XML 문서 타입의 분류
(Taxonomy of XML Document Types)

이 정 원 † 박 승 수 †
(Jung-Won Lee) (Seung-Soo Park)

요 약 XML이 가지는 표현의 유연성은 정보검색, 문서관리, 데이타 마이닝 등의 분야에 커다란 영향을 미치고 있다. 그러나 새로운 XML 관련 기술들이 XML 문서의 특성을 체계적으로 파악하지 않고 일부 특성만을 다룰 수 있도록 개발되어 다른 타입의 XML 문서를 적용하면 성능 저하를 가져오기도 한다. 따라서 우수한 결과를 내는 방법론이라 하더라도 그 결과에 대한 신빙성을 뒷받침할 만한 척도가 미비한 실정이며 개발된 다양한 방법론을 응용 분야나 문서집합의 특성에 따라 적절하게 선택할 수 있는 기준도 모호하다. 본 논문에서는 XML이 가지는 특성을 토대로 크게 요소(element), 구조, 그리고 데이타를 중심으로 분류한 XML 문서 타입의 분류 체계(taxonomy)를 제안하고자 한다. 체계적인 XML 문서 타입의 분류 체계는 새로 개발된 XML 기술들이 어떠한 XML 문서 타입을 처리 할 수 있는지를 평가할 수 있게 함으로써 결과의 신빙성을 증진시킬 수 있다. 또한 대상 XML 문서를 분류 체계에 의거하여 처리 가능한 타입들을 제시함으로써 응용 분야에 적합한 XML 기술을 선택할 수 있는 기초를 제공한다. 제안된 분류 체계를 토대로 XML 문서 마이닝에 적용해 본 결과 전처리(preparations) 과정에서 고려할 수 있는 XML 문서 타입을 밝혀내고 실험 선정 대상 문서의 타입을 제시할 수 있었다.

키워드 : XML, 분류 체계, 구조 발견, 오토마타

Abstract oping and applying XML techniques. One key aspect of our taxonomy is the support of the credibility of the result by evaluating which XML document types can be processed by a method. Another key aspect is to provide a basis for determining which is the best for target XML document types. Application with preparations for XML document mining shows that our taxonomy may present XML document types to be able to consider during the preparation process and target XML document types to be used for experiments.

Key words : XML, taxonomy, structure discovery, automata

1. 서 론

XML은 정보표현의 유연성으로 웹상에서 데이타 표현과 교환을 위한 주요 표준으로 널리 사용되고 있다. XML은 임의의 태그를 사용하여 요소(element)를 표현하고 그 요소들은 계층 구조를 갖도록 구성된다[1]. 이러한 특징은 정보 검색, 문서관리, 데이타 마이닝 등의 분야에 여러 가능성 및 도전의 기회를 제공하고 있다.

대부분의 XML 관련 연구(XML 저장, 질의, 검색, 분류 등)에서 분석 및 실험 대상으로 사용하고 있는 XML 문서는 문서의 길이에 상관없이 단순 구조로부터 수십 개의 요소들이 복잡하게 내포되어 있는 구조에 이르기까지 다양하다. XML 구조 발견, 질의 처리 및 최적화, 문서 마이닝 등의 대부분의 연구에서는 XML 문서를 트리 혹은 그래프로 표현하고 있다. 사용된 XML 문서는 사이클을 가질 수도 있고, 요소가 공유되어 표현되기도 하며 어떤 구조는 다소 플랫 하기도 하고 또 복잡하게 얽혀있기도 하다. 이렇게 XML이 요소의 내포관계에 의해 구조적으로 트리 혹은 그래프로 표현되면서 XML은 구조적으로 다양한 특성들을 가질 수 있다. 현재 대부분의 XML 관련 연구들이 연구 목적에 따라 일부 타입의 문서에만 초점을 두고 있으며 실험 또한 문서의 구조적인 복잡도와는 상관 없이 문서의 크기 혹은 데이타의 크기에 따른 성능 평가를 하고 있어 어떠한 XML 타입을 고려할 수 있는지에 대한 분석은 미흡하다. 이러한 XML 문서의 타입에 대한 고찰이 제대로 이루어지지 않으면 다음과 같은 문제가 발생한다.

첫째, 새로운 기법의 결과에 대한 신빙성 문제이다. 예를 들어 XML 문서에서 스키마를 추출하여 데이타베

† 정 회 원 : 이화여자대학교 컴퓨터학과 교수
jungwony@ewha.ac.kr
sspark@mm.ewha.ac.k
논문접수 : 2003년 8월 1일
심사완료 : 2004년 11월 17일

이스에 저장하는 방법이 있다고 하자. 저장 데이타를 질의하여 결과를 얻는 시간을 토대로 성능평가를 하지만 과연 그 방법이 문서의 구조를 얼마나 다양하게 고려하였는가에 대한 평가 없이 성능 평가 결과를 완전히 믿을 수 없는 것이다. 또 다른 예로 XML 문서 마이닝을 위한 새로운 방법을 가지고 높은 정확도(accuracy)를 얻었지만 과연 XML의 특성들을 얼마나 고려한 것인지, 다른 타입의 문서를 적용하면 형편없이 떨어지는 건 아닌지 의심할 수 있다.

둘째, XML을 위해 개발된 다양한 방법론 적용시 문서의 타입에 따라 어떤 방법을 선택하는 것이 좋을 지에 따른 선택 기준이 모호하다. XML의 유연한 표현 방식으로 인하여 한 가지 기법이 반드시 모든 XML 문서의 타입에서 대해 좋은 성능을 보이는 것이 아니다. 주로 플랫한 구조를 가지는 문서에서 질의의 경로 표현(path expression)을 처리하는데 우수한 성능을 보이는 방법론을 선택한다든지, XML 데이타의 순서 정보를 요구하지 않는 시스템에서 데이타의 상하위 관계는 물론 형제 노드(sibling)의 순서정보까지 유지하는 스키마 추출 방법을 선택하는 오류를 범할 수도 있다.

셋째, 적절한 실험대상을 선정할 수 있는 기준이 모호하다. XML의 새로운 기법을 연구하고도 실험대상을 적절하게 선택하지 못하여 어떤 이유로 성능이 저하되고 있는지 파악하기 어렵다. 또한 다양한 실험을 한 것인지에 대한 평가도 어렵다.

따라서 XML 문서가 가지는 타입을 구조적으로, 의미적으로 파악할 수 있도록 체계적인 분류를 구축하는 작업이 요구된다. XML 문서 타입의 체계적인 분류는 다음과 같은 것을 가능하게 한다.

첫째, 새로운 XML 방법론이 고려할 수 있는 XML 문서 타입을 제시할 수 있다. 전체 타입들 중 방법론에 반영된 타입의 수를 기준으로 방법론의 처리 능력(coverage)을 계산한다면 같은 성능을 보이는 두 방법론이 있을 때 처리 능력이 높은 것이 더 우수한 방법이라 말할 수 있다.

둘째, 대상 XML 문서의 집합의 특성을 분류 체계에 의거하여 분석한 뒤, 그 문서집합의 타입과 추구하는 목적이 일치하는 방법론을 선택할 수 있다. 따라서 방법론이 적용될 시스템의 성능 향상을 도모할 수 있다.

셋째, 다양한 타입의 XML 문서를 기반으로 실험하고 문제점을 파악하여 성능을 개선시킬 수 있다.

다시 말해 체계적인 XML 문서 타입의 분류 체계는 XML 관련 연구의 기초를 제공하고 응용 분야에 따른 선택적인 적용, 적용시 결과에 대한 평가와 더불어 방법론의 XML 특성에 대한 처리 능력을 평가함으로써 결과의 신빙성을 증진시킬 수 있다. 따라서 본 논문은 XML의 의미와 구조의 차이에서 비롯된 다양한 XML 문서의 타입에 따른 분류 체계를 구축하고자 한다. 본 연구 결과는 다음과 같은 분야에서 응용될 수 있다.

XML 문서를 위한 저장 기법 개발 시, XML 문서는 그 표현에 있어 많은 유연성을 가지고 있으므로 저장 기법에 따라 그 문서가 가지는 여러 가지 타입 중에 고려할 수 있는 부분과 고려하기 어려운 부분이 발생한다. 이때 제안된 분류 체계는 저장 기법에 따라 고려할 수 있는 타입이 무엇이며, 또 그 요소가 얼마나 중요한지를 판가름 할 수 있는 척도를 제공할 수 있기 때문에 저장 기법의 성능의 우수성을 미리 예견해 볼 수 있다. 또한 문서의 타입에 따라 어떠한 저장 기법을 선택할 지를 결정할 수 있게 한다. 한편, XML 문서를 대상으로 하는 질의 언어에 대해 분류 체계를 토대로 지원되는 질의의 종류가 얼마나 되는지, 다양한 문서의 구조에 대해 필요한 질의가 유연하게 지원되는지를 측정해 볼 수 있다. 또한 문서 타입에 따라 적합한 질의 언어를 선택하여 질의 할 수 있도록 하는 기초를 제공할 수 있다. 이외에도 XML 문서 마이닝의 문서 분석 과정에 적용함으로써 마이닝 결과에 대한 평가만이 아닌 분석 방법이 어떠한 XML의 타입들을 고려할 수 있는지를 평가할 수 있게 해준다. 이는 마이닝 결과가 정확도로 제시되는 것과 동시에 얼마나 XML의 의미(semantics)를 반영하였는가에 대한 방법론의 타당성을 뒷받침할 수 있는 근거가 될 수 있다.

논문의 구성은 2장에서는 관련 연구를, 3장에서는 XML 문서를 오토마타에 기반을 두고 형식화 한 뒤, XML의 특성을 일관적으로 추출 및 분류한다. 이러한 분류 체계는 단일 문서의 특성/다중 문서간의 관계를 중심으로 XML 문서가 가질 수 있는 타입을 구조, 요소, 그리고 데이타(PCDATA type)로 나누어 분류한다. 이어 4장에서는 44개의 세부 항목들을 분석하고 각 경우의 XML 사례를 든다. 5장에서는 분류 체계를 XML 문서 마이닝을 위한 전처리 과정에 적용하여 이 방법론이 어떠한 XML의 특성들을 반영하는지 평가하고 실험 대상 문서의 타입을 제시한다. 마지막으로 6장에서는 결론 및 향후 연구를 기술한다.

2. 관련 연구

기존의 XML 관련 연구 중 특성을 비교, 분류하기 위한 연구는 주로 XML 질의 언어와 XML 스키마 언어 분류[2-5]에 국한된다. 이러한 연구들은 많은 XML 언어들간의 공통점과 차이점에 초점을 두고 언어의 분류 체계를 제시하고 동시에 표준 안으로서 가장 강력하고 효과적으로 만드는 특징들을 규정 짓고 있다. 그러나 XML 문서 자체에 대한 분석은 이루어지고 있지 않다.

DTD(Document Type Definition)나 XML-Schema는 그 이름에서도 알 수 있듯이 XML 문서 구조를 요약한 것이기는 하나 균형적(balanced)인 구조인지, 공유 노드(shared node) 존재 등 실질적인 문서의 특징을 알아내기 위해서는 또 다시 분석과정을 거쳐야 한다. 더욱이 요소의 의미적인 특징을 표현할 수 있는 방법이 없다. 또한 다중 문서간의 관계를 파악하기 위해서는 단순한 XML 언어 분류 차원이 아닌 XML 문서 자체에 대한 분석을 기반으로 한 새로운 분류 체계가 필요하다.

반구조적인 문서를 형식화 하기 위해 오토마타를 기반으로 하거나[6,7] first-order logic 혹은 규칙 기반 언어인 Datalog[8]를 이용하기도 한다. 어떠한 방법을 사용하더라도 근본적으로는 XML 모델은 모두 트리 혹은 그래프의 모습을 갖는다. 트리나 그래프는 이진 트리, B+ 트리, Spanning 트리 등 이미 정형화된 타입들이 존재[9]하나 XML 문서의 구조적 특징상 트리의 가짓수를 2개만으로 가정할 수도, 항상 하나의 노드가 균일한 수의 가지를 가지게 설계할 수도 없다. 또한 이미 트리 포함 문제[10], 공통 트리 구조를 찾는 문제[11,12]를 통해 트리의 구조적인 특성을 확장할 수 도 있으나 이러한 연구는 XML, 적어도 반구조적인 문서를 염두에 두고 개발된 방법이 아니기 때문에 노드와 에지가 모두 레이블을 가지는 XML의 구조적인 특성을 반영할 수 없다. 한편 [13]에서 XML의 스키마 충돌 문제를 다루고 있다. 그러나 이는 XML 문서 자체가 아닌 XML 스키마의 형이나 이름 충돌 문제를 국소적으로 다루고 있다.

3. XML 문서 타입의 분류 체계 구축을 위한 사전 작업

본 논문에서 제시하는 분류 체계는 XML 요소, 계층적 구조, 그리고 데이타를 중심으로 한 XML의 특성에만 제한을 둔다. 먼저 분류 체계를 구축하기 이전에 XML을 위한 형식 모델과 XML 문서 타입을 도출해 내는데 기초가 되는 범주(category)를 정의 할 필요가 있다.

3.1 XML 문서의 형식 모델

XML의 구조는 그 특성을 비교, 추출하기 위하여 NFA(Nondeterministic Finite Automata)를 이용[14]하여 다음 정의 1과 같이 NFA-XML로 형식화한다. 그리고 요소는 속성이나 타입과 같은 부가적인 정보를 제거한 순수 요소 정보만을 가정한다.

정의 1. NFA-XML M = (Q, D, Σ_N, Σ_T, δ, q_0, F)

Q : 유한 상태 집합. 요소가 컨텐트를 유도하지 않으면 새로운 상태가 생성된다.

D : 유한 데이타 집합(컨텐트)

Σ_N : 컨텐트를 유도하지 않고 새로운 상태를 생성시키는 유한 요소 집합.

Σ_T : 직접 컨텐트 D를 유도하는 유한 요소 집합.

δ : 전이 함수. 만약 어느 요소 $a \in \Sigma_N$이면 $\delta(q,a) = p_1, p_2, ..., p_n$(p는 새로운 상태) 이고 $a \in \Sigma_T$이면 $\delta(q,a) = D_1, D_2, ..., D_n$이다.

q_0 : 시작 상태. 루트 요소는 시작 상태로의 전이를 일으킨다.

F : 종결 상태. 컨텐트를 유도하는 요소 ($\in \Sigma_T$) 만이 종결상태로의 전이를 일으킨다.

즉, 각 요소는 유한 오토마타의 에지 레이블이 되며 전이함수 δ의 입력 스트링 a의 역할을 맡는다. 각 상태는 요소에 의해서 발생되는 전이의 시작 및 끝점을 의미한다. 그리고 실제 데이타를 유도하는 단말 요소(Σ_T)는 모두 종결 상태에 도달한다고 가정한다. 이로써 XML 문서를 루트 요소로부터 종결 상태인 데이타 집합으로의 유한 오토마타를 만들 수 있다. 예를 들어 다음 그림 1(a)의 예제 XML 문서는 (b)의 NFA-XML로 형식화 될 수 있다. 그리고 NFA-to-DFA(Deterministic Finite Automata) 변경 알고리즘과 상태 최소화 알고리즘[14]을 이용하여 그림 1(c)와 같이 중복된 요소(element와 구조정보)를 모두 최소화한 DFAXML을 얻을 수 있다. 앞으로 XML의 추출 특성 및 분류 기준은 모두 DFA-XML을 기반으로 비교, 분석될 것이다.

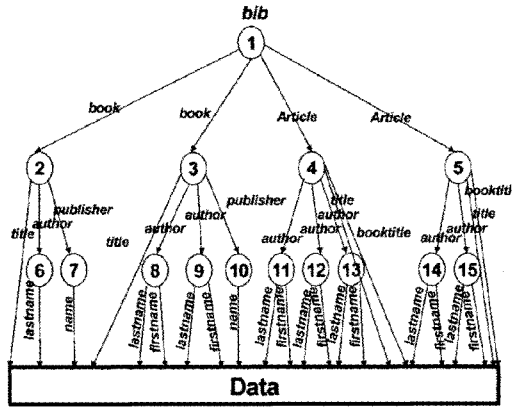
3.2 분류 체계를 위한 범주

분류 체계를 위한 범주는 각 단계별로 완전성(completeness)을 보장해야 한다. XML을 위한 분류체계는 다음 그림 2와 같은 8개의 기준을 중심으로 3개의 하위 계층을 갖는다.

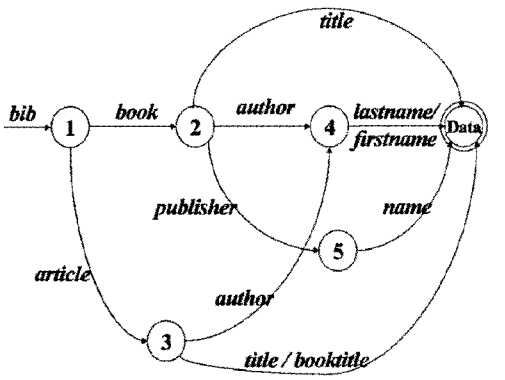
- **기준 1. XML 문서의 수(single/multiple)** : 먼저, XML 문서는 가장 개괄적으로 하나의 XML 문서 내에서 발생하는 타입인가 아니면 다중 문서간의 관계에 의해 발생하는 타입인가에 따라 나눌 수 있다. 어떤 XML 응용은 단일 문서의 특징을 다루는 데 중점을 둘 수 있고 또 다른 응용에서는 서로 다른 문서들간의 관계에서 발생하는 특징에 더 초점을 둘 수 있다. 여기에서 'multiple' 분류는 먼저 각각의 문서가 'single'의 분류에 따른 타입들로 분석되고 그 결과를 토대로 다중 문서간의 구조 혹은 요소간의 관계를 분류하는 것을 의미한다. 따라서 'multiple'의 분류는 'single'의 모든 타입을 고려한 후 그 관계에 따라 나뉜다.
- **기준 2. 단일 문서내의 구성 요소(element/structure/data)** : 하나의 XML 문서는 태그로 표현되는 부분과 데이타, 내포된 구조로 분석될 수 있다. 본 논

```
<bib>
 <book year="1995">
   <title>An Introduction to Database Systems</title>
   <author><lastname>Date</lastname></author>
   <publisher><name>Addison-Wesley</name></publisher>
 </book>
 <book year="1999">
   <title>Data on the Web: from Relations to Semistructured Data &amp;amp; XML</title>
   <author><firstname>Serge</firstname><lastname>Abiteboul</lastname></author>
   <author><firstname>Peter</firstname><lastname>Buneman</lastname></author>
   <author><firstname>Dan</firstname><lastname>Suciu</lastname></author>
   <publisher><name>Morgan-Kaufman</name></publisher>
 </book>
 <article year="1999" type="inproceedings" month="June">
   <author><firstname>Mary</firstname><lastname>Fernandez</lastname></author>
   <author><firstname>Alin</firstname><lastname>Deutsch</lastname></author>
   <author><firstname>Dan</firstname><lastname>Suciu</lastname></author>
   <title>Storing Semi-structured Data Using STORED</title>
   <booktitle>ACM SIGMOD</booktitle>
 </article>
 <article year="1995" type="inproceedings" month="Jan">
   <author><firstname>Norman</firstname><lastname>Ramsey</lastname></author>
   <author><firstname>Mary</firstname><lastname>Fernandez</lastname></author>
   <title>The New Jersey Machine-Code Toolkit</title>
   <booktitle>USENIX</booktitle>
 </article>
</bib>
```

(a) bibliography.xml



(b) NFA-XML



(c) 최소화된 DFA-XML

그림 1 XML 문서의 형식화

그에 포함되는 속성, 엔터티와 같은 것은 으로 설명하기 위한 것으로 보고 필수 ㄱ 있는 요소만을 고려한다. 그리고 구 에서 형식화한 XML 문서 모델 으로 한다. 물론 요소와 구조를 별

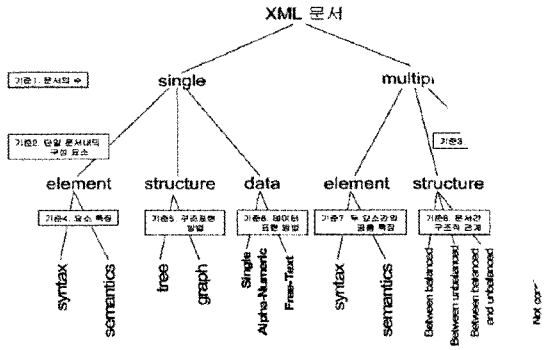


그림 2 XML 문서의 분류 기준

개의 것으로 분류하지 않고 요소를 구조를 형성하는 계층적인 관계로 포함시킬 수도 있다. 그러나 단지 요소를 구조를 이루는 하나의 형식 노드로 봄으로써 이를 충돌[13]과 같은 의미적인 차이에서 발생하는 문제점을 부각시키기는 어렵다. 따라서 본 분류 체계에서는 요소의 의미적인 측면을 강조하기 위해 구조와 동등한 레벨로 분류하였다.

- **기준 3. 다중 문서간의 관계(element/structure/data)** : 단일 문서를 구성하는 요소를 다중 문서로 확장하면 구성 요소들간의 관계로 볼 수 있다. 따라서 임의의 요소들간의 관계, 임의의 두 문서간의 구조적인 관계, 그리고 데이타간의 관계로 나누어 볼 수 있다. 그러나 다중 문서간의 데이타 관계를 가늠한다는 것은 형식적으로 분류하자면 숫자/숫자, 숫자/문자 등등으로 무의미한 비교가 되고 의미적으로는 무궁무진한 텍스트의 의미적인 관계를 1:1로 따지는 것이 불가능하므로 데이타 분류는 본 논문에서는 고려하지 않는다.
- **기준 4. 요소의 특징(syntax/semantics)** : 하나의 요소는 단어로 구성되므로 한 단어가 가지는 구문적인 정보와 의미적인 정보로 구분할 수 있다. 예를 들어 'author'라는 요소는 모두 소문자로 구성되며 '저자'라는 의미를 가지고 'writer'와 유사한 의미를 가진다.
- **기준 5. 구조 표현 방법(tree/graph)** : XML 문서의 내포적인 구조는 단순하게는 상하위 요소가 단일한 트리로 표현될 수 있고 만약 싸이클이 존재한다면 그래프로 표현되기도 한다. 기본적으로 XML 문서는 단일 루트 노트로부터 출발하기 때문에 forest는 발생하지 않는다.
- **기준 6. 데이타 표현 방법(single alphanumeric/free-text)** : XML 문서에서 요소의 실질적인 데이타는 알파벳이나 수를 포함한 단일어일 수도 있고 문장 태일 수도 있다.

- **기준 7. 두 요소간의 공통 특징(syntax/semantics)** : 다중 문서로 요소의 특징이 확장되더라도 'syntax'간의 특징은 동일하게 적용된다. 그러나 'semantics'의 특징은 사전적 유사어, 복합어, 생략어, 어근이 같은 경우 등으로 세분화되는 유사어 관계인가 아니면 다의어 관계인가로 재분류 된다.
- **기준 8. 문서간의 구조적인 관계(between balanced/between unbalanced/between bamanced and unbalanced)** : 이미 단일 문서에서 내포된 구조는 트리나 그래프로 표현되었다. 이때, 그래프로 표현된 문서에서 싸이클을 발생시키는 요소를 제거하고 트리화한 뒤 트리의 특징에 따라 균형 혹은 불균형 등으로 분류한다.

각 단계별 분류 범주는 위의 8가지 기준을 중심으로 단계별로 완전성을 보장하도록 구축하였으며 좀 더 자세한 범주별 타입은 다음 절에서 설명한다.

3.3 XML 문서 타입의 분류 체계

앞 절에서 제시한 큰 범주를 기반으로 하여 다음 표 1은 XML 문서의 타입을 분류한 것이다. 궁극적으로 'Naming' 필드가 XML 문서가 가질 수 있는 타입을 명명한 것이다. 또한 'Note'부분은 일부 타입의 경우 파라미터의 값에 따라 그 타입이 결정될 수 있다. 예를 들어 트리 차수(degree)의 균형 요소(balance factor)로서 β를 1로 보느냐 차수의 반으로 보느냐에 따라 문서의 타입이 결정된다. 이러한 파라미터는 실제 응용 시 구체적으로 정의될 수 있다.

모든 단일 XML 문서는 트리나 그래프로 표현된다. 따라서 단일 문서의 구조 타입은 표현된 구조의 트리적인 특성과 트리를 파괴하는 특성으로 그래프(이 때, 그래프는 방향성 그래프만으로 제한)적인 특성을 토대로 분류한다. 트리 구조는 높이와 차수의 균형 여부에 따라 'balanced tree'와 'unbalanced tree'로 구분할 수 있다. 자세한 정의는 다음 장에서 사례와 함께 설명한다. 또 균형 여부에 따라 구분된 'balanced tree' 구조는 레벨 수에 따라 'flat'한가, 'multilevel'인가로 구분하고 'unbalanced tree' 구조는 그 높이와 차수의 균형 여부에 따라 다시 조합된다.

한편 트리 구조를 깨는 그래프적인 특성으로 'cycle', 'shared node', 그리고 'ordering'이 있다. 'Multiple documents'의 구조 타입은 그래프 타입을 제거한 트리 타입만의 조합으로 생성된다. 즉, 그래프 타입은 단일 문서가 가지는 특성으로 남고 트리 타입은 전체적인 구조의 윤곽을 파악할 수 있게 하는 것으로 다중 문서간의 비교가 의미 있다. 'balanced tree'의 3가지 조합, 'unbalanced tree'의 6가지 조합, 그리고 'balanced와 unbalanced tree'사이에서 6가지 조합이 있을 수 있다.

거의 모든 XML 문서에서 사용되는 요소는 단일 문자열로 표현된다. 따라서 요소의 분류는 'single'이나 'multiple' 모두 문자열의 의미(semantics)와 구문(syntax)으로 분류할 수 있다. 하나의 문자열이 다른 문자열과의 관계에서 유사어(synonyms), 다의어(polysemy), 포함관계어(hyponymy), 반대어(antonymy) 등[15]이 존재할 수 있다. 그러나 대개의 XML 사용자들이 단일 문서 내에서는 고의적으로 같은 의미를 다른 단어를 사용해서 정의(synonyms)하거나 하나의 단어를 여러 의미로 사용(polysemy)하지 않는다고 가정한다. 실제 여러 XML 문서를 고찰한 결과 한 문서 내에서 이러한 예를 찾아 보기 어려웠다. 또한 본 분류 체계가 자연어 처리를 위한 시소러스를 구축하고자 하는 의도가 아닌 실제 XML 응용에서 사용될 수 있도록 하기 위한 목적이 있는 만큼 반대어와 같은 관계까지 파악하여 비교를 위한 계산의 복잡도를 증가시킬 필요는 없다. 따라서 'single document'의 요소에 대한 의미는 수 변화와 품사 변화에 의한 동일어근을 갖는 요소들만을 고려한다. 또한 구문상에서 대소문자를 구별하는가와 특수 문자를 포함한 것들을 허용하는가에 따라 구문의 차이를 보일 수 있다. 'multiple documents'의 요소는 다중 사용자로부터 정의된 요소들을 비교하므로 단일 문서의 경우 보다는 좀더 분류가 세분화 되어 같은 유사어라고 하더라도 사전적 유사어, 합성어, 그리고 생략어가 더 포함된다. 그리고 다의어를 추가적으로 고려한다.

XML의 데이타는 데이타 중심으로 표현되었는가 혹은 절이나 문장도 포함하는가에 따라 분류된다. 대부분의 XML 문서는 데이타와 텍스트가 혼합된 형태를 취한다. 그러나 다중 문서간에서는 자유롭게 정의된 데이타간의 관계를 파악한다는 것은 무의미하므로 다중문서에서는 고려 하지 않는다. 그럼, 다음 장에서 이러한 범주별 사례를 토대로 XML 문서 타입을 자세히 설명한다.

4. XML 문서 타입

이 장에서는 단일 문서 내에서 발생하는 'structure' 분류 → 'element' 분류 → 'data' 분류 → 다중 문서간에 발생하는 'structure' 관계 분류 → 'element' 관계 분류 순으로 설명한다.

4.1 단일 문서의 구조 분류(Single Document/Structure)

하나의 XML 문서의 구조의 트리 특성과 트리를 파괴하는 특성으로 나누어 'tree/graph(non-tree and directed)' 타입으로 구분한다.

4.1.1 트리 타입

트리로 표현되었을 때 그 트리의 높이와 차수에 따라

표 1 XML 문서 타입의 분류 체계

Category			XML Document Type			Naming	Note
Single	Structure	Tree	Balanced (both height and degree)	Flat		Flat	height <= α
				Multilevel		Multilevel	height ≥ α
			Unbalanced (either height or degree)	Height unbalanced	Degree unbalanced	Free	β = difference between degrees of subtrees
					Degree balanced	OddLevel	
				Height balanced	Degree unbalanced	OddDegree	
		Graph (non-tree and directed)	Cycle	Self-referred		Self-Cycle	
				Ancestor-referred		Ancestor-Cycle	
				Returned		Returned-Cycle	
			Shared node	On the same path		SharedSP	
				On different paths		SharedDP	
			Ordering	Sibling		SiblingOrder	
	Element	Semantics	Synonymy	The same root of words	Changes in the number	S-Sym1	
					Changes in parts of speech and the tense	S-Sym2	
		Syntax	Upper and lower case differentiation	Special character	Allowed	S-Syn1	
					Not allowed	S-Syn2	
			No differentiation	Special character	Allowed	S-Syn3	
					Not allowed	S-Syn4	
	Data		Single alpha-numeric			SingleData	
			Free text			FreeText	
Multiple	Structure	Between balanced	Homogeneous	Flat : Flat		BalancedFF	
				Multilevel : Multilevel		BalancedMM	
			Heterogeneous	Flat : Multilevel		BalancedFM	
		Between unbalanced	Homogeneous	Free : Free		UnbalancedFrFr	
				OddLevel : OddLevel		UnbalancedLL	
				OddDegree : OddDegree		UnbalancedDD	
			Heterogeneous	Free : OddLevel		UnbalancedFrL	
				Free : OddDegree		UnbalancedFrD	
				OddLevel : OddDegree		UnbalancedLD	
		Between balanced and unbalanced	Balanced flat tree	Flat : Free		FFr	
				Flat : OddLevel		FL	
				Flat : OddDegree		FD	
			Balanced multilevel tree	Multilevel : Free		MFr	
				Multilevel : OddLevel		ML	
				Multilevel : OddDegree		MD	
	Element	Semantics	Synonyms	Synonyms in dictionary		M-Sym1	γs= synonyms/ total elements between documents
				Compound word		M-Sym2	
				Abbreviation		M-Sym3	
				The same root of words	Changes in the number	M-Sym4	
					Changes in parts of speech and the tense	M-Sym5	
			Polysemy			M-Sym6	γp
		Syntax	Upper and lower case differentiation	Special character	Allowed	M-Syn1	
					Not allowed	M-Syn2	
			No differentiation	Special character	Allowed	M-Syn3	
					Not allowed	M-Syn4	

균형 트리 인가 아닌 가로 다시 세분화할 수 있다. 높이 균형 트리[16]의 정의는 다음과 같다.

정의 2. 높이-균형 트리(*height-balanced tree*)
공백 트리는 높이-균형을 이룬다. 만약 트리 A가 높이에 있어 1이하의 차이를 가지는 서브트리들만을 가지면서 서브 트리들 역시 높이-균형이라면 트리 A는 높이 균형-트리다.

다음 그림 3(c)의 경우 노드 1은 3 개의 서브트리를 갖는다. 정의 2에 의해 노드 1은 높이가 3, 4, 3인 높이-균형 서브트리를 가진다(그림에서 하나로 합쳐 표현된 data 노드는 실제로 분리되어 있는 단말 노드로 하나의 레벨로 표현될 수 있다.). 또한 재귀적으로 노드 2는 다시 높이 2, 1의 높이-균형 서브트리를 갖는다. 이런 식으로 모든 노드에 적용해 보면 모든 노드가 높이-균형임을 알 수 있다.

한편, 차수-균형 측면에서 XML 문서는 하나의 노드가 몇 개의 자손을 가진다고 미리 정의하기 어렵다. 그러나 트리의 가짓수를 제한하지 않은 트리에서 차수-균형 정의는 현재까지는 정의된 바 없다. 따라서 본 논문에서 정의 2를 이용하여 다음과 같이 정의한다.

정의 3. 차수-균형 (degree-balanced tree)
공백 트리는 차수-균형을 이룬다. 만약 서브트리 A가 차수에 있어 β이하의 차이를 가지는 서브트리들만을 가지면서 서브트리들 역시 차수-균형이라면, 트리 A는 차수-균형 트리다.

이 때 β를 가지의 최대수에 따른 상대적인 차이로 볼 때 차수의 반으로 정의 할 수도 있고 좀 더 강한 제약으로서 1로 정의 할 수도 있다. 차수는 β에 의존적이나 만약 β=degree/2로 정의한다면 그림 3(c)는 차수-균형이다. 그림 3(a)는 그림 1의 서브트리들이 모두 단말 노드로 차수-균형이다. 그러나 3(b)는 노드 2부터 n까지의 자손수가 β이하의 차이를 보이는 경우에만 차수-균형이다.

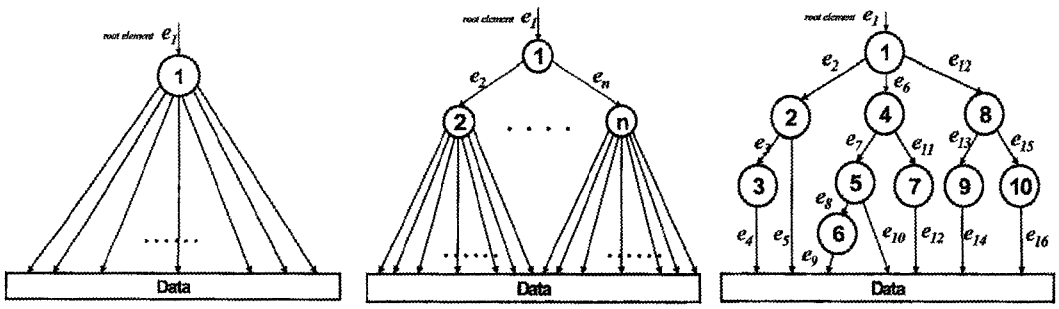
본래 XML은 계층적인 구조를 가지고 데이타의 의미를 좀 더 정확하게 표현하고자 하는 목적을 가진다. 하지만 과대한 계층 구조는 실제 데이타를 가지는 요소의 레벨이 깊어지면서 질의 처리나 XML 데이타 저장 등과 같은 분야에 성능 저하를 가져올 수 있다. 따라서 데이타의 의미를 적절한 수의 요소로 표현하는 것은 매우 중요하다.

만약 구조가 높이와 차수에 있어 모두 균형적이라면 레벨의 수에 따라 다음과 같이 'flat' 또는 'multilevel'로 구분할 수 있다.

- Flat(height <= α) : 어느 정도의 깊이를 가져야 'flat'한지 'multilevel'인지를 구분하는 α파라미터에 의해 정해진다. XML 문서는 항상 하나의 루트 요소를 가져야 하므로 어떠한 XML문서이든지 그림 3(a)처럼 level은 2이상이다. 더욱이 요소를 중복해서 많은 데이타를 표현하는 만큼 대부분의 문서는 3이상의 level을 갖는다. 따라서 구조가 'flat'한지 아닌지는 파라미터 α=2(그림 3(a))에 의해 결정할 수도 있고 α=3(그림 3(b))으로 정할 수도 있다. 또 임의의 어떤 수로도 응용에 따라 정할 수 있다.
- Multilevel(height > α) : 데이타의 계층 구조가 α보다 크면 계층적인 XML로 다룬다. 예를 들어 다음 그림 3(c)는 XML의 내포 관계를 이용하여 'multilevel'로 구성된 형태이다.

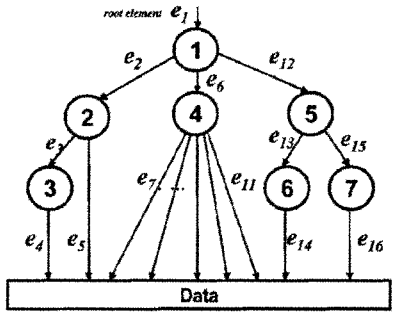
한편 비균형-트리 타입들은 높이가 비균형인가 차수가 비균형인가에 따라 다음 3가지 타입-'OddDegree', 'OddLevel', 그리고 'Free'-으로 분류된다.

- OddDegree(height-balanced and degree-unbalanced) : 만약 그림 3(c)에서 노드 4가 계층 구조를 가지지 않고 flat하게 다음 그림 4(a)와 같이 표현된다면 이는 노드 1의 서브트리의 차수가 2, 5, 2로 β를 degree/2로 정한다고 하더라도 균형적이지 않다. 이렇게 하나의 노드가 유난히 많거나 적은 수의 가지를 가지는 트리 타입을 말한다.

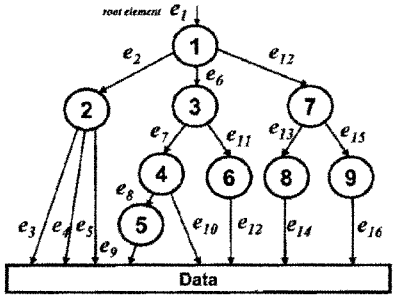


(a) Flat (if α = 2) (b) Flat (if α = 3) (c) Multilevel (height = 5)

그림 3 균형 트리 구조의 높이



(a) OddDegree



(b) OddLevel

그림 4 비균형-트리 타입

- OddLevel(height-unbalanced and degree-balanced) : 만약 그림 3(c)에서 노드 1의 가장 왼쪽의 서브트리가 'flat'한 구조로 그림 4(b)와 같다면 서브트리의 높이가 1, 3, 2로 높이에 있어 비균형적인 트리가 된다. 이렇게 하나의 노드가 유난히 많거나 적은 수의 높이를 가지는 트리 타입을 말한다.
- Free(height-unbalanced and degree-unbalanced) : 트리가 'OddLevel'과 'OddDegree'의 성격을 모두 가지고 있으면 'Free' 타입이라 할 수 있다.

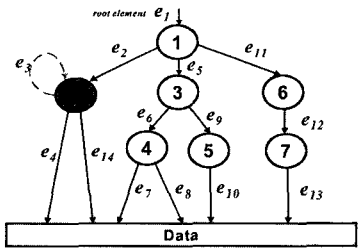
4.1.2 그래프 타입

XML 문서를 반드시 트리의 형태로만 표현할 수 있는 것은 아니다. 종종 사이클이 발생한다거나 하나의 노드를 공유한다거나 하는 트리를 파괴하는 타입들이 존재한다. 여기에서 그래프는 본래 XML의 특성상 한 방향으로 계층적 의미를 갖는 방향 그래프만을 가정한다.

- Cycle : XML의 구조상 사이클이 존재하는 경우, 발생 범위에 따라 자기 자신을 참조함으로써 발생하는 사이클(Self-cycle), 조상-참조에 의해 발생하는 사이클(ancestor-cycle), 다른 가지로 분기했다가 다시 귀환함으로써 사이클이 발생하는 경우(returned-cycle) 3가지로 구별된다. 'Self-cycle'은 요소가 스스로를 반복해서 참조하는 경우를 말하며 경로상에 같은 요소가 반복 발생하게 된다. 다음 그림 5(a)에서 노드 2는 e_3 요소에 의해 반복해서 참조될 수 있다. 예를 들어 이진 트리를 위한 DTD 문법으로 <!ELEMENT node(leaf|(node, node))> <!ELEMENT leaf (#PCDATA)>가 있을 때, 예제 XML 문서는 다음 그림 5(a)의 예제 XML 문서와 같이 표현될 수 있다. 조상-참조는 요소가 자기를 포함하는 경로상의 조상을 참조함으로써 사이클이 발생하는 경우를 말한다. 다음 그림 5(b)에서 노드 8은 자신의 부모 노드인 노드 3을 e_3로 참조함으로써 사이클이 발생한다. 예에서 보면 <url>에 의해 <body>, <link>가 반복된다. 귀환-참조의 경우 요소가 다른 가지의 노드를 참조하고 다시 원점(그 요소나 그 요소의 조상노드)으로 돌아오는 경우에도 사이클이 발생한다. 다음 그림 5(c)의 노드 4는 e_8에 의해 노드 5로 전이 되었다가 e_{10}으로 인해 노드 4로 귀환 된다. 이러한 관계는 그림 5(c)의 예에서 보듯이 <worksfor>와 <manager>, <child>와 <mother> 등과 같은 요소간의 의미적 관계에 의해서 발생할 수 있다.
- Shared node : 각기 다른 노드들이 하나의 노드를 공유할 수 있다. 공유 노드의 위치에 따라 같은 경로상에 있는 노드에 의해 공유되는가(SharedSP) 혹은 다른 경로상에서 공유되는가(SharedDP)에 따라 분류한다. 이는 경로를 축약해서 표현하거나 분기와 합병이 일어나는 경우에 발생한다. 'SharedSP'는 같은 경로상의 노드를 공유하는 경우로 다음 그림 6(a)의 상태 3은 상태 1과 2의 서로 다른 요소 e_5과 e_3에 의해 공유되는 노드이다. 예를 보면 <topic>은 <customized>와 <news>의 하위 요소가 된다. 구조상 경로를 최소화 시키는 경우 발생할 수 있다. 'SharedDP'는 다른 경로상의 노드를 공유하는 경우로 다음 그림 6(b)의 상태 5는 상태 3과 6의 서로 다른 요소 e_8와 e_{11}에 의해 공유된다. 경로상의 요소의 축약 형태인 'SharedSP'와는 달리 분기했다가 다시 합병되는 경우를 말한다.
- Ordering : DFA-XML 표현에서 각 노드와 요소들의 위치와 순서가 의미를 가질 수 있다. 트리 구조로 표현되면서 XML문서는 자연스럽게 상하위 요소간의 순서가 존재하게 된다. 또한 한 노드의 자손 노드들 순서를 유지하는 경우엔 'SiblingOrder' 타입이다. 그래프에서 하나의 노드에서 다음 노드를 가중치를 기준으로 선택할 수 있듯이 XML 손 노드의 순서를 유지할 수 있다. 대부분 D 라 작성된 XML 문서는 'SiblingOrder' 타 다. XML 응용에 따라 상하위 순서는 반 므로 따로 타입을 정의 하지 않는다.

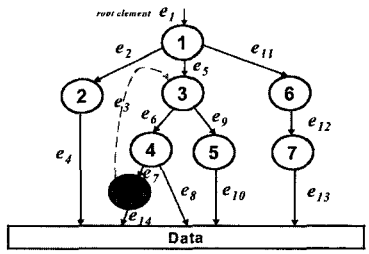
4.2 단일 문서의 요소 분류(Singl Element)

XML 문서의 구조를 구성하는 요소



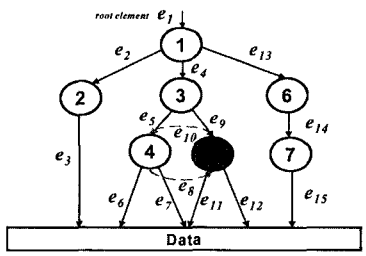

```
:
<node>
  <node>
    <node>
      <node> <leaf> 0203.jpg </leaf> </node>
    </node>
    <node> ··· </node>
  </node>
  <node> ···
:
```

(a) e_3 요소에 의한 노드 2의 자기-참조(Self-cycle)




```
:
<webpage>
  <page>
    <headlines>
      <title> Information Retrieval </title>
    </headlines>
    <body>
      <link>
        <headlines>
          <title> new indexing method </title>
        </headlines>
        <url>
          <body>
            <link>
              <image> paper1.jpg </image>
            </link>
          </body>
        </url>
      </link>
    </body>
:
```

(b) 노드 8의 조상-참조(Ancestor-cycle)




```
:
<company>
  <name> x-radar </name>
  <address> 11-1, daehyun </address>
  <manager>
    <name> Lee </name>
    <position> CEO </position>
    <worksfor>
      <name> x-radar </name>
      <address> 11-1, daehyun </address>
    </worksfor>
  </manager>
</company>
:
```

(c) 노드 5의 귀환-참조(Returned-cycle)

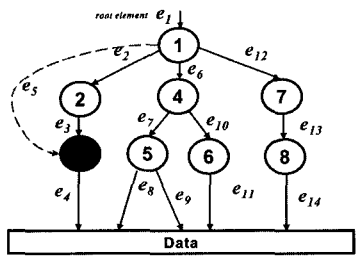
그림 5 사이클 타입

로 정의할 수 있으므로 같은 단어를 사용해서 요소를 정의하는 경우에도 다양한 의미의 차이(polysemy)를 가져올 수 있고 반면 다른 단어를 사용하여 정의된 요소라 하더라도 같은 의미(synonyms)를 가질 수 있다. XML 요소는 크게 의미적인 측면과 구성상의 특징을 중심으로 나눈다.

4.2.1 의미(Semantics)

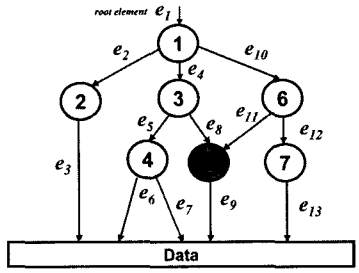
요소의 의미의 차이를 식별하는 것은 XML의 구조를 평가할 때에도 커다란 영향을 미친다. 예를 들어 서로 다른 요소로 정의되어 있는 <author>와 <writer>의 경우 같은 요소로 인식이 되어야만 그 뒤에 구조적으로 요소가 중복되어 있는지, 경로가 중복되어 있는지 등을 판단할 수 있는 것이다. 이러한 요소의 의미의 유사성을 'synonymy'이라 한다. 그러나 하나의 문서 내에서 사용자가 고의적으로 '저자'라는 표현을 <author>라고 표기했다가 <writer>라고 바꾸었다가 하는 경우는 없다고 가정한다. 실제 여러 문서를 대상으로 분석한 결과 한 문서 내에서 혼용된 예를 찾아 보기 힘들었다. 따라서 하나의 XML 문서에서의 요소의 의미의 차이는 동일 어근을 갖는 단어에 국한한다. 따라서 합성어나 생략어 등의 좀 더 다양한 유사어 종류는 다중 문서간의 관계를 따지는 서로 비교 가능한 4.5절의 'multiple documents'에서 논의 된다.

- Synonymy : 한 문서 안에서의 유사어는 동일한 어근을 갖는 문제에 국한한다. 동일 어근을 가지고 표현하는 데에는 <chapters>와 <chapter>와 같은 수 변화와 <information>과 <inform>과 같은 품사 변화, 그리고 <informed>, <informing>과 같은 시제 변화를




```
:
<recent_news>
  <news>
    <headlines>
        <topic> airline crash </topic>
    </headlines>
  </news>
  <customized>
    <topic> election day··· </topic>
  </customized>
</recent_news>
:
```

(a) 동일 경로상의 공유 노드(SharedSP)




```
:
<recent_news>
  <news>
    <headlines>
        <topic> airline crash </topic>
    </headlines>
  </news>
  <customized>
    <social>
        <topic> airline crash </topic>
    </social>
  </customized>
</recent_news>
:
```

(b) 다른 경로상의 공유 노드(SharedDP)

그림 6 공유 노드 타입

예를 들 수 있다.

4.2.2 구문(Syntax)

단일 문서나 다중 문서간에서 요소의 구문상의 특징은 대소문자를 구별할 것인가에 따라 구분하고 또 밑줄, 하이픈, 쉼표(comma), 마침표(period), * 등과 같은 특수 문자를 허용하는가에 따라 다시 재분류 될 수 있다. 일반적인 XML에서는 대소문자를 구별하는 것을 원칙으로 하지만 응용 목적에 따라 구분하지 않을 수도 있다.

4.3 단일 문서의 데이타 분류(Single Document/Data)

요소 사이, 즉 여는 태그와 닫는 태그사이에 표현되는 데이타의 종류에 따라 단순 문자열 혹은 자유 텍스트 형식으로 구분할 수 있다. XML 응용 분야에 따라 같은 구조를 가진 XML 문서라 하더라도 표현되는 데이타의 종류에 따라 이를 저장, 인덱싱, 그리고 질의하는 방식이 달라질 수 있다.

4.4 다중 문서간의 구조 관계에 따른 분류(Multiple Documents/Structure)

다중 문서간의 구조적인 관계는 각각의 문서가 단일 문서의 분류에 따른 타입들로 분석되고 그 결과를 토대로 그 구조간의 관계를 분류하는 것을 의미한다. 따라서 다중 문서간의 관계는 단일 문서의 모든 타입을 고려한 후 그 조합에 따라 나뉜다. 단일 문서에서 트리를 파괴하는 요소들을 제거한 후 트리의 타입만을 가지고 조합한다. 다중 문서간의 비교는 기본적으로 두 문서간의 관계로 확장한 것으로 본다. 그러므로 크게 3가지 범주(균형-트리간, 비균형-트리간, 그리고 균형과 비균형 트리간의 조합-로 분류한다.

4.4.1 균형-트리간의 조합 (Between balanced trees)

균형 트리간의 조합은 'Flat' 한 트리(F)와 'Multi-level' 트리 타입(M)의 조합으로 flat : flat(balanced-FF), multilevel : multilevel (balancedMM), flat : multilevel(balancedFM)으로 조합된다.

4.4.2 비균형-트리간의 조합(Between unbalanced trees)

비균형-트리간의 조합은 'Free(Fr)', 'OddLevel(L)', 'OddDegree(D)'의 조합에 따라 6가지(unbalncedFrFr, unbalancedLL, unbalancedDD, unbalancedFrL, unbalancedFrD, unbalancedLD) 타입으로 조합된다.

4.4.3 균형과 비균형-트리간의 조합 (Between balanced and unbalanced trees)

균형-트리와 비균형-트리간의 조합은 'Flat(F)', 'Multilevel(M)', 'Free(Fr)', 'OddLevel(L)', 'OddDegree(D)'의 조합에 따라 6가지(FFr, FL, FD, MFr, ML, MD) 타입으로 조합된다. 이 때, 비교된 트리 타입의 높이와 차수간의 격차에 따라 더 세분화 될 수도 있다.

4.5 다중 문서간의 요소 관계에 따른 분류(Multiple Documents/Element)

4.2절에서 설명한 바와 같이 요소의 의미 차이 식별은 구조간의 분류를 위해서 기본적으로 제공되어야 하는 기술이다. 단일 문서의 분류에서는 한 문서 내에서는 사용자가 고의적으로 같은 요소 설명을 위해 다른 용어를

사용하지 않는 것을 가정하였다. 따라서 수 변화와 품사 변화만이 'synonymy'의 대상이 되었지만 다중 문서에서는 다중 사용자를 가정한다. 따라서 같은 데이타 설명을 위해서도 서로 다른 요소를 사용할 수 있으므로 서로 유사한 요소를 발견해 내는 일은 매우 중요하다. 어느 사용자는 <bib>라는 요소를 bibliography의 약자로 사용했고 다른 사용자는 어류의 일종인 pout을 말하기 위해 사용했다고 하자. 이렇게 하나의 요소가 여러 의미로 해석될 수 있는 것을 'polysemy' 관계라고 할 수 있다. 또한 bibliography를 표현하기 위해 <bib>을 쓰기도 하고 <biblio>를 사용하기도 한다. 이러한 여러 유사어의 정의(synonymy), 다의어(polysemy)와 같은 의미 분류[15]를 설명하고 구문 분류는 단일 문서와 동일하여 생략한다.

4.5.1 유사어(Synonymy)

유사한 요소로 인식될 수 있는 종류는 사전적으로 유사한 경우, 두개 이상의 단어로 결합된 복합어, 생략어, 그리고 단일 문서와 마찬가지로 동일 어근을 갖는 용어가 있다. 다중 문서간의 요소간의 관계는 전체 요소에서 유사어가 차지하는 비율을 계산하여 파라미터 γ_s를 설정할 수 있다. γ_s가 높으면 높을수록 유사한 요소를 사용하고 있는 문서들이 된다.

- 사전적 유사어(M-Sym1) : 같은 데이타를 설명하기 위하여 사전적으로 유사한 단어, 즉 동의어로 요소를 정의할 수 있다. <search>는 [searching, hunt, hunting, lookup] 등으로, <record>는 [recordbook, book] 등으로 다양하게 표현될 수 있다.
- 복합어(합성어, M-Sym2) : 두개 이상의 단어가 합쳐져서 새로운 의미를 나타낼 수도 있고 본래의 단어의 의미를 강조할 수 있다. 예를 들어 <publisher>는 동일한 의미로 [publishing_house, publishing_company] 등으로 사용될 수 있다. 반면 <profiledesc>, <moreinfo>, <older_article> 등과 같이 사전적으로는 예측 불가능한 형태도 있다.
- 생략어(M-Sym3) : XML은 일반적으로 생략어를 사용하는 경우가 많다. HTML 같은 마크업 언어를 보면 이미 정의된 태그들은 거의 생략어임을 알 수 있다. XML도 마찬가지로 <firstname>을 <fname>으로, <bibliography>를 <bib>으로, 그리고 <paragraph>를 <p> 또는 <para> 등으로 사용한다. 한 문서는 생략어를 사용하고 다른 문서는 사용하지 않았더라도 이를 유사한 요소로 인식할 수 있어야 한다. 생략하는 방법에 따라 Mister를 Mr., Dr., pages를 pp로 생략하는 것처럼 일반적으로 사용되는 생략어가 있는가 하면 paragraph를 p, description을 dsc로 생략하거나 Customer Relationship Management(CRM)과 같은 전문 용어의 생략과 같이 특수한 생략어가 존재한다.
- 동일 어근에 대한 설명은 단일 문서의 분류와 같다(M-Sym4, 5). 그러나 한 문서 내에서 발생한 요소간의 구별이 아닌 서로 다른 문서의 요소간의 관계를 의미한다.

4.5.2 다의어(Polysemy)

일반적으로는 XML에서 의도적으로 하나의 단어를 여러 의미로 사용하는 경우는 드물다. 즉, 한문서안에서 <bib>이 bibliography 의미로 사용하다가 갑자기 어류를 표현하기 위해 재사용할 사용자는 아무도 없을 것이다. 그러나 <title>이 사용되었을 때, 부모가 <person>이냐, <book>이냐에 따라 약간의 의미의 차이가 발생한다. <name> 역시, <author> 아래에 있느냐, <car_brand> 아래에 있느냐에 따라 각기 다른 종류의 이름을 설명하게 될 것이다. 또한 <more_info>와 같은 추상적인 단어는 어느 요소 아래에 있느냐에 따라 설명할 수 있는 데이타가 달라 질 것이다. XML에서는 이렇게 상하위 요소에 의해 의미 변화를 일으키는 경우를 다의어 타입이라 할 수 있다. 역시 전체 요소에서 다의어 관계의 비율을 계산하여 파라미터 γ_p를 설정할 수 있다.

지금까지는 XML 문서가 가질 수 있는 타입을 중심으로 하나의 문서 내에서 그리고 다중 문서간에서 발생할 수 있는 커다란 범주 하에서 44개의 세부적인 타입을 갖는 분류 체계를 구축하였다. 체계적인 XML 문서 타입의 분류는 XML 관련 연구의 기초를 제공하고 응용 분야의 선택적인 적용, 적용시 결과에 대한 평가와 더불어 방법론의 XML 타입에 대한 처리 능력을 평가함으로써 결과의 신빙성을 증진시킬 수 있다. 그렇다면 다음 장에서는 본 분류 체계가 XML 관련 연구에 어떻게 적용될 수 있는지를 기술한다.

5. 분류 체계의 응용

XML을 처리하는 응용 분야에 따라 단일 문서의 분류 결과가 중요할 수도 있고 단일 문서와 다중 문서 분류가 모두 중요할 수도 있다. 예를 들어 일정한 DTD를 공유하는 XML 문서들을 데이타베이스에 저장하기 위해, 일부 손실되는 구조를 감수하고라도 최적화된 스키마를 추출하기 위해서는 단일 문서의 분류가 주로 분석 대상이 되어야 할 것이다. 또한 XML 질의를 위한 경로 표현식을 지원하기 위해서도 단일 문서의 분류들을 집중적으로 고려해서 설계되어야 할 것이다. 반면 상이한 구조와 요소 이름을 가지는 여러 문서들을 합병하기 위해서나 유사한 XML 문서들을 군집화(clustering) 혹은 분류(classification)하기 위해서는 'single/multiple' 분류 결과 모두가 분석 대상이 되어야 할 것이다.

본 논문에서 분류한 XML 문서 타입의 분류 체계를 기반으로 XML 문서 마이닝을 위한 전처리 과정[17]에 적용해 보았다. XML 문서를 마이닝 하기 위해서 XML 문서의 특징을 추출해 내고 이를 질량화 하는 전처리 과정에, 분류 체계에서 분석된 XML 문서 특성을 얼마나 고려할 수 있는 지 파악해 본다. 이러한 적용은 다음 두 가지를 알아내어 제안된 XML 문서 마이닝을 위한 전처리 방법의 타당성을 입증하는 데에 그 목표를 둔다.

- 방법론의 타입 반영도 : 제안된 XML 마이닝 방법론이 특징을 추출해 내는 문서 분석 과정에서 어떠한 XML의 특징을 다룰 수 있는가?
- 실험 대상 선정의 적정 : 마이닝 방법의 검증을 위해 선정된 문서는 어떠한 XML 타입을 포함하는가?

먼저 간단히 XML 문서 마이닝을 위한 전처리 과정을 설명하고[17] 이 방법론이 단계별로 분류 체계에서 분류된 타입들을 반영할 수 있는지 설명한다. 그리고 간단한 예를 통하여 분류 체계를 기반으로 그 문서의 타입을 제시한다.

5.1 의미-기반 XML 문서 마이닝을 위한 전처리 과정

XML 문서 마이닝은 크게 군집화와 분류로 나눌 수 있다. 이러한 연산을 위해서는 각각의 문서들의 특징에 대한 양적 측량이 중요하다. 예를 들어 유사한 문서들끼리 군집화 하기 위해 문서간의 거리를 계산하거나, 또는 주어진 문서와 유사한 문서들을 분류 하기 위해 기준 문서와의 유사 정도를 계산하게 된다. 이 때, 문서의 유사도를 정확하게 계산하기 위해서 XML 문서의 의미(semantics)가 충분히 고려되어야 한다. 즉 마이닝 결과의 정확도는 마이닝 알고리즘도 중요하지만 전처리 과정에서 얼마나 XML의 특징을 정확하게 반영하였는가에 달려 있다. 따라서 정확한 XML 문서 마이닝 결과를 얻기 위해서 XML의 의미를 충분히 반영할 수 있도록 XML 문서를 준비하고 그 결과를 질량화 하는 방법이 요구된다. 따라서 이미 제안된 XML 문서 마이닝을 위한 전처리 과정이 본 논문에서 제안한 XML 문서의 분류 체계에 나타난 타입들을 얼마나 반영할 수 있는가를 분석해 본다. 먼저 [17]에서 제안된 마이닝의 전처리 방법론을 간략하게 설명하면 다음 그림 7과 같다.

- XML 구조 발견 및 최소화 : DTD나 XML Schema가 아닌 XML 문서 자체를 정의 1의 NFA-XML로 형식화 하고 동일한 요소를 가지고 중복된 전이를 일으키는 상태들을 합병하여 DFA(Deterministic Finite Automata)-XML로 표현한다. 다시 동일(equivalent) 상태를 합병하는 상태-최소화 알고리즘[14]을 이용하여 DFA-XML을 최소화 시킨다.
- 경로 시퀀스 추출 : 최소화된 DFA-XML의 루트 요소로부터 데이타를 유도하는 단말 요소까지의 경로 정보를 추출한다. 이때 상태 숫자는 생략하고 요소만을 연결한다.
- 유사어를 가지는 요소로 확장 : 경로 시퀀스를 구성하는 용어(요소이름)들을 WordNet과 사용자-정의 라이브러리를 이용하여 유사어를 검색하고 이를 확장하여 각 요소에 대해 확장-요소 벡터(extended-element vector)를 만든다. 그리고 요소간의 유사 행렬을 구축하여 각 문서들간의 가장 유사한 요소 쌍을 식별한다.
- 경로 시퀀스 재명명 : 각 문서들은 추출된 경로 시퀀스들을 정수로 재명명 하기 위해 매핑 테이블을 구축한다. 기준 문서의 요소들을 1..n까지 숫자화하고 다시 비교 문서의 요소들을 숫자화 한다. 이때 유사한 요소는 동일 숫자로 명명하게 된다.
- 최대 유사 경로 찾기 : 변형된 순차 패턴 마이닝 알고리즘을 이용하여 두 문서간의 최대 유사 경로를 구한다.
- 유사 척도 생성 및 유사성 계산 : 도출된 최대 유사 경로와 유사성 계산 척도를 토대로 문서간의 유사성을 계산한다

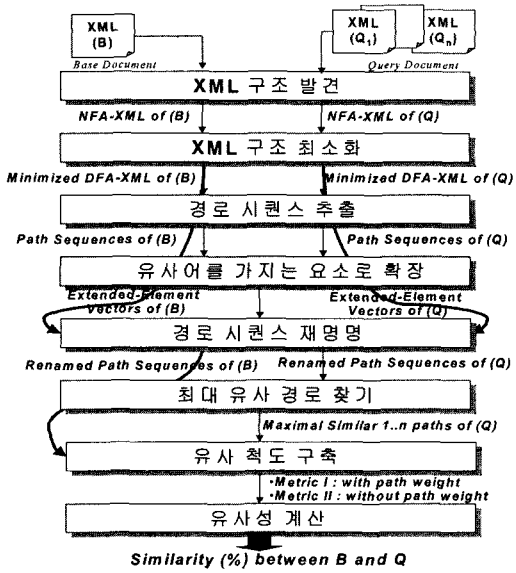


그림 7 의미-기반 XML 문서 마이닝의 전처리 과정

다음 두 절에서는 이 전처리 방법론에 대해 분류 체계를 기반으로 방법론의 타입 반영 정도와 선정 대상의 문서 타입을 분석한다.

5.2 분류 체계를 기반으로 전처리 방법론의 XML 문서 타입의 반영

엄밀하게 XML 의미를 추출해 내는 단계는 유사 척도를 만들기 전까지의 단계로 본다. 따라서 전처리 과정을 총 4 단계로 보고 본 논문에서 분류된 분류 체계 타

입의 반영여부에 따라 다음 표 2와 같이 나타낼 수 있다. 각 단계에서 생성된 산출물 중심으로 Minimized DFA→Extended-element vectors→Renamed sequences→Maximal similar paths 단계를 거치면서 하나의 XML 문서의 타입을 반영하고 다중 문서간의 관계 고려 하게 된다. 나누어진 4 단계에서 각 분류 타입들의 처리가 완성되는 단계에 √표시를 하였다. 'Multiple documents/structure'와 'polysemy' 분류는 궁극적으로 유사도를 계산할 때 반영되지만 그 기초 자료가 완성되는 단계를 기준으로 표시하였다. 마지막 'consideration' 필드는 최종적으로 분류된 타입을 고려할 수 있는가(○), 전혀 고려하지 못하느냐(×), 그리고 부분적으로 고려하는가(△)로 표기하였다.

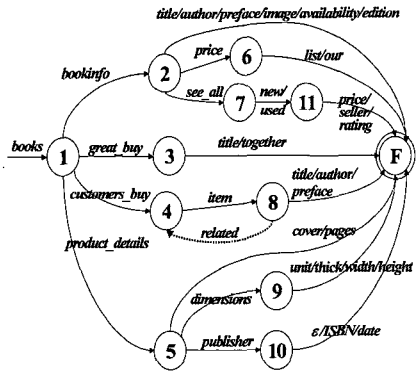
표 2로 인하여 의미-기반 XML 문서 마이닝을 위한 전처리 방법은 전체적으로 여러 타입들을 반영하지만 형제 노드간의 순서, 데이타를 반영하지 못한다는 것을 알 수 있다. 이는 문서 마이닝 시 같은 레벨에 위치한 요소라면 용어의 나열 순서가 용어 가중치[18]에 영향을 미치지 않기 때문에 굳이 형제 노드간의 순서를 고려할 필요가 없다. 또한 여기에서 XML의 의미는 구조와 구조를 구성하는 요소 정보만을 가정한 것이므로 데이타를 고려하지 않는다. 다의어(polysemy)는 전처리 방법에서 상하위어들이 서로 다른 것이라면 같은 용어라 하더라도 다른 의미를 내포하고 있을 것으로 판단할 수 있다. 하지만 분명히 다른 의미를 표현하고 있으나 상하위어가 같다면 다의어로 인식할 수 없다. 따라서 부분적으로 고려할 수 있기 때문에 △라고 마크 하였다.

이와 같이 XML의 새로운 방법론의 XML 문서 타입의 처리 능력을 일목요연하게 제시함으로써 응용 분야에 따른 개발된 방법론의 선택, 기법의 부족한 측면 보강, 같은 결과를 내는 다른 방법론과의 타입 여부 비교 등을 가능하게 할 수 있다.

5.3 문서 타입 분석

5.3.1 온라인 서점 관련 문서

다음 그림 8은 XML 문서로부터 구조를 추출, 최소화시킨 DFA-XML을 보여준다. 두 문서는 온라인 서점에



(a) a.xml

(b) b.xml

그림 8 대상 문서의 DFA-XML

표 2 의미-기반 XML 문서 마이닝 전처리 방법론의 XML 문서 타입 반영

Taxonomy of XML Document Types				Preparations				Consideration
				Minimized DFA	Extended-Element Vectors	Renamed Sequence	Maximal Similar Paths	
Single	Structure	Tree	*According to balanced and unbalanced*	√				○
		Graph	*Cycle*	√				○
			Shared node	√				○
			SiblingOrder					×
	Element	Semantics and syntax			√			○
	Data	Single alpha-numeric and free-text						×
Multiple	Structure	*According to the combinations*					√	○
	Element	Semantics	*Synonym*			√		○
			Polysemy				√	△
		Syntax			√			○

표 3 XML 문서 타입

Category		a.xml	b.xml
Single	Structure	*Free tree(α=1, β= degree/2), Ancestor-cycle, SharedSP*	*OddDegree tree(α=1, β= degree/2), SharedSP, SharedDP*
	Element	*S-Syn3*	*S-Syn3*
Multiple	Structure	*UnbalancedFrD*	
	Element	*M-Syml, 2, 3, 4, 5 γ_s=0.34), M-Sym6 (γ_p=0.068), M-Syn3*	

서 산출된 문서로 동일한 목적(책 판매 정보)을 갖는다. 실제 'WordNet'이라는 책을 찾은 두 온라인 서점의 결과 페이지를 참조한 것이다. 편의상 두 문서는 결과 페이지의 서평 부분이나 책의 데이타 부분에 관련된 부분을 제거한 일부 정보만을 보여주고 있다. 사용하고 있는 요소는 같은 목적, 즉 책을 판매하기 위한 목적을 지니므로 많은 동의어들로 구성되어 있다. 또한 유사 요소로 표현된 구조는 일부 유사하다.

이 문서의 타입을 분류 체계를 토대로 분석하면 표 3과 같다. 다중 문서간의 구조 타입을 파악할 때에는 공유 노드와 사이클과 같은 트리를 파괴하는 요소는 제거하고 분석한다. 또한 파라미터 값으로 α=1, 그리고 β= degree/2로 보았을 때의 트리 타입이다.

a.xml 문서는 <related>에 의해 조상-참조 사이클(Ancestor-cycle)이 발생하고 <new>는 <used>와 함께 노드 11을 같은 경로상에서 공유(SharedSP)하고 있다. 이러한 non-tree 타입을 제거하면 books.great_buy.title.⊥(⊥는 종결 상태의 데이타를 의미)와 같은 서브트리(height=3)와 books.bookinfo.see_all.new.price. ⊥경로를 포함한 서브트리(height=5)를 동시에 갖는다. 또 노드 2는 8개의 가지(하나의 에지로 표현된 title/author/preface/image/availability/edition 에지는 실제 6개의 에지들로 표현된다.)를 가지고 있고 노드 4는 1개의 가지를 갖는다. 따라서 a.xml문서는 높이와 차수 모두 비균형적인 트리(Free)이다. 한편, b.xml은 노드 6을 노드 2와 7이 다른 경로상에서 공유(SharedDP)하고 <writer>와 <foreword>는 노드 6을 같은 경로상에서 공유(SharedSP)하고 있다. 또한 트리에서 노드 2와 3을 비교해 보았을 때 차수만 비균형(OddDegree)적이다. 따라서 두 문서의 관계는 UnbalncedFrD 타입이 된다.

그리고 두 문서의 요소간의 관계는 γ_s(ratio of synonyms to total elements), γ_p(ratio of polysemy to total elements)로 요약될 수 있다. 즉 유사한 요소들은 <preface>와 <foreword>, <image>와 <picture>등으로 34%, 다의어는 <book_information>과 <item>아래 있는 <title> 등 6.8%, 그리고 완전히 똑 같은 요소 10%로 이러한 parameter의 총 합은 두 문서가 같은 목적으로 사용되고 있는 element들(50.8%)에 대한 통계가 된다. 그리고 두 문서가 모두 대소문자를 구분하지 않으며 특수 문자인 _(underbar)를 허용(M-Syn3)하고 있다.

이렇게 제시된 문서의 타입을 분류 체계를 기반으로 분석함으로써 기존에 XML응용에 선정된 문서들을 단순히 그 크기로 비교했던 것과는 달리 구조의 복잡도, 다중 문서간의 관계, 요소 사용의 유사 정도 등을 한꺼번에 비교할 수 있다.

5.3.2 전자상거래 관련 문서

표 4 XML 문서 타입 분석 (α=2, β= ⌈ degree/2 ⌉)

분 류		books	electronics	music	video
단일	구조	Free	Free	OddLevel	Free
	엘리먼트	S-Syn3			
다중	구조와 엘리먼트	UnbalancedFrFr			book과 electronics
		M-Syml (γ_s=0.043), M-Syn3 (동일 element : 0.43)			
		UnbalancedFrL			book과 music
		M-Syml, 2 (γ_s=0.097), M-Syn3 (동일 element : 0.55)			
		UnbalancedFrFr			book과 video
		M-Syml (γ_s=0.073), M-Syn3 (동일 element : 0.33)			
		UnbalancedFrL			electronics와 music
		γ_s=0.0, M-Syn3 (동일 element : 0.55)			
		UnbalancedFrFr			electronics와 video
		γ_s=0.0, M-Syn3 (동일 element : 0.51)			
		UnbalancedFrL			music과 video
		M-Sym2 (γ_s=0.071), M-Syn3 (동일 element : 0.46)			

다음 표 4는 전자상거래 문서 4개를 대상으로 분석한 결과이다. 이 문서들은 매우 복잡한 구조를 가지고 있고 팔고자 하는 물건은 'books', 'electronics', 'music CD', 그리고 'video tape'으로 서로 다르지만 전자상거래에서 사용되는 문서들이다.

네 가지 범주의 문서가 대게는 **Free** 트리이고 **music** 범주만이 **OddLevel**이다. 동일 엘리먼트의 사용은 33%~55%까지 다양했고 **'book'**과 **'music'**범주의 문서들이 유사어 포함 총 64.7%의 엘리먼트를 공유하고 있음을 한 눈에 알 수 있다.

지금까지 XML 문서 타입에 대한 분류 체계를 기반으로 XML관련 방법론이 어떠한 타입을 반영할 수 있고 또 선정된 대상 문서가 어떠한 타입을 갖는지 분석 가능하다는 것을 보였다. 이는 XML 문서 타입에 대한 분류 체계가 새로운 방법을 개발하거나 방법들을 비교, 평가하는 데 기준이 되는 기초 자료를 제공하고 방법론 검증 시 대상 문서의 다양한 선정을 가능하게 할 수 있음을 보인 것이다.

6. 결론 및 향후 연구

본 논문에서는 XML 문서의 타입을 요소와 구조 그리고 데이타 측면에서 도출해 내고 이를 체계적으로 분류하였다. 크게 단일 문서 내에서의 특성과 다중 문서간의 관계를 중심으로 세부적으로 44가지의 타입을 얻을 수 있었다. 구축된 분류 체계를 XML 문서 마이닝의 전처리 과정에 적용해 봄으로써 이전에 알 수 없었던 방법론의 특성과 실험 대상의 특성을 한 눈에 파악할 수 있었다. 따라서 어느 XML 관련 응용 분야이든지 적용 방법론이 XML 문서의 타입들을 얼마나 반영하고 있는지, 그리고 대상 문서는 어떤 타입인지를 파악하기 위해서 본 논문에서 제시한 분류 체계를 사용할 수 있다. 또한 연구 초기 단계에서 처리해야 할 XML의 문제들에 대해 체계적인 계획을 수립할 수 있는 기초를 제공할 수 있다. 앞으로는 적용 분야를 XML 저장 기법, XML 질의 처리 등에 확대해 나가며 분류 체계를 보다 확장할 계획이다.

참 고 문 헌

[1] World Wide Web Consortium, "Extensible Markup Language(XML) 1.0 (2nd Edition)," W3C Recommendation, Oct. 2000.(http://www.w3c.org/TR/REC-xml).

[2] Makoto Murata, Dongwon Lee, and Murali Mani, "Taxonomy of XML Schema Languages using Formal Language Theory," Extreme Markup Languages, Montreal, Canada, August. 2001.

[3] Gyeong-Ja Jang and Kiho Lee, "XML-QL to SQL Translator for Processing XML Data," Journal of KISS(Korean Information Science and Society), Vol. 8(1), Feb. 2002.

[4] Dongwon Lee, and Wesley W. Chu, "Comparative Analysis of Six XML Schema Languages," SIGMOD Record 29(3), pages 76-87, 2000.

[5] Angela Bonifati, and Stefano Ceri, "Comparative Analysis of Five XML Query Languages," SIGMOD Record 29(1), pages 68-79, 2000.

[6] R. Goldman and J. Widom, "DataGuides : Enabling Query Formulation and Optimization in Semistructured Databases," proc. of the 23rd International Conference on Very Large Data Bases, pages 436-445, Athens, Greece, August 1997

[7] Murata Makoto, "Hedge automata : a formal model for XML schemata," http://xml.coverpages.org/hedgeAutomata.html.

[8] S. Abiteboul, P. Buneman, and D. Suciu, Data on the Web : from relations to semistructured data and XML, Morgan-Kaufmann, 2000.

[9] E. Horowitz, S. Shani, and D. Mehta, Fundamentals of Data Structures in C++, Computer Science Press, 1995.

[10] P. Kilpelainen and H. Mannila, "The Tree Inclusion Problem," Proc. the International Joint Conference on the Theory and Practice of Software Development (TAPSOFT'91), Vol. 1: Colloqium on Trees in Algebra and Programming (CAAP '91), pages 202-214, 1991.

[11] Bunke Horst and Shearer Kim, "A Graph Distance based on the Maximal Common Subgraph," Pattern Recognition Letters, Elsevier Science, (19)3-4 : pages 255-259, 1998.

[12] Jason T. L. Wang et al, "An Algorithm for Finding the Largest Approximately Common Substructures of Two Trees," IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 20(8), pages 889-895, 1998.

[13] Seung-Won Lee et al., "The Classification of Conflicts on Schema Integration with XML Schema," proc. of KISS Fall Conference, 2001.

[14] A.V.Aho, R.Sethi and J.D.Ullman. Compilers : Principles, Techniques, and Tools, Addison Wesley, 1986.

[15] C.Fellbaum, WordNet : An Electronic Lexical Database, Cambridge:MIT Press. 1998.

[16] http://www.nist.gov/dads/HTML/height/heightBalancedTree.html

[17] Jung-Won Lee, Kiho Lee, and Won Kim, "Preparations for Semantics-based XML Mining," proc. of IEEE International Conference on Data Mining (ICDM '01), pages 345~352, San Jose, Nov. 2001.

[18] Gerard Salton and Michael J. McGill, Introduction

to Modern Information Retrieval, McGraw-Hill Book Company, New York, 1983

이 정 원

1993년 이화여자대학교 전자계산학과 이학사. 1995년 이화여자대학교 전자계산학과 이학석사. 1995년 2월~1997년 5월 LG 종합기술원 연구원. 1997년 9월~2003년 2월 이화여자대학교 컴퓨터학과 공학박사. 2000년 6월~2000년 8월 IBM Almaden (USA) 연구소 인턴쉽. 2003년 3월~2004년 2월 이화여자대학교 공학연구소 박사후연구원. 2004년 3월~8월 이화여자대학교 컴퓨터학과 연구교수. 2004년 9월~현재 이화여자대학교 컴퓨터학과 전임강사. 관심분야는 XML, 데이타마이닝, 프로그래밍 언어

박 승 수

1974년 서울대학교 수학과 학사. 1976년 한국과학기술원 전산학 석사. 1988년 미국 텍사스 대학 전산학 박사. 1988년~1991년 미국 캔사스대학 컴퓨터학과 조교수. 1991년~현재 이화여자대학교 컴퓨터학과 교수. 관심분야는 인공지능, 데이타마이닝, 바이오인포매틱스