更新を考慮した XML 部分文書検索システムの精度の改善

欅　惇志[†,††]　宮崎　純[†]　波多野賢治[†††]　山本豪志朗[†]　武富　貴史[†]
加藤　博一[†]

† 奈良先端科学技術大学院大学情報科学研究科　〒 630–0192 奈良県生駒市高山町 8916–5
†† 日本学術振興会　〒 102–0083 東京都千代田区麹町 5–3–1
††† 同志社大学文化情報学部　〒 610–0394 京都府京田辺市多々羅都谷 1–3
E-mail: †{atsushi-ke,miyazaki,goshiro,takafumi-t,kato}@is.naist.jp, ††khatano@mail.doshisha.ac.jp

あらまし　本稿では，更新を考慮した XML 部分文書検索における，検索精度の改善に関する取り組みについて議論する．更新を考慮した XML 部分文書検索では，蓄積された文書量が十分でない時点での索引語の重み算出において，一部の統計量を正確に算出することができない．その結果，検索精度の低下を招くという問題が存在する．そこで本稿では，更新を考慮した XML 部分文書検索システムによって作成された検索結果のリストに対して，部分文書間の包含関係を考慮した部分文書統合による検索結果の再構成を行うことで，検索精度の向上を目指す．評価実験の結果，再構成手法を適用した場合に検索精度の向上が確認され，差分更新を行わずに新たな索引を再構築した場合と同程度の検索精度に達成することができた．更に，不十分な量の部分文書からも正確な統計量を近似的に算出することを目的として提案された手法と，検索結果の再構成手法を併用した場合には，新たな索引を再構築した場合よりも高精度検索を実現することが可能であることが確認された．

キーワード　XML 部分文書検索，文書の更新，検索結果再構築

An Improvement of XML Element Retrieval Considering Document Updates

Atsushi KEYAKI[†,††], Jun MIYAZAKI[†], Kenji HATANO[†††], Goshiro YAMAMOTO[†], Takafumi TAKETOMI[†], and Hirokazu KATO[†]

† Graduate School of Information Science, Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara, 630-0192 Japan
†† The Japan Society for the Promotion of Science, 5–3–1 Kojimachi
Chiyoda-ku, Tokyo 102–0083, Japan
††† Faculty of Culture and Information Science, Doshisha University
1-3 Tatara Miyakodani, Kyotanabe City, 610–0394 Japan
E-mail: †{atsushi-ke,miyazaki,goshiro,takafumi-t,kato}@is.naist.jp, ††khatano@mail.doshisha.ac.jp

1. はじめに

XML 部分文書検索の検索単位は XML 文書 (構造化文書) 中の一部分，即ち部分文書(注1) (element) である．そして XML 部分文書検索の目的は，文書中からユーザの求める情報を含む部分文書を特定して提示することである．そのため，ユーザは自らが求める情報が文書中のどの部分に記述されているのかを探す必要がなく，XML 部分文書検索システムを利用することで，検索におけるにユーザの負担を軽減することが可能である．

従来の XML 部分文書検索に関する研究においては，高精度検索に関する研究や高速検索に関する研究，もしくはそれらの両立を目指す研究が取り組まれているものの，これらの研究では固定の文書集合を対象としており，文書の更新は考慮していない．その一方で，XML 部分文書検索における検索対象の一

(注1)：部分文書については 2. 節にて後述する．

つである Web 文書 (XHTML 文書や wiki 文書をはじめとした構造化文書) は，文書の追加・削除・書換といった更新が頻繁に行われており，例えば英語版 Wikipedia では一時間に 4,000 から 8,000 回の記事更新が発生している(注2)．もし更新に対応しなかった場合には，Web 上の情報と検索システムの保持する情報に不一致が生じ，その結果検索システムの利便性が低下する．そこで我々は，高精度検索を維持しつつ，文書の更新を考慮した XML 部分文書検索手法の提案を行う．

その際，以下の二つの項目を目標として掲げる．

（1） 文書の更新を考慮した XML 部分文書検索システムの構築

（2） 文書の更新によって文書集合の持つ統計量が変化した場合にも有効な検索精度向上手法の適用

一つ目の目標に関して，我々は極力検索精度を低下させずに，速やかに文書の更新を反映させる XML 部分文書検索システムを構築した [14]．その中で，我々は

- 索引の高速な差分更新手法
- 正確な大域的重み推定手法

を提案した．既存の XML 部分文書検索システムでは，文書の更新が発生すれば，新たな索引(注3)を再構築していたが，検索システムに蓄積される文書数が増加するに連れて索引構築時間が長時間に及ぶことになる．そのため，我々は既存の XML 部分文書検索システムに差分更新機能を付与した．更に，差分更新コストを軽減すべく，高精度検索を実現する上で不要なデータを選定し更新対象から除外するフィルタを提案した．これにより，更新性能は向上し，検索精度や再現率の低下は確認されなかった．

また，検索システム運用初期などのように，検索システムに蓄積された文書数が少量の場合や，文書の更新に伴って新たなトピックが急激に出現した場合などにおいて，索引語算出時に利用される統計量のうち文書集合全体から算出される大域的重みを正確に算出することができないという問題が存在する．そこで我々は，文書の更新が発生した時点で利用可能な文書集合から正確な大域的重みを推定するための path 式統合手法を提案した．その結果，検索精度の低下が軽減された．

二つ目の目標に関して，前述の更新を考慮した XML 部分文書検索システムによって，文書更新の高速な反映を実現したものの，従来の XML 部分文書検索と比較して検索精度が低下した．そこで本稿では，更新を考慮した XML 部分文書検索システムの精度改善を目指す．その際，更新に伴って大域的重みが変化した場合にも有効な検索精度向上手法の適用が望ましい．

我々は過去に，各部分文書に付与されたスコア (クエリに対する適合度) とそれらの部分文書の包含関係を考慮した検索結果の再構成手法の提案を行った [13]．再構成手法では，文書中の最適な粒度の部分文書を特定し，有用な部分文書から順番にランキングを行なった上で検索結果として提示することを目的としており，文書の更新が発生しない固定の文書集合に対する評価実験の結果，適用することで検索精度の向上が確認された．これらの再構成手法は更新を考慮した XML 部分文書検索システムによって算出されたスコアに対しても適用可能である．そのため，更新を考慮した XML 部分文書検索システムにおいても，文書中の最適な粒度を発見し検索精度を向上させることが可能であるのかどうか，評価実験によって明らかにする．

(注2)：http://www.wikichecker.com/editrate/

(注3)：本稿における索引とは本論文における索引とは，検索システムに格納され，検索結果を提示する上で用いられるデータ群である．

2. XML 部分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 部分文書検索の位置づけについて述べ，更に部分文書検索の概要を説明する．

2.1 文書検索と XML 部分文書検索の比較

XML 部分文書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，それらをクエリに対する適合度の大きさで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自ら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可能である．

関連する技術として，スニペット [6] が存在する．多くの Web 検索システムは，検索結果として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．しかしながら，スニペットは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．実際に，インターネット利用者に対して行った Web 検索の信頼性に関する調査 [7] では，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ことと，その反面，必ずしも満足な結果が得られているわけではないということが報告されている．このことから，検索システム利用者は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．

その点，部分文書検索システムでは文書中の要点を一目で知ることができるため，ユーザにとって有用な技術となる．

2.2 部分文書と部分文書間の重複関係

部分文書の概要について説明するために，図 1〜図 3 を用いて具体例を示す．

まず，図 1 は文書番号 (DocumentID, DocID) 1 の 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．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，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 木の各要素ノードのノード名に対応しており，タグの入れ子は要素ノードの親子関係によって表現され

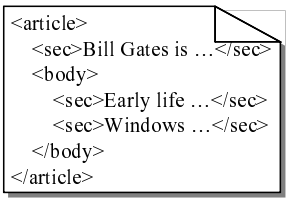

図 1　XML 文書

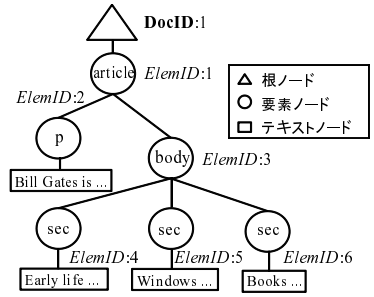

図 2　XML　木

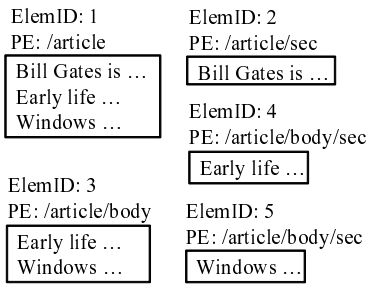

図 3　部 分 文 書

ている．また，各要素ノードに対して，文書順に部分文書番号 (ElementID, ElemID) を割り振る．図 3 の各部分文書は，図 2 の XML 木の各要素ノード以下に含まれる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ノードの重複が発生するのはこのためである．なお，ある要素ノード (部分文書) に着目した際に，先祖にあたる要素ノードを粒度の大きなノード，逆に子孫にあたる要素ノードを粒度の小さなノードと表現する．

仮にユーザが “Early life …”，“Windows…” に関する情報を求める場合，XML 部分文書検索では ElemID 3 の部分文書を提示する．これは，部分文書中にユーザが求める情報全てを含み，余分な情報を含まないためである．このように，ユーザの情報要求に対し必要十分な情報を提示することが XML 部分文書検索における高精度検索である．

2.3　XML 文書に対する検索

XML 検索において，ユーザの情報要求を表す際には，キーワードの指定と文書構造の指定を行うことが可能である．キーワードの指定のみを行うクエリは Content Only (CO) クエリ，キーワードと構造の両方を指定するクエリは Content And Structure (CAS) クエリと呼ばれる．

CO クエリは従来のテキスト文書検索と等価な問合せ方法であり，ユーザが XML 文書の文書構造に関する事前知識がない場合にも CO クエリであれば用いることが可能である．それに対して，CAS クエリは，XML 文書の最大の特徴のうちの一つである文書構造を利用しており，特定の粒度や内容を指定した問合せを行うことが可能である．

CO クエリと CAS クエリそれぞれの，XPath I (NEXI) [12] で表現された具体例を示す．CO クエリ:`//*[about(., "Bill")]` の検索結果の候補は，テキストノードに “Bill” を含む部分文書である．図 3 中では ElemID 1, 2 が該当する．

CAS クエリ:`//article[about(., "Gates")]//sec[about(., "Windows")]` は CO クエリより複雑である．まず前半の `//article` 以下に着目すると，1) テキストノードに “Gates” を含み，path 式の末尾が `article` で終わる部分文書を指定する．そして後半の`//sec` 以下に着目すると，2) テキストノードに “Windows” を含み，path 式の末尾が `sec` である部分文書を指定する．条件 1)，2) を組み合わせ，条件 1) を満たす部分文書を先祖に持つような，条件 2) を満たす部分文書が検索結果である．図 3 中では ElemID 5 のみクエリの制約を満たす．

2.4　部分文書検索における検索結果の提示方法

同一文書中の部分文書間には包含関係があるため，部分文書検索の枠組みでは検索結果として抽出される部分文書間で重複が発生する可能性がある．いくつかの先行研究では重複が発生することを考慮せずに，クエリ処理によって算出される部分文書の適合度に従ってランキングされた順位付きリストを提示していた．本論文では，このようなクエリ処理によって得られた，重複を排除するために特別な処理を行っていないリストを**重複リスト** と呼ぶこととする．そして，文献 [3] において重複リストを用いた場合の検索精度の低下が報告されて以来，多くの研究ではリスト中から重複を排除した**非重複リスト**を用いている(注4)．現存する最大の XML 検索のためのプロジェクトである INEX (INitiative for Evaluation of XML retrieval) project(注5) の XML 情報検索用のトラック全般 [2] においても非重複リストを使用している(注6)．

重複リストから非重複リストを作成する際にはいくつかのアプローチが採用されており，主なアプローチを以下に列挙する．

（1） 一文書からは，文書中で最もスコアの高い部分文書を一つのみ抽出する

（2） 部分文書のスコアの降順に，重複が起こらない限り複数の部分文書を抽出する

（3） 部分文書の統合による検索結果再構成する

（4） 文書全体を提示する

(1) では適合度の高い部分文書のみを取り出すことが可能である．ただし，一つの文書から一つの部分文書のみを取り出すために，当然ながら，抽出された部分文書以外の適合箇所は抽出することができない．

(2) では一つの文書から複数の部分文書の抽出を行うため，より多くの適合箇所の抽出を行うことが可能であるが，単に適合度の降順に，重複が起こらないように部分文書を抽出した場合には，適合箇所を取りこぼす可能性が存在する．図 4 を用い

(注4)：重複が発生していなければ，一つの XML 文書から複数の部分文書を抽出することは制限されていない．

(注5)：`http://www.inex.otago.ac.nz/`

(注6)：各クエリに対して上限 1,500 件まで，抽出すべき部分文書が存在する限り抽出し提示する．

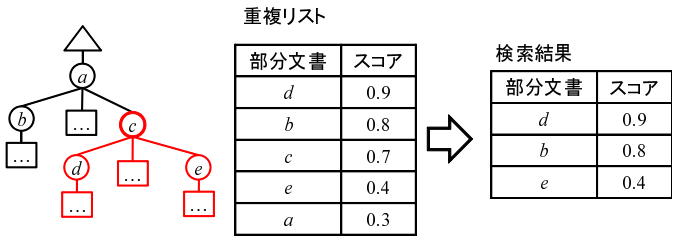

図 4　部分文書抽出

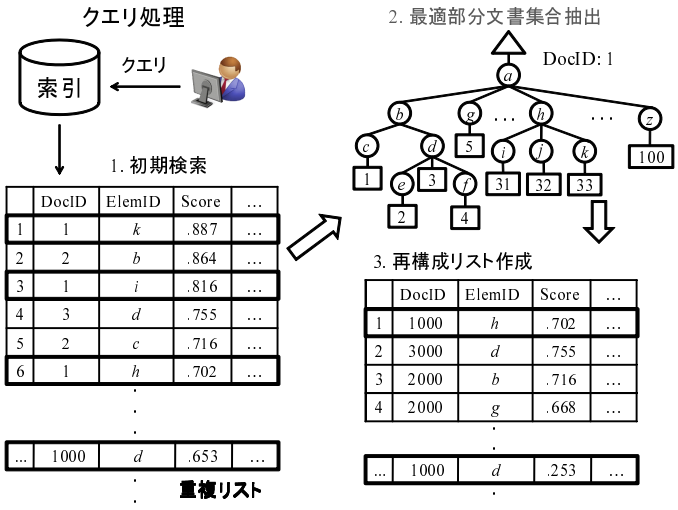

図 5　再構成リストの作成手順

て，適合箇所を取りこぼす具体例を示す．

このとき，赤く着色された，ノード c を根とする完全部分木と対応する部分文書 (以降，ノード x を根とする完全部分木と対応する部分文書のことを単に x と表現する) が適合部分であるとする．重複リスト中の部分文書を，スコアの降順に重複が生じないように抽出を行うと，まずは d が抽出される．続いて b が抽出され，その後に c の抽出が試みられるが，c は既に抽出されている d と重複が発生するために，c は抽出されずに破棄される．その結果，全ての適合部分を網羅して抽出できない．

そこで，我々は (3) の部分文書の統合による検索結果再構築手法の提案を行った [13]．各部分文書に付与された得点やテキストサイズ，更に部分文書間に存在する包含関係に着目して部分文書の統合を行い，検索結果の再構成を行う．詳細については次節にて述べる．

なお，(4) のアプローチでは適合箇所を網羅的に抽出することが可能であるものの，文書中のクエリと適合しない箇所も取り出す可能性がある．また，文書全体を検索結果として提示することとは，即ち，文書検索と同義であるため，部分文書検索の本来の趣旨からも外れ，適切なアプローチとは考えにくい．

3.　部分文書の統合による検索結再構成

部分文書の統合による検索結果再構成手法では，クエリに対する適合箇所を特定し，より情報要求に合致する部分文書を検索結果上位に提示することを目指す．具体的な処理手順を以下に示す．

処理 (1)　まず，部分文書検索技術を用いて各部分文書に対して初期検索を行い，初期検索スコアを計算する (図 5 (1))．この処理によって，重複リストを取得する．

処理 (2)　処理 (1) で得られた重複リストから，文書単位でクエリに対する適合部分文書を抽出する (図 5 (2))．この処理では，部分文書間の各スコアと重複を考慮した部分文書の統合を行う．以後この処理で作成される部分文書集合を**最適部分文書集合**と呼ぶ．

処理 (3)　処理 (2) で作成された最適部分文書集合をリスト形式で提示するために順位付けを行う．その際，有益な検索結果を上位に提示することを目的としたスコアリング手法を提案した (図 5 (3))．この手順で作成される，文書中の最適粒度の部分文書で構成され，検索結果が適切に順位付けされた非重複リストを**再構成リスト**呼ぶ．

以降，それぞれの処理について詳細に述べる．

3.1　重複リストの作成

クエリに対する適合度の降順に部分文書をランキングした重複リストを作成するにあたり，まずは全ての部分文書中の各索引語に対して重み付けをして，それらの重みを用いて部分文書の得点を算出する．

部分文書検索用の索引語の重み付け手法には TF-IPF [4] や BM25E [5]，部分文書検索用のクエリ尤度モデル [8] などが存在するが，本稿では部分文書検索においては最も高精度なスコアリング手法の一つとされる BM25E を用いる．

本稿において重複リストを作成する際には，文献 [14] にて提案した更新を考慮した XML 部分文書検索システムを用いる．詳細については 4. 節にて後述する．

3.2　最適部分文書集合の抽出

部分文書統合処理では，重複が発生した場合に，初期検索スコアの高低は考慮せず，単に粒度の大きな部分文書を抽出し，粒度の小さな部分文書は検索結果から破棄する．ここで再び図 4 を用いて部分文書統合処理の具体例を示す説明すると，先ほどと同様に，まずは d，続いて b が抽出される．その後，c の抽出が試みられるが，c は既に抽出された d と包含関係を持つ．従って，c を抽出し，d を除外する．

粒度の小さな部分文書は粒度の大きな部分文書の一部分から構成されるため，結果として全ての適合部分を抽出することが可能となる．しかしながら，部分文書の統合を繰り返す度に検索結果として抽出される部分文書の粒度が大きくなるため，最終的に文書全体が検索結果となりうるという問題がある．従って，大きすぎる粒度の部分文書が検索結果となることを抑制するため，各文書から抽出可能なテキストサイズ，即ち抽出制限を設定する．

テキストサイズの大きな文書ほど，様々な内容に関する記述を含むために，ユーザの情報要求に合致しない内容の記述が多くなると考えられる．従って，適合部分として提示する際に適切な部分文書のテキストサイズは一定のテキストサイズ以下であるとし，各文書に対して制限を設けることとする．なお，テキストサイズと索引語数には一定の相関があり，更に，各索引語の文字列長の影響を軽減するために，抽出制限は索引語数で制限を行う．

一つの XML 文書中から抽出できる索引語数を EL，文書番号が $DocID$ の XML 文書に含まれるテキストノードのうち最適部分文書集合に含まれるテキストノードの索引語数を τ_{DocID} とすると，最適部分文書集合を作成する際に，$\tau_{DocID} < EL$ を満たす限りは初期検索スコアの降順に部分文書の抽出を繰り返す．τ_{DocID} が EL を超える場合は抽出を行わず，部分文書を破棄する．

3.3 再構成リストの作成

最適部分文書集合の作成が完了すれば，最適部分文書集合に含まれる各部分文書に対してスコアを付与し，最終的にユーザに提示する再構成リストを作成する (図 5 (3))．その際，三種類のアプローチが存在する．

（1） 初期スコアによるランキング
（2） Bottom-Up スコアリング手法
（3） Top-Down スコアリング手法

(1) に関して，抽出された部分文書は重複リスト作成時に算出された初期スコアを持つ．初期スコアによるランキングではこれらの値の降順にランキングを行う．

初期スコアを用いることでランキングを行うことは可能であるが，初期検索スコアをそのまま利用するということは，従来の手法と同様，各部分文書を独立した文書として個別にスコアを計算していることと変わらない．我々が部分文書の再構成を適用した理由は，同一 XML 文書間に存在する関係を踏まえて検索結果を決定する必要があると考えたためである．このような経緯から，各部分文書と関連する部分文書の持つ統計量を考慮した，二種類の最適部分文書集合へのスコアリング手法を提案した．即ち，(2) 子孫部分文書の持つ統計量を先祖部分文書のスコア計算に用いる **Bottom-Up スコアリング手法**と，(3) 先祖部分文書の持つ統計量を子孫部分文書のスコア計算に用いる **Top-Down スコアリング手法**である．以降，それぞれの手法について詳述する．

3.3.1 Buttom-Up スコアリング手法

部分文書統合では，既に抽出されている部分文書と新たに抽出される部分文書の間に重複が発生した際に，子孫に当たる部分文書を最適部分文書集合から取り除き，代わりに先祖にあたる部分文書を挿入する．このとき，初期検索スコアを比較すると，先祖部分文書は子孫部分文書よりも低いスコアを持っていたことになる．そのため，検索結果をランキングする際に，初期検索スコアをそのまま用いれば，先祖部分文書は子孫部分文書が提示されるはずであった順位よりも下位で提示されることになる．つまり，元は高い順位で提示されるはずであった子孫部分文書は先祖部分文書の一部として提示されることになるが，その際に本来子孫部分文書が提示されはずであった順位よりも下位で提示されることになる．

我々はこのような高い初期検索スコアを付与された部分文書を子孫に持つ部分文書は，より上位の検索結果として提示されることが妥当であると考えた．そこで，Bottom-Up スコアリング手法では，子孫に高い初期検索スコアを持つ部分文書に対して，初期検索スコアよりも高いスコアを付与し，検索結果上位で提示されるようにする．その際，部分文書自身の持つ統計量と，子孫部分文書の持つ統計量の両方を考慮したスコアリングを行う．

以上の議論より，Bottom-Up スコアリング手法を以下のように定式化する．このとき，f_a を先祖部分文書，f_d を子孫部分文書のうち最も初期検索スコアの高い部分文書とする．

$$s_{bu}(f_a) = \frac{|f_d|}{|f_a|} \cdot s(f_d) + \frac{|f_a| - |f_d|}{|f_a|} \cdot s(f_a) \quad (1)$$

ただし，$s_{bu}(f_a)$ を f_a の再計算されたスコア，$|f_x|$ を f_x のテキストサイズ，$s(f_x)$ を f_x の初期検索におけるスコアとする．

3.3.2 Top-Down スコアリング手法

クエリキーワードの中には，語としてさまざまな意味を持つ索引語が存在する．語の持つ複数の意味から適切な意味を特定することは困難であるが，他のクエリキーワードとの共起情報は語の意味を特定する上で大きな手助けになる．つまり，もしある部分文書中に多くの種類のクエリキーワードが出現するのであれば，その部分文書中のクエリキーワードはクエリによって意図される本来の意味を持つと考えられる．このような考えのもと，我々は Top-Down スコアリング手法を提案した．

その際，部分文書中に出現するクエリキーワードの意図は，文書単位におけるクエリキーワードの意図を引き継いでいると考えられるために，有用性の判断においては文書単位のクエリキーワードの種類数を考慮する．これらを踏まえて，f をスコア計算される部分文書，D_f を f が含まれる文書とすると，Top-Down スコアは以下の数式で表現することができる．

$$s_{td}(f) = s(f) \cdot keyword(D_f) \quad (2)$$

ただし，$s_{td}(f)$ を f の Top-Down スコア，$s(f)$ を f の初期検索スコア，$keyword(D_f)$ を D_f に含まれるクエリキーワードの種類数とする．

4. 更新を考慮した XML 部分文書検索システム

重複リストを作成する際に利用する，更新を考慮した XML 部分文書検索システム [14] の説明を行う．

4.1 システムの機能と索引構造

更新を考慮しない従来の XML 部分文書検索システムが持つ機能は，データベース (索引) の構築機能と，クエリ処理機能である．それに対して，我々が提案した更新を考慮した XML 部分文書検索システムでは，更に文書の更新，即ち，文書の追加と削除，そして書換に伴って起こる索引の差分更新機能を持つ．

それに伴って，従来の更新を考慮しない XML 部分文書検索システムと我々が提案したシステムでは，システムが保持する索引の構成も異なる．提案システムの索引構造を図 6 に示す．従来システムでは，索引語の重みを格納した索引と (図 6 中の `Term` 索引と `Tag-term` 索引，`RA` 索引に相当)，構造を指定したクエリを処理する際に用いる索引 (`Path` 索引) の二つのタイプの索引を保持する．それに対して，提案したシステムは，文書の追加時に高速に索引の更新を行うことを実現するために，索引語の重み計算時に利用する統計量のうち，予め算出可能な統計量を索引に格納している (図 6 中の `Term` 索引と `Tag-term`

-Term (DocID, ElemID, 索引語, 索引語重み, **child_st**, **child_ed**, PathID, 部分文書長, VersionID)
-Tag-term (DocID, ElemID, タグ, 索引語, 索引語重み, **child_st**, **child_ed**, PathID, 部分文書長, VersionID)
-RA (DocID, ElemID, 索引語, 索引語重み, VersionID)
-Path (PathID, path 式)
-GW-Path-term (PathID, 索引語, 頻度 [, クエリ尤度モデル用スムージング値])
-GW-Path (PathID, 頻度, 総部分文書長)
-Term-filter (タグ, 索引語, 閾値)

図 6 索 引 構 造

索引，GW-Path-term 索引と GW-Path 索引，Term-filter 索引が該当)．ただし，これらの索引はクエリ処理時には用いられることはないため，従来システムと提案したシステムにおけるクエリ処理と重複リストの作成方法は同様である．

なお，部分文書の統合による検索結果再構築手法の適用には，文献 [14] で定義した索引構造に，いくつかの属性を付与する．部分文書の統合処理を行う際には，同一文書に属する部分文書間に存在する先祖・子孫関係を知る必要があり，効率的にクエリ処理を行うためにはクエリ処理時に任意の二つの部分文書間の包含関係を直ちに知ることが必要となるためである．部分文書間の包含関係を求めるには様々なアプローチが考えられるが，本システムにおいては，各部分文書の子孫部分文書の範囲の ElemID を格納することとする．つまり，子孫ノードの開始ノード番号である child_st と，子孫ノードの終了ノード番号である child_ed を Term 索引と Tag-term 索引へ付与する．これらの統計量の算出は両索引へ登録されるその他の統計量を算出する際に同時に算出可能であるため，child_st と child_ed を追加することで生じるオーバーヘッドは極めて小さい．

ここでは索引に関する細かな議論を行わないが，更新システムでは高速なクエリ処理によるユーザビリティの向上を目的とし，Top-k 検索 [1] が利用可能である．

4.2 精度低下の抑制と高速な索引更新

更新を考慮した XML 部分文書検索システムでは，高速な差分更新を可能とすることを目的として，1) システムに蓄積された文書量が十分でない場合にも正確な統計量を近似するための path 式統合手法と， 2) 検索結果として提示されない対象を選定するためのフィルタを提案した．

1) に関して，索引語の重み付けに用いる統計量は大別して二種類存在し，それぞれ，各部分文書中から算出される統計量である局所的重み (部分文書中に出現する索引語の出現頻度，部分文書中に含まれる索引語の個数など) と，テストコレクション全体から得られる統計量である大域的重み (同一 path 式で表される部分文書の個数や，その部分文書のうちの特定の索引語を保持する部分文書の個数など) である．なお，XML 部分文書検索において大域的重みを算出する際には，全ての部分文書を画一的に扱って統計量を算出するのではなく，同じ属性を持つを考えられる部分文書ごとに統計量を算出することが一般的である．属性の分類方法にはいくつかのアプローチが存在するものの，path 式が同じ部分文書は同じ属性を持つと扱うアプローチが効果的であると言われている [9]．しかしながら，システムに蓄積された文書数が十分ではない時点で起こりがちである，出現頻度が少ない path 式に関する大域的重みを算出する際には正確な値を算出することができないという問題を抱える．そこで，path 式ごとに属性を分類するのではなく，類似した path 式を纏めて一つの属性として扱うことで，蓄積された XML 文書数が十分ではない時点でも正確な大域的重みを近似的に算出することを目指す．

article
sec
emp

p_1: /article/sec/emp/sec
p_2: /article/emp/sec
p_3: /article/emp/sec/sec

図 7 ST 法による分類結果

その際の path 式統合手法に関して，path 式はタグの出現順序と出現頻度を厳密に考慮したタグ列として見なすことができるため，

（1） path 式の出現順序の制約を緩和するアプローチ
（2） path 式の出現頻度の制約を緩和するアプローチ
（3） path 式の出現順序と出現頻度の両方を緩和するアプローチ

の三種類のアプローチを提案した．評価実験の結果，中でも特に，(3) のアプローチを用いた場合に最も検索精度の低下を軽減させる効果的であることが判明した．

具体例を挙げて (3) の path 式統合手法の説明を行う．(3) のアプローチでは，path 式に含まれるタグ名のみに着目し，同じタグ集合で構成される path 式を同じ属性に分類する．図 7 の path 式 p_1, p_2, p_3 は何れも article タグ， sec タグ，emp タグから構成されるために統合される．従って，p_1, p_2, p_3 の path 式を持つ部分文書は纏められて大域的重みが算出される．

2) に関して，差分更新システムでは，文書の更新が発生してから短時間で索引を更新することを目指す．その際，XML 部分文書検索では一つの文書中から複数の部分文書が取り出される(注7)ため，部分文書検索の枠組みには大きなコストが必要である．従って，更新コスト削減を目的として，文書中の全ての粒度の部分文書を更新対象として扱うのではなく，検索結果として不適切であると見なすことが可能な部分文書を更新対象から除外する．また，クエリ処理において利用されることのない索引語を除外することで更新コストの削減を目指す．本稿との関連度が低いため，詳細については議論しないが，検索精度を低下することなくフィルタを適用可能であることが評価実験によって確認されている．

5. 評 価 実 験

本節では，更新を考慮した XML 部分文書検索システムに対して部分文書統合による結果再構築を行い，検索精度にどのような影響を及ぼすのか調査する．

(注7)：評価実験にて用いたテストコレクションである INEX 2008 Wikipedia collection は一つの文書中に平均 67 個の部分文書が含まれている．

表 1　検 索 精 度

	全クエリ		CO クエリ		CAS クエリ	
	iP[.01]	MAiP	iP[.01]	MAiP	iP[.01]	MAiP
索引再構築	.664	.213	.656	.253	.671	.178
baseline	.639	.140	.609	.174	.634	.133
初期スコア	.662	.228	.656	.291	.666	.174
BU	.668	.190	.654	.237	.680	.149
TD	.656	.189	.623	.230	.684	.154
BU·TD	.656	.190	.624	.231	.683	.154

表 2　path 式統合手法使用時の検索精度

	全クエリ		CO クエリ		CAS クエリ	
	iP[.01]	MAiP	iP[.01]	MAiP	iP[.01]	MAiP
索引再構築	.670	.214	.669	.257	.671	.177
baseline	.655	.199	.638	.219	.670	.182
初期スコア	.671	.227	.662	.291	.678	.171
BU	.674	.217	.656	.273	.689	.168
TD	.653	.215	.610	.266	.690	.172
BU·TD	.653	.215	.610	.267	.689	.171

表 3　INEX project 参加者の他システムとの精度比較

Team	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
提案手法 (BU)	.6898	.6736	.5647	.4789	.2167
Renmin Univ. of China	.5969	.5969	.5815	.5439	.2486
Queensland Univ. of Tech.	.6232	.6220	.5521	.4617	.2134
Univ. of Amsterdam	.6514	.6379	.5901	.5280	.2261

5.1　テストコレクションと評価尺度

評価実験では，INEX プロジェクトが提供する部分文書検索用テストコレクションである，INEX 2008 テストコレクションを利用する．テストコレクションは三つの要素から構成されており，1) 約 66 万件の英語版 Wikipedia 記事，2) 68 個のクエリ (CO クエリ:36 個，CAS クエリ:32 個)，3) システムの有用性を評価するための評価ツール，である．本テストコレクションでは，一つのクエリに対して 1,500 件の部分文書を回答として提示することが可能である．

なお，INEX プロジェクトにおける検索精度の公式尺度は，再現率 1% 点における精度，すなわち，検索結果上位の精度である iP[.01] である．加えて，再現率 0% ～ 100% までの 101 点の平均精度，すなわち，再現率が高い時点の精度も考慮したシステム全体の検索精度である mean average interpolated precision (MAiP) による評価も行う．

実験で使用した計算機は，CPU: Intel Xeon X7560 (2.3GHz) を 4 つ，512GB のメモリ，4.5TB のディスクアレイ装置 (1TB SATA HDD×12 の RAID 1 構成，計算機とは 4Gbps FC による接続) を搭載する．OS は Oracle Enterprise Linux 5.5 であり，システムは GNU C++ を用いて実装を行い，索引は BerkeleyDB にて構築した．

5.2　単純な差分更新システムを利用した評価実験

部分文書統合手法によって検索精度にどのような影響が生じるのか計測を行った結果を表 1 に示す．計測対象の手法として，差分更新を行わずに新たな索引を再構築する手法，単純に差分更新を行う手法 (baseline)，四種類の部分文書統合による検索結果再構成手法，即ち，初期スコアによるランキング手法，Buttom-Up スコアリング手法 (BU)，Top-Down スコアリング手法 (TD)，BU と TD を併用する手法 (BU·TD) の，全六つのアプローチである．また，各手法に対して，全てのクエリを用いた場合の検索精度 (iP[.01] と MAiP)，CO クエリのみを用いた場合の検索精度，CAS クエリのみを用いた場合の検索精度をそれぞれ掲載する．

なお，部分文書の統合手法において用いる，一つの文書から抽出することが可能な索引後数である抽出制限 EL の値は，今回用いたテストコレクションにおいて最適な値であるという結果が得られてた $EL = 1000$ を設定する．

実験の結果，baseline の手法と比較して，いずれの再構成手法を適用した場合においても，CO クエリと CAS クエリ共に検索精度が向上した．

公式尺度である iP[.01] に着目すると，BU において最も高精度であるという結果が得られ，この結果は更新を考慮しない XML 部分文書検索システムを用いた場合の実験と同様の結果である．baseline と比較して 5% の精度向上が確認され，索引を再構築した場合と遜色のない検索精度を示した．

また，初期スコア手法と，その他の三つの統合手法とを比較すると，CAS クエリを用いた場合にはより高精度検索が可能であるものの，CO 検索では多かれ少なかれ検索精度が低下した．このように，CO クエリと CAS クエリで，その振る舞いが大きく異なることが判明したが，その原因として，CO クエリにおける検索結果として適切な粒度よりも大きすぎる粒度の部分文書抽出された可能性が考えられる．というのも，構造の制約が存在する CAS クエリに対して，制約が存在しないために抽出制限の閾値を超えない限りは際限なく大きな粒度を抽出することが可能であるためである．これらの結果を踏まえて，CO クエリと CAS クエリごとに抽出制限の設定行うことで検索精度の向上が示唆された．これに関しては今後の課題とする．

5.3　path 式統合手法を適用した差分更新システムを用いた場合の評価実験

続いて，4.2 節の (3) のアプローチの path 式統合手法を用いて作成した索引に対して，5.2 節と同様の実験を行った．表 2 の通り，実験の結果は概ね表 1 による実験結果と同様の傾向を示した．また，path 式統合手法を用いた場合の BU は，索引を再構築した場合よりも高い検索精度を示した．

他の INEX project 参加者のシステムと検索精度の比較を行うことで本提案の有効性を確認した結果を表 3 を用いて説明する．比較対象として，我々と同様に CAS クエリと CO クエリ両方を用いて評価実験を行っている参加者 3 チームとの比較を行った [2] ところ，我々の提案手法は iP[.01] において最も高い検索精度を示した．これにより，提案手法は十分に有益な検索技術であると確認できた．なお，提案システム以外は文書の更新を考慮していないという点においても提案システムの有用性を主張することが可能である．

なお，提案システムにおいて，クエリが発行されて重複リストが作成するために必要な時間は 1 クエリにつき平均 1.89 秒

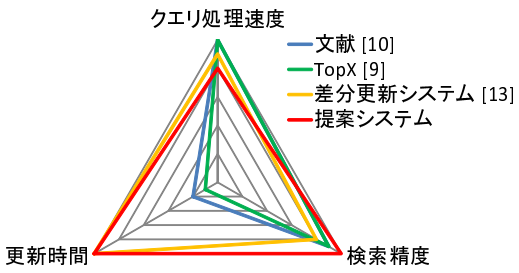

図 8　関連研究と本研究の位置付け

であり，重複リストから再構成リストを生成するために必要な時間は平均で 0.48 秒であった．つまり合計のクエリ処理時間は 2 秒程度である．

前述の議論より，path 式統合手法と検索結果の再構成手法を併用した場合には，高速な索引更新を実現しつつ，高精度・高速な検索を実現することが可能であるという結論が導かれた．

6. 関 連 研 究

図 8 を用いて本研究の位置付けを説明する．従来の XML 情報検索に研究では，検索精度の高精度化を目的とした研究と，クエリ処理の高速化を目指した研究，もしくはそれらの両立を目指した研究 [10], [11] が存在する．

Theobald らの提案した TopX [10] では，効率的な検索のためには二つのタイプの索引を提案している．一方は部分文書の得点付けに利用され，もう一方はクエリの構造制約の確認に利用される．更に，XML 部分文書検索に特化した Top-k アルゴリズムを考案している．コストベースによるクエリ処理を行うことで，構造制約の確認を行うタイミングの決定や，優先して読み込む索引語の特定を行なっている．

また，Trotman らは低コストにクエリ処理を行うため，索引に登録されるデータの圧縮と削減方法を提案している [11]．

なお，上記の研究においては，高精度検索を実現すべく，索引語の重み付け手法として提案システムと同様に BM25E [5] を用いている．従って，上記の研究共に，検索精度の高精度化とクエリ処理速度の高速化に取り組んでおり，何れも目標を達成しているといえる．しかしながら，文書の更新が発生した場合には新たな索引の再構築を行うために，更新に必要な時間が長時間に及ぶ．このような問題を解決すべく，我々は更新を考慮した XML 部分文書検索システム [14] を提案した．そして本稿では更に，文献 [13] にて提案した再構成手法を適用することで，検索精度において関連研究よりも高い精度を達成した．

7. お わ り に

本稿では，更新を考慮した XML 部分文書検索システムが引き起こす検索精度の低下を解決するために，部分文書の統合による検索結果の再構成手法を適用した．これにより，システムが作成する検索結果のリストから，文書中の最適な粒度の結果を取り出し，適切な順序にて検索結果をランキングを行う．

評価実験の結果，更新を考慮した場合でも，再構築手法を適用することで，新たに索引を再構築した場合よりも高精度な検索を達成することが確認された．

今後の課題として，Bottom-Up スコアリング手法や Top-Down スコアリング手法において，CO クエリと CAS ごとに適切なパラメータの設定をするが考えられる．また，より効率的なクエリ処理アルゴリズムの適用や，並列処理によるクエリ処理の高速化も同様に今後の課題である．

謝　　辞

本研究の一部は，JSPS 科研費 (特別研究員奨励費，基盤研究 (A)(課題番号:22240005)，基盤研究 (C)(課題番号:23500121)，若手研究 (B) (課題番号:22700248)) の支援による．ここに記して謝意を表す．

文　　献

[1] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. A Survey of Top-k Query Processing Techniques in Relational Database Systems. *ACM Computing Surveys (CSUR)*, Vol. 40, pp. 1–58, 2008.

[2] Jaap Kamps, Shlomo Geva, Andrew Trotman, Alan Woodley, and Marijn Koolen. Overview of the INEX 2008 Ad Hoc Track. In *INEX 2008 Workshop Pre-proceedings*, pp. 1–28, 2008.

[3] Gabriella Kazai, Mounia Lalmas, and Arjen P. de Vries. The Overlap Problem in Content-Oriented XML Retrieval Evaluation. In *Proc. of the 27th ACM SIGIR*, pp. 72–79, 2004.

[4] Fang Liu, Clement Yu, Weiyi Meng, and Abdur Chowdhury. Effective Keyword search in Relational Databases. In *Proc. of ACM SIGMOD*, 2006.

[5] Wei Liu, Stephen Robertson, and Andrew Macfarlane. Field-Weighted XML Retrieval Based on BM25. In *Formal Proc. of INEX 2005 Workshop*, Vol. 3977 of *LNCS*, 2006.

[6] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*, pp. 157–159. Cambridge University Press, 2008.

[7] Satoshi Nakamura. Trustworthiness Analysis of Web Search. *Journal of Japanese Society for Artificial Intelligence*, Vol. 23, No. 6, pp. 767–774, 2008.

[8] Paul Ogilvie and Jamie Callan. Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval. In *Formal Proc. of INEX 2005 Workshop*, Vol. 3977 of *LNCS*, 2006.

[9] Benjamin Piwowarski and Patrick Gallinari. A Bayesian Framework for XML Information Retrieval: Searching and Learning with the INEX Collection. *Journal of Information Retrieval*, Vol. 8, No. 4, pp. 655–681, 2005.

[10] Martin Theobald, Holger Bast, Debapriyo Majumdar, Ralf Schenkel, and Gerhard Weikum. TopX: Efficient and Versatile Top-k Query Processing for Semistructured Data. *The VLDB Journal*, Vol. 17, No. 1, pp. 81–115, 2008.

[11] Andrew Trotman, Xiang-Fei Jia, and Shlomo Geva. Fast and Effective Focused Retrieval. In *Formal Proc. of INEX 2009 Workshop*, Vol. 6203 of *LNCS*, 2010.

[12] Andrew Trotman and Börkur Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In *Formal Proc. of INEX 2004 Workshop*, Vol. 3493 of *LNCS*, 2005.

[13] 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, 2010.

[14] 欅惇志, 宮崎純, 波多野賢治, 山本豪志朗, 武富貴史, 加藤博一. XML 部分文書検索における索引の高速な差分更新と高精度検索. 第 5 回 Web とデータベースに関するフォーラム, 2012.