Xtree와 문서 유사도에 기반한 XML 문서 검색*

이은정[0] 박우창
덕성여자대학교 전산대학원
{ejlee[0],ucpark}@duksung.ac.kr

Searching XML Documents based on Xtree and Documents Similarity

Eunjeong Lee[0] Uchang Park
Duksung Women's University, Department of Computer Science

요 약

인터넷의 성장으로 인해 준구조적인 문서 표준의 하나인 XML 문서의 사용이 증가하고 있다. 본 연구는 인터넷이나 XML 데이터베이스에서 저장된 스키마 정보가 주어지지 않는 많은 양의 XML 문서를 대상으로 주어진 XML 문서에 가장 가까운 문서들을 찾는 방법을 제시한다. 먼저 XML 문서들의 스키마 정보를 얻기위하여 XML 문서에 해당하는 카디널리티 정보를 포함하는 xtree로 변환하고, 변환된 문서들에 대하여 XML 각 요소에 대한 유사도와 문서 구조에 대한 유사도를 계산하여 가장 유사도가 가까운 XML 문서를 결과로 제시한다. 본 논문의 방법은 스키마가 알려지지 않는 XML의 문서들에 대한 검색을 할 수 있고 유사도를 이용하여 문서의 근사 검색을 할 수 있는 장점이 있다.

1. 서 론

XML(Extensible Markup Language)은 인터넷 상 에서 준구조적인 데이터 표현의 표준으로 XML 문서의 구조는 대부분 DTD(Document Type Definition)에 의해 기술되어 진다. XML 문서의 증가에 따라 XML 문서를 대상으로한 검색의 필요성이 증가하고 있고, 기존의 XML 문서 검색은 스키마에 해당되는 DTD가 제공된다. 그러나 DTD가 없는 XML 문서 검색은 기존의 검색 방법을 사용할 수 없다. 본 연구는 인터넷이나 XML 데이터베이스에 저장된 DTD 같은 스키마 정보가 주어지지 않는 많은 양의 XML 문서를 대상으로 주어진 XML 문서에 가장 가까운 문서들을 찾는 방법을 제시한다.

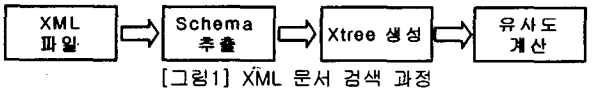
본 연구는 먼저 스키마 정보가 없는 XML 문서에서 XML Schema 정보를 추출하여 xtree로 변환한다. 다음으로 xtree 구조에서 유사도를 계산한다. 스키마 정보가 없는 XML 문서에서 DTD 자동 추출 방법은 [4,6]에서 연구된 바 있다. DTD는 간편하기 때문에 XML 문서의 스키마를 표현하는 데 많이 사용되고 있지만 XML DTD는 XML 문서의 요소 반복 횟수를 지정한다는 것보다 반복 여부에 대한 정규식만을 제공한다. 유사도 측정에 있어서는 DTD 보다 XML Schema를 이용하는 방법이 반복의 횟수까지 고려하므로 정확한 검색을 할 수 있다. Boris[1]는 XML 문서로부터 XML Schema를 추출하는 알고리즘을 제시한 바 있다. Behrens[5]는 XML 스키마를 표현하는 xtree를 제시하였다. DTD가 주어진 문서에서 XML 문서들간의 유사도 기반 군집화는 Lee[2]가 연구하였다. 본 연구에서도 이 방법을 이용하여 문맥 자유 문법(Context Free Grammar) 기반의 XML Schema를 유도한다. XML Schema를 xtree로 변환하고 유사도를 계산하여 XML 문서를 검색한다.

본 논문의 구성은 다음과 같다. 2장에서는 Extended CFG와 xtree에 관한 설명과 유사도 계산 알고리즘을 제시하고, 3장에서는 검색 시스템 설계를 보이며, 4장에서는 결론 및 향후 연구 과제를 설명한다.

*본 연구는 한국과학재단 연구비 (과제번호 R06-2002-003-01005-0) 지원으로 수행되었음

2. XML 문서 검색 알고리즘

본 논문에서는 XML 문서에 대하여 ECFG(Extended Context Free Grammars)를 기본으로 하여 XML Schema를 추출한다. 추출 후 XML 스키마를 xtree로 변환하고, 검색 대상 XML 문서들에 대하여 유사도를 계산한다. 이 과정을 그림으로 보이면 다음과 같다[그림1]



[그림1] XML 문서 검색 과정

2.1 확장 문맥 자유 문법(Extended CFG)

첫 번째 단계로 XML 문서를 ECFG(Extended Context Free Grammar)로 변환한다. Boris[1]의 변환 방법을 본 연구에서 사용한다. ECFG는 기존의 CFG에 비단말과 단말 노드의 발생 빈도를 표현하는 방법이다. ECFG는 G=(T,N,D,δ,Start)의 5개의 튜플에 의해 정의 되어 진다. T, N과 D는 각각 단말(terminal) 항, 비단말(nonterminal) 항과 데이터타입들을 나타내며, Start는 시작 비단말 값이고, δ은 A-> α의 생성 규칙의 집합을 나타낸다. 여기서 α는 항에 관한 범위 정규식이다. XML 문서의 요소와 ECFG 사이의 관계를 표시하면 표 1과 같다.

[표1] XML Schema와 ECFG 요소들의 대응 관계

XML Schema 요소	ECFG 기호
Element name(tag)	단말(terminal) 항
Element definition	생성규칙(production)
Elementary datatype	데이터타입(datatype)
Named complex type	비단말(nonterminal) 항
Abstract complex type	비단말(nonterminal) 항
Complex type definition	1개 이상 생성규칙(production)
Sequence element group	생성규칙의 연속 패턴
Choice element group	생성규칙의 disjunction 패턴

XML 문서에서 ECFG를 유도하는 방법은 알고리즘 1과 같다.

[알고리즘 1] XML 데이터에서 문법(ECFG)과 Schema 추출

0. XML 문서를 요소집합 I로 분리한다.
1. 구조적으로 나타내어진 I 집합으로부터 XML 문서를 확장 문맥 자유 문법으로 변환한다.
   1.1.비단말 요소를 생성한다;
   1.2.비단말 요소를 유사한 내용과 문맥에 따라 병합한다;
   1.3.단말 노드의 데이터타입을 결정한다;
   1.4.범위 정규 표현식의 비단말 생성규칙의 내용 집합을 일반화한다.
2. ECFG G를 XML Schema로 변환한다.

그림2는 XML 문서에서 유도된 XML Schema의 ECFG 표현법으로 나타내면 다음과 같다.

ECFG G=(T,N,D,δ,Start) 이고,
T={author,zip,city,street,firstname,lastname},
N={Start,NameType,AddrType},
D={String,PositiveInteger},
δ의 규칙은 다음과 같다.

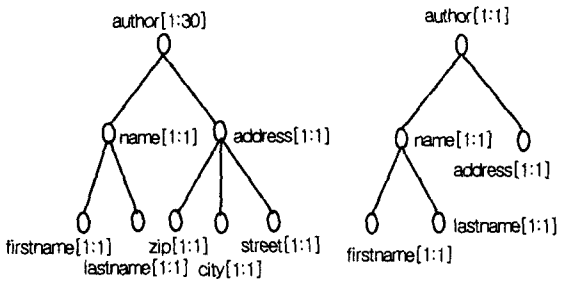
Start→(author:String) [0:30] (name:NameType) [0:1]
(addresstype:AddrType) [1:1]
NameType→(firstname:String) [1:1]
(lastname:String) [1:1]
AddrType→(zip:PositiveInteger) [1:1]
(city:String) [1:1] (street:String) [1:1]

```
<element name='author' type='String' maxOccurs='30'>
 <complexType>
  <element name='name' type='NameType' />
  <element name='address' type='AddressType' />
 </complexType>
 <complexType name='NameType'>
  <group minOccurs='1' maxOccurs='1'>
  <element name='firstname' type='String' />
  <element name='lastname' type='String' /></group>'
 </complexType>
 <complexType name='AddrType'>
  <group minOccurs='1' maxOccurs='1'>
  <element name='zip' type='PositiveInteger' />
  <element name='city' type='String' />
  <element name='street' type='String' /></group>
 </complexType>
</element>
```

[그림2] XML Schema 예

2.2 Xtree

본 논문에서는 Behrens[5]가 제시한 xtree(XML Tree)에 카디널리티 정보를 노드에 추가하여 제시한 확장된 xtree를 사용한다. xtree란 기존의 DTD를 트리 형태로 나타내는 방법에서 착안한 것으로, XML Schema를 트리 형태로 나타낸 것이다. 확장된 xtree는 ECFG의 정규표현 방법에서 나타나는 카디널리티를 포함한 tree의 형태이다. 그림3의 (a)는 그림 2를 확장된 xtree로 나타낸 것이다.



(a) 그림2에 대한 xtree 예 (b) xtree의 다른 예

[그림3] 확장 xtree의 예.

2.3 유사도 기반 XML 문서 검색

주어진 XML 문서와 XML 데이터베이스에 저장된 여러 개의 XML 문서들 중 비슷한 XML 문서를 검색하기 위해서 스키마 요소들 사이의 유사도 측정이 필요하다. XML의 DTD의 카디널리티 연산자(cardinality operator)인 ?, *, +은 한 요소가 인스턴스 문자 안에서 몇 번 나타나는지를 표현한다. XML Schema는 DTD에 비교하여 요소 선언에서 속성의 형태로 표현할 수 있는 두 개의 제한 조건을 사용할 수 있는데 그것은 minOccurs, maxOccurs 두 가지이다. 이 속성들의 값은 요소가 몇 번 나타날 수 있는지를 지시하므로, DTD의 카디널리티 연산자보다 훨씬 정교하게 사용할 수 있다.

본 연구의 XML문서간 유사도(similarity) 측정은 Mong[2]에서 사용한 DTD 기반 유사도 측정 방법을 사용한다. 단 본 연구는 XML Schema를 사용하기 때문에 Mong의 CS(제약 유사도) 계산식과 달리 문서의 정확한 카디널리티에 기반한 (식 1)을 사용한다. XML 문서의 유사도 ES는 다음과 같이 계산된다.

ES(Element Similarity) = a×SS + b×LCS + c×IDS
a+b+c=1, a,b,c 값은 사용자가 임의로 정의
SS = PCC × BS
LCS = PCC × BS
PCC = (경로에 대한 BS의 합)/(경로의 크기)
IDS = (자식노드의 BS의 합)/(자식의 수)
BS = OS + CS
OS = 0(같은요소 없음), 0.8^{depth}(같은요소 있음)
CS = (식1 참조)

본 논문에서 제시하는 CS 계산을 위한 카디널리티 테이블은 아래의 [표2]와 같이 나타낼 수 있다. 입력 받은 XML 문서와 DB안에 있는 XML 문서들의 카디널리티를 적용하여 좀 더 비슷한 문서를 검색하게 된다. XML Schema에서 minOccurs 값이 0이고, maxOccurs 값이 1이라면 범위식[0:1]로 표현되어진다. [표2]에 나타나 있는 CS(제약 유사도) 중 A와 B는 검색입력의 범위식이 [l1,u1], 검색대상 범위식이 [l2,u2]일 경우 다음과 같이 계산한다. CS 값은 0~1 사이의 값을 가진다.

CS = [(l1+l2 - D1) / (l1+l2) × 1/2] +
[(u1 + u2 - D2) / (u1+u2) × 1/2],
D1 = abs(l1-l2), D2 = abs(u2-u1) [식 1]

[표2] CS(제약 유사도) 계산 테이블.

검색대상범위식 / 검색입력범위식	[0:n]	[1:n]	[n1:n2]	none
[0:n]	1	0.5	A	0
[1:n]	0.5	1	B	0
[n1:n2]	A	B	1	0
none	0	0	0	1

위의 식에 기반하여 유사도 ES를 얻기 위한 계산 예는 다음과 같다.

[예제1] 그림3의 확장된 xtree를 이용하여 유사도를 계산해 보면 다음과 같다.

OS(author[1:30],author[1:1])=1
CS(author[1:30],author[1:1])=Low(0.5)+Upper(0.03)=0.53
BS(author[1:30,author[1:1])=0.67 (w1+w2=1)
(CS에서 이용하는 유사도 카디널리티의 가중치를 크게 하기 위해서 w2의 값을 0.7로 크게 준다.)
PCC(author[1:30],author[1:30],author[1:1],author[1:1], Threshold)=0.91
SS(author[1:30],author[1:1])=PCC*BS=0.61
BS(firstname[1:1],firstname[1:1])=1.0
BS(lastname[1:1],lastname[1:1])=1.0
PCC(firstname[1:1],author[1:30],firstname[1:1],author[1:1],Threshold)=(0.61+1.0+1.0)/3=0.87
PCC(lastname[1:1],author[1:30],lastname[1:1],author[1:1],Threshold)=(0.61+1.0+1.0)/3=0.87
LS(firstname[1:1],author[1:30],firstname[1:1],author[1:1], Threshold)=0.87
LS(lastname[1:1],author[1:30],lastname[1:1],author[1:1],Threshold)=0.87
author들의
LC(author[1:30],author[1:1],Threshold)=(0.87+0.87)/max(2,5)=0.35
ID(name[1:1],name[1:1])=BS(name[1:1],name[1:1])/max(2,2)=1.0
따라서, 유사도 ES=0.33*0.61+0.33*0.35+0.33*1.0=0.65 이다.

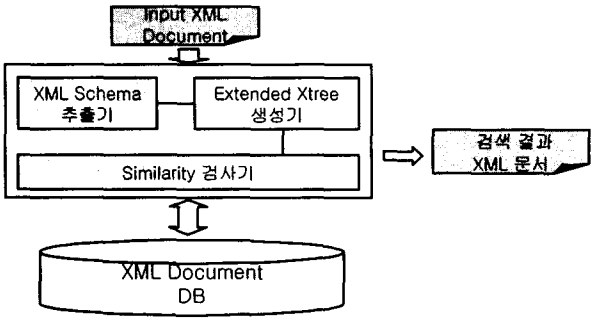
3. 검색 시스템 설계

본 논문에서는 [그림4]과 같이 검색 시스템을 설계하였다. 전체 과정은 알고리즘 2와 같다.

[알고리즘 2] 검색 알고리즘

```
알고리즘 : 유사도 기반 XML 문서 검색
   Input: XML 문서
   Output: XML Schema와 유사한 요소 매치
Step 1: Input된 XML Document에서 XML Schema를 추출
Step 2: 추출된 Schema를 이용하여 Extended xtree생성
Step 3: 생성된 Extended xtree를 이용하여 유사성 검색
while Order ≥ Threshold do {
      Step 4: Similarity의 Order를 계산
      Step 5: XML DB에서 가장 유사한 XML Document 검색
}
```

알고리즘 2에 기반한 검색 시스템 구조는 그림 4와 같다. 입력받은 XML 문서와 유사한 것을 XML DB로부터 검색하기 위해서 입력 문서에 대하여 다음의 3가지 단계가 필요하게 된다. 첫째는 XML Schema 추출기이고, 둘째는 확장 xtree 생성기이며, 셋째는 유사도 검사기이다. 검색 단계는 다음과 같다. 첫째, 검색할 XML 문서를 입력 받는다. 둘째, 입력받은 문서가 XML Schema 추출기로 들어가서 XML Schema가 추출된다. 셋째, 추출된 Schema가 확장 xtree 생성기에 들어가서 확장 xtree가 생성된다. 넷째, 이미 존재하는 XML 문서들과 유사도를 검사하기 위해서 생성된 확장 xtree가 유사도 검색기에서 검사를 하게 된다. 다섯째, DBMS의 확장 xtree와 유사도의 정도를 계산하여 가장 유사한 문서를 검색하게 된다.



[그림4] 유사도 기반 XML 문서 검색 시스템

4. 결론 및 향후 연구 과제

본 연구는 스키마가 주어지지 않은 XML 문서를 질의에 사용하여 XML 문서들 중에서 유사한 문서를 검색하는 시스템을 설계하였다. XML 문서 검색을 위하여, XML 문서에서 XML Schema를 추출하여 확장 xtree를 생성하고, xtree에 유사도를 측정하기 위한 카디널리티를 표현하여 문서간 유사도를 계산하였다. 스키마 추출시 XML DTD는 XML Schema를 사용하는 경우보다 카디널리티 표현의 제약점이 있기 때문에 XML Schema를 사용하였다. 연구를 통하여 첫째, DTD 같은 스키마 정보가 주어지지 않은 XML 문서에서의 Schema를 이용하여 확장 xtree를 구성하였고, 둘째, xtree를 이용하여 카디널리티를 포함한 유사도를 계산하였다.

향후 연구 과제는 설계에 따른 검색 시스템을 구현하고, 실제 XML 데이터에 적용하는 것이다.

5. 참고문헌

[1] Boris Chidlovskii, "Schema Extraction from XML Collections," Proc. ACM/IEEE-CS joint conference on Digital libraries, 2002.
[2] Mong Li Lee, Liang Huai Yang, Wynne Hsu, Xia Yang, "XClust:Clustering XML Schemas for Effective Integration," Proc. International Conference on Information and Knowledge Management, 2002
[3] Jon Duckett 외 8명, PROFESSIONAL XML Schemas, Wrox, 2001
[4] Minos Garafalalos, Aristides Gionis Rajeev Rastogi, S.Seshadri, Kyuseok Shim, "XTRACT:A System for Extracting Document Type Descriptors from XML Documents," In Proc. ACM SIGMOD, 2000.
[5] Ralf Behrens, "A Grammar Based Model for XML Schema Integration," BNCOD, 2000.
[6] Jason San Key, Raymond K.Wong, "Structural Inference for Semistructured Data," Proc. international conference on Information and Knowledge Management, 2001.