3N-8

大規模 XML データにおける効率的な重複データ検出

小柳涼介 † 天笠俊之 ‡ 北川博之 ‡

† 筑波大学情報学群情報科学類 ‡ 筑波大学システム情報系情報工学域

1 はじめに

Extensible Markup Language (XML) [1] は現在，様々な用途に広く使われている．しかし同時に，内容が類似している情報の散在がしばしば見かけられる．このような類似した複数の情報を見つけ出すことができれば，情報の相互補完，重複部分の除去，コピーの検出等，様々な用途に活用できる．特に，与えられた XML データ (クエリ XML) に対して，データベース中の XML データ内において類似したすべての部分木を列挙する問題を類似部分木探索問題と呼び，これまでにいくつかの手法が提案されている [1, 2]．

XML は半構造データであり，テキスト情報に加えて，データ自体が木構造に基づく構造を持つという特徴を持つ．XML データの構造を考慮した類似度計算手法として代表的な物に，XML データを木構造とみなして木編集距離 [3] を適用する方法がある．しかし一般に木編集距離の計算は時間計算量が高コストであるため，巨大な XML ファイルを対象とした場合，効率的な計算手法が必要不可欠となる．

類似部分木探索問題に対する既存手法である TASM [1] や Structure Search[2] では，テキストノードについて，値の完全一致だけを評価しているため，テキストの類似性を考慮した評価は行っていない．テキストノードの値の類似度を調べることで XML データの類似度が概算ができ，処理の効率化を実現できる可能性がある．

本稿では，類似部分木探索問題に対して Jaccard 係数に基づくテキストを考慮した類似度と，木編集距離 (TED) に基づく構造を考慮した類似度を混ぜあわせた類似度を考え，巨大な XML データを処理する事を想定した効率的な類似度計算アルゴリズムを提案する．

Efficient Duplicate-detection in Large XML Data
Ryosuke KOYANAGI†(rkoya@kde.cs.tsukuba.ac.jp),
Toshiyuki AMAGASA‡(amagasa@cs.tsukuba.ac.jp) and
Hiroyuki KITAGAWA‡(kitagawa@cs.tsukuba.ac.jp)
†College of Information Sciences, University of Tsukuba
‡Faculty of Engineering, Information and Systems, University of Tsukuba

[1] http://www.w3.org/XML/

2 提案手法

2.1 問題定義

本手法では，XML データ D に含まれる任意の部分木 D_i とクエリ Q 間の構造類似度，テキスト類似度を以下のように定める．

$$Sim_s(D_i, Q) = 1 - \frac{min(TED(D_i, Q), |Q|)}{|Q|} \quad (1)$$

$$Sim_t(D_i, Q) = Jaccard(W_{D_i}, W_Q) \quad (2)$$

ここで，D_i は D における後行順番号 (2.3 節) i 番目のノードを根とする部分木を表し，Q はクエリ XML を，$|Q|$ はクエリ XML に含まれるノード数を W_{D_i}, W_Q はそれぞれ D_i と Q が含む単語の集合を表す．

これら二つの類似度をパラメータ α で重み付けをし統合した値を XML データ D_i, Q 間の類似度とする．XML データ D における類似部分木探索問題とは，与えられた Q, θ, α に対して，以下の条件を満たす D 中の全ての部分木 D_i を列挙する問題である．

$$\alpha Sim_s(D_i, Q) + (1 - \alpha) Sim_t(D_i, Q) > \theta \quad (3)$$

この問題をナイーブに解こうとした場合，巨大な XML データを入力とすると膨大な計算コストおよび空間コストが必要になるという問題がある．これらの問題点を，以下のアプローチを取ることで解決する．

2.2 閾値による枝刈り

$|D_i|$ と $|Q|$ が著しく異なっている場合類似度も小さくなり，式 3 より，類似度の下限が決まっていることと Q が固定されていることから，予め類似条件を満たしうるような D_i のサイズの範囲を求められる．これにより，得られる結果を損なわず過剰に大きい部分木についての計算を無視することができる．

構造類似度とテキスト類似度それぞれについて必要な下限値を $minSim_s, minSim_t$ とした時，式 (1-3) から D_i, Q が類似条件を満たすならば式 (4, 5) が満たされる．

$$minSim_s |Q| \leq |D_i| \leq (2 - minSim_s)|Q| \quad (4)$$

$$minSim_t |W_Q| \leq |W_{D_i}| \leq \frac{|W_Q|}{minSim_t} \quad (5)$$

この条件により，ノード数か単語数が範囲外である部分木については類似条件を満たし得ないので類似度の計算を省く事ができる．

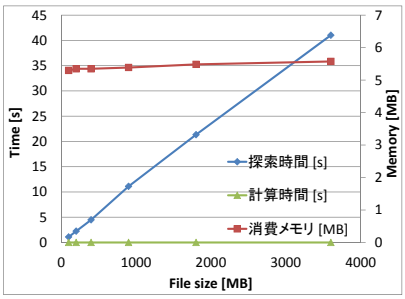

図 1: D の大きさの違いによる実験

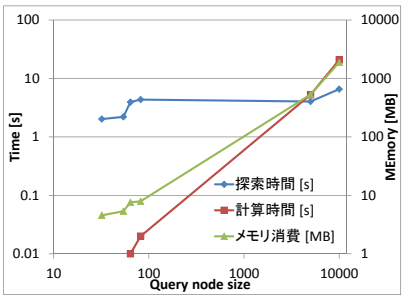

図 2: Q の大きさの違いによる実験

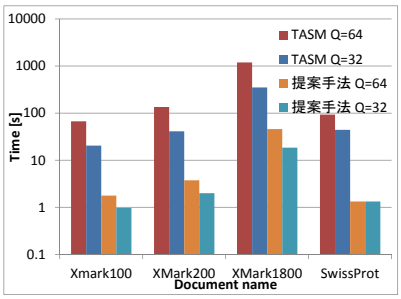

図 3: 既存手法との比較

2.3 Post-order Queue によるストリーム処理

木に含まれるノードを Post-order (後行順) で並べた時，任意の部分木は根ノードを末尾とする連続したノードのみによって構成される．また，式 (4) より類似度を計算しなければいけない部分木のノード数の上限が $\lfloor (2 - minSim_s)|Q| \rfloor$ 個であるため，対象となる部分木の類似度の計算のために保持しておかなければならないノードの数はクエリ XML サイズと閾値のみに依存することになる．これによりメモリに乗り切らないような巨大な XML データに対しても一度のディスクへのシーケンシャルアクセスで処理が遂行できる．

2.4 アルゴリズム

以上のアイデアを取り入れたアルゴリズムの大まかな流れは以下のようになる．

1. D の Post-order Queue から 1 ノード分読み込む
2. D_i のノード数と単語数が上限値以上ならスキップ
3. D_i の単語集合を構成し，テキスト類似度を求める
4. テキスト類似度が類似条件を満たし得るなら構造類似度を求める
5. 類似していれば部分木を結果リストに追加する
6. 以上の処理を全ノードに対し行う

3 評価実験

実際に XML ファイル使用し，木編集距離計算時間 (計算時間) とそれ以外の時間 (探索時間) を計測した．D として人工データである XMark ベンチマークと実データである SwissProt を使用した．なお，本実験は全て $\theta = 0.8, \alpha = 0.5$ としている．なお，すべての実験は 64bit Windows7, Intel Core i7 @ 3.20GHz CPU, 8GB RAM で構成されたラップトップ PC で実行され，プログラムは x86_64-w64-mingw32 でコンパイルを行った．

D のファイルサイズを変化させ実行した結果を図 1 に示す．探索時間はほぼ D の大きさに比例しており，消費メモリと計算時間は文書サイズによらず一定であった．

D に Xmark200 を使用し Q の大きさを変化させて実行した結果を図 2 に示す．Q が大きくなるにつれ計算時間，消費メモリは増加，探索時間は微量に増加した．

人工データ，実データの両方に対して $|Q| = 32, 64$ にて既存手法の TASM との実行時間の比較を行った結果を図 3 に示す．実行時間について全てのデータセットで既存手法を上回る性能を確認することができた．SwissProt では特に実行時間が短いが，これは XML データに含まれるテキストノードが持つ平均単語数が少ないからだと考えられる．

4 まとめ

テキストノードに対する柔軟な類似性判定と木構造に対する厳密な類似性判定を混ぜあわせた類似度を定義し，既存手法に比べ高度な類似性判定手法を提案した．また，提案手法に対し巨大な XML ファイルを対象とし，時間計算量，空間計算量の双方ともに効率的であるアルゴリズムを提案，実際の実験で人工データ，実データともに既存手法よりも短い計算時間で実行可能であることが確認できた．

今後の展望として，木編集距離や Jaccard 係数以外の類似度指標の適用，切り出した単語のふるい分け，Bloom Filter を利用した高速化，インデックス作成による高速化，処理の並列化への対応などが挙げられる．

謝辞

本研究の一部は JSPS 科研費 25330124 の助成を受けたものである．

参考文献

[1] Augsten, N., Barbosa, D., Bohlen, M. M. and Palpanas, T.: Efficient Top-k Approximate Subtree Matching in Small Memory, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 23, No. 8, pp. 1123–1137 (2011).

[2] Cohen, S.: Indexing for Subtree Similarity-search Using Edit Distance, in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pp. 49–60, New York, NY, USA (2013), ACM.

[3] Bille, P.: A Survey on Tree Edit Distance and Related Problems, *Theor. Comput. Sci.*, Vol. 337, No. 1-3, pp. 217–239 (2005).