클러스터의 주요항목 가중치 기반 XML 문서 클러스터링

황 정 희[†]

요 약

발달된 인터넷 환경과 데이터 교환 표준 언어로서 확정되고 있는 XML을 기반으로 하여 대량의 웹 문서들이 생산되면서 정보 추출의 대상은 자연스럽게 웹 문서로 이동하게 되었다. 이에 따라 급속히 증가하고 있는 XML 문서에 대한 구조, 통합 및 검색을 위한 연구들이 있다. 이 논문에서는 XML 문서들에 대한 질의 처리, 검색 등을 효율적으로 처리하기 위한 기반으로써 빈발구조 중심의 XML 문서를 클러스터링 하는 방법을 제안한다. 첫째 XML 문서를 트리 구조로 표현하여 분리하고 분리된 구조들을 대상으로 빈발하게 발생하는 구조들을 추출한다. 둘째 각 XML 문서에서 추출된 빈발 구조들을 트랜잭션의 항목으로 취급하여 클러스터링을 수행한다. 클러스터링을 수행할 때 각 클러스터의 생성 및 생성된 전체 클러스터의 응집도를 함께 고려하는 주요항목 가중치를 이용한다. 셋째 기존연구와의 비교 실험을 통해 제안하는 방법의 우수성을 증명한다.

키워드 : XML 구조, 문서 클러스터링, XML 클러스터링, XML 문서

Clustering XML Documents Considering The Weight of Large Items in Clusters

Jeong Hee Hwang[†]

ABSTRACT

As the web document of XML, an exchange language of data in the advanced Internet, is increasing, a target of information retrieval becomes the web documents. Therefore, there are researches on structure, integration and retrieval of XML documents. This paper proposes a clustering method of XML documents based on frequent structures, as a basic research to efficiently process query and retrieval. To do so, first, trees representing XML documents are decomposed and we extract frequent structures from them. Second, we perform clustering considering the weight of large items to adjust cluster creation and cluster cohesion, considering frequent structures as items of transactions. Third, we show the excellence of our method through some experiments which compare with the previous methods.

Key Words : XML Structure, Document Clustering, XML Clustering, XML Document

1. 서 론

최근에 XML(External Markup Language)은 정부, 산업체 그리고 학계로부터 많은 연구와 관심을 받으며 인터넷 환경에서 데이터 저장과 전송을 위한 표준으로 자리 잡고 있다. 반구조적이고 표현의 유연성이 높은 XML 형식은 레이블을 갖는 트리 구조로 표현할 수 있고 트리의 각 노드는 XML 문서의 각 태그와 매핑 된다. 그리고 웹상에 점차 증가하고 있는 XML 문서들에 대한 연구에는 질의처리, 통합 및 정보검색 등이 있다[1, 2]. 서로 다른 구조를 갖는 XML 문서들을 대상으로 질의, 통합 및 검색을 하는 것보다는 유사한 구조를 갖는 XML 문서들을 대상으로 질의처리를 위한 인덱스 및 통합을 위한 스키마 구성을 하는 것이 더 간단하다. 그러므로 인덱스, 통합 및 정보검색의 범위를 축소시킬 수 있는 XML 문서의 클러스터링에 대한 연구들[3,4,5]이 있다.

XML 문서는 트리 혹은 그래프로 표현한다. XML 문서는 사이클을 포함하는 구조로 표현될 수 있기 때문에 복잡한 구조 관계를 가진다. 이렇게 다양한 구조의 XML로부터 공통의 서브 트리 구조를 추출하기 위해 트리 마이닝 알고리즘(TREEMINER)[6]을 제안하였고, [7]은 공통으로 발생하는 연관된 레이블의 쌍을 기반으로 연관규칙을 이용하는 그룹화 방법도 제안하였다. 그리고 XML 문서로부터 스키마를 추출하기 위하여 [8]은 문서를 데이터 그래프와 스키마 그래프로 표현하고 비트맵 인덱싱을 이용하여 스키마 구조를 발견하는 방법을 제안하기도 하였다.

유사 구조 및 공통의 구조를 검색하는 기존의 연구들

† 정 회 원 : 남서울대학교 전임강사
논문접수 : 2006년 5월 2일, 심사완료 : 2006년 9월 19일

[6,7,8]은 XML 스키마 또는 DTD의 구조정보가 있는 XML 문서들을 대상으로 한다. 그러므로 구조정보가 없는 XML 문서들에 적용하기 어렵다는 문제점이 있었다. 그리고 XML 문서들을 구조중심으로 클러스터링하기 위해서는 구조를 추출하는 방법이 필요하다. 기존 연구에서는 대부분 하나의 XML 스키마로부터 생성된 XML 문서들 또는 유사한 구조를 갖는 XML 문서들을 대상으로부터 구조적 연관성 및 빈발 구조를 추출하였다. 그러나 유사한 구조의 XML 문서 집합이 아닌 서로 다른 XML 문서들을 대상으로 구조를 분류하는 방법이 필요하다. 따라서 이 논문에서는 유사한 구조가 아닌 XML 문서들에도 적용할 수 있는 구조 중심의 클러스터링 방법을 제안한다. 이를 위해 첫째, XML 문서를 트리 구조로 표현하여 분리하고 분리된 구조들을 대상으로 빈발하게 발생하는 구조들을 추출한다. 둘째 각 XML 문서에서 추출된 빈발 구조들을 트랜잭션의 항목으로 취급하여 클러스터링을 수행한다. 클러스터링을 수행할 때 각 클러스터의 생성 및 생성된 전체 클러스터의 응집도를 함께 고려하는 주요항목 가중치를 이용한다. 셋째 기존연구와의 비교실험을 통해 제안하는 방법의 우수성을 증명한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구 및 기존 연구의 문제점을 제시하고, 3장에서는 XML 문서로부터 구조를 추출하기 위한 구조의 분리 및 추출방법을 기술한다. 4장에서는 각 클러스터의 주요항목에 대한 가중치를 고려하는 구조 중심의 클러스터링 방법을 제안한다. 5장에서는 기존 연구와의 실험 및 결과를 분석한다. 6장에서는 결론 및 향후 연구를 기술한다.

2. 관련연구

XML 문서를 클러스터링하기 위한 기초가 되는 연구로는 XML의 구조 특성을 고려하여 구조관계를 발견하는 데이터 마이닝 기법 및 XML 문서에 대한 구조 추출 방법들이 있다. 이 절에서는 이들의 연구 내용과, 기존의 XML 문서 클러스터링 방법의 문제점을 제시한다.

XML의 초기연구에는 XML을 효율적으로 저장하기 위해 데이터 마이닝 기법을 적용하기 위한 시도들이 있었다. 즉 반구조 데이터(semi-structured data)를 관계형 데이터베이스에 효율적으로 저장, 관리하기 위해 마이닝 기법을 이용하는 연구들[9, 10]이 있다. [9]는 반구조 데이터와 질의 언어로 표현되는 관계형 데이터와의 매핑을 위해 빈발 패턴 마이닝(frequent pattern mining)기법을 이용하였다. 데이터 지지도와 데이터 질의에 대한 결합 지지도를 계산하여 그들의 결합 지지도가 클수록 저장 패턴(storage pattern)의 빈발도를 증가시킨다. 이 연구는 반구조 데이터를 관계형 모델로 매핑하기 위해 두 가지의 비용함수, 즉 저장비용과 질의 비용을 함께 고려하는 최적화 문제를 다루고 있다. [10]은 반구조 데이터의 집합에서 스키마의 발견과 함께 이미 발견한 스키마를 유지하는 것에 대한 중요성을 강조하였다. 이를 위해 기존의 연관규칙에서 점진적인 규칙 생성을 위해 효율적으로 사용되었던 "negative border" 개념을 트리로 표현하고, 이것을 빈발 구조의 패턴으로 유지하기 위한 방법으로 이용하였다.

XML 문서로부터 구조를 추출하기 위한 연구로써 [11]은 연관규칙을 사용하여 빈도가 높은 문서 트리의 경로를 찾는 방법을 제안하였고, 여러 문서들에 대해 계층적으로 링크하고 있는 링크 정보를 대상으로 하였다. [1]은 빈발 태그의 트리 패턴을 발견하는 방법을 제시하였고 이를 위해 OEM(Object Exchange Model)으로 표현한다. 그리고 모든 트리에 대해 만족하는 최소의 일반화된 태그의 트리 패턴을 발견하고 이를 점차로 확장시켜 최대 빈발 태그의 트리 패턴을 탐색하는 마이닝 알고리즘을 제안하였다.

이와 같이 데이터 마이닝 기법을 이용하여 XML의 구조 특성을 고려하는 빈발 구조 추출 및 공통 구조의 추출은 유사한 구조의 XML 문서들을 자동으로 분류하는 클러스터링의 기초 과정이다. XML 문서의 구조적 분류를 위한 기존의 연구 [3-5]들에서는 일반적으로 클러스터링 알고리즘에 사용되고 있는 벡터 모델 기반의 유사도 측정 및 거리 측정 기반의 K-means 알고리즘 그리고 트리 구조의 유사성을 측정하기 위한 tree edit distance 등의 기법을 이용한다. 그러나 이러한 기법들은 XML 문서 집합의 구조 특성을 고려하여 벡터모델 및 거리 측정 방법을 정의하여 적용해야 하고, 문서의 양에 유연하게 적용할 수 없다는 문제점이 있다. 또한 [12]에서는 이러한 기존 연구의 문제점을 개선하기 위하여 트랜잭션 기반의 클러스터링을 제안하였으나 클러스터를 할당하기 위해 클러스터의 참여도를 위한 특정 기준 값을 실험을 통해 제시하여야 하고 이를 적용하는 클러스터링 수행 절차가 복잡하였다. 따라서 이 논문에서는 클러스터의 할당 방법을 개선하고, XML 문서들을 대상으로 실험하여 제안방법의 효율성을 증명한다. 제안하는 방법은 기존의 실험결과와 비교하여 클러스터링하는 시간을 단축할 수 있으며 클러스터를 할당하기 위해 반복된 실험을 통해 발견해야 하는 비교 기준값의 절차를 없앨 수 있다.

3. 클러스터링을 위한 XML 문서 전처리

XML 문서의 구조적 유사성을 판별하기 위해서는 XML 문서의 구조 경로에 대하여 서로 다른 문서들이 같은 구조를 얼마나 공유하고 있는지를 파악해야 한다. 이를 위해 각 XML 문서의 구조적 특성을 추출하고, 추출된 구조를 중심으로 공통 구조의 정도에 따라 구조간의 유사성을 파악한다. XML 문서의 구조 정보를 나타내기 위해서는 XML 문서에서 부모 및 자식과 같은 엘리먼트간의 계층적 포함관계와 형제와 같은 엘리먼트간의 수평적 관계를 모두 표현할 수 있는 모델링 과정이 필요하다. 일반적으로 XML 문서를 구성하는 정보 표현의 기본 단위는 엘리먼트이고 각 엘리먼트는 순서가 있으며 레이블이 있는 노드들로 구성된 트리 형태로 모델링된다. 이 논문에서는 XML 문서의 구조적 특성을 추출하기 위하여 트리에서 값을 갖는 엘리먼트를 중심

으로 구조를 분리한다.

XML 문서를 나타내는 하나의 트리 구조 전체에 대하여 구조를 추출하는 것은 구조 관계를 고려해야 하기 때문에 복잡하다. 그러므로 트리의 루트 노드부터 엘리먼트의 내용 값을 포함하고 있는 단말노드까지의 경로 정보를 반영할 수 있는 분리 구조 모델, Mine X_Path(mX_Path)를 통해 빈발 구조들을 추출한다. 이하에서는 Mine X_Path를 mX_Path로 표현하고 이에 대한 구조의 구성은 다음과 같다.

$$mX\_Path(ni) = \{Prefix\_path(ni), Content\_node(ni)\}$$

$mX_Path(n_i)$는 $PrefixPath(n_i)$와 $ContentNode(n_i)$으로 구성되며, 값을 갖는 엘리먼트의 단말노드 n_i에 대하여 루트부터 단말노드까지의 순서적 경로 및 동일한 부모 노드의 형제 노드의 정보를 포함하여 나타낸다. $PrefixPath(n_i)$는 루트부터 단말노드의 전노드인 n_{i-1}까지의 경로 정보를 의미하고 $PrefixPath(n_i) = (n_1/n_2/.../n_i)$으로 표현한다. $ContentNode(n_i)$는 동일한 $PrefixPath(n_i)$를 갖는 단말노드들의 정보를 나타내고 $ContentNode(ni) = \{s_1, s_2, ..., s_k\}$로 표현한다. 그러므로 하나의 XML 문서는 계층적인 구조로 이루어진 여러 개의 단말노드들로 구성되며, 분리되는 구조는 mX_Path_i의 집합으로 표현한다.

하나의 XML 문서에서 분리된 mX_Path_i집합으로부터 빈발구조를 추출하기 위해서는 구조적 연관성을 빠르게 발견하는 구조의 단순화 및 엘리먼트들을 구분하는 과정이 필요하다. 즉, 구조의 일관성있는 표현 및 동일한 의미의 엘리먼트들을 식별하기 위하여 구조를 단순화하여 표현하고 동일한 의미를 다르게 사용하는 엘리먼트의 구분이 필요하다. 이를 위해 구조들을 단순화하고 재명명하기 위해 엘리먼트 매핑 테이블을 구성하고 이를 참조한다. 엘리먼트 매핑 테이블은 구조의 이름을 단순한 알파벳으로 표현하고, 엘리먼트의 의미식별을 위해 WordNet 시소러스[13]을 이용하여 유사어(synonym)의 판별 및 동일한 의미의 엘리먼트를 동일한 구조명으로 나타낼 수 있도록 한다.

level 1		level 2		level 3		level 4		level 5	
element	rename	element	rename	element	rename	element	rename	element	rename
bookinventory	a	nation	b1	English	c1	booktitle	d1	name	e1
		book	b2	title	c2	author	d2	born	e2
						count	d3	died	e3
								nationality	e4

(그림 1) 엘리먼트 매핑 테이블

아래의 XML 문서는 XML 문서의 구조적 특징을 추출하는 방법을 보여주기 위한 예로써 제시된 문서이다.

```
<bookinventory>
  <nation>
    <English>
      (booktitle)Great Expectations(/booktitle)
    </English>
  </nation>
  <book>
    (title)Great Expectations(/title)
      <author>
        (name)Charles Dickens(/name)
        (born)1812(/born)
        (died)1879(/died)
        (nationality)English(/nationality)
      </author>
      (count)10 (/count)
  </book>
</bookinventory>
```

위의 XML 문서를 대상으로 생성된 엘리먼트 매핑 테이블은 (그림 1)이고, 이를 참조하여 내용 값이 있는 엘리먼트를 중심으로 표현한 분리 구조 집합, mX_Path은 <표 1>이다.

<표 1>과 같이 하나의 XML 문서에 대하여 단순화된 구조명과 분리 구조 모델 mX_Path에 의해 표현된 구조집합을 대상으로 빈발한 구조를 발견한다. 구조의 발견을 위해 구조의 순서와 빈발도를 함께 고려하는 [14]의 순차패턴 마이닝 기법을 이용한다.

순차 패턴 마이닝기반으로 빈발 구조를 추출하는 방법은 우선 <표 1>의 mX_Path_i에서 직선의 사각형으로 표시된 PrefixPath부분에 대하여 주어진 지지도를 만족하는 구조들에 대한 순차패턴 마이닝 기법을 적용하고, 추가적으로 점선 사각형의 ContentNode부분의 자식노드 수가 지지도를 만족하면 이에 대한 부모노드의 경로인 PrefixPath부분을 빈발구조로 포함하여 탐사과정을 마친다. 그림 2는 구조를 추출하는 전체의 과정을 보여준다.

<표 1>에 대해, 지지도를 0.5로 가정하면 적어도 2(전체의 path 수 4 * 0.5)번이상 발생하는 PrefixPath부분에 대한 빈발구조는 Len_1{a:4, b2:3, c2:3}, Len_2{a/b2 :3}, Len_3 {a/b2/c2 :2}이다(표현형태는 {구조:빈발도}를 의미한다). 그리고 ContentNode부분을 추가적으로 고려하여 Len_4{a/b2/c2/d2 : 4}의 빈발 구조를 추출한다.

각 XML 문서들로부터 순차패턴 마이닝 기법을 이용하여 발견된 빈발 구조들은 해당 문서의 대표구조로 고려하여 유사한 구조들을 그룹화 하는 클러스터링을 수행한다. 발견된 빈발구조는 Len_1부터 Len_n까지의 여러 가지 구조 길이를

<표 1> mX_Path 구조집합

LeafNode_Paths	Original Paths	mX_Path_i
1	bookinventory/nation/English/booktitle	
2	bookinventory/book/title	{(a/b1/c2), (d1)}
3	bookinventory/book/title/author/name	{(a/b2), (c2)}
4	bookinventory/book/title/author/born	{(a/b2/c2/d2), (e1,e2,e3,e4)}
5	bookinventory/book/title/author/died	{(a/b2/c2), (d3)}
6	bookinventory/book/title/author/nationality	
7	bookinventory/book/title/count	

1단계 : 각 XML 문서의 단말 노드 중심의 구조 분리(트리모델이용)
- 1.1 mX_path의 PrefixPath 부분 구성
- 1.2 mX_path의 ContentNode 부분 구성
- 1.3 mX_path 집합 생성

2단계 : mX_path 집합으로부터 빈발 구조 추출(순차패턴 마이닝적용)
- 2.1 mX_path의 PrefixPath 부분에서 빈발 구조 추출
- 2.2 mX_path의 ContentNode에서발생하는 단말 노드의 수를 고려하여 빈발 구조 추가
- 2.3 빈발 구조 추출 집합 구성

(그림 2) 빈발 구조 추출 과정

갖는다. 그러나 탐색된 모든 빈발 구조들을 클러스터링을 위한 기초데이터로 입력하는 것은 무리가 있다. Len_1과 같은 구조는 대표 구조로 고려할 만큼 중요도가 없기 때문이다. 따라서 일정 길이 비율 즉, 문서의 최대 빈발 구조 * 일정 길이 비율 σ (0<σ≤1)을 만족하는 길이의 구조만을 클러스터링을 위한 데이터로 입력한다.

위의 <표 1>에 대해 일정 길이 비율 σ를 0.6이라 할 때, 최대 빈발구조 Len_4에 이를 적용하면 0.6(4*0.6 = 2.4)이므로 빈발 구조는 Len_3이상의 구조 Len_3{a/b2/c2}, Len_4 {a/b2/c2/d2}만이 클러스터링에 입력된다.

4. 주요항목의 가중치 기반 XML 문서 클러스터링

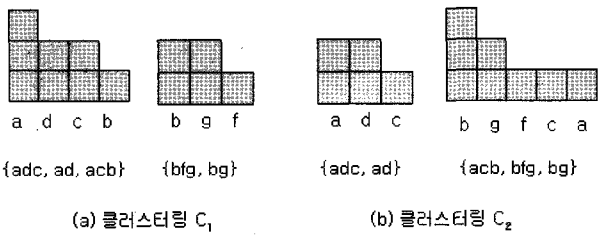
일반적인 문서 클러스터링이란 특정 문서 집합 내에 있는 각 문서들 간의 유사도를 측정하여 유사한 문서들을 그룹화하는 것이다. 이 논문에서는 클러스터를 만드는 과정에서 유사도가 높은 구조의 문서들을 하나의 클러스터로 형성하기 위해 히스토그램을 이용하는 구조기반의 클러스터링을 수행한다. 즉, XML 문서를 트랜잭션으로 가정하고 각 문서를 대표하는 빈발 구조들을 트랜잭션의 항목들로 취급하여 많은 데이터에도 유연하게 적용할 수 있는 [15]의 기본 이론을 적용한다.

양질의 클러스터들을 생성하기 위해서는 유사한 구조 특성을 갖는 XML 문서들을 같은 그룹으로 분리하는 것이다. 이를 위해서는 클러스터들을 구분하여 분류하기 위한 기준 함수가 필요하다. 우리가 제안하는 클러스터 할당 방법을 설명하기 위해 그림 3을 통해 기본 이론을 기술한다.

주어진 5개의 트랜잭션이 다음과 같이 있을 때 그림 3은 (a), (b)와 같이 클러스터링 되는 경우를 히스토그램으로 나타낸다. 히스토그램의 가로축은 클러스터를 구성하는 항목들을 의미하고 세로축은 각 항목에 대한 누적항목의 분포를 나타낸다.

t1 = { a, d, c }, t2 ={ a, d }, t3 = { a, c b }
t4 = { b, f, g }, t5 ={ b, g }

(a)의 오른쪽과 (b)의 왼쪽에 나타나는 히스토그램은 같은 형태이므로 비교를 생략하고, (a)의 왼쪽과 (b)의 오른쪽 히스토그램에서 항목 수(W)에 대한 누적항목 총합(H)의 비율(HR = H/W)을 비교해 보면 (a)는 8/4=2(항목:누적; a:3, d:2, c:2, b:1)이고 (b)는 8/5 = 1.6(항목:누적; b:3, g:2, f:1, c:1, a:1)이다. 이 비교를 통해 알 수 있는 것은 클러스터에 포함된 항목에 대한 누적항목의 비율이 클수록 양질의 클러스터를 형성한다는 것이다. 따라서 우리는 항목에 대한 누적 항목의 분포를 나타내는 히스토그램을 기반으로 양질의 클러스터를 형성한다.



(그림 3) 클러스터링 형태를 나타내는 히스토그램

각 XML 문서로부터 추출된 빈발구조들은 문서 트랜잭션에 대한 항목들로 고려한다. 클러스터의 집합은 C = $\{C_1,C_2,...,C_n\}$, 문서를 나타내는 트랜잭션 집합은 D = $\{d_1,d_2,..,d_m\}$, 빈발구조 항목들의 집합은 F = $\{f_1, f_2,..,f_r\}$로 표기한다. 그리고 각 문서를 클러스터에 할당하기 위한 기준 함수는 다음과 같이 정의한다.

[정의 1] 클러스터 할당도

전체 클러스터에 대해 각 클러스터를 구성하는 항목 수에 대한 누적항목 총합의 비율을 합산한 값을 클러스터 할당도라고 한다. 이것을 식으로 표현하면 다음과 같다.

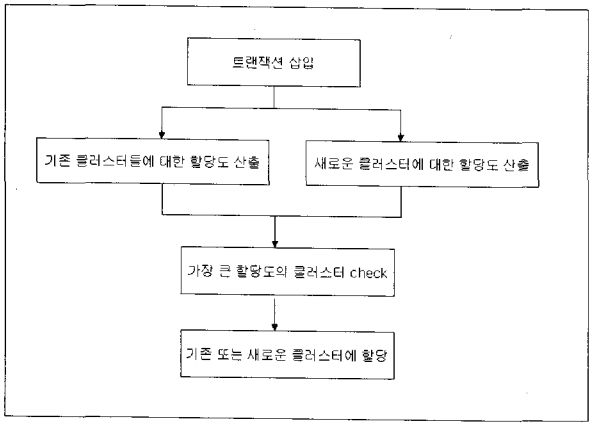
$$Profit\_C = \frac{\sum_{i=1}^{n} P(Ci)}{|C|} = \frac{\sum_{i=1}^{n} \frac{HR(Ci)}{W(Ci)}}{|C|}$$

여기서 |C|는 클러스터의 수를 의미한다. 각 클러스터에 있는 항목 수 $W(C_i)$와 항목 수에 대한 누적 항목총합 $H(C_i)$의 비율을 $HR(C_i)$로 나타낸다. $HR(C_i) = H(C_i)/W(C_i)$이므로 클러스터 할당도는 다음과 같은 식으로 나타낸다.

$$Profit\_C = \frac{\sum_{i=1}^{n} \frac{H(Ci)}{W(Ci)^2}}{|C|}$$

클러스터의 할당을 위한 기준이 되는 함수인 클러스터 할당도는 새로운 트랜잭션이 삽입되었을 때 (그림 4)와 같은 과정을 통해 클러스터 할당이 수행된다. 이에 대한 적용 예는 (예 1)이다.

(예 1) 3개의 트랜잭션들을 포함하고 있는 클러스터 C_1 = {a:3, b:3, c:1}, C_2 = {d:3, e:1, c:3}가 있을 때 새로운 트랜잭션 d_4 = {f, c}가 삽입된다고 가정한다. 클러스터 C_1에 할당되었을 경우의 클러스터 할당도는 $\frac{\frac{9}{4^2}+\frac{7}{3^2}}{2}$ = 0.67(항목:누적; a:3, b:3, c:2, f:1)이고, 클러스터 C_2에 할당되었을경우의 클러스터 할당도는 $\frac{\frac{7}{3^2}+\frac{9}{4^2}}{2}$ = 0.67(항목:누



(그림 4) 클러스터 할당 과정

적; d:3, e:1, c:4, f:1)로 같은 값의 결과를 얻는다. 그리고 새로운 클러스터(C_3)를 생성하여 할당했을 경우의 클러스터 할당도는 $\frac{\frac{7}{3^2}+\frac{7}{3^2}+\frac{2}{2^2}}{3}$ = 0.685(항목:누적; f:1, c:1)이다. 그러므로 클러스터 할당을 위해 C_1,C_2,C_3에 할당될 경우에 대한 할당도를 비교하고, 가장 큰 할당도를 나타내는 새로운 클러스터(C_3)를 생성하여 트랜잭션을 할당한다.

위의 예제를 통해 알 수 있는 것은 클러스터 할당도는 각 클러스터의 항목에 대한 누적항목의 비율만을 고려한다. 즉, 클러스터를 구성하고 있는 각 항목의 중요도를 고려하지 않고 있다. 이것은 다음과 같은 문제점을 발생시킨다. 첫째, 클러스터의 구성항목에서 중요도가 높은 항목위주로 클러스터링 하는 경우보다 더 낮은 클러스터의 응집도를 나타내는 클러스터들이 생성될 수 있다. 둘째, 새로운 클러스터에 대한 할당도는 $\frac{\text{항목수}}{\text{항목수}^2}$의 값을 포함하므로 기존 클러스터에 삽입했을 경우의 할당도보다 더 높은 값이 산출되는 경우가 많이 나타난다. 그리므로 적정 수 이상의 클러스터들을 생성하는 문제점이 발생한다. 우리는 이 문제를 해결하기 위하여 클러스터의 항목에 대한 중요도를 고려할 수 있는 주요항목과 비주요항목을 다음과 같이 정의한다.

[정의 2] 클러스터의 주요항목

클러스터 C_i에 포함된 트랜잭션 수에 대해 주어진 최소 지지도 $\Theta(0 < \Theta \leq 1)$의 비율을 만족하는 항목들을 클러스터의 주요항목이라 한다. 클러스터 C_i에 포함된 트랜잭션의 수를 $|C_i(D)|$라 할 때, sup = $\Theta * |C_i(D)|$ 이상의 지지도를 나타내는 주요항목의 집합($C_i(L)$)은 다음과 같이 표현한다.

$$C_i(L) = \{L_1, L_2, .., L_j\}$$

주요항목은 각 항목을 포함하는 트랜잭션이 최소한 sup 이상이 되는 빈발항목이다. 예를 들어, 클러스터에 포함된 6개의 트랜잭션 t1={a,b,c}, t2={a,b,c,d}, t3={a,b,c,e}, t4=[a,b,f}, t5={d,g,h}. t6={d,g,i}에 대해 최소 지지도 Θ = 0.6가 주어졌을 때, sup = 4(0.6 * 6)이고, 이를 만족하는 주요항목은 {a,b}이다.

[정의 3] 클러스터의 비주요항목

클러스터 C_i에 포함된 트랜잭션 수에 대해 주어진 최소 지지도 $\Theta(0 < \Theta \leq 1)$의 비율을 만족하지 못하는 항목들을 클러스터의 비주요항목이라 한다. 클러스터 C_i에 포함된 트랜잭션의 수를 $|C_i(D)|$라 할 때, sup = $\Theta * |C_i(D)|$를 만족하지 못하는 비주요항목의 집합($C_i(S)$)은 다음과 같이 표현한다.

$$C_i(S) = \{S_1, S_2, .., S_j\}$$

따라서 클러스터 C_i에 존재하는 모든 항목은 주요항목 또는 비주요항목에 속하므로 $C_i(F) = C_i(L) \cup C_i(S)$이다.

(정의 2)와 (정의 3)은 양질의 클러스터 생성을 위해 기준함수인 클러스터 할당도에서 고려하지 못한 각 항목의 중요도를 다음과 같은 가중치를 부여하여 고려한다. 비주요항목에 대한 가중치는 주요항목보다 낮은 가중치를 부여하여 높은 클러스터의 응집도를 유도한다.

$$\delta = \begin{cases} 1 & \text{주요항목} \\ 0 < \delta < 1 & \text{비주요항목} \end{cases}$$

위와 같은 가중치의 부여는 클러스터 할당에 있어 비주요항목보다 주요항목의 중요도를 우선적으로 고려하기 위함이다. 위와 같이 가중치를 고려하면 두개 이상의 클러스터가 같은 할당도가 산출되는 경우에 주요항목을 많이 포함하는 클러스터에 할당된다. 항목의 가중치를 고려하는 클러스터 할당과정을 (예 1)에 대해 적용하면 다음과 같다.

(예 2) 각 클러스터의 주요항목 집합을 $C_1(L)$ = {a, b}, C_2 = {d, c}라 하고, 새로운 트랜잭션 d_4 = {f, c}가 클러스터 C_1에 삽입되었을 때의 가중치에 의한 클러스터 할당도는 $\frac{\frac{7+(0.8\times2)}{4^2}+\frac{7}{3^2}}{2}$ = 0.657이고(f, c : 비주요항목), 클러스터 C_2에 할당되었을 경우의 클러스터 할당도는 $\frac{\frac{7}{3^2}+\frac{7+(1+0.8)}{4^2}}{2}$ = 0.663 이다(f :비주요항목, c :주요항목). 그리고 새로운 클러스터를 생성하여 할당했을 경우의 클러스터 할당도는 $\frac{\frac{7}{3^2}+\frac{7}{3^2}+\frac{(0.8\times2)}{2^2}}{3}$ = 0.651 이다(f, c : 비주요항목). 그러므로 가장 큰 할당도의 값을 나타내는 C_2에 할당한다.

기존 클러스터에 새로운 트랜잭션의 삽입으로 인한 항목들에 대해서만 가중치를 부여하여 클러스터 할당도를 계산한다. 기존 클러스터에 이미 포함되어 있는 항목들에 대한 가

```
알고리즘 : Create_clusters
입력 : 트랜잭션의 빈발 구조 항목
출력 : 유사한 구조 항목의 트랜잭션으로 구성된 클러스터

largest_profit = 0;
largest_cluster = 0;
count = cluster_num + 1;        // 기존 및 새로운 클러스터에 대한 할당도 검사
for i = 1 to count              // 가장 큰 값의 할당도를 갖는 클러스터 탐색
 {
   geasan_profit = (Σ_{i=1}^{n} H(Ci) / W(Ci)^2) / |C|   // 주요항목 가중치를 고려한 할당도 계산

    if  largest_profit < geasan_profit
       {  largest_profit = geasan_profit;
          largest_cluster = i;
       }
 }
if largest_cluster = count              // 새로운 클러스터 할당
  {
  c_t_su = 1;                           // 클러스터내 트랜잭션 수 1 초기화
  creating new_cluster
}
else
  c_t_su = c_t_su + 1;                  // 클러스터내 트랜잭션 수 증가
```

(그림 5) 클러스터링 알고리즘

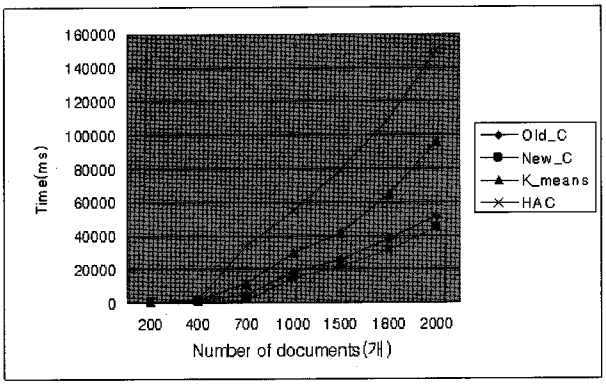
중치는 별도로 부여하지 않는다. 이는 이미 삽입될 때 고려되어 할당되었고, 모든 항목에 대한 가중치 유지는 많은 저장 공간과 처리 시간을 소비하기 때문이다. 그리고 새로운 클러스터를 생성하는 경우에는 삽입되는 항목들은 비주요항목에 대한 가중치를 적용한다. 이것은 새롭게 생성되는 클러스터의 항목들은 주요항목이라고 고려하기 어려우며, 적정 수의 클러스터 생성을 유도하기 위함이다.

(그림 5)는 이 논문에서 제안하는 가중치를 고려하는 클러스터링 알고리즘을 나타낸다.

주요항목의 가중치를 고려하는 클러스터링은 전체적인 클러스터의 응집도를 고려하면서 클러스터가 생성되며, 생성된 클러스터들은 주요항목 위주의 유사한 구조들로 구성되어 있는 XML 문서들을 포함한다. 또한 각 클러스터의 주요항목 관리는 해쉬 테이블을 이용한다. 즉, 하나의 클러스터에 포함되어 있는 모든 항목 정보를 관리하기 위한 항목 해쉬 테이블(item hash table)과 주요항목만을 관리하기 위한 주요항목 해쉬 테이블(large hash table)을 이용하여 각 클러스터의 주요항목 및 그에 대한 지지도 변화를 관리한다. 비주요 항목은 전체의 항목에서 주요항목을 제외한 나머지 항목으로 고려하여 클러스터링을 수행한다.

5. 실험 및 평가

본 논문에서 제안하는 XML 문서 클러스터링에 대한 실험을 위해, 위스콘신의 XML 데이터 뱅그[16]과 ACM SIGMID[17]에서 제공되는 XML 문서, 그리고 해당 사이트의 DTD를 가지고 IBM's AlpaWords XML generator를 이용하여 2000개를 실험 문서를 생성하였다. 각 XML 문서에서 추출된 최대 빈발 구조의 평균 길이는 5.5이고, 이에 대해 클러스터링을 위한 빈발 구조의 최소 길이는 length_rate 0.5



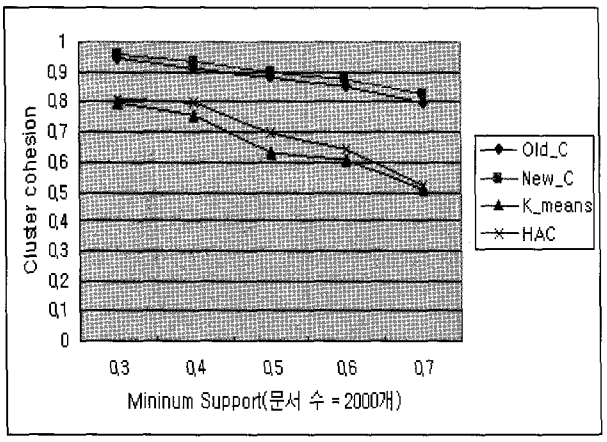
(그림 6) 클러스터링 수행시간

로 하는 min_length 3(5.5 * 0.5) 이상의 빈발 구조들을 가지고 실험하였다. 실험에서 비교 할 클러스터링 알고리즘으로는 [12]에서 제안한 알고리즘 Old_C, 그리고 기존의 XML 문서 클러스터링에서 많이 사용되고 있는 HAC(Hierarchical Agglomerative Clustering)와 K-means 알고리즘이다. 실험결과에서는 본 논문에서 제안하는 클러스터링 방법을 New_C로 표기한다. (그림 6)은 문서 수의 증가에 따라 각 클러스터링의 평균 수행시간을 비교한 결과이다.

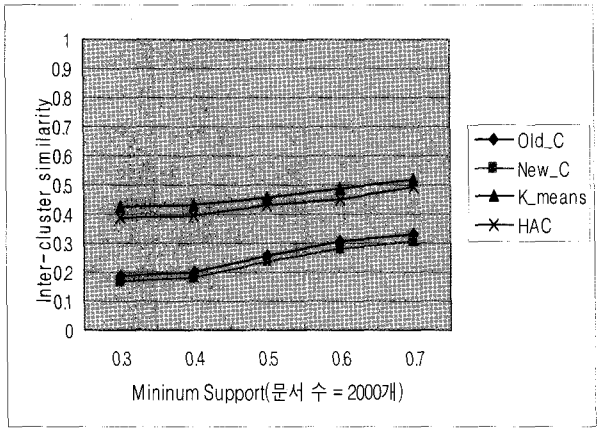
수행된 각 알고리즘은 모두 문서의 수가 증가할수록 수행시간이 증가한다. Old_C는 본 논문에서 제안한 New_C보다 작은 차이지만 더 많은 시간이 소요되는 것을 알 수 있다. Old_C는 주요항목을 이용한 클러스터의 참여도를 고려하여 클러스터의 생성과 클러스터 할당 이익을 비교하는 2 단계의 과정을 거쳐 클러스터링을 수행한다. 반면에 New_C는 각 클러스터의 주요항목에 대한 가중치를 이용하여 클러스터를 할당하므로 빠른 수행속도를 보이는 것으로 판단된다. 또한 HAC는 K_means알고리즘보다 많은 수행시간의 차이를 보였고, 수행된 알고리즘 중에서 가장 느린 속도를 보였다. HAC는 유사도를 비교하기 위해 매트릭스를 구성한다. 그리고 매트릭스의 유사도를 통해 단계적으로 클러스터들을 구성한다. 본 논문에서 제안한 클러스터의 주요항목을 구성하여 클러스터를 생성하는 과정보다 HAC에 의한 매트릭스 구성이 클러스터링 수행이 더 많은 시간이 소요되는 것임을 알 수 있다.

클러스터의 응집도와 클러스터간의 유사도에 대한 실험을 수행하였다. XML 문서에 대한 구조 중심의 클러스터링은 유사한 구조를 갖는 XML 문서들을 동일한 그룹으로 그룹화하는 것이며, 서로 다른 그룹의 XML 문서의 구조와는 차별성이 있어야 한다. 클러스터의 응집도는 각 클러스터에 포함되어 있는 전체 항목에 대해 주요항목이 차지하는 비율을 고려하여 계산하였다. 그리고 클러스터간의 유사도는 각 클러스터의 주요 항목 집합에 대한 공통의 주요항목 비율을 계산하여 측정하였다. 클러스터의 응집도는 1의 값에 가까울수록 좋은 응집도를 나타내고, 클러스터간의 유사도는 0에 가까울수록 유사성이 거의 없는 차별화된 클러스터의 생성을 의미한다.

기존의 HAC와 K-means에서는 주요항목 개념을 사용하



(그림 7) 클러스터 응집도



(그림 8) 클러스터간의 유사도

지 않으므로 HAC과 K-means알고리즘에 의해 생성된 클러스터로부터 본 논문에서 이용하는 주요항목 개념에 의해 각 클러스터에서 주요항목을 추출하여 실험을 하였다. (그림 7)은 각 알고리즘을 수행하여 이들에 대한 클러스터 응집도를 비교한 결과이다.

클러스터의 응집도는 유사한 구조의 문서들에 대한 분포 정도를 의미한다. 실험 결과를 통해 HAC와 K-means는 본 논문에서 제안하는 New_C나 Old_C에 비해 상대적으로 낮은 결과를 보였다. 클러스터 응집도의 값은 1에 가까울수록 양질의 클러스터링 결과를 나타낸다. K-means 알고리즘은 HAC보다 더 낮은 값의 클러스터 응집도를 나타내었다. 이를 통해 HAC 알고리즘은 K-means 보다 수행시간은 더 많이 소요되지만 클러스터의 응집도에서는 더 좋은 결과를 보인다는 것을 알 수 있었다. 그리고 New_C는 Old_C에 비해 클러스터 응집도가 더 좋은 결과를 보여 주었고, 이는 주요항목에 대한 가중치를 이용하는 것이 클러스터링 수행에 효율적으로 작용하는 것을 보여준 것이다.

K-means 알고리즘은 거리 계산에 의해, 그리고 HAC는 유사도 측정에 의해 클러스터들을 생성한다. 이들 알고리즘에서는 클러스터링의 수행을 종결시키기 위해 미리 주어지는 클러스터의 수 k의 값이 필요하다. 다른 알고리즘과의 동일한 조건에서의 클러스터 응집도와 클러스터간의 유사도 비교를 위해 실험에서는 New_C나 Old_C에 의해 자동으로 생성되는 평균적인 클러스터의 수 8-12를 적용하여 반복 실험하였다. (그림 8)은 클러스터간의 유사도를 비교한 실험 결과이다.

클러스터간의 유사도는 각 클러스터간의 차별성을 고려하는 것으로, 낮은 클러스터간의 유사도는 전체적으로 양질의 클러스터 생성을 의미한다. 그림 8에 의해 본 논문에서 제안하는 New_C가 다른 알고리즘에 비해 가장 낮은 값의 클러스터간 유사도를 보여준다. 그리고 HAC는 클러스터 응집도와 마찬가지로 K-means보다 더 좋은 결과의 클러스터 유사도 결과를 보여준다.

위 실험 결과에 의해 New_C가 일반적인 클러스터링 알고리즘 HAC, K-means보다 더 좋은 클러스터링 결과를 유도한다는 것을 알 수 있었다. 그리고 Old_C에서 이용하는 클러스터 참여도에 의한 클러스터 생성 및 클러스터 할당의 계산방법보다 New_C가 양질의 클러스터를 생성하는 것을 보여 주었다.

6. 결 론

본 논문에서는 XML 문서의 구조를 중심으로 유사한 구조 패턴을 갖는 XML 문서들을 그룹화하는 클러스터링 방법을 제안하였다. XML 문서는 사용자가 임의로 정의하는 엘리먼트로 이루어지는 구조적인 특성을 가지고 있다. 그리고 엘리먼트들은 XML 문서를 대표하는 키워드와 같은 역할을 하므로 구조에 초점을 두어 문서를 분류하는 연구들이 발표되고 있다. 이 논문에서 제안하는 XML 문서의 클러스터링 방법은 XML 문서를 트리로 표현하고 트리를 분리한다. 그리고 분리된 트리를 기반으로 빈발 구조의 패턴을 발견하고 이들 구조를 기초 데이터로 입력하여 클러스터링을 수행한다. 클러스터링의 수행은 트랜잭션 데이터를 취급하는 클러스터링 방식을 변형하여 수행한다. 각 클러스터들의 빈발 구조항목인 주요항목위주로 클러스터를 생성하기 위하여 주요항목에 대한 가중치를 부여하고 이를 중심으로 클러스터를 생성한다. 제안된 주요항목에 대한 가중치를 고려하는 클러스터링 방법은 클러스터링을 수행하면서 생성되는 각 클러스터들에 대한 응집도를 전체적으로 고려하는 클러스터 할당도를 이용하므로 양질의 클러스터링 결과를 유도한다는 것을 실험을 통해 알 수 있었다.

향후 연구로는 XML 문서의 클러스터링 결과를 이용하여 XML 문서의 통합을 위해 각 클러스터로부터 스키마를 추출하는 방법 및 질의 처리를 신속하게 처리하기 위한 구조적 특성을 고려하는 인덱스 기법 등에 대한 응용 연구를 수행할 것이다.

참 고 문 헌

[1] T. Miyahara, Y. Suzuki, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda, "Discovery of Frequent Tag Tree Patterns in Semistructured Web Documents," The 6th Pacific Asia Conference, Advances in Knowledge Discovery and Data Mining(PAKDD), 2002.

[2] J. T. Wang, D. Shasha, G. J. S. Chang, "Structural Matching and Discovery in Document Databases," Proceedings of the ACM SIGMOD on Management of Data, 1997.

[3] A. Doucet, H. A. Myka, "Naive Clustering of a Large XML Document Collection," Proceedings of INEX Workshop, 2002.

[4] J. Yoon, V. Raghavan, V. Chakilam, "BitCube: Clustering and Statistical Analysis for XML Documents," Proceedings of the International Conference on Scientific and Statistical Database Management, 2001.

[5] M. L. Lee, L. H. Yang, W. Hsu, X. Yang, "XClust: Clustering XML Schemas for Effective Integration," Proceedings of the ACM International Conference on Information and Knowledge Management, 2002.

[6] M. Zaki, "Efficiently Mining Frequent Tree in a Forest," Proceedings of the ACM SIGKDD International Conference, 2002.

[7] A. Termier, M. C. Rouster, M. Sebag, "TreeFinder: A First Step towards XML Data Mining," Proceedings of IEEE International Conference on Data Mining (ICDM), 2002.

[8] S. W. Kim, et al, "Indexing and Retrieval of XML-encoded Structured Documents in Dynamic Environment", Lecture Notes in Computer Science(LNCS) Vol.2480, 2002.

[9] A. Deutsch, M. F. Fernandez, and D. Suciu, "Storing Semistructured Data with STORED," Proceedings of ACM SIGMOD International Conference on Management of Data, pp.431-442, 1999.

[10] D. Katsaros, "Efficiently Maintaining Structural Associations of Semistructured Data," Panhellenic Conference on Informatics, LNCS 2563, 2003.

[11] K. Wang and H. Liu, "Discovery Typical Structures of Documents: A Road Map Approach," In ACM SIGIR Conference on Information Retrieval, 1998.

[12] J. H. Hwang, K. H. Ryu, "A Clustering Technique using Common Structures of XML Documents," KISS, Vol.32, No.6, 2005.

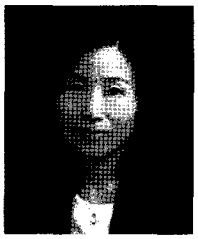
[13] http://www.cogsci.princeton.edu/~wn/wn2.0

[14] J. Pei, J. Han, B. M. Asi, H. Pinto, "PrefixSpan: Mining Sequential Pattern Efficiently by Prefix-Projected Pattern Growth," Proceedings of International Conference on Data Engineering(ICDE), 2001.

[15] Y. Yang, X. Guan, J You, "CLOPE : A fast and effective clustering algorithm for transaction data," Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002.

[16] NIAGARA query engine. http://www.cs.wisc.edu/niagara/data.html.

[17] http://www.acm.org/sigmod/record/xml, 2001.



황 정 희

e-mail : jhhwang@nsu.ac.kr

1991년 충북대학교 전산통계학과(이학사)

2001년 충북대학교 전자계산학과(이학석사)

2005년 충북대학교 전자계산학과(이학박사)

2006년~현재 남서울 대학교 전임강사

관심분야 : XML, 데이터 마이닝, 유비쿼터스 컴퓨팅, 능동 데이터베이스, 시공간 데이터베이스