データベースシステム　122－48
(2000. 7. 27)

構造化文書を対象とした文字列検索とベクトル検索の統合について

渡邉 正裕†　波多野 賢治‡　吉川 正俊‡　植村 俊亮‡　中村 均†

†国立特殊教育総合研究所 教育工学研究部
‡奈良先端科学技術大学院大学 情報科学研究科

†〒 239–0841 神奈川県横須賀市野比 5–1–1
‡〒 630–0101 奈良県生駒市高山町 8916–5

{masahiro, nakamr}@nise.go.jp, {hatano, yosikawa, uemura}@is.aist-nara.ac.jp

あらまし　情報検索システムへの関心が急速に増大している．その背景には，ネットワーク技術の発達に伴い誰もが氾濫する情報資源にアクセス可能になり，より的確に情報を絞り込みたいというな要求が大きくなったことがある．特にSGMLやXMLなどの構造化文書が増加するにしたがって，文書の構造や文書に付加された情報を手がかりにした検索に関する要求も生じている．本稿ではこのような情報検索システムへの要求に応じるべく，XPathを利用して文字列検索とベクトル検索を統合する手法を提案する．この手法によって文字列検索による厳密なパターンマッチと，ベクトル検索による意味的なマッチングの両方が合わせて利用可能になる．その際，XPathを拡張して利用する方法と，XPathを拡張せずにそのまま利用する方法を提案し，議論する．

キーワード　文字列検索，ベクトル検索，XPath，ランキング

Integration of Two Search Models by XPath: Keyword Match and Vector Match

Masahiro Watanabe†, Kenji Hatano‡, Masatoshi Yoshikawa‡,
Shunsuke Uemura‡, and Hitoshi Nakamura†

†Department of Educational Technology, National Institute of Special Education
‡Graduate School of Information Science, Nara Institute of Science and Technology

†5–1–1 Nobi, Yokosuka, Kanagawa 239–0841, Japan
‡8916–5 Takayama, Ikoma, Nara 630–0101, Japan

{masahiro, nakamr}@nise.go.jp, {hatano, yosikawa, uemura}@is.aist-nara.ac.jp

Abstract　The information retrieval systems attract a great deal of attention. Progress in network technology makes everyone come up against a flood of information. Their attention to information retrieval systems then produce higher request for retrieval performance. Especially the many researches targeted to structured document such as SGML or XML are running. In this paper, we integrate keyword match and vector match retrieval model using XPath in order to meet the demand for higher retrieval performance. By proposed method, we can use both of strict pattern match by "Keyword Match" and conceptual match by "Vector Match". Two methods are proposed and discussed in the paper. In one method we extend XPath, and in the other, we need not alter XPath.

Key words　exact match, vector match, XPath, ranking

1 はじめに

近年のWWWの普及に伴って，検索エンジンに対する要求も急速に高まっている．特に，WWWページを大量に収集，索引付けしてデータベース化して提供する形式のサービスが，網羅性，即時性の点で優れていることから増加している．このような形式のサービスでは一般に**文字列照合検索**か**ベクトル照合検索**のいずれかが用いられていることが多い．**文字列照合検索**では，文字列の一致によってパターンマッチを行うため，返ってきた結果には問合せで指定した文字列が確実に含まれていることが保証されるという長所がある．人名などの固有名詞は厳密な照合が要求される．この場合には，文字列照合を用いれば良い．しかし，表記自体が重要な意味をもたない場面では，内容が一致しているものは検索結果に含めた方が望ましい．たとえば，文字列検索だけでは'知的障害'について検索したいとき，意味的に同義である'精神薄弱'を含む文書を検索することができない．

ベクトル照合検索は，厳密に字面が一致しなくても，意味的に近ければ高い類似度 (similarity) を返す．したがって利用者による単語の表記のゆれに対しても比較的頑健 (robust) な性能を発揮できる．また，検索結果に付随して類似度が得られるので，これを結果の順位付けなどさまざまなことに利用可能である．単純な文字列照合検索ではしばしば検索結果が皆無になってしまったり，膨大になってしまったりするが，ベクトル照合検索では順位付けを行うことによって検索結果集合の大きさを利用者が自由に制御できる．また，現在は差別用語として避けられているが，以前は**知的障害**と同様の意味と用法で用いられていた**精神薄弱**というような語の存在を想定する場合，**精神薄弱**という語を含む文書も意味的に照合して検索結果に含めることができればたいへん有効である．こういった場合に文字列検索は不向きであり，ベクトル検索を用いた方が適していると考えられる．

本稿では文字列検索とベクトル検索をXPathを用いて統合する方法を提案し，二つの検索手法の長所を両方利用する．両者の橋渡しとしてXPathを用いることによって，XPathのもつ強力な構造記述能力の一部を問合せ機能に利用することが可能になる．これによってたとえば，章や節の兄弟関係や親子関係を指定して，文字列検索とベクトル検索を組み合せたような問合せを記述することが可能になる．

一般のサーチエンジンでは**近接検索**として**指定された近さで特定の単語が含まれている文書**を検索する機能を備えている．一方，構造を利用した文書検索の関連研究[5]の多くは，親子関係を区別しても，兄弟関係，つまり長男と末っ子を区別しない．これは，構造に着目した研究の多くが**文書構造の深さ**を重視しているためだと考える．本研究では構造上の深さだけでなく，表記文字列上の近接も問合せに対する適合度に影響を持つと考え，これを考慮に入れた問合せ評価手法を提案する．

以下，2章で，本研究で文字列検索とベクトル検索の統合に用いるXPathと，関連する研究を紹介する．

2 XPathおよび関連研究

XML Path Language (XPath)[1]は1999年にW3C勧告になった，XML文書の位置特定に用いる記法である．XPathは，XSLTおよびXPointerで使用するように設計されているが，そこではXMLのすべての要素，属性等は木構造のノードとして扱われる．

検索対象として文書の部分に着目することによって検索の精度を向上させようとする研究は，数多く試みられてきた．たとえば，TextTiling[2]の手法は，文書を細かく分割した部分をベクトル化することによって，文書内での文脈の変化をより正確に捉えようとするものである．

構造化文書では，一般に木構造になる文書の論理構造情報の存在がはじめから仮定でるが，これを文書検索やデータベースシステム構築に利用した研究[3]が行われている．BUS(Bottom Up Scheme)[4]は構造化文書の木構造の下位のノードのベクトル化を，そこに含まれる単語の頻度情報から行い，順次ボトムアップに上位のノードをベクトル化する手法を提案している．絹谷らの研究[5]では，文字列上の近接だけでなく，文書の構造を考慮に入れた近接サーチを提案している．

3 文字列照合検索とベクトル照合検索を統合するために

本研究では文字列照合検索とベクトル照合検索の統合を目標としている．本研究における**文字列照合検索**とは，問合せ中で指定された文字列を含む文書，またはそのような文書の部分構造を特定する検索であると定義する．たとえば，図1，図2のような構造を持つ文書を想定した場合に，次のような問合せが文字列照合検索である．

文字列'知的障害'を含む *article* エレメントを検索せよ (1)

また，本研究における**ベクトル照合検索**とは，問合せ中で指定された意味ベクトルと，類似しているような文書ベクトルに対応する文書，または文書の部分構造を特定する検索であると定義する．たとえば，図1，2のような構造を持つタグ付き文書を想定した場合に，次のようなベクトル照合検索の問合せが考えられる．

意味ベクトル知的障害に類似した文書ベクトルに
対応する *article* エレメントを検索せよ (2)

本研究では文字列照合検索とベクトル照合検索両方の統合利用が最終目標であるが，それはたとえば次のような問合せの実現を含む．

文字列'知的障害'を *title* エレメントに含み，
意味ベクトル学習障害に類似した文書ベクトルを
section エレメントに含むような文書を検索せよ (3)

<article>
text1
<para foo = "bar">
text2
</para>
</article>

図 1: タグ付き文書を図示した例

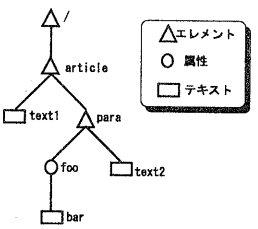

図 2: 図 1 の文書の木構造

本研究では XPath を利用して文字列検索とベクトル検索の統合を行う．しかし，XPath はベクトル型が定義されていない．したがって，まず，XPath でベクトルを扱えるようにベクトル型を導入する．

関数:*vector* **vector**(*object?*)

vector 関数によって文書木中のノードや単語などの任意のオブジェクトをベクトルに変換することができる．

4 文字列照合検索とベクトル照合検索の統合

本研究では，XPath を利用して文字列照合検索とベクトル照合検索の統合を行う．3章で XPath に *vector* 型を導入したが，この *vector* 型の引数をとるベクトル照合関数 **SIMILAR** と，**is_similar** を本節では導入し，文字列照合関数 **CONTAIN**，**contains** と統合利用する．

本研究において，**Query(問合せ)** は **Head(ヘッド)** と **Bool(ブール式)** の対で構成される．

[001] **Query** ::= '{' **Head** '|' **Bool** '}'

[001] (問合せ) ::= '{' (ヘッド) '|' (ブール式) '}'

ヘッドは問合せの左の部分に記述される検索対象のオブジェクトである．ヘッドはブール式に記述される条件によって限定される．

[002] **Bool** ::=

Ex
| ¬ **Bool**
| **Bool** ∧ **Bool**
| **Bool** ∨ **Bool**

[002] (ブール式) ::=

(式)
| ¬ (ブール式)
| (ブール式) ∧ (ブール式)
| (ブール式) ∨ (ブール式)

文字列照合検索とベクトル照合検索を XPath を用いて統合するために次に述べる 2 通りの基本方針を提案する．

- XPath を拡張せずに，XPath を外部から利用
- XPath ですでに定義されている **contains** をそのまま利用し，さらにベクトル照合を新たに定義するという XPath の拡張を行い，XPath を主体的に利用

[003] **Ex** ::= **NonExtXPathEx** | **ExtXPathEx**

[003] (式) ::= **(XPath 非拡張版式)** | **(XPath 拡張版式)**

[004] **NonExtXPathEx** ::= **CONTAIN_Ex** | **SIMILAR_Ex**

[004] **(XPath 非拡張版式)** ::= **(CONTAIN 式)** | **(SIMILAR 式)**

[005] **ExtXPathEx** ::= **Ex_contains** | **Ex_is_similar**

[005] **(XPath 拡張版式)** ::= **(contains 式)** | **(is_similar 式)**

4.1 XPath を拡張しない方法

文字列照合の関数 **CONTAIN** とベクトル照合の関数 **SIMILAR** を定義し，これらの外部から XPath を利用する．

4.1.1 XPath を拡張しない場合の照合関数

XPath を拡張しない場合の照合関数 **CONTAIN**, **SIMILAR** を定義する．

関数:*boolean* **CONTAIN**(*string, string*)

CONTAIN 関数は，1 番目の引数に指定した文字列が 2 番目の引数に指定した文字列を含んでいる場合に真を返し，それ以外は偽を返す．

例）文字列 '知的障害' を含む *article* を検索せよ

$\{a | a \in (//article) \wedge$ **CONTAIN** $(a,$'知的障害'$)\}$

関数:*long* **SIMILAR**(*vector, vector*)

SIMILAR 関数は，1番目の引数に指定したベクトルが2番目の引数に指定したベクトルと類似している度合いを0以上1以下の類似度 (similarity) として返す．

例）意味ベクトル‘知的障害’に類似している *article* を検索せよ

$\{a|a \in (//article) \wedge \textbf{SIMILAR}\ (a, vector('知的障害'))\}$

4.1.2 CONTAIN と SIMILAR の評価

CONTAIN 関数，**SIMILAR** 関数の評価を次により定義する．S_i を文字列，V_j をベクトルとして，

$$\textbf{CONTAIN}(S_1, S_2) ::= \begin{cases} 1, if\ S_1\ \textbf{contains}\ S_2 \\ 0, otherwise. \end{cases} \quad (4)$$

$$\textbf{SIMILAR}(V_1, V_2) ::= cosine(V_1, V_2) \quad (5)$$

したがって，$0 \leq \textbf{SIMILAR}(V_1, V_2) \leq 1$

4.2 XPath を拡張する方法

XPath のコア関数ライブラリで定義されている文字列照合の関数 **contains** と，新たに定義するベクトル照合の関数 **is_similar** を用いて，拡張された XPath を主体にして問合せを記述する．

4.2.1 XPath を拡張する場合の照合関数

XPath に拡張を行い，ベクトル照合関数 **is_similar** を導入する．既存の文字列照合関数 **contains** と新しく定義したベクトル照合関数 **is_similar** を合わせて利用する．

関数: *boolean* **contains**(*string, string*)

contains 関数は XPath のコア関数ライブラリで定義される関数で，1番目の引数に指定した文字列が2番目の引数に指定した文字列を含んでいる場合に真を返し，それ以外は偽を返す．

例）ドキュメントルートの *article* という名前の子孫エレメントで，文字列‘知的障害’を含むものを検索せよ

$\{a|a \in (//article[\textbf{contains}\ (.,'知的障害')])\}$

関数: *long* **is_similar**(*vector, vector*)

is_similar 関数は XPath のコア関数ライブラリに新たに追加する関数で，1番目の引数に指定したベクトルが2番目の引数に指定したベクトルと類似している度合いを0以上1以下の類似度 (similarity) として返す．

例）ドキュメントルートの *article* という名前の子孫エレメントで，‘知的障害’という意味ベクトルに類似したものを検索せよ

$\{a|a \in (//article[\textbf{is\_similar}\ (a, vector('知的障害'))])\}$

4.2.2 contains と is_similar の評価

contains 関数，**is_similar** 関数の評価を次により定義する．S_i を文字列，V_j をベクトルとして，

$$\textbf{contains}(S_1, S_2) ::= \begin{cases} 1, if\ S_1\ \textbf{contains}\ S_2 \\ 0, otherwise. \end{cases} \quad (6)$$

$$\textbf{is\_similar}(V_1, V_2) ::= cosine(V_1, V_2) \quad (7)$$

したがって，$0 \leq \textbf{is\_similar}(V_1, V_2) \leq 1$

4.3 ブール式の評価

本節ではブール式の評価を定義する．単純式の連言 (conjunction) と選言 (disjunction) は，次のように評価する．ただし，文書の構造を評価に反映しないという意味で，これを**基本的評価**と呼ぶ．
構文規則の‘Ex’(式) を q_i として，

$$q_1 \wedge q_2 \wedge \ldots \wedge q_i = q_1 \cdot q_2 \cdot \ldots \cdot q_i \quad (8)$$

$$q_1 \vee q_2 \vee \ldots \vee q_i = 1 - (1 - q_1)(1 - q_2)\ldots(1 - q_i) \quad (9)$$

式 (8),(9) より，構文規則の‘Bool’を Q_j として，

$$0 \leq Q_j \leq 1 \quad (10)$$

といえる．

4.3.1 文書スコア

構造化文書に問合せを適用する際，本研究では**文書スコア**という概念を導入する．**文書スコア**の概念を導入することによって，構造化文書において，文字列照合検索とベクトル照合検索の評価を統一的に扱うことが容易になる．それは次の理由による．

- 文字列照合検索を，結果のスコアが0か1(つまり，一致か直交) の極端なベクトル照合検索検索とみなすことが可能
- 二種類の照合検索の結果を数値化し，文書構造の下位からボトムアップに計算することが可能

構造を持った文書に問合せを適用した評価を**文書スコア**と定義する．文書ノード D_i について，ある式 Q_j を評価した結果のスカラ値を文書／部分**文書スコア** $S(D, Q_j)$ と呼ぶ．文書スコアは式に依存して決定する文書木中のノードの属性である．本研究で導入する文書スコアは，文書中に含まれる単語数を木構造の下位ノードからボトムアップに足し合わせてゆくことにより，上位のノード

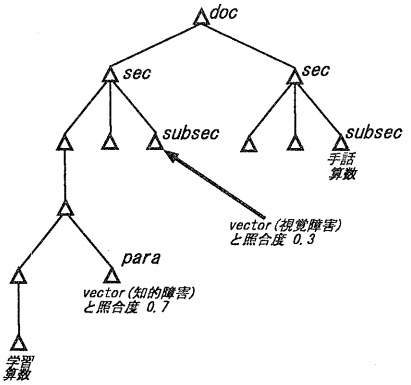

図 3: 文書スコアの例

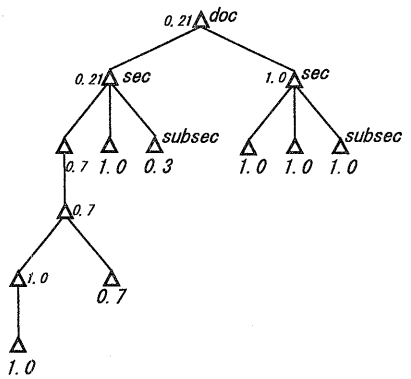

図 4: 文書スコア計算の概略

に含まれる単語数を求め，構造化文書全体をベクトル化する手法である BUS[4] と親和性の高いものである．式 (10) より，

$$0 \leq S(D, Q_j) \leq 1 \tag{11}$$

である． 例を挙げて**文書スコア**を説明する．

(☆) 意味ベクトル '知的障害' に照合する *para* エレメントと意味ベクトル '視覚障害' に照合する *subsec* エレメントをもち，/doc/sec/subsec エレメントに文字列 '手話' を含むような *doc* エレメントを検索せよ

$\{doc|(/doc)$
$\wedge$**SIMILAR**$(/doc//para, vector(\text{知的障害}))$
$\wedge$**SIMILAR**$(/doc//subsec, vector(\text{視覚障害}))$
$\wedge$**CONTAIN**$(/doc/sec/subsec, \text{'手話'})\}$

図 3のような構造をもった文書に対して，(☆) のような問合せが発行されたとする．その評価は，数値化されて図 4のようにボトムアップに計算してゆく．この問合せ評価の計算の過程で文書の構造に割り当てられる数値を本研究では**文書スコア**と呼ぶ．

1. **CONTAIN** 文を評価する．評価の対象となるノードに照合すれば文書スコア 1 を，照合しなければ 0 を割り当てる．
2. **SIMILAR** 文を評価する．評価の対象となるノードでベクトル照合を行い，その評価値を求め，その値を文書スコアとして割り当てる．
3. 前述の **CONTAIN** 文，**SIMILAR** 文の評価の対象が重複している場合，評価の積を文書スコアとする．
4. **CONTAIN** 文，**SIMILAR** 文のいずれの評価の対象にもならない文書の葉ノードには，1 を文書スコアの初期値として割り当てる．
5. 中間ノードでは，子ノードのもつ文書スコアの積を自分の文書スコアとする．
6. **head** に記述された検索対象エレメントに評価が到達すれば，そのときの文書スコアが問合せに対する適合度である．

すなわち，例 (☆) の問合せが発行されたときの**文書スコア**が計算される様子を図 4に示す．このとき，*doc* エレメントの文書スコアは 0.21 であることが分かる．

4.3.2 問合せ評価のアルゴリズム

文書木中の照合するノードに対して問合せのボディ部のルールをボトムアップに評価してゆく．ヘッドに記述されているエレメントに対応するノードに到達した時点での文書スコアが**問合せの評価**である．

4.4 問合せの例

文字列照合検索とベクトル照合検索を統合した問合せの例として次を挙げ，4.1節と 4.2節で提案した二通りの手法での問合せ記述例を示す．

例) **文字列 '肢体不自由' を含み，意味ベクトル '知的障害' に類似する *article* エレメントを検索せよ**

XPath 非拡張手法 $\{a|a \in (//article) \wedge$ **CONTAIN** $(a, \text{'肢体不自由'}) \wedge$**SIMILAR**$(a, vector(\text{'知的障害'}))\}$

XPath 拡張手法 $\{a|a \in (//article[$**contains**$(.,\text{'肢体不自由'})]$ $[$**is_similar**$(a, vector(\text{'知的障害'}))])\}$

5 近接

WWW におけるサーチエンジンの中には利用者に指定された二つの単語が一定の文字数以内で近接していることを検索条件に指定できるものもある．構造化文書で

ない，通常の文書を対象にした従来のサーチエンジンでは，第1章の終わりと第2章の始まりに二つの単語が出現していても，それらの出現は近接とみなされていた．しかし，検索対象として構造化文書を想定すれば，第1章の終わりと第2章の始まりに二つの単語が出現している場合，文字列上では近接していても，文書の意味構造上では必ずしも近接していないと判断することが可能である [5]．

本研究では，構造化文書の特徴を活かした文書の意味構造上での近接の検索条件だけでなく，従来のサーチエンジンで採用されている文字列上での近接の検索条件も重要であると考え，両方を支援する．

対象とする文書木中のノードの集合に対して，これらが相互に近接しているかどうかを評価するために関数 **Prx** を導入する．対象とするノード集合を**目的ノード集合 (Object Node Set)** と呼ぶことにする．

本研究では**目的ノード集合**内の各要素が相互に近接している場合，このブール問合せに高い評価を与える．また，次節で定義する**近接度**が高いときにノード集合内の各要素は相互に**近接**しているものとする．

5.1 近接度

問合せを評価する際に，近接を考慮に入れるための関数，**Prx** を導入する．ただし，$W_1,\ldots,W_i$ は近接を評価する対象の単語集合とする．

関数:*long* **Prx**$(W_1, W_2, \ldots, W_i)$

Prx 関数は，引数に指定した単語集合の各要素が相互に近接している度合いを 0 以上 1 以下のスカラ値として返す．関数 **Prx** を導入することによって，生成規則 [004] が次のように拡張される．

[**004'**] NonExtXPathEx ::= CONTAIN_Ex | SIMILAR_Ex | Prx_Ex

[**004'**] (XPath 非拡張版式) ::= (CONTAIN 式) | (SIMILAR 式) | (Prx 式)

この場合に **Prx** を適用した問合せ例は次のようになる．

(※) 例) 文字列 '**算数**' を含む *article* エレメントを，*article* エレメント中で三つの文字列 '**障害**', '**学習**', '**手話**' がなるべく近接しているように検索せよ

近接を考慮した XPath 非拡張手法で記述した例

$\{a | a \in (//article)$
$\wedge$**CONTAIN**$(a,'$算数$')$
$\wedge$**Prx**$('$障害$','$学習$','$手話$')\}$

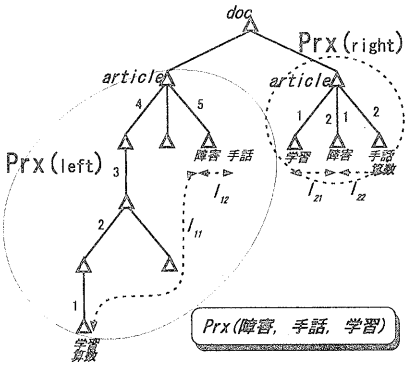

図 5: 問合せの適用例

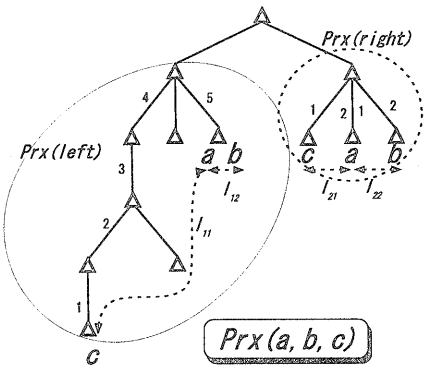

図 6: 近接の説明

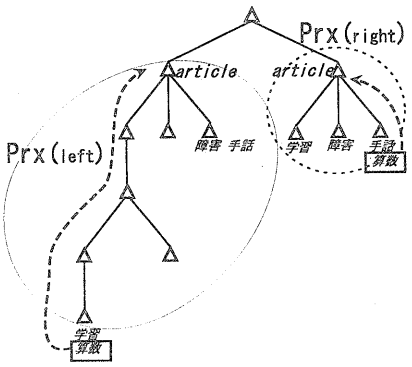

図 7: 問合せ例における文書スコア

5.1.1 親等：文書の構造

文書の意味構造上での近接，**親等**を定義する．具体的には，たとえば，二つの単語 '特殊教育' と 情報検索 が

文章中で単語数の上では同じ5単語離れているケースがあったと仮定する．第一のケースが‘特殊教育’も‘情報検索’も1章に含まれている場合，第二のケースが‘特殊教育’が第1章で‘情報検索’が第2章に含まれている場合であれば，本研究では前者の方が文書の構造上‘特殊教育’と‘情報検索’が近接していると定義する．この概念を[親等]として定量的に定義する．

図6に示すような文書木に対して式$Q=$‘$a \wedge b \wedge c'$が照合したとする．たとえば，図中の実線で囲まれた部分木と，点線で囲まれた部分木がQにマッチングする部分木の候補である．Qに対する**目的ノード集合** $\{n_a, n_b, n_c\}$において，表記上で最初のノードから最後のノードまでにたどる枝の数kを，その照合ノード集合に対する**親等**と定義する．ただし，文字列aを含むノードをn_aで表す．また，g番目の照合ノード集合に対する親等をk_gで表す．照合する部分木の候補が複数ある場合，どの部分木のノード集合の方が文書構造的に相互により近接しているか，**親等**の概念を利用して定量的に比較することができる．

5.1節の(※)の問合せを例に**親等**を説明する．‘*//article*’について，*article*の子孫の部分に注目すればよいことが分かる．次に，‘**CONTAIN**(a,‘算数’)’について，*article*の子孫の部分に‘算数’が2か所出現していることがわかる．最後に**Prx**(‘障害’,‘学習’,‘手話’)について，近接を判断するための**目的ノード集合**の組合せの候補として，たとえば図5の**Prx(left)**と**Prx(right)**が挙げられる．**Prx(left)**の目的ノード集合 $\{n_{\text{‘障害’}}, n_{\text{‘学習’}}, n_{\text{‘手話’}}\}$において，表記上最初のノードである$n_{\text{‘学習’}}$から，表記上最後のノードである$n_{\text{‘手話’}}$まで文書木構造の枝を5本たどる．したがって，**Prx(left)**は5**親等**である．同様にして，**Prx(right)**は2**親等**であることが求められる．以上から，文書構造上の観点からは**Prx(left)**の要素より**Prx(right)**の要素の方が相互に近接していると判断する．

5.1.2 単語数による距離：文字列上の近接

文字列上での近接，**単語数による距離**を定義する．具体的には，たとえば，二つの単語‘特殊教育’と‘情報検索’が文章中で単語の数の上で何単語離れているかが**単語数による距離**である．

Qに対する**目的ノード集合** $\{n_a, n_b, n_c\}$の要素を文書中での表記文字列順(表記単語順)にソートする．表記上の最初のノードから出発して，i番目のノードと$i+1$番目のノードの間の単語数を，g番目の目的ノード集合に対するi番目の**単語数による距離**l_{gi}とする．g番目の目的ノード集合に関する**単語数による距離**L_gは次で定義される．

$$L_g = \sum_{j=1}^{i-1} l_{gj}$$

5.1節の(※)の問合せを例に**単語数による距離**を説明する．図5のような構造を持った文書に対して問合せ(※)が照合されたとする．5.1.1節と同様に，**Prx**(‘障害’,‘学習’,‘手話’)について，近接を判断するための**目的ノード集合**の組合せの候補として，図5の**Prx(left)**と**Prx(right)**が挙げられる．ここで，**Prx(left)**について，**目的ノード集合**の要素を文書中での表記文字列順(表記単語順)にソートすると，(‘障害’,‘学習’,‘手話’)となる．表記上の最初にあたる‘障害’から最後にあたる‘手話’までの単語数が**Prx(left)**に関する**単語数による距離**である．つまり，図5から，$l_{11}+l_{12}$が**Prx(left)**に関する**単語数による距離**である．同様に，**Prx(right)**に関する**単語数による距離**は$l_{21}+l_{22}$ということになる．

5.1.3 近接度の定義

g番目の目的ノード集合に対する近接度**Prx**を次のように定義する．ただし，目的ノード集合の要素数を$i(1 \leq i)$とする．

$$\mathbf{Prx}(W_1, W_2, \ldots, W_i) = \frac{1}{(k_g+1)l_{g_1} + (k_g+1)l_{g_2} +, \ldots, +(k_g+1)l_{g_{i-1}}} \quad (12)$$

また，式(12)の分母は明らかに1以上であるので，

$$0 \leq \mathbf{Prx}(W_1, W_2, \ldots, W_i) \leq 1 \quad (13)$$

である．

5.1.4 近接を考慮した文書スコア

式(10)，式(13)より，近接を考慮した文書スコア$S'(D)$は，

$$0 \leq S'(D) \leq 1 \quad (14)$$

6 実際の文書の木構造と問合せの例

図8，9に研究所の研究紀要の木構造の例を図示する．本研究で提案する手法によって想定される，図8，9に対する問合せの例を以下に示す．

1. 文字列‘学習効果’を含む*section*エレメントを検索せよ

   { *section* | (*//*/section*) ∧ (*section* **CONTAIN** ‘学習効果’) }

2. *vector*(学習障害)に類似したベクトルをもつ*section*エレメントを検索せよ

   { *section* | (*//*/section*)∧(*section* SIMILAR *vector*(学習障害)) }

3. *title*エレメントに文字列‘算数’を含む文書を検索せよ

   { *doc* | (*/doc/*//title*)∧ **CONTAIN**(*title*, ‘算数’) }

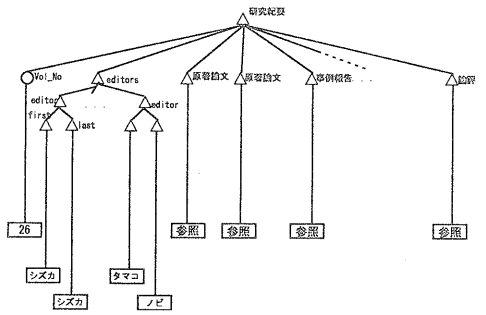

図 8: 研究紀要の木構造

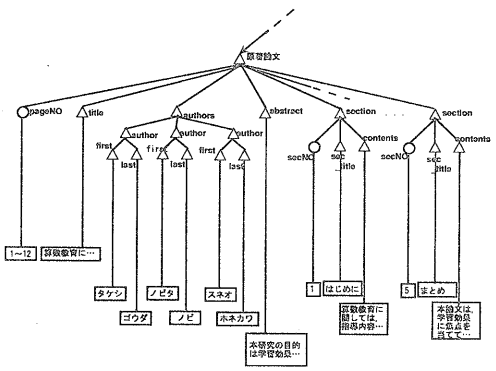

図 9: 原著論文の木構造

4. *author* エレメントにノビタ氏を含み，最後の *section* エレメントのベクトルが *vector*(コンピュータ教育) に類似した文書を検索せよ

$$\{\, doc \mid ((/doc/*//author/first) \wedge \mathbf{CONTAIN}(first, \text{'ノビタ'})) \wedge ((/doc/*//section) \wedge \mathbf{SIMILAR}(section, vector(\text{コンピュータ教育})))$$

7 おわりに

本稿では XPath を利用して文字列照合検索とベクトル照合検索を統合して利用する問合せの記述手法を提案した．本稿で提案したアプローチを利用すれば，文字列照合のもつ厳密なマッチングと，ベクトル照合のもつ意味的マッチングや順位付けの両方利点が文献検索に利用できるだけでなく，XPath がもっている強力な構造記述能力をも利用することができる．

参考文献

[1] World Wide Web Consortium. XML Path Language (XPath) version 1.0. http://www.w3.org/TR/xpath, November 1999. W3C Recommendation 16 November 1999.

[2] Marti A.Hearst. Texttiling:a quantitative approach to discourse segmentation. *Technical report of the University of California at Berkeley*, 1993.

[3] R. Sacks-Davis, T. Arnold-Moore, and J. Zobel. Database Systems for Structured Documents. In *Proc. of the International Symposium on Advanced Database Technologies and Their Integration (ADTI'94)*, pp. 272–273, Oct. 1994.

[4] Dongwook Shin, Hyuncheol Jang, and Honglan Jin. BUS: An Effective Indexing and Retrieval Scheme in Structured Documents. In *Proc. of the 3rd ACM Conference On Digital Libraries (DL'98)*.

[5] 絹谷弘子, 吉川正俊, 植村俊亮. 構造化文書の設計の異種性解消のための「文書群」の導入と検索機能の実現. 情報処理学会研究報告, 00-DBS-121-20 / 00-FI-58-20.