情報処理学会論文誌　データベース　Vol. 1　No. 1　1–12 (June 2008)

XML 情報検索における利得と閲覧コストに基づく検索結果の取得と評価

清　水　敏　之[†1]　　吉　川　正　俊[†1]

XML 情報検索システムは XML 文書中の要素を検索単位に用い，問合せに対して関連する要素を特定する．我々は，システムが要素を取得して利用者に提示することで，利用者はその要素を閲覧するコストを支払い，問合せに関連する情報を利得として取得すると考えた．閲覧コストを導入することで検索結果の量を制御し，XML 情報検索で問題となる検索結果中の入れ子を適切に扱う．利用者は支払ってもよい閲覧コストの合計量を指定し，それに対してシステムは柔軟に入れ子しない要素集合を取得する．我々はこの指定された閲覧コストの中で利得を最大にする問題を定式化した．この問題は NP 困難でるため，貪欲解法を考え，さらにその理論的な上界値を求めた．上界値を利用することでシステムの絶対評価が可能となる．我々は INEX テストコレクションを利用してシステムを実装し，上界値の精度を確認した．

Retrieval and Evaluation for XML-IR Based on Benefit and Reading Effort

Toshiyuki Shimizu[†1] and Masatoshi Yoshikawa[†1]

XML information retrieval (XML-IR) systems search for relevant elements in XML documents for given queries. In our scheme, we suppose we pay reading effort and obtain benefit for the retrieved relevant element. We can control the total output size of result elements and handle nesting elements by introducing the concepts of benefit and reading effort. The system flexibly retrieves non-overlapping elements within the reading effort specified by users. We formalized the problem of maximizing the benefit for a given reading effort, and found no unique algorithm as the optimal and practical solution for the problem. We therefore decided to use an upper bound of the benefit that is obtained by the system for system evaluation, and proposed evaluation metrics based on the upper bound changing the amount of reading effort. We confirmed the effectiveness of the upper bound and that its quality was sufficient for most queries.

1. はじめに

大量の XML 文書の中から，ある話題に関して知りたいとき，キーワードを用いて検索を行い，関連する部分のみを取得することが考えられる．XML 情報検索では，一般に XML 文書中の要素を検索の単位とし，問合せに関連する要素を取得する．たとえば，XML 化された論文の場合，問合せに関連する節や段落を取得することができる．

XML 情報検索システムの主な対象は，論文や本などを XML 化した，いわゆる文書指向 XML である．一般に，文書指向 XML は自然言語文による長いテキストを含んでおり，スキーマが複雑である傾向があるため，キーワードを用いた検索の要求が強いと考えられる．論理構造を持つ文書は一般的に XML でも表現することが可能であるため，本研究では標準的に利用可能である XML を対象とするが，構造化文書一般に関しても技術が応用可能であると思われる．なお，DBLP 文献データベース[1] のようなデータ指向 XML に対するキーワード検索としては，LCA（Lowest Common Ancestor）の概念を利用し，問合せキーワードをすべて含む最小の部分木を取得してくることが考えられている[2),3)]．

XML 情報検索システムを用いることで，利用者は関連度の高い部分のみを閲覧することができるが，一般に検索単位が要素であるため，検索結果中に入れ子が存在する可能性があり，どのように入れ子を扱うかは XML 情報検索の焦点の 1 つである．たとえば，ある問合せに対し，ある段落要素 e_p が非常に関連度が高く，その段落を含む節要素 e_s も e_p までではないものの関連している，といった状況がありうる．

このような状況において，単純にすべての要素を順序付けして取得する XML 情報検索システムは e_p と e_s をこの順で取得するが，その場合，利用者は e_s の内容を閲覧する際には e_p に相当する内容をすでに閲覧しており，システムは重複する内容を提示しているということができる．我々は，このような内容の重複は冗長であり，避けるべきであると考えている．この場合，e_p か e_s かどちらかのみを提示すべきであり，両方ともを提示すべきではない．そして，利用者の要求に応じてどちらの要素を提示するかを決めることを考えた．短い結果のみを閲覧して簡単に結果を把握できればよいと考える利用者には e_p を提示し，もっと深く問合せに関連する内容について知りたいと考えている利用者には e_s を提示することが望ましいと考えられる．単に固定の要素リストを検索結果として提示するのではな

†1 京都大学情報学研究科
Graduate School of Informatics, Kyoto University

く，利用者の要求に応じて検索結果を柔軟に変えて取得することを提案する．

また，XML 情報検索では検索結果として返ってくる要素の大きさは多様性に富み，大きな要素，たとえば根要素（文書全体）を返すこともあれば，小さな要素を返すこともある．そのため，あるシステムとそれとは別の他のシステムで同じ個数の要素が検索結果として返されていても，それらを閲覧するコストは大きく異なる場合がある．また，たとえば 10 個の要素が返されたとしても，その 10 個の要素を閲覧することがどれくらい大変か，10 個という個数からは分からない．利用者が検索結果として返される要素リストを上位から閲覧していくことを想定すると，上位 k 件の結果を取得する top-k 検索が重要であり，XML 情報検索における top-k 検索の研究[4),5)] もある．しかし，利用者は top-k 検索において k を与えても，それにより得られる要素集合に関し，閲覧する量を制御できない．我々は k という件数ではなく，検索結果要素に対してかけることのできる閲覧コストの総量を利用者が指定することで，検索結果の量を制御することを考えた．

検索結果の量を制御し，入れ子を適切に扱うため，我々は XML 情報検索システムは件数ではなく閲覧コストを明示的に扱うべきであると考えた．利用者は支払ってもよい閲覧コストの総量を閾値として指定し，システムは指定された閲覧コスト以内で読むことができる要素集合を取得する．利用者が結果要素を閲覧することでその要素から利得を得ると考え，システムはより多くの利得がある要素を取得する．繰り返し同じ内容を閲覧しても利得は増加しないと考えられ，より多くの利得を取得するために検索結果中の入れ子は除去される．入れ子する要素に関して，利用者は先祖要素を閲覧することでそのすべての子孫要素を閲覧していると見ることができる．

我々が想定する XML 情報検索システムは以下のようなものである．利用者は，キーワードを用いて問合せを生成し，システムに問い合わせる．同時に，どの程度の量を検索結果として受け入れることができるかを閲覧コストの閾値として指定する．システムはそれに対し，指定された閲覧コストの閾値以内で閲覧可能な要素集合を返す．返された検索結果に対し，利用者はさらに結果を閲覧したいと感じた場合は閲覧コストの閾値を増やし，逆にもっと絞った結果を閲覧したいと感じた場合は閲覧コストの閾値を減らす．閲覧コストの閾値に応じて，システムは検索結果の量を調整するが，いずれの閾値に対してもつねに入れ子による内容の重複のない要素集合を返す．

我々は，各要素に対して利得と閲覧コストが与えられた際に，与えられた閲覧コストに対して利得を最大化する問題を定式化した．我々の考える XML 情報検索システムは，利用者が入力する閲覧コストの閾値に対して，その閲覧コスト内で閲覧可能な関連要素をシステムが柔軟に取得するため，閲覧コストの閾値指定を c から c' へ増加させた際に c に対する検索結果は c' に対する検索結果に含まれていないかもしれない．我々は，利用者が閲覧コストの閾値を変化させて検索結果を閲覧することを考えると，閲覧コストをの閾値を増加させた際には検索結果に包含関係があるべきであると考え，検索結果の単調性を満たすようなシステムが実用的であると考えた．

我々が定式化したこの問題はナップサック問題[6)] の一種であり，最適解を見つけることは NP 困難であることを示す．また，最適解は一般的に検索結果の単調性を満たさないため，貪欲解法によってこの定式化した問題を扱うことにした．

閲覧コストに関しては，問合せには依存せず，要素の長さを用いて算出ができると考えられる．よって，システムの性能は要素の利得をどのように算出するかに依存する．我々は，利得と閲覧コストを用いた XML 情報検索システムの性能評価手法も提案する．評価のためには要素に対して人手で付与した実際の利得が利用できると考えた．実際の利得を用いて定式化した問題を解くことでシステムが到達可能な利得の理論的な上限値を得ることができ，実装したシステムが得ることができた利得がその上限値に対してどれくらいの割合であったかを観測することでシステムの評価ができる．指定する閲覧コストの閾値を変化させて，獲得できる利得の上限値に対する割合を観測する．上限値の取得は定式化した問題の最適解を算出することで得ることができるが，最適解を算出することは NP 困難であるため，最適解の上界値を用いることにした．我々の用いた上界値は INEX 2005[7)] のほとんどの問合せで十分な性能を与えることを示す．

2. 関 連 研 究

2.1 利得と閲覧コスト

XML 情報検索システムの評価手法として要素中での関連するテキストの長さを利用する HiXEval[8)] が提案されている．HiXEval は問合せに対して関連するテキストをより多く取得し，関連しないテキストをできるだけ取得しないようなシステムが良い評価となる．要素長を閲覧コスト，関連テキスト長を利得と見なすと HiXEval の考え方は我々のものと同様であると見ることができ，実際，我々は要素長を閲覧コストの算出に，関連テキスト長を利得の算出に利用することを考えている．

しかし，我々が考えている利得と閲覧コストは，3.1 節で示すとおり，兄弟要素間の情報の補完や，連続して閲覧する際の読みやすさを考えおり，この HiXEval のモデルを拡張している．また HiXEval は単に XML 情報検索の評価指標を提案するにとどまっているが，

我々は検索結果の取得手法に関しても考え，利用者の閲覧コストの指定に対してシステムが柔軟に関連要素を取得してくる検索モデルを提案している．

2.2　検索結果の取得

INEX 2005[7] では XML 情報検索システムの精度評価のために，要素を取得する 3 つの戦略を定義している．*Thorough* では単純にすべての要素を関連度順に取得する．*Thorough* では検索結果の要素中に入れ子により内容に重なりがある場合がある．*Focused* では重複を排除した検索結果を取得する．*Focused* では検索結果中の先祖子孫関係にある要素集合のうち，1 つの要素（*focused element*）のみしか結果に含めることができない．*Focussed* では冗長性を排除できるが，検索結果が固定された要素リストであり，取得された要素以外の要素の関連度は分からないため，XML 情報検索システムの利点を損なう可能性がある[9]．そして，同じ文脈の要素を連続して取得するために，同一の文書内の関連する要素をまとめて取得するのが *Fetch and browse* である．

INEX 2006[10] では *Relevant in context* と *Best in context* が追加された．*Relevant in context* は *Fetch and browse* と似た取得戦略であり，検索結果として得られる要素を文書ごとにまとめて取得する．*Fetch and browse* では，文書内において関連する要素を関連度順に取得するが，それに対し，*Relevant in context* では，その名のとおり，コンテキストの把握が重要視されるため，同じ文書内の結果要素は文書順に取得する．また，*Fetch and browse* では，すべての要素が取得対象になるが，*Relevant in context* では，冗長性を排除し，入れ子関係にある要素を同時に取得せず，*focused element* のリストを取得する．また，*Best in context* では文書ごとに読み始めるのに妥当な箇所（BEP: Best Entry Point）を 1 カ所のみ取得する戦略である．*Fetch and browse* や *Relevant in context* のように関連要素を文書ごとにまとめて取得する点も XML 情報検索を考えるうえで重要な点であるが，検索結果として問合せとの関連度順に並んでいる要素リストを得ることができればそれらを文書ごとにまとめて提示することは容易であるため，ここではそのような検索結果のグループ化は考えない．

INEX の検索戦略では XML 情報検索システムは固定された要素リストを返すことを考えている．そのような結果の取得はシステムの評価には重要であるが，我々は，実際の利用者の要求を考え，利用者の要求に応じて柔軟に検索結果を取得することを考えた．利用者が支払ってもよい閲覧コストの総量を指定し，システムは入れ子のない要素集合を柔軟に取得する．XML 情報検索において入れ子を考慮することは重要であるが，*Focused* 戦略は *focused element* のみを取得し，固定された *focused element* のリストを返すため，柔軟性に欠ける．*Focused* 戦略ではすでに取得した要素の先祖要素はけっして取得しない．

XML 情報検索の検索結果中の入れ子の扱いに関する研究としては，文献 9) があげられる．文献 9) では，ある要素が検索結果として出力されると，その子孫要素と先祖要素のスコアを再計算し，それ以降の結果を再ランキングする．しかし，検索結果は固定の要素リストである点は変わらず，また，閲覧コストのような考え方は導入されていない．

2.3　評 価 手 法

XML 情報検索システムに対する評価手法はいまだ確立されていないが，INEX 2005 では XCG（eXtended Cumulated Gain）に基づく評価手法が用いられている[11]．XCG に基づき，システム指向の評価尺度である ep/gr（effort-precision/gain-recall）やユーザ指向の評価尺度である nxCG（normalized extended Cumulated Gain）が利用されている．XCG に基づいたこれらの評価尺度では理想的なシステムに対して実システムが相対的にどれくらい優れているかを計測する．これらの評価尺度では件数が基準となり，たとえば上位 10 件の検索結果を取得した時点での理想値に対する累積利得の比などを計測するが，我々は件数ではなく閲覧コストを基にシステムを評価することを考えている．

INEX 2007 では HiXEval が評価手法として用いられている[8),12]．しかし，HiXEval では XCG において考えていた理想システムが存在せず，システム間の相対評価を行うことはできても，システムの絶対評価を行うことはできない．HiXEval を用いて得られた値が低くても，理想的なシステムを用いても低い値しか得ることができないかもしれず，システムが理想的なものと比較してどの程度良いのか分からないといえる．我々の場合では利得の上界値を比較対象に用いることでシステムの絶対評価を可能とする．

3.　検索結果の取得

3.1　利得と閲覧コスト

検索結果要素に包含関係がある場合，検索結果を閲覧していくと，すでに見たことのある内容が出現することがある．また，要素の大きさは多様性に富むため，同じ 1 件の結果でも利用者がその要素を読むのにどれくらい大変か分からない．我々は利得と閲覧コストを導入することでこれらの問題に対処した．本節では，利得と閲覧コストに関し，その特徴を議論する．

3.1.1　利　　得

ある問合せに対して，関連する要素を読むことにより利用者はその問合せに関する利得を得ると考える．本論文では，従来の情報検索や XML 情報検索にのっとり，単純化のため，

検索対象となる XML 文書は同じ内容を繰り返し述べないと仮定した．この仮定は，従来の情報検索システムが似たような内容を述べている別の文書を取得し，精度再現率による評価においても似たような文書それぞれを正解と見なして考えることが一般的であることを考えると妥当であると考えられる．この仮定に基づいて考えると，基本的には，ある要素の利得はその子要素集合の利得の和であると考えられる．ただし，まとめて読むことにより兄弟要素間で問合せに関する内容を補完しているといった状況も考えられるため，和よりも多少大きな利得を取得できることも考えられる．我々は，利得に関し以下のような特徴があると考えた．

仮定 1　利得の性質

ある要素の利得はその子要素集合の利得の和と等しいか，または大きい．

3.1.2　閲覧コスト

要素の閲覧コストはその要素を読むのにかかるコストである．算出には基本的に要素に含まれるテキスト長を利用するのが適切であると思われる．閲覧コストは問合せには依存せず，要素それ自体に応じて決まる．基本的には，ある要素の閲覧コストはその子要素の閲覧コストの和と考えられる．ただし，まとめて読むことにより同じ文脈の内容を続けて読むことができ，それぞれ個別に出されるよりも楽に読むことが可能であると思われるため，和よりも多少小さな閲覧コストとなることも考えられる．我々は，閲覧コストに関し以下のような特徴があると考えた．

仮定 2　閲覧コストの性質

ある要素の閲覧コストはその子要素集合の閲覧コストの和と等しいか，または小さい．

3.2　b/e グラフ

我々が考える XML 情報検索システムは，問合せに対して検索対象の XML 文書集合中の各要素の利得を算出し，検索結果を取得する．利用者が閲覧コストの閾値 c を指定し，システムは指定された閲覧コスト内で読むことが可能なより多くの利得を持つ要素を取得する．

まったく同じ内容を繰り返し閲覧しても利得は増えないと考えられるため，仮に検索結果中に入れ子がある場合は，最も根要素に近い要素のみを残し，他の要素を除去できる．この除去を行っても利得の総量は変わらない．よって検索結果には入れ子する要素は含まれない．

図 1 にある問合せに対してあるシステムが算出した利得と閲覧コストの例を示す．図 1

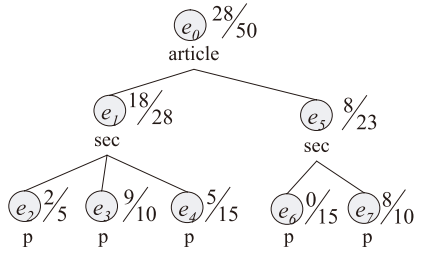

図 1　算出された利得と閲覧コストの例

Fig. 1　An example of calculated benefit and reading effort.

では XML 文書が木構造で表現され，それぞれの要素の利得と閲覧コストが「利得/閲覧コスト」の形式で要素の隣に示されている．単純化のため，ここでは具体的な利得と閲覧コストの算出式は想定していないが，これらの値は仮定 1 と仮定 2 を満たしている．

我々は，要素を読むことによって利得を取得するためにはその要素すべてを読まなければならないと仮定している．つまり，要素を途中まで読んでもその読んだ割合に応じた利得を取得することはできない．図 1 のように利得と閲覧コストが算出されたとき，利用者が閲覧コストの閾値として 15 を指定した場合，利得を最大化する要素集合は $\{e_3, e_2\}$ であり，20 を指定した場合は $\{e_3, e_7\}$ となる．

検索結果取得のアルゴリズムは我々が b/e グラフ（benefit/effort グラフ）と呼ぶグラフ上で表現することができる．検索結果取得アルゴリズムの挙動は，閲覧コストの閾値 c を変化させたときの利得の総量を描くことで表現する．たとえば，$0 \leq c < 10$ のとき ϕ を，$10 \leq c < 20$ のとき $\{e_3\}$ を，$20 \leq c < 38$ のとき $\{e_3, e_7\}$ を，$38 \leq c < 50$ のとき $\{e_1, e_7\}$ を，$50 \leq c$ のとき $\{e_0\}$ を取得するようなアルゴリズム A1 と，$0 \leq c < 5$ のとき ϕ を，$5 \leq c < 15$ のとき $\{e_2\}$ を，$15 \leq c < 25$ のとき $\{e_2, e_3\}$ を，$25 \leq c < 38$ のとき $\{e_2, e_3, e_7\}$ を，$38 \leq c < 50$ のとき $\{e_1, e_7\}$ を，$50 \leq c$ のとき $\{e_0\}$ を取得するようなアルゴリズム A2 は，b/e グラフ上で**図 2** のように表現できる．

3.3　問題の定式化

閲覧コストの閾値が与えられたときに利得を最大化する問題は入れ子による制約が追加されたナップサック問題[6] の変形であると見なすことができる．この問題 (P) は以下のように定式化される．

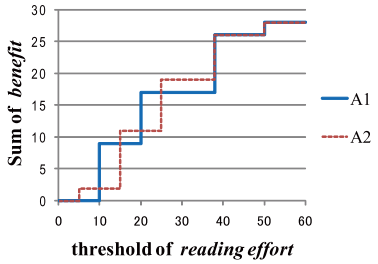

図 2　b/e グラフ上での A1 と A2
Fig. 2　A1 and A2 on b/e graph.

$$
\text{(P)}\quad
\left|\begin{array}{ll}
\text{maximize} & z(x)=\sum_{i=1}^{n} b_i x_i \qquad (1)\\
\text{subject to} & \sum_{i=1}^{n} r_i x_i \le c \qquad (2)\\
& x_i \in \{0,1\} \qquad (3)\\
& x_{j_1}+x_{j_2}+\ldots+x_{j_m} \le 1 \qquad (4)\\
& \quad \text{for any } e_{j_1}, e_{j_2}, \ldots, e_{j_m}\\
& \quad \text{which are elements on}\\
& \quad \text{a path from root to leaf}\\
& b_k \ge \sum_{e_i \in e_k.children} b_i \qquad (5)\\
& r_k \le \sum_{e_i \in e_k.children} r_i \qquad (6)
\end{array}\right.
$$

ここで，要素 e_i に対する利得を b_i とし，閲覧コストを r_i とする．また，利用者が指定した閲覧コストの閾値を c とする．e_i を結果に含める場合は x_i を 1 とし，結果に含めない場合は 0 とすることで表現できる．条件 (4) は内容が重複する（入れ子関係にある）要素は結果に含まれないことを示している．また，条件 (5) と条件 (6) はそれぞれ仮定 1 と仮定 2 から導き出されたものである．

この問題 (P) は，もし検索対象の XML 文書がそれぞれ 1 つの要素のみしか含まない場合は通常のナップサック問題になるため，ナップサック問題を拡張した問題であると考えられる．ナップサック問題は NP 困難であるため，この問題 (P) も NP 困難である．

さらに利用者は閲覧コストの閾値 c を変化させて検索結果を柔軟に取得することを想定しているため，c を変化させたときの検索結果どうしの関係にも制約を導入する．図 1 の状況において，問題 (P) を解き，利得を最大化するシステムは c が 15 のとき $\{e_3, e_2\}$ を取得し，20 のとき $\{e_3, e_7\}$ を取得する．しかしこの場合，c が 15 のときには利用者に提示される e_2 の内容を 20 のときには提示しないことになる．我々はこのような状況は避けるべきであると考えた．c を増やした際に提示する内容に包含関係がない場合は，利用者の検索結果の把握が困難になると思われる．検索結果取得アルゴリズムは以下で定義される検索結果の単調性を満たすことが重要であると考えた．

定義 1　検索結果の単調性

閲覧コストの閾値が c であるときの検索結果を $E^c = \{e_1^c, e_2^c, \ldots, e_n^c\}$ とし，c' のときの検索結果を $E^{c'} = \{e_1^{c'}, e_2^{c'}, \ldots, e_m^{c'}\}$ とすると，任意の c，c' で以下が成り立つとき，そのアルゴリズムは検索結果の単調性を満たす．ただし，*ancestor-or-self* (e) は e の先祖と e 自身からなる要素集合を返すものとする．

$$\textit{If } c \le c', \textit{ then } \forall\, e \in E^c, \exists\, e' \in E^{c'} \textit{ s.t. } e' \in \textit{ancestor-or-self}\,(e) \qquad \square$$

閲覧コストの閾値を c から c' に増やす際には，c に対して得られる検索結果の内容は c' に対する検索結果の内容に含まれていなければならない．ここで，条件 (3) と条件 (4) より問題 (P) を考えると，要素 e' は要素 e に対して一意に決まる．我々は実用的なシステムは検索結果の単調性を満たさなければならないと考えた．3.2 節のアルゴリズム A1，A2 は検索結果の単調性を満たす．

3.4　利得の上界

検索結果の単調性を満たす検索結果取得アルゴリズムは様々なものが考えられるが，一般的に，任意の閲覧コストの閾値 c に対してつねにそのほかのアルゴリズムを上回る利得を取得できるようなアルゴリズムは存在しない．図 1 の状況では，c が 15 のときに利得が最大になる $\{e_3, e_2\}$ を取得するアルゴリズムは，検索結果の単調性を考えると c が 20 のときに利得が最大になる $\{e_3, e_7\}$ を取得することができない．

もし，任意の c で最大の利得を得ることができるようなアルゴリズムがあれば，その利得をシステム評価の比較対象に用いることができるが，この問題ではそのようなアルゴリズムは存在しない．よって，取得可能な利得の上界値を求め，それをシステム評価の基準に用いることにした．問題 (P) の最適解は検索結果の単調性を満たさないが，上界値となる．

しかし問題 (P) は NP 困難であるため問題を解く計算コストが高い．よってさらに問題 (P) の上界値を算出し用いることにした．

定理 1　問題 (P) の上界値

問題 (P) の条件 (3) を緩和し $0 \leq x_i \leq 1$ とした (P) の連続緩和問題 (P') の最適値は (P) の上界を与える．

証明　(P') は (P) における条件を緩和しているため，(P) がとりうる値は (P') においても必ず得ることができ，(P) がとりうる値は (P') がとりうる値に含まれる．よって (P') の最適値は (P) の上界を与える．　□

(P') に対する最適解を算出するために，我々は**図 3** のアルゴリズムを考えた．アルゴリズムの入力は $list_{in}$ と c である．$list_{in}$ は，すべての i に関して $b_i/r_i \geq b_{i+1}/r_{i+1}$ を満たし，もし b_i/r_i と b_{i+1}/r_{i+1} が等しい場合は $r_i \leq r_{i+1}$ となるような要素のリストであり，c は閲覧コストの閾値である．アルゴリズムの出力は利得の最適値 z' と z' を与える $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ である．

(P') の最適解を算出する例として，図 1 のように利得と閲覧コストが算出され，閲覧コストの閾値 c として 40 が設定された場合を考える．この場合，入力される $list_{in}$ は $\{e_3(b_3/r_3 = 9/10 = 0.9), e_7(0.8), e_1(0.64), e_0(0.56), e_2(0.4), e_5(0.35), e_4(0.33)\}$ となる．まず，e_3 が処理され，$\boldsymbol{x} = \{e_0, e_1, e_2, \ldots, e_7\}$ は $\{0,0,0,1,0,0,0,0\}$ となり z' は 9 となる．同時に，先祖要素である e_1 と e_0 の利得と閲覧コストが調整される．e_1 に関しては b_1 が 9 になり，r_1 が 18 になる．e_0 に関しては b_0 が 19 になり，r_0 が 40 になる．これらの調整後の利得の値はこれらの先祖要素を後で処理する場合に獲得できる量であり，調整後の閲覧コストの値は後で処理する場合に必要とされる量である．これらの調整を受け，$list_{in}$ 中の要素は再順序付けされる．この場合，$list_{in}$ は $\{e_7(0.8), e_1(0.5), e_0(0.48), e_2(0.4), e_5(0.35), e_4(0.33)\}$ となる．さらに c が $40 - 10 = 30$ となる．次に e_7 が処理され，$\boldsymbol{x}$ は $\{0,0,0,1,0,0,0,1\}$ となり，z' は $9 + 8 = 17$ となり，$list_{in}$ は $\{e_1(9/18 = 0.5), e_2(0.4), e_0(0.37), e_4(0.33), e_5(0)\}$ となり，c は $30 - 10 = 20$ となる．次に e_1 が処理され，e_1 の子孫である e_3 に関し，$x_3 = 0$ となる．さらに e_2 と e_4 が $list_{in}$ から取り除かれる．$\boldsymbol{x}$ は $\{0,1,0,0,0,0,0,1\}$ となり，z' は $17 + 9 = 26$ となり，$list_{in}$ は $\{e_0(2/12 \simeq 0.17), e_5(0)\}$ となり，c は $20 - 18 = 2$ となる．最後に e_0 が処理されるが，$x_0 = 1$ とすることはできず，x_0 を $2/12 \simeq 0.17$ とし，x_1 と x_7 が $1 - 2/12 \simeq 0.83$ となる．$\boldsymbol{x}$ は $\{0.17, 0.83, 0, 0, 0, 0, 0, 0.83\}$ となり，z' は $26 + 2 * 0.17 \simeq 26.3$ となり，ここで処理が終了となってこの時点の値が最終的な結果となる．

```
Input: list_in, c
Output: z', x
 1: z' = 0
 2: while ((e_i = top(list_in))! = null) do
 3:    remove e_i from list_in
 4:    if (r_i > c) then
 5:       x_i = c/r_i
 6:       for (e_d ∈ e_i.descendants) do
 7:          if (x_d == 1) then
 8:             x_d = 1 − x_i
 9:          end if
10:       end for
11:       z' + = b_i * x_i
12:       break
13:    end if
14:    x_i = 1
15:    z' + = b_i
16:    for (e_d ∈ e_i.descendants) do
17:       x_d = 0
18:       remove e_d from list_in
19:    end for
20:    for (e_a ∈ e_i.ancestors) do
21:       b_a − = b_i
22:       r_a − = r_i
23:       rerank e_a in list_in
24:    end for
25:    c− = r_i
26: end while
27: return z', x
```

図 3　(P') の最適解算出アルゴリズム
Fig. 3　Optimal solution for (P').

定理 2　(P') の最適解

図 3 のアルゴリズムは (P') の最適解を与える．

証明　要素 e_r の子孫要素集合の中から入れ子しない要素集合 $\{e_{k_1}, e_{k_2}, \ldots, e_{k_m}\}$ を取り出したとき，$x_r = \alpha$，$x_{k_i} = 1 - \alpha$（$1 \leq i \leq m$）とすると (P') の条件は満足される．この場合，利得の総量は以下のとおりである．

$$b_r * \alpha + \sum_{i=1}^{m} (b_{k_i} * (1 - \alpha))$$

$$= b_r * \alpha + \sum_{i=1}^{m} b_{k_i} - \sum_{i=1}^{m} (b_{k_i} * \alpha)$$
$$= \left(b_r - \sum_{i=1}^{m} b_{k_i} \right) * \alpha + \sum_{i=1}^{m} b_{k_i} \quad (7)$$

同様に閲覧コストの総量も式 (7) 中で b_i を r_i で置き換えることで算出することができる．つまり，$x_r = \alpha$，$x_{k_i} = 1 - \alpha$（$1 \leq i \leq m$）とすることは，利得が $b_r - \sum_{i=1}^{m} b_{k_i}$ であり，閲覧コストが $r_r - \sum_{i=1}^{m} r_{k_i}$ であるような仮想的な要素 e_v を想定し，$x_v = \alpha$，$x_{k_i} = 1$（$1 \leq i \leq m$）とすることに等しい．

よって，処理した要素の先祖要素の利得と閲覧コストを調整し（20–24 行目），その時点で利得が増える割合の一番大きい要素 e_i を取得していくことで，最も効率良く利得を獲得していくことができる．e_i をすべて取得することができない場合は，取得できる割合に応じた量を取得することで最適値となる（4–13 行目）．e_i をすべて取得することができる場合，つまり $x_i = 1$ とすることが可能である場合は，その子孫要素 e_d に関して $x_d = \alpha > 0$ とすると，獲得する利得は $b_i * (1 - \alpha) + b_d * \alpha$ となり，条件 (5) から $b_i \geq b_d$ であるため，$b_i * (1 - \alpha) + b_d * \alpha \leq b_i$ となるため，$x_d = \alpha > 0$ とする意義はない（16–19 行目）．e_i を処理する前にその子孫要素を処理している場合は，すでに処理している子孫要素に関し，$x_{k_i} = 1$ となっているのに加え，仮想要素を取得して $x_v = 1$ とすることになるため，$b_r = b_v + \sum_{i=1}^{m} b_{k_i}$ の利得を $r_r = r_v + \sum_{i=1}^{m} r_{k_i}$ の閲覧コストをかけて取得することになり，実際には $x_r = 1$，$x_{k_i} = 0$ となる（17 行目）．□

定理 1 と定理 2 より，(P) の上界値は図 3 のアルゴリズムによって得ることができる．

3.5　単純貪欲取得アルゴリズム

検索結果の単調性を満たす検索結果取得アルゴリズムの 1 つとして，図 3 のアルゴリズムと同様に処理を進めるが，要素すべてを取得できない場合にその要素を取得せず，そこで終了するものが考えられる．図 3 のアルゴリズムの 5 から 11 行目を取り除いたものがそれにあたる．図 3 のアルゴリズムは (P') に対するアルゴリズムであるが，5 から 11 行目を取り除いたこのアルゴリズムは (P) の条件 (3) を満足する．このアルゴリズムを単純貪欲取得アルゴリズムと呼ぶことにする．

上界値を求めた際と同様の状況において，単純貪欲取得アルゴリズムは基本的には上界値を求める際と同様に処理を進めるが，最後の e_0 に関する処理をスキップする．e_1 の処理終了後が最終状態となり，結果は，$\boldsymbol{x} = \{0, 1, 0, 0, 0, 0, 0, 1\}$，$z = 26$ となる．

3.6　再帰貪欲取得アルゴリズム

単純貪欲取得アルゴリズムによって取得してきた要素の閲覧コストの合計値が利用者が指定した閲覧コストの閾値 c よりも小さくなる場合は，その余った閲覧コストに対してまだ取得していない要素を取得してくることにより，獲得できる利得の量を増やすことができると考えられる．

ただし，検索結果の取得に際し，定義 1 の検索結果の単調性を満たすことが重要であるため，余った閲覧コストに対して任意の要素を取得してくることは不適切である．検索結果の単調性を満たしつつ，要素を取得するためには，取得する要素が次に取得予定であった要素の子孫要素である必要がある．余った閲覧コストに対してさらに要素を取得する改善アルゴリズムは**図 4** のアルゴリズムのようになる．このアルゴリズムを再帰貪欲取得アルゴリズムと呼ぶことにする．

利用中の例では，閲覧コストの閾値 c として 30 が指定されたときに，単純貪欲取得アルゴリズムでは $\boldsymbol{x} = \{0, 0, 0, 1, 0, 0, 0, 1\}$，$z = 17$ となるが，再帰貪欲取得アルゴリズムでは，e_7 を取得した後に，次に取得予定の e_1 の子孫のうち，最も利得獲得効率の良い e_2 を取得して，結果は $\boldsymbol{x} = \{0, 0, 1, 1, 0, 0, 0, 1\}$，$z = 19$ となる．

定理 3　再帰貪欲取得アルゴリズムの優位性

任意の閲覧コストの閾値に対して，再帰貪欲取得アルゴリズムによって取得する要素集合から得られる利得の合計値は，単純貪欲取得アルゴリズムによって取得する要素集合から得られる利得の合計値よりも大きいか等しくなる．

定理 3 は明らかであるため，証明は省略する．単純貪欲取得アルゴリズムに比べてつねに悪くない結果が得られるため，システムの実装は再帰貪欲取得アルゴリズムを用いて行う．

4. 評 価 手 法

我々は b/e グラフ上でのシステムの挙動に基づいた XML 情報検索システムの評価を提案する．ある問合せに対してシステムが図 1 のように利得と閲覧コストを算出した場合，単純貪欲取得アルゴリズムの挙動は**図 5** 中の “simple” で，再帰貪欲取得アルゴリズム挙動は “recursive” で示すことができる．

b/e グラフで考えると，3.4 節で考えた取得可能な利得の上界は単純貪欲取得アルゴリズムで要素を処理した時点の点を線形補間することで表現できる（図 5 中の “upper bound”）．単純貪欲取得アルゴリズムは，ちょうど要素を処理した時点では利得を最大化する最適な結果を取得していることが分かる．

```
Input: list_in, c
Output: z, x
 1: z = 0
 2: while ((e_i = top(list_in))! = null) do
 3:    if (!retrieve(e_i)) then
 4:       break
 5:    end if
 6: end while
 7: return z, x
 8:
 9: function retrieve(e_i) {
10:    remove e_i from list_in
11:    if (r_i > c) then
12:       retrieveDescendants(e_i)
13:       return false
14:    end if
15:    x_i = 1
16:    z+ = b_i
17:    for (e_d ∈ e_i.descendants) do
18:       x_d = 0
19:       remove e_d from list_in
20:    end for
21:    for (e_a ∈ e_i.ancestors) do
22:       b_a− = b_i
23:       r_a− = r_i
24:       rerank e_a in list_in
25:    end for
26:    c− = r_i
27:    return true
28: }
29:
30: function retrieveDescendants(e_i) {
31:    e_m = top ranked element that is
      descendant of e_i and x_m = 0
32:    if (e_m == null) then
33:    return
34:    end if
35:    if (retrieve(e_m)) then
36:       retrieveDescendants(e_i)
37:    end if
38: }
```

図 4　再帰貪欲取得アルゴリズム
Fig. 4　Recursive greedy retrieval algorithm.

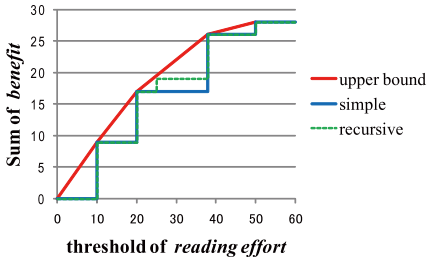

図 5　b/e グラフの例
Fig. 5　An example of b/e graph.

システムの評価のために，問合せに対する要素ごとの実際の利得が利用可能であるとする．しかし，実際の利得はシステム評価者しか知りようがなく，システムを実装する際には利用できない情報である．XML 情報検索システムの実装者は，実際の利得になるだけ近い値をシステムに推測させることで，より良いシステムを目指していくことになる．我々は，閲覧コストに関しては問合せに依存しない値であるため，実際の値はシステム実装者にも既知であると考えた．

システムの評価は，b/e グラフ上で閲覧コストの閾値を変化させたときのシステムが獲得した実際の利得を表現し，それを獲得可能な利得の上界値と比較することで行うことができると考えた．上界値はそれぞれの要素から獲得できる実際の利得を用いて図 3 のアルゴリズムを適用することで算出することができる．もし，実装したシステムがそれぞれの要素に対して実際の利得とまったく同じ値を算出することができた場合，再帰貪欲取得アルゴリズムを用いて得られた結果は上界値と非常に近い値を獲得できると考えられる．我々はそのようなシステムを準理想システムと呼ぶ．

例として，ある問合せに対してシステムが図 1 のように利得と閲覧コストを算出したが，実際の利得は**図 6** のようであった場合を考える．この場合，再帰貪欲取得アルゴリズムを用い，閲覧コストの閾値が 40 であったとき，システムは $\boldsymbol{x} = \{0, 0, 1, 1, 0, 0, 0, 1\}$ とすることで利得の合計値 20 を取得するが，上界値は $\boldsymbol{x} = \{0, 0, 0, 1, 1, 0, 1, 0\}$ とした際の 33 である．**図 7** に，この場合の b/e グラフを示す．図 7 では，上界値を “upper bound” で，システムの挙動を “system” で示している．また，実際の利得と等しい利得を算出できた準理想システムの挙動を “quasi ideal” で示している．b/e グラフを用いることで上界値と比較したシステムの性能を直観的に把握することができる．上界値と比較することでシステムの

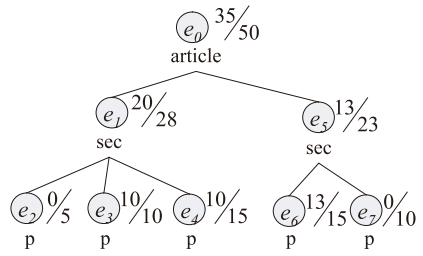

図 6　実際の利得と閲覧コスト
Fig. 6　Actual benefit and reading effort.

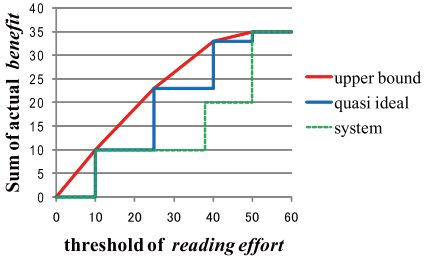

図 7　b/e グラフを用いた評価
Fig. 7　b/e graph for evaluation.

絶対評価が可能となる．

INEX 2005 では XCG（eXtended Cumulated Gain）に基づく評価手法が用いられており[11]，XCG に基づいた評価尺度では理想的なシステムに対して自分たちのシステムが相対的にどれくらい優れているかを計測する．我々の場合は上界値を比較対象に用いることで XCG の考え方を適用することができる．たとえば，iMAep（interpolated Mean Average effort precision）[11] と同様に考え，iMArep（interpolated Mean Average reading effort precision）を導入することができる．

iMArep は個々の問合せに対して算出される iArep（interpolated Average reading effort precision）の平均値である．利得の総量の 0%，1%，2%，...，100%の 101 点での，その利得を得るのにシステムが必要とする閲覧コストに対し，その利得を上界値として取得するのに必要な閲覧コストの割合を求め，その割合の平均を求めたものが iArep である．上界値の取得は単純貪欲取得アルゴリズムで要素を処理した時点の点を線形補間することで得ることができる．

5. 実　　験

我々は INEX プロジェクト[13] の提供する XML 情報検索システムのためのテストコレクションを利用して実験を行った．このテストコレクションは XML 文書集合，問合せ集合（Topics），問合せに対する正解データ（Assessments）からなる．INEX 2005 の正解データから実際の利得と閲覧コストを算出する手法を考え，利得と閲覧コストを用いる簡単なシステムの実装を行い，INEX 2005 の問合せに対し b/e グラフを取得した．また準理想システムの上界値に対する性能を見ることで上界値の精度を確認した．

5.1　INEX 2005 の正解データ

INEX 2005 では問合せに対する正解データは Exhaustivity（ex）と Specificity（sp）の二次元の情報からなる[⋆1]．Exhaustivity はどの程度問合せに対して議論しているかの指標であり，Highly exhaustive（HE），Partially exhaustive（PE），Not exhaustive（NE）の 3 段階を考えている[⋆2]．我々は HE を 1，PE を 0.5，NE を 0 と数値化した．Specificity は要素の内容全体に対しどの程度問合せに関連する内容になっているかの指標であり，問合せに関連する内容の長さ（$rsize$）を要素の内容全体の長さ（$size$）で割ることにより求めている．

要素 e_i の ex，sp，$rsize$，$size$ を $ex(e_i)$，$sp(e_i)$，$rsize(e_i)$，$size(e_i)$ とすると，ex，sp，$rsize$，$size$ の性質から以下の式が成り立つ．ここで，$e_i.parent$ を e_i の親要素とし，$e_i.children$ を e_i の子要素集合とする．

$$sp(e_i) = rsize(e_i)/size(e_i) \tag{8}$$

$$ex(e_i) \leq ex(e_i.parent) \tag{9}$$

$$rsize(e_i) = \sum_{e_j \in e_i.children} rsize(e_j) \tag{10}$$

$$size(e_i) = \sum_{e_j \in e_i.children} size(e_j) \tag{11}$$

5.2　実際の利得と閲覧コストの算出

INEX 2005 の正解データから要素の実際の利得と閲覧コストを算出することを考え，以

⋆1 なお，INEX 2006 の正解データでは Specificity のみを用いている．
⋆2 要素が非常に小さい場合に対して Too Small（TS）を導入しているが，TS は NE と等価であると考える．

下の式を用いた．

$$b_i = ex(e_i)^{\alpha} * rsize(e_i)^{\beta} \quad (\alpha \geq 0, \beta \geq 1) \tag{12}$$

$$r_i = size(e_i)^{\gamma} \quad (0 \leq \gamma \leq 1) \tag{13}$$

式 (9) と式 (10) より，式 (12) は仮定 1 を満足する．また，式 (11) より，式 (13) は仮定 2 を満足する．INEX 2006 の正解データでは Specificity のみを用いているが，上記の式では $\alpha = 0$ とすることで容易に対応できる．

5.3　システムの実装

閲覧コストに関しては，問合せに依存しない値であり，実際の値がシステム実装時にも利用可能であると考えているため，システムの実装の焦点は利得の算出にある．それぞれの要素に対して仮定 1 を満足するような利得を推測し，算出していくことになる．本実験では 2 つのシステムを実装した．

5.3.1　tf-ief を利用したシステム

1 つめのシステムでは，$tf\text{-}ief$ を基にした利得の算出式を用いることにした[*1]．$tf\text{-}ief$ 値が大きい，つまり特定性の高い語が高頻度で現れる要素ほど検索語に関する情報が得られると思われるため，利得は大きくなると思われる．また，複数の検索語を用いる場合，より多くの種類の語が出現する要素ほど利得が大きくなると考えた．

$$b_i = \frac{n}{|q|} * \sum_{t \in q} (tf * ief) \tag{14}$$

$$ief = ln\frac{N+1}{ef} \tag{15}$$

ここで，tf は索引語出現回数であり，ief は索引語の特定性である．n は問合せ中の語のうち，e_i に出現するものの数である．q は問合せであり，$|q|$ は問合せ q 中の検索語数である．ief の算出には全要素数 N と索引語が出現する要素の数 ef を用いる．

XML 文書における索引語の重みの算出には様々な提案があるが[14)–17)]，ここでは仮定 1 を満足することが重要であると考え，仮定 1 を満たすような簡単な式を用いた．

5.3.2　関連する文の長さを利用したシステム

もう 1 つのシステムとしては，INEX の正解データが問合せに関連するテキスト長 ($rsize$) を利用していることを考え，利得を要素中の関連する文の長さを基に算出することを考えた．文を基本単位と考え，要素中で関連するテキスト長を関連する文の長さの和を用いて推測する．

$$b_i = \frac{n}{|q|} * \frac{\sum_{s_j \in rel(e_i)} (sl_j * \frac{n_{s_j}}{|q|})}{el} \tag{16}$$

ここで，n，q，$|q|$ は $tf\text{-}ief$ の場合と同じであり，el は要素の長さである．sl_j は文 s_j の長さであり，n_{s_j} は問合せ中の語のうち，s_j に出現するものの数である．$rel(e_i)$ は要素 e_i 中の関連する文の集合を返す．本実験では，問合せ中の語のうち少なくとも 1 つの語を含むような文は問合せに関連する文であると見なした．

5.4　システムの評価

我々は 5.3 節の 2 つのシステムに関し，実際の利得と閲覧コストは 5.2 節の手法によって得られると考え，b/e グラフを取得した．例として INEX 2005 の問合せのうち，Topic 206 と Topic 207 のものを**図 8** と**図 9** に示す．本実験では $\alpha = 0.5$，$\beta = 1$，$\gamma = 1$ とした．これらの図では，“upper bound” は獲得可能な利得の上界値を示し，“quasi ideal” は準理想システムの挙動を示し，“tf-ief” は 5.3.1 項の $tf\text{-}ief$ を利用したシステムの挙動を示し，“sentence” は 5.3.2 項の関連する文の長さを利用したシステムの挙動を示している．

システムの性能は b/e グラフを用いることで直観的に把握できる．Topic 206 ではどちらのシステムもあまり良い結果を取得できなかったが，Topic 207 では比較的良い結果を取得できていることが観測できる．さらに Topic 207 では関連する文の長さを利用したシステムの方が $tf\text{-}ief$ を利用したシステムよりも全般的に多くの利得を獲得できていることが分

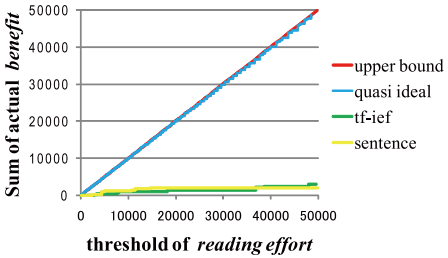

図 8　Topic 206 の b/e グラフ
Fig. 8　b/e graph of Topic 206.

[*1] ief は inverse element frequency の略である．

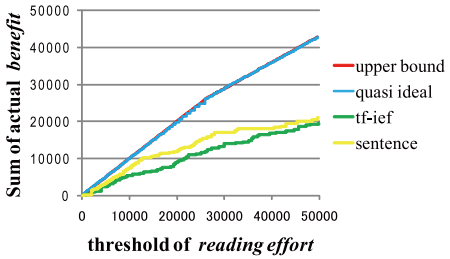

図 9　Topic 207 の b/e グラフ
Fig. 9　b/e graph of Topic 207.

かる．

また，準理想システムの挙動が上界値とほぼ同じであることが見てとれるため，我々が提案した上界値は実用に耐えうる精度であると考えられる．**表 1** に INEX 2005 の 29 個の問合せに対する準理想システムの iArep 値を示す．ほとんどの値は 0.9 以上であり上界値の性能は良いことが分かる．Topic 209 のようないくつかの問合せでは比較的値が低いが，これはそのような問合せに関しては良い結果が比較的大きな要素であるためである．29 個の問合せの iArep 値を平均した iMArep 値は 0.90 であり，Topics 209，217，239 を取り除いて考えた iMArep 値は 0.93 であった．

6. おわりに

我々は利得と閲覧コストの概念を導入し，それらに基づく XML 情報検索システムの検索結果の取得と評価を提案した．利用者が閲覧コストの閾値を入力し，システムはその閲覧コスト以内で読むことが可能なより多くの利得を持つ要素を取得する．我々は与えられた閲覧コストに対して利得を最大化する問題を定式化し，上界値を算出してシステムの評価に利用した．

今後の課題としては，要素中で利用者に実際に読んでほしい部分とそうでない部分を区別して表示することで，より適切な部分のみを利用者に提示することが考えられる．また，提示する結果の数が多くなると，利用者の閲覧時の煩雑さが増加すると思われる．この問題に対処するため，閲覧時の切替えコストの概念を導入することを検討している．さらに，PDF などから作成された XML 文書の場合は，検索結果を PDF のレイアウト上に重畳して提示することも考えられ[18]，そのような提示手法との統合も課題である．

表 1　準理想システムの iArep
Table 1　iArep for quasi ideal system.

Topic ID	iArep
202	0.88
203	0.93
205	0.93
206	0.98
207	0.99
208	0.82
209	0.71
210	0.95
212	0.96
213	0.94
216	0.91
217	0.73
218	0.91
219	0.82
221	0.97
222	0.99
223	0.97
227	0.92
228	0.91
229	0.94
230	0.88
232	0.84
233	0.92
234	0.98
235	0.97
236	0.86
237	0.97
239	0.72
241	0.91

参考文献

1) Ley, M.: DBLP Computer Science Bibliography. http://www.informatik.uni-trier.de/~ley/db/
2) Xu, Y. and Papakonstantinou, Y.: Efficient keyword search for smallest LCAs in XML databases, *Proc. 2005 ACM SIGMOD International Conference on Management of Data*, Baltimore, Maryland, USA, pp.537–538 (2005).
3) Liu, Z. and Chen, Y.: Identifying meaningful return information for XML key-

word search, *Proc. 2007 ACM SIGMOD International Conference on Management of Data*, Beijing, China, pp.329–340 (2007).
4) Theobald, M., Schenkel, R. and Weikum, G.: An efficient and versatile query engine for TopX search, *Proc. 31st International Conference on Very Large Data Bases*, Trondheim, Norway, pp.625–636 (2005).
5) Kaushik, R., Krishnamurthy, R., Naughton, J.F. and Ramakrishnan, R.: On the integration of structure indexes and inverted lists, *Proc. 2004 ACM SIGMOD International Conference on Management of Data*, Paris, France, pp.779–790 (2004).
6) Martello, S. and Toth, P.: *Knapsack problems: algorithms and computer implementations*, John Wiley & Sons Inc, New York (1990).
7) Malik, S., Kazai, G., Lalmas, M. and Fuhr, N.: Overview of INEX 2005, *Proc. 4th International Workshop of the Initiative for the Evaluation of XML Retrieval*, Lecture Notes in Computer Science, Vol.3977, Dagstuhl Castle, Germany, pp.1–15, Springer-Verlag (2005).
8) Pehcevski, J. and Thom, J.A.: HiXEval: Highlighting XML retrieval evaluation, *Proc. 4th International Workshop of the Initiative for the Evaluation of XML Retrieval*, Lecture Notes in Computer Science, Vol.3977, Dagstuhl Castle, Germany, pp.43–57, Springer-Verlag (2005).
9) Clarke, C.L.A.: Controlling overlap in content-oriented XML retrieval, *Proc. 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Salvador, Brazil, pp.314–321 (2005).
10) Malik, S., Trotman, A., Lalmas, M. and Fuhr, N.: Overview of INEX 2006, *Proc. 5th International Workshop of the Initiative for the Evaluation of XML Retrieval*, Lecture Notes in Computer Science, Vol.4518, Dagstuhl Castle, Germany, pp.1–11, Springer-Verlag (2006).
11) Kazai, G. and Lalmas, M.: INEX 2005 evaluation measures, *Proc. 4th International Workshop of the Initiative for the Evaluation of XML Retrieval*, Lecture Notes in Computer Science, Vol.3977, Dagstuhl Castle, Germany, pp.16–29, Springer-Verlag (2005).
12) Pehcevski, J., Kamps, J., Kazai, G., Lalmas, M., Ogilvie, P., Piwowarski, B. and Robertson, S.: INEX 2007 evaluation measures, *INEX 2007 Pre-Proceedings* (2007).
13) INEX: INitiative for the Evaluation of XML Retrieval. http://inex.is.informatik.uni-duisburg.de/
14) Cohen, S., Mamou, J., Kanza, Y. and Sagiv, Y.: XSEarch: A semantic search engine for XML, *Proc. 29th International Conference on Very Large Data Bases*, Berlin, Germany, pp.45–56 (2003).
15) Grabs, T. and Schek, H.-J.: ETH Zürich at INEX: Flexible information retrieval from XML with PowerDB-XML, *Proc. 1st Workshop of the INitiative for the Evaluation of XML Retrieval*, Schloss Dagstuhl, Germany, pp.141–148 (2002).
16) Amer-Yahia, S., Curtmola, E. and Deutsch, A.: Flexible and efficient XML search with complex full-text predicates, *Proc. 2006 ACM SIGMOD International Conference on Management of Data*, Chicago, USA, pp.575–586 (2006).
17) Shimizu, T., Terada, N. and Yoshikawa, M.: Kikori-KS: An effective and efficient keyword search system for digital libraries in XML, *Proc. 9th International Conference on Asian Digital Libraries*, Lecture Notes in Computer Science, Vol.4312, Kyoto, Japan, pp.390–399, Springer-Verlag (2006).
18) Shimizu, T. and Yoshikawa, M.: XML information retrieval considering physical page layout of logical elements, *Proc. 10th International Workshop on the Web and Databases*, Beijing, China (2007).

(平成 19 年 12 月 20 日受付)
(平成 20 年 4 月 7 日採録)

(担当編集委員　太田 学)

清水　敏之（正会員）

2003 年名古屋大学工学部電気電子・情報工学科卒業．2005 年同大学大学院情報科学研究科博士前期課程修了．2008 年京都大学情報学研究科博士後期課程修了．博士（情報学）．現在，日本学術振興会特別研究員．XML データベース，特に XML 索引付け，XML 全文検索，XML 情報検索に興味を持つ．ACM，日本データベース学会各会員．

吉川　正俊（正会員）

1980 年京都大学工学部情報工学科卒業．1985 年同大学大学院工学研究科博士後期課程修了．工学博士．同年京都産業大学計算機科学研究所講師．同大学工学部助教授，奈良先端科学技術大学院大学情報科学研究科助教授，名古屋大学情報連携基盤センター教授，同大学大学院情報科学研究科教授を経て，2006 年より京都大学大学院情報学研究科教授，現在に至る．1989〜1990 年南カリフォルニア大学客員研究員，1996〜1997 年ウォータルー大学客員准教授．XML データベース，多次元空間索引等の研究に従事．電子情報通信学会，ACM，IEEE Computer Society 各会員．日本データベース学会理事．