2D-5

XPath 曖昧検索を用いた効果的な XML データ検索

矢野令† 川崎直丸†
東芝ソリューション株式会社†

1. はじめに

近年の XML(Extensible Markup Language)形式のデータ利用の隆盛に伴い、XML データを蓄積する XML データベース(以下、XML-DB)が各社から製品化されてきている。このような背景において、XML-DB には異なるスキーマに準拠する XML データが混在し、それらを横断した検索が要求される。

しかし、開発者が複数のスキーマを熟知して、それらに対する XPath などの検索構文を作成することは困難である。また、XML データの形式や内容自体に意味があるため、XML-DB への格納時に、統一的な構造や語彙に変換することも困難である。

このような場合、一般的には全文検索を行なうことで、検索語を含む複数の XML データを取得する。しかし全文検索方式では、例えば title 要素に「東芝」が含まれている記事を探したくても、関連の低い場所に「東芝」が含まれている記事も該当してしまうため、ノイズが高くなり、検索者の意図が反映できない。

そこで本研究では、検索時に XPath を動的に変換し、他スキーマに準拠する XPath を生成する手法を提案する。これにより、検索者の意図を損ねることなく、ノイズを抑えて、複数の異なるスキーマに準拠した XML データを横断検索できる。本論文では本提案手法を XPath 曖昧検索と呼称する。

2. 関連研究

検索時に XPath などの検索構文を変換するアプローチとしては、[1][2][3]などがある。これらのアプローチは、XML Schema と RDF(Resource Description Framework)を用いて、構造の変換を定義する手法である。本提案では、明確に XML Schema が定義されていない XML データも対象とし、RDF を用いないシンプルな方法を提案する。

3. 提案手法

3.1 概要

本手法の概要と流れを図 1 に示す。

A retrieval technique for XML data using XPath expression transformation
YANO Rei†, KAWASAKI Naomaru†
† Toshiba Solutions Corporation

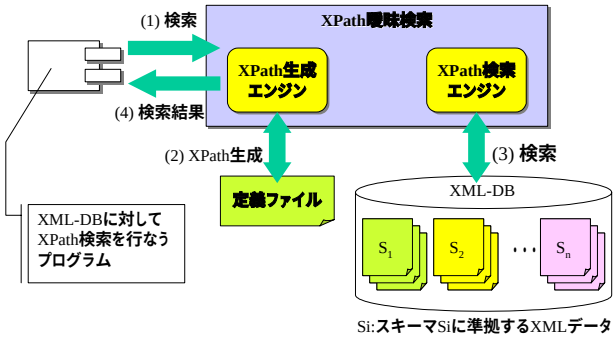

図 1 XPath 曖昧検索概要

(1) 検索を行なうプログラムはスキーマ S_1 に準拠する XPath で検索を行なう。
(2) XPath 生成エンジンは定義ファイルから、S_1 以外のスキーマに準拠する XPath を生成する。
(3) 生成された XPath を union 演算子で連結し、XML-DB を検索する。
(4) 検索条件に合致する、スキーマ S_1～S_n に準拠した XML データを返却する。

以降では、手順(2)の定義ファイルを 3.2 節で、XPath 生成のアルゴリズムを 3.3 節で詳述する。

3.2 定義ファイル

定義ファイルでは、同等の内容を意味する XPath に対して同一のパスラベルを、同一種のスキーマに準拠する XPath に対しては同一のスキーマラベルを付与する。このラベルをもとに他スキーマに準拠した XPath を生成する。定義ファイルの例をリスト 1 に示す。リスト 1 は、h1 要素と title 要素は同じタイトル(title)であることを、h1 要素の class 属性と category 要素は同じ分類(class)であることを定義している。

リスト 1:定義ファイル例
(左からスキーマラベル、パスラベル、XPath を示す)

```
html title /html/body/h1/text()
rss  title /rss/channel/title/text()
html class /html/body/h1/@class
rss  class /rss/channel/category/text()
```

3.3 XPath 生成のアルゴリズム

XPath 生成の手順は、入力 XPath に対応するパスラベルの取得と、対応 XPath の生成に大きく分けられる。パスラベルの取得では、入力 XPath に含まれる「*」や「//」を正規化し、定義ファイルにおけるどのパスラベルに対応するかを判

別する。対応 XPath の生成では、該当パスラベルを元に他スキーマの XPath を取得し、入力 XPath に含まれる述部([]で指定した条件部)を付与する。この時、述部に含まれる XPath についても対応 XPath 生成を行なう。以下に、2 つの処理のアルゴリズムを示す。

- **パスラベルの取得**

A-1. 入力 XPath から述部を削除する。
A-2. A-1 の結果と同等な XPath とパスラベルを定義ファイルから取得する。
A-3. 入力 XPath の述部が持つ XPath と同等な XPath とパスラベルを定義ファイルから取得する。

次に、得られたパスラベルをキーとして、対応 XPath の生成を行なう。

- **対応 XPath の生成**

B-1. A-2 で得られたパスラベルと同一のパスラベルを持つ XPath(候補 XPath と呼称)とスキーマラベルを取得する。
B-2. A-3 で得られたパスラベルと同一のパスラベルを持ち、B-1 で得られたスキーマラベルを持つ XPath を取得する。
B-3. 入力 XPath の述部が持つ XPath を、B-2 で得られた XPath に置換、候補 XPath に付与する。
B-4. B-1 で候補となる XPath 全てについて B-1～B-3 を繰り返す。

リスト 1 の定義フィイル例に基づくと、以下のような対応 XPath が生成される。

入力 XPath

```
/*//h1[@class='level1']/text()
```

対応 XPath

```
/html/body/h1[@class='level1']/text()
/rss/channel[category/text()='level1']/title/text()
```

4. 評価結果

RSS1.0、RSS2.0、Atom、NewsML の XML データを用いて、本提案手法の評価を行なった。この評価では、(1)通常の XPath 検索、(2)全文検索、(3)XPath 曖昧検索による検索結果を比較した。検索は「本文に'XML'の文字列を含むエントリ(記事)」を取得することを目的とした。XML データ全体のエントリ数を表 1 に示す。

表 1：評価に用いた XML データ

スキーマ	全体のエントリ数(件)	本文に'XML'を含むエントリ数(件)
RSS1.0	173	7
RSS2.0	90	4
Atom	72	5
NewsML	246	146

(1)と(3)の入力 XPath には次の XPath を用いた。

```
//item[contains(description, 'XML')]
```

また(2)の全文検索では次の XPath を用いた。

```
/*[contains(.//*, 'XML') or contains(.//@*, 'XML')]
```

(3)の XPath 曖昧検索で用いた定義ファイルをリスト 2 に示す。

リスト 2：定義ファイル

```
rss1.0  entry   /rdf:RDF/item
rss2.0  entry   /rss/channel/item
atom    entry   /feed/entry
newsml  entry   /NewsML
rss1.0  body /rdf:RDF/item/description
rss2.0  body /rss/channel/item/description
atom    body /feed/entry/summary
newsml  body /NewsML/NewsItem/NewsComponent/
                    NewsComponent/ContentItem/DataContent
```

それぞれの手法による検索結果を表 2 に示す。

表 2：検索結果

検索手法	該当件数
(1)XPath 検索	11
(2)全文検索	256
(3)XPath 曖昧検索	162

検索結果のうち、(2)の全文検索では、NewsML において本文以外に含まれる'XML'の文字列も該当してしまうため、ノイズが多くなる結果となっている。また、RSS/Atom の検索結果については、エントリ単位での取得ができないため、検索者の意図を損ねる検索結果となる。

一方、(3)の XPath 曖昧検索では、(1)の検索結果に加えて、Atom/NewsML の XML データも適切に取得できていることが解る。加えて、検索者の意図を損ねずに、エントリ単位での取得ができている。

以上より、異なるスキーマに準拠する XML データを横断して検索する手法として、本提案手法は有効であると言える。

5. 今後の課題

本提案手法の今後の課題としては、定義ファイルの作成方法が挙げられる。定義ファイルを人間が記述するのは GUI を用いたとしても手間がかかる。XML-DB に蓄積された XML データから機械学習を行ない、XPath の意味的な類似性を導出することで、半自動的に対応関係を作成する必要がある。

参考文献

[1] A. Y. Halevy, Z. G. Ives, P. Mork, I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications, WWW2003, pp.556-567, 2003
[2] I. F. Cruz, H. Xiao, F. Hsu. An ontology-based framework for XML semantic integration. IDEAS'04, pp.217-226, 2004
[3] Tous R, Garcia R, Rodriguez E, Delgado J. Architecture of a Semantic XPath Processor. Application to Digital Rights Management. EC-Web 2005, pp.1-10, 2005