DOI: 10.3745/KIPSTD.2008.15-D.2.171

XML 문서의 유사 경로 검색을 위한 인덱싱 시스템

이 범 석[†] · 황 병 연[††]

요 약

1998년 W3C에 의해 XML 표준이 제정된 이래로, XML을 사용하는 문서가 급증하였다. 이에 따라 방대한 양의 XML 문서들을 효율적으로 관리하고 검색하기 위한 많은 시스템들이 개발되고 있다. 특히 비트맵 인덱스 기법을 사용한 BitCube는 이러한 분야의 대표적인 시스템이다. 비트맵 인덱스 기법을 이용하여 유사한 경로를 대상으로 클러스터링을 수행한 경로 비트맵 인덱스 시스템(LH06)은 기존의 BitCube 시스템이 유사경로 검색을 할 수 없는 문제점을 개선하였다. 유사경로 검색 시스템은 정확히 일치하는 경로뿐만 아니라, 사용자가 질의한 경로와 유사한 경로까지도 빠르게 검색해 낼 수 있다는 장점을 가진다. 그러나 경로 사이의 유사도를 계산하는 알고리즘이 가진 몇 가지 문제점들로 인해 유사하다고 볼 수 있는 두 경로의 유사도를 계산할 수 없어서 서로 다른 클러스터로 인식되고, 이는 의미 없는 클러스터의 수를 증가시키는 문제점을 야기한다. 이러한 문제점의 해결을 위해 본 논문에서는 보다 합리적이고 정확한 경로 유사도 계산 방법을 제안하고, 기존 시스템과의 성능평가를 통해 제안하는 방법이 더 낫다는 것을 증명한다.

키워드 : XML, 인덱싱 시스템, 유사 경로, 클러스터링

An Indexing System for Retrieving Similar Paths in XML Documents

Bum-Suk Lee[†] · Byung-Yeon Hwang[††]

Abstract

Since the XML standard was introduced by the W3C in 1998, documents that have been written in XML have been gradually increasing. Accordingly, several systems have been developed in order to efficiently manage and retrieve massive XML documents. BitCube−a bitmap indexing system−is a representative system for this field of research. Based on the bitmap indexing technique, the path bitmap indexing system(LH06), which performs the clustering of similar paths, improved the problem that the existing BitCube system could not solve, namely, determining similar paths. The path bitmap indexing system has the advantage of a higher retrieval speed in not only exactly matched path searching but also similar path searching. However, the similarity calculation algorithm of this system has a few particular problems. Consequently, it sometimes cannot calculate the similarity even though some of two paths have extremely similar relationships; further, it results in an increment in the number of meaningless clusters. In this paper, we have proposed a novel method that calculates the similarity between the paths in order to solve these problems. The proposed system yields a stable result for clustering, and it obtains a high score in clustering precision during a performance evaluation against LH06.

Key Words : XML, Indexing system, Similar path, Clustering

1. 서 론

1998년 W3C는 XML[1] 표준을 제정하였다. 데이터를 어떻게 표현할 것인가에 중점을 두었던 HTML과 달리, XML은 데이터를 어떻게 구조화하여 보다 효과적으로 전달하거나 저장할 것인지에 그 목적을 가지고 개발되었다. XML의 장점은 유연성이다. 이로 인해 데이터를 쉽게 구조화할 수 있어서 초기에는 B2B나 B2C에서 사용되다가, 최근에는 RSS(RDF Site Summary)나 XML을 지원하는 다양한 웹 애플리케이션을 통해 방대한 양의 XML 문서가 유통되고 있다. 그러나 XML의 장점인 유연성은 개인들에 의해 문서의 구조를 쉽게 변경할 수 있게 하였고, 다양한 구조의 XML 문서가 만들어졌다. 이에 따라 다양한 구조를 갖는 많은 양의 XML 문서를 효율적으로 관리하기 위한 시스템의 연구[2,3,4]가 진행되고 있다.

이와 관련된 연구로 3차원 비트맵 인덱스를 사용한 BitCube[5,6] 시스템이 있다. 이 시스템은 XML 문서가 포함한 문서이름 d, 경로 p, 단어 w를 기준으로 세 개의 축을 생성하고, (d, p, w) 쌍의 값이 존재하면 1, 존재하지 않으면 0의 값을 갖는 3차원 비트맵 인덱스를 생성한다.

※본 연구는 2007년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음
† 준 회 원 : 가톨릭대학교 컴퓨터공학과 박사과정
†† 종신회원 : 가톨릭대학교 컴퓨터정보공학부 교수
논문접수 : 2007년 7월 23일, 심사완료 : 2007년 11월 26일

그리고 생성된 비트맵 인덱스를 문서가 포함하고 있는 경로 p의 존재 유무에 따라 유사도를 계산하고 유사한 경로를 포함한 문서들끼리 클러스터링을 수행한다. 이 시스템은 다른 상용 XML 문서 검색 시스템들[7,8]과의 성능평가에서 우수한 성능을 입증하였다. 그러나 BitCube 시스템은 문서의 경로에 새로운 노드(엘리먼트나 애트리뷰트)가 추가되었을 때 그 구조가 유사하지만 서로 다른 구조로 인식하게 되는 문제점을 가지고 있었다. 이러한 문제점을 해결하여 유사한 구조를 가지는 경로 클러스터링을 수행하는 기법[9]이 제안되었다. 본 논문에서는 이 기법을 LH06이라고 부르도록 한다. LH06은 경로 유사도 계산식을 제안하고, 이 식을 이용하여 유사한 경로들을 대상으로 클러스터링을 수행하였다. 이 시스템은 BitCube의 문제점을 해결하였으나, 경로를 구성하는 노드의 순서가 바뀌는 경우에는 유사도를 계산하지 못하는 등의 몇몇 문제점을 가지고 있었다.

본 논문에서는 이러한 문제점을 해결하기 위하여 보다 합리적이고 정확한 경로 유사도 계산 방법을 제안한다. 제안하는 방법은 두 경로를 서로 똑같이 만들기 위한 노드의 삽입과 삭제, 치환 연산을 정의하고, 각각의 연산 비용을 이용하여 두 경로 사이의 거리를 계산한다. 거리를 이용하여 유사도를 계산하고 서로 같은 경로일 경우는 1을, 서로 완전히 다른 경우는 0의 값을 갖는다. 제안하는 시스템은 LH06과의 성능평가에서 보다 안정적인 클러스터링 결과와 높은 정확도를 보여주었다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 LH06에 대해 소개하고, 그 문제점을 살펴본다. 3장에서는 본 논문에서 제안하는 경로 유사도 계산식을 정의하고, 이를 이용한 유사 경로 클러스터링 시스템을 소개한다. 4장에서는 성능평가를 통해 제안한 방법을 사용한 시스템이 기존 LH06에 비해 나은 클러스터링 성능을 가진다는 것을 증명한다. 마지막으로 5장에서는 결론 및 향후 연구 계획에 대해 논의한다.

2. 관련연구

기존의 XML 검색을 위한 3차원 비트맵 인덱싱인 BitCube는 경로를 중심으로 유사 문서를 수집하고 검색하는 기법이다. 같은 경로를 많이 포함하는 문서들은 구조가 유사하다고 볼 수 있으므로 동일한 클러스터에 수집된다. 그러나 기존의 3차원 비트맵 인덱싱은 경로를 있는 그대로 추출함으로써 같은 경로를 포함하는 문서를 인식할 수는 있으나 유사 경로를 포함하는 문서를 인식하는 것은 불가능하다. 그러므로 하나의 문서에 구조적인 변형이 가해진다면 문서에서 변경 전에 추출한 경로와 변경 후에 추출된 경로가 정도에 상관없이 완전히 다른 것으로 인식된다. 이로 인해 유사 문서의 클러스터링 및 검색에 현저한 성능저하가 발생한다. 이러한 문제점을 해결하기 위해 LH06의 연구에서는 다음과 같이 두 개의 정의를 제시하였다[9].

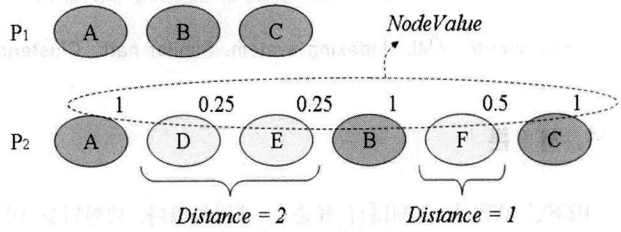
[정의 2.1] 두 경로 중에서 더 적은 노드를 갖는 경로를 기준 경로 P_1이라 하고 더 많은 노드를 갖는 경로를 비교 대상 경로 P_2라고 한다. 기준 경로 P_1과 비교 대상 경로 P_2의 경로 구성 유사도 $P.C.Sim(P_1,P_2)$는 다음과 같이 정의된다. (여기에서 $NodeNum(P)$는 경로 P를 구성하는 모든 노드의 개수이다.)

$$P.C.Sim(P_1,P_2)=\frac{\sum_{i=1}^{NodeNum(P_2)} NodeValue(P_2,N_i)}{NodeNum(P_2)}$$

[정의 2.2] 경로 P를 구성하는 노드 N의 노드 값 $NodeValue(P, N)$ 은 다음과 같이 정의된다. d는 기준 경로에 존재하는 노드가 발견될 때 까지 기준 경로에 존재하지 않는 노드들의 개수, 즉 일치하지 않는 노드의 개수이다.

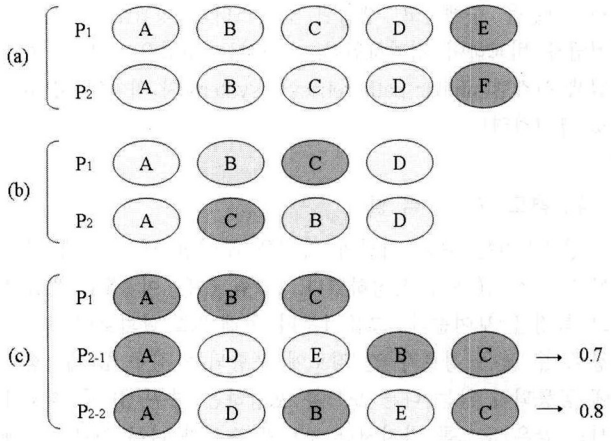
$$NodeValue(P,N)=\frac{1}{2^d}$$

(그림 1)은 A.B.C로 구성된 기준 경로 P_1에 대하여, A.D.E.B.F.C와 같이 구성된 비교 대상 경로 P_2의 유사도를 계산하는 방법을 보여준다. 정의 2.1과 정의 2.2에서 제시한 식을 이용하여 P_1과 P_2의 경로 구성 유사도를 구하면, 우선 P_2에 존재하는 노드의 NodeValue를 계산한다. A노드의 경우 P_1과 P_2에 동일하게 보여지므로 1의 NodeValue를 가질 수 있고, D와 E 노드는 A와 B 사이의 거리가 2이므로 정의 2.2에 의해 각각 0.25의 NodeValue를 가지게 된다. 동일한 방법으로 B, F, C의 NodeValue는 각각 1, 0.5, 1이 된다. 이 NodeValue의 합을 P_2의 노드 개수인 6으로 나누면 된다. 정리하면 (1+0.25+0.25+1+0.5+1)/6≒0.67이 된다.



(그림 1) 경로 구성 유사도의 계산

위에서 살펴본 경로 구성 유사도 계산식은 다음과 같은 여러 가지 문제점을 가지고 있다. 1) 다른 구성을 가진 경로 사이의 유사도 계산이 불가능하다. 즉, (그림 2)의 (a)와 같은 두 경로에 대해서는 P_2에 존재하는 노드 F의 $NodeValue$를 구할 수 없으므로, 위의 식으로는 계산이 불가능하다. 2) (그림 2)의 (b)와 같이 노드의 순서가 다른 두 경로 사이의 유사도 계산도 1)과 같은 이유로 P_2에 존재하



(그림 2) 경로 구성 유사도 계산식의 문제점

는 노드 D의 $NodeValue$를 구할 수 없다. 3) 또한 거리 기반의 유사도를 계산하기 때문에 (그림 2)의 (c)와 같이 두 개의 노드가 존재하는 위치에 따라 기준 경로 P_1과 비교 대상 경로 $P_{2\text{-}1}$, $P_{2\text{-}2}$에 대한 각각의 유사도가 0.7과 0.8로 다르다.

앞에서 지적한 LH06의 논문이 제안한 유사도 계산식의 문제점 때문에 유사한 경로를 기준으로 클러스터링이 수행될 때 유사한 경로로 볼 수 있는 경로 구성이 서로 다른 클러스터에 저장될 수 있다. 이와 같이 유사한 경로가 다른 클러스터에 저장되면 경로 검색을 수행할 때 원하는 결과가 선택된 클러스터가 아닌 다른 클러스터에 포함되어있을 확률이 높아지게 되고, 다시 말하면 재현율(Recall)의 품질저하가 일어나게 된다.

3. 유사 경로 클러스터링 시스템

3.1 경로 유사도의 계산

이 장에서는 2장에서 설명했던 경로 구성 유사도 계산의 문제점을 보완하기 위해 새로운 경로 유사도 계산 방법을 제안한다. 이것을 계산하기 위해 우선 삽입, 삭제, 치환의 세 가지 연산에 대해 정의한다.

[정의 3.1] *insert(x)*는 해당 경로에 새로운 노드 x를 삽입하는 연산이고, 이 연산을 수행하는데 드는 비용은 1이다.

Cost(insert(x)) = 1

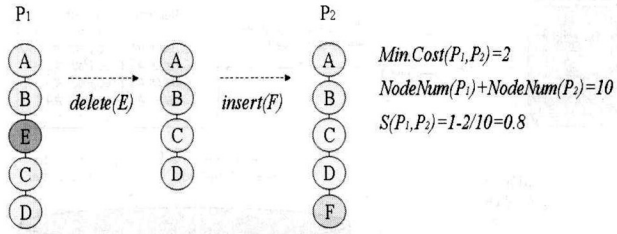
[정의 3.2] *delete(x)*는 해당 경로를 구성하는 노드 중 x를 삭제하는 연산이고, 이 연산의 수행비용은 1이다.

Cost(delete(x)) = 1

[정의 3.3] *replace(x, y)*는 해당 경로의 노드 x를 y로 치환한다. 이 연산의 수행은 *delete(x)*와 *insert(y)*를 수행하는 비용을 합한 2이다.

Cost(replace(x,y)) = Cost(delete(x)+insert(y)) = 2

위의 정의를 이용하면 어떤 임의의 경로를 특정한 형태의 경로로 변경하기 위한 비용을 계산할 수 있다. 이 비용의 최소 비용을 $Min.Cost(P_1,P_2)$라고 하면, 두 경로 사이의 유사도 $S(P_1,P_2)$는 다음과 같이 정의된다.



(그림 3) 두 경로 P1을 P2 사이의 유사도

[정의 3.4] 경로 P_1과 P_2 사이의 유사도 $S(P_1,P_2)$는 다음 식과 같다. P_1을 P_2로 변경하는 최소 비용을 P_1과 P_2의 노드 개수의 합으로 나누고, 그 값을 1에서 뺀다. 따라서 이 식에 따르면, 유사도 값은 0과 1 사이의 값을 가지게 되고, P_1과 P_2가 완전히 다른 경우 0, 완전히 일치하는 경우 1이다.

$$S(P_1, P_2) = 1 - \frac{Min.Cost(P_1, P_2)}{NodeNum(P_1) + NodeNum(P_2)}$$

(그림 3)은 정의 3.4를 이용하여 실제 두 경로 사이의 유사도를 구하는 예를 보여준다. 두 경로 P_1:A.B.E.C.D와 P_2:A.B.C.D.F는 P_1을 P_2로 변형하기 위해 *delete(E)*와 *insert(F)* 연산을 각각 한 번씩 수행했으므로 최소 비용($Min.Cost(P_1,P_2)$)은 2가 되고, 두 경로에 포함된 노드의 개수 합이 10이다. 이를 정의 3.4의 유사도 계산식에 대입하면, 두 경로의 유사도는 0.8이 된다.

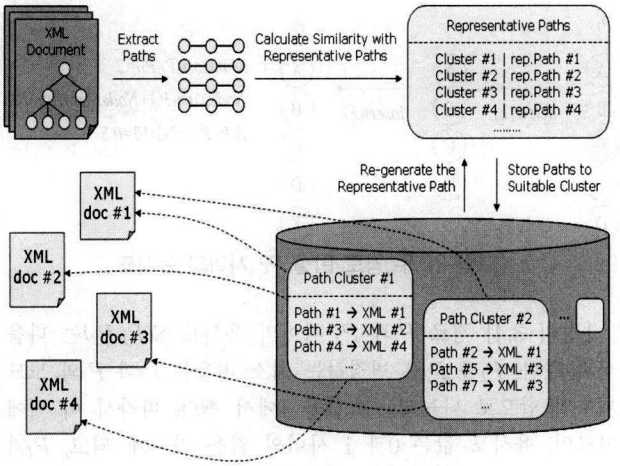
제안하는 방법은 기존의 방법으로 유사도를 계산할 수 없었거나 정확하지 않았던 (그림 2)의 예에서도 사용될 수 있다. 우선 (그림 2)의 (a)는 경로가 서로 공통되지 않은 다른 노드를 포함한 경우이다. 제안하는 방법의 유사도 계산식을 이용하면, 그 유사도는 0.8이 된다. (b)는 두 경로를 구성하는 노드의 순서가 다른 경우이다. 이 경우의 유사도는 0.75이다. (c)는 거리 기반의 기존 방법으로 기준 경로 A.B.C에 대해 두 경로가 서로 다른 유사도를 가졌었지만, 제안하는 방법으로는 기준 경로에 대해 두 경로 모두 0.75의 유사도를 가진다.

3.2 유사 경로 클러스터링

본 절에서는 제안한 경로 유사도 계산식을 이용하여, 유사한 경로들끼리 인덱스 클러스터를 생성하고 XML 문서를 인덱싱 하는 시스템을 소개한다. (그림 4)는 제안하는 유사 경로 클러스터링 시스템의 구조를 보여준다.

먼저 입력된 문서에서 경로들을 추출하고, 각 경로와 대표 경로 테이블에 저장되어있는 클러스터 대표 경로 사이의 유사도를 비교하여 유사한 경로들을 모아놓은 클러스터를 생성한다. 새로운 경로가 입력될 때마다 클러스터의 대표 경로는 항상 새롭게 갱신된다. 또한 클러스터에 적재된 경로들은 특정한 XML 문서에 대한 링크를 유지한다.

(그림 4)의 클러스터링 과정을 알고리즘으로 나타내면 (그림 5)와 같다. XML 문서 d가 들어오면 시스템은 문서의



(그림 4) 경로 클러스터링 시스템의 구조

경로를 추출하고, 입력된 경로와 각 경로 클러스터의 대표 경로 사이의 유사도를 비교한다. 유사도가 임계값을 넘는 클러스터가 존재하지 않는다면, 새로운 경로 클러스터를 생성하고 입력된 경로를 삽입한다. 만약 임계값을 넘는 클러스터가 존재한다면, 해당 클러스터에 경로를 저장한다. 이 과정이 끝난 후에 새로운 클러스터나 또는 경로가 저장된 클러스터의 대표 경로를 생성하고 대표 경로 테이블을 갱신한다. 이 과정을 통해 제안하는 시스템은 항상 임계값 이상의 클러스터를 유지한다.

4. 성능평가

본 논문에서 제안하는 계산식의 성능 평가를 위해 Java 1.4.2를 이용하여 프로그램을 구현하였고, 윈도우 2000 Server 플랫폼이 설치된 펜티엄 4 1.7 GHz CPU와 1024MB RAM을 가진 PC환경에서 실험을 수행하였다. 성능평가는 두 부분으로 나누어 진행하였다. 첫 번째는 경로 유사도 식이 경로 클러스터링에 적합한지를 검증하기 위한 실험이었고, 두 번째 실험은 경로 유사도 식을 이용하여 실제로 클러스터링을 수행하고, 생성된 클러스터의 정확도를 LH06의 기법과 비교하여 평가하였다. 제안하는 방법은 유사한 경로 검색 시스템(similar path retrieval system)의 약자인 SPARS로 표기하였다.

4.1 경로 유사도 식 평가

성능평가를 위해 임의의 특징을 부여한 네 그룹의 경로 쌍을 각각 10개씩 생성하였다. (그림 6)은 이 경로 쌍들과 그 특징을 보여준다. 그룹 1부터 차례대로 살펴보면, 1) 두 경로 중 짧은 경로가 긴 경로에 포함되는 경우, 2) 두 경로에 공통되지 않는 다른 노드를 포함하는 경우, 3) 두 경로가 서로 같은 노드를 가지지만 노드의 순서가 다른 경우, 4) 서로 같은 경로 구성에 대해 다른 유사도를 가지는 경우이다.

그룹 1에서부터 그룹 4까지 각각의 경로 쌍에 대해 유사도 계산을 수행해 보았다. (그림 7)의 (a)는 그룹 1의 경로 쌍에 대한 유사도 계산 결과를 보여준다. 제안하는 방법은 기존의 방법에 비해 경로 쌍에 따라 다르지만 비교적 0과 1 사이에서 고른 분포를 가지는 유사도 결과를 가진다. 이는 클러스터링 수행에 적절하다는 것을 의미한다. 그룹 2와 그룹 3은 (그림 7)의 (b)와 (c)에서 보여주는 것과 같이 제안한 유사도 계산식으로는 고르게 분포된 유사도를 얻을 수 있었지만, 기존의 방법으로는 모두 계산이 불가능하므로 0의 유사도를 얻게 되었다. 이는 기존 유사도 계산식의 문제점을 보여주는 현상으로, 두 경로가 서로 다른 노드를 포함하거나 노드의 순서가 다른 경우에는 유사도를 구하지 못하고, 따라서 서로 유사한 두 경로를 비교할 경우에도 0의 유사도를 가지게 된다.

(그림 8)은 그룹 4에 대한 유사도 계산 수행 결과를 보여준다. 그룹 4에서 기존의 유사도 계산식은 두 경로 쌍에 대해 서로 다른 유사도 계산 결과 값을 가지는 것을 알 수 있다. 또한 그 결과 값의 차이가 경로의 구성에 따라 매우 크게 변화한다. 특히 비교하려는 경로 P_2에 존재하는 기준경로 P_1에 존재하지 않는 노드들이 특정한 두

```
Algorithm clustering(d)

for (int i = 0;path[i] != null;i++){ // path[i] is the set of paths in d
   if (numberOfCluster = 0){ // If there are no existing clusters
      cid = createNewCluster(path[i]); // Create a new cluster with path[i] and get the cluster id
   }else{
      (sim,cid) = getClusterSim(path[i]); // Get similarity value and cluster id that is the most similar to path[i]
      if (sim >= threshold){ // If the similarity value is the same or greater than the threshold value
         insertPath(cid,path[i]); // Insert path[i] into the selected cluster identified in the cid operation
      }else{
         cid = createNewCluster(path[i]); // Create a new cluster with path[i] and determine the cluster id
      }
   }
   repPath = getRepPath(cid); // Recalculate the representative path of the updated cluster
   updateRepPathTable(cid,repPath); // Update the representative path table. If there is no repPath with the inputted cid
value, then insert new cid and repPath values
}
```

(그림 5) 경로 클러스터링 알고리즘

	예제 경로 쌍	특징
그룹 1	P_1 : B.C.D P_2 : A.B.C.D.E.F	두 경로 중 짧은 경로가 긴 경로에 포함
그룹 2	P_1 : A.B.C.D.E P_2 : A.B.C.D.F	두 경로에 공통되지 않는 다른 노드를 포함
그룹 3	P_1 : A.B.C.D P_2 : A.C.B.D	두 경로가 서로 같은 노드를 가지지만, 노드의 순서가 다른 경우
그룹 4	P_1 : A.B.C $P_{2\text{-}1}$: A.D.E.B.C $P_{2\text{-}2}$: A.D.B.E.C	서로 같은 경로 구성에 대해 다른 유사도를 가지는 경우

(그림 6) 네 그룹의 경로 쌍과 그 특징

	A	B	C	D	E	F	G	H	I	J
LH06 method	0.515	0.7	0.294	0.382	0.176	0.761	0.548	0.661	0.737	0.85
SPARS method	0.625	0.625	0.454	0.454	0.3	0.687	0.708	0.708	0.823	0.823

(a) Group 1

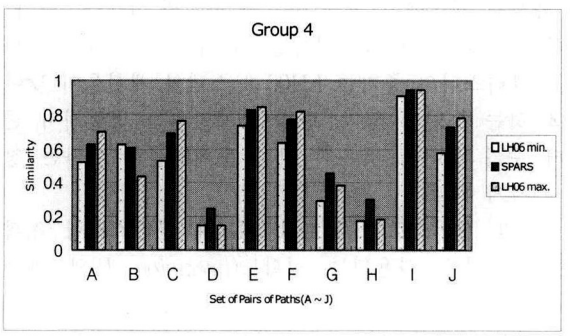
	A	B	C	D	E	F	G	H	I	J
LH06 method	N/A(=0)									
SPARS method	0.774	0.521	0.608	0.687	0.875	0.727	0.324	0.12	0.925	0.785

(b) Group 2

	A	B	C	D	E	F	G	H	I	J
LH06 method	N/A(=0)									
SPARS method	0.778	0.714	0.525	0.625	0.293	0.254	0.92	0.627	0.508	0.513

(c) Group 3

(그림 7) 그룹 1, 2, 3의 유사도 계산 수행 결과



(그림 8) 그룹 4의 유사도 계산식 결과

노드 사이에 밀집해 있으면 기준경로 P_1과 P_2의 유사도는 낮아진다(LH06 min.). 반대로, 이러한 노드들이 여러 노드들 사이에 고르게 분포되어있으면, 노드의 value가 높아지므로 유사도도 함께 높아진다(LH06 max.). 이는 (그림 6)의 그룹 4에서 예제 경로 쌍 P_1과 $P_{2\text{-}1}$, P_2과 $P_{2\text{-}2}$의 유사도가 각각 0.7과 0.8인 것에서 알 수 있다. 이러한 차이는 노드 D와 E가 모여 있는지, 분산되어 있는지 여부로 결정된다. 이에 비하여 제안하는 방법은 항상 일정한 경로 유사도(0.75)를 보이는 것을 알 수 있었다. 이는 제안된 방법이 경로 P_1을 P_2로 변형하기 위한 삽입, 삭제, 치환 연산을 경로의 어느 위치에서나 수행할 수 있기 때문에, 노드의 위치가 다르더라도 유사도 계산에는 영향을 주지 않는다는 것을 나타낸다.

4.2 유사 경로 클러스터링 평가

이 절에서는 경로 유사도 계산식을 실제로 클러스터링에 적용한 실험 결과를 제시한다. 실험 데이터는 NewsML[10] 스키마로 작성된 실제 XML 문서 데이터를 사용하였으며, 80개의 문서에서 1670개의 경로를 추출하였다. 이 데이터는 로이터(REUTER) 뉴스기사를 XML 형태로 저장한 것으로 실험에 사용된 문서의 전체 크기는 약 15.6Megabyte이다. 실험은 세 부분으로 나누어 진행하였다.

첫 번째는 생성된 클러스터의 수를 비교하였다. (그림 9)는 제안하는 유사도 계산 기법과 기존 기법에 대해 임계값(threshold)을 변화시키며 생성되는 클러스터의 수를 확인한

Threshold	LH06	SPARS
0.2	15	7
0.4	18	16
0.6	262	120
0.8	429	302

(그림 9) 임계값의 변화에 따른 클러스터의 수 비교

Threshold	Number of Transformation	LH06	SPARS
0.2	0	15	7
	1	26	7
	2	52	7
0.4	0	18	16
	1	33	16
	2	64	17

(그림 10) 노드 순서 변경에 따른 클러스터의 수 변화

Threshold		LH06	SPARS
0.2	Precision	0.779	1
	Recall	0.859	1
0.4	Precision	0.579	1
	Recall	0.346	1

(그림 11) 클러스터링의 품질 비교

결과이다. 기존 기법은 임계값에 따라 다르지만, 항상 제안하는 기법에 비해 적게는 1.12배에서 많게는 2배 정도 많은 수의 클러스터를 생성한다.

두 번째 실험으로는 앞의 실험에서 생성한 클러스터에 저장된 경로들을 대상으로 노드의 순서를 변경하여 추가로 저장해보았다. 이는 기존의 기법에서 노드의 순서가 다른 경우 유사도를 계산하지 못하여 유사도가 0이 되는 현상이 실제 클러스터링을 수행할 때 어떠한 영향을 주는지 확인하기 위함이다. 이 실험에서는 다음과 같이 임의로 작성된 데이터를 사용하였다. 변경된 노드의 수를 1개와 2개로 나누어 진행하였는데, 우선 1개의 노드에 대해 변경을 하기 위해서 root노드를 제외한 가장 상위의 노드를 가장 하위로 이동하였다. 예를 들면, (root).A.B.C.D인 경로를 (root).B.C.D.A로 변형하였다. 이 방법으로 노드의 순서를 변경시킨 경로를 추가로 저장하였을 때, 기존 방법은 임계값이 0.2일 때와 0.4일 때 모두 약 2배만큼 증가하였다. 2개의 노드에 대해 순서를 변경하였을 때에는 다시 1개의 노드 순서를 변경한 후의 클러스터 수의 약 2배만큼 증가하였다. (그림 10)은 이 실험의 결과를 보여준다.

이러한 결과는 경로의 깊이가 2 이하여서 노드 순서 변경이 일어나지 않은 몇몇 경로를 제외하고, 노드 순서가 변경된 모든 경로에 대해 새로운 경로로 인식하였음을 보여준다. 반면 제안하는 유사도 계산 방법은 노드의 순서가 유사도 계산에 영향을 주기는 하지만, 유사도 결과 계산하지 못하는 경우는 없음을 알 수 있다. 이처럼 의미 없는 클러스터의 증가는 LH06 시스템의 재현율이 낮아지도록 하는 요인이 되며, 반대로 제안하는 방법에서 클러스터 수가 줄어든 것은 이러한 문제점을 보완하는 장점을 가지게 된다.

마지막 실험으로는 정보검색[11]과 클러스터링 정확도[12]의 평가에 사용되는 정확도(*precision P*)와 재현율(*recall R*)을 계산하여 클러스터링 품질을 평가하였다. (a) a_i를 생성된 클러스터 C_i에 포함된 정확하게 클러스터링된 경로의 수(correctly clustered), (b) b_i를 C_i에 포함된 잘못 클러스터링된 경로의 수(misclustered), (c) c_i를 C_i에 클러스터링 되어야 할 경로들 중 C_i에 존재하지 않는 경로의 수라 가정한다면, 이때 정확도 P와 재현율 R은 각각 $P = \sum_i a_i/(\sum_i a_i+\sum_i b_i)$와 $R = \sum_i a_i/(\sum_i a_i+\sum_i c_i)$로 정의된다. 이 실험은 임계값 0.2와 0.4에서 클러스터링을 수행한 다음 제안하는 기법에 대해 기존 기법에서 생성된 클러스터의 상대적인 정확도를 평가하였다. (그림 11)은 제안하는 방법의 클러스터링 품질을 1로 보았을 때, 기존 기법의 상대

적인 클러스터링 정확도와 재현율의 수준을 비교하여 보여준다.

5. 결론 및 향후 연구 계획

본 논문에서는 XML 문서의 경로 클러스터링을 위한 기존의 경로 구성 유사도 계산식에서 드러난 몇 가지 문제점을 개선하기 위해 새로운 경로 구성 유사도 계산식을 제안하였다. 기존의 계산식을 이용할 수 있는 범위가 기준 경로 P_1과 비교 대상 경로 P_2를 비교하기 위해, 1) 항상 짧은 경로가 기준 경로 P_1이 되어야 하는 점, 2) 짧은 경로 P_1을 구성하는 모든 노드가 비교하려는 P_2 경로에 포함되어야 하는 점과 같은 한계를 가지고 있었고, 3) 기준 경로 P_1과 비교하려는 경로 P_2의 노드 순서가 다른 경우 유사도 계산이 불가능하였고, 4) 기준 경로 P_1에 대해 경로 구성 순서가 다른 두 경로 $P_{2\text{-}1}$, $P_{2\text{-}2}$의 유사도가 서로 다르게 계산되는 등의 문제점을 가지고 있었다.

본 논문에서는 이러한 문제점을 해결하기 위한 새로운 경로 구성 유사도 계산식을 제안하였다. 제안한 방법은 두 경로를 동일한 형태로 바꾸기 위한 최소 비용을 계산하고, 이를 이용하여 0과 1 사이의 경로 유사도를 계산하였다. 또한 실제 XML 데이터와 임의의 데이터를 이용하여 기존 방법과 제안한 방법으로 각각 클러스터링을 수행하였고, 임계값의 변화에 따른 클러스터의 수 비교 실험, 노드 순서 변경에 따른 클러스터 수 비교 실험, 정확도와 재현율을 이용한 클러스터링 정확도 측정을 실시하였다. 이 실험들을 통해 제안한 유사도 계산 방법이 기존 방법에 비해 안정적인 클러스터링 성능을 제공함을 증명하였다. 또한 기존의 유사도 계산식을 이용한 클러스터링에서는 노드의 순서가 바뀌는 경우 바뀐 노드의 수가 증가함에 따라 기존에 생성된 클러스터의 수만큼의 새로운 클러스터가 생성되는 문제점을 보인 반면, 제안하는 방법을 이용한 클러스터링에서는 이러한 문제점이 발생하지 않음을 알 수 있었다.

물론 본 논문에서 제안하는 방법에도 유사도 계산이 합리적이지 못한 사례가 있었다. 예를 들어 두 경로 A.B.C.D.E.F.G.H와 E.F.G.H.A.B.C.D의 경우 노드의 순서만 변경되었음에도 0.5의 낮은 유사도를 가진다. 또한 이 예에 비해 경로 구성 노드가 같지 않은 A.B.C.D.E.F.G.H와 W.X.Y.Z.A.B.C.D 경로 쌍의 유사도 역시 0.5를 가진다. 이러한 문제점은 실제 클러스터링 시스템을 구축할 때, 본 논문에서 제안한 유사도 계산방법과 더불어 경로를 구성하는 노드의 유사도를 적절하게 반영해주는 것이 필요함을 의미한다. 하지만 이와 같이 경로를 구성하는 노드의 유사도와 본 논문에서 제안하는 유사도 계산식에 대해 각각 얼마의 가중치를 부여해야 하는지를 결정하는 것은 매우 어렵다. 향후 연구로 이러한 부분에 대해 가중치 비율의 트레이드 오프 관계를 찾아내는 것은 의미가 있을 것이다. 또한 경로 유사도 계산식을 XML 문서의 구조 유사도 계산에 적용하는 연구를 진행하는 것도 본 논문과 관련된 연구 과제이다.

참 고 문 헌

[1] http://www.w3.org/TR/2000/REC-xml-20001006

[2] T. Dalamagas, T. Cheng, K. J. Winkel, and T. Sellis, "Clustering XML documents using structural summaries," In Proc. of the EDBT Workshop on Clustering Information over the Web (ClustWeb04), Heraklion, Greece, 2004.

[3] J. H. Hwang and K. H. Ryu, "Clustering and Retrieval of XML Documents by Structure," Lecture Notes in Computer Science, Vol.3481, Springer Berlin, 2005.

[4] U. Park and Y. Seo, "An Implementation of XML Documents Search System based on Similarity in Structure and Semantics," In Proc. of the Web Information Retrieval and Integration, 2005(WIRI '05), pp. 97-103, April 2005.

[5] J. P. Yoon, V. Raghavan, V. Chakilam, and L. Kerschberg, "BitCube: A Three-Dimensional Bitmap Indexing for XML Documents," Journal of Intelligent Information System, Vol.17, pp. 241-254, 2001.

[6] J. Yoon, V. Raghavan, and V. Chakilam, "BitCube: Clustering and Statistical Analysis for XML Documents," In Proc. of the 13th Int'l Conf. on Scientific and Statistical Database Management, Virginia, Jul. 2001.

[7] D. Egnor and R. Lord, "XYZFind: Structured Searching in Context with XML," In Proc. of ACM SIGIR Workshop, Athens, Greece, 2000.

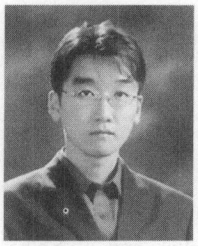
[8] XQEngine. http://www.fatdog.com

[9] Jae-Min Lee and Byung-Yeon Hwang, "Path Bitmap Indexing for Retrieval of XML Documents," Lecture Notes in Computer Science, Vol.3885, Springer-Verlag, Apr. 2006.

[10] NewsML, http://www.newsml.org

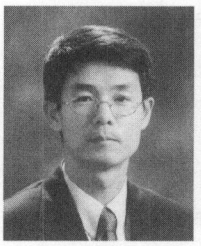
[11] C. J. van Rijisbergen. "Information Retrieval," Butterworths, London, 1979.

[12] T. Dalamagas, T. Cheng, K. J. Winkel, and T. Sellis, "A Methodology for Clustering XML Documents by Structure," Information Systems, Vol.31, Issue 3, Elsevier Science Ltd., pp. 187-228, May, 2006.



이 범 석

e-mail : leez79@gmail.com
2004년 가톨릭대학교 분자생물학과 (이학사), 국사학과(문학사)
2006년 가톨릭대학교 컴퓨터공학과 (공학석사)
2006년~현재 가톨릭대학교 컴퓨터공학과 박사과정 재학
관심분야 : XML, 웹2.0, 정보검색, 웹서비스, 생물정보



황 병 연

e-mail : byhwang@catholic.ac.kr
1986년 서울대학교 컴퓨터공학과(공학사)
1989년 한국과학기술원 전산학과 (공학석사)
1994년 한국과학기술원 전산학과 (공학박사)
1994년~현재 가톨릭대학교 컴퓨터정보공학부 교수
1999년~2000년 University of Minnesota Visiting Scholar
2007년~2008년 California State University, Sacramento Visiting Scholar
관심분야 : XML 데이타베이스, 데이타 마이닝, 지리정보시스템, 정보검색