XML데이터를 위한 효율적인 구조 정보 추출 기법

민 준 기†

요 약

XML 데이터가 웹 상의 데이터 표현 및 교환의 표준으로 각광 받음으로써, XML에 대한 관심이 증대되고 있다. XML 문서의 구조 정보는 몇 가지 중요한 역할을 수행한다. 이러한 중요성에도 불구하고 XML 문서의 구조정보는 필수 요소가 아니다. 따라서, 이러한 구조 정보를 추출하기 위한 다양한 연구들이 진행되어 왔다. 본 논문에서, 우리는 XML 문서를 위한 간결하고 정확한 DTD를 추출하는 기법을 제안한다. 특히 XML 문서의 구조 정보를 위한 DTD의 내용 모델을 DTD와 XML Schema의 혼합 내용(mixed contents)의 타당성 제약 조건을 이용하여 제한하고 본 논문에서 제안하는 몇 가지 경험적 규칙들을 적용함으로써, 우리는 간결성과 효율적을 이룩하였다. 실제 DTD를 이용한 실험을 통하여 본 논문에서 제안하는 기법이 기존의 접근 방법들에 비하여 뛰어남을 보였다.

키워드 : XML, DTD, XML Schema

Efficient Structural Information Extraction for XML Data

Min Jun-Ki†

ABSTRACT

There has been an increasing interest in XML since it is spotlighted as the standard for data representation and exchange in the Web. The structural information for XML documents serves several important purposes. In spite of its importance, the schema is not mandatory for XML documents. Thus, much research to extract structural information for XML document has been conducted. In this paper, we present a technique for efficient extraction of concise and accurate DTD for XML documents. By restriction of DTD content model using the mixed content model of DTD and XML Schema as well as applying some heuristic rules proposed in this paper, we achieve the efficiency and conciseness. The result of an experiment with real life DTDs shows that our approach is superior to existing approaches.

Key Words : XML, DTD, XML Schema

1. 서 론

최근 정보화 사회가 발전함에 따라서 다양한 형태의 전자 데이터들의 양이 급속하게 증가하고 있는 실정이며, 다양한 분야에서 수집, 생성된 정보들을 공유하고, 활용하고자 하는 상호 운영성에 대한 필요성이 점점 증가하고 있다. 따라서 W3C (WWW Consortium)에서는 정보 표현 및 교환의 표준 포맷으로 XML (eXtensible Markup Language) [4]을 제안하였다. 기존의 인터넷은 HTML(Hypertext markup language)의 등장과 더불어 비약적으로 성장해 왔다. 그러나, HTML은 주로 문서의 정보보다는 문서의 외형(appearance)을 표현할 수 있는 기능을 제공하고 있다. 따라서, 문서의 논리적 구조 및 문서의 의미 표현 능력과 같은 요구 사항을 만족시키지 못하고 있는 실정이다. 정보 검색 (information retrieval) 분야에서 이러한 요구 사항을 만족하는 SGML(Standard Generalized Markup Language) [7]이 제안 되었었다. 그러나, SGML은 구문의 복잡성으로 인하여 많은 응용 분야에서 외면되었다. 따라서, SGML보다 간결하면서도 문서의 구조 및 의미를 손쉽게 표현할 수 있고 새로운 태그와 속성을 새로이 추가할 수 있는 기능을 제공하는 XML은 인터넷 기반의 다양한 응용 분야에서 각광받고 있다.

XML 데이터는 엘리먼트(element)들이 계층적으로 내포된 집합으로 구성되며, 각 엘리먼트들은 시작 태그와 끝 태그로 표현된다. 각 태그는 해당 엘리먼트의 의미를 표현한다. 이러한 태그를 이용하여 XML은 XML 문서 내의 내용(content) 및 표현(representation), 즉 구조를 구분한다. 덧붙여서, 각 엘리먼트는 데이터 값을 가지거나 내포된 자식 엘리먼트의 연속(sequence)을 가질 수 있으며 이름-값 쌍으로 이루어진 애트리뷰트(attribute)들을 가질 수 있다. 특히, 엘리먼트 내에는 동일한 태그를 지니는 자식 엘리먼트들이 임의의 순서로 생성될 수 있다.

이러한 다양한 XML 문서에 대한 구조 정보를 표현하기 위하여 Document Type Definition (DTD) [4], Document Content Description (DCD) [3], XML Schema [8]와 같은 구

† 정 회 원 : 한국기술교육대학교 인터넷미디어공학부 조교수
논문접수 : 2006년 10월 17일, 심사완료 : 2007년 4월 9일

조 표현 언어들이 제안되었다. DTD나 XML Schema와 같이 XML 문서의 구조 정보를 명시하는 언어들은 정규 표현식 (regular expression) 등을 이용하여 임의의 엘리먼트의 자식 엘리먼트들의 연속(sequence)이 지켜야 하는 구조 정보를 표현한다.

XML 데이터에 대한 구조 정보는 다음과 같은 중요한 목적에 사용될 수 있다. 우선, XML 문서의 구조를 설계하는 설계자에게 기존의 XML 문서에 대한 구조 정보를 알려 줄 수 있다. 또한, XML 데이터를 데이터 베이스에 저장하는데 있어서 XML 문서의 구조 정보는 보다 효율적인 저장 방안을 생성하는데 중요한 정보로써 사용될 수 있다 [17]. XML 데이터를 검색하기 위하여 XPath, XQuery 와 같은 언어들이 W3C에서 제안되었다. XML 질의 언어들은 트리 형태의 모델을 따르는 XML 데이터를 검색하기 위하여 경로 표현식 (Path Expression)을 사용한다. 이러한 질의 처리를 하는데 있어서 구조 정보를 활용하면 보다 효율적으로 질의 결과를 얻을 수 있으며 또한 보다 효율적인 질의 계획 수립을 위한 통계 정보 추출 및 유지에 많은 도움을 얻을 수 있다[12].

그러나 이러한 중요성에도 불구하고, XML 문서에 대한 구조 정보는 필수 요소가 아니며, 인터넷 상에 존재하는 많은 XML 문서는 자신의 구조 정보를 동반하지 않고 있다. 다시 말하면, 기존의 관계형 데이터와 같은 것은 항상 해당 데이터의 구조 정보, 즉 schema가 항상 같이 존재하지만, 임의의 XML 문서에 대한 구조 정보를 표현하는 DTD 나 XML Schema는 존재할 수도 있고 존재하지 않을 수도 있다. 이러한 유연성은 XML 데이터의 확산에 많은 도움을 주었으나, 구조 정보가 동반되지 않은 XML 문서들은 구조 정보로부터 얻을 수 있는 장점들을 활용할 수 없다. 따라서, XTRACT[9], DDbE[2], DTD-Miner[13], XStruct[11]와 같은 XML 문서로부터 구조 정보를 추출하는 기법들이 제안되었다.

위에서 언급한 바와 같이 DTD등은 정규 표현식을 이용하여 구조 정보를 표현한다. 어떠한 예제로부터 정규 표현식을 생성하는 것은 기계 학습 (machine learning) 분야에서 오랫동안 연구되어 왔다 [1, 5]. 그러나, 결정적 유한 오토마타 (deterministic finite automata)를 이용한 표현 방식을 이용한다고 할 때, 모든 가능한 정규 언어를 다항식 시간 내에 파악 할 수 없음을 Angluin[1]이 보였으며, 또한 사전 지식 없이 XML 문서를 생성한 자의 의도를 파악하기는 매우 어렵다.

다음의 예제 XML 문서를 살펴보자.

```
<book>
        <title> title1 </title>
        <author>
<first>first1 </first> <last> last 1 </last>
        </author>
        <author>
                <last> last2 </last> <first> first2 </first>
        </author>
</book>
```

(그림 1) 간단한 XML 문서 예제

직관적으로, 그림 1에 있는 author 엘리먼트에 대한 간단한 구조 정보는 단순히 모든 자식 엘리먼트 (즉. first 와 last 엘리먼트)의 이름을 (first|last)* 와 같이 나열하는 것이다. 여기서 '|'는 정규 표현식의 or를 나타내며 *는 해당 요소가 0번 이상 나타날 수 있음을 표현한다. 이러한 접근 방법은 간결한 구조 정보를 제시하지만, 자식 엘리먼트 간의 순서 정보가 사라지는 단점이 발생한다. 이에 반하여, 가장 정확한 구조 정보는 or 연산자를 이용하여 모든 자식 엘리먼트들의 연속을 합병하는 것이다. 즉 ((first last) | (last first)) 로 author 엘리먼트의 구조 정보를 표현하는 것이다. 그러나 이러한 방법은 구조 정보의 크기가 커지는 경향이 있다. 즉, 간결성과 정확성은 서로 상반된 관계에 있음으로, 그 사이의 적절한 균형을 파악하는 것은 매우 어려운 문제이다. 따라서, 본 논문의 목적은 임의의 엘리먼트의 자식 엘리먼트들의 연속에 대한 구조 정보를 간결하면서도 정확한 정규 표현식을 효율적으로 유출해내는 것이다.

본 논문에서 간결성을 달성하기 위하여 구조 정보를 표현하는 제한된 구조 정보 표현 모델을 고안하였으며, 효율성을 위하여 구조 정보 생성을 위한 경험적 규칙론을 생성하였다. 본 논문에서 제안하는 제한된 구조 정보 표현 모델은 DTD 형태로 바로 변환이 가능하며, 간략화 된 XML Schema 형태로도 변환이 가능하다. 또한 타입 추론 기능을 이용하여 엘리먼트가 가지고 있는 데이터 값의 타입을 추론할 수 있다.

본 논문에서 제안하는 방법에 대한 효율성을 검증하기 위하여 실제 DTD 이용하여 실험을 수행하였으며, 실험 결과는 본 논문에서 제안하는 방법에 의한 결과 DTD가 높은 정확성을 보여주었으며 기존의 기법에 비하여 결과 DTD 생성에 있어서 20에서 200배 더 빠름을 보였다.

본 논문의 나머지 부분의 구성은 다음과 같다. 2장에서는 기존의 구조 정보 추출 기법들에 대하여 살펴 본다. 3장에서는 본 논문에서 제안하는 효율적인 구조 정보 추출 기법을 대하여 자세히 기술하고 4장에서 실험을 통하여 제안된 기법의 성능을 평가하고 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

XML 데이터로부터 구조 정보를 추출하기 위한 연구들은 많이 있어왔다[10, 14]. 그러나, 기존의 기법에 의하여 추론된 구조 정보는 그 질이 너무 떨어지거나, 또는 구조 정보를 추출하는데 너무 많은 시간을 낭비하게 된다.

XML 문서로부터 간결하고 정확한 구조 정보를 추출하는 문제를 최초로 제기한 연구는 XTRACT[9]이다. XTRACT는 XML 문서의 구조 정보를 표현하는 언어로 DTD를 선택하였다. XTRACT에서는 여러 개의 DTD 중에서 MDL 원칙(Minimum Description Length Principle)[15]을 따르는 DTD를 선택한다. MDL 원칙 하에서 가장 좋은 DTD는 DTD 자체를 표현하기 위한 정보량과 그 DTD를 이용하여 표현된 실제 데이터의 양이 최소화된 것이다. 예를 들어, 엘리먼트의 연속들의 집합 I = {ab, abab, abababab}가 있을 때,

이를 표현할 수 있는 DTD는 (a|b)*, (ab)* 등이 있다. 첫 번째 DTD (a|b)*를 표현하기 한 정보량은 6이고 이를 가지고 첫 번째 엘리먼트 연속 ab를 표현하는데 필요한 정보량은 3이다. 이는 우선 (a|b)가 총 2번 반복됐다는 정보를 위한 1, 그리고 처음은 a가 선택되었다는 정보 1, 두 번째는 b가 선택됐다는 정보 1. 따라서 합은 3이 된다. 마찬가지로 abab를 표현하기 위해서는 5의 정보량이 필요하다. 따라서, (a|b)*를 위한 총 정보량은 (6+3+5+7)= 21이다. 두번째 DTD (ab)*를 표현하기 위해서는 정보량 5가 필요하다. 이 DTD를 가지고 첫 번째 엘리먼트 연속 ab를 표현하기 위해서는 (ab)가 1번 반복되었다는 정보만 있으면 된다. 따라서 정보량은 1이다. abab를 표현하기 위해서도 정보량은 1만 필요하다. 따라서, (ab)*의 총 정보량은 5+1+1+1 = 8이 된다. 따라서, MDL 원칙하에서는 (a|b)* 보다는 (ab)*가 엘리먼트의 연속들의 집합 I를 보다 잘 표현한다고 말할 수 있다.

XTRACT에서 엘리먼트 연속 집합을 표현할 수 있는 모든 가능한 DTD들과 엘리먼트 연속들의 정보량을 계산하는 것은 NP-hard 문제 중의 하나인 FLP(Facility Location Problem)과 같은 문제여서 시스템 성능에 큰 부담이 된다[9]. 따라서 XTRACT에서는 FLP의 approximation 방법 중 하나인 randomized 알고리즘을 이용하여 O(N2logN)의 시간 복잡도가 필요하다.

Moh 등 [13]은 XML 문서로부터 DTD를 추출하기 위하여 DTD-Miner 란 방법을 제안하였다. DTD-Miner에서는 XML 문서를 트리 형태의 그래프로 표현하고, 이 트리를 탐색하면서 DTD를 위한 신장 그래프(spanning graph)를 생성하는 방법을 제안하고 있다.

Wong과 Sankey[20]는 엘리먼트 T의 자식 엘리먼트 연속의 집합을 프리픽스 트리 오토마톤 (Prefix Tree Automaton: PTA)으로 표현한다. 이 경우 현재 XML 문서에 나타나 있는 엘리먼트 연속만을 표현할 수 있음으로 Minimum Message Length(MML) [18] 개념을 이용하여 PTA 상에 나타나는 상태들을 합병하여 일반화된 구조 정보를 추출하는 기법을 제안하였다. 이는 위의 XTRACT에서 사용한 MDL principle과 유사한 개념으로 성능 저하의 요인이 된다.

또한 [19]에서는 XML 트리 노드들 중 유사한 입력 간선과 출력 간선을 가지는 것들을 그룹화하는 지속적 클러스터링 기법을 이용한 개략적 구조 정보 추출 기법을 제안하였다. 이는 단순히 유사 입력 및 출력 간선만을 그룹화 함으로써 결과 구조 정보의 질이 떨어지게 된다. 또한 XStruct [11]에서는 복수개의 XML 문서들에 대한 공통된 구조 정보를 추출하는 방안을 제안하였다.

위에서 언급한 기존의 기법들은 XML 엘리먼트의 구조를 기술하기 위한 내용 모델에 대한 제한을 두고 있지 않아서 구조 생성을 위한 탐색 공간이 커지게 되어 결과 생성을 위한 시간이 오래 걸리며 생성된 구조 정보가 간결하지 못하다는 문제점을 지니게 된다.

3. 구조 정보 추출

본 절에서는 본 논문에서 제안하는 구조 정보 추출 기법에 대하여 자세하게 다룬다. 본 논문에서 제안하는 방법은 임의의 엘리먼트 T에 대한 자식 엘리먼트들의 연속들의 집합으로부터 구조 정보를 추출 하는 것으로써, 복수 개의 XML 문서로부터도 공통된 구조 정보를 추출 할 수 있으나, 설명의 편의를 위하여 하나의 XML 문서로부터 구조 정보를 추출하는 것을 가정한다.

또한 본 논문에서 제안하는 기법을 생성된 구조 정보는 DTD나 XML Schema로 표현이 가능하다.

3.1 제한된 엘리먼트 내용 모델

앞에서 언급한 바와 같이, 이미 많은 연구자들은 예제로부터 정규 표현식을 유추하는 것의 어려움을 인식하고 있다. 따라서, 기존의 관련된 많은 문헌들처럼, 우리는 생성되는 정규 표현식의 구조를 제한한다. 엘리먼트 내용 모델[4]은 XML 문서 내의 한 엘리먼트의 내용을 기술한다. 우선, 간결성을 위하여, 우리는 XML 문서에 나타나는 각 엘리먼트의 이름 (즉, tag)을 고유한 기호로 표현한다. 또한, 효율적으로 내용 정보를 추출하기 위하여, 우리는 엘리먼트 내용 모델의 구성을 다음과 같이 제한시킨다.

Definition 1. 엘리먼트 내용 모델

엘리먼트 내용 모델 E := $(T_1 \dots T_k)^{<min,max>}$

요소(Term) Ti := $(s_{i1}^{opt} \dots s_{ij}^{opt})^{<min,max>}$ //기호의 연속

또는 $(s_{i1}^{opt} | \dots | s_{ij}^{opt})^{<min,max>}$ //기호의 선택

여기서, min = 0 or 1, max >= 1, opt = true 또는 false

우리의 엘리먼트 내용 모델은 min과 max를 이용하여 발생 빈도 정보를 유지하고 상호 연관성이 있는 기호들은 요소(Term)로 통합됨으로써 DTD나 XML Schema로의 변환이 가능하다. 특히, 요소의 경우 XML 스키마의 복합 타입으로 자연스럽게 변환될 수 있다. 이 모델은 max를 이용하여 최대 발생 빈도 정보를 유지하고, min을 이용하여 필수 발생 요소인지 또는 선택 발생 요소인지를 표시할 수 있도록 하였다. 그러나, 우리의 엘리먼트 내용 모델은 XML 스키마를 위한 상속, 다항성 및 이름 공간 (name space)에 관한 정보들은 포함하고 있지 않다. 왜냐하면 이러한 정보들을 유추해 내는 것은 매우 어렵거나 때에 따라서 불가능한 경우가 존재하기 때문이다. 기호의 opt 플래그는 요소 안의 기호가 선택 사항인지 아닌지를 표현한다. 즉 opt = false는 필수 기호임을 나타냄. 간결성을 위하여, opt가 false이면 이를 생략하고, min과 max가 둘 다 1이면 min, max를 생략한다.

예를 들어, 정규 표현식 (a b* c?)* 는 우리의 엘리먼트 내용 모델을 이용하여 $E=(T_1\ T_2\ T_3)^{<0,\infty>}$, 여기서 T_1 = (a), $T_2=(b)^{<0,\infty>}$, $T_3 = (c^{opt})$로 표현할 수 있다. 그러나, 우리의 내용 모델은 모든 종류의 정규 표현식을 나타낼 수는 없다. 예를 들어, (a (b|c+) d)* 와 같은 경로 표현식에 있어서 (b|c+)를 하나의 요소로 표현할 수 없으므로 우리의 내용 모델로는 표현할 수 없다. 그 대신, 좀 더 일반화 시켜, $((a)(b|c)^{<1,\infty>}$

(d))$^{<0,\infty>}$로 표현할 수 있다.

더욱이 간결성을 달성하기 위하여 우리는 엘리먼트 내용 모델에 중복 불가 타입 특성을 포함시켰다.

Property 1. 중복 불가 타입 특성

엘리먼트 내용 모델 E가 $(T_1...T_k)^{<min,max>}$라 하고 $\sum(T_x) = \{s_{xy}|s_{xy}$는 T_x안에 나타나는 기호이고 T_x는 E의 요소임$\}$ 라고 하자.

Disjoint Term: 만약 $i \neq j$ $(1 \leq i, j \leq k)$이면 $\sum(T_i) \cap \sum(T_j) = \varnothing$.

Disjoint Symbol: $\sum(T_i) = \{s_{i1},...,s_{in}\}$인 E안에 들어있는 각 요소 T_i에 대하여, 만약 $a \neq b$ $(1 \leq i, j \leq k)$이면 $s_{ia} \neq s_{ib}$.

Property 1을 쉽게 설명하자면, 동일한 기호는 엘리먼트 내용 모델에서 한번 이상 나타나서는 안 된다는 것이다. 이러한 중복 불가 타입 특성은 DTD와 XML Schema의 혼합 내용(mixed contents)의 타당성 제약 조건이지만, 실 세계에 존재하는 많은 DTD나 XML Schema는 Property 1을 대부분 만족한다 [16]. 또한 DTD나 XML Schema의 특별한 제약 조건은 1-unambiguity 제약 조건이다. 간단하게 이야기하면 1-unambiguity 제약 조건은 임의의 언어의 한 단어로의 유도 방법은 주어진 문법에서 오직 하나만 존재해야 한다는 것이다. 예를 들어 다음과 같은 하나의 정규 문법(regular grammar) , $s_1*s_2?s_1*$이 있다고 할 때, 정규 언어의 한 단어 s_1은 처음 s_1*에 의하여 유도되는지 마지막 s_1*에 의하여 유도되는지 알 수가 없게 된다. 따라서 $s_1*s_2?s_1*$은 1-unambiguity 제약 조건을 만족하지 못한다.

즉, XML 문서 (즉, 언어)를 위한 구조 정보(즉, 문법)는 1-unambiguity를 만족하여만 한다. 또한 주어진 문법이 1-unambiguity하다는 것은 대응되는 오토마타가 결정적(deterministic)하는 것과 동일하다[6]. 특성 1에 따라서 하나의 기호는 오직 한번만 엘리먼트 내용 모델에 나타남으로써 항상 결정적이다. 다시 말하면, 특성 1은 DTD나 XML Schema의 제약 조건인 1-unambiguity를 만족하는 충분 조건이 된다.

3.2 엘리먼트 구조 정보 추출

우리의 DTD 추출 알고리즘은 그림 2에서 보이는 바와 같이 중요한 몇 단계로 구성되어 있다. 우리는 알고리즘은 상향식(bottom-up) 접근 방법을 취한다. 각 엘리먼트 e의 자식 엘리먼트들의 배치 순서로부터 하나의 엘리먼트 내용 모델이 생성된다. 이렇게 생성된 엘리먼트 e에 대한 엘리먼트 내용 모델들은 통합되어 엘리먼트 e에 대한 내용 모델이 된다. 예를 들어, 그림 1에 있는 엘리먼트 book에 대한 내용 모델은 (title author$^{<1,2>}$)이고, 엘리먼트 author를 위한 내용 모델은 두 개의 엘리먼트 모델 (first last)와 (last first)을 통합 형태인 (first|last)<1,2>이다.

알고리즘	
1.	엘리먼트 e의 자식 엘리먼트 연속을 집합 I로 수집
2.	e를 위한 엘리먼트 내용 모델 EI 추출
2.1	I에 있는 각 연속을 분할
2.2	각 연속을 위한 엘리먼트 내용 모델을 추론
2.3	모든 엘리먼트 내용 모델들을 E_I 로 취합
3. E_I를 DTD 또는 XML Schema 형태로 변환	

(그림 2) DTD 추출 알고리즘

우리의 알고리즘은 입력 연속의 분할로부터 시작한다. 분할 단계에서, 입력 연속 N는 하위 연속 N_1, N_2, ..., N_K로 분해 된다. 하나의 하위 연속 N_i는 다음과 같은 특성을 따른다: N_i에 존재하는 하나의 기호는 연속적으로 나타나지 않는 이상 오직 한번만 나타난다. 또한 한 기호가 aaaa와 같이 연속으로 나타날 경우 N_i는 해당 기호로만 구성된다, 여기서 $\underline{N_i}$는 (a)$^{<1,4>}$와 같이 해당 기호의 반복 횟수를 유지한다.

예제 1. 주어진 연속 abcbcddefefddgggabcbc 에 대하여, 분할 결과는:

(abc), (bc), (d)$^{<1,2>}$, (ef),(ef),(d)$^{<1,2>}$,(g)$^{<1,3>}$, (abc), (bc)

여기서 어떤 기호가 $\hat{N}_i$상에 한번만 나타나는가를 조사하는데 드는 시간 복잡도는 O(d)이다. d는 N에 존재하는 구별되는 기호들의 수이다. 따라서, 분할 단계의 시간 복잡도는 O(dM)임을 알 수 있다. 여기서 M은 연속 N의 길이이다.

두 번째 단계는 분할 단계의 결과를 이용하여 하나의 입력 연속에 대한 엘리먼트 내용 모델을 추론하는 단계이다. 이 단계의 기본적인 규칙은 반복 최대 수를 이용하여 반복되는 부분을 가진 하위 연속들을 병합하는 것이다. 예제 1의 경우를 볼 경우, (abc) (bc)는 (a)(bc)$^{<1,2>}$로 병합할 수 있다.

우선, 하나의 엘리먼트 내용 모델 안의 요소(Term) T_i로 하위 연속의 반복되는 부분 N_a가 병합되는 경험적 규칙을 제시한다.

합병 규칙, fold(T_i, N_a)

T_i가 기호의 seq-term이라고 하는 기호의 연속, $(s^{opt}_{i1}, ..., s^{opt}_{ij})^{<n,x>}$ 또는 or-term이라고 불리는 기호의 선택, $(s^{opt}_{i1}|...|s^{opt}_{ij})^{<n,x>}$ 라고 하고, $N_a = (n_1...n_b)^{<1,r>}$이라고 가정하자.

F1. if $\sum(N_a) \not\subset \sum(T_i)$, fold($T_i$, N_a)는 정의되지 않음.

F2. T_i가 or-term 이라고 하자. fold(T_i,N_a)= $(s^{opt}_{i1}|...|s^{opt}_{ij})^{<n, x'>}$ 다. 여기서, N_a에 존재하는 하나의 기호는 최대 rb번 만큼 반복될 수 있으므로, 만약 E의 max값이 1이면, x′=x+rb이고 그렇지 않다면, x′ = max(r,rb) 이다.

F3. T_i가 seq-term이라고 하자. 만약 E의 max 값이 1이고 T_i의x 값이 1, 그리고 $(s^{opt}_{i1}, ..., s^{opt}_{ik}) \cap \sum(Na) = \varnothing$ 이면, fold(T_i, N_a) = $(s^{opt}_{i1}, ... , s^{opt}_{ik})$ fold($(s^{opt}_{ik+1}, ..., s^{opt}_{ij})^{<n,1>}$, N_a).

F4. T_i가 seq-term이라고 하자. If E의 max 값이 1 보다 크거나 또는 T_i의 x 값이 1보다 크거나 또는 $s_{i1} \in \sum(N_a)$이고 $\forall$ $n_m \in \sum(N_a)$ 에 대하여, $n_m = s_{ik}$ 일때 $n_{m+1} = s_{i1}$ (for $k < l$)이면, fold(T_i, N_a) = $(s^{opt|optional1}_{i1}, \cdots, s^{opt|optionalj}_{ij})^{<n, x'>}$, 여기서 만약 $s_{ip} \notin \sum(N_a)$ (for $1 <= p <= j$) 이면, $optional_p$= true, 그렇지 않으면

optional$_p$ = false, 그리고 만약 E의 max 값이 1이면 x' = x +r 이고 그렇지 않으면 x' = max(x,r) 이다.

F5. T_i가 seq-term이라고 하자. 만약 E의 max 값이 1 보다 크거나 또는 T_i의 x 값이 1 보다 크거나 또는 $s_{il} \in \sum(N_a)$이고 $n_m = s_{ik}$이고 $n_q = s_{il}$ ($k > l$이고 $m < q$) 인 $n_m, n_q \in \sum(Na)$ 존재할 때, $fold(T_i, N_a) = (s^{opt}_{il}|...|s^{opt}_{ij})^{<n, x'>}$, 여기서 만약 E의 max = 1이면 x' = jx+rb 그렇지 않다면 x' = max(jx, rb).

다음의 예제 2는 위의 합병 규칙에 대한 적용 예이다.

예제 2. E의 max 값은 1이라고 하자.

1) F2의 적용: $fold((a|b|c)^{<1,5>}, (bc)) = (a|b|c)^{<1,7>}$
2) F3의 적용: fold((abcd),(bc)) = (a)fold((bcd),(bc))
3) F4의 적용: $fold((abcd)^{<1,5>}, (bc)) = (aoptbcdopt)^{<1,6>}$
4) F5의 적용: $fold((ab)^{<1,2>},(ba))=(a|b)^{<1,6>}$

위의 합병 규칙을 적용한다고 하더라도 Property 1의 disjoin term 조건을 유지하기에는 어렵다. 예제 1을 고려해 보면, (abc),(bc),$(d)^{<1,2>}$,(ef)(ef) 는 하나의 엘리먼트 내용 모델 E= $(T_1\ T_2\ T_3\ T_4)^{<1,1>}$로 합병된다. 여기서 $T_1 = (a)^{<1,1>}$, $T_2 = (bc)^{<1,2>}$, $T_3 = (d)^{<1,2>}$, 그리고 $T_4 = (ef)^{<1,2>}$이다. 여기서, 다음 하위 연속 $(d)^{<1,2>}$를 E에 합병하려고 시도하려고 할 때, 기호 d는 T_3에 이미 존재함을 알 수 있다. 이를 처리 하기 위하여, $((a)^{<1,1>}\ (bc)^{<1,2>}\ (d|e|f)^{<1,8>})^{<1,1>}$과 같이 or-term을 사용하는 방법이 존재할 수 있으며, $((a)^{<0,1>}\ (bc)^{<0,2>}\ (d)^{<1,2>}\ (ef)^{<0,2>})^{<1,2>}$와 같이 optional을 이용할 수도 있다.

일반적으로 or-term에 존재하는 기호들은 서로 가까운 위치에 나타난다. 따라서, 이러한 지역성(locality)에 기반하여, 임계값 Threshold에 따라서 두 가지 대안 중에 한가지를 선택하도록 하였다. 다음에 제시되는 규칙은 엘리먼트 내용 모델 E에 하위 연속 N_a가 어떻게 합병 또는 삽입되는지를 나타낸다.

변환 규칙(Transformation Rules)

엘리먼트 내용 모델 E는 $(T_1 \cdots T_j \cdots T_k)^{<min,\ max>}$라고 가정하자. 여기서 하위 연속 N_a를 T_j에 삽입하거나 합병하려고 한다. N_a'은 N_a의 prefix로서 N_a'안에 있는 모든 기호들은 E의 T_i에 존재하거나, E에 나타나지 않는다 라고 하자.

R1. N_a'의 모든 기호가 E에 나타나지 않는다면, N_a'를 j+1 번째 위치에 T_{j+1}로써 삽입한다. 이 경우 만약 E의 max 값이 1 보다 크다면, N_a'은 이전 단계에서는 나타나지 않았던 것이므로, T_{j+1}의 min 값은 0으로 설정한다.

R2. 만약 $0 < j\ \ i <$ Threshold, $T_i \cdots T_j$는 새로운 하나의 요소 $T_i = (s_{il}|\cdots|s_{im})^{<n,x>}$, 여기서 $\cup^j_{l=i}\sum(T_l) = \sum$(new T_i), $n = \Pi^j_{l=I}$ (min of T_l), 그리고 $x = \sum^j_{l=1}(T_l$의 max 값 * T_l의 기호들의 갯수)

R3. 만약, j-i >= Threshold, $T_1,\cdots T_{i-1}$과 $T_{j+1},\cdots T_k$의 min 값들을 0으로 설정하고 E의 max 값을 1 증가 시킨다.

R4. 만약, j-i <= 0, $T_1,\cdots T_{i-1}$의 min 값들은 0으로 설정.

그리고, 규칙 R_2, R_3, R_4에 대하여서는 $fold(T_i, N_a')$이 적용된다. 그리고 N_a의 나머지 부분에 대하여 위의 변환 규칙이 다시 적용된다. 마지막으로, 더 이상의 하위 연속이 남아 있지 않다면, $T_{j+1},\cdots T_k$의 min 값들을 0으로 설정한다.

다음의 예제는 예제 1에 대하여 변환 규칙과, 합병 규칙이 어떻게 적용되는지를 보여준다.

예제 2. 하위 연속들이 (abc), (bc), $(d)^{<1,2>}$, (ef), (ef), $(d)^{<1,2>}$, $(g)^{<1,3>}$, (abc), (bc) 같이 주어지고, Threshold는 2라고 하자. 그러면, 엘리먼트 내용 모델 E는 다음과 같이 얻어진다.

삽입 (abc) : E = ((abc)) //By R1
삽입 (bc) : E = $((a)(bc)^{<1,2>})$ //R4, F3, F4
삽입 $(d)^{<1,2>}$: E = $((a)(bc)^{<1,2>}\ (d)^{<1,2>})$ //R1
삽입 (ef) : E = $((a)(bc)^{<1,2>}\ (d)^{<1,2>}\ (ef))$ //R1
삽입 (ef) : E = $((a)(bc)^{<1,2>}\ (d)^{<1,2>}\ (ef)^{<1,2>})$ //R4, F4
삽입 $(d)^{<1,2>}$: E = $((a)(bc)^{<1,2>}\ (d|e|f)^{<1,8>})$ //R2. F2
삽입 $(g)^{<1,3>}$: E = $((a)(bc)^{<1,2>}\ (d|e|f)^{<1,8>}\ (g)^{<1,3>})$ //R1
삽입 (abc) : E = $((a)(bc)^{<1,2>}\ (d|e|f)^{<1,8>}\ (g)^{<1,3>})^{<1,2>}$ //R3, F3, R5, F4
삽입 (bc) : E = $((a)(bc)^{<1,2>}\ (d|e|f)^{<1,8>}\ (g)^{<1,3>})^{<1,2>}$ //R4, F4
마지막으로, E = $((a)(bc)^{<1,2>}\ (d|e|f)^{<0,8>}\ (g)^{<0,3>})^{<1,2>}$
E의 DTD 모양은 ((a)(bc)+(d|e|f)*(g)*)+
또한 E의 XML Schema는 다음과 같다.

```
<xsd:complexType ···>
    <xsd:sequence maxOccurs ="2">
        <xsd:element name ="a">
        <xsd:group ref = "T1" maxOccurs="2">
        <xsd:group ref = "T2" minOccurs ="0" maxOccurs="8">
        <xsd:element name = "g" minOccurs="0" maxOccurs ="3">
    </xsd:sequence>
</xsd:complexType>
<xsd:group id ="T1">
    <xsd:sequence>
      <xsd:element name ="b">
      <xsd:element name ="c">
    </xsd:sequence>
</xsd:group>
<xsd:group id ="T2">
    <xsd:choice>
        <xsd:element name ="d">
        <xsd:element name ="e">
        <xsd:element name ="f">
    </xsd:sequence>
</xsd:group>
```

마지막으로 모든 엘리먼트 내용 모델들을 모아서 공통 요소 추출 및 or를 이용하여 최종 엘리먼트 내용 모델을 만든다. 직관적으로, 우리는 공통 prefix나 suffix term 들을 뽑아내고 나머지는 or를 이용하여 통합하는 형태를 취한다. 예를 들어, 두 엘리먼트 내용 모델 $((a)^{<n1,\ x1>}\ (b)^{<n2,x2>}\ (d)^{<n3,x3>})^{<n4,x4>}$와

((a)$^{<n5,\ x5>}$ (c)$^{<n6,x6>}$ (d)$^{<n7,x7>}$) $^{<n8,x8>}$에 대하여, 병합 단계에서 ((a)$^{<MIN(n1,n5),MAX(x1,x5>}$ (b|c)$^{<MIN(n2,n6),MAX(x2,x6)>}$ (d)$^{<MIN(n3,n7),MAX(x3,x7)>}$) $^{<MIN(n4,n8),MAX(x4,n8)>}$이 생성된다.

3.3 데이터 타입 추출

XML의 구조 정보 표현 언어인 DTD의 경우 엘리먼트의 데이터 값의 타입은 #PCDATA로 한정됨으로 데이터 타입을 추출할 필요가 없다. 그러나 XML Schema의 경우에는 다양한 데이터 타입을 지원하고 있다. 따라서, 본 절에서는 추출될 구조 정보를 XML Schema로 표현할 때 사용할 수 있는 데이터 타입 추출 방안에 대하여 설명한다.

그림 3은 데이터 타입 추론을 위한 알고리즘을 나타낸다.

데이터 타입 추론기는 각 태그에 대한 데이터 값들의 타입을 귀납적으로 추론하여 추론된 타입 결과를 inferredType에 저장한다.

엘리먼트 e의 데이터 값들의 집합을 D라고 할 때, 데이터 타입 추론기는 D에 속해 있는 원소들을 하나씩 읽어 들여 Infor_type()함수를 이용하여 단일 데이터 값을 타입을 추론한다. Infor_type() 함수는 입력 받은 데이터 값 d가 '0'~'9'로 구성되어 있고 맨 처음 0이 아닐 경우 정수 타입 (integer)으로 추론한다. 또한, 모든 문자가 '0'~'9'와 '.'이고, '.'은 한 번만 나타나며 첫 번째와 두 번째 문자가 각각 '0'과 '.' (예, 0.xxx), 또는 첫 번째 문자가 0이 아닐 경우 (예, xxx.xxx), 이를 소수 타입 (*float*)으로 추론한다. 이외의 경우에는 모두 텍스트 (string)로 추론한다.

현재까지 추론된 데이터 타입 (inferredType)이 *undefined* 이거나 정수(integer)라면 (라인 7-12), 데이터 값의 타입 (type)이 *integer*라면 (라인 9) inferredType은 *integer*가 되고, type이 *float*이라면 (라인 10) inferredType은 *float*이 된다. 이는 실수 타입을 가지고도 정수를 표현 할 수 있기 때문이다. 반면 type이 *string*이면 (라인 11) 현재까지 추론된 타입이 비록 *integer*일지라도 *string*로 바뀐다. 이것은, 정수를 사용하여 텍스트를 표현할 수는 없지만 텍스트를 사용하여 정수를 표현할 수는 있기 때문이다.

```
Procedulre TypeInference(DataValueSet D)
 1. begin
 2. inferredType := undefined
 3. for( i = 0; i < |D|; i++){
 4.       get a DataValue d from D
 5.       type := Infer_type(d)
 6.       switch(inferredType){
 7.             case undefined:
 8.             case integer:
 9.                   if(type = integer) InferredType := integer
10.                   else if(type = float) InferredType := float
11.                   else InferredType := string
12.              break
13.             case float:
14.                   if(type = string) InferredType := string
15.              break
16.             case string:
17.                   //do nothing
18.              break
19.       }
20. }
21. end
```

(그림 3) 데이터 타입 추론 알고리즘

현재까지 추론된 타입 결과 (inferredType)가 *float*일 경우에는 데이터 값의 타입이 *string*일 경우에만 InferredType 값을 *string*으로 수정한다 (라인 13-15). 또한 현재까지 추론된 타입 결과가 *string*일 경우에는 추론된 타입 결과를 변동할 필요가 없다 (라인 16-18).

그림 3에서 보는 바와 같이 본 연구에서 사용된 타입 추론기는 3가지의 타입 (integer, float, string)만을 추론해 낸다. 이외에도 boolean, Date와 같은 타입을 추론할 수 있도록 확장이 가능하나 이러한 확장은 구조 정보 추출 전체 성능을 저하시킬 수 있음으로 위의 3가지 타입으로만 한정하였다.

4. 실 험

효율성과 정확성을 보이기 위하여, 우리는 본 논문에서 제안하는 방법과 XTRACT, DDbE를 비교하였다. DTD-Miner는 세부 동작 원리가 자세하게 나와있지 않아서 본 실험에서는 제외하였다. XTRACT는 원래 C++로 구현되어 있으며 상업적 용도로 만들어져 있어서 소스 코드가 공개되어 있지 않다. 따라서, 효율성에 있어서 공정성을 기하기 위하여 Java를 이용하여 XTRACT를 구현하였다. Java를 이용한 XTRACT 구현에 있어서, Facility Location Problem (FLP) 근사 방법 대신에 욕심쟁이 경험론 (Greedy heuristic)을 이용하여 XTRACT의 MDL 하위 시스템을 구현하였다. 따라서, 우리가 구현한 XTRACT를 원칙 XTRACT 알고리즘의 자체 구현 보다 적은 시간을 소모한다.

위에서 언급한 바와 같이 본 논문에서 제안하는 방법의 구조 정보 결과는 엘리먼트 내용 모델은 XML 스키마를 위한 상속, 다항성 및 이름 공간 (name space)에 관한 정보들은 포함하고 있지 않은 간단한 형태의 모델로서 본 논문에서 제안한 엘리먼트 내용 모델을 그룹과 복합 타입을 이용하여 간단하게 XML Schema로 변환이 가능하며 이 부담은 수십 밀리초에서 수백 밀리초에 불과하다. 또한, 비교 대상 기법인 XTRACT와 DDbE 는 DTD 추출을 위한 툴로서 XML Schema형태로 결과가 생성되지 않는다. 또한 생성된 DTD로 부터 XML Schema로의 변환 비용은 위의 부담과 유사하거나 더 클 것으로 예상된다. 따라서 본 실험에서는 DTD 추출에 대한 성능만을 다룬다.

데이터 세트로서 XTRACT에서 사용된 실제 DTD를 사용하였다. 이 DTD는 미국 신문 연합 (Newspaper Association of America)의 분류화 광고 표준 (Classified Advertising Standard)로부터 얻어진 것이다. 원본 DTD는 표 1 두 번째 열에 있다.

DTD의 정확도를 측정하기 위하여, IBM의 XML Generator를 이용하여 각 DTD마다 1000개의 element를 생성하였다. 이 실험은 Intel Pentium IV-1.0G, 512 Mbyte 기계에 Windows XP 운영체제에서 실행하였다.

〈표 1〉 생성된 DTD

No.	Original DTD	Our Approach	XTRACT	DDbE ver2
1	a|b|c|d|e	(a|b|c|d|e)	a|b|c|d|e	(a|b|c|d|e)
2	(a|b|c|d|e)*	(a|b|c|d|e)*	(a|b|c|d|e)*	((a|(e|b|c|a|d|(d+c))|(e|a|(e|c|d|(e+b+))|c|b|(e+b+)|d|(d+c))|d|e|b|c|(e+b+)|(dcb)|(d+c))*)
3	ab*c*	(ab*c*)	(ab*c*)	(a(b|c)+)
4	a*b?c?d?	(a*b?c?d?)	a*b?c?d?	(((a|b)|c|d|a|((b|c)|d|b|c)|b)+)
5	(a(bc)+d)*	(a(bc)+d)*	(a(bc)*d)*	-
6	(ab?c*d?)*	(a(b|c|d)*)*	-	((((ac+ac+d)|(a+b)|a)| ... |(c+dab))*)

〈표 2〉 DTD 추출 시간 (단위: sec)

No.	Our Approach	Simple XTRACT	DDbE ver2
1	0.4	25.8	228
2	0.65	25.6	229
3	1.27	25.0	230
4	0.5	24.1	231
5	2.33	131	-
6	1.89	-	267

표 1에서는 1000개의 element들로 구성된 각 데이터 세트로부터 얻어진 DTD를 보인다. 표 1에 나타난 XTRACT의 결과 DTD는 [9]에서 얻었으며, 다른 DTD들은 본 실험을 통하여 얻었다. 우리 실험에서는 DDbE version 2.0을 사용해서 XTRACT논문에서 나타난 DDbE의 결과와는 다르다.

표 1에서 보이는 바와 같이 본 논문에서 제안하는 기법은 다른 기법들과 비교하여 동일하거나 보다 낳은 DTD를 추출함을 알 수 있다. DDbE는 보통 너무 복잡한 DTD 결과를 생성하고 다섯 번째 DTD 에서는 그 결과를 생성하지 못했다. 또한, XTRACT 논문에서는 여섯 번째 DTD의 결과가 보고되지 않았다.

표 2에서는 각 접근 방법 대한 DTD 생성 시간을 보여준다. 우리의 접근 방법은 수 초안에 DTD를 생성하며, 가장 좋은 성능을 보여준다. 이에 반하여, XTRACT는 우리의 접근 방법에 비하여 적어도 20배 느리며, *DDbE*는 우리의 방법보다 200배 이상 느리다. 위에서 보인 바와 같이 우리의 접근 방법은 다른 접근 방법에 비하여 대단히 뛰어난 성능을 보인다. 더욱이, 우리의 접근 방법은 가장 정확한 결과를 생성한다.

5. 결 론

XML 데이터의 구조 정보는 효율적인 저장 및 검색 등에서 사용할 수 있는 중요한 정보이다. 그러나, 구조 정보의 중요성에 반하여, 많은 XML 문서는 자신의 구조 정보를 가지고 있지 않다. 따라서, XML문서로부터 효율적인 구조 정보 추출을 위한 많은 접근 방법들이 제안되었다. 기존의 접근 방법들은 구조 정보를 표현하기 위한 내용 모델에 대한 제약을 두지 않았다. 따라서 탐색 공간의 증가로 인한 성능 저하가 발생하며 DDbE와 같은 접근 방법들의 생성 결과의 질이 매우 나쁘다. 따라서 본 논문에서는 임의의 엘리먼트의 자식 엘리먼트들의 연속에 대한 구조 정보를 간결하면서도 정확한 정규 표현식을 효율적으로 유출하는 기법을 제안하였다. 간결성을 위하여, 우리는 DTD와 XML Schema의 혼합 내용 모델의 제약 조건을 이용한 엘리먼트 내용 모델을 고안하였으며, 효율성을 위하여 합병 규칙과 변환 규칙이라고 불리는 경험론적 규칙들을 제안하였다.

간단히 설명하면, 합병규칙은 인접되어 발생되는 유사 하위 시퀀드들을 합병하는 규칙이며 변환 규칙은 멀리 떨어져 있는 유사 하위 연속들에 대하여 지역성을 고려하여 합병하는 규칙이다. 본 논문에서 제안하는 기법에 대한 효율성을 보이기 위하여 실제의 DTD를 이용한 실험을 수행하였다. 이 실험 결과, 본 논문에서 제안하는 방법이 가장 정확한 결과를 생성함을 보였으며 비교 대상인 XTRACT 보다는 최대20배 정도 빠르며 DDbE보다는 200 배 정도 빠름을 보였다.

참 고 문 헌

[1] D. Angluin, "Equivenance queries and approximate fingerprints," In Proceedings of the workshop on computational Learning Theory, 1989.

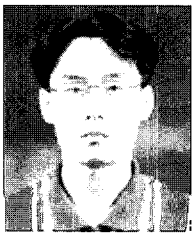
[2] L. Berman and A. Diaz, Data Descriptors by Example (DDbE), IBM alphaworks, http://www.alphaworks. ibm.com/tech/DDbE, 2001.

[3] T. Bray, C. Frankston, and A. Malhatro, "Document Content Description for XML," W3C submission, http://www.w3.org/TR/NOTE-dcd, 1998.

[4] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau eds., Extensible Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation, http: //www.w3.org/TR/REC-xml, 2006.

[5] A. Brazma, "Efficient Identification of Regular Expressions from Representative Examples," In Proceedings of ACM COLT, 1993.

[6] A. Bruggemann Klein, D. Wood, "One-unambiguous regular grammar," Inf. Comput., 142(2), pp.182-206, 1998.
[7] M. Bryan, "An Introduction to the Standard Generalized Markup Language (SGML)," http://www.personal.u-net.com/~sgml/sgml.html
[8] D. C. Fallside, P. Walmsley, XML Schema Part 0, W3C Recommendation, http://www.w3.org/TR/xmlschema-0, 2004.
[9] M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim, "XTRACT: A System for Extracting Document Type Descriptors from XML Documents," In Proceedings of ACM SIGMOD, 2000.
[10] R. Goldman, J. Widom, "DataGuides: Enable Query Formulation and Optimization in Semistructured Databases," In Proceedings of VLDB Conf., 1997.
[11] J. Hegewald, F. Naumann, M. Weis, " XStruct: Efficient Schema Extraction from Multiple and Large XML Document," In Proceedings of International Conference of DataEngineering Workshop (ICDEW), 2006.
[12] Juliana Freire, Jayant R. Haritsa, Maya Ramanath, Prasan Roy, Jérôme Siméon, "StatiX: making XML count," In Proceedings of ACM SIGMOD, 2002.
[13] C. H. Moh, E. P. Lim, and W. K. Ng, "DTD Miner: A Tool for Mining DTD from XML Documents," In Proceedings of International Workshop on Advance Issues of E Commerce and Web Based Information Systems(WECWIS), 2000.
[14] S. Nestorov, J. Ullman, J.Wiener, and S. Chawathe, "Representative Objects: Concise Prepresentation of Semistructured, Hierarchical Data," In Proceedings of IEEE ICDE, pp.79-90, 1997.
[15] J. Rissanen, "Modeling by shortest data description," Automatica, Vol. 14, 1978.
[16] Robin Cover. The XML Cover Pages. http://www.oasis open.org/cover/xml.html, 2001.
[17] J. Shanmugasundaram, K. Tufte, C. Zhang, H. Gang, D. J. DeWitt, and J. F. Naughton, "Relational databases for querying XML documents: Limitations and opportunities," In Proceedings of VLDB Conf., 1999.
[18] C. S. Wallace, D. M. Boulton, "An Information Measure for Classification," Computer Journal, Vol. 11, 1968.
[19] Q.Y. Wang, J. X. Yu, and K. -F. Wong, "Approximate graph schema extraction for semi structured data," In Proceedings of the International Conference on Extending Data Technology (EDBT), 2000.
[20] R. K. Wong, J. Sankey, "On Structural Inference for XML Data," Technical Report UNSW-CSE-TR-0313, The University of New South Wales.



민 준 기

e-mail : jkmin@kut.ac.kr
1995년 숭실대학교 전자계산학과(공학사)
1997년 한국과학기술원 전자전산학과 (공학석사)
2002년 한국과학기술원 전자전산학과 (공학박사)
2002년~2003년 한국과학기술원 연수연구원(Post Doc.)
2003년~2004년 한국과학기술원 초빙교수
2004년~2005년 전자통신연구원 선임연구원
2005년~현 재 한국기술교육대학교 인터넷미디어공학부 조교수
관심분야 : Query Processing, XML, Stream Data, Sensor Network