論文 / 著書情報
Article / Book Information

論題(和文)	XML類似性に着目した抽象タスク間ワークフローに基づくファイル操作推薦手法
Title(English)	
著者(和文)	鈴木凌太郎, 横田治夫
Authors(English)	Ryotaro Suzuki, Haruo Yokota
出典(和文)	第8回データ工学と情報マネジメントに関するフォーラム論文集, , , F7-F
Citation(English)	, , , F7-F
発行日 / Pub. date	2016, 3

XML類似性に着目した抽象タスク間ワークフローに基づくファイル操作推薦手法

鈴木凌太郎† 横田 治夫††

† 東京工業大学工学部情報工学科 〒152-8552 東京都目黒区大岡山 2-12-1
†† 東京工業大学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
E-mail: †rsuzuki@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし 近年, ファイルシステム内のファイル数は急激に増加しており, 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. 我々は, そのような労力や時間を削減するため, ユーザのファイル操作履歴における一連の作業を抽象化してパターンマイニングを行うことで, 直近のファイル操作に続くファイルと作業を推定する手法の研究を行ってきた. その推定においては, ファイル間のテキスト類似度より, ファイルの構造情報等が有用となる. 本研究では, 複数の XML ファイルで構成される Office 文書を対象に, XML 間の葉ノードに着目した類似度計算を抽象化におけるクラスタリング処理へ組み込むことを試みる. 先行研究の推定結果と比較する事で, 提案手法の評価を行う.

キーワード ファイルレコメンデーション,XML 類似度, クラスタリング, 抽象化, ワークフロー

1. はじめに

近年, 情報の増加によってファイルシステムのファイル数が劇的に増加しており, 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. ユーザのオフィスにおける作業時間の約 4 分の 1 が検索作業だと言われ [1], ユーザが生産的な活動に専念するためには, 検索作業の時間や労力の削減が重要になる.

また, ユーザの検索対象は文書等の情報源としてのファイルだけに限らず, 仕事を達成するためのノウハウを得るための情報も探している. 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自動的にマイニングし, それを元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的な作業に専念させることが出来ると考え, 推薦手法の開発研究が先行研究として行われてきた [12]. この手法ではユーザのファイル操作履歴における一連の作業を抽象化しパターンマイニングを行うことで, ユーザの直近のファイル操作に続くファイルと作業を推定し, それを元に推薦を行う. 推定においてはファイル間の類似度としてファイル名, コピー関係に着目した類似度の 2 種類を用いているが, 主な検索対象になるファイルは文書形式であるため, ファイル名を用いた類似度計算をするよりもファイルのフォント, 配置, サイズなどのスタイル情報や構造情報に着目し類似度計算をすることによって, より精度の高い推定が可能になると考えられる.

そこで本研究では, 複数の XML ファイルで構成される Office 文書を対象にして, XML 間の葉ノードに着目した類似度計算を抽象化におけるクラスタリング処理へ組み込むことを試みる. また先行研究との推定結果と比較することで, 提案手法の評価を行う.

2. 関連研究

既存の関連文書推薦の技術としては協調フィルタリング [4] が知られている. この手法では, ユーザの今後の行動を過去の行動や趣向の似ているユーザの行動をもとにして予測を行う. この手法をファイル推薦システムで利用することは可能であるが, 協調フィルタリングでは操作の順番が厳しく, 過去に行った同一ファイル操作しか推薦することはできない問題点がある.

先行研究で以前提案された手法 [12] では, ユーザの一連の作業を抽象化しパターンマイニングを行うことで, 操作パターンがわずかに異なるようなパターンも共通のパターンとして考えて推薦に利用することができる.

また, 既存の XML 類似度計算手法としては多くの方法が存在するが, Tree Edit Distance(TED) [3] は最もよく知られている手法である. これは一般的な文書の比較に用いられているレーベンシュタイン距離 [2] を木構造に適用したもので精度が高いという特徴があるが, 最低でも計算量が $O(n^3)$ と高いため, 文書量が多い XML 文書や多くの XML 文書を比較することには向いていない. 精度が高く計算量は多くはないアルゴリズムとしては LAX [9], LAX+ [10] が知られている. これは XML 木の葉ノードを比較することによって構造情報を考慮した類似度計算を行う. さらに LAX+の拡張アルゴリズムとして文書中のキーワードに着目し精度を上げた KLAX, LAX&KEY [11] も知られている. Office 文書 (docx, xlsx, pptx) は複数の XML ファイルで構成されているため, これらの XML ファイル間類似度手法を用いることにより Office 文書ファイル間の類似度を測ることが可能である.

これらを踏まえ本研究では, 先行研究の推薦手法 [12] へ, 既存の XML ファイル類似度計算手法（KLAX, LAX&KEY）を組み込むことによって精度向上を目指す. 次節からは必要となる具体的な手法の説明を行う.

3. 推薦手法

本節では, 推薦手法 [12] の説明を行う. ここでタスクは一定

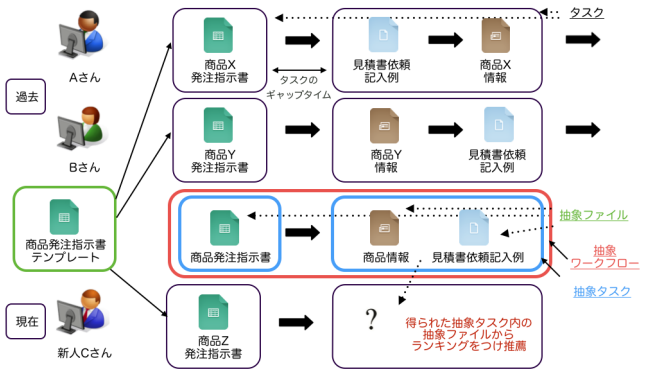

図 1　推薦手法 [12] の概略図

時間（タスクのギャップタイム）内に操作されたファイルとその操作の集合というように定義され, またワークフローは一定時間（ワークフローのギャップタイム）内に発生したタスクの集合と定義されている. コピー関係, ファイル名によって類似度の高いファイル同士はクラスタ化され, 抽象ファイルとして定義される. ファイル間の類似度については式 (1) のように定義される.

$$Sim(fileA, fileB) = a \cdot simFileName(fileA, fileB) + b \cdot simCopy(fileA, fileB) \quad (1)$$

ここで $simFileName, simCopy$ がそれぞれファイル名類似度, コピー関係類似度となっており, a, b はそれぞれの重みである. また $(fileA, fileB)$ が類似度を計算するファイルのペアである. クラスタリングの際は $a + b$ で割って正規化した値を用いる.

ファイル名類似度についてはファイル名を一般名詞単位のシーケンスと見なし, 式 (2) でシーケンス同士から LCS を求めることでファイル間の類似度を計算している.

$$simFileName(fileA_w, fileB_w) = \frac{len_w(LCS_w(fileA_w, fileB_w))}{min_w(len_w(fileA_w), len_w(fileB_w))} * d. \quad (2)$$

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 を表している.

またコピー関係による類似度とは, コピー元とコピー先の関係を持っている 2 つのファイルの間の類似度のことである. 例えば同じテンプレートからコピーされたファイル同士には, 利用目的のある程度共通していると考えられる. 具体的には以下の式 (3) によって, コピー操作に関わるファイル同士の類似度を定義している.

$$simCopy(fileA, fileB) = \begin{cases} c^{steps} & (if\ copy\ relationship\ exists) \\ 0 & (otherwise). \end{cases} \quad (3)$$

$fileA$ と $fileB$ がコピー関係を持っているとする. ここで, $steps$ は $fileA$ から $fileB$ までの遷移ステップ数, c が減衰係数を表す. この手法では c を 0.9 に設定されている. 例えば ファイル A, B と C は「A → B → C」の順番でコピーされた場合, 親子関係の A と B の間の類似度は 0.9 で, 親孫関係の A と C 間の類似度は減衰し 0.81 となる.

図 1 は A さん, B さんの過去のファイル操作履歴を元にして, 新人 C さんに次のファイルとその操作を推薦するという状況になっている. この図を用いた推薦までのフローの説明は以下のとおりである.

（1） A さんの商品 X 発注指示書の操作と B さんの商品 Y 発注指示書の操作は, ファイル名の類似度が高く同じテンプレートからファイルが作成されているという関係（コピー関係）から, 抽象ファイルとしてまとめられる.

同様にして, 商品情報と見積書依頼記入例という抽象ファイルが作成される.

（2） 1 で作成された複数の抽象ファイルをタスクとしてまとめることによって抽象タスクが得られ, さらに抽象タスクをまとめることで抽象ワークフローが得られる.

（3） 新人の C さんが商品 Z の発注指示書を操作していることと過去のファイル操作履歴から, 次に操作するファイルは, 商品情報または見積書依頼記入例の抽象ファイル内のファイルであると推定される.

（4） 推定された抽象ファイル内からファイルを選定し推薦を行う. 選定の基準は過去の利用頻度, 最近使用された順, 共起頻度順, 類似度順から選択し, それぞれのファイルにスコアリングをして値の高いものが推薦結果となる.

タスクの中には順不同であるような操作が含まれていると考えられるが, 抽象化したことにより商品情報と見積書依頼記入例は順不同の操作として, ここではよりスコアの高いものを次の操作として推薦できるようになっている.

既存の推薦手法の説明は以上である. 次節では, XML ファイル間類似度計算手法の説明をする.

4. XML ファイル間類似度計算手法

本節では XML ファイル間類似度を求めるアルゴリズムである LAX+, KLAX, LAX&KEY の説明と, これらの手法により求めた XML 類似度を統合し Office ファイル間類似度を計算する手法である SOS [10] を説明する.

4.1 LAX+

LAX+は LAX [9] を拡張した XML 類似度計算アルゴリズムで, XML 木のペア (f_{base}, f_{target}) を入力とし類似度 $S_{xml}(f_{base}, f_{target})$ を出力する. LAX+は対称性があるため $S_{xml}(f_{base}, f_{target})$ は $S_{xml}(f_{target}, f_{base})$ と等しい. XML 木の類似度は XML の部分木の類似度から求まり部分木の類似度は葉ノードの一致度によって計算を行う. 計算過程は以下のとおりである.

（1） **XML 木から部分木の作成**:
まず XML 木から部分木を作る際の基準として, ふさわしいノードをカッティングポイントとして定義する. XML 木内の各ノード x について, 以下の式を適用していくことでカッティングポイントを決定する.

$$w_x = |children(x)| \times d_x^{\alpha} \quad (4)$$

$|children(x)|$ は x の子ノードの数であり, d_x は x から葉ノードまでの最大深さである. また α は重みである. f_{base} は分割され部分木の集合 $T_{base} = \{t_u\}$ となり, 同様に f_{target} も分割され部分木の集合 $T_{target} = \{t_v\}$ となる.

（2） **XML 部分木間の類似度計算**:
部分木のペア (t_u, t_v) を base 部分木と target 部分木の要素から作り, 以下の式 (5) を適用することで類似度を計算できる.

$$S_{sub}(t_u, t_v) = |Matched\_Leaf(t_u, t_v)| \quad (5)$$

$Matched\_Leaf(t_u, t_v)$ は, t_u と t_v の中での一致した葉ノードの数である. ここでの “一致” は, 葉ノードの内容が完全に一致していることを指している.

（3） **部分木間の類似度を用いた XML 木の類似度計算**:
$S_{xml}(f_{base}, f_{target})$ は, XML 木 f_{base} と f_{target} を用いて計算した LAX+による XML 類似度で, 以下の式で計算する.

$$\begin{aligned} &S_{xml}(f_{base}, f_{target}) \\ &= min(\\ &\quad \frac{\sum_{t_u \in T_{base}} max_{t_v \in T_{target}}(S_{sub}(t_u, t_v))}{|Leaf(f_{base})|}, \\ &\quad \frac{\sum_{t_v \in T_{target}} max_{t_u \in T_{base}}(S_{sub}(t_u, t_v))}{|Leaf(f_{target})|} \\ &) \end{aligned} \quad (6)$$

4.2 キーワードベースの類似度

ここでは KLAX, LAX&KEY で用いられるキーワードベースの類似度計算について説明する. まず XML 内のコンテンツに含まれるキーワード（名詞のみ）を取り出し, それらを各ファイルのキーワードとする. ここでのコンテンツとは Office 文書を開いた際の読み手が見る内容のことである. また, 名詞は言葉の一部となっているようなものは除いて取り出すこととする ([5], [6]). こうして取り出した名詞を元にして, キーワードベースの類似度は以下のようなコサイン類似度を用いて計算する.

$$S_{keyword}(f_{base}, f_{target}) = \frac{B \cdot T}{|B||T|} \quad (7)$$

◇ B は, base ファイルの内容または base 葉ノードに含まれるキーワードのベクトル

◇ T は, target ファイルの内容または target 葉ノードに含まれるキーワードのベクトル

◇ $|B|$ はベクトル B の大きさ.

◇ $|T|$ はベクトル T の大きさ.

LAX+のように構造情報のみで類似度を測った場合の問題点として, 内容がほぼ同じであったとしても式 (5) によって, 完全一致しない限り類似度が 0 になってしまうことが挙げられる. 例えば図 2 のような 2 つの部分木を比較した場合, 部分木の内容はほぼ一致しているのにもかかわらず LAX+ではスコア 0 になる. こういった場合は, キーワードベースの類似度も考慮することで精度を向上させることができる. (図 2 の subtree1,2 内の②はタグと属性, 属性の値の内容は等しいが, 要素のテキストを持っていないため葉ノードのマッチングはしていないことに注意)

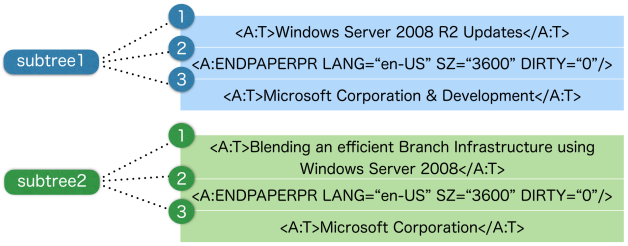

図 2　LAX+の問題点

4.3 KLAX

KLAX は LAX+ [10] を拡張したアルゴリズムで, 文書の構造情報に着目し XML 類似度を測る LAX+とは異なり, 文書中のキーワードの一致度も考慮したアルゴリズムである.

KLAX を計算する際には, コンテンツとは独立してタグと属性, また属性の値についての類似度を計算している. これらすべてが一致していれば式 (7) を用いてキーワードベースの類似度を計算する. 式は以下のとおりである.

$$\begin{aligned} &S_{K\_leaf}(l_u, l_v) \\ &= \begin{cases} 0 & \text{if tags are not same} \\ \gamma + ((1-\gamma) \cdot S_{keyword}(l_u, l_v)) & \text{otherwise} \end{cases} \end{aligned} \quad (8)$$

本研究では $\gamma = 0.5$ で行う. ここで (l_u, l_v) は, 部分木のペア (t_u, t_v) 内の対応する葉ノードのペアである.

KLAX では, LAX+の第 2 段階の式 (5) が以下のように置き換えることで計算結果を得る.

$$S_{sub}(t_u, t_v) = \frac{\sum_{l_u \in Leaf(t_u)} max_{l_v \in Leaf(t_v)}(S_{K\_leaf}(l_u, l_v))}{min(|Leaf(t_u)|, |Leaf(t_v)|)} \quad (9)$$

ここで $|Leaf(t_u)|$ と $|Leaf(t_v)|$ は, 部分木 t_u と t_v の中にある葉ノードの数である.

4.4 LAX&KEY

LAX&KEY は LAX+の類似度とキーワードベースの類似度を別々で計算し, これらの重み付きの線形和とることで最終的な類似度とする. 式は以下のとおり.

$$\begin{aligned} &S_{LAX\&KEY}(f_{base}, f_{target}) = \beta \cdot S_{xml}(f_{base}, f_{target}) \\ &\quad +(1-\beta) \cdot S_{keyword}(f_{base}, f_{target}) \end{aligned} \quad (10)$$

◇ $S_{xml}(f_{base}, f_{target})$ は, (f_{base}) と (f_{target}) の LAX+で計算した類似度である

◇ $S_{keyword}(f_{base}, f_{target})$ は, (f_{base}) と (f_{target}) のキーワードベースの類似度である

◇ β は重みで, LAX+とキーワードベースの類似度それぞれの値の効果を調整する ($\beta \in (0, 100)$). 本研究では $\beta = 50$ で行う.

4.5 SOS

OOXML 文書は docx, xlsx, pptx のような複数の XML ファイルで構成される文書で, Zip アーカイブとして保存されているため解凍することで構造を展開できる. OOXML 文書の中身は幾つかの Markup Language(ML) で定義されており, さらに ML は複数の XML ファイルで構成されている. 例えば, WordProcessing ML はワープロ, SpreadSheet ML は表計算, Presentation ML はプレゼンテーションを記述するために用いられる ML であり, それぞれの中身には ML の構成に必要な複数の XML ファイルが含まれている. この節で説明する SOS(Structure-Based Office Document Search) は, OOXML 文書間の類似度を内部の XML ファイルを利用して木の階層を考慮しつつ計算する手法である.

o_q は OOXML 文書における類似度検索のクエリで, o_i $(1 \geqq i \geqq n)$ はデータベース中の中にある OOXML 文書であるとする. 各 OOXML 文書 o_i は, 複数のパーツ $P_i = \{p_{ij} | 1 \leqq j \leqq J_i\}$ で構成されており, また各パーツ p_{ij} は, XML ファイル $F_{ij} = \{f_{ijk} | 1 \leqq k \leqq K_{ij}\}$ の集合で構成されている. また, o_q パーツの集合 P_q と XML ファイルの集合 F_{qj} を含んでいる.

OOXML 文書間の類似度を計算する手順は以下のとおりである.

（1） **XML ファイル間の類似度計算**:
XML のファイル名が同じものについて計算を行うので, まず XML ファイルのペア (f_{qjk}, f_{ijk}) を作る. ペアは各 OOXML 文書 o_q と o_i の j 個目のパーツである. 以下では, 各ペア $S_{xml}(f_{qjk}, f_{ijk})$ の類似度を計算する.

（2） **パーツ間類似度 S_{parts} を S_{xml} の総乗により計算**:
$S_{parts}(p_{qj}, p_{ij})$ は, j 個目のパーツの類似度である.

$$S_{part}(p_{qj}, p_{ij}) = \frac{\sum_{f_{qjk} \in F_{qj}, f_{ijk} \in F_{ij}} S_{xml}(f_{qjk}, f_{ijk})}{max(|F_{qj}|, |F_{ij}|)} \quad (11)$$

（3） **OOXML 文書間類似度を S_{parts} の総乗により計算**:
$S_{oo}(o_q, o_i)$ は, OOXML 文書のペア (o_q, o_i) の類似度である.

$$S_{oo}(o_q, o_i) = \sum_{p_{qj} \in P_q, p_{ij} \in P_i} \left(\frac{w_j}{\sum_{l=1}^{|P_q|} w_l} \times S_{part}(p_{qj}, p_{ij}) \right) \quad (12)$$

ここで w_j は, j 個目のパーツに関する重みである.

5. 提 案 手 法

本研究では節 3 で説明した推薦手法へ節 4 で説明した XML ファイル間類似度計算手法（KLAX, LAX&KEY）を組み込むことによって, 精度向上を目指す.

本研究における XML ファイル間類似度計算手法は, 節 3 フローの (1) において組み込みを行う. その際には, 既存のファイル間類似度手法のファイル名による類似度とコピー関係類似度との重み付き線形和をとったものを最終的なファイル間の類似度とする. 以下の式 (13) は最終的なファイル間類似度の式である

$$\begin{aligned} Sim(fileA, fileB) &= a \cdot simFileName(fileA, fileB) \\ &+ b \cdot simCopy(fileA, fileB) \\ &+ c \cdot simXML(fileA, fileB) \end{aligned} \quad (13)$$

ここで $simFileName, simCopy, simXML$ がそれぞれファイル名類似度, コピー関係類似度, XML ファイル間類似度となっており, a, b, c はそれぞれの重みである. $simXML$ には KLAX, LAX&KEY のどちらかを利用した場合の XML ファイル間類似度を選択する. また ($fileA, fileB$) が類似度を計算するファイルのペアである. a, b, c をそれぞれ適切な値に設定することで, 既存の手法よりもさらに柔軟で精度の高い推薦が可能になると考えられる. クラスタリングの際には, この類似度を $a + b + c$ の値で割り正規化した値を用いる.

6. 評 価 実 験

本節では 2 つの実験を行う. 1 つは既存のファイル間類似度と XML ファイル間類似度の適切な重み付けパラメータを調べる実験で, もう 1 つは XML ファイル間類似度を導入したことによる推薦結果の比較実験である. どちらの実験についても, 既存の推薦手法 [12] とそこへそれぞれ KLAX, LAXKEY を組み込んだ場合について行う. 最後には実行時間について述べる.

6.1 対象のログデータ

ログにはアクセスしたタイムスタンプ, ユーザ ID, ファイルパス, 操作などの情報が時系列順に記録されている. 操作としては, 「新規」, 「書込」, 「削除」, 「改名」, 「参照」などがある. ログを 8:2 の割合で分割し, 古い 8 割を学習用, 新しい 2 割を評価用とする. またログにはプログラムによる自動操作がノイズとして入ってしまうことが考えられるので, 人手でクリーニングを行う. 自動操作が入ってしまうと特定の数秒の間に大量のファイル操作ログが残ってしまい, タスクやワークフロー抽出の際に悪影響が出る. 評価用のログはユーザごとに分け, ワークフローの分割時間を元にログを分割し, これらをテストケースとして作成する.

本研究では, 我々の研究室内のファイルサーバーにおけるファイルアクセスログを利用する. ログの実験データの詳細は表 1, 操作の割合は図**??**のとおりである. また, XML ファイル間類似度計算で測定する OOXML 文書数は表 2 のとおりである. OOXML 文書数がはファイル数に比べてかなり低めになっているが, これは XML ファイル間類似度計算可能な OOXML 文書数という意味であり, ログには残っているが今は削除されてしまっているような OOXML 文書数は含まれないことからこのような数になっている.

また以前の研究 [12] では企業から提供されたアクセスログを用いており, 今回用いる研究室内のログとは大きく傾向が異なることに注意されたい.

表 1 実験データ

ユーザ数	期間	レコード数	ファイル数
25	約 2 年	34471	8161

表 2　OOXML 文書数

total	docx	xlsx	pptx
94	32	30	32

6.2　評価方法

評価方法の具体的な例として, 図 3 を参照しつつ説明する. 評価用ログから得られたテストケースにおいて, 先頭 2 操作を入力とすることによりシステムから推薦結果集合が得られる. 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を正解集合として, 結果集合と正解集合から, 以下に示す適合率, 再現率, F 値 [8] を求められる.

$$適合率 = \frac{|\,結果集合 \cap 正解集合\,|}{|\,結果集合\,|} \tag{14}$$

$$再現率 = \frac{|\,結果集合 \cap 正解集合\,|}{|\,正解集合\,|} \tag{15}$$

$$F\ 値 = \frac{2 * 適合率 * 再現率}{適合率 + 再現率} \tag{16}$$

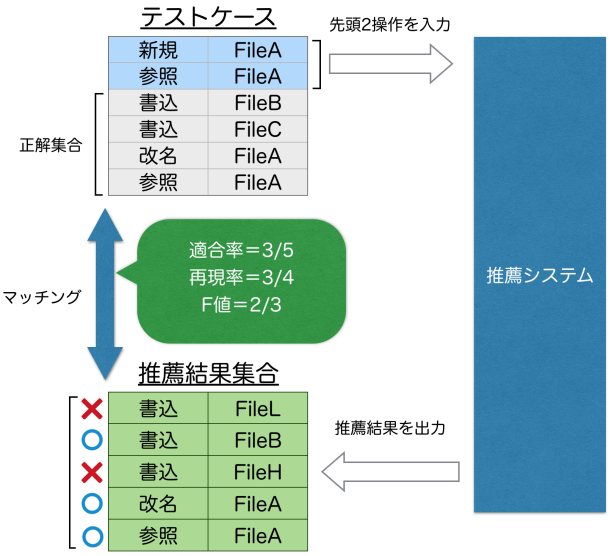

図 3　評価方法の例

6.3　既存のファイル間類似度計算手法との比較実験

6.3.1　各類似度の重みパラメータチューニング実験

この実験の目的は, 提案した XML ファイル間類似度を組み込む際の重みが推薦結果にどう影響するかを調べることと最適な重みパラメータを調べることである.

ここでは, 既存のファイル間類似度（ファイル名, コピー関係）に XML ファイル間類似度を組み込む際の線形和の重み a, b, c の値を調節し適切な値を調べる. この重みは抽象ファイルを求めることに大きく影響し, また抽象ファイルは推薦システムの重要な要素になるため, 推薦結果に大きく効いてくるパラメータである. a, b, c それぞれについて 0 ～ 3 程度の幅で変化させ, 平均適合率, 平均再現率, 平均 F 値を見ることで評価を行う.

基本的にはファイル名類似度はすべてのファイルペアに対して定義できるが, コピー関係, XML ファイル間類似度については一部のファイルペア間のみで定義される類似度になるため, a は必ず 0 以上に設定する. またコピー関係に関しては, 実験ログから得られる改名操作を用いて擬似的なコピー関係としている.

（1）　KLAX を用いた重みパラメータチューニング

表 3, 表 4, 表 5 が実験結果である. 色が薄いセルがより高い値となっており, 各表の最大値のセルは赤字で囲んである. 一番左の列の $(1,0), (1,1)...$ がファイル名類似度とコピー関係類似度の組 (a, b) を示しており, 一番上の行の $0, 1, 2, 3$ は XML ファイル間類似度 c となっている.

KLAX を組み込むことにより評価結果として, 平均適合率と特に平均再現率は高いものを得ることができるようになった. これによって, 類似度の高いファイルが同じクラスタにより纏まりやすくなったと言える. 既存の結果として最も平均 F 値が高いものは $(a : b : c) = (2 : 3 : 0)$ の時で 0.144 である. KLAX を組み込んだ結果としては $(a : b : c) = (3 : 3 : 1)$ の時の 0.173 が最も高く, 適切なパラメータとなる.

$c = 0$ の中ではファイル名による類似度と比べて, コピー関係による類似度の方が信頼性があると考えられる. 例えば, ユーザがあるテンプレートファイルからファイルをコピーして新規で他のファイルを作成した時, 両ファイル名が必ずしも一致しないが, コピー操作を用いることによって確実に両者の関係が判明する.

このような時には特にコピー関係が影響してくるため, $(a : b : c) = (1 : 2 : 0)$ や $(a : b : c) = (2 : 3 : 0)$ の時に既存手法の中では平均 F 値が高くなっていることが分かる.

一方, c が 1 以上の時は必ずしも b が a より高いということはなく, むしろ b より a をやや高めか同じにした場合の方が, 全体的に高い評価結果となっており, また $a = 3$ の時に関してはすべての評価結果において導入後の方が高い結果が得られている. これは類似度計算の手法が増えたことにより, 類似度を正規化する際に 1 つ 1 つの値は相対的に低くなるからで, コピー関係や XML 類似度とは違いどのファイル間においても定義できるファイル名類似度の重みを高くしておくことで安定した結果が得られていることが分かる.

（2）　LAX&KEY を用いた重みパラメータチューニング

表 6, 表 7, 表 8 が実験結果である. KLAX 同様, 色が薄いセルがより高い値となっており, 各表の最大値は青いセルで囲んである.

LAX&KEY を組み込むことにより評価結果として, 多くの場合は評価結果の向上につながってはいるが, KLAX の時と比べるとそれほど値が伸びていないものもいくつか見受けられた. KLAX は LAX&KEY に比べて葉ノードのキーワードの一致による類似度の効き方が大きいため, その違いがこうして出てきたと考えられる.

既存の結果として最も平均 F 値が高いものは $(a : b : c) = (2 : 3 : 0)$ の時で 0.144 である. LAX&KEY を組み込んだ結果としては $(a : b : c) = (3 : 3 : 1)$ の時の 0.170 が最も高く, KLAX を組み込んだ結果と合わせて, 提案手法では $(a : b : c) = (3 : 3 : 1)$ が適切なパラメータだと言える.

表 3　各重みに対する平均適合率 (KLAX)

(a,b)\c	0	1	2	3
(1,0)	0.390	0.282	0.341	0.201
(1,1)	0.282	0.341	0.207	0.178
(1,2)	0.336	0.246	0.197	0.192
(1,3)	0.301	0.197	0.192	0.192
(2,0)	0.390	0.355	0.282	0.366
(2,1)	0.361	0.275	**0.425**	0.341
(2,2)	0.282	0.405	0.282	0.239
(2,3)	0.346	0.320	0.240	0.246
(3,0)	0.390	0.383	0.370	0.282
(3,1)	0.383	0.350	0.282	0.352
(3,2)	0.341	0.287	0.352	0.323
(3,3)	0.282	0.379	0.323	0.341

表 4　各重みに対する平均再現率 (KLAX)

(a,b)\c	0	1	2	3
(1,0)	0.112	0.120	**0.147**	0.109
(1,1)	0.104	0.145	0.112	0.087
(1,2)	0.101	0.122	0.099	0.124
(1,3)	0.105	0.099	0.124	0.124
(2,0)	0.112	0.102	0.120	0.140
(2,1)	0.071	0.120	0.119	**0.147**
(2,2)	0.104	0.113	0.145	0.096
(2,3)	0.109	0.100	0.099	0.122
(3,0)	0.112	0.121	0.121	0.120
(3,1)	0.121	0.122	0.121	0.141
(3,2)	0.104	0.123	0.142	0.143
(3,3)	0.104	0.143	0.144	0.145

表 5　各重みに対する平均 F 値 (KLAX)

(a,b)\c	0	1	2	3
(1,0)	0.118	0.126	0.153	0.113
(1,1)	0.115	0.152	0.115	0.089
(1,2)	0.133	0.125	0.100	0.110
(1,3)	0.101	0.100	0.110	0.110
(2,0)	0.118	0.125	0.126	0.170
(2,1)	0.095	0.125	0.142	0.153
(2,2)	0.115	0.135	0.152	0.103
(2,3)	0.144	0.112	0.105	0.125
(3,0)	0.118	0.122	0.132	0.126
(3,1)	0.122	0.127	0.127	0.167
(3,2)	0.105	0.129	0.167	0.143
(3,3)	0.115	**0.173**	0.143	0.152

表 6　各重みに対する平均適合率 (LAX&KEY)

(a,b)\c	0	1	2	3
(1,0)	**0.390**	0.283	0.342	0.201
(1,1)	0.282	0.334	0.186	0.154
(1,2)	0.336	0.107	0.203	0.192
(1,3)	0.301	0.203	0.198	0.198
(2,0)	**0.390**	**0.390**	0.283	0.349
(2,1)	0.361	0.293	0.339	0.342
(2,2)	0.282	0.298	0.334	0.232
(2,3)	0.346	0.308	0.233	0.238
(3,0)	**0.390**	0.361	0.352	0.283
(3,1)	0.383	0.363	0.283	0.328
(3,2)	0.341	0.287	0.328	0.323
(3,3)	0.282	0.378	0.323	0.334

表 7　各重みに対する平均再現率 (LAX&KEY)

(a,b)\c	0	1	2	3
(1,0)	0.112	0.121	0.131	0.109
(1,1)	0.104	0.127	0.088	0.063
(1,2)	0.101	0.104	0.102	0.124
(1,3)	0.105	0.102	0.128	0.128
(2,0)	0.112	0.103	0.121	0.117
(2,1)	0.071	0.122	0.114	0.131
(2,2)	0.104	0.109	0.127	0.078
(2,3)	0.109	0.103	0.081	0.104
(3,0)	0.112	0.117	0.124	0.121
(3,1)	0.121	0.112	0.121	0.114
(3,2)	0.104	0.121	0.114	0.128
(3,3)	0.104	**0.140**	0.130	0.127

表 8　各重みに対する平均 F 値 (LAX&KEY)

(a,b)\c	0	1	2	3
(1,0)	0.118	0.128	0.138	0.113
(1,1)	0.115	0.135	0.092	0.065
(1,2)	0.133	0.107	0.103	0.110
(1,3)	0.101	0.103	0.113	0.113
(2,0)	0.118	0.126	0.128	0.148
(2,1)	0.095	0.129	0.144	0.138
(2,2)	0.115	0.131	0.135	0.086
(2,3)	0.144	0.124	0.087	0.107
(3,0)	0.118	0.142	0.131	0.128
(3,1)	0.122	0.125	0.128	0.141
(3,2)	0.105	0.128	0.141	0.128
(3,3)	0.115	**0.171**	0.128	0.135

6. 3. 2　XML ファイル間類似度導入による効果の検証実験

ここでは XML ファイル間類似度を導入したことによる推薦結果の変化について比較する．比較については，前述のチューニング実験において c が 1 以上の評価結果の中で最も F 値が高かった場合の推薦結果と，$c = 0$ の評価結果の中で最も平均 F 値が高かった $(a, b, c) = (2 : 3 : 0)$ の推薦結果を比べることにより行う．

（1）　**KLAX 導入による効果**

表 5 より，KLAX 導入後 (c が 1 以上) の評価結果の中で最も F 値が高かった重みは $(a, b, c) = (3, 3, 1)$ である．これと KLAX 導入前 $(c = 0)$ の評価結果の中で最も平均 F 値が高かった $(a, b, c) = (2 : 3 : 0)$ の推薦結果を比較する．表 9 はテストケースの入力パターンに対する変化をまとめたものである．

新たに推薦可能になっているテストケースを調べると，タスクの推定が不可能であったものが可能になっていたり，タスクの推定ができていても，ワークフローの推定ができていないものが可能になっていたりするものが見られた．

また推薦不可能になっているものでは，類似度の高いタスクがなく無理やりタスクを推定していたケースにおいて，推薦不可能にしているものも見られ，ノイズの除去として働いている例も見られた．

こうして推薦結果が可能，不可能に変化したテストケース入力パターンを見ると，推薦システムへの 2 つの入力ファイルとしては docx ファイルが圧倒的に多く，xlsx ファイルが少数だが含まれており，pptx ファイルは 0 というような傾向であった．

これまでの結果から，docx ファイルが入力であった時にはより精度の高い推薦が可能になっていることが分かった．

xlsx ファイルが入力である場合の推薦結果への寄与が少なくなっている理由としては，xlsx には ID や数字などのデータが多く含まれているため，キーワードの一致を重視した KLAX は効きづらくなっているからと考えられる．また pptx が入力になっているようなテストケースでは推薦結果に変化がない理由として，今回実験で XML 類似度を測定した OOXML 文書の中では，pptx ファイルは book_reading20160101, progress_report0101 のような名前がほぼ同じで日付の部分だけ異なるようなファイルが多く含まれていたため，ファイル名類似度によってすでに正確な類似度が計算されており，変化がなかったと考えられる．

（2）　**LAX&KEY 導入による効果**

表 8 より，c が 1 以上の評価結果の中で最も F 値が高かった重みは $(a, b, c) = (3, 3, 1)$ である．同様にこの結果と LAX&KEY

表 9　テストケースの入力パターンに対する KLAX 導入後の推薦と評価結果の変化

テストケースの入力パターン	変化
doc doc	F 値が上昇した
docx docx	新しく推薦可能になった
docx docx	F 値の低い結果が推薦不可になった（ノイズの除去）
xlsx doc	新しく推薦可能になった
ppt ppt	新しく推薦可能になった

導入前 ($c = 0$) の評価結果の中で最も平均 F 値が高かった $(a, b, c) = (2 : 3 : 0)$ の推薦結果を比較する. 表 9 はテストケースの入力パターンに対する変化をまとめたものである.

基本的な変化としては, KLAX 導入時と変わらない傾向が見られた.

テストケース入力パターンを見ると, 推薦システムへの 2 つの入力ファイルとしては doc, docx と pptx ファイルが多く, xlsx ファイルが少数だが含まれているというようになっていた.

docx については KLAX の時と同様で, docx の場合は単純に類似度の精度が上がったことで, 推薦結果の向上につながったのだと考えられる. xlsx の場合, LAX&KEY は構造情報に着目した LAX+とキーワード類似度を合わせて用いているため, 特にキーワードに着目した KLAX とは違う傾向が見られるように思われたが, KLAX とあまり変化がなかった.

また LAX&KEY を用いた xlsx の類似度計算結果を見てみたところ, どれもかなり低い値になっておりうまく類似度が測れなかったようである. まったく同じファイル間の類似度を KLAX で計算した結果では, かなり高いものもあれば低いものもみられたため, 類似度自体は KLAX と LAX&KEY の両手法で大きく異なっているが, 推薦結果はあまり変わらないという結果となっていた. こういった結果が出る理由としては, 多少のファイルの類似度の精度が上がりファイルクラスタの質が上がったとしても対象になっている xlsx ファイルは多くはないため, 必ずしも推薦結果に反映されてくるわけではないからであると考えられる.

今回 pptx がテストケースの入力に含まれているものは, 評価結果が下がってしまったり, 推定したタスクだけ変わり評価結果が変わらないなどしているものだった. LAX&KEY を用いた xlsx の類似度計算結果を見てみたところどれもかなり低い値になってしまっており, もともと実験データの pptx はファイル名がかなり一致しているものが多かったため, ファイル名で精度の高い類似度を計算できていたのに最終的な類似度を劣化させてしまったからだと考えられる.

まとめると特に docx が入力であるような場合には KLAX 同様効果が現れ, xlsx では変化がなく, pptx の場合には多少劣化してしまうという結果になった.

6.4 実行時間

表 11 に Office 文書数とその総容量, 表 12 に KLAX と LAX&KEY の実行時間を示す. pptx は画像ファイルが多く含まれているため他よりも総容量が大きくなっている.

表 13 に KLAX での重み比に対する実行時間, 表 14 に LAX&KEY での重み比に対する実行時間をそれぞれ示す.

表 10　テストケースの入力パターンに対する LAX&KEY 導入後の推薦と評価結果の変化

テストケースの入力パターン	変化
doc doc	F 値が上昇した (3 件)
doc doc	F 値が下がってしまった
docx docx	F 値の低い結果が推薦不可になった（ノイズの除去）
docx docx	新しく推薦可能になった
doc pptx	F 値が上昇した
xlsx doc	新しく推薦可能になった
ppt ppt	F 値が上昇した
pptx pptx	F 値が上昇した
pptx pptx	F 値の低い結果が推薦不可になった（ノイズの除去）
pptx pptx	F 値が下がってしまった

推薦システムにおける推薦にかかる時間は, 推定可能なタスクの数によって大きく異なる. これは推定したタスクによって推定するワークフローのスコアリング計算の回数が大きく変化するためである. また, c を高くしていくにつれて徐々に計算時間が減少していく傾向が見られているが, これはテストケースの入力パターンに対する推薦と評価結果の変化でわかったように, 提案手法導入後の方がノイズとなるような推定タスクが少なくなったため, その結果推定, 推薦にかかる計算時間が削減できていると考えられる.

表 11　office 文書数とその総容量

種類	数	容量
docx	32	8.5MB
xlsx	30	9.6MB
pptx	32	127.4MB

表 12　KLAX と LAX&KEY の実行時間

計算手法	ファイル種類	時間 (sec)
KLAX	docx	4922.143
KLAX	xlsx	446.736
KLAX	pptx	2589.649
LAX&KEY	docx	40.824
LAX&KEY	xlsx	11.715
LAX&KEY	pptx	136.133

表 13　KLAX での重み比に対する実行時間 (秒)

(a,b)\c	0	1	2	3
(1,0)	3712	1144	441	414
(1,1)	1126	468	474	501
(1,2)	777	400	357	425
(1,3)	456	363	431	427
(2,0)	3712	2292	1144	1580
(2,1)	2711	1021	1594	453
(2,2)	1126	371	1468	271
(2,3)	1503	351	290	395
(3,0)	3712	8229	1587	1144
(3,1)	3979	2234	1000	1219
(3,2)	1878	1309	641	1383
(3,3)	1126	406	1500	468

表 14　LAX&KEY での重み比に対する実行時間 (秒)

(a,b)\c	0	1	2	3
(1,0)	3712	1015	451	428
(1,1)	1126	561	468	418
(1,2)	777	377	343	415
(1,3)	456	339	405	418
(2,0)	3712	2573	1015	927
(2,1)	2711	1316	905	685
(2,2)	1126	972	561	267
(2,3)	1503	733	270	390
(3,0)	3712	4458	1760	1015
(3,1)	3979	1891	1115	670
(3,2)	1878	1350	716	361
(3,3)	1126	412	1650	561

7. まとめと今後の課題

7.1 ま と め

本研究では, 既存の推薦システム [12] における抽象化する際のファイル間類似度計算において, 新たに複数の XML で構成される office ファイル間類似度の導入を行った. 既存のファイル類似度計算手法との適切な重みパラメータを求める実験を行い, また導入における効果の検証実験を行った.

既存の研究では, クラスタリングの際にファイル名とコピー関係を用いた類似度計算を行っていたが, 複数の XML ファイルで構成される Office 文書を対象にし XML 類似度手法を新たに組み込むことにより, ファイルのフォント, 配置, サイズなどのスタイル情報や構造にフォーカスした類似度計算が可能となる. 結論としては, 今回提案した XML ファイル間類似度をファイル名とコピー関係と組み合わせて新たに導入することで推薦結果の精度を向上させることが可能であることがわかった.

提案手法では, 特にファイル名のような見た目の類似度ではなくファイルの構造に着目した類似度であったため, 既存の類似度と差別化して新たにファイル間類似度の尺度を定義でき, 精度向上につながった. XML 類似度計算手法としては KLAX, LAX&KEY の 2 つについて実験を行い, よりファイル内のキーワードの類似度に着目した KLAX の方が評価結果として良い結果が得られた.

7.2 今後の課題

ファイル名を用いた類似度はすべてのファイル間において定義可能であるが, 今回導入した XML ファイル間類似度は OOXML 形式の office ファイル間のみであるため, 対象となるファイルが限られる. 今回用いたログデータは研究室内でのファイルアクセスログであるため, office ファイルの利用は多くはなく実験データとして適していない部分がある. そのため office ファイルの利用頻度の高い, 企業のオフィス等における事務作業のアクセスログを用いて実験することが必要であると考えられる.

8. 謝　　辞

本研究の一部は, 科研 25240014 の助成により行われた. また本研究で東京工業大学横田研究室内のファイルアクセスログを用いることに関して, 東京工業大学の人を対象とする研究倫理審査委員会の承認を得ている. 本論文の作成にあたり, 終始適切な助言を賜り丁寧に指導していただいた横田教授始め, 関係者の方々に深く感謝する.

文　　献

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

[2] V.I.Levenshtein, “BINARY CODES CAPABLE OF CORRECTING DELETIONS,INSERTIONS,AND REVERSALS”, (Presented by Academician P.S.Novikov January 4, 1965) Translated from Doklady Akademii Nauk SSSR, Vol. 163, No.4,pp.845-848,August,1965 ,Original article submitted January 2,1965

[3] K.C.TAI. “The Tree-toTree Correction Problem”, Journal of the Association for Computing Machinery Vol 26, No 3, July 1979

[4] Greg Linden, Brent Smith, and Jeremy York, “Amazon.com recommendations: Item-to-item collaborative filtering”, IEEE Internet Computing, Vol. 7, No. 1, pp. 7680, 2003.

[5] Stanford Log-linear Part-Of-Speech Tagger. http://nlp.stanford.edu/software/tagger.shtml

[6] Part Of Speech Tagging PHP/ir. http://phpir.com/part-of-speech-tagging

[7] Xinyou Yin, Jan Goudriaan, Egbert A. Lantinga, Jan Vos, and Huub J. Spiertz. “A flexible sigmoid function of determinate growth”, Annals of Botany, Vol. 91, No. 3, pp. 361371, 2003.

[8] Ricardo Baeza-Yates and Berthier Ribeiro-Neto, “Information Retrieval”, Addison Wesley, 1999.

[9] W. Liang and H. Yokota, “LAX: An Efficient Approximate XML Join Based on Clustered Leaf Nodes for XML Data Integration”, Proc. BNCOD, Springer LNCS 3567, pp. 82–97, 2005.

[10] 上垣外英剛, 渡辺陽介, 横田治夫, “Office XML 文書の部分木に着目した類似スタイルのファイル検索”, 第 4 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.

[11] Apichaya Auvattanasombat, 渡辺 陽介, 横田 治夫, “An Evaluation of Similarity Search Methods Blending Structures and Keywords in XML Documents”, Proc. The 15th International Conference on Information Integration and Web-based Applications Services (iiWAS2013) ,2013.

[12] 宋強, 川端貴幸, 伊藤史朗, 渡辺陽介, 横田治夫, “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”, 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 (DEIM2013), 2013.