでも協調フィルタリングが使える．しかし，協調フィルタリングでは操作の順番に厳しすぎる上，過去に行ったことのある同一のファイル操作作業しか推薦できない問題点がある．

そこで，我々の研究グループは，協調フィルタリングとは違い，ファイルアクセスログを解析することで，頻繁に発生するファイル群へのアクセス系列をワークフローとして抽出し，利

用者の必要なファイルをワークフローに基づいて推薦するシステムを研究・開発してきた [3]．アクセスログ中には，同一の目的や手順で操作されたにも関わらず，個々のファイル名の一致しないアクセス系列が多く出力される．本稿では，そのような完全一致しないアクセス系列群を抽象化し，一つのワークフローとして抽出・活用するための手法を扱う．複数の抽象化方式と，抽象化されたワークフローを用いた推薦候補の選出方式を提案し，比較評価を行う．

本論文の構成は以下の通りである．2. 節では，本研究が想定する課題と提案手法のアウトラインについて述べる．3. 節では，提案するファイル名によるファイル間類似度計算方法について説明する．4. 節では，抽象化されたワークフローを用いて推薦候補の選出方法について説明する．5. 節では，評価実験について述べる．6. 節では，関連研究を説明する．最後に 7. 節では本稿をまとめ，今後の課題を取り上げる．

2. 想定する課題と提案手法のアウトライン

本研究が想定する課題を図 1 を用いて説明する．ユーザ C が新規で「商品 Z 発注指示書」を作成した．ユーザ C の作業を支援するために，次に実行する可能性の高いファイル操作を予測して，ユーザ C に推薦することが想定する課題である．「商品 Z 発注指示書」は新規新規作成されたファイルのため，過去に利用された履歴は存在しない．本研究のアプローチとして，過去の類似する作業パターンをまず抽出して，それに基づいて推薦を行う．例えば，図 1 の過去にユーザ A とユーザ B が行った操作パターンにおいては，それぞれ違うファイルを異なる順番で操作しているが，同じ作業を行っていると言える．このような同じ作業を抽象パターンとして扱うために，操作シーケンスを抽象化する．我々は類似するファイルのグループ化処理として，ファイルの抽象化を行う．例では，三つの見積書ファイルにすべて同じテンプレートファイルから作成されているなどの共通性があれば，類似ファイルとして抽象化できる．同様に他のファイルについても抽象化を行う．次に，操作順番のわずかに異なる場合でも共通のパターンとして抽出するために，操作列をタスク単位で分割する．各タスク内のファイル同士の順番を無視する．これらにより，ユーザ A と B の操作履歴から「発注指示書」→「見積依頼記入例」,「商品情報」のような抽象作業パターンを抽出する．本研究では，抽出された一連の操作のシーケンスを頻出抽象ワークフローと呼ぶ．頻出抽象ワークフローの中の細かい操作シーケンス部分を抽象タスクと呼ぶ．ユーザ C が現在この抽象作業パターンをやっていると推定できたと仮定すると，次にユーザ C に「見積依頼記入例」か「商品情報」から適切なファイルを選択して推薦することが可能である．

ここで，提案手法の中の幾つかの用語の概念を具体的に説明する．

[**タスクとワークフロー**] ユーザごとに分割されたファイルアクセスログから，前後二記録の間のアクセス時間差（タスクのギャップタイム）が一定以上であれば，一つのタスクが終了し，新たなタスクが開始されたと判断し，このようなファイル単位

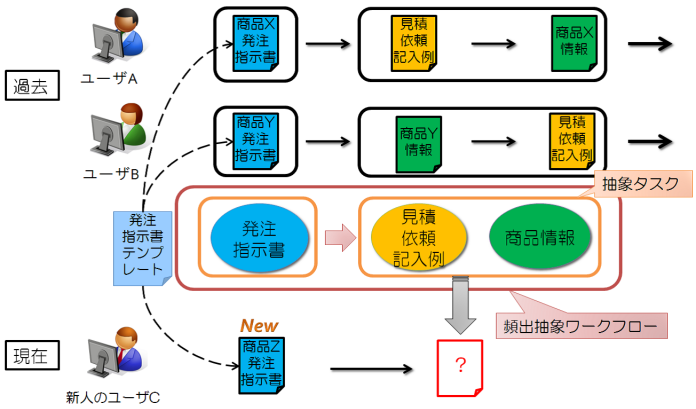

図 1　想定する課題

の作業シーケンスを**タスク**として切り出す．タスクはファイル名と操作のペアの集合（順序がない）である．また, 同様なような方法で，タスク間よりも長い一定時間差（ワークフローのギャップタイム）によって**ワークフロー**を切り出す．ワークフローはタスクのシーケンスである．

[**抽象タスクと抽象ワークフロー**] ファイル名の一部が異なっても，性質的に同じファイルをまとめるために，ファイルをグループ化する．例えば，図 1 では三つの商品発注指示書ファイルは性質的に同じため,「発注指示書」としてまとめる．そのような処理を行うために，ファイル間の類似度を定義する必要がある．ファイル同士の類似度を「コピー関係による類似度」と「ファイル名による類似度」の二つによって定義する. [3] ではファイル名による類似度の計算方法を一つしか提案しなかったが，本稿の 3. 節ではファイル名による類似度の複数の計算方法を提案し，評価する．

ファイル間類似度を使って，ファイルの抽象化を行う．具体的には，ファイル間類似度によって，ファイルをクラスタリングし，タスク内に出現するファイルをファイルクラスタに置き換える．更に，ファイル間の類似度を用いて，タスク間同士の類度度を計算し，類似度が高いタスク群をまとめて，タスクの抽象化を行う．**抽象タスク**とは，複数の抽象化されたファイル操作から成る意味のある纏まりである．**抽象ワークフロー**とは，抽象タスクのシーケンスである．

[**頻出抽象ワークフロー**] 抽象ワークフローの集合から，出現頻度が低いものを除くために，シーケンシャルパターンマイニング [4] で頻出するものを**頻出抽象ワークフロー**として抽出する．

[**推薦**] ユーザの現在の作業を監視し，ユーザがどの作業パターンを行っているかを推定する．具体的に，まずユーザが行っているタスクを過去のログから抽出した抽象タスクと比較して，類似する抽象タスクを探す．次に，推定した抽象タスクを使って，最も適切な頻出抽象ワークフローを推定する．詳細な推定方法については [3] を参照されたい．推定できた頻出抽象ワークフローに基づいて推薦する時に，ユーザが現在行って

表 1　ファイル名によるファイル間類似度の計算方法

比較方法/最小単位	キャラクタ単位で比較	品詞単位で比較
LCS を求める ことで比較	キャラクタ単位の LCS [3]	品詞単位の LCS
LD を求める ことで比較	キャラクタ単位の LD	品詞単位の LD
集合として比較	2-gram の集合	品詞の集合

いるタスクに類似する抽象タスクの属している頻出抽象ワークフローから，次に実行されそうな抽象タスクが特定できる．その抽象タスクに対応するファイルクラスタから適切なファイルを選んで推薦することになる．しかし，図 1 の「見積依頼記入例」や「商品情報」のようなファイルクラスタは複数のファイルを含むため，どのファイルを優先的にユーザに推薦するかという課題がある．我々このような次に実行されそうな抽象タスクのファイルクラスタからユーザに適する具体ファイルを選ぶ作業をファイルのランキングと呼ぶ. [3] ではファイルクラスタ内で過去に一番アクセスされた回数が多いファイルを優先的に推薦する方法と，一番最近に使われたファイルを推薦する方法の二つのランキング方法を提案した．本稿の 4. 節では，更に複数のランキング方法を提案し，比較評価を行う．

次の 3. 節では，提案するファイル名によるファイル間類似度計算方法について説明する．

3. ファイル名によるファイル間類似度の計算

本節では，ファイル名によるファイル間類似度の計算方法について述べる．表 1 で示している通り, [3] で提案した方法に加えて，新しく 5 通りの方法を提案する．本節では，この 6 通りのファイル名によるファイル間類似度計算方法についてそれぞれ説明する．

以降では，ファイルのフルネームを「ファイル名」と「拡張子」に分けて考える．ファイル名及び拡張子の両方とも類似するほど，お互いの類似度が高いとみなす．ファイル名は部分一致を考慮するが，拡張子は原則的に完全一致のみとする．例えば,「仕様書_version1.docx」と「仕様書_version2.docx」のようなファイル同士は，利用目的が似ていることが推定でき，類似度を高く与える．逆に，お互いの拡張子が異なる場合，ファイル同士の類似度を弱める．尚，ユーザの使用するソフトのバージョンの違いによって，同じ種類のファイルに違う拡張子がつけられる可能性があるため，例外として，*doc* と *docx*, *ppt* と *pptx*, *xls* と *xlsx* を同じものと見なす．

以下，ファイル名に基づくファイル間類似度の計算方法を説明する．

3.1 キャラクタ単位で比較

ファイル名同士を比較する時に，ファイル名の文字列を比較する最小単位として，キャラクタ単位と品詞単位の二つをここで扱う．先に，キャラクタ単位で比較する方法を述べる. 品詞単位で比較する方法は 3.2 節で説明する．

3.1.1 キャラクタ単位の LCS による方法

[3] では，それぞれのファイル名をキャラクタ単位のシーケンスに分割し，式 (1) で類似度を計算している．

$$filenameSimLCS_{character}(fileA_c, fileB_c) = \frac{len_c(LCS_c(fileA_c, fileB_c))}{min_c(len_c(fileA_c), len(fileB_c))} * d. \quad (1)$$

ここで，$len_c(fileX)$ は $fileX_c$ のファイル名のキャラクタ単位のシーケンス長，$min_c(len_c(fileA_c), len_c(fileB_c))$ は $fileA_c$ と $fileB_c$ の中の短い方のキャラクタ単位ファイル名シーケンス長，$LCS_c(fileA_c, fileB_c)$ は $fileA_c$ と $fileB_c$ の最長共通サブシーケンス (Longest Common Subsequence, 以下 LCS) [5] 長を表す. 変数 d は拡張子が異なる場合の減衰を表現するためのものである．もし拡張子が一致すれば，d=1 とし，そうでない場合は 1 未満の値となる（実験で我々は d=0.8 と設定した）. ここの d は以下の式 (2), 式 (3), 式 (4), 式 (5) と式 (6) の中の d と共通する．

3.1.2 キャラクタ単位の LD による方法

LCS を求める以外に，キャラクタのシーケンス同士の類似度合を測る方法として，編集距離 (Levenshtein Distance, 以下 LD) [6] がある．LD は，二つの文字列がどの程度異なっているかを示す数値である．具体的には，文字の挿入や削除，置換によって，一つの文字列を別の文字列に変形するのに必要な手順の最小回数として与えられる．ファイル名をキャラクタ単位のシーケンスとしてみる時に，LD を使う計算式を式 (2) で示す.

$$filenameSimLD_{character}(fileA_c, fileB_c) = (1 - \frac{LD_c(fileA_c, fileB_c)}{max_c(len_c(fileA_c), len_c(fileB_c))}) * d. \quad (2)$$

ここで，$LD_c(fileA_c, fileB_c)$ は $fileA_c$ と $fileB_c$ の間の編集距離，$max_c(len_c(fileA_c), len_c(fileB_c))$ は $fileA_c$ と $fileB_c$ の中の長い方のキャラクタ単位ファイル名シーケンス長を表す.

3.1.3 キャラクタ単位の 2-gram による方法

3.1.1 節と 3.1.2 節の方法では，ファイル名をキャラクタのシーケンスと見なすため，高い類似度スコアを得るには，シーケンス中の要素間の順番が一致する必要がある．順番が一致しないと，類似度が低くなる．若干順番が違っても構わない計算方法として，ファイル名に出現したキャラクタを集合として扱う方法が考えられる．しかし，1 文字ではノイズが多くなるため，2-gram の集合で計算する方法を提案する．2-gram とは，任意の文字列が 2 文字だけ続いた文字列のことである．例えば,「データベース」を 2-gram に分割すると,「デー」,「ータ」,「タベ」,「ベー」,「ース」の五個の 2-gram が得られる．我々はファイル名からの 2-gram の集合を求め，式 (3) でダイス係数 [7] で集合同士の類似度を計算する．

$$filenameSim_{2gramSet}(fileA_{2gram}, fileB_{2gram}) = \frac{2|fileA_{2gram} \cap fileB_{2gram}|}{|fileA_{2gram}| + |fileB_{2gram}|} * d. \quad (3)$$

X_{2gram} は $fileX$ のファイル名の中の 2-gram の集合を表す.

3.2 品詞単位で比較

ファイル名をキャラクタ単位で比較すると，正しく類似度が取れない場合がある．よって，ファイル名を意味のある品詞に分割してから，品詞単位のシーケンスとして比較する方法を追加する．先に，品詞の分け方について説明する．我々は日本語形態素解析システム Sen [8] でファイル名を品詞単位に分割して，ノイズの品詞を除去する．ここでは，ノイズの品詞は助詞，数字，未知語とする．これらの品詞はファイルを分類するために有用な意味を持たない．更に，名詞を「東京工業大学」や「キヤノン」のような専有名詞と「レポート」や「仕様書」のような一般名詞の二種類に分ける．ファイルを抽象化するために，ファイルを「レポートファイル群」や「発注書ファイル群」のような利用目的ごとに分類するには，専有名詞も有用ではないため除去する．一般名詞だけを利用して，ファイル名によるファイル間類似度計算に利用する．

3.2.1 品詞単位の LCS による方法

ファイル名を一般名詞単位のシーケンスと見なし，式 (4) でシーケンス同士から LCS を求めて，類似度を計算する．

$$filenameSimLCS_{word}(fileA_w, fileB_w) = \frac{len_w(LCS_w(fileA_w, fileB_w))}{min_w(len_w(fileA_w), len_w(fileB_w))} * d. \quad (4)$$

ここで，$len_w(fileX_w)$ は $fileX_w$ の品詞単位のファイル名シーケンス長，$min_w(len_w(fileA_w), len_w(fileB_w))$ は $fileA_w$ と $fileB_w$ の中の短い方の品詞単位ファイル名シーケンス長，$LCS_w(fileA_w, fileB_w)$ は $fileA_w$ と $fileB_w$ の LCS を表す．

3.2.2 品詞単位の LD による方法

ファイル名を品詞に分割してから，一般名詞だけを抽出する．ファイル名を一般名詞単位のシーケンスと見なし，式 (5) でシーケンス同士から LD を求めて，類似度を計算する．

$$filenameSimLD_{word}(fileA_w, fileB_w) = (1 - \frac{LD_w(fileA_w, fileB_w)}{max_w(len_w(fileA_w), len_w(fileB_w))}) * d. \quad (5)$$

ここで，$LD_w(fileA_w, fileB_w)$ が $fileA_w$ と $fileB_w$ 間の品詞単位の編集距離である．$max_w(len_w(fileA_w), len_w(fileB_w))$ が $fileA_w$ と $fileB_w$ の中の長い方の品詞単位ファイル名シーケンス長を表す．

3.2.3 品詞単位の集合による方法

3.2.1 節と 3.2.2 節の方法では，ファイル名を名詞のシーケンスと見なすため，高い類似度スコアを得るには，シーケンス中の要素間の順番が一致する必要がある．順番が一致しないと，類似度が低くなる．よって，同じファイル名に出た一般名詞間の順番を無視して，集合として扱う方法を追加する．式 (6) でダイス係数を利用して，集合同士を比較して，類似度を計算する．

$$fileNameSim_{wordSet}(fileA_{ws}, fileB_{ws}) = \frac{2|fileA_{ws} \cap fileB_{ws}|}{|fileA_{ws}| + |fileB_{ws}|} * d. \quad (6)$$

ここで，X_{ws} が $fileX$ のファイル名の中の一般名詞の集合を表す．

以上の 6 通りのファイル名によるファイル間類似度の計算方法の比較は 5.3 節で説明する．

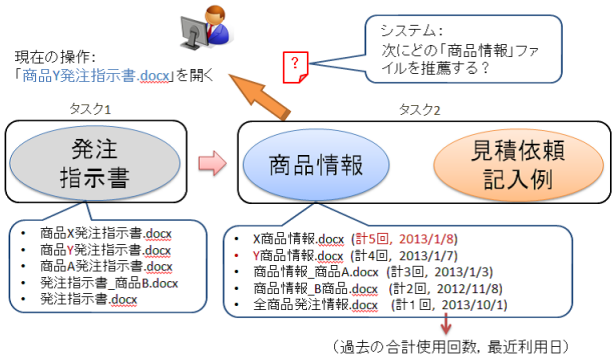

図 2 ファイルクラスタの具体ファイルのランキング

4. ファイルクラスタ内のファイルのランキング

2. 節で述べたように，推定できた頻出抽象ワークフローに基づいて推薦する時に，ユーザが現在行っているタスクに類似した抽象タスクの属している頻出抽象ワークフローから，次の抽象タスクが特定でき，抽象タスクのファイルクラスタ内にあるファイルを選んで推薦する．

例えば，図 2 で示すように，ユーザの一連の操作に基づいて，操作パターン「発注指示書ファイルクラスタ」→「商品情報ファイルクラスタ」→「見積依頼記入例ファイルクラスタ」を実行していると判定できたと仮定する．ユーザが現在「発注指示書ファイルクラスタ」の中の「商品 Y 発注指示書.docx」を操作しているとする．次に「商品情報ファイルクラスタ」から具体的なファイルをユーザに推薦することになるが，「商品情報ファイルクラスタ」は 5 個のファイルを要素として含むため，この 5 個のファイルから最も適切なファイルを優先的に推薦しないといけない．本節は推薦候補となるファイルクラスタから具体ファイルをランキングする方法について述べる．

4.1 利用頻度，使用時間に基づくランキング

我々の提案してきたシステム [3] では，二つのランキング方法を提案してきた．一つ目は，ファイルクラスタ内のファイル群において，過去に利用回数が一番多いファイルを優先的に推薦する方法である．二つ目は，ファイルクラスタ内のファイル群において，一番最近に使われたファイルを優先的に推薦する方法である．

しかし，これらの方法では適切でない場合が起こりうる．図 2 では，ファイルクラスタ「商品情報ファイルクラスタ」の中の 5 個のファイルの中で，「X 商品情報.docx」が過去における利用回数が一番多く，かつ一番最近に利用されたため，以前提案してきた二つの方法だと，優先的な推薦対象になってしまう．

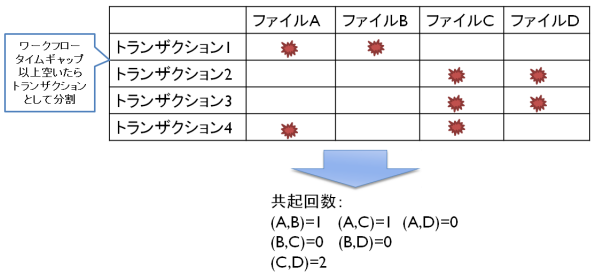

図 3　ファイル同士共起回数のカウント

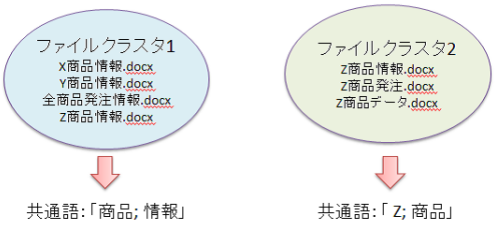

図 4　ファイルクラスタのラベルの例

しかし，今ユーザが操作しているファイルが「商品 Y 発注指示書.docx」に対しては，ユーザにとって本当に望ましいものが「X 商品情報.docx」でなく,「Y 商品情報.docx」が正解だと考えられる．この二つの方法は，次に推薦するファイルクラスタ中のファイルをランキングする時に，現在ユーザが操作しているファイルとの関係を反映させていないことが原因だと考えられる．そこで，ランキングする時に，ユーザが操作しているファイルとの共起関係やファイル名の関係を扱う二つの方法を 4.2 節と 4.3 節で導入する．

4.2　共起頻度順にランキング

過去によく一緒に利用したファイル同士には高い関連性があると考えられる．図 2 の例では,「商品 Y 発注指示書.docx」を編集する際には, 同商品の「Y 商品情報.docx」との共起（一緒にアクセスされる）が，別商品の「X 商品情報.docx」と共起よりも多く発生すると考えられる．共起回数を見ることによって,「X 商品情報.docx」ではなく,「Y 商品情報.docx」を優先的に推薦する．ここで，ユーザの操作しているファイルとの比較対象として，過去に共起したすべてのファイルではなく，頻出抽象ワークフローを推定した後に，次に操作する可能性の高いファイルクラスタの中のファイルだけを推薦候補とする．

共起回数の計算方法を図 3 を使って説明する．過去のログをワークフローを切り出す時に使ったパラメータでトランザクションに分ける．一つのトランザクションに出現したファイル同士の共起回数を 1 として数える．この図では，ファイル C とファイル D がトランザクション 2 と 3 に一緒にアクセスされたため，共起回数は 2 である．次にレコメンドするファイルクラスタが含むファイルをそれぞれユーザ直前に操作されたファイルとの共起回数を調べ，共起回数の高い順にランキングする．

この方法では，過去に使用履歴が多いファイルほど，有効となる可能性が高い．弱点としては，新規ファイルを作成した場合で，そのファイルは過去に使用履歴がないため，共起情報が存在しない．つまり，この方法は過去に使われたファイルにしか適応できない．

4.3　ファイル間類似度順にランキング

まず，単純にファイル名による類似度をそのまま用いて候補ファイルと直前に操作したファイルと類似度を求め，類似度の高い順に推薦する考え方がある．ここでも図 2 を用いて説明する．直前に操作されたファイルが「商品 Y 発注指示書.docx」に対して,「商品情報ファイルクラスタ」から 3. 節で述べた方法で，ファイル名類似度が高いファイルを探すと，一番類似するのが「全商品発注情報.docx」となり,「Y 商品情報」ではない．ここで，5 個の候補のファイル名を見ると,「商品」と「情報」の品詞が共通であることが分かる．こういう共通に出現する品詞も使ってファイル間類似度を計算すると,「Y 商品情報」の類似度が低くなり，優先的に推薦できない理由となる．

よって，我々はまず各ファイルクラスタ内のファイル名にでる品詞を統計し，一番頻出する品詞をそのファイルクラスタの共通語とする．もし，出現頻度が一番高い品詞が複数個存在するなら，それらを全部共通語にする．例えば，図 4 では，ファイルクラスタ 1 の共通語は「商品」と「情報」である．ファイルクラスタ 2 の共通語は「Z」,「商品」である．こういう共通語は多くの推薦候補ファイルに持たれるため，上で述べたように，類似度の計算上逆効果を及ぼす場合もあるため，ファイル名からクラスタ内共通語を除いた部分だけを利用して，ユーザの操作しているファイルと 3. 節で述べた方法で類似度を計算し（実験では 3.1.1 節での方法を利用した），高い順にランキングする．

5.　評 価 実 験

本節で，二つの実験を行う．5.3 節の実験では，3. 節で述べた 6 通りのファイル名によるファイル間類似度計算方法の比較を行い，どの方法が一番優れるかを調べる．5.4 節の実験では，4. 節で述べた，ファイルクラスタ内のファイルの複数のランキング方法の中で，一番優れる方法を調べる．

5.1　実験データ

実験では，実際に企業から提供された二つのファイルアクセスログを用いて行った．実験データの詳細を表 2 で示す．ログデータには，ユーザ名，アクセス時刻，ファイルパス，操作などの情報が時系列に記録されている．操作としては,「表示」,「更新」,「チェックアウト」,「チェックイン」などがある．

ログを 7 対 3 の割合で分割し，古い 7 割のログを学習用として，タスク，ワークフローの抽出を行い，新しい 3 割のログを評価用とする．評価用のログは，ユーザごとに分け，提案手法と同じ抽象ワークフローのギャップタイム (30 分) を用いて，ログを分割し，さらに，複数のテストケースを作成した．

5.2　実 験 方 法

評価方法について図 5 を用いて説明する．テストケース 1 つにつき，後述する適合率，再現率，F 値 [9] を求め，全テスト

表 2　実験データ

	ユーザ数	期間	レコード数	ファイル数	テストケース数
ログ A	22	約 8ヶ月間	9917	1417	151
ログ B	17	約 3ヶ月間	27706	8706	167

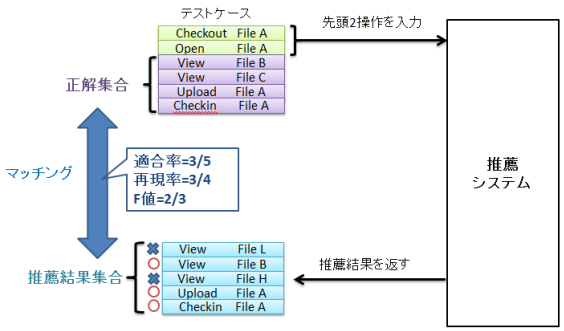

図 5　評価方法の例

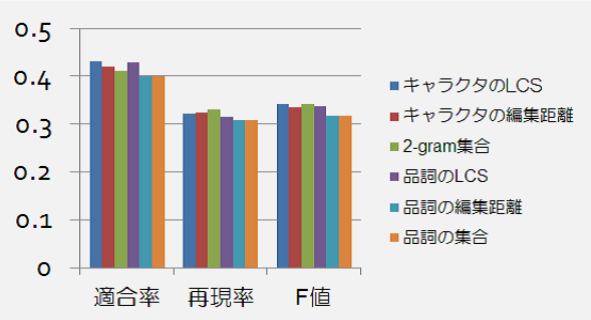

図 6　ログ A を使ったファイル名による類似度計算方法の比較

ケースでのそれぞれの平均をとる．テストケースのファイル操作列のうち，先頭から 2 操作をシステムへの入力とし，推薦結果集合を得る．テストケースのファイル操作列の先頭から 2 操作を除いた残りのファイル操作列を正解集合とする．結果集合と正解集合をマッチングすることで，以下の式で定義される適合率，再現率と F 値を求める．

$$\text{適合率} = \frac{|\,\text{結果集合} \cap \text{正解集合}\,|}{|\,\text{結果集合}\,|} \tag{7}$$

$$\text{再現率} = \frac{|\,\text{結果集合} \cap \text{正解集合}\,|}{|\,\text{正解集合}\,|} \tag{8}$$

$$F\text{ 値} = \frac{2 * \text{適合率} * \text{再現率}}{\text{適合率} + \text{再現率}} \tag{9}$$

5.3　ファイル名によるファイル間類似度計算方法の比較実験

この実験の目的は，3. 節で提案してきた各ファイル名によるファイル間類似度算出方法を比較することである．この実験においては推薦する時のファイルのランキング方法として，利用頻度順にランキングする方法を使った．

ログ A での実験結果を図 6 で示す．F 値でみると，キャラクタ単位での LCS，編集距離，2-gram 集合と品詞単位の LCS の四つの方法同士には大きい差がなかった．また，ファイル名を

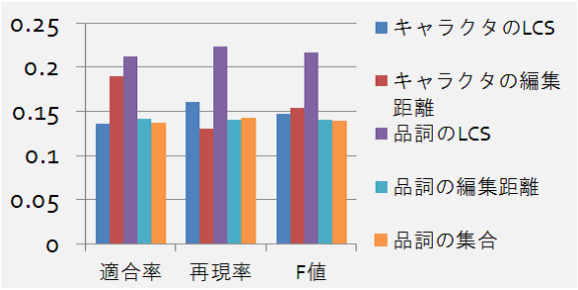

図 7　ログ B を使ったファイル名による類似度計算方法の比較

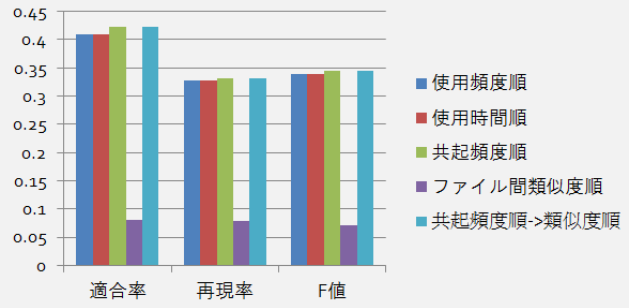

図 8　ログ A を使ったファイルランキング方法の比較

品詞に分割して，編集距離と集合で比較する方法は，キャラクタ単位のより適合率，再現率と F 値が下がったことも分かった．これの原因として，ファイル名を品詞単位で比較すると，共通部分が少なくなり，類似度が得られにくくなったことが考えられる．今回のログでは全部で 1400 個のファイルがあり，キャラクタ単位で類似度を計算する時，ほぼすべてのファイルが少なくとも一個以上の他ファイルと 0 より大きい類似度を持っているが，品詞に分割してから比較すると，約 1200 個のファイルしか他ファイルと 0 より大きい類似度を持つものが出なかった．つまり，200 前後のファイルが他のすべてのファイルと類似度が 0 となり，ファイルの抽象化の恩恵を受けられなかったと考えられる．また，ログ B での実験結果を図 7 で示す．品詞の LCS の方法が一番優れたことが分かった．結論として，どの類似度計算方法が良いかはログに依存する面があるが，品詞の LCS 法が比較的に安定していることが分かった．

5.4　ファイルクラスタ内にあるファイルのランキング方法の比較実験

この実験の目的は，4. 節で提案した 4 つのファイルのランキング方法を比較することである．

ログ A での実験結果を図 8 で示す．ファイル間類似度算出方法として，2-gram の集合の方法を使った．利用頻度或いは使用時間に基づくランキングする方法と比べ，今回新たに提案する共起頻度順にランキングする方法が適合率，再現率と F 値で全部やや優れたことが分かった．しかし，単体でファイル間類似度順にランキングする方法が出来が悪かった．その原因としては，今回の実験データでは，推薦候補のファイル名からファ

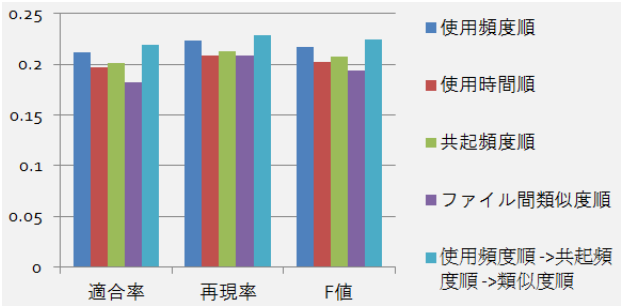

図 9　ログ B を使ったファイルランキング方法の比較

イルクラスタのラベルを除くと，非特徴部分がわずかしか残らず，ユーザが利用しているファイルと類似度をとると 0 になってしまうケースが多すぎたことが原因だと考えられる．今後こういう場合にも対応できる改善法を探す必要がある．また，共起頻度順と類似度順を組み合わせた方法（すべてのテストケースに対して，先に共起頻度順にランキングし，もし何も正しく推薦できなかったら，類似度順ランキングする方法に切り替わる）は，単体の共起頻度順の方法と変わらなかった．これの原因は，ログ A に対して，共起頻度順と類似度順のぞれぞれ推薦できたテストケースは包含関係を持っているからだ．

ログ B での実験結果を図 9 で示す．ファイル間類似度算出方法として，品詞の LCS 法を使った．実験結果では，使用頻度順は一位，共起頻度順は二位だった．また，使用頻度順，共起頻度順と類似度順を組み合わせた方法では，使用頻度順の単体より適合率，再現率と F 値とも高かった．つまり，使用頻度順でランキングする方法では正しく推薦できないが，共起頻度順や類似度順を使うと，正しく推薦できたテストケースがある．実験の結論として，共起頻度順と類似度順を組み合わせることで，推薦結果を改善できたことが分かった．

6. 関 連 研 究

小田切らによる FI 法 [10] はファイルの推薦ではなく，ファイル検索を目的とした研究である．ある一定期間内に使用したファイル群を一つのトランザクションに入れて，頻出するファイルの組み合わせを発見することによって，仮想的ディレクトリを生成する手法である．FI 法のアプローチとしては，よく一緒にアクセスされたファイル同士の関連度が高いことである．本研究のファイルクラスタ内のファイルのランキングする段階での「過去によく一緒にアクセスされたファイル同士に関連度が高く，このようなファイルを優先的に推薦する」点では同じ考え方である．しかし，FI 法ではワークフローの概念がなく，あるファイルと過去によく一緒に使われたファイル群がすべて推薦の候補となる．こうすると，ノイズが多く，精度よく推薦することが難しい．我々の研究では，まずユーザの操作パターンを推定してから，特定されたファイルクラスタに絞って，過去によく一緒に使われたファイルを探すため，ユーザの現在の操作パターンに沿って，適切な推薦ができる．

Wu らもファイル検索を目的にして、ファイルアクセスログから関連ファイルを発見する手法を提案した [11]。Wu らの研究ではタスク間の関連度を計算する際に、ファイル間の改名、移動、コピー操作を考慮した。我々の研究でのファイル間類似度計算にコピー操作による類似度を加味する点では同じ考え方である。Wu らの研究では，同一作業に関連するファイルは頻繁に近い時間に使われる傾向から、このようなファイル集合を「タスク」として抽出する。しかし，ファイル及びタスクの抽象化の概念がない．我々の提案手法ではタスクを抽象化する点と抽象化されたタスクのシーケンスを抽象ワークフローと定義する点が違う。

7. まとめと今後の課題

本研究では，抽象化する時のファイル名による類似度の計算方法と，推薦する時の次に推薦使用とするファイルクラスタの中のファイルのランキング方法を新しく提案した．更に，実験を通して，提案手法を比較評価した．

結論としては，類似度算出方法として，品詞の LCS 法は比較安定していることが分かった．また，ファイルランキング方法として，今回新たに提案する共起頻度順や類似度順の方法を組み合わせることで，推薦結果を改善できた．

今後の課題として，複数のファイルランキング方法を如何に組み合わせるの検討と，処理速度の改善などが挙げられる．

文　　献

[1] Susan Feldman, Joshua Duhl, Julie Rahal Marobella, and Alison Crawford. The hidden costs of information work. *IDC WHITE PAPER*, March 2005.

[2] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, Vol. 7, No. 1, pp. 76–80, 2003.

[3] 宋強, 川端貴幸, 伊藤史朗, 渡辺陽介, 横田治夫. ファイルレコメンデーションのためのファイル利用履歴に基づくタスク間ワークフロー抽出手法. 第四回データ工学と情報マネジメントに関するフォーラム (DEIM2012), 2012.

[4] Jianyong Wang and Jiawei Han. Bide: Efcient mining of frequent closed sequences. *Proceedings of 20th International Conference on Data Engineering*, pp. 79–90, 2004.

[5] Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, Vol. 24, No. Issue 4, pp. 664–675, 1977.

[6] Wilbert Heeringa. *Measuring Dialect Pronunciation Differences using Levenshtein Distance*. Doctoral dissertation. University of Groningen, 2004.

[7] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, Vol. 26, No. 3, pp. 279–302, 1945.

[8] http://www.mlab.im.dendai.ac.jp/ yamada/ir/MorphologicalAnalyzer/Sen.html.

[9] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[10] 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. 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, 2009.

[11] Yi Wu, Kenichi Otagiri, Yousuke Watanabe, and Haruo 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)*, pp. 364–378, 2011.