情報処理学会論文誌　データベース　Vol.6　No.4　1–16 (Sep. 2013)

文書の更新を考慮した高精度XML部分文書検索手法の提案

欅 惇志[1,2,a)]　宮崎 純[1,b)]　波多野 賢治[3,c)]　山本 豪志朗[1,d)]　武富 貴史[1,e)]　加藤 博一[1,f)]

受付日 2013年3月20日, 採録日 2013年7月7日

概要：本論文では，文書の更新を考慮した高精度 XML 部分文書検索の実現を目指す．検索システムにおいて，文書の更新に対応しなかった場合，適切な検索結果を提示できず検索システムの利便性が低下するが，文書の更新発生時に検索索引を一から再構築した場合には索引構築時間が長時間に及ぶため，本論文では索引の差分更新を行う．しかし，索引の差分更新時に，システムに蓄積された文書数が十分ではない時点や，文書の更新にともなって語の統計量が変化した場合，文書集合全体から算出される統計量である大域的重みを正確に推定できない可能性がある．これらの問題を解決するため，既存の XML 部分文書検索システムに索引の差分更新機能を付与し，さらに，高速な差分更新を行うために文書中の重要な箇所と索引語のみを索引へ追加するためのフィルタと，正確な大域的重み推定のための path 式統合手法を用いた索引語の重み付け手法を提案する．評価実験の結果，統計量が変動しない文書集合に対して，提案手法は単純な差分更新と比較し，検索精度を 4%向上させつつ，索引の更新速度を 25%高速化した．また，統計量が変動した場合にも，速やかにその変化に追従して検索精度を向上させることが可能であることが判明した．

キーワード：XML 部分文書検索，文書索引，差分更新，大域的重み，性能評価

A Proposal of Accurate XML Element Retrieval Considering Document Updates

Atsushi Keyaki[1,2,a)]　Jun Miyazaki[1,b)]　Kenji Hatano[3,c)]　Goshiro Yamamoto[1,d)]
Takafumi Taketomi[1,e)]　Hirokazu Kato[1,f)]

Received: March 20, 2013, Accepted: July 7, 2013

Abstract: In this paper, we propose a method for accurately retrieving XML elements considering document updates. If document updates are not handled in a search system, users cannot obtain appropriate search results, which reduces the usefulness of the search system. We apply an incremental approach to update an index because a rebuild-from-scratch approach takes longer time. In addition, global weights, i.e., the statistics computed with all documents in the search system, may not be accurate when a few number of documents is indexed or when global weights change drastically. To solve these problems, we propose to extend a function of incremental updates of indices to general XML element retrieval systems, with filters to reduce the update cost by eliminating unimportant elements and terms. Moreover, we apply a method for integrating path expression which estimates accurate global weights in term calculation. Experimental results showed that our proposed method can be up to 25% faster to update indices than the simple incremental updates and can improve the search accuracy by 4% with document set of static statistics. The proposed method can also search accurately, even under continuous changes in the statistics of the documents.

Keywords: XML element retrieval, document index, incremental update, global weight, performance evaluation

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

[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, Ikoma, Nara 630–0192, Japan
[2] 日本学術振興会
Japan Society for the Promotion of Science, Chiyoda, Tokyo 102–0083, Japan
[3] 同志社大学文化情報学部
Faculty of Culture and Information Science, Doshisha University, Kyotanabe, Kyoto 610–0394, Japan
[a)] atsushi-ke@naist.ac.jp
[b)] miyazaki@is.naist.jp
[c)] khatano@mail.doshisha.ac.jp
[d)] goshiro@is.naist.jp
[e)] takafumi-t@is.naist.jp
[f)] kato@is.naist.jp

書）中の部分文書（element）であり，その目的は，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．クエリに対する適合箇所と，適切な粒度の部分文書に含まれるテキストはおおむね一致するため [1]，適切な検索結果を提示することができればXML 部分文書検索システムを利用することでユーザの検索時間の短縮と適合箇所を発見する労力の軽減を期待することができる．

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 [2], [3], [4], [5] や高速検索に関する研究 [6], [7], [8], [9]，もしくはそれらの両立を目指す研究 [10], [11], [12], [13] が取り組まれてきた．これらの研究が取り組まれ始めた当初は，まずは固定の文書集合から各検索技術の性能を向上させることが最優先の課題であったが，現在これらの検索技術は成熟しつつあるため，さらなる実用性を考慮する段階に到達したといえる．

XML はデータ交換の標準フォーマットとして広く利用され現在までに数多く産出されており，今後ますます多くの XML 文書が作成されると考えられる．当然それら XML 文書は内容が書き換えられたり，場合によっては削除されたりするなどの更新操作が頻繁に発生していると考えられる．実際，XML 形式へ変換可能[*1]であり XML 部分文書検索技術の適用対象の 1 つである Wikipedia（英語版）では 1 時間に 4,000 から 8,000 回の記事更新が発生している[*2]．文書の追加・削除・書換に対応しなかった場合には，新たに追加された文書を検索結果として提示することができない，すでに削除されている文書を検索結果として提示する，古い情報を基にしたスコアリングを行う，といった問題が起こる．したがって，検索システムの運用の際，これらの文書の更新に対応しなければその利便性の低下が懸念されるため，本論文では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．

本研究には以下の 2 つの目標が存在する．

(1) 文書の更新が発生すれば可能な限り高速に検索システムに反映させる．
(2) 正確な大域的重みを推定することで，正確な索引語の重み付けを行う．

1 つ目の目標を達成するためには，

- 文書の更新にともなって更新可能な索引の構築
- 索引更新処理の高速化

の 2 点が必要である．従来の XML 部分文書検索システムでは文書の更新が発生した場合には新たな索引[*3]を一から再構築することで更新に対応するため，蓄積される文書数の増加につれて索引再構築に時間を要することになる．そのため，本論文では差分更新アプローチによって索引の更新を行う．

また，索引に差分更新機能を持たせるだけでなく，既存の XML 部分文書検索で利用される索引構造を拡張することで文書の追加・削除・書き換えの各更新を可能な限り高速に反映させる手法を提案する．また，文書の追加時において，XML 部分文書検索では文書数に対して検索対象となる部分文書数が大幅に増大する[*4]ため，すべての更新対象を格納した場合には索引更新速度の低下が起こる．この問題に対して，文書中の重要箇所やクエリ処理において大きな影響を及ぼす索引語のみを登録するためのフィルタを提案する．

2 つ目の目標に関して，差分更新アプローチにより索引を更新する場合，

- 検索システム運用初期など，システムに蓄積された文書量が十分ではない時点
- 文書の更新が行われる過程で新たなトピックの出現などにより急激に語の統計量が変化した場合

などにおいて，索引語の重みを正確に算出できない可能性がある．これは，索引語の重み付けを行う際に用いる統計量の一部は文書集合全体から算出される大域的重みであるが，前述の状況において正確な大域的重みを算出することができないためである．そのため，path 式統合手法を用いて正確な大域的重みを推定することで，どのような状況においても正確な索引語の重み付けを目指す．

提案手法の有効性を検証するため，文書の更新にともなって文書集合中のトピックがほとんど変化しない場合と，文書集合中に新たなトピックが急激に出現する場合における評価実験を行う．前者の実験では，限られた量の文書集合から正確な大域的重みを推測することが可能であるのかどうかを検証する．そして後者の実験では，提案する推測手法は大域的重みの変化に対してどの程度正確に重みを算出することが可能であるのかを明らかにする．

以降，2 章で基本的事項について，3 章で関連研究について述べる．4 章では 1 つ目の目標である索引の高速な差分更新実現のための取り組みについて，5 章では 2 つ目の目標である正確な大域的重みの推定に関しては議論する．6 章で評価実験を行い，7 章で本論文をまとめる．

2. XML 部分文書検索

2.1 XML 部分文書

図 1 は文書 ID（document ID，DID）が 1 の XML 文書である．XML 文書には 2 つのタイプが存在し，一般的には以下のように分類されている [14]．

データ指向型 XML 文書　各テキストノード中には数値や記号，単語，複合語などの属性値が格納され，表形式で表現できないデータの維持，管理に用いられること

[*1] http://dumps.wikimedia.org/enwiki/
[*2] http://www.wikichecker.com/editrate/
[*3] 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うえで用いられるデータ群である．
[*4] 評価実験で用いる INEX 2008 Wikipedia collection において，1 つの XML 文書中に含まれる部分文書は平均 161 個である．

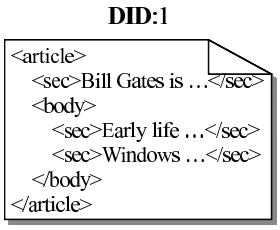

図 1　XML 文書
Fig. 1　An XML document.

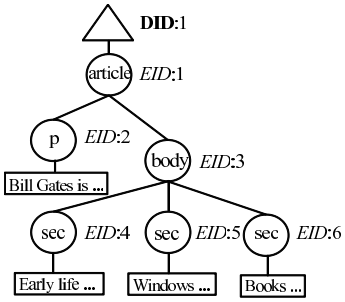

図 2　XML 木
Fig. 2　An XML tree.

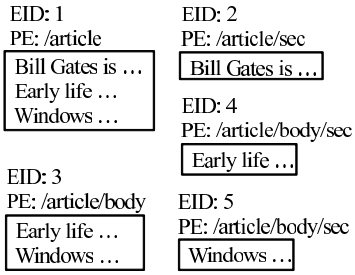

図 3　部分文書
Fig. 3　XML elements.

が多い．

文書指向型 XML 文書　各テキストノード中には文章が 1 つ以上含まれ，マークアップ文書の記述に用いられることが多い．

XML 部分文書検索技術の適用対象は主に文書指向型 XML 文書であるため，本論文で想定する XML 文書も同様に文書指向型 XML 文書である．

図 1 の XML 文書から取り出される部分文書を**図 3** に示す．各部分文書には部分文書 ID（element ID，EID）が付与され，これは文書の前方から後方へ走査した際に出現する順序（文書順）に従って値が割り振られる．各部分文書は，DID と EID によって一意に識別される．このとき，XML 文書内の開始タグと終了タグに囲まれるそれぞれのテキストが各部分文書と対応している．

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ている．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と対応する[*5]．つまり，文書全体を表す `article` ノードは子孫に存在するテキストノードすべてを持ち，`body` ノードは子ノードである 3 つの `section` ノードに含まれるそれぞれのテキストノードを保持する．このとき，包含関係（先祖・子孫関係）にある部分文書間においてテキストノードの重複が発生する．

各部分文書の path 式（path expression，PE）をあわせて記載するが，これは文書の根から部分文書へ至る経路上に存在するタグを “/” で区切り結合した文字列である．

仮にユーザが図 1 の “Early life . . .”，“Windows. . .” に関する情報を要求する場合，XML 部分文書検索では図 3 の EID 3 の部分文書を提示する．これは，部分文書中にユーザが要求する情報すべてを含み，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．

2.2　XML 文書に対するクエリ

ユーザの情報要求の表現には，キーワードの指定と文書構造の指定とがある．キーワードの指定のみを行うクエリは CO（Content Only）クエリ，キーワードと構造の両方を指定するクエリは CAS（Content And Structure）クエリと呼ばれる [15]．

CO クエリと CAS クエリそれぞれの XPath I（NEXI）[16] での問合せ例を示す．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 EID 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．検索結果の候補部分文書は，条件 (1) を満たす部分文書を先祖に持つような，条件 (2) を満たす部分文書である．

条件 (1)　前半の `//article[about(., "Gates")]` に着目すると，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．

条件 (2)　後半の `//sec[about(., "Windows")]` に着目すると，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．

図 3 中では EID 5 のみクエリの制約を満たす．

3.　関連研究

3.1　高精度・高速な XML 検索

3.1.1　高精度 XML 検索

XML 部分文書検索においての最重要課題の 1 つは正確な検索を行うことである．適合部分文書を発見する際に，まず部分文書中の各索引語に対して任意のスコアリング手法によって重みを算出し，それらの重みからクエリと適合する部分文書を特定するアプローチが主流である．

XML 部分文書検索において利用されるスコアリング手法の多くは，文書検索用のスコアリング手法をもとに拡張

[*5] 各部分文書にはタグも含まれるが，評価ツールによって解釈されるのはテキストノードのみであるため，本論文において部分文書とは各ノード以下のテキストノードを統合した文字列とする．

されている．各スコアリング手法ごとにさまざまな統計量を用いてスコアリングが行われるが，それら統計量を大まかに分類すると，各文書（部分文書）から算出される統計量である局所的重み（local weight）と，文書集合全体から算出される統計量である大域的重み（global weight），定数・パラメータの 3 種類から構成される．局所的重みはただちに算出可能であるのに対して，文書集合全体を走査しなければならない大域的重みは即座に算出できない．

文書検索と部分文書検索間の最大の違いは，大域的重みの算出方法である．文書検索においてはすべての文書は同じ属性を持つと見なして，すべての文書を母集団として大域的重みを算出する．部分文書においては，同じ属性を持つと見なす部分文書群に分類し，それぞれを母集団として大域的重みの算出を行う．

属性の分類にはいくつかのアプローチがあるが，その中でも同じ path 式を持つ部分文書は同じ属性を持つとして統計量を算出するアプローチが最も高精度とされている [17]．たとえば図 3 において，EID 4，5 はいずれも同じ path 式 `/article/body/sec` を持つため，これらは同じ属性を持つと判断される．

部分文書検索用の索引語の重み付け手法には TF-IPF [2] や BM25E [4]，部分文書検索用のクエリ尤度モデル [5] などが存在する．

3.1.2　高速 XML 検索

XML 部分文書検索において，高速な検索を実現することも重要である．

高速な検索のためにさまざまな取り組みが行われており，

(1) Top-k 検索の適用 [10]
(2) 索引に格納されるデータを圧縮や削減してクエリ処理時に読み込むデータ量を最小限に抑制 [11]

などのアプローチが存在する．Top-k 検索はスコアの上位 k 件の結果を高速に提示するクエリ処理法である．さまざまなアルゴリズムが提案され [18]，索引語の重みを計算したうえで索引へ格納する手法や，さらに索引語の重みと付随する情報から構成されるタプルを索引語の重み順にソートする手法がある．これらにより，クエリ処理時に索引中から少数のタプルを読み込むだけで検索結果を特定することができ，高速な検索が可能となる．さらに，ランダムスキャン用の索引を補助的に用い，任意の部分文書中の索引語の重みを参照できるようにするアプローチもある [18]．

Theobald らは効率的な検索のために 2 種類の索引と Top-k アルゴリズムを提案している [10]．一方の索引は部分文書の得点付けに利用され，もう一方は CAS クエリが発行された場合にクエリの構造制約の確認に利用される．そのうえで，コストベースのクエリ処理を提案しており，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っている．

Trotman らは低コストな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]．

また，文献 [10], [11] では高速な XML 検索を目指しつつ，高精度検索を目的としたスコアリング手法 [4] を利用することで高精度かつ高速な検索の実現を目指す．

なお，これら従来の高精度・高速な検索に関する研究では，テストコレクションのような固定の文書集合からユーザの情報要求と合致する適合部分文書を高速に提示することを目的としているため，文書の更新を高速に検索システムに反映させることを目的としていない．

3.2　文書更新の反映

文書の追加・削除・書き換えといった更新が頻繁に発生する場合，これらの更新を検索システムに反映させることは重要である．仮に検索システムが文書の更新に対応しなかった場合，検索対象文書と検索システム上のデータに不一致が発生する．その結果適切な検索結果を提示できないという問題が起こるため，文書の更新への対応は不可欠である．

近年，文書の更新への対応を目指す研究が着目されており，Chen らは情報抽出分野においてこの課題に挑戦している [19]．統計的情報抽出技術を利用する際には文書集合全体の持つ統計量を利用するため，既存研究では文書の更新が発生すればそのたびに文書集合全体に対して情報抽出技術の適用を行っていた．そのため，文書集合の増大にともない，文書の更新の発生から情報抽出が行われるまでに遅延が発生する．この遅延を短縮すべく，Chen らは過去の文書集合に対する情報抽出を行った際に作成した中間結果を再利用する手法を提案している．

Neumann らは RDF データの検索において過去のスナップショットの情報を利用する手法の提案を行っている [20]．Ren らはグラフ構造の変遷を高速に検索することを目指し，最新のグラフだけでなく過去のスナップショットも保持する手法を提案している [21]．

これらの研究では，いずれも過去のスナップショット情報を中間結果として利用することで，高速に文書の更新を反映させている．我々も同様に過去のスナップショットにおける中間結果，すなわち，構築された索引を差分更新することで，速やかに文書の更新に対応するという点で関連しているが，本研究は XML 部分文書検索システムを対象としているという点で異なる．なお，XML 部分文書検索システムを含め，検索システムは低コストで索引の更新を行いつつも高性能を保つことが期待される．ここでの高性能を保つとは高精度検索を維持することである．

なお，文書検索の転置索引の差分更新による高速化に関してさまざまな研究が存在するが [22], [23], [24]，これらは索引のデータ構造と物理的な格納方法の効率化を目指しており，高精度検索を保ちつつ，索引の差分更新のための効率的なデータ管理の仕組みを提案する本研究とは異なる．

4. 索引の高速な差分更新

一般的な構成の XML 情報検索システム [10], [11] では，索引構築機能とクエリ処理機能を備える．それらのシステムでは更新が発生すれば索引を一から再構築するため文書量に比例して索引構築時間が増加するが，提案システムは，文書の更新が発生すれば高速に更新を行うために索引の差分更新を行う．さらに差分更新の効率化のため，文書更新時の索引更新コストを削減する 2 つのフィルタを提案する．

4.1 既存機能の拡張

実時間で索引の差分更新を実現するためには，新たに追加された文書に対する索引更新時において，高速に索引語の重みを算出する必要がある．その際，索引語の重み算出のたびに，時間を要する大域的重み計算を行うと更新性能が著しく低下する．そのため，索引に大域的重みを格納することで索引の高速な差分更新を実現する．ただし，索引構築法やクエリ処理は高精度かつ高速な XML 検索を実現する従来のシステムと同様である．

4.1.1 索引語の重み付け手法

本論文では XML 部分文書検索において，最も高精度なスコアリング手法の 1 つとされる BM25E を用いる [25]．BM25E [4] は確率モデルに基づくスコアリング手法である．古典的なスコリング手法である TF-IPF [2] は，索引語の重みを算出する際に索引語の出現頻度の情報が利用される．それに対して，BM25E による重み付けでは，さらに部分文書の持つ索引語数も考慮している．path 式 p で表されるある部分文書 e 中の索引語 t の重み $w(p,e,t)$ は以下の数式で算出される．

$$w(p,e,t) = \frac{(k_1+1)tf_{e,t}}{k_1((1-b)+b\frac{el_e}{avel_p})+tf_{e,t}} \cdot \log\frac{N_p - pf_{p,t}+0.5}{pf_{p,t}+0.5} \quad (1)$$

ただし，$tf_{e,t}$ を部分文書 e 中の索引語 t の出現頻度，el_e を部分文書 e の索引語数，$avel_p$ を p で表される部分文書の平均索引語数，N_p を p で表される部分文書数，$pf_{p,t}$ を p で表され，かつ，t を含む部分文書数，k_1, b をパラメータとする．パラメータのチューニングを行った結果をふまえ，以降の実験では $k_1 = 2.5$ と $b = 0.85$ を設定する．

最終的に e のスコア $s(e)$ は，クエリキーワード集合を T として以下の式で算出される．

$$s(e) = \sum_{t_i \in T} w(p,e,t_i) \quad (2)$$

4.1.2 索引構造

文献 [10] を含む既存研究の多くでは，索引語の重みはあらかじめ算出されたうえで索引に格納されており，クエリの構造制約の確認も索引を用いて行っている．提案する索引でも同じ方式を採用する．さらに，索引の差分更新において重要となる高速に索引語の重みを算出するため，大域的重みも索引に格納する．

-Term (DID, EID, 索引語, 索引語重み, PID, 部分文書長, VID)
-Tag-term (DID, EID, タグ, 索引語, 索引語重み, PID, 部分文書長, VID)
-RS (DID, EID, 索引語, 索引語重み, VID)
-Path (PID, path 式)
-GW-Path-term (PID, 索引語, 頻度)
-GW-Path (PID, 頻度, 総部分文書長)
-Term-filter (タグ, 索引語, 閾値)

図 4　索引構造
Fig. 4　Index structure.

文献 [10] と同様に RDB スキーマの形式で記述した索引構造を図 4 に示す．ただし，下線が引かれた属性は主キーである．

Term 索引，Tag-term 索引，RS 索引，そして Path 索引は高精度・高速な検索を実現するために用いられる．一方，GW-Path-term 索引と GW-Path 索引には，即座に算出が困難な各種大域的重みを格納し，索引語の重み算出計算を高速化するために用いる．式 (1) 中の大域的重みは $avel_p$，N_p，$pf_{p,t}$ が該当する．なお，Term-filter 索引については 4.3.2 項において議論する．

4.1.3 Top-k 検索によるクエリ処理

高速なクエリ処理によるユーザビリティの向上のため，提案システムではスコアの上位 k 件を高速に発見する Top-k 検索 [18] を利用する．Term 索引中のタプルは索引語ごとに，Tag-term 索引中のタプルはタグ–索引語ペアごとにまとめられ，索引語の重みの大きさの降順にソートされている．クエリ処理時には各索引語（構造指定クエリでは各タグ–索引語ペア）ごとに上位 k 件のタプルのみ索引から取得され，各タプルに含まれる索引語の重みを部分文書ごとに加算することで部分文書のスコアを算出する．もし，部分文書の正確なスコアを算出する場合など，特定の索引語の重みを参照する必要があれば，ランダムスキャンを行うことで任意の部分文書の索引語の重みを読み込むことが可能である．これにより，Top-k 検索による高精度・高速なクエリ処理を実現する．

クエリ処理の際，CO クエリに対しては Term 索引，CAS クエリに対しては Tag-term 索引をシーケンシャルに読み込むことで検索結果候補の部分文書を取り出す．次に RS 索引を補助的に用いてランダムスキャンを行うことで各部分文書の正確な値を算出する．

もし，CAS クエリがクエリの構造制約を 2 組以上持つ場合には，各部分文書の持つ path 式 ID

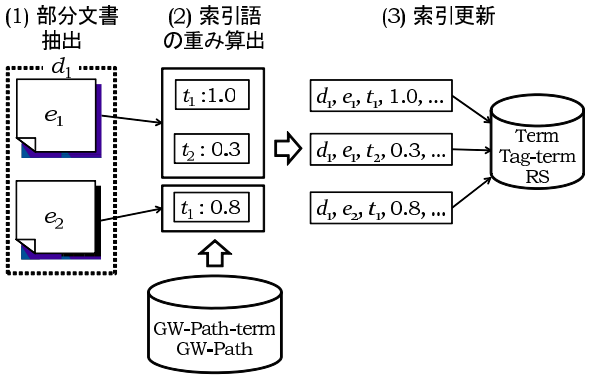

図 5　文書追加時の更新処理
Fig. 5　Updating process for insertion of a document.

(path expression ID, PID) がクエリの構造制約を満たすのかどうか確認を行う必要がある．たとえば，CAS クエリ `//article[about(., "Gates")]//sec[about(., "Windows")]` の構造制約を満たす path は複数存在し，`/article/sec` や `/article/body/dev/sec`, `/article/body/sec/sec` などが該当する．各部分文書の path 式が制約を満たすのかどうかを 1 つずつ確認することは高コストであるため，クエリが入力された時点で `path` 索引を用いてクエリの構造制約を満たす PID を列挙し，検索結果にはクエリの構造制約を満たす PID をもつ部分文書のみを提示する．

4.2　文書更新時の処理

4.2.1　文書追加

文書の追加時には，以下の手順で処理を行う．

(1) 文書から部分文書を取り出す．
(2) 各部分文書中に含まれる索引語の重みを算出する．
(3) 索引の更新を行う．

図 5 とあわせて，各処理を説明すると，まず追加された文書 d_1 から部分文書 e_1，e_2 が取り出される．次に，部分文書 e_1 中の索引語 t_1，t_2，そして部分文書 e_2 中の索引語 t_1 の重みを算出する．その際，`GW-Path-term` 索引と `GW-Path` 索引を用いれば即座に大域的重みを参照可能でき，高速に索引語の重みを算出することが可能となる．

索引語の重みが算出されれば，索引の更新を行い文書追加時の処理を完了する．索引語の重みは付随情報とともに `Term` 索引と `Tag-term` 索引，`RS` 索引に格納される．

以上のとおり，高速な索引語重み算出によって実時間で索引の差分更新を実現できる．なお，索引更新の最小単位は文書単位だが，更新の効率化を目的として，複数の文書単位に拡張することも可能である．

4.2.2　文書削除

文書の削除が発生した場合，削除対象文書に関連するタプルは索引中に散在しているため，即座に該当タプルすべてを削除することは高コストである．その代わりに，削除

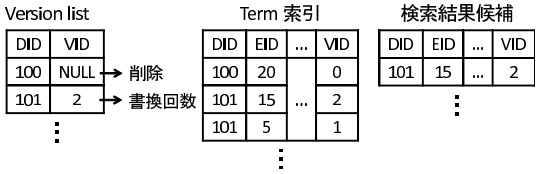

図 6　Version list とそのクエリ処理例
Fig. 6　Query processing with version list.

が起こった時点でただちに該当文書のタプルを削除するのではなく，削除された文書を記録し，クエリ処理時に検索結果候補から除外することで文書削除に即座に対応する．

削除文書情報を管理するため，DID とバージョン ID (Version ID, VID) のペアを格納する Version list を利用する．削除された文書の DID は Version list に書き込むが，その際の VID には削除を表す `NULL` 値を用いる．Version list 中に該当文書の DID がすでに存在する場合には VID に `NULL` をセットする．たとえば，**図 6** 中の DID 100 の VID は `NULL` であるため，DID 100 の文書は削除済みである．

もし，Version list の増大化にともなってクエリ処理効率が低下する場合には，計算機の負荷が低い状況を見計らって Version list 中の DID を持つタプルを索引から削除し，それが完了すれば Version list 中の該当文書の情報も削除することで効率的なクエリ処理が可能となる．

4.2.3　文書書換

文書の書換が発生した際にも，書換情報を即座に反映させるべく Version list を利用する．その際，書き換え前文書に関連するタプルの削除と書き換え後文書の索引への追加を組み合わせることで書き換えを実現する．ただし，削除は前項の削除処理に準じる．次に，書き換え後の最新文書を索引へ追加し，クエリ処理時には最新のデータ以外を検索結果候補から除外することで，文書書き換えに対応する．

なお，本論文では，文書の書き換えは文書単位で行う．つまり，文書のうちの一部の部分文書のみが書き換えられた場合にも，文書に含まれる部分文書すべてが書き換えられたと見なして処理する．仮に部分文書単位による書き換えを行った場合には Version list が増大化しクエリ処理の効率の低下が予想される．また，書き換えによって文書構造が変化する可能性があり，更新前と更新後の部分文書の対応付けによるコストも発生する，といった問題が起こるためである．ただし，文書単位による処理によって，更新の影響を受けない部分文書は書き換え前と同一内容のタプルが重複して索引に格納されるという問題が起こるため，効率的な部分文書単位の書き換えへの対応は今後の課題の 1 つである．

書き換えが起こると，まず，書き換えられた文書の DID が Version list 中にあるかどうかを確認する．リスト中に

存在し，値が NULL である場合には VID に 0 を代入し，それ以外は VID の値に 1 を加算する．リスト中に存在しなければ VID を 1 として新たに書き加える．たとえば図 6 中の DID 101 の VID は 2 であるため，文書 101 は 2 度の書き換えを経ていることを表す．続いて文書の追加時と同様の手順で Term 索引，Tag-term 索引，RS 索引の更新を行うが，その際に先ほど Version list へ書き込んだ値と等しい VID を各タプルにセットする．このとき，書き換えによって文書構造が変化する場合もあるため，同一の EID であったとしても，異なるバージョン間では異なる部分文書を示す可能性がある．なお，初めて文書が追加されたときには VID は 0 にセットされる．

クエリ処理時には，削除時と同様に VID を確認することでデータの有効と無効を判別する．読み込んだタプルの VID が，リスト中の該当文書の DID と紐付く VID と等しい場合，もしくはリスト中に DID が存在しない場合は，読み込んだタプルは最新状態のものである．もしリスト中の VID よりも小さい場合には最新状態のタプルではないため，検索結果の候補から除外する．

削除時と同様，Version list の増大化にともなってクエリ処理効率の低下が起こる場合には，計算機の負荷が低い状況を見計らって索引から古いバージョンのタプルの削除を行う．その際，索引中の最新のデータの VID を 0 に書き換え，処理が完了すれば Version list 中の該当文書の情報も削除することで効率的なクエリ処理が可能となる．

4.3　更新対象選定のためのフィルタ

検索精度に影響を及ぼさない部分文書や索引語を選定し，更新対象から除外することを目指してフィルタを提案する．フィルタによって更新対象を事前に削減すれば索引更新に要する時間が短縮されるのは明らかであるものの，検索結果として解となりうるものを除外した場合には検索精度および再現率に悪影響を及ぼす．したがって，フィルタを設定する際には慎重に条件を設定する必要がある．

4.3.1　部分文書フィルタ

我々は文書集合全体からいかなるクエリに対しても解にならない部分文書を選定する研究 [26] や，検索結果としてより適切な粒度を決定する研究 [27] に取り組んできた．これらの研究で得られた知見として，文書全体などの大きな粒度の部分文書にはユーザの情報要求に適合しない箇所を含む可能性があり，逆に，小さな粒度の部分文書はそれ単体ではユーザの情報要求を満足させることができず，部分文書の粒度としては中程度の粒度が適切であるということが判明した．ここでは，比較的判定容易な，小さな粒度の部分文書の中でも不要である可能性が高い部分文書を除外することに焦点を当てる．

まず，除外すべき部分文書，すなわち小さな粒度の部分文書について説明を行う．XML 文書の中には目次や本文，リンク集など複数の論理的な内容から成り立っているものも多く存在するが，目次やリンクなどはキーワードが列挙されているのみである．これらの情報は情報検索を行ううえでナビゲーショナルである可能性はあるが，直接情報要求を満たす可能性は低い．自然言語処理によって文書を要約する際の要件の 1 つとして，「提示される情報のみで理解可能な情報である」ことがあげられており [28]，実験で利用するテストコレクションの評価尺度も同様の観点による評価が行われているため，本システムでは，それ単体で理解不可能な部分文書は検索結果から除外する．また，表やリスト中の各セルの値なども，それ単体で理解不能であるために除外されることが望ましいと考えるが，田邊ら [29] の提案と同様に文書コンテンツとデータコンテンツを判別したうえで検索結果の構築を行うことで，必要に応じて表やリストの一部を周辺テキストとあわせて提示することも可能である．

これらをふまえて，小さすぎる粒度の不要な部分文書の条件として，過去の研究 [30] で用いた (1) に加えて，さらに (2) と (3) の条件を以下に示す．

(1)　含まれる索引語がきわめて少ない部分文書：閾値 τ_{el}
(2)　きわめて深い path 式を持つ部分文書：閾値 τ_{depth}
(3)　希少な path 式を持つ部分文書：閾値 τ_{Zipf}

ただし，2.1 節でも述べたとおり，目次などからのみ構成される部分文書が除外されたとしても，除外される部分文書を含む先祖部分文書は索引付けされるため，目次などのキーワードが検索対象から除外されるわけではない．

(1) は，目次や参照情報などの本文以外の付加情報に含まれる索引語数や異なり語数は小さくなることによる．

(2) は，表やリストは深い階層構造で表現されることがあるが，表やリスト全体のうちの一部のみが提示された場合にユーザはそれらの内容を適切に解釈することができないため，情報提示の単位としては不十分であるためである．

(3) は，path 式の出現頻度の少ない path 式を除外するためである．文書集合中での出現頻度が低い path 式は，適切に重み付けが行われていない可能性が高い[*6]．このため，Zipf の法則 [31] によって中頻度の path 式の閾値を求め，閾値以下の低頻度の path 式を持つ部分文書を除外する．中頻度の目安 f は以下の式 (3) で算出できる．

$$f = \frac{\sqrt{8F_1 + 1} - 1}{2} \tag{3}$$

ただし，F_1 は文書集合中の出現頻度が 1 回のみである path 式の種類数である．

ここで，部分文書フィルタの動作例を**図 7** を用いて説明する．ある文書 d_1 が追加され，その部分文書 e_1，e_2，e_3，e_4 が取り出されたとする．部分文書 e_1，e_2，e_4 はそれぞれ，含まれる索引語数が少ない，深い path 式を持つ，希少

[*6] 5.2 節で詳細に議論を行う．

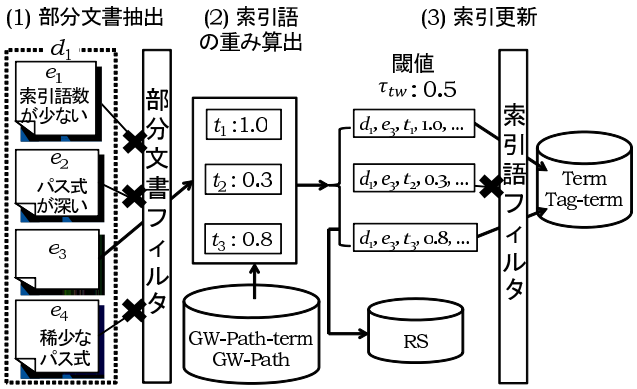

図 7　部分文書フィルタと索引語フィルタ
Fig. 7　Element and term filters.

な path 式を持つという理由で部分文書フィルタによって除外される．その結果，部分文書 e_3 中の索引語のみ重みが算出される．

4.3.2　索引語フィルタ

提案システムでは Top-k 検索を行うために，重みの小さな索引語はクエリ処理時に読み込まれない．そのような索引語を更新対象から除外することで更新コストの軽減を目指すため，閾値 τ_{tw} を索引に含まれる各タグ–索引語ペアの n 番目に大きな索引語の重みとする．我々の研究成果 [32] において，閾値 τ_{tw} を小さくすると検索精度は低下しないが削減効果は低く，逆に閾値を大きくすると削減効果は大きいが検索精度が低下するということが判明している．このため，索引語フィルタによって登録される索引語が削減された場合にも，部分文書の正確な値を算出することを可能とするため，`RS` 索引には索引語フィルタを適用しない．なお，索引語数が n 件未満の場合にも索引語フィルタを適用せず，閾値は即座に参照できるよう，あらかじめ `Term-filter` 索引に格納する．

索引語フィルタの動作を図 7 を用いて説明する．τ_{tw} はタグ–索引語ペアごとに閾値を決定するが，簡略化のためにここでは τ_{tw} を 0.5 とし，3 つの索引語 t_1，t_2，t_3 が索引に追加されようとしているとする．索引語 t_1，t_3 は閾値よりも大きな重みを持つために `Term` 索引，`Tag-term` 索引，`RS` 索引すべての索引に追加され，索引語 t_2 は閾値よりも小さな値を持つために `RS` 索引のみに追加される．

5.　正確な大域的重み推定

5.1　索引の差分更新による影響

提案システムを評価する前に，差分更新を行うことによってどのような影響が発生するのかを予備実験を行い確認する．

5.1.1　テストコレクションと実験環境

予備実験では，INEX プロジェクト[*7] 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以下の 3 つの要素から構成されている．

- 約 66 万件の英語版 Wikipedia 記事
- 68 個のクエリ（CO クエリ：36 個, CAS クエリ：32 個）
- 検索システムの有用性を評価するための評価ツール

本テストコレクションでは，1 つのクエリに対して最大 1,500 件の部分文書を回答として提示する．なお，INEX プロジェクトにおける検索精度の公式尺度は，検索結果上位の精度である再現率 1%点における精度（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 MAiP（mean average interpolated precision）による評価も行う．

実験で使用した計算機は，CPU：Intel Xeon X7560（2.3 GHz）を 4 つ，512 GB のメモリ，4.5 TB のディスクアレイ装置（1 TB SATA HDD ×12 の RAID 1 構成，計算機とは 4 Gbps FC による接続）を搭載する．OS は Oracle Enterprise Linux 5.5 であり，実験システムは GNU C++ を用いて実装を行い，索引は BerkeleyDB 5.3 を利用して構築した．その際，効率的に索引を構築するために，重複するデータは BerkeleyDB の二次索引を用いて構築を行った．これにより，索引のディスクサイズや I/O コストが軽減される．

5.1.2　実験手順

差分更新が発生する以前の状態の索引を**初期索引**，初期索引を作成するための文書集合を**初期文書**，索引を差分更新するための追加文書集合を**更新文書**と呼称する．ここでは，初期文書と更新文書の持つ統計量が等しい状況を想定する．このためには，文書をランダムに抽出して初期文書と更新文書に分類する．なお，本節の実験では文書の追加に焦点を当てる．

実験は，まず初期文書から初期索引の構築を行い，続いて，更新文書を用いて `Term` 索引と `Tag-Term` 索引，`RS` 索引の差分更新を行うという手順で実行する．また，検索精度の測定では，過去の研究 [27] の知見を利用し，索引語数が一定数よりも少ない部分文書を検索結果から除外する[*8]．

5.1.3　予備実験の結果

文書集合全体に対する初期文書の割合を変化させて検索精度を計測した結果を**表 1** に示す．たとえば，割合が 20%のときには初期索引は全 66 万文書のうちの 20%の文書から構築され，残りの 80%の文書によって索引の差分更新が行われる．計測項目は，すべての文書の更新が完了した時点における iP[.01] と MAiP による検索精度である．また，差分更新の特徴を調査するため，差分更新に要する時間と，初期索引の構築と差分更新の総処理時間もあわせて計測した．なお，従来の新しい索引を再構築するアプロー

[*7] https://inex.mmci.uni-saarland.de/

[*8] 閾値は後述の 6.1 節で設定したものと同じ値を用いた．

表 1　索引の差分更新による検索精度への影響
Table 1　Effects of incremental index updating on iP[.01] and MAiP.

初期文書割合 (%)	iP[.01]	MAiP	更新時間 (h)	総処理時間 (h)
10	.589	.168	6.9	7.5
30	.618	.194	6.0	8.0
50	.644	.202	4.9	8.4
70	.648	.196	3.7	8.9
90	.652	.202	2.4	9.5
from-scratch	.664	.213	7.7	7.7

チを *from-scratch* とする．これは初期文書として 100%の文書を持つことと同義である．

表 1 の結果より，iP[.01] と MAiP いずれも初期索引の割合が小さいほど検索精度が低下することが確認された．その原因として，大域的重み算出に用いる文書数が十分でない場合には，正確な値を算出できないためであると考えられる．適切な差分更新を実現するためには，不正確な大域的重みの問題を解決する必要がある．

ここで，索引の更新アプローチごとの更新処理時間について述べる．初期文書と更新文書がそれぞれ 50%であったとすると，差分更新アプローチでは更新に 4.9 時間要する．それに対して再構築アプローチでは 7.7 時間費やした．つまり，差分更新を行うことで，総処理時間は増加するものの，更新時間は短縮された．この結果から，文書の更新を速やかに検索システムに反映させるうえでは差分更新が効果的であるといえる．また，初期文書の割合が大きい場合ほど更新性能が低下した理由は，1 度に大量のタプルを索引付けする場合には効率的に索引の構築が可能だが，少量のタプルを索引へ差分更新することで各タプルの更新箇所の探索や，索引自体の物理的なデータ構造の変化が必要となるためである．

5.2　正確な大域的重み推定手法

大域的重みは文書集合全体から得られる統計量であるために，十分な量の文書が蓄積されていなければ正確な値を算出することができない．それに対して，新しい文書が追加されるたびに大域的重みの更新を行うことで緩やかに正確な大域的重みへ推移させることが可能であると考えられる．これ自体は妥当な考え方ではあるものの，正確な大域的重みを算出するにあたり，十分な量の文書がつねに追加されるとは限らない．さらに，文書の持つ統計量が大きく変動した場合には，一刻も早くその変化に対応しなければならない．これらをふまえ，その時点で蓄積されている文書から，可能な限り正確な大域的重みを推定することが可能な枠組みを設ける必要がある．

そこで本論文では，大域的重みの算出において path 式ごとに統計量を算出する枠組みから拡張し，類似 path 式を 1 つの属性として統合することで，各属性に分類される部分文書の量を増加させ正確な大域的重みを推測する．これを実現するため，我々が提案した path 式統合手法 [33] を利用する．この手法は本来，出現頻度の低い path 式を持つ部分文書に対しても正しく大域的重みを算出するためのものである．母集団の標本数が不足しているために十分な統計量を得られないという点では，問題点は同一であるため，path 式統合手法は今回の問題にも有効である．

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

図 8　path 式の例
Fig. 8　Examples of path expression.

article
sec
emp

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

図 9　ST 法による分類結果
Fig. 9　Classification result of ST method.

文献 [34] では，構造化文書中のタグは大別して，1) 独立したデータを囲んでいるタグと，2) 文を途中で切ることがあるタグ，と定義している．本論文では，1) のタグとは構造を表すタグであり，`article` タグや `sec` タグなどが該当すると見なす．また，2) のタグとは修飾や属性，特定のコンテンツを表すタグであり，`emp` タグ，`person` タグ，`table` タグなどが該当すると見なす．部分文書の粒度の観点では構造を表すタグがより重要な情報を含むが，ユーザが自らタグの定義が可能な XML 文書においてはタグの分類を行うことが困難である．さらに，これら 2 種類のタグはその出現においてそれぞれ独立関係にあると考えられるため，あるタグの組合せから複数個の path 式が生成される可能性がある．それらの path 式が持つセマンティクスは同等である，もしくは何らかの類似性を持つと考えられるため，これらの path 式を区別して扱うことは必ずしも適切とはいいきれない．このような理由から path 式を統合する際にタグの出現順序や出現頻度に着目する．

path 式統合時によるコストに関し，適用時には path 式中のタグの出現順序や出現頻度の計算のみであるので，索引の更新速度への影響は無視できる程度である．

以降，3 種類の path 式統合について論じる．

5.2.1　Set of Tags（ST）法

ST 法は，path 式中のタグの出現順序と出現頻度の観点から path 式の制約を緩和する．具体的には，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 式を同じ属性として分類する．

図 8 の path 式の統合結果を**図 9** に示す．path 式 p_1，p_2，p_3 はいずれも `article` タグ，`sec` タグ，`emp` タグから構成されるために統合される．したがって，p_1，p_2，p_3

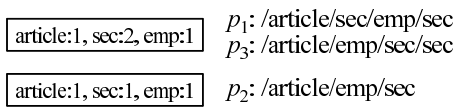

図 10　BT 法による分類
Fig. 10　Classification result of BT method.

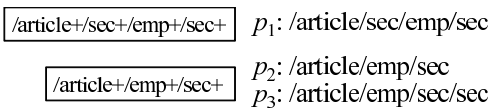

図 11　OT 法による分類
Fig. 11　Classification result of OT method.

はまとめられて大域的重みが算出される．

5.2.2　Bag of Tags（BT）法

BT 法では path 式中のタグの出現順序の観点から path 式の制約を緩和する．すなわち，bag-of-words [28] の考え方と同様に，タグの出現順序を考慮せず，タグ名とその出現頻度のみに着目する．

図 10 は BT 法による図 8 の path 式統合結果である．path 式を統合するにあたり，まず各 path 式中のタグの名前とその頻度を列挙する．次に，各タグの出現頻度が等しい path 式の統合を行う．その結果，p_1 と p_3 はいずれも `article` タグが 1 度，`sec` タグが 2 度，`emp` タグが 1 度出現しているために統合される．

5.2.3　Order of Tags（OT）法

OT 法による path 式統合では，path 式中で連続して出現するタグの出現頻度の観点から path 式の制約を緩和する．構造化文書中において表やリスト構造が定義されることは一般的であり，表やリスト中のデータが入れ子状の構造中で定義されることもまた頻繁に行われることである．そのため，表やリスト中の各データを定義する `col` タグなどは，path 式中に同一のタグが複数回連続で出現することが起こりうる．このような状況において入れ子構造の深さそのものが重要な情報を持っているわけではなく，表やリストが定義されていること自体が重要であるため，タグの出現頻度が異なる場合にも path 式の持つ属性は大きく変わらないとのアイデアに沿い，OT 法では path 式中で連続して出現するタグを 1 つに集約して path 式統合を行う．

図 8 の path 式の OT 法による統合結果は**図 11** のとおりである．p_3 に含まれる `sec` タグは連続して出現しているために，これらのタグは 1 つに集約される．その結果，p_2 と p_3 はいずれも 1 度以上 `article` タグが出現した後に `sec` タグ，さらに `emp` タグが出現しているために統合される．

6.　評価実験

提案する 2 つのフィルタと path 式統合の有用性を確認するために，検索精度と索引の差分更新性能を評価する．

フィルタと path 式統合手法を適用しない，単純な差分更新手法を *simple* 手法とする．提案手法には複数のバリエーションが存在し，大域的重みの算出方法はデフォルトの path 式ベースの手法，ST 法，BT 法，OT 法の 4 種類である．同様に，部分文書フィルタには 3 種類のパラメータ，索引語数の閾値 τ_{el}，path 式の深度の閾値 τ_{depth}，Zipf の法則による希少な path 式の頻度の閾値 τ_{el} が存在する．また，部分文書フィルタの有用性の確認のために，ランダムに部分文書を更新対象から除外する手法の評価も行う．索引語フィルタでは更新対象から除外する索引語の重みの閾値 τ_{tw} を設定する．新しい索引を再構築する *from-scratch* は差分更新ではないため厳密には比較不能であるが，従来の索引再構築手法との比較のために，実験ではフィルタおよび path 式統合手法を用いずに構築した *from-scratch* での結果も記載する．

表 2　τ_{el} の検索精度への影響
Table 2　Effects of τ_{el} on iP[.01].

τ_{el}	10	15	20	25	30	35
iP[.01]	.633	.664	.640	.639	.640	.617

本章の評価実験では 2 種類の文書集合，すなわち，文書集合内のトピックがほとんど変動しない静的統計量の文書集合と，文書集合内のトピックが大幅に変化する動的統計量の文書集合に対して評価を行う．前者の実験では限られた数の文書から正確な大域的重みを推測することが可能であるのかどうかを確認する．後者は，トピックの出現や動的変化にともなって大域的重みが変化した場合にも，正確な大域的重みを速やかに推測可能であるのかどうかを確認する．

実験の手順は次のとおりである．まず，部分文書フィルタと索引語フィルタの閾値を設定するための実験を行う．次に，静的統計量の文書集合に対して，提案手法の有用性を検証する．このとき，初期文書としては全体の 50%の文書を用いる．続いて，動的統計量の文書集合に対して提案手法の有用性を確認する．この実験では，文書の追加と削除を組み合わせた場合の評価を行う．

6.1　部分文書フィルタと索引語フィルタの閾値設定

部分文書フィルタはきわめて少ない索引語を持つ部分文書，きわめて深い path 式を持つ部分文書，希少な path 式を持つ部分文書を選別する．本節では，それぞれの閾値を設定するための実験を行う．

from-scratch によって構築した索引に対して検索を行い，検索結果を決定する際に閾値 τ_{el} よりも短い文書長の部分文書を除外した場合の iP[.01] を**表 2** に示す．その結果から，最も検索精度の高くなる $\tau_{el} = 15$ を設定する．

表 3 の All は，文書集合全体に含まれる部分文書のうち，path 式の深さが τ_{depth} の値以下の部分文書の割合である．我々の高精度検索に関する研究 [27] によって得られ

表 3　path 式の深さと部分文書の割合
Table 3　Depth of elements and their ratios of total.

path 式の深さ	3	4	5	6	7
All	.19	.44	.69	.88	.96
Top	.69	.89	.97	.99	1.00

表 4　n の変動による検索精度への影響
Table 4　Effects of n on iP[.01].

n	no filter	1500	5000	10000	30000
iP[.01]	.639	.629	.631	.641	.635

た検索結果のうち，上位 1,500 件以内の部分文書のみを対象とした際の値も Top として掲載する．文書集合全体と検索結果上位では，その部分文書の深さに違いが見られたため，うまく閾値を設定することで検索精度を低下させずに不要な部分文書を除外することができると考えられる．検索結果上位の部分文書のうちのほとんどが深さ 6 以下の部分文書であるため，深さが 6 以上の path 式を持つ部分文書を除外すべく $\tau_{depth} = 6$ を設定する．

次に Zipf の法則により中頻度の path 式の閾値 τ_{Zipf} を設定する．前述の τ_{el} と τ_{depth} は文書集合を通してつねに有効な値であるが，希少な path 式の種類はその時点までに蓄積された文書の影響が大きいため，τ_{Zipf} の算出には初期文書を用いる．初期文書中の path 式の頻度を列挙して式 (3) にあてはめたところ，中頻度の目安の値は 166.3 となったため，$\tau_{Zipf} = 167$ を設定し，167 回以上出現した path 式のみを更新対象として扱う．

索引語フィルタは，閾値 τ_{tw} よりも重みの小さな索引語を更新対象から省く．閾値を設定するために行った実験の結果を**表 4** に示す．τ_{tw} の $n = 10000$ のとき，すなわち，各タグ–索引語ペアにおいて 10,000 番目に大きな索引語の重みを閾値として設定した場合に検索精度が最大となったために，これを閾値として設定する．

6.2　静的統計量の文書集合における評価実験

from-scratch，*simple*，ならびに提案手法の各バリエーションの，1 文書あたりの平均差分更新時間，索引のディスクサイズ，iP[.01]，MAiP の値を**表 5** に掲載する．以降，すべての実験のあらゆる手法において，検索結果から索引語数 15 未満の部分文書を除外（閾値 τ_{el} と同値）した検索精度を測定している．なお，*from-scratch* アプローチの索引更新時間は差分更新時間ではなく，索引の再構築時間を文書数で割ったものを表す．

6.2.1　path 式統合手法に関する評価実験

まず，検索精度向上を目的とした path 式統合手法について述べる．*simple* の値と比較して，ST 法，BT 法，OT 法はいずれも iP[.01] の精度が向上した．このとき，文書の更新が発生しない状況において行った path 式統合手法の精度向上率 [33] よりも，本実験における精度向上率が高

表 5　提案手法の効果
Table 5　Results of our methods.

run ID	索引更新時間 (ms/doc)	索引サイズ (GB)	iP[.01]	MAiP
from-scratch	(42.1)	111	.664	.213
simple	53.4	111	.639	.200
rnd_flt（10%）	49.8	107	.635	.168
rnd_flt（20%）	45.1	100	.603	.154
rnd_flt（30%）	41.2	95	.612	.148
rnd_flt（40%）	37.2	92	.612	.141
rnd_flt（50%）	33.8	88	.594	.137
rnd_flt（60%）	30.0	77	.569	.137
ST 法	53.3	111	.655	.199
BT 法	53.2	111	.652	.206
OT 法	53.2	111	.653	.207
τ_{el}	45.5	97	.646	.202
τ_{depth}	49.1	106	.651	.204
τ_{Zipf}	51.1	109	.649	.198
elem filter	40.1	94	.651	.204
term filter	48.8	104	.641	.196
two filters	39.3	89	.652	.201
ST_filters	40.1	88	.662	.204

くなった．これは，すべての文書を利用可能な状態では，正確な大域的重みを算出するうえで十分な量の部分文書を確保できていない path 式の割合がそれほど高くなかったのに対して，索引の差分更新時においては，正確な大域的重みを算出するうえで十分な量の部分文書を確保できない path 式の割合が高かったため，より効果が高くなったものと考えられる．

INEX の公式尺度 iP[.01] においては，正確な大域的重みを算出するうえで ST 法が最も効果的であり，*simple* と比較して 2.57%高い検索精度を示した．ただし，MAiP の値については ST 法は *simple* と同等であるが，BT 法と OT 法では向上した．また，path 式統合手法を適用した場合の更新性能は *simple* と同程度であった．

6.2.2　更新コスト削減のためのフィルタに関する評価実験

続いて，索引の差分更新コスト削減のためのフィルタに関する実験結果について述べる．τ_{el}，τ_{depth}，τ_{Zipf} はそれぞれ *simple* と比較して iP[.01] の精度低下を抑制しつつ，更新コストの削減に貢献した．これら 3 つのパラメータの組合せを検証した結果，すべてのパラメータを利用した場合に最も効果的であったため，部分文書フィルタ（elem filter）ではすべてのパラメータを利用した．結果的に部分文書フィルタでは *simple* と比較して更新速度が 23.6%向上し，検索精度も多少向上した．

ここで，提案する部分文書フィルタの有効性を検証するため，更新対象の部分文書をランダムに除外するランダムフィルタ（rnd_flt）による計測も行った．除外する部分文書の割合を 10〜60%の範囲で計測した結果，除外する部分文書の割合が大きいほど更新速度が向上し索引のディスク

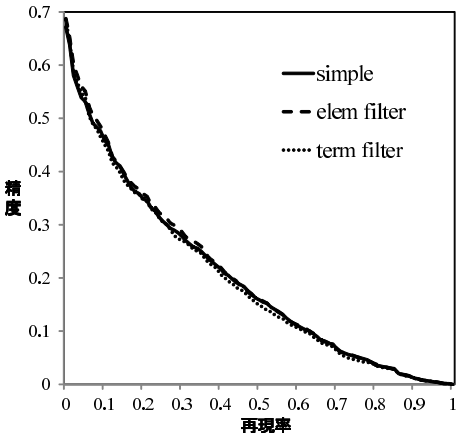

図 12　フィルタによる再現率への影響
Fig. 12　Effects of element and term filters on search accuracies.

サイズも減少したが，検索精度が低下するという結果が得られた．部分文書フィルタと同程度の更新速度を示したランダムフィルタ（30%削減および 40%削減）と比較したところ，部分文書フィルタではより高精度な検索精度を示し，その有効性は明らかである．

同様に，索引語フィルタ（term filter）も検索精度の低下を軽減させつつ，更新速度は 6.70%向上した．これらをふまえ，2 つのフィルタを併用した場合（two filters）の結果も測定する．部分文書フィルタに 3 つすべてのパラメータを用いた場合に最も効果的であった．このとき，更新速度は 26.3%向上し，検索精度は *simple* と比較し 2.14%向上した．また，MAiP の値はいずれの手法との組合せにおいても *simple* と同程度を示した．

ここで，フィルタによって除外された部分文書や索引語によって再現率の低下が引き起こされるのかどうか検証を行うため，*simple* と部分文書フィルタ，索引語フィルタの再現率–精度曲線を**図 12** に示す．その結果，3 つの手法いずれもおおむね同様の推移を行ったことから，フィルタによる再現率への悪影響はほとんどないといえる．なお，再現率が 15%あたりまでは部分文書フィルタ手法による精度が *simple* より高く，逆に再現率 50〜70%周辺で逆転が起こるが，その後は似た振舞いをする．また，索引語フィルタは再現率 6%までは *simple* よりも高精度を示し，それ以降は精度が逆転した．

6.2.3　path 式統合手法とフィルタの組合せの評価実験

path 式統合手法によって検索精度の向上が確認され，フィルタを利用することで更新性能の向上が確認されたが，これらの手法を組み合わせて利用することで高精度かつ高い更新性能を実現することが可能かどうかの確認を行う．ST 法と two filters の組合せ（ST_filters）では，更新性能は two filters と比較して多少低下したものの（*simple* より 24.9%上昇），iP[.01] の検索精度においてはいずれの手法よりも高くなった（*simple* より 3.73%上昇）．MAiP の値は *simple* と同程度であった．

表 6　INEX project 参加者の他システムとの精度比較
Table 6　Search accuracies compared with those of other INEX participants.

Team	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
ST_filters + 再構成	.6898	.6736	.5647	.4789	.2167
Renmin Univ. of China	.5969	.5969	.5815	.5439	.2486
Queensland Univ.	.6232	.6220	.5521	.4617	.2134
Univ. of Amsterdam	.6514	.6379	.5901	.5280	.2261

これらの結果から，提案手法を組み合わせることで検索精度と更新性能がともに向上することが明らかになった．しかしながら，提案手法の検索精度は文書の更新が発生しない状況（*from-scratch*）における検索精度よりも低下するという結果も確認された．そこで，追加として我々の高精度検索のための検索結果再構成手法 [30] をさらに適用することで検索結果にどのような影響を及ぼすのかを調査した．この再構成手法では，文書中の検索結果として最適な粒度の部分文書を特定し，有用な部分文書を上位にランキングするようにスコアリングを行う．再構成手法の適用の際には，各部分文書のスコアと各部分文書の先祖・子孫関係の情報のみを用いるために，本論文での提案手法への適用は容易である．

ST_filters へ再構成手法を適用した結果，iP[.01] と MAiP はそれぞれ 0.6736 と 0.2167 であり，*from-scratch* よりも高精度検索を実現することができた．この ST_filters へ再構成手法を適用させた手法と，他の INEX project 参加グループの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 6** 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 [15] との比較を行ったところ，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効な検索技術であると確認できた．特筆すべき点は，我々の提案手法のみが文書の更新にともなう高速な索引の更新が可能ということである．

クエリ処理速度に関しては，いずれの手法を用いた場合においても 1 クエリあたり 1〜2 秒で処理が可能であり，再構成手法の適用を行う際には 1 クエリあたりさらに 0.5 秒程度必要であった．つまり，提案手法は 1 つのクエリに対してたかだか 2.5 秒程度で検索結果を提示することが可能であり，この程度の時間は検索システム利用者にとって許容範囲であると考えられる．

6.3　動的統計量の文書集合における評価実験

これまでの評価実験では，文書集合中のトピックは変動せず，文書集合の持つ統計量は一定である状況を想定していた．しかし，特定のトピックの文書が急激に増加して，文書集合の持つ統計量が大幅に変動することがありうる．このような場合も考慮し，統計量の異なる初期文書と更新

表 7　カテゴリとクエリの個数
Table 7　Categories and their number of queries/keywords.

カテゴリ名	クエリ数	キーワード数
Technology and applied sciences	18	54
Culture and the arts	20	51
Natural and physical sciences	9	24
Society and social sciences	4	13
History and events	4	11
Philosophy and thinking	3	8
General reference	3	7
Health and fitness	2	7
People and self	3	6
Geography and places	2	5
Mathematics and logic	0	0
Religion and belief systems	0	0

文書を用いることで統計量が変動する状況を設定して提案手法の有用性を確認する．また，本実験では，提案システムが文書の追加以外の更新処理に対しても，効率的に索引更新とクエリ処理を行うことが可能であるかの検証を行うため，文書の追加と削除を組み合わせた索引更新を行う．

この実験で用いる文書集合を作成する手順を以下に示す．

(1) あるトピックに関する文書を取り出す．
(2) その他の文書から初期索引を構築する．
(3) (1) で取り出された文書を用いて索引の差分更新を行いつつ，(2) で索引付けられた文書を索引から削除する．

まず，トピックに基づく文書集合の分類方法について述べる．ある文書が特定のトピックに属するかどうか判断するために，Wikipedia 内のカテゴリを利用する．Wikipedia には大小さまざまなカテゴリが存在し，主要なカテゴリとしては**表 7** に記載する 12 のカテゴリが存在する．まず，68 個のクエリをこの 12 個のカテゴリに手作業による分類を行った．各クエリは 1〜5 個程度のキーワードを含むために，結果としてカテゴリごとにキーワード集合を取り出すことができる．“Technology and applied sciences”（*technology*）カテゴリは比較的多くのクエリとキーワードを含むために，後ほどの実験ではこのカテゴリを利用する．文書集合の分類に関して，ある文書がカテゴリに属するキーワードを含めば，文書はそのカテゴリに属するとする．以下，接辞処理された後のキーワードを列挙する．

technology カテゴリ：aircraft, applied, automobil, aviat, bay, bletchlei, break, car, code, colossu, compani, comput, databas, detect, engin, expert, file, filter, format, graphic, imag, inform, instal, intrus, invent, java, languag, linux, manag, mechan, metadata, mine, motor, museum, network, nikola, open, oper, park, patent, program, raid, record, retriev, rotari, secur, social, sourc, storag, system, tata, tesla, virtual, wireless

表 8　動的文書集合による評価実験
Table 8　Experimental results using the document set with dynamic statistics.

索引付けられた文書数	*technology*（iP[.01]）		
（$\times 10^4$ documents）	*from-scratch*	*simple*	ST_filters
38（25%更新）	.585	.524	.532
47（50%更新）	.575	.525	.548
57（75%更新）	.620	.546	.563
66（100%更新）	.624	.578	.592

動的に統計量が変化するトピックに関する評価を行うことが目的であるために，評価実験では *technology* カテゴリに属するクエリのみを対象とする．*technology* カテゴリに分類された文書は約 38 万文書，それ以外の文書は約 28 万文書であった．そのため，約 28 万の文書から初期索引を構築し，その後の索引更新によってさらに 38 万文書を索引付けする．この実験では，索引の差分更新の途中経過の 4 段階（文書の更新がそれぞれ 25%，50%，75%，100%完了した時点）においての検索精度の計測を行った．また，文書の削除が起こった際には Version list を利用して削除を反映させ，索引から該当タプルの削除は行わない．

表 8 に *from-scratch* と *simple*，そして提案手法である ST_filters の iP[.01] の結果を記載する．ST_filters は *from-scratch* と比較した場合には検索精度の低下が確認されたものの，いずれの時点においても *simple* より高精度を実現した．また，同様に Culture and the arts カテゴリに対する評価実験においても提案手法を用いることで同様の結果が得られ，文書更新に追従して速やかな検索精度の向上が確認された．索引の更新初期においてはカテゴリに属する索引語はほとんど出現していないために大域的重みを正確に算出できず，更新にともなってカテゴリに属する索引語が増加するため，大域的重みの正確性と検索精度が向上する．その過程において，提案する path 式統合手法を用いれば蓄積される索引語数が少ない時点からより正確な大域的重みを推定でき，提案手法はより高精度な検索を実現することができたと判断できる．これらの結果から，新たなトピックが出現する状況下においても提案手法を利用することでより正確な検索が可能と示唆された．

なお，索引更新処理中の，文書の削除時の Version list への DID および VID の登録におけるコストはきわめて小さく，無視できる程度のコストであった．クエリ処理に関しても，文書の削除が起こらない環境での検索では 1 クエリあたりの平均処理速度は 1.89 秒だったのに対して，タプルの読み込みごとに Version list を確認し，VID の値によって処理を変更した場合には 2.02 秒と，両者の処理時間にそれほど大きな差は生じなかった．これらの結果から，本論文で提案した文書の更新を考慮した XML 部分文書検索では，Version list を利用することで，高精度かつ高速なクエリ処理を維持しつつ，文書の削除にも即座に対応すること

が可能であるといえる．

7. おわりに

本論文では，高精度な XML 情報検索を維持しつつ，文書の更新に高速に対応する手法の提案を行った．目標として，(1) 文書の更新が発生すれば，極力高速に検索システムに反映させることと，(2) 正確な大域的重みを推定することで可能な限り検索精度を維持することを目指した．

文書の更新の高速な反映のため，既存の XML 部分文書検索システムを拡張させることで差分更新機能を付与し，さらに不要な更新対象選定のためのフィルタの提案と，path 式統合手法による正確な大域的重み推定を行った．

静的統計量の文書集合に対する評価実験の結果，提案手法を用いることで，*simple* 手法と比較し，検索精度を最大 4%向上させつつ，索引の差分更新処理を 25%高速化した．さらに検索結果の再構成手法を適用した場合には，文書の更新を考慮しないシステムよりも検索精度を向上させることを示した．また，動的統計量の文書集合に対しても提案手法により検索精度が改善され，追加された文書が少量の時点から検索精度を向上させた．

今後の課題として，クエリ処理速度の向上のためのより効率的な Top-k アルゴリズムの適用や，更新速度向上のための並列処理による XML 文書解析の適用などが考えられる．また，構造化文書の 1 つである Web 文書への提案手法の適用や，提案手法への Web 検索において用いられるリンク解析技術などの適用の検討なども今後の課題である．

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．

参考文献

[1] Kampsa, J. and Koolen, M.: On the Relation between Relevant Passages and XML Document Structure, *Proc. SIGIR 2007 Workshop on Focused Retrieval*, pp.28–32 (2007).

[2] Liu, F., Yu, C., Meng, W. and Chowdhury, A.: Effective Keyword search in Relational Databases, *Proc. ACM SIGMOD* (2006).

[3] Robertson, S., Zaragoza, H. and Taylor, M.: Simple BM25 Extension to Multiple Weighted Fields, *Proc. 13th ACM CIKM* (2004).

[4] Liu, W., Robertson, S. and Macfarlane, A.: Field-Weighted XML Retrieval Based on BM25, *Formal Proc. INEX 2005 Workshop*, LNCS, Vol.3977 (2006).

[5] Ogilvie, P. and Callan, J.: Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval, *Formal Proc. INEX 2005 Workshop*, LNCS, Vol.3977 (2006).

[6] Schmidt, A., Kersten, M. and Windhouwer, M.: Querying XML Documents Made Easy: Nearest Concept Queries, *Proc. 17th ICDE*, p.321, IEEE (2001).

[7] Xu, Y. and Papakonstaninou, Y.: Efficient Keyword Search for Smallest LCAs in XML Databases, *Proc. ACM SIGMOD*, pp.527–538, ACM (2005).

[8] Li, G., Feng, J., Wang, J. and Zhou, L.: Effective Keyword Search for Valuable LCAs over Xml Documents, *Proc. 16th ACM CIKM*, pp.31–40 (2007).

[9] Hristidis, V. and Koudas, N.: Keyword Proximity Search in XML Trees, *IEEE Trans. Knowledge and Data Engineering* (*TKDE*), Vol.18, No.4, pp.525–539 (2006).

[10] Theobald, M., Bast, H., Majumdar, D., Schenkel, R. and Weikum, G.: TopX: Efficient and Versatile Top-k Query Processing for Semistructured Data, *The VLDB Journal*, Vol.17, No.1, pp.81–115 (2008).

[11] Trotman, A., Jia, X.-F. and Geva, S.: Fast and Effective Focused Retrieval, *Formal Proc. INEX 2009 Workshop*, LNCS, Vol.6203 (2010).

[12] Liu, Z. and Chen, Y.: Identifying Meaningful Return Information for XML Keyword Search, *Proc. ACM SIGMOD*, pp.329–340 (2007).

[13] Huang, Y., Liu, Z. and Chen, Y.: Query Biased Snippet Generation in XML Search, *Proc. ACM SIGMOD*, pp.315–326 (2008).

[14] Blanken, H., Grabs, T., Schek, H.-J., Schenkel, R. and Weikum, G.: *Intelligent Search on XML Data: Applications, Languages, Models, Implementations, and Benchmarks*, LNCS, Vol.2818, Springer-Verlag (2003).

[15] Kamps, J., Geva, S., Trotman, A., Woodley, A. and Koolen, M.: Overview of the INEX 2008 Ad Hoc Track, *INEX 2008 Workshop Pre-proceedings*, pp.1–28 (2008).

[16] Trotman, A. and Sigurbjörnsson, B.: Narrowed Extended XPath I (NEXI), *Formal Proc. INEX 2004 Workshop*, LNCS, Vol.3493 (2005).

[17] Piwowarski, B. and Gallinari, P.: 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).

[18] 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).

[19] Chen, F., Feng, X., Ré, C. and Wang, M.: Optimizing Statistical Information Extraction Programs Over Evolving Text, *Proc. 28th IEEE ICDE* (2012).

[20] Neumann, T. and Weikum, G.: xRDF3X: Fast Querying, High Update Rates, and Consistency for RDF Databases, *Proc. 36th VLDB*, pp.256–263 (2010).

[21] Ren, C., Lo, E., Kao, B., Zhu, X. and Cheng, R.: On Querying Historical Evolving Graph Sequences, *Proc. 37th VLDB* (2011).

[22] Tomasic, A., García-Molina, H. and Shoens, K.: Incremental Updates of Inverted Lists for Text Document Retrieval, *Proc. ACM SIGMOD* (1994).

[23] Lester, N., Zobel, J. and Williams, H.E.: In-Place versus Re-Build versus Re-Merge: Index Maintenance Strategies for Text Retrieval Systems, *Proc. 27th Australasian conference on Computer Science* (2004).

[24] Margaritis, G. and Anastasiadis, S.V.: Low-cost Management of Inverted Files for Online Full-Text Search, *Proc. 18th ACM CIKM* (2009).

[25] Geva, S., Kamps, J. and Trotman, A.: *Advances in Focused Retrieval*, Springer Berlin (2009).

[26] 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).

[27] Keyaki, A., Hatano, K. and Miyazaki, J.: Result Re-

construction Approach for More Effective XML Element Search', *International Journal of Web Information Systems* (*IJWIS*), Vol.7, No.4, pp.360–380 (2011).
[28] Manning, C.D., Raghavan, P. and Schutze, H.: *Introduction to Information Retrieval*, pp.157–159, Cambridge University Press (2008).
[29] 田邊　翼，清水敏之，吉川正俊：XML キーワード検索における要素の特性を考慮した検索結果の構築，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 (2012).
[30] 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).
[31] Maron, M.E.: Automatic Indexing: An Experimental Inquiry, *J. ACM*, Vol.8, pp.404–417 (1961).
[32] 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).
[33] Keyaki, A., Hatano, K. and Miyazaki, J.: Relaxed Global Term Weights for XML Element Search, *Formal Proc. INEX 2010 Workshop*, LNCS, Vol.6932 (2011).
[34] 徳田隆志，田島敬史：XML タグの文書構造上の役割に関する自動分類，*DBSJ Journal*, Vol.8, No.1, pp.1–6 (2009).

欅 惇志　（学生会員）

2009 年同志社大学文化情報学部文化情報学科卒業．2011 年同大学大学院文化情報学研究科文化情報学専攻博士前期課程修了．現在，奈良先端科学技術大学院大学情報科学研究科情報科学専攻博士後期課程在学中．2012 年より日本学術振興会特別研究員（DC2）．高精度かつ高性能な XML 情報検索に関する研究等に従事．2013 年第 5 回データ工学と情報マネジメントに関するフォーラム（DEIM 2013）にて最優秀論文賞（Ph.D. セッション）受賞．電子情報通信学会，日本データベース学会，ACM 会各員．

宮崎 純　（正会員）

奈良先端科学技術大学院大学情報科学研究科准教授．博士（情報科学）．1992 年東京工業大学工学部情報工学科卒業．1997 年北陸先端科学技術大学院大学情報科学研究科博士後期課程修了．同大学助手を経て，2003 年より現職．2000～2001 年テキサス大学アーリントン校客員研究員．2003～2007 年科学技術振興機構さきがけ研究員．高性能・高機能データベースならびに情報検索の研究に従事．電子情報通信学会，日本データベース学会，ヒューマンインタフェース学会，ACM，IEEE Computer Society 各会員．

波多野 賢治　（正会員）

同志社大学文化情報学部准教授．博士（工学）．1995 年神戸大学工学部計測工学科卒業．1999 年同大学大学院自然科学研究科博士後期課程修了．同年日本学術振興会未来開拓学術研究事業研究員，奈良先端科学技術大学院大学情報科学研究科助手．2005～2006 年米国 AT&T Labs-Research 客員研究員．2006 年同志社大学文化情報学部専任講師，2008 年より現職．2007 年電子情報通信学会論文賞受賞．Web 情報検索，XML データベース等の研究に従事．電子情報通信学会，日本データベース学会，ACM，IEEE Computer Society 各会員．

山本 豪志朗　（正会員）

2004 年大阪大学基礎工学部システム科学科卒業．2006 年同大学大学院基礎工学研究科博士前期課程修了．2008～2009 年日本学術振興会特別研究員．2009 年大阪大学大学院基礎工学研究科博士後期課程修了．同年オウル大学客員研究員．同年岡山大学大学院自然科学研究科助教．2011 年より奈良先端科学技術大学院大学助教．2012～2013 年オウル大学客員研究教授．博士（工学）．ヒューマンコンピュータインタラクション，拡張現実感の研究に従事．計測自動制御学会，日本バーチャルリアリティ学会各会員．

武富 貴史　（正会員）

2006 年佐世保高等工業専門学校専攻科電気電子工学専攻卒業．2011 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．現在，同大学情報科学研究科助教．博士（工学）．電子情報通信学会，日本バーチャルリアリティ学会，IEEE 各会員．

加藤 博一　（正会員）

1986 年大阪大学基礎工学部制御工学科卒業．1988 年同大学大学院修士課程修了．1989 年同大学基礎工学部助手．1996 年講師．1998 年ワシントン大学客員研究員．1999 年広島市立大学情報科学部助教授．2003 年大阪大学大学院基礎工学研究科助教授．2007 年より奈良先端科学技術大学院大学情報科学研究科教授．博士（工学）．拡張現実感，ヒューマンインタフェースの教育研究に従事．日本バーチャルリアリティ学会，ヒューマンインタフェース学会，電子情報通信学会，ACM，IEEE 等各会員．

（担当編集委員　堀井 洋）