Journal of Korea Multimedia Society Vol. 11, No. 8, August 2008(pp. 1043-1050)

XML 문서의 구조와 내용을 고려한 유사도 측정

김 우 생†

요 약

XML(Extensible Markup Language)은 인터넷 상에서 데이터 표현과 교환을 위한 표준으로 자리 잡고 있다. 웹의 발전과 함께 XML 문서들이 정보 검색, 문서 관리, 데이터 마이닝 등의 응용에서 폭 넓게 사용되면서 구조적으로 정보가 풍부한 이러한 문서들을 자동으로 처리하고 검색하는 기술들이 요구되고 있다. 본 연구에서는 XML 문서의 구조와 내용을 고려하여 유사한 문서들을 검색하는 새로운 방법을 제안한다. XML 문서의 구조적 유사성은 간단한 스트링 매칭 기법으로 찾고, 문서 내용의 유사성은 문서 요소(element)들의 이름과 경로를 고려한 가중치를 통해 찾는 방법으로 전체의 시간 복잡도는 비교되는 두 문서의 크기에 선형적으로 비례한다.

Similarity Measure based on XML Document's Structure and Contents

Woosaeng Kim†

ABSTRACT

XML has become a standard for data representation and exchange on the Internet. With a large number of XML documents on the Web, there is an increasing need to automatically process those structurally rich documents for information retrieval, document management, and data mining applications. In this paper, we propose a new method to measure the similarity between XML documents by considering their structures and contents. The similarity of document's structure is found by a simple string matching technique and that of document's contents is found by weights taking into account of the names and positions of elements. The overall algorithm runs in time that is linear in the combined size of the two documents involved in comparison evaluation.

Key words: XML, similarity(유사도)

1. 서 론

인터넷의 성장으로 웹 상의 정보를 손쉽게 접근할 수 있게 되었고 이로 인하여 많은 새로운 정보들이 만들어지고 유포되고 있다. 웹의 발전과 함께 XML (Extensible Markup Language)과 같이 구조적으로 정보가 풍부한 문서들이 정보 검색, 정보 교환, 문서 관리, 데이터 마이닝 등의 응용에서 폭넓게 사용되고 있다. XML은 웹 상에서 데이터 표현과 교환의 수단으로 제안된 표준 마크업 언어로 HTML과는 달리 사용자가 요소(element)를 자유롭게 정의할 수 있어 요소가 데이터 정보 자체뿐 아니라 데이터의 의미도 표현할 수 있다. 일반적인 XML 요소는 시작 태그(tag), 문자열 및 종료 태그로 구성된다. XML은 기존 문서의 비순차적인 구조와는 다르게 이러한 요소들의 순차적이고 계층적인 구조로 이루어져 있기 때문에, XML 문서를 트리로 취급하고 문서의 요소들을 노드들로 변환하면 XML 문서를 대응하는 트리 구조

※ 교신저자(Corresponding Author) : 김우생, 주소 : 서울시 노원구 월계동 447-1(139-701) 전화: 02)940-5217, FAX: 02)909-0998, E-mail : kwsrain@kw.ac.kr
접수일 : 2007년 10월 16일, 완료일 : 2008년 5월 19일

† 정회원, 광운대학교 컴퓨터공학부 교수
※ 본 논문은 2007년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음

로 표현할 수 있다[1].

XML의 폭발적인 증가와 함께 이러한 문서들을 자동으로 처리하고 검색하고 이를 통해 문서들을 군집화하는 기술들이 요구되고 있다. 유사한 문서들을 군집화할 경우 질의 처리 과정에서 불필요한 데이터를 제외함으로써 결과적으로 탐색 공간의 크기를 줄일 수 있어 보다 효율적인 질의 처리를 가능하게 한다. XML 문서를 검색하거나 군집화 하기 위해서는 XML 문서간의 유사성을 비교하는 기술이 필요하다. XML 문서에 얼마나 공통된 요소들이 포함되어 있는가를 바탕으로 문서의 유사성을 찾는 방법이 연구되었다. 이러한 방법들에서는 동일한 요소의 이름뿐 아니라 동의어 집합이나 유사어 확장을 통해 유사한 요소의 이름들도 고려하는 연구와[2,3], 요소의 출현 빈도도 고려하여 요소의 가중치를 계산하는 방법도 연구되고 있다[4,5].

XML과 같이 트리에 기반한 문서들에 있어서 노드의 삽입, 삭제, 그리고 변경 등의 편집 연산(edit operation)을 통해 하나의 트리를 다른 트리로 바꾸는 방법으로 문서들 간의 유사성을 평가하는 연구도 있다[6,7]. 선택적이고 반복적인 요소들을 가질 수 있는 XML 문서의 특징을 고려하여 edit distance라는 방법을 통해 하나의 트리를 다른 트리로 바꾸는 방법에 대한 연구가 있고[8], edit distance 방법에 XML 요소들의 의미를 추가로 고려하여 XML 문서의 유사성을 계산하는 방법도 연구되었다[9].

XML 요소의 의미와 구조 정보를 반영하여 문서의 유사성을 찾는 방법들도 진행되고 있다. XML 문서에서 요소들을 추출하여 두 문서의 요소들에 대하여 유사 이름과 경로를 고려한 가중치를 부여하는 방법으로 유사도를 계산하는 연구가 있다[10,11]. 서로 다른 XML 문서들이 트리의 경로를 얼마나 공유하는가의 정도에 따라 문서간의 유사성을 판별하는 연구가 있다. XML 요소를 시소로스(thesaurus)를 이용하여 유사어와 합성어로 구성된 확장 요소 벡터로 확장하고 유사 행렬을 구축하여 요소들의 유사성을 판별하고, 오토마타를 이용하여 XML의 구조를 최소화한 후 변형된 순차 패턴 마이닝 알고리즘을 통해 문서간의 최대 유사 시퀀스를 추출하였다[12]. 반면 XML 문서에 순차 패턴 마이닝 알고리즘을 적용하여 빈발 경로를 찾고 이를 통해 XML 문서 군집화를 시도한 연구도 있다[13].

이산 푸리에 변환을 사용하여 XML 문서의 유사성을 비교하는 연구가 있다. 이러한 방법에서는 문서의 데이터들을 먼저 숫자들로 변환한 후에 푸리에 변환을 적용하여 주파수 데이터로 변환시킨다. 두 문서간의 거리는 두 주파수 데이터 크기의 차이로서 계산이 된다. XML 문서 요소들의 계층적 구조 관계를 함수로 변환하여 문서의 구조적 유사성을 비교하는 연구가 있고[14,15], XML 문서에 대응하는 트리를 전위 탐색(preorder search) 하면서 요소들을 부호화 함수를 통해 유일한 신호로 변환하여 문서의 구조뿐 아니라 내용의 유사성도 비교하는 연구가 있다[16,17].

본 논문은 XML 문서간의 유사성을 측정하기 위하여 문서의 구조와 내용을 고려한 새로운 방법을 제안한다. 본 연구에서는 문서에 대응 하는 트리를 순회하면서 노드들의 위치에 기반한 스트링을 추출하여 이들을 비교하는 방법으로 문서간의 구조적 유사성을 검사한다. 본 연구에서는 또한 문서 요소들의 이름과 문서 계층 구조내의 요소들의 위치를 고려한 적절한 가중치를 할당하는 방법으로 문서간의 내용적 유사성을 검사한다. 본 연구에서 제안하는 XML 문서간의 유사성을 측정하는 방법의 시간 복잡도는 비교되는 두 문서의 크기에 선형적으로 비례한다.

논문의 구성은 다음과 같다. 2장에서는 XML 문서의 유사한 구조와 유사한 내용을 찾는 방법을 설명한다. 3장에서는 XML 문서 유사도를 계산하는 알고리즘을 설명하며 몇 가지 실험과 다른 방법들과의 비교를 통해 제안한 방법이 효율적이라는 것을 보여준다. 마지막으로 4장에서 결론을 맺는다.

2. XML 문서의 유사 구조와 유사 내용

XML 문서의 검색은 입력 XML 문서와 유사한 XML 문서들을 찾는 과정이다. XML 문서간의 유사도를 계산하는 일반적인 척도는 없지만 두 문서의 구조와 내용이 유사하다면 유사도가 있는 문서이다. 본 연구에서 두 문서의 구조가 유사하다는 것은 두 문서의 구조가 같거나 두 문서가 서로 간에 포함 관계를 갖는 경우를 의미하며, 두 문서의 내용이 유사하다는 것은 문서의 대응하는 요소들의 이름이 같거나 유사한 것을 의미한다. 예를 들어, 그림 1(a)와 유사한 문서는 같은 계층 구조와 유사한 요소 이름들을

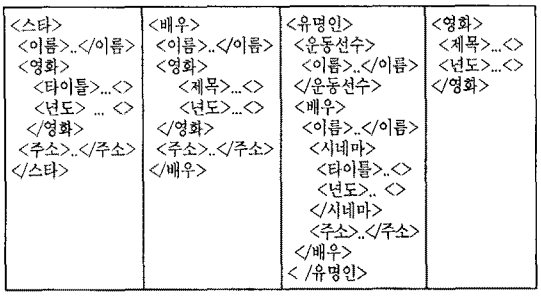


그림 1. XML 문서

갖는 그림 1(b)이다. 그러나 그림 1(c)도 그림 1(a)와 같은 계층 구조를 포함하고 있으며 동시에 유사한 요소 이름들을 갖기 때문에 유사한 문서이다. 반면 그림 1(d)는 그림 1(a)의 일부 계층 구조와 같으며 유사한 요소들을 갖고 있기에 역시 유사한 문서이다. 따라서 본 논문에서는 그림 1(a)와 같은 XML 문서가 주어졌을 때 그림 1(b) 뿐 아니라 그림 1(c)나 그림 1(d)와 같이 구조와 내용이 유사한 문서들을 효율적으로 찾는 방법을 제안한다.

그림 2는 본 논문에서 제안하는 XML 문서 검색 시스템 흐름도로 다음과 같은 단계들을 거친다. 첫 번째 단계에서는 XML 문서를 대응하는 트리 구조로 바꾸고 트리의 노드들을 순회하여 노드의 위치에 기반한 스트링을 추출한다. 두 번째 단계에서는 비교되는 두 문서의 스트링들을 스트링 매칭 기법을 적용해 두 문서의 구조가 유사한가를 검사한다. 마지막 단계에서는 비교되는 두 문서의 대응하는 요소들의 이름과 경로 등을 고려한 후 가중치를 부여하는 방법으로 두 문서의 내용이 어느 정도 유사한가를 계산한다.

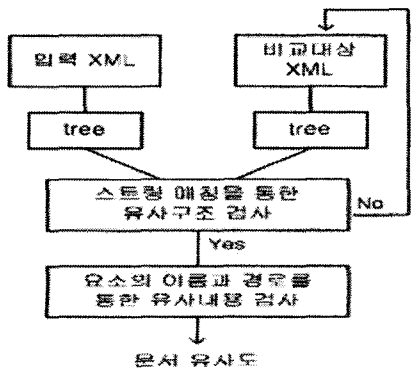


그림 2. XML 문서 검색 시스템 흐름도

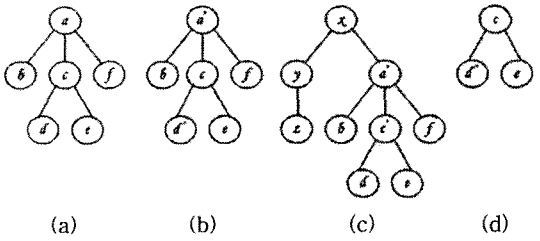


그림 3. XML 문서에 대응하는 트리

2.1 유사 구조를 찾는 방법

XML 문서는 정의에 의하여 요소들이 정렬되어 있으므로 본 논문에 언급하는 트리는 정렬된 트리(ordered tree)를 의미한다. 그림 1(a), 그림 1(b), 그림 1(c), 그림 1(d)의 문서를 트리로 변환하면 각각 그림 3(a), 그림 3(b), 그림 3(c), 그림 3(d)가 된다. 단, 그림 1의 요소 이름들은 이해를 돕기 위해 알파벳으로 재 명명하였다. 또한 그림 3에서 예를 들어, 'a'' 노드는 'a' 노드와 유사한 이름을 가진 노드를 의미한다. 그림 1(a)와 그림 1(b)의 문서는 구조가 같기 때문에 그림 3(a)와 그림 3(b)의 트리 구조도 같음을 알 수 있다. 그림 1(c)의 문서는 그림 1(a)의 문서를 포함하기 때문에, 그림 3(c)의 트리도 그림 3(a)의 트리를 포함하고 있음을 알 수 있다. 즉, 그림 3(c)에서 노드 'a''를 루트(root)로 둔 부분트리(subtree)는 그림 3(a)와 같다. 반면 그림 1(d)의 문서는 그림 1(a) 문서에 포함되기 때문에, 그림 3(d)의 트리는 그림 3(a)의 트리에 포함되는 것을 알 수 있다.

[정의 1] 구조 유사도

XML 문서 d_1과 d_2가 구조 유사도를 가진다는 의미는 d_1의 트리를 t_1, d_2의 트리를 t_2라고 할 때 $t_1 = t_2$ 또는 $t_1 \subset t_2$ 또는 $t_2 \subset t_1$인 경우이다. 여기서 $t_n \subset t_m$은 트리 t_n이 트리 t_m의 부분트리라는 의미이며, $t_n = t_m$는 $t_n \subset t_m$이며 동시에 $t_m \subset t_n$인 경우이다.

XML 문서의 계층적 구조는 트리의 구조에서도 반영된다. 예를 들어, 그림 1(a) 문서에서 '이름'과 '영화' 요소는 '스타' 요소에 내포되어 있는데, 그림 3(a) 트리에서는 'b'와 'c' 노드는 'a' 노드의 자식 노드들로 구성되어 있음을 알 수 있다. 따라서 XML 문서 내 요소들 간의 계층적 구조는 대응하는 트리의 레벨(level)로 표현할 수 있다. 본 논문에서는 XML

```
EDFS {
① 스택에 루트 노드 삽입
② 스택 맨 위 노드 출력;
   if 노드가 자식들이 있으면 스택에
     오른쪽 자식 순으로 삽입;
   else 노드를 스택에서 삭제하고 트리의
        대응되는 노드에 삭제 플래그 표시;
③ if 스택이 비었으면 종료;
  else ②번 수행;}
```

그림 4. EDFS 의사 코드

문서의 이러한 계층적 구조를 대응하는 트리를 통해 구하기 위해 깊이 우선 탐색(DFS: Depth First Search)을 확장한 확장된 깊이 우선 탐색(EDFS: Extended Depth First Search)을 제안한다. 그림 4는 EDFS의 의사 코드(pseudo code)다.

그림 3(a)와 같은 트리를 DFS로 방문한 순서는 *'abcdef'*이나, EDFS로 방문한 순서는 *'abcdecfa'*이다. 이와 같이 일부 노드가 중복하여 나타나는 이유는 대응하는 XML 문서에서 시작 요소뿐 아니라 종료 요소도 표현하여 문서의 계층적 구조를 나타내기 위함이다. 그림 1(a) 문서 내 요소들 간의 계층적 구조를 최상위 요소 '스타'를 레벨 0로 시작하여 표현하면 '0,1,1,2,2,1,1,0' 숫자열이 된다. 그림 3(a) 트리를 루트로부터 EDFS로 방문하며 노드의 레벨을 출력하면 동일한 '0,1,1,2,2,1,1,0' 숫자열이 된다. 트리의 전체 노드의 수가 n 일 때 EDFS에 걸리는 시간은 기존 DFS에 걸리는 시간과 같기 때문에 시간 복잡도는 $O(n)$이다.

[정의 2] EDFS 숫자열

트리를 루트로부터 EDFS로 방문한 노드의 레벨 순서가 EDFS 숫자열이다.

두 XML 문서의 계층적 구조가 같은 경우는 같은 EDFS 숫자열을 갖지만 두 문서의 계층적 구조가 다는 경우는 다른 EDFS 숫자열을 갖는다. 그러나 두 XML 문서가 서로 간에 포함 관계에 있을 때 EDFS 숫자열은 서로 다르나 EDFS 숫자열에서 이웃하는 숫자간의 차이열은 서로 간에 포함 관계를 갖는다.

[정의 3] EDFS 차이열

EDFS 숫자열을 $\alpha_1,\alpha_2,...,\alpha_n$라고 할 때, EDFS 차이열은 $\alpha_2-\alpha_1$, $\alpha_3-\alpha_2$,..., $\alpha_n-\alpha_{n-1}$의 숫자열이다.

예를 들어, 그림 3(a)의 EDFS 숫자열은 '0,1,1,2,2,1,1,0'이고 그림 3(c)의 EDFS 숫자열은 '0,1,2,1,1,2,2,3,3,2,2,1,0'로 서로 다르다. 반면, 그림 3(a)의 EDFS 차이열은 '1,0,1,0,1,0,1'이고 그림 3(c)의 EDFS 차이열은 '1,1,1,0,1,0,1,0,1,0,1,1'이다. 즉, 그림 3(c)의 EDFS 차이열이 그림 3(a)의 EDFS 차이열을 포함하고 있음을 알 수 있다. 마찬가지로 그림 3(d)의 EDFS 차이열은 '1,0,1'로 그림 3(a)의 EDFS 차이열 '1,0,1,0,1,0,1'에 포함되어 있음을 알 수 있다. 이러한 EDFS 차이열간의 포함관계는 스트링 매칭 문제로 볼 수 있다. 비교 대상의 EDFS 차이열의 길이가 n이고 비교 할 EDFS 차이열의 길이가 m일 때 패턴 매칭에 걸리는 시간 복잡도는 $O(n+m)$이다[18].

[정리 1] 두 XML 문서가 구조 유사도를 가지면 대응하는 두 트리의 EDFS 차이열은 서로 간에 포함 관계를 갖는다.

[증명] 두 XML 문서 d_1과 d_2의 대응하는 트리 t_1과 t_2의 EDFS 숫자열을 각각 $\alpha_1,\alpha_2,...,\alpha_n$과 $\beta_1,\beta_2,...,\beta_m$이라고 가정한다. (i) 만약 $t_1=t_2$이면 $\alpha_1=\beta_1$, $\alpha_2=\beta_2$,..., $\alpha_n=\beta_m$이기 때문에 $\alpha_2-\alpha_1=\beta_2-\beta_1$, $\alpha_3-\alpha_2=\beta_3-\beta_2$, ..., $\alpha_n-\alpha_{n-1}=\beta_m-\beta_{m-1}$이다. (ii) 만약 $t_1\subset t_2$ 이면 α_i에 대응하는 하나의 숫자를 β_i라고 가정 할 때 $\beta_{t+1}=\beta_t+(\alpha_2-\alpha_1)$, $\beta_{t+2}=\beta_{t+1}+(\alpha_3-\alpha_2)$, . . . , $\beta_{t+n-1}=\beta_{t+n-2}+(\alpha_n-\alpha_{n-1})$이기 때문에 $\beta_{t+1}-\beta_t=\alpha_2-\alpha_1$, $\beta_{t+2}-\beta_{t+1}=\alpha_3-\alpha_2$,..., $\beta_{t+n-1}-\beta_{t+n-2}=\alpha_n-\alpha_{n-1}$이다. (iii) 만약 $t_2\subset t_1$ 이면 β_i에 대응하는 하나의 숫자를 α_t라고 가정 할 때 $\alpha_{t+1}=\alpha_t+(\beta_2-\beta_1)$, $\alpha_{t+2}=\alpha_{t+1}+(\beta_3-\beta_2)$,...,$\alpha_{t+n-1}=\alpha_{t+n-2}+(\beta_n-\beta_{n-1})$이기 때문에 $\alpha_{t+1}-\alpha_t=\beta_2-\beta_1$,$\alpha_{t+2}-\alpha_{t+1}=\beta_3-\beta_2$,...,$\alpha_{t+n-1}-\alpha_{t+n-2}=\beta_n-\beta_{n-1}$이다.

2.2 유사 내용을 찾는 방법

두 XML 문서가 구조 유사도를 가질 때 두 문서의 내용도 유사한가를 검사하기 위하여 본 연구에서는 변화된 두 트리에서 대응하는 노드 이름의 유사성뿐 아니라 노드의 상대적 경로 등을 고려하여 가중치를 주는 방법을 사용한다. 먼저 노드 이름의 유사성은 어휘의 명사, 동사, 형용사적인 사용자 동의어 집합이나 WordNet과 같은 시소로스를 찾아주는 기능을 이용하여 구할 수 있다[19]. 예를 들어, 입력 문서의 한 노드 이름이 '배우'이고 비교 문서의 대응하는 노드 이름이 '스타'로 서로 다른 경우에도 시스템은 '스

타'의 동의어들을 WordNet과 같은 유사어 데이터베이스를 통해 추출한 후 두 단어의 일치하는 정도에 따라 0에서 1사이의 적절한 가중치를 할당하게 된다. XML 문서의 특성상 비교하는 어떤 두 노드가 내용적으로 다를 경우 두 노드의 자식들 간에도 내용이 유사하지 않다. 따라서 본 연구에서는 비교하는 두 노드가 내용적으로 다를 경우는 자식 노드들은 더 이상 고려하지 않는다.

대응하는 노드 이름이 같거나 유사하더라도 트리 내에서의 노드 위치에 따라 가중치는 달라질 수 있다. 예를 들어, 그림 3(a)의 'f' 노드는 내용면에서 그림 3(c)의 'f' 노드 보다는 그림 3(b)의 'f' 노드와 더 유사하다. 이것은 유사한 문서를 검색 할 때에 노드에 이르는 경로가 짧을수록 더 높은 비중을 갖기 때문이다[10,12]. 또한 대응하는 노드 이름이 유사하고 같은 경로에 있더라도 대응하는 노드의 동기(sibling)들의 숫자에 따라서 가중치는 다르게 평가될 수 있다. 즉, 대응하는 노드의 동기들이 많이 있는 경우보다는 동기들이 적은 경우에, 대응하는 노드의 가중치는 더 높은 비중을 갖게 된다. 따라서 본 연구에서는 대응하는 노드의 경로와 동기들의 많고 적음을 가중치에 반영하기 위해 부모 노드가 n ($n \geq 2$) 개의 자식 노드가 있다면 자식 노드 가중치는 부모 노드 가중치를 자식 노드의 수로 나눈 즉, 부모 노드 가중치/n으로 한다. 단, 자식이 단지 하나일 경우도 경로를 고려하여 자식 노드 가중치는 부모 노드 가중치/2로 한다. 예를 들어, 그림 3의 각 노드에 대한 경로와 동기들을 고려한 가중치는 루트 노드의 가중치를 1이라고 가정 할 때 그림 5에 나와 있다.

트리에서 모든 노드들의 가중치 합이 1이 되도록 정규화된 가중치를 구하기 위해서는 먼저 넓이 우선 탐색(BFS: Breadth First Search)을 통해 각 노드의 경로와 동기들을 고려한 가중치를 구한 후 다시 BFS를 통해 전체 노드 가중치로 나누면 된다. 예를 들어,

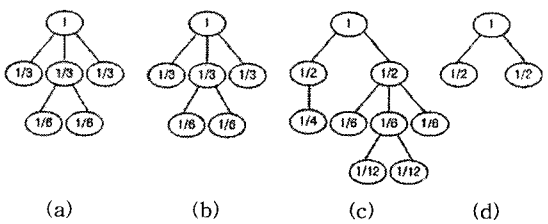


그림 5. 경로와 동기를 고려한 트리 노드의 가중치

그림 5(a) 각 노드의 정규화된 가중치는 각 노드의 경로와 동기들을 고려한 가중치 1, 1/3, 1/3, 1/3, 1/6, 1/6을 전체 노드 가중치 7/3으로 나눈 3/7, 1/7, 1/7, 1/7, 1/14, 1/14이 된다. 트리의 루트로부터 BFS 방식으로 노드를 방문하면서 경로와 동기들을 고려한 각 노드의 정규화된 가중치를 구하는 의사 코드는 그림 6과 같다.

두 XML 문서가 구조 유사도를 가질 때 대응하는 노드 이름의 유사성과 경로 등을 고려하여 내용 유사도를 구하는 의사 코드는 그림 7과 같다. 의사 코드의 마지막 부분에서 내용 유사도 가중치를 2로 나누어 주는 것은 두 XML 문서의 구조와 내용이 일치할 때의 가중치를 1로 하기 위함이다. XML 문서간의 내용 유사도를 구하기 위해서는 두 개의 트리를 각각의 루트로부터 BFS로 방문하면서 비교하기 때문에 트

```
노드의 정규화된 가중치 {
① 큐(queue)에 루트 노드를 삽입 & 루트 노드 가중
        치:= 1 & 전체 노드 가중치:= 1;
/* 각 노드 가중치 구하기 */
② 큐에서 노드 삭제
   if 노드가 자식들이 있으면 {
     큐에 왼쪽 자식 순으로 삽입;
     if 자식수가 1이면 자식 가중치 := 노드 가중치/2;
     else 각 자식 가중치 := 노드 가중치/자식 수;
     전체 노드 가중치 := 전체 노드 가중치 + 자식 가
                    중치 * 자식 수;}
③ if 큐에 노드가 있으면 ②번 수행
   /* 각 노드 정규화된 가중치 구하기 */
   else 각 노드 정규화된 가중치:= 각 노드 가중치 /
                          전체 노드 가중치;}
```

그림 6. 노드의 정규화된 가중치를 구하는 의사 코드

```
내용 유사도(tree t1, tree t2) {
① t1, t2에 대응하는 큐 Q1, Q2에
  각각의 루트 노드 삽입 & 내용 유사도 가중치:= 0;
② Q1에서 노드n, Q2에서 노드m을 삭제;
   if 노드n과 노드m의 이름이 유사하면{
    내용 유사도 가중치:=이름_유사도*
      (노드n 정규화된 가중치+
       노드m 정규화된 가중치)+내용 유사도 가중치;
     if 노드n이 자식들이 있다면
        Q1에 왼쪽 자식 순으로 삽입;
     if 노드m이 자식들이 있다면
        Q2에 왼쪽 자식 순으로 삽입;}
③ if Q1과 Q2에 노드가 있다면 ②번 수행;
   else 내용 유사도 가중치:= 내용 유사도 가중치/2;}
```

그림 7. 내용 유사도 의사코드

리의 전체 노드의 수가 n 일 때 전체 시간 복잡도는 $O(2n)$이다.

3. XML 문서 유사도 및 검증 실험

두 XML 문서의 문서 유사도는 앞 절에서 설명한 구조 유사도와 내용 유사도를 통해 구한다. 우선 두 XML 문서에 대응하는 두 트리를 구한다. 다음으로 두 트리의 EDFS 차이열을 구한 후 스트링 매칭 기법을 적용하여 두 차이열 사이의 포함 관계를 조사한다. 이 때 길이가 짧은 차이열은 길이가 긴 차이열에 한 번 이상 포함될 수 있다. 길이가 짧은 차이열을 갖는 트리와 길이가 긴 차이열을 갖는 트리의 부분트리 간에 내용 유사도를 구한다. 2개의 XML 문서에 대한 문서 유사도를 구하는 의사 코드는 그림 8과 같다.

아래 그림들은 그림 9(a)의 입력 문서와 비교한 문서들의 XML 문서 유사도가 큰 순으로 나열한 것이다. 그림 9(b)는 그림 9(a)와 구조가 같고 모든 내용도 같은 경우로 문서 유사도가 1이다. 그림 9(c)는 그림 9(a)와 구조는 같으나 모든 내용은 유사한 경우이다. 두 대응하는 노드의 이름이 유사한 경우의 이름 유사도를 0.8로 가정하면 그림 9(c)의 문서 유사도는 0.8이 된다. 그림 9(d)는 그림 9(a)와 구조는 같고 입력 문서의 일부 상위 노드들과 비교 문서의 일부 상위 노드들의 이름이 같은 경우로 문서 유사도는 0.71이다. 그림 9(e)는 그림 9(a)의 구조를 포함하고

```
XML 문서 유사도 {
① XML 문서 d1, d2를 대응하는
   트리 t1, t2로 변환;
② t1, t2의 EDFS 차이열 s1, s2를 구함;
③ 스트링 매칭으로 s1, s2의 포함관계를 조사;
④ if s1이 s2에 포함되며
   패턴이 존재하는 위치가 p1,...,pn 이면
     XML 문서 유사도:=max(
      내용 유사도(t1, t2의 노드pi의 부분트리),...,
      내용 유사도(t1, t2의 노드pi의 부분트리));
  else if s2가 s1에 포함되며
  패턴이 존재하는 위치가 p1,...,pn이면
     XML 문서 유사도:= max(
      내용 유사도(t2, t1의 노드pi의 부분트리),...,
      내용 유사도(t2, t1의 노드pi의 부분트리));
  else 문서 유사도:=0;}
```

그림 8. XML 문서 유사도 의사 코드

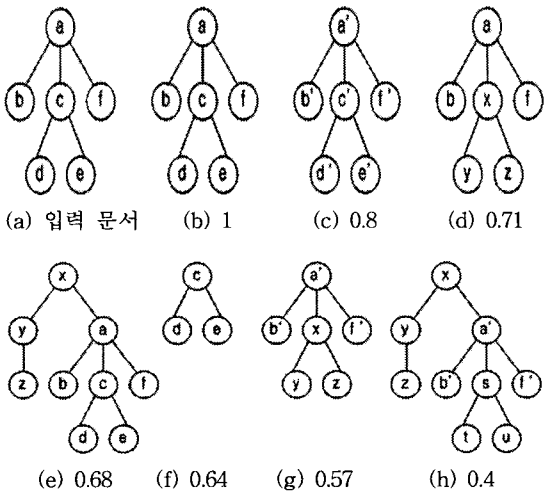


그림 9. 문서 유사도 비교

있으며 입력 문서의 일부 상위 노드들과 비교 문서의 일부 하위 노드들의 이름이 같은 경우로 문서 유사도는 0.68이다. 그림 9(e)의 문서 유사도가 그림 9(d)의 문서 유사도보다 작은 이유는 입력 문서와 대응하는 비교 문서 노드들의 경로가 더 길기 때문이다. 그림 9(f)는 그림 9(a)의 일부 구조와 같고 입력 문서의 일부 하위 노드들과 비교 문서의 상위 노드들의 이름이 같은 그림 9(e)와 비슷한 경우로서 문서 유사도는 0.64이다. 그림 9(g)는 그림 9(a)와 구조는 같으나 입력 문서의 일부 상위 노드들과 비교 문서의 일부 상위 노드들의 이름이 유사한 경우로 문서 유사도는 0.57이 된다. 마지막으로 그림 9(h)는 그림 9(a)의 구조를 포함하며 입력 문서의 일부 상위 노드들과 비교 문서의 일부 하위 노드들의 이름이 유사한 경우로 문서 유사도는 0.4가 된다. 우리는 실험 결과를 통해 비교하는 문서 노드들의 내용이 같은 경우가 내용이 유사한 경우보다 문서 유사도가 높고, 비교하는 문서 노드들의 경로가 짧은 경우가 경로가 긴 경우보다 문서 유사도가 높게 나옴을 알 수 있다.

본 논문에서 제안한 기법을 평가하기 위해서 표 1과 같이 기존에 연구된 여러 유형별의 XML 유사도 검색 기법들과 비교를 시도하였다. XML 문서에 포함된 요소들의 교집합의 정도에 의해 문서의 유사성을 찾는 방법은 가장 간단한 방법으로 볼 수 있으나 문서의 구조적인 정보는 고려하지 못하는 문제점이 있다[2-5]. XML과 같이 트리에 기반한 문서들에 있어서 하나의 트리를 다른 트리로 바꾸는 방법은 문서

표 1. 기존 XML 문서 유사도 기법들과의 비교

	공통요소 방법	tree edit 방법	빈발 경로 방법	경로/요소 방법	푸리에/구조 방법	푸리에/구조 내용방법	제안하는 방법
전제 조건	제한 없음	제한 없음	제한 없음	제한 없음	제한 없음	제한 있음	제한 없음
구조 검사	X	O	△	△	△	O	△
내용 검사	O	X	O	O	X	O	O
복잡도	$O(n)$	$O(n^2)>$	$O(n^2)>$	$O(n^2)>$	$O(n \log n)$	$O(n \log n)$	$O(n)$

들 간의 구조적인 유사성은 평가하나 내용적인 유사성은 제대로 평가하지 못하는 문제점이 있다[6-9]. 순차 패턴 마이닝 알고리즘 등을 사용하여 문서내의 빈발 경로를 찾아내어 구조와 내용에 관한 비교를 하는 방법은 공통된 빈발 경로만을 고려하기 때문에 대표되는 일부 구조만을 통해 유사성을 검사한다[10,11]. 또한 이 방식은 각 문서 간 유사성 결과는 기준 문서에 따라 그 결과가 달라진다. 즉 기준 문서가 무엇이냐에 따라 최대 유사 경로의 비율이 달라지기 때문에 유사성 결과는 대칭적이지 않다. 두 문서의 요소들에 대하여 경로와 내용을 고려한 가중치를 통해 유사도를 계산하는 방법은 검색 대상 문서에서 입력 문서의 루트 노드와 가장 유사한 노드를 찾은 후 해당 일부 구조만을 가지고 문서의 유사성을 검사하며, 또한 이 방식은 검색 대상 문서가 입력 문서에 포함되는 구조적 관계는 검사할 수가 없다[12,13]. XML 문서 요소들의 계층적 구조 관계를 함수로 변환한 후 이산 푸리에 변환을 사용한 방법은 문서 골격의 구조적 유사성만을 비교하며 문서 내용의 유사성은 비교하지 못한다[14,15]. 반면 문서의 요소들을 부호화 함수를 통해 유일한 신호로 변환하여 문서의 구조와 내용의 유사성을 비교하는 방법은 문서의 요소들을 유일한 신호로 변환하기 위하여 비교하고자 하는 모든 문서들의 정보를 사전에 알고 있어야 한다는 가정을 전제로 하고 있다[16,17]. 또한 이 방식은 문서의 구조가 전위 탐색에 의존하게 되는 문제점이 있으며, 두 문서의 크기에 차이가 많이 나서 두 신호열의 길이를 같게 정규화 할 때에 이에 따른 오류가 발생할 수 있다. 본 연구에서 제안하는 방법은 문서 구조가 동일하거나 포함 관계가 있는지를 검사하나 단지 스트링 매칭 기법으로 유사성을 검사할 수 있으며, 요소들의 이름과 경로 등을 고려한 가중치를 통해 문서 내용의 유사성도 함께 계산함으로써 검색의 정확성도 높였다.

4. 결 론

XML 문서가 데이터 표현과 교환을 위한 표준으로 자리 잡고 여러 응용에서 사용됨에 따라 유사한 문서들을 효율적으로 검색하고 군집화하는 방법들이 요구되고 있다. 본 논문은 XML 문서들 간에 어느 정도의 유사성이 있는지를 측정하기 위하여 문서의 구조와 내용의 유사성을 효율적으로 검사하는 새로운 방법을 소개하였다. 본 연구에서 제안하는 방법에서는 문서 구조의 유사성은 두 문서가 같거나 포함 관계가 있는지를 간단한 스트링 매칭 기법을 사용하고 구하고, 내용의 유사성은 대응하는 구조상에서 각 노드 이름의 유사성과 경로 등을 고려한 가중치를 통해 구하기 때문에 시간 복잡도는 비교되는 두 문서의 크기에 선형으로 비례한다. 향후 연구 과제로는 제안하는 방법에 기반하여 문서간의 다른 형태의 구조적 관계에서의 유사성도 효율적으로 검사할 수 있는 방법에 대한 연구가 필요하다고 본다.

참 고 문 헌

[1] R. Behrens, "*A Grammar Based Model for XML Schema Integration*," BNCOD, 2000

[2] 이정원, 이혜수, 이기호, "유사어 백터 확장을 통한 XML 태그의 유사성 검사," 정보과학회 논문지: 소프트웨어 및 응용, 제29권, 제9호, 2002.

[3] 이강석, 송인상, 김응모, "태그 간 동의어 집합을 통한 XML 문서 유사도 측정," 한국정보과학회 2007년도 가을학술 발표논문집, Vol.34, No.2, 2007.

[4] 김수희, 조명찬, 한예지, "내용기반 XML 문서의 검색," 한국정보과학회 2003년도 가을 학술 발표논문집, 제30권, 제2호, 2003.

[5] 한예지, 한창우, 서동혁, 김수희, "XML 문서의

내용기반 검색을 위한 인덱싱 모델 및 색인어의 가중치 부여," 한국정보과학회 2004년도 봄 학술발표 논문집, 제31권, 제1호, 2004.

[6] K. Zhang, D. Shasha, and J. T. L. Wang, "*Approximate tree matching in the presence of variable length don't cares*," J. Algorithms, Vol.16, No.1, 1994.

[7] S. Chawathe and H. Garcia-Molina, "*Meaningful change detection in structured data*," Proceedings of the 1997 ACM SIGMOD, 1997.

[8] A. Nierman and H. V. Jagadish, "*Evaluating structural similarity in XML documents*," Fifth International Workshop on the Web and Databases, 2002.

[9] J. Tekli, R. Chbeir, and K. Yetongnon, "*A Hybrid Approach for XML Similarity*," International Conference on Current Trends in Theory and Practice of Computer Science, 2007.

[10] 박우창, 서여진, "구조와 내용 유사도에 기반한 XML 웹문서 검색시스템 구축," 인터넷정보학회논문지, 제6권, 제2호, 2005.

[11] 이은정, 박우창, "Xtree와 문서 유사도에 기반한 XML 문서 검색," 한국정보과학회 2003년도 봄 학술발표 논문집, 제30권, 제1호, 2003.

[12] 이정원, 이기호, "유사성 기반 XML 문서 분석 기법," 정보과학회논문지 : 소프트웨어 및 응용, 제29권, 제5·6호, 2002.

[13] 황정희, 류근호, "순차패턴에 기반한 XML 문서 클러스터링," 정보처리학회 논문지, 제10-D권, 제7호, 2003.

[14] 이호석, "함수 변환과 FFT에 의한 XML 문서의 구조 비교," 한국정보 과학회 2006년도 한국컴퓨터종합학술 대회논문집(C), 2006.

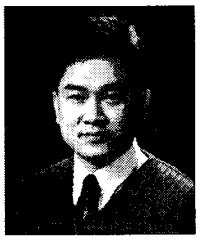
[15] 이호석, "함수 변환 모델링에 의한 XML 문서의 유사성 비교에 대한 연구," 한국정보과학회 2006년도 한국컴퓨터 종합학술대회논문집(C), 2006.

[16] S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese, "*Detecting Structural Similarities between XML Documents*," Proc. Fifth Int'l Workshop Web and Databases, 2002.

[17] S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese, "*Fast Detection of XML Structural Similarity*," IEEE Trans. on Knowledge and Data Engineering, Vol.17, No.2, Feb. 2005.

[18] D. E. Knuth, H. Morris, and V. R. Pratt, "*Fast pattern matching in strings*," SIAM Journal on Computing, June 1997.

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



김 우 생

1985년 서울대 수료 및 Univ. of Texas 전산학 학사
1987년 Univ. of Minnesota 전산학 석사
1991년 Univ. of Minnesota 전산학 박사
2001년 UC 버클리 대학교 교환교수
1992년~현재 광운대학교 컴퓨터공학부 교수
관심분야 : 데이터베이스, 멀티미디어