Vol.2011-DBS-153 No.32
2011/11/3

XML 情報検索のための動的な索引管理手法の一提案

欅　惇志[†1]　宮崎　純[†1]　波多野　賢治[†2]
山本　豪志朗[†1]　加藤　博一[†1]

XML 情報検索では文書のうちの一部分，すなわち部分文書を単位とした情報検索を行う．これまでの研究では，ある時点までに蓄積された XML 文書集合 (XHTML で記述された Web 文書など) から索引を構築することで高速検索の実現を目指しているが，これらの研究ではデータの更新を考慮していない．新たに追加されたデータに対しても即座に検索対象として利用することができなければ，その時々の需要に合わせた情報検索を行うことができないが，既存の索引スキーマでは新たなデータに対して高速に索引語の重みを計算できないという問題が起こり，また，索引の更新には膨大なコストを要する．そこで本稿では，データ更新を考慮した XML 情報検索を実現するために，データの更新に適した索引スキーマを提案し，データ更新におけるコストを最小限に留めるための二種類のフィルタを提案する．

An Approach of Dynamic Maintenance of Indices for XML Element Retrieval

Atsushi Keyaki,[†1] Jun Miyazaki,[†1] Kenji Hatano,[†2]
Goshiro Yamamoto[†1] and Hirokazu Kato[†1]

In XML element retrieval, search engines return parts of XML documents as search results. Though existing studies create indices to search efficiently, they only use XML documents which are accumulated before the indices are generated and never consider updating of the indices. To cope with the dynamic users' information needs, we should update indices as new data arrives or existing data changes. However, we can not calculate term weights in new data with indices in existing studies because they are specialized in efficient query processing. Moreover, it is costly to update. Therefore, we propose a new schema for indices to achieve efficient query processing and updating. In addition, we propose two kinds of filters to reduce costs on update.

1. はじめに

現在，様々なデータが構造化文書の形式によって記述・作成されている．中でも特に Extensible Markup Language (XML) の利用が顕著であり，XHTML によって記述された Web 文書や，電子商取引サイトにおける商品データ，オフィスアプリケーションのファイルフォーマットなど，多くの場面において用いられている．

大量の XML データの出現に伴い，これらのデータからユーザが必要とする情報を取り出す際に検索技術を効果的に利用することは必然といえる．その際，XML データは大別して二種類に分類され，それぞれの XML データの特性に合わせた検索のアプローチが存在する．二種類の XML データとは即ち，DBLP データのようにテキストノード中に名詞もしくは名詞句が一つ含まれるようなデータ指向 XML と，XHTML にて記述された Web 文書のようにテキストノードに自然文が含まれるような文書指向 XML である[1)]．データ指向 XML に対する検索では主に LCA[2)], SLCA[3)], VLCA[4)] などの，特定の条件を満たすノード[⋆1]を検索結果として提示することを目指している．それに対して，文書指向 XML に対する検索とは，文書検索における情報検索技術を XML 文書に適用したものであり，単にクエリキーワードの出現箇所を発見するのではなく，ユーザの情報要求を満たす部分を選別して提示することを目指している．なお，本研究で取り扱う対象の XML データは文書指向 XML であり，以下は全て文書指向 XML に関しての議論である．

文書指向 XML に対する XML 情報検索ではこれまで，検索精度の向上を目指した研究[5),6),7)] と検索速度の向上を目指した研究[8),9),10)] が取り組まれてきた．情報検索システムの目的はユーザの要求に合致した情報を提示することであるために高精度検索を目指す必要があることは言うに及ばない．また，XML 情報検索ではその性質上，一つの XML 文書から複数の検索対象 (部分文書) が取り出され[⋆2]，検索に要する時間は文書検索と比較して長時間に及ぶため，高速検索を実現することも重要な課題である．また，これらの両立を

†1 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
†2 同志社大学文化情報学部
Faculty of Culture and Information Science, Doshisha University

⋆1 XML データを木構造と見立てた際に，自身を根とする完全部分木中に全てのクエリキーワードを含むようなノードや，それらのノードのうち更に深さや経路上のタグ名に制約を持たせたノード，などが該当する．

⋆2 XML 情報検索用のテストコレクションである INEX Wikipedia collection[11)] では，一つの XML 文書に約平均 67 個の部分文書が含まれる．部分文書については 2 節で詳述する．

目指した研究[8),9)] も存在するが，実運用を想定した XML 情報検索システムを目指す上ではこれらの目標だけでは十分ではない．なぜならば，既存研究ではデータの更新は考慮していないためである．現実世界では常に新しいデータが生み出され続け，特に Wikipedia のように複数人での編集を前提としたコンテンツではデータの書き換えが頻繁に発生する．これらのデータに対しても即座に検索対象として利用することができなければ，その時々の需要に合わせた情報検索を行うことができない．そこで本稿では，データ更新を考慮した XML 情報検索の実現を目指す．

更新を考慮した XML 情報検索を実現するためには，以下の二つの手順，即ち，(1) 新規データ中に含まれる索引語の重みを算出する，(2) 算出された索引語の重みを索引に登録する，を踏む必要がある．(1) に関して，索引語の重みを算出するためには文書集合全体から算出される統計量を用いるのが一般的である[5)]．しかしながら，従来の研究では高速な検索を実現するため，検索結果を提示する際に不要な統計量はデータベースへ格納しておらず，新たに追加された XML 文書中の索引語の重みを即座に算出することができない．従って，データの更新に対応するためには，予め索引語の重み計算に必要な統計量をデータベースへ登録し，高速な検索を維持しつつ即座に新規索引語の重み計算を可能とする索引スキーマの提案を行う必要がある．また，XML 情報検索では元の XML 文書数 (含まれる索引語数) と比較して遥かに多い個数の部分文書 (索引語) が取り出されるために，存在する全ての索引語の重みを算出するためには膨大なコストを費やす必要がある．そこで，データ更新におけるコストを最小限に留めるために **Element Filter** を設け，検索結果として提示される可能性の低い部分文書に含まれる索引語の重みの計算は行わない．そして，(2) では文書検索における索引再構築に関する関連研究で得られた知見をもとに，XML 情報検索への拡張を行う．その際，全ての索引語の重みを索引へ登録すると索引更新に膨大な時間を費やす必要がある．そのため，クエリ処理の特定を踏まえて全ての索引語を登録するのではなく，更新による効果が見込まれる索引語のみを更新すべく **Token Filter** の提案を行う．これにより，更新コストを最低限に抑えつつ，高精度 · 高速な XML 情報検索の実現を目指す．

2. XML 情報文書検索に関する基本事項

本節では，文書検索に対する XML 情報検索の位置づけについて述べ，更に部分文書の概要と XML 情報検索の特徴の一つである部分文書間の重複関係について述べる．

2.1 文書検索と XML 情報検索の比較

XML 情報検索と，一般的な Web 検索システムなどを初めとした文書検索の違いについて説明する．

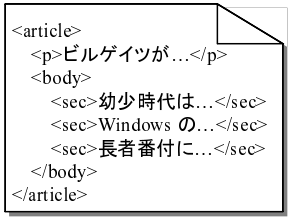

図 1 XML 文書
Fig. 1 XML Document

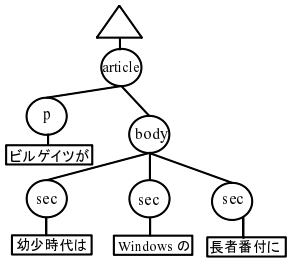

図 2 XML 木
Fig. 2 XML Tree

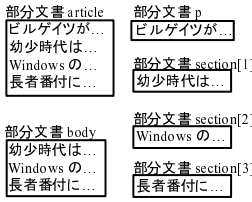

図 3 部分文書
Fig. 3 XML Fragment

多くの Web 検索システムはクエリに適合する文書のリストを提示する際に，スニペット[12)] と呼ばれる 150 文字前後の要約文も併せて提示する場合が多い．スニペットはクエリキーワードとその周辺のテキストを抽出する技術であり，検索結果中からいずれの文書が閲覧するのに適切であるのかをシステム利用者が判断するための要約文である．多くの検索システム利用者がスニペットを利用している反面，文章の文脈を考慮しないために理解不能なスニペットが生成される可能性があるため，必ずしも満足な結果が得られているわけではない[13)]．このことから，スニペットのみから情報要求を満たすことはできず，結局のところ文書検索システム利用者は文書を閲覧し，必要な箇所を自ら発見しなければならない．

これに対して，XML 情報検索における最大の関心は，クエリに適合する箇所，即ち部分文書を抽出し，それらに順位付けを行い提示することである．多くの Web 検索システムがクエリに適合する文書のリストを提示するのに対して，XML 検索システムはクエリに適合する部分そのもののリストを提示することができる．これにより，ユーザは文書中から情報要求を満たす部分を発見する必要がなくなるために，情報検索を行う際のユーザの負担を大きく軽減することができる．

2.2 部分文書とその重複関係

1 節で述べた部分文書の概要と重複について説明するために，図 1〜3 を用いて具体例を示す． まず，図 1 は XML 文書の例であり，図 2 は図 1 の XML 文書を木構造で表現した図である．XML 文書をはじめとする構造化文書は一般的に木構造で表現することができ，文書構造の視認性の向上を目的として度々木構造で表現される．本稿においても同様に，適宜 XML 文書を木構造と見立てて議論を進めることとする．このとき，XML 文書のそれぞ

れの開始タグと終了タグが XML 木の各ノード名に対応しており，タグの入れ子はノードの親子関係によって表現されている．図 3 の各部分文書は，図 2 の XML 木の各ノード以下に含まれる全てのテキストノードを結合した文字列と対応する．つまり，文書全体を表す `article` ノードと対応する部分文書は子孫に存在するテキストノード全てを結合した文字列であり，`body` ノードと対応する部分文書はその子ノードである三つの `section` ノードに含まれるそれぞれのテキストノードを結合した文字列である．包含関係 (先祖 · 子孫関係) を持つ部分文書間においてテキストの重複が発生するのはこのためである．

このとき，仮に情報要求を満たす内容が「幼少時代は…」と「Windows の…」,「長者番付に…」であった場合には，ユーザに対して `body` ノード以下の部分を提示することが適切であり，最も有益な検索結果である．

3. 関 連 研 究

3.1 XML 情報検索に関する関連研究

ここでは，XML 情報検索における二つの目標である高精度検索と高速検索に関して，それぞれ 3.1.1 節，3.1.2 節にて関連研究の説明を行う．

3.1.1 高精度な XML 情報検索に関する関連研究

XML 情報検索に用いられる検索技術は，文書単位を検索粒度としたテキスト文書に対する情報検索技術を XML 情報検索用に拡張して用いられることが多い．例えば，代表的な XML 情報検索用スコアリング手法である TF-IPF[14] は，ベクトル空間モデルの文書検索技術である TF-IDF[15] を XML の経路式を考慮し拡張させたスコアリング手法である．同様に，確率モデルに基づいた文書検索用スコアリング手法である Okapi's BM25[16] を構造化文書検索用に拡張させた BM25F[17] や，XML 情報検索 (部分文書検索) 用に拡張させた BM25E[5] なども存在する．現在の XML 情報検索に関する研究では，TF-IPF や BM25F (BM25E) をベースに拡張された検索技術が主流である．

上記のような，クエリに含まれる各索引語の重みを利用して検索結果を発見するアプローチ以外には，クエリによって指定される構造情報を用いたスコアリング手法[6] や，索引語の重みだけでなく各部分文書の持つ情報を考慮して検索結果として有用な検索結果である部分文書を特定するというアプローチ[7] も存在する．

また，18) では予め検索対象から解として不適切な部分文書を省くことを目指している．解として不適切な部分文書とは，如何なるクエリに対しても解とはならない部分文書のことであると定義しており，これらの部分文書を検索対象から除外したとしても検索精度が悪化することはなく，更には検索対象を軽減させることで検索時間の短縮にも繋がると考えている．これら不要部分文書の要件としては (1) 極端に小さな XML 部分文書，もしくは，(2) 極端に大きな部分文書または XML 文書全体である，としており，これらを特定する際に，XML 文書の統計量を利用することで不要な部分文書のテキストサイズの閾値を設定する手法を提案している．評価実験の結果，不要と判定された部分文書を検索対象から省くことによって，検索精度と検索速度双方の向上が可能であるということが明らかになった．

3.1.2 高速な XML 情報検索に関する関連研究

高速な XML 情報検索を実現するために，クエリ処理に必要最低限の情報のみをデータベースへ登録することや，Top-k アルゴリズムによる高速なクエリ処理を行われるなどによって取り組まれている．高速な XML 情報検索を目指す研究では，5), 17) といった索引語の重み付け手法を利用することで，高精度かつ高速な XML 情報検索の実現を目指しているものも存在する[8),9)]．索引語の重みは様々な統計量を用いた複雑な数式によって算出されるが，これはユーザからクエリが投げかけられる以前の事前処理段階で算出しておくことで，検索速度を低下させることなく高精度検索を実現することが可能である．

TopX[8] では，高速な検索実現を目的として Tag–Term 索引と Tag 索引の二種類の索引を構築している．Tag–Term 索引はタグと索引語のペアに対して一つのリストが作成されており，各部分書に対する得点計算に用いられる．また，リスト中の各エントリとしてノード識別子 (文書番号 + ノード番号) と対応する索引語の重みが格納されており，索引語の重みの降順に予めソートを行う．Tag 索引は，各部分文書がクエリによって指定される構造を満たすのかどうかを確認する用途で用いられ，タグごとに定義されるリスト中の各エントリとしては，タグを満たす部分文書のノード識別子が列挙されている．

また，効率的にクエリ処理を行い検索結果を提示するために，Top-k アルゴリズムのうちの一つである Threshold Algorithm[19] (TA) が利用されている．TA の概要を説明すると，各部分文書の得点を算出するに当たって，何らかの値 (TopX では索引語の重み) によってソートされた複数のリスト (TopX ではクエリを Tag–Term に分割し，該当する全ての Tag–Term ペアのリスト) に対して順番に，二種類のアクセス方法によってデータの読み込みを行う．一つ目はリストを降順に走査してデータを読み込むシーケンシャル · アクセス (SA) であり，二つ目はリストの任意のデータを探索してデータを読み込むランダム · アクセス (RA) である．なお，SA は RA よりも低コストであるため，原則 SA を行い，必要に応じて RA が行われる．読み込まれたデータから，何れかの部分文書が持ちうる最大の値と，それぞれの部分文書が持つ最低限の値を求めつつ，閾値を越える得点を持つと判明し

た部分文書を検索結果として確定させるアルゴリズムである．なお，クエリ処理時は主にTag–Term 索引によって得点計算が行われ，Tag 索引は補助的に用いられるため，Tag 索引中のリストへは RA のみ行われる．

上記の研究で構築する索引スキーマは高速なクエリ処理を行うことに特化したスキーマであるために，必ずしもデータ更新を行う際に適切であるとはいえない．データの更新が発生した場合には，新たに追加された索引語の重みを算出する必要が出てくるが，20), 5) などの代表的なスコアリング手法で用いられる統計量は，(1) 計算対象データから算出できる統計量，(2) 文書集合全体から算出される統計量，(3) パラメータや定数，の三種類に分類される．その際，(1), (3) は予め算出する必要はないが，(2) に関しては予め算出して格納しておかなければ，その都度文書集合全体を走査することとなり，高速な索引更新，ついては高速な情報検索は不可能である．従って，本研究では高速なクエリ処理とデータの更新両方を目的とした索引スキーマを考案する必要がある．

3.2 索引再構築に関する関連研究

ここでは，文書検索において取り組まれてきた索引再構築に関する関連研究について述べる．索引の再構築の際には以下の三種類の方式が存在する[21)]．

re-build 方式 re-build 方式は何れかのタイミングで，その時点での最新の文書集合を用いて索引を丸ごと作成し直すことで索引を最新に保つ方針である．従って索引構築中には検索システムを休止させる，もしくは，常時運用サービスであればサービス用の索引と新データの索引の二つを保持する必要がある．索引の一部更新は不要となるために複雑な処理は必要ないというメリットはあるものの，大量データを対象とした検索システムは索引の構築に長時間を費やす必要があるために不向きであり，サービス休止時間の発生もしくはディスク使用量の増大が起こるというデメリットが存在する．

re-merge 方式 re-merge 方式では，新たなデータが追加された際には索引の一部を動的に部分更新する．その際，新しく追加されたデータから作成したメモリ上のリストと，既存のディスク上のリストを統合し，ディスク上の別の場所に最新のリストを作成する．リストの統合後は何れともの古いリストを廃棄する．なお，メモリ上のリストとディスク上のリストの統合は，メモリ上のデータが一定以上溜まった時点で行われる．

in-place 方式 in-place 方式においても，新たなデータが追加されれば，索引の一部に対して動的な部分更新を行う．このとき，更新対象の各リストの末尾にデータの追記を行う．ただし，追記を繰り返す過程でリストの末尾に空きスペースがなくなった場合には，リスト長よりも空きスペースが十分な，ディスク上の別のスペースにリストを丸ごと移動させる．なお，データの追加が発生する度にリストの更新を行うのではなく，一定量のデータが蓄積した段階で纏めてリストの末尾にデータを書き込むことで，ディスクの I/O の回数を減らし，結果として更新コストを軽減させることが可能である．

また，in-place 方式と re-merge 方式を組み合わせて，効率的な索引の更新を目指す研究も取り組まれている[22),23)]．更に，索引更新に関する研究ではデータの追加に焦点を絞って議論している研究が中心ではあるが，24) ではデータの削除も踏まえた効率的な索引管理手法の提案を行なっている．これらのデータの追加と削除を組み合わせることで，本研究で想定しているデータの編集にも対応することが可能である．

なお，これらの索引再構築手法ではクエリ処理に TA を利用することを前提とはしていないために，リストに含まれるデータのソートを考慮していない．従って，上記の技術と TA を組み合わせる際にはリストの更新時にデータのソートを行う必要がある．

4. 提 案 手 法

2 節と 3 節を踏まえて，データの更新を考慮しつつ，高精度 · 高速な XML 情報検索を実現する上での課題と，その解決のために我々が行った取り組みについて述べる．

提案システムでは，既存の検索システムで考慮している索引構築，クエリ処理に加えて，データの追加に対する処理の定義を行う．以下に，提案システムのフレームワークを以下に示す．なお，索引構築，クエリ処理においては高精度 · 高速検索を満たす XML 情報検索システムである TopX[8)] をベースに拡張を行う．

(1) 初期索引構築 まず，初期索引の構築を行う (図 4(1))．高速にクエリ処理を行うことだけを目的とするのではないために，提案システムの索引スキーマでは従来手法では考慮されていなかった，索引語の重み計算に必要な統計量の格納も行う．

(2) クエリ処理 続いて，クエリ処理のパートではユーザのクエリを用いて索引に問い合わせて，検索結果を取り出してユーザへ提示する (図 4(2))．その際，XML 情報検索における高速なクエリ処理を行う際に多用される TA を用いる．

(3) データの追加 最後に，データの追加パートでは，新たに追加されたデータに対しても検索対象として扱うことができるように索引語の重み計算と索引への登録を行う (図 4(3))．また，索引の更新手法は文書検索における索引更新技術を参考にする．

以降，上記の処理を行う上で必要な，索引スキーマの定義とデータ追記時の処理について述べる．その際，部分文書単位の検索の特性である，元データ量と比較した場合に起こるデータ量の増大に対応するため，更新に関わるコストを削減するための処理の提案を行う．

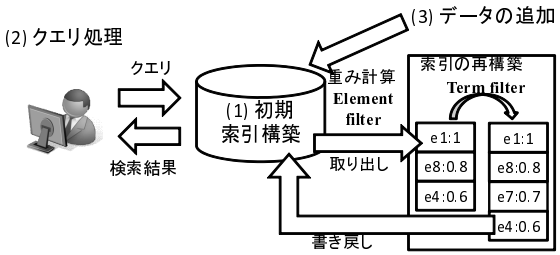

図 4 提案手法の概略図
Fig. 4 Overview of Proposed Method

4.1 索引スキーマの定義

ここでは，高速なデータ更新と高速なクエリ処理を両立させるための索引の提案を行う．本研究では TopX と同様に部分文書の得点計算用の索引と，構造制約の確認用の索引の二種類を構築するが，提案システムでは，Tag–Term 索引の代わりに Path–Term 索引を構築する．その理由として，過去の調査[25] の結果，索引語の出現するタグのみに着目するのではなく path 式を考慮することでより高精度な情報検索を行うことが可能であると判明したためである．しかしながら，検索精度が向上するというメリットが存在する一方で，計算量 · データ量ともに増大するというデメリットが発生する．これは，Tag–Term 索引ではある索引語の重みは，その索引語を含む何れの部分文書中でも同じ重みとして扱われるが，Path–Term 索引では同じ索引語であったとしても，出現する部分文書ごとにその重みを異なる基準で算出，登録する必要があることに起因する．結果として，パス式と索引語のペアごとに作成されるリストに登録される総データ数が大幅に増大する．従って，Path–Term 索引を用いる上では検索速度を向上させるために工夫を行うことが重要となる．また，Path 索引は TopX の Tag 索引と同様に構造の制約確認を目的として構築を行う．

それぞれの索引に格納される各エントリのデータとしては，TopX と同様に Path–Term 索引では識別子 (文書番号 + ノード番号) と索引語の重みであり，Path 索引では識別子である．索引語の重み算出には，高精度情報検索に適切であると言われるスコアリング手法である BM25E[5] を利用する．その際，ある部分文書 i における索引語 j の重み $w_{i,j}$ は以下の数式で算出される．

$$w_{i,j} = \frac{(k_1+1)tf_{i,j}}{k_1((1-b)+b\frac{el}{avel})+tf_{i,j}} \log \frac{N-df_j+0.5}{df_j+0.5} \quad (1)$$

ただし，$tf_{i,j}$ は部分文書 i に含まれる索引語 j の索引語頻度，df_j は同じ path 式で表される部分文書群中の索引語 j の部分文書頻度，N は同じ path 式で表される部分文書数，el は部分文書 i の部分文書長 (部分文書中に含まれる索引語数)，$avel$ は同じ path 式を持つ部分文書群の平均部分文書長，k_1, b はパラメータであり，過去の実験を踏まえて，それぞれ 2.5, 0.85 を設定する．

ここで，新たなデータが追加された際の索引語の重み算出について議論を行う．既存の索引中の索引語の重みと同様に，追加された索引語の重みの計算する際には BM25E による重み計算を行うが，$tf_{i,j}$, el などは新たなデータから算出される統計量であるために予め計算することはできず，k_1, b といった各種パラメータや定数は固定の値を用いるために索引語の重みを算出する際に参照することは容易である．その一方で，df_j, N, $avel$ といった統計量は文書集合全体から得られる値であるために，予めデータベースへ格納しておかなければ即座に索引語の重みを算出することはできない．故に，これらの統計量を Path–Term 索引と Path 索引に含まれるリストのヘッダとして登録する．

また，クエリ処理時には TA を用いるために，Path–Term 索引中ではシーケンシャル · アクセス (SA) 用とランダム · アクセス (RA) 用の二種類のリストを作成する必要がある．それに対して Path 索引は補助的に用いられるため，RA 用のリストのみで十分である．それぞれのリストの区別のため，$PathTerm_{seq}$ リスト，$PathTerm_{rand}$ リスト，$Path_{rand}$ リストと呼び分ける．なお，SA 用のリストは索引語の重みの降順によるソート，RA 用のリストは識別子の昇順によるソートを行うこととする．

これらを踏まえて，Path–Term 索引と Path 索引の概略図をそれぞれ図 5[⋆1]，図 6 に示す．

4.2 データの追加と，更新コストの削減

新しいデータが追加された際の手順について述べる．まずは追加されたデータに含まれるそれぞれの索引語に対して重みの算出を行い，その後該当リストへの登録を行う．

索引語の重み計算は 4.1 節の索引に含まれる情報を活用することで索引語の重みを計算可能であるが，部分文書単位での情報検索では，元の XML 文書数 (含まれる索引語数) と比較して遥かに多い個数の部分文書 (索引語) が取り出されるために，存在する全ての Path–Term ペアに対して索引語の重みを算出するためには膨大なコストを費やす必要がある．そこで，データ更新におけるコストを最小限に留めるために **Element Filter** を設け，検索結果として提示される可能性の低い部分文書に含まれる索引語の重みの計算は行わない．

⋆1 ヘッダ中に存在する Top-k_1 や Top-k_2 に関しては，4.2.2 節にて述べる．

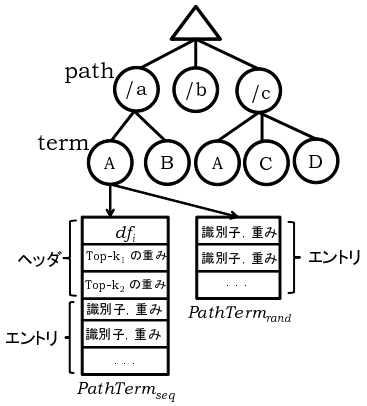

図 5 Path–Term 索引構成
Fig. 5 Path–Term index

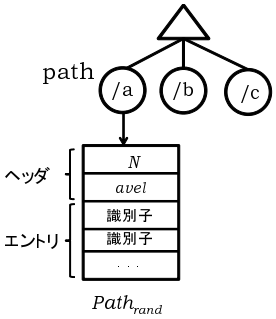

図 6 Path 索引構成
Fig. 6 Path index

そして，その後の索引更新では文書検索における索引再構築に関する関連研究で得られた知見を利用する．提案システムのプロトタイプでは 21) で提案されている三種類の方式のうち，単独で用いた場合のコストが総じて低い結果が得られた re-merge 方式を採用することとする．また，このとき，全ての索引語の重みを索引へ登録すると，索引更新に膨大な時間を費やす必要があることが予想される．そのため，全ての索引語を登録するのではなく，更新による効果が見込まれる索引語のみを更新すべく **Token Filter** の提案を行う．これにより，更新コストを最低限に抑えつつ，高精度 · 高速な XML 情報検索の実現を目指す．

4.2.1 Element Filter

検索結果として提示することが不適切な部分文書や提示される可能性が低い部分文書に対して索引語の重み算出を行ったとしても，その効果は薄い．従って，データが追加された時点でそれらの部分文書を特定し，索引語の重み計算を行わないことで更新における計算コストを削減するというのが Element Filter のアイデアである．解にならない部分文書の条件としては，

(1) 索引語数の小さな部分文書
(2) 極めて複雑な path 式を持つ部分文書
(3) 解となりにくい path 式を持つ部分文書

などの条件を考慮する．(1), (2) は 18) や 7) において，解となり得ない部分文書の条件として報告されている．また，(3) 解となりにくい path 式の判断方法であるが，path 式ごとに語の重みの期待値が異なることを利用する．即ち，各 $PathTerm_{seq}$ リストの Top-k 件目の値から path 式そのものの期待値を計測することが可能であると考えている．これらの期待値に対して閾値を設定することで，高い索引語の重みを持つことが期待できない path 式を特定し，索引語の算出対象から除外する．

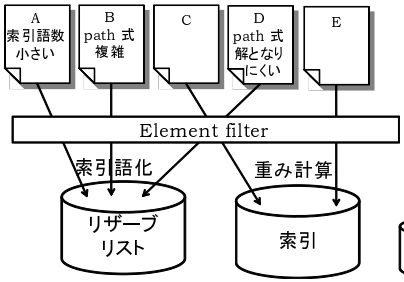

図 7 Element Filter による索引語の重み計算対象の分類
Fig. 7 Element Filter

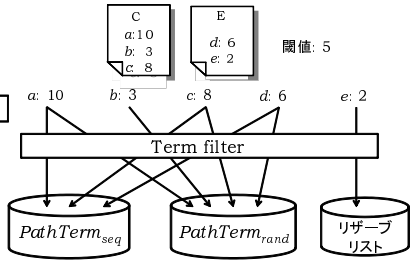

図 8 Term Filter による登録リスト分類
Fig. 8 Term Filter

また，重みの計算を行わなかったデータがその後必要になる可能性がある．例えば，あるデータが既に検索対象として考慮されているかどうか確認する必要がある場合や，索引中のリストに含まれるデータ数が k 件以下になってデータをリストへ補充する必要が出てきた場合などである．これらの事態に対応するために，計算対象外のデータを廃棄するのではなく，リザーブリストというデータ保持リストへ登録を行う．なお，その際には記号削除，不要語処理，接辞処理などの索引語化処理のみを行う．

これらを踏まえて，Element Filter の動作を図 7 に示すと，部分文書 A, B, D はそれぞれ Element Filter によって計算対象外と認められるためにリザーブリストへ登録され，部分文書 C, E に含まれる索引語は重み計算が行われる．

4.2.2 Term Filter

TA によるクエリ処理では，索引語の重みの降順にソートされた複数リストに対して SA を行い，補助的に RA を行うことで高速なクエリ処理を行う．つまり，提案システムにおいては原則 $PathTerm_{seq}$ リストを読み取ることで部分文書の得点の計算を行うが，$PathTerm_{seq}$ リストへのアクセスは SA しか行われないため，索引語の重みの小さなデータを登録しても読み取られる可能性が低い．これらのデータを $PathTerm_{seq}$ に登録しても効果が薄いこと

表 1 部分文書長
Table 1 Length of the elements

部分文書長	部分文書数	累積割合
1	14439812	0.33
2	12498711	0.61
3	4583937	0.72
4	2393175	0.77
5	1340885	0.80
10	299095	0.86
15	206232	0.89
20	125633	0.90
100	36319	0.97

表 2 path 式の深さ
Table 2 Depth of path expressions

パス式の深さ	部分文書数	累積割合
1	659387	0.015
2	1313934	0.040
3	6493011	0.19
4	11039412	0.44
5	11046938	0.69
6	8316062	0.88
7	3470587	0.96
10	91445	0.99
30	2993	1.00

表 3 Term Filter の効果の推定
Table 3 An estimation of Term Filter

初期索引割合	更新対象割合
20%	0.00041
50%	0.00060
80%	0.0012

が見込まれるため，$PathTerm_{rand}$ のみに登録すれば十分である．その際の分類は，Top-k 番目のデータの索引語の重みを閾値として利用する．これは，予め図 5 のヘッダへ Top-k 番目のデータの重みの値を登録しておくことで，その都度リストを走査することが不要である．なお，Top-k における k の値は必要に応じて複数の k を設定することが可能である．また，Path 索引に含まれる $Path_{rand}$ リストへも全ての該当データを登録する必要があるが，Path–Term ペアの個数と比較して Path の個数は非常に小さいために，更新コストを削減する上では Path–Term 索引への登録データの削減を取り組むことがより効果的である．

以上より，Term Filter の動作を図 8 に示す．$PathTerm_{seq}$ の登録に関する閾値が 5 であったとすると，索引語 a, c, d の重みは閾値よりも大きな値であるために $PathTerm_{seq}$ と $PathTerm_{rand}$ 両方へ登録される．また，索引語 b, e の重みは閾値よりも小さな値であるために $PathTerm_{rand}$ へ登録される．

4.2.3 二種類のフィルタに関する予備実験

コスト削減のために提案する二種類のフィルタに関して，その効果を見積もるための予備実験を行う．予備実験には INEX Wilkipedia Collection[11] を用いた．

Element Filter に関する予備実験について述べる．表 1 と表 2 はそれぞれ，文書集合から取り出された部分文書の部分文書長と path 式の深さである．過去の研究結果から部分文書長が 15 以下の部分文書は解として不適切である可能性が高いという傾向が判明しており，仮にそのまま適用した場合には 9 割近くの部分文書を検索対象から除外することが可能である[⋆1]．path 式の深さに関してはどの程度の深さにおいて有効であるのかについては調査が必要であるが，過去の手法[7] において得られた検索結果上位の結果に含まれる部分文書の持つ path 式の深さは，平均は 3.3 であったため，適切な値を設定することが出来れば計算対象の削減に結びつけることも可能であると考えられる．

続いて，Token Filter によって削減効果を見込めるかどうか知るために予備実験を行った．今回の予備実験では Top-k の k を 1,500 と設定し，全てのデータを登録した際のリスト長が k よりも十分に大きな 3,000 件以上を格納しているリストに対して実験を行った．リスト長の大きなリストのみを対象とした理由は，リスト長が k 件未満のデータでは Term Filter の対象外であり，データの値の大きさに関わらず更新回数を削減することができないためである．実験方法は，予め文書集合のうちの一部から初期索引を構築し，その後残りのデータを追加する際には Top-k のデータの値よりも大きな重みを持つ索引語のみリストへ登録する．表 3 の通り，初期索引の割合が 20%, 50%, 80% 何れの場合にも新たなデータが追加された際のリストの更新割合は極めて低くなった．今回の実験ではあくまでも，理想的な結果が得られる見込みのあるリストを対象としているため，実用の際には常にこれ程の効果が得られるとは限らないが，潜在的に高い効果を発揮することが期待できると判明した．

5. おわりに

データの追加，編集に対応するために，更新を考慮した XML 情報検索のフレームワークを提案した．その際，高精度 · 高速検索と両立するために索引スキーマの定義と，索引更新コストを最低限に留めるための二種類のフィルタの考案を行った．

今後の課題として，提案システムの実装と評価を行い，また，その結果を分析して Element Filter と Term Filter の最適化を行う．また，今回の提案ではデータ編集の際の手順として，該当データの全削除と新規データの追加を前提としているが，データの編集により適切な更新方法を考慮する必要がある．これは，データの編集が行われた影響範囲のデータのみを更新することで，全てのデータの更新を行う必要がなく，より高速な索引構築が可能であると考えられるためである．また，本稿では考慮していない，語の価値 (大域的重み) の大幅な変更への対応なども必要であると考えられる．

⋆1 ただし，検索対象部分文書の個数の削減であって，計算対象の索引語数の個数の削減割合ではない．

謝辞 本研究の一部は，科研費補助金基盤研究 (A)(課題番号:22240005) ならびに基盤研究 (C)(課題番号:2350012100) の支援による．ここに記して謝意を表す．

参 考 文 献

1) Blanken, H., Grabs, T., Schek, H.-J., Schenkel, R. and Weikum, G.: *Intelligent Search on XML Data: Applications, Languages, Models, Implementations, and Benchmarks*, Lecture Notes on Computer Science, Vol.2818, Springer-Verlag (2003).
2) Schmidt, A., Kersten, M. and Windhouwer, M.: Querying XML Documents Made Easy: Nearest Concept Queries, *Proceedings of the 17th International Conference on Data Engineering*, IEEE, p.321 (2001).
3) Xu, Y. and Papakonstaninou, Y.: Efficient Keyword Search for Smallest LCAs in XML Databases, *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ACM, pp.527–538 (2005).
4) Li, G., Feng, J., Wang, J. and Zhou, L.: Effective Keyword Search for Valuable LCAs over Xml Documents, *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, pp.31–40 (2007).
5) Liu, W., Robertson, S. and Macfarlane, A.: Field-Weighted XML Retrieval Based on BM25, *Advances in XML Information Retrieval and Evaluation*, Lecture Notes on Computer Science, Vol.3977, Springer Berlin, pp.161–171 (2006).
6) 波多野賢治，シーハムアメルヤヒア，ディベッシュスリバスタバ：XML 情報検索における構造問合せを利用した部分文書スコアリング，電子情報通信学会技術研究報告，Vol.254, No.107, pp.13–18 (2007). DE2007-117.
7) 欅 惇志，波多野賢治，宮崎 純：有益な検索結果提示のための部分文書再構成手法の提案，情報処理学会論文誌：データベース，Vol.4, No.1, pp.1–13 (2011).
8) Theobald, M., Bast, H., Majumdar, D., Schenkel, R. and Weikum, G.: TopX: Efficient and Versatile Top-k Query Processing for Semistructured Data, *Proceedings of the 31st Conference on Very Large Data Bases*, pp.625–636 (2005).
9) Trotman, A., Jia, X.-F. and Geva, S.: Fast and Effective Focused Retrieval, *INEX'09 Proceedings of the Focused retrieval and evaluation*, pp.229–241 (2009).
10) Shimizu, T. and Yoshikawa, M.: Full-Text and Structural Indexing of XML Documents on B+ Tree, *IEICE Transactions on Information and Systems*, Vol.E89-D, No.1, pp.237–247 (2006).
11) Kamps, J., Geva, S., Trotman, A., Woodley, A. and Koolen, M.: Overview of the INEX 2008 Ad Hoc Track, *Advances in Focused Retrieval*, Lecture Notes on Computer Science, Vol.5631, Springer Berlin, pp.1–28 (2008).
12) Manning, C.D., Raghavan, P. and Schutze, H.: *Introduction to Information Retrieval*, pp.157–159, Cambridge University Press (2008).
13) 中村聡史：情報信頼に対する信頼性調査および結果，人工知能学会誌， Vol.23, No.6, pp.767–774 (2008).
14) Grabs, T. and Schek, H.-J.: PowerDB-XML: A Platform for Data-Centric and Document-Centric XML Processing, *Proceedings of the First International XML Database Symposium*, Lecture Notes on Computer Science, Vol. 2824, Springer Berlin, pp.100–117 (2003).
15) Salton, G. and Buckley, C.: Term-Weighting Approaches in Automatic Text Retrieval, *Journal of Information Processing and Management*, Vol.24, No.5, pp.513–523 (1988).
16) Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M. and Gatford, M.: Okapi at TREC-3, *The Third Text REtrieval Conference (TREC-3)*, pp.109–126 (1995).
17) Robertson, S., Zaragoza, H. and Taylor, M.: Simple BM25 Extension to Multiple Weighted Fields, *Proceedings of the 13 ACM International Conference on Information and Knowledge Management*, pp.42–49 (2004).
18) 波多野賢治，絹谷弘子，吉川正俊，植村俊亮：XML 文書検索システムにおける文書内容の統計量を利用した検索対象部分文書の決定，電子情報通信学会論文誌，Vol.J89-D, No.3, pp.422–431 (2006).
19) Ilyas, I.F., Beskales, G. and Soliman, M.A.: A Survey of Top-k Query Processing Techniques in Relational Database Systems, *ACM Computing Surveys (CSUR)*, Vol.40, pp.1–58 (2008).
20) Liu, F., Yu, C., Meng, W. and Chowdhury, A.: Effective Keyword search in Relational Databases, *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, ACM, pp.563–574 (2006).
21) Lester, N., Zobel, J. and Williams, H. E.: In-Place versus Re-Build versus Re-Merge: Index Maintenance Strategies for Text Retrieval Systems, *Proceedings of the 27th Conference on Australasian Computer Science*, Vol.26, pp.15–23 (2004).
22) Büttcher, S. and Clarke, C. L. A.: A Hybrid Approach to Index Maintenance in Dynamic Text Retrieval Systems, *Proceedings of the 28th European Conference on Information Retrieval*, pp.229–240 (2006).
23) Margaritis, G. and Anastasiadis, S.V.: Low-Cost Management of Inverted Files for Online Full-Text Search, *Proceedings of the 18 th ACM Conference on Information and Knowledge Management (CIKM)*, pp.455–464 (2009).
24) Büttcher, S. and Clarke, C. L. A.: Indexing Time vs. Query Time Tradeoffs in Dynamic Information Retrieval Systems, *Proceedings of the 14th ACM Conference on Information and Knowledge Management (CIKM)*, pp.317–318 (2005).
25) Keyaki, A., Hatano, K. and Miyazaki, J.: Relaxed Global Term Weights for XML Element Search, *INEX 2010 Workshop Formal Proceedings*, LNCS, Vol.6932, pp. 71–81 (2011).