의미 분석과 부호화된 구조 분석을 이용한 XML 자동 변환

양홍준 ○ 곽동규 문현주 유재우

숭실대학교 컴퓨터학과

{garamyunseul○, coolman}@ss.ssu.ac.kr hyunjoomoon@gmail.com cwyoo@ssu.ac.kr

Automating XML documents Transformations based on Semantic and Encoded Structure Analysis

Hong-Jun Yang, Dong-Guy Kawk, Hyun-Joo Moon, Chae-Woo Yoo

Dept. of Computing, Soongsil University

요 약

XML은 W3C 표준으로 채택된 이후로 많은 어플리케이션에서 데이터를 표현하는 방법으로 사용되고 있다. XML문서는 특정 어플리케이션에 종속적이기 때문에 XSLT를 이용하여 변환한 뒤 사용하게 된다. 그러나 변환에는 많은 노력, 시간과 비용이 소요되기 때문에 이를 자동으로 변환하는 시스템을 구축하는 것이 최선의 방법이다. 이를 위해서 XTGen이나 XSLT 스크립트 시스템이 기존에 제안되었지만 사용자가 엘리먼트간의 관계를 수동으로 처리하는 방식이거나 변환 문서간 단말 노드의 1:1 매칭이라는 제약과 대규모 변환에 어려움이 있다. 본 논문은 JAWS를 이용한 엘리먼트간의 의미 관계 분석과 DTD의 구조를 분석하여 XSLT를 생성함으로써 기존 시스템들의 단점을 보완하고 더 높은 정확성을 보장한다는 장점을 가지고 있다.

본 논문에서 제안하는 시스템은 XML 문서를 변환하기 위한 XSLT를 자동으로 생성하여 XML 문서를 변환하는 모든 과정을 자동화 함으로써 문서 변환에 따르는 비용의 절감할 수 있을 것으로 기대된다.

1. 서 론

XML[1]은 W3C 표준으로써 현재 많은 어플리케이션을 통해 여러 분야에서 다양한 목적과 형식을 가지고 사용되고 있다.

이러한 XML 문서를 사용하는 어플리케이션의 내용 변경이 생기거나 문서 교환 시스템에서 양단간의 문서 양식이 틀린 경우와 같은 사용 목적의 변화가 생기면 그 목적에 따라 기존 XML 문서의 구조를 변경해야 한다.

이처럼 XML 문서의 재정의가 필요한 경우 새로운 양식에 맞는 XML 문서를 다시 작성하는 것보다 XSLT[2]를 이용한 기존 문서의 변환이 더욱 효율적이다. 이러한 XML 문서의 재사용은 지금까지 축적해온 데이터들을 재사용함으로 인해 많은 비용의 감소를 이루어 낼 수 있는 장점이 있다. 하지만 사용자 정의 태그를 지원하는 XML의 장점은 변환할 문서간의 엘리먼트가 의미하는 것이 정의하는 사람에 따라 다를 수 있어 자동으로 분석하고 변환함에 있어서 어려워 단점이 될 수 있다.

이런 문제점을 보완하기 위해 XML 문서의 구조 정보를 표현하는 DTD나 XML 스키마(Schema)를 사용한다. 이를 분석한 정보를 가지고 XSLT를 기술하여 XML 문서를 변환을 할 수 있다.

하지만 이런 경우 사용자가 변환 과정을 모두 처리함으로써 정확성은 높아지나 태그의 수나 변환할 문서의 양이 많을 경우 변환하는 작업에 소요될 시간과 비용 또한 높아지기 때문에 변환 작업을 자동화하는 것이 좋다.

기존에 제시된 XTGen[3]에서는 자동변환 생성기를 구성하는 과정에 사용자가 개입하여 엘리먼트간의 관계를 수동으로 처리 함으로써 대량의 문서 변환 시 많은 비용의 소요가 예상된다. XML 문서의 빠른 변환을 위한 XSLT 스크립트[4]는 문서간 단말 노드의 완벽한 1:1 대응만 허락 함으로써 변환 대상 문서간 엘리먼트 이름이 변화가 되었을 경우 변환에 어려움이 있다.

이러한 문제점들을 해결하기 위하여 본 논문에서는 다음과 같은 방법을 제안한다.

첫째, XML 문서 작성자간 같은 데이터를 다른 엘리먼트 이름을 사용하여 표현했을 경우를 JAWS[5]를 이용하여 태그 간 의미 관계를 분석한다.

둘째, 구조 분석을 위한 준비 단계로써 각 태그를 트리비얼 도큐먼트 인코딩(Trivial Document Encoding)[6]하여 값을 추출한다.

셋째, 인코딩한 값을 LCS(Longest Common Subsequence) Method[7]를 사용하여 문서간 구조의 변화를 분석한다.

그리고 마지막으로 각 단계에서 분석된 정보를 참조하여 각 엘리먼트간의 관계를 구분 짓고, XML 문서의 변환을 위한 XSLT를 자동 생성한다.

본 논문의 2장에서 위에 언급된 관련 연구를 소개하고, 3장 본론에서 본 논문에서 제안하는 전체 시스템 구성에 대해 논한 후 4장에서 XSLT Generator의 구성을 논한다. 5장에서 결론 및 향후 과제를 끝으로 논문을 마친다.

2. 관련 연구

2.1. XTGen

XTGen은 서로 다른 DTD 기반의 XML 문서를 변환하는 변환기를 자동 생성하는 시스템이다. 이 시스템은 XML 문서를 검증하고 DTD를 추출하는 컴포넌트(XDA), 검증된 DTD 문서 구조 정보 형식을 변경하는 컴포넌트(DDA), 문서 구조 정보를 보여주고 상호 변환을 위해 매핑 할 수 있도록 하는 컴포넌트(DIV), 두 문서간의 매핑 정보를 추출하는 컴포넌트(MDO), 추출된 매핑 정보를 바탕으로 2개의 XML 문서를 상호 변환하는 변환기를 생성하는 컴포넌트(TCG)로 구성된다. XTGen은 XML 문서를 자동 변환할 수 있는 변환기를 생성하여 변환기 생성의 비용 감소와 모듈화를 통한 쉬운 유지보수가 장점이다.

하지만 이 시스템은 변환기를 생성하는 과정 중 DIV 컴포넌트와 MDO 컴포넌트 사이에 유저가 엘리먼트간의 변환 관계를 기술해야 한다는 단점과 변환 대상이 되는 XML 문서의 작은 변화에도 새로운 변환기를 만들기 위하여 모든 작업을 다시 해야 하는 단점이 있다.

2.2. XML 문서의 빠른 변환을 위한 XSLT 스크립트

XML 문서의 변환하는 XSLT 스크립트를 생성하는 시스템이다. 어휘 및 구조 유사도를 이용한 빈도 지시자간의 대응관계를 생성하고, 이를 바탕으로 적은 수의 템플릿을 포함하는 XSLT 스크립트를 생성하여 보다 빠른 속도로 XML 문서를 변환 한다. 하지만 단말 노드간의 1:1 매칭이 되지 않는다면 문서 변환을 할 수 없다는 단점과 대용량의 문서를 처리한다면 효율성이 떨어진다는 단점이 있다.

3. 본 론

일반적인 XML 문서의 변환에서는 각 엘리먼트가 의미하는 것이 문서 작성자의 의도에 따라 각각 다르기 때문에 변환 과정에서 사용자가 변환 관계를 명확하게 구분 지어주어야 한다. 이러한 문제점으로 인해 XML 문서의 자동변환에는 어려움이 있어 이를 해결하기 위한 방법을 제시한다.

본 논문에서 제안하는 시스템은 변환 대상 문서간에 엘리먼트명의 완벽한 1:1 대응이 아니어도 JAWS을 이용해서 엘리먼트 사이의 관계를 분석하고 이를 바탕으로 문서 변환을 자동으로 처리하여 XSLT를 생성함으로써 기존에 제안된 시스템들의 단점을 최소화하였다.

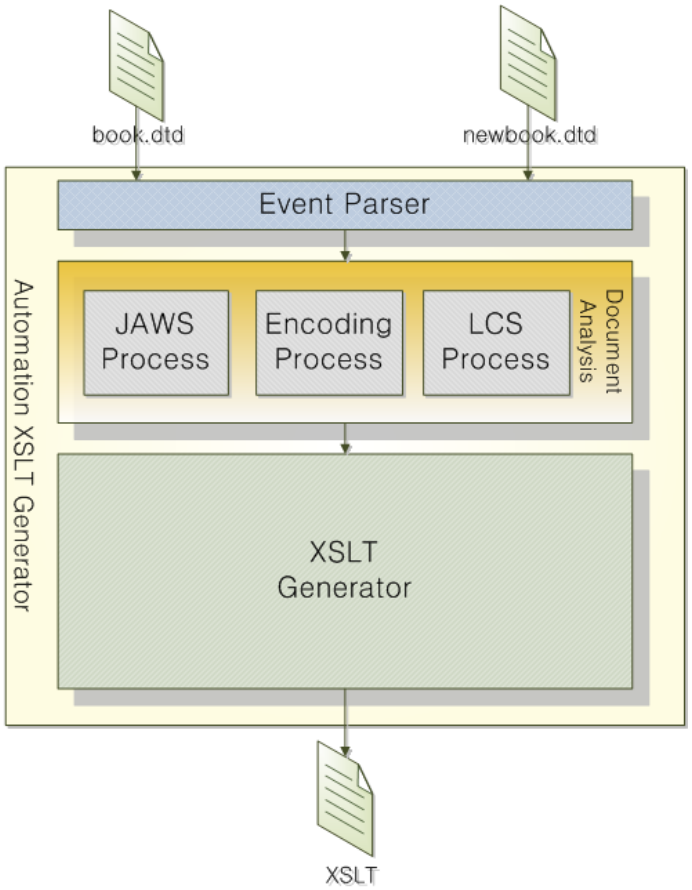

그림 1. 전체 시스템 구조도

그림 1에서 나타낸 이 시스템은 이벤트 파서가 DTD를 파싱하면서 태그명을 넘겨준다. 이 데이터를 이용해 3개의 프로세스를 통합한 문서 분석 작업(Document Analysis)을 거쳐 변환 규칙을 담은 XSLT를 자동으로 생성하게 된다.

이를 위한 각 프로세스에 대한 관련 연구를 다음 절에서 소개하고 이를 구현하기 위한 알고리즘을 pseudo-code를 이용하여 나타내었다.

아래 표 1은 변환 과정에 사용될 XML문서와 구조 정보를 담고 있는 DTD 문서의 내용이다.

표 1. 문서 구조를 표현한 DTD 문서와 XML 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT xml ((book))>
<!ELEMENT book (#PCDATA | author | publisher | date)*>
```

```
<!ELEMENT author (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT date (#PCDATA)>
```

(A)book.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xml SYSTEM "book.dtd">
<xml>
   <book> 장미의 시대
    <author> 움베르트 에코 </author>
    <publisher> 열린책들 </publisher>
    <date> 2002-10-10 </date>
   </book>
</xml>
```

(B)DTD1.dtd에 유효한 book.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT xml ((publisher))>
<!ELEMENT publisher (#PCDATA | writer)*>
<!ELEMENT writer (#PCDATA | book | date)*>
<!ELEMENT book (#PCDATA)>
<!ELEMENT date (#PCDATA)>
```

(C)newbook.dtd

3.1. JAWS(Java API for WordNet Searching)

JAWS은 브렛 스펠(Brett Spell, Southern Methodist University)이 만든 워드넷(WordNet)[8]을 이용한 JAVA API이다. 워드넷(WordNet)은 조지 A. 밀러 교수(George A. Miller, Princeton University)가 제안한 시스템으로써 영어 단어를 명사(noun), 동사(verb), 형용사(adjective), 부사(adverb)와 같은 품사 별로 구분지어 정의하고, 입력된 단어에 대한 동의어(Synonym), 반의어(Antonym), 상의어(Hypernym), 하의어(Hyponym), 부분어(Meronym), 전체어(Holonym) 등과 같은 관계들의 집합을 나타내는 SynSet을 이용하여 의미적 분석에 이용할 수 있기 때문에 온톨로지 시스템이나 XML 문서의 유사도 분석에 이용되고 있다.

그러나 워드넷(WordNet)은 웹으로 결과를 받아오기 때문에 워드넷(WordNet) DB를 로컬에 두고 사용자가 입력한 단어를 가지고 검색하여 값을 돌려주는 기능을 구현한 JAWS를 사용하여 본 논문에서 제안하는 시스템에서 이용하였다.

JAWS의 장점은 워드넷(WordNet) 기반의 JAVA 어플리케이션을 구현하기 쉽도록 지원해 줌과 더불어 JAVA의 장점인 멀티 플랫폼 지원을 그대로 가지는 장점이 있다.

본 논문에서 제안하는 시스템에서는 JAWS를 이용하여 두 문서의 각 엘리먼트의 의미를 분석하고 엘리먼트 간 관계를 정의한다.

3.2. Trivial Document Encoding

트리비얼 도큐먼트 인코딩(Trivial Document Encoding)은 XML 문서에서 태그가 나타나는 순차대로 인코딩 하여 값을 주는 방식으로, 한 엘리먼트의 시작 태그는 양수 값을 가지고, 종료 태그는 음수 값을 가진다.

트리비얼 도큐먼트 인코딩(Trivial Document Encoding)은 WSL(Without Structural Loss)로써, 인코딩 후에도 XML 문서의 구조를 표현 할 수 있으며, 특별한 파싱(Parsing) 기법이 필요치 않아 시스템 자원의 확보에 이점을 가지는 장점이 있다.

아래 표 2는 트리비얼 도큐먼트 인코딩의 알고리즘을 pseudo-code로 나타낸 것이다.

표 2. Trivial Document Encoding 알고리즘

```
입력 : 태그이름
출력 : 인코딩된 값이 저장되어 있는 배열

encoding(태그명)
{
      태그명 저장용    배열1;
      인코딩 값 저장용 배열2;
      var a = 1;

      while(tagname != null)
      {
            if(tagname == STARTTAG){
                  배열1[a]에 태그명 저장;
                  배열2[a]에 변수1 저장;
                  a++;
            }else(tagname == ENDTAG){
                  var tmp;
                  배열1[a]에 태그명 저장;
                  태그명에서 '/'삭제;
                  tmp = 배열1에서 태그명으로 인덱스 검색;
                  배열2[tmp]의 값을 배열2[a]에 음수로 저장;
                  a++;
            }
}
```

3.3. LCS(Longest Common Subsequence) Method

LCS Method는 주어진 두 개의 문자열에서 가장 긴 공통의 서브시퀀스를 찾아내는 방법으로, 주어진 문자열 A = α1α2α3……αn, B = β1β2β3……βm, C = ɣ1ɣ2ɣ3……ɣp 가 있을 때, 문자열 C를 포함하는 문자열 A에서 n - p 캐릭터가 지워질 수 있고, 문자열 C를 포함하는 문자열 B에서 m - p 캐릭터가 지워질 수 있다면, 문자열 C는 문자열 A와 B의 공통 서브시퀀스가 된다.

위와 같은 방법을 이용해 X라는 문자열과 Y라는 문자열이 입력으로 들어왔을 때, LCS Method를 이용해 서브시퀀스를 산출하면 아래 표 4의 Z와 같다.

표 3. LCS Method 예

Old String *(X)*	New String *(Y)*	LCS Method *(Z)*
AXCYDWEABE	ABCDE	A~~X~~B~~C~~~~Y~~D~~W~~E~~A~~B~~E~~

위의 방법을 사용하여 비교 문서간 인코딩 된 값을 가지고 공통의 서브시퀀스를 제외한 나머지 부분은 결국 어떤 엘리먼트의 위치가 달라졌는지 알 수 있다. LCS Method를 본 논문에서 제안하는 XLST 스크립트 자동 생성 시스템에 적용하기 위한 알고리즘을 표 5에 pseudo-code로 표현하였다.

표 4. LSC Method 알고리즘

```
입력 : DTD1.dtd가 인코딩된 값이 저장된 배열1,
       DTD2.dtd가 인코딩된 값이 저장된 배열2
출력 : LCS Method가 적용된 배열3

LCS(배열1, 배열2)
{
    LCS값 저장용 배열3;
    var  x = 1;
    var  y = 1;
    var  z = 1;

    while(배열1과 배열2의 끝이 아니면)
    {
        if(배열1[x] == 배열2[y]){
            배열3[z].name에 배열1[x]의 이름 저장;
            배열3[z].flag에 0 저장;
            x++; y++; z++;
        } else{
            if( x == y ){
                배열3[z].name에 배열1[x] 값 저장;
                배열3[z].flag에 1 저장;
                x++; z++;
            }else(  i < j ){
                배열3[z].name에 배열2[y] 값 저장;
                배열3[z].flag에 2 저장;
                j++; z++;
            }
        }
    }
}
```

4. Automation XSLT Generator

XSLT는 W3C 표준으로 XML 문서를 다른 XML 문서나 HTML 또는 그 이외의 문서로 변환 할 수 있도록 해주는 언어이다. XML은 데이터의 구조적 표현을 통해 많은 이점을 주었지만 데이터의 교환이 있을 경우에 양쪽 모두 동일한 DTD나 XML 스키마를 사용해야 한다는 제약이 있었다. 이 문제를 해결하기 위한 XLST는 양쪽의 DTD나 XML 스키마가 달라도 변환을 통해 교환에 필요한 양식을 맞출 수 있다.

본 논문에서 제안하는 시스템에서는 이러한 XSLT의 장점을 이용하여 문서 분석을 통한 데이터를 가지고 XML 문서 변환에 필요한 XSLT를 자동으로 생성한다.

Automation XSLT Generator의 과정은 다음과 같다.

첫째, 이벤트 파서가 변환에 필요한 두 DTD 문서를 입력 받아 각 태그가 발생하는 순서대로 태그명 저장한 데이터를 넘겨준다.

둘째, 이를 넘겨 받은 Document Analysis 과정에서는 이벤트 파서가 넘겨준 데이터를 기반으로 JAWS 프로세스와 트리비얼 도큐먼트 인코딩을 하여 LCS Method를 실행. 두 문서간의 구조가 변화된 부분을 찾는다.

먼저 JAWS를 통해 두 문서의 모든 엘리먼트간의 관계를 분석한다.

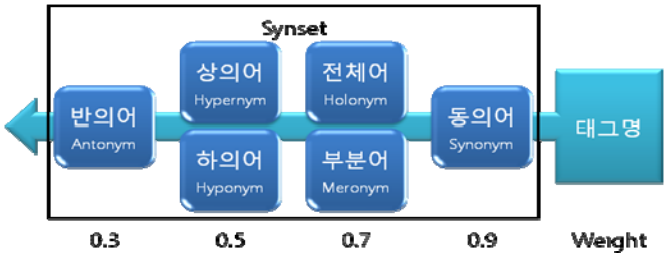

그림 2. JAWS를 통해 넘겨 받은 Synset의 Weight 값

그림 2는 JAWS를 통해 엘리먼트명을 검색했을 경우, 넘어오는 Synset을 이용해 각 엘리먼트명의 관계를 구분 짓기 위한 Weight의 설정 값을 표현한다.

단, JAWS를 이용하여 분석하기 이전에 비교할 두 엘리먼트명이 철자까지 100% 동일한 경우 Weight 값을 분석 과정 없이 바로 1.0으로 준다.

표 1의 (A)와 (C)의 엘리먼트 간 관계를 JAWS를 통해 분석하면 다음 표 5과 같은 결과가 나온다.

표 5. 두 문서간 엘리먼트명 비교표

book.dtd / sunbook.dtd	xml	book	author	publisher	date
xml	1.0	0.0	0.0	0.0	0.0
publisher	0.0	0.0	0.0	1.0	0.0
writer	0.0	0.0	0.9	0.0	0.0
book	0.0	1.0	0.0	0.0	0.0
date	0.0	0.0	0.0	0.0	1.0

다음으로 표 1의 (A)와 (C)를 트리비얼 도큐먼트 인코딩을 하여 나오는 결과 값을 아래 표 6를 통하여 나타내었다.

표 6. DTD 문서를 Trivial Document Encoding한 결과

Sequence	Encoding	Value
S_0	δ_d(<xml>)	1
S_1	δ_d(<book>)	2
S_2	δ_d(<author>)	3
S_3	δ_d(</author>)	-3
S_4	δ_d(<publisher>)	4
S_5	δ_d(</publisher >)	-4
S_6	δ_d(<date>)	5
S_7	δ_d(</date>)	-5
S_8	δ_d(</book>)	-2
S_9	δ_d(</xml>)	-1

(A) book.dtd

Sequence	Encoding	Value
S_0	δ_d(<xml>)	1
S_1	δ_d(<publisher>)	2
S_2	δ_d(<writer>)	3
S_3	δ_d(<book>)	4
S_4	δ_d(</book>)	-4
S_5	δ_d(<date>)	5
S_6	δ_d(</date >)	-5
S_7	δ_d(</writer>)	-3
S_8	δ_d(</publisher>)	-2
S_9	δ_d(</xml>)	-1

(C) newbook.dtd

위의 표 6에서 나타낸 두 DTD 문서를 트리비얼 도큐먼트 인코딩(Trivial Document Encoding)한 결과 값을 LCS Method를 통해 구조 분석을 한 결과를 아래 표 7에서 나타내었다.

표 7. LCS Method를 이용한 표 1의 (A), (C) 구조 분석

(A)	(C)
1,2,3,-3,4,-4,5,-5,-2,-1	1,2,3,4,-4,5,-5,-3,-2,-1

(A)와 (C)의 LCS Method 결과
1,2,3,~~-3~~,4,-4,5,-5,<u>-3</u>,-2,-1

표 1의 (A)와 (C)를 트리비얼 도큐먼트 인코딩(Trivial Document Encoding)하여 LCS Method로 값을 구해본 결과는 -3이 두 문서 사이의 공통 서브시퀀스가 아닌 것으로 나온다. 이를 통해 -3으로 인코딩 된 (A)의 엘리먼트 'author'의 종료 태그의 위치가 바뀌었다는 것을 유추할 수 있다.

셋째, 마지막 과정으로써 위 두 단계를 거쳐 나온 정보를 기반으로 XSLT Generator가 문서 변환을 위한 XSLT를 자동으로 생성한다.

우선 (A)와 (C)의 트리비얼 도큐먼트 인코딩한 결과를 가지고 같은 인코딩 값에 대한 태그명을 비교하여 같다면 그대로 매칭시키고, 만약 같은 인코딩 값에 대한 각 문서의 태그명이 다르다면 문서 내의 모든 태그와 비교하여 똑같은 태그와 매칭시킨다. 모든 태그를 조사했음에도 불구하고 동일한 태그가 없을 경우 JAWS프로세스의 결과값을 통해 가장 높은 Weight 값을 가지는 태그와 매칭시켜 XSLT를 생성한다.

5. 결론 및 향후 연구과제

데이터를 표현하는데 있어서 다양한 용도로 사용되고 있는 XML은 사용자 태그 선언으로 인한 태그 간 관계를 기계적으로 판별하기 어렵기 때문에 자동 변환이 어렵다는 단점을 가지고 있다. 본 논문에서는 이러한 문제점을 해결한 XML 문서 자동 변환 시스템을 구현하기 위하여 JAWS와 트리비얼 도큐먼트 인코딩을 통한 LCS Method를 이용하여 조금 더 정확성을 높인 자동 변환 시스템을 제안한다. 이러한 방법을 통하여 기존 시스템의 단점을 보안하고 제한적일 수 밖에 없는 XML 문서의 자동 변환의 폭을 넓힐 수 있는 방법을 제안하였다. 그러나 변환 결과의 정확성을 높이기 위하여 워드넷(WordNet)을 이용한 엘리먼트 이름 사이의 관계를 분석하는데 소요되는 시간과 각 프로세스 별 처리 시간으로 인하여 많은 시간이 소요되므로 이를 줄이는 연구가 추후에 필요하다.

참고 문헌

[1] W3C. Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, October 6, 2000. Available at http://www.w3.org/TR/REC-xml(June 10, 2002).

[2] W3C. XSLT Requirements Version 2.0, W3C Recommendation, 14 February 2001, Available at http://www.w3.org/TR/xslt20req

[3] 심민석, 유대승, 엄정섭, 강만모, 이명재, "XEGen : XML 변환기 생성을 위한 컴포넌트 기반 시스템", 한국정보과학회, 한국정보과학회 학술발표논문집 한국정보과학회 2001년도 봄 학술발표논문집 제28권 제1호(A), pp. 310 ~ 312, 2001. 4.

[4] 신동훈, 이경호, "XML 문서의 빠른 변환을 위한 XSLT 스크립트", 한국정보과학회, 정보과학회논문지 : 컴퓨팅의 실제 제11권 제6호, pp. 538 ~ 550, 2005. 12.

[5] Brett Spell, JAWS(Java API for WordNet Searching), February 13, 2008.Available at http://engr.smu.edu/~tspell/

[6]Sergio Flesca, Giuseppe Manco, Elio Masciari, Luigi Pontieri, and Andrea Pugliese, “Fast Detection of XML Structural Similarity”, IEEE Computer Society Vol. 17, No. 2, pp. 160-175 FEBRUARY 2005.
[7] Aho, A., Hirschberg, D., and Ullman, “Bounds on the complexity of the longest common subsequence problem”. J. ACM 23, 1 (Jan. 1976), 1-12.
[8] C.Fellbaum, WordNet : An Electronic Lexical Database, Cambridge : MIT Press. 1998.