클러스터의 히스토그램을 이용한 XML 문서의 점진적 클러스터링 기법

(An Incremental Clustering Technique of XML Documents using Cluster Histograms)

황 정 희 †

(Jeong Hee Hwang)

요 약 이 논문에서는 XML 문서에 대한 효율적인 검색과 통합을 위한 기초연구로써 XML 문서들에 대한 구조 중심의 클러스터링 기법을 제안한다. 기존 연구에서 문서간의 구조적 유사도를 기반으로 클러스터를 형성해 가는 것과는 다르게 많은 데이타를 빠르게 처리할 수 있는 트랜잭션 데이타를 취급하는 알고리즘을 변형하여 적용한다. 각 클러스터에 포함되어 있는 항목들에 대한 누적 분포를 나타내는 히스토그램을 이용하여 전체적인 클러스터링의 응집도를 고려하는 클러스터링을 수행한다. 기존 연구와의 실험을 통해 클러스터링 처리 시간의 향상과 양질의 클러스터를 생성하는 것을 알 수 있었다.

키워드 : XML 마이닝, XML 클러스터링, 구조 추출, XML 문서

Abstract As a basic research to integrate and to retrieve XML documents efficiently, this paper proposes a clustering method by structures of XML documents. We apply an algorithm processing the many transaction data to the clustering of XML documents, which is a quite different method from the previous algorithms measuring structure similarity. Our method performs the clustering of XML documents not only using the cluster histograms that represent the distribution of items in clusters but also considering the global cluster cohesion. We compare the proposed method with the existing techniques by performing experiments. Experiments show that our method not only creates good quality clusters but also improves the processing time.

Key words : XML Mining, XML Clustering, Structure Extraction, XML Document

1. 서 론

정보표현의 유연성으로 XML은 웹상에서 데이타 표현과 교환을 위한 표준으로 널리 사용되고 있고, XML은 임의의 태그를 사용하여 엘리먼트를 표현하고 그 엘리먼트들은 계층 구조를 갖도록 구성된다[1]. 기존의 텍스트와 다른 이와 같은 특징으로 인해 문서의 빈발 구조 및 구조 기반의 문서 분류를 위한 데이타 마이닝 기법에 대한 연구가 이루어지고 있다. 그리고 XML을 기반으로 하는 응용연구의 증가와 더불어 구조적 특성을 갖는 XML 문서들에 대한 통합과 검색에 대한 관심도 증가하고 있다. 유사한 구조 패턴을 갖는 XML 문서들을 그룹화 하는 것이 이들 연구의 실질적인 작업의 시작이다[2-4]. 그룹화된 문서들은 검색과 통합 그리고 질의를 위한 탐색 범위를 줄여주는 효과가 있기 때문이다.

기존의 문서 클러스터링 방법들[2-4]은 대부분 유사도 매트릭스를 이용하는 계층적 클러스터링과 거리 기반의 K-Means 알고리즘을 이용한다. 알고리즘 수행을 위해 문서의 특징을 추출하고 벡터 공간 모델에 의해 문서를 표현한다. 그리고 문서 벡터와 중심 벡터와의 정확한 유사도 측정을 위해 Jaccard, Euclidian, Dice 등과 같은 유사도 측정 방법을 이용한다. 그러나 유사도 측정을 위한 측정 기준 및 문서를 벡터로 표현하는 방법의 정의가 어렵고, K-Means 알고리즘은 문서 특성을 고려하여 클러스터의 수를 미리 지정해야 하는 어려움이 있다.

이와 같은 기존 알고리즘의 문제점을 해결하기 위해 이 논문에서는 다음과 같은 과정으로 클러스터링을 수행한다. 첫째, XML 문서를 트리로 표현하여 구조를 분리한다. 둘째, 분리된 구조에서 빈발하게 발생하는 구조

† 정 회 원 : 남서울대학교 컴퓨터학과 전임강사
jhhwang@nsu.ac.kr
논문접수 : 2006년 5월 2일
심사완료 : 2007년 3월 15일

들을 추출한다. 추출된 구조들은 각 문서의 특징으로 간주한다. 셋째, 각 문서를 하나의 트랜잭션으로 간주하고 빈발구조들을 트랜잭션의 항목으로 간주하여 클러스터링을 수행한다. 각 클러스터에 포함되어 있는 항목들에 대한 누적 분포를 나타내는 히스토그램에 의해 누적 비율이 높은 항목들로 이루어진 클러스터들을 생성한다. 넷째, XML 문서의 삽입과 삭제의 변화를 고려할 수 있는 점진적인 평가 방법에 의해 클러스터링을 수행한다.

이 논문에서 제안하는 클러스터링 방법은 문서들에 대한 유사도를 반복적으로 측정하는 방법을 이용하지 않으므로 빠른 처리와 많은 양의 데이타에도 유연하게 적용할 수 있다. 그리고 클러스터들의 항목에 대한 누적 분포를 나타내는 히스토그램을 기반으로 밀도 높은 항목들을 지칭하는 주요항목들의 가중치를 고려하므로 전체적으로 양질의 클러스터들을 생성한다.

이 논문의 구성은 다음과 같다. 2장에서는 XML 문서의 클러스터링과 연관된 관련연구를 기술한다. 3장에서는 클러스터링의 전처리과정으로써 XML 문서의 구조적 특성을 추출하는 방법을 기술하고, 4장에서는 구조 항목들의 분포를 나타내는 히스토그램을 기반으로 하는 클러스터링 과정을 설명한다. 그리고 XML 문서의 삽입과 삭제에 효율적으로 적용하기 위한 점진적 평가 기반의 클러스터링 방법을 기술한다. 5장에서는 실험을 통해 기존 연구와 비교 평가하고 6장에서 결론을 맺는다.

2. 관련연구

XML 문서의 클러스터링과 연관된 구조 추출 및 마이닝 기법에 대한 기존 연구를 살펴보고 이들에 대한 문제점과 해결방안을 기술한다.

2.1 XML 문서의 구조와 데이타 마이닝

XML의 초기 연구에는 XML을 효율적으로 저장, 관리하기 위해 데이타 마이닝 기법을 적용하기 위한 시도들이 있었다. 이들 연구에서는 반구조 데이타(semi-structured data)를 관계형 데이타베이스에 효율적으로 저장하고 관리하기 위해 마이닝 기법을 이용한다.

데이타 마이닝 기법을 이용하여 XML 문서의 구조 관계를 발견하기 위한 방법으로 [5]에서는 하나의 문서에서 엘리먼트들에 대한 구조 발견을 위한 intra-structured mining과 여러 문서간의 구조적 연관성 발견을 위한 inter-structured mining 기법으로 분류한다. 임의의 XML 문서를 미리 정의된 문서 클래스의 어느 클래스에 속하는지를 예측하기 위한 분류 규칙(classification)과 다양한 문서간의 유사성을 식별하기 위한 군집화(clustering) 등을 적용할 수 있음을 언급하였다. 그러나 구체적인 알고리즘을 제시하지는 않았다.

연관규칙을 사용하여 빈도가 높은 문서 트리의 경로를 찾는 방법을 [6]에서 제안하였으며, 이 연구에서는 한 문서내의 구조 정보를 기준으로 하지 않고 여러 문서들 간에 계층적으로 링크하고 있는 링크 정보를 대상으로 하는 특징을 갖는다. 공통의 서브 트리 구조를 추출하기 위해 [7]은 트리 마이닝 알고리즘(TREEMINER)을 제안하였고, 이 알고리즘은 트리에서 중첩구조(embedded tree)까지 발견할 수 있다는 특징이 있다. 그러나 이 알고리즘은 유사한 트리 구조집합에만 적용할 수 있다. [8]은 레이블 되어 있는 트리 구조의 집합에서 공통으로 발생하는 연관된 레이블의 쌍(edge)을 기반으로 연관규칙을 이용하여 그룹화하고, 그것에 대하여 최대 빈발 구조를 추출한다. 그러나 연관된 레이블의 쌍을 기반으로 하기 때문에 두개 이상의 자식노드를 갖는 다중 관계의 트리 구조는 탐색할 수 없다는 단점이 있다.

일반적으로 반구조 문서는 문서의 구조나 형식이 명확하지 않으므로 자동적인 반구조 데이타의 처리를 위해서는 구조에 의한 문서의 분류가 필요하다[6,9]. 서로 다른 DTD들을 통합할 때 적당한 통합 구조의 DTD를 찾기 위한 방법으로 [10]에서는 각 DTD를 단순화하여 엘리먼트의 유사성을 기반으로 DTD 중심의 클러스터링을 수행하는 방법을 제시하였다. 그러나 같은 응용 도메인의 DTD만을 대상으로 하므로 구조 정보가 없는 XML 문서에는 적용할 수 없다는 단점이 있다. [3]에서는 텍스트 중심의 XML 문서(text-centered XML)들에 대하여 문서의 구조를 표현하는 태그와 일반 텍스트의 내용을 가지고 클러스터링 하는 방법을 제시하였다. 문서 클러스터링을 위해 각 문서가 갖는 N개의 특성을 N차원의 벡터로 표현하고 거리 값을 기반으로 하는 K-Means 알고리즘을 이용하였다. [4]에서는 문서간의 구조에 대한 공통 구조의 존재여부를 0, 1로 표현하는 비트(bit)를 이용하였고, 이에 대한 연산자 bit-wise exclusive or operator(xOR)를 제시하여 비트 맵 인덱스에 의한 XML 문서의 클러스터링 방법을 제시하였다. 그러나 많은 양의 XML 문서에 적용할 경우에는 많은 공간을 차지한다는 문제점이 있다.

XML 문서의 트리 구조를 요약하는 변형 과정을 통해 [11]에서는 구조적 유사성을 edit distance를 이용하여 클러스터링을 수행한다. 그러나 edit distance에 의한 유사도 측정은 단지 비교하는 트리 또는 그래프의 쌍에 대한 일대일 매핑에 의해서만 정확한 비교를 수행할 수 있다는 단점이 있다. [12]에서 제시된 방법에서는 항목에 대한 가중치를 고려하지 않고 주요항목의 우선순위를 고려하기 위하여 클러스터 참여도 정의를 통해 이를 이용하였다. 그러나 클러스터링 수행 과정이 복잡하고,

반복적인 실험을 통해 적정 기준 값을 발견해야 하는 단점이 있었다.

2.2 기존 연구의 문제점 및 해결방안

XML 문서 클러스터링과 관련된 기존 연구들은 첫째, 유사 도메인의 구조를 갖는 구조 집합 및 동일 DTD의 XML 문서들을 대상으로 하는 클러스터링 방법을 제시하고 있다. 둘째, 기존의 XML 문서를 위한 클러스터링에서는 구조적 유사성 측정을 위해 계층적 집적 클러스터링 및 K-Means 알고리즘을 사용한다. 그러나 XML 문서의 특성을 고려하는 유사성 측정 및 거리 측정 기준 함수를 정의하기 어렵고, 주어진 클러스터의 수에 따라 클러스터링 결과가 달라질 수 있다는 문제점이 있다. 셋째, 지속적으로 삽입 또는 삭제되는 XML 문서들에 대한 클러스터링 방법을 제시하지 않고 있다.

이러한 기존 연구의 문제점들을 해결하기 위하여 이 논문에서는 구조 정보가 없는 XML 문서 집합에 대하여 각 문서가 갖는 구조적 특성을 추출하고 이를 이용하는 클러스터링 방법을 제시한다. 구조를 그룹화하는 과정에서 거리 및 유사도 계산에 의한 매트릭스를 이용하지 않고 많은 데이타에도 유연하게 적용할 수 있는 트랜잭션 데이타를 위한 클러스터링 알고리즘을 변형하여 적용한다. 이는 XML 문서를 트랜잭션으로 간주하고 문서의 특징을 나타내는 구조들을 트랜잭션의 항목으로 간주하여 수행하는 것으로써, 빠른 클러스터링의 처리와 더불어 생성되는 클러스터들의 질을 향상시킨다.

3. 트리 분리 및 구조 추출

클러스터링을 위해 XML 문서를 트리로 표현하고, 트리 구조를 분리하는 과정을 설명한다. 기존 연구의 [9]에서도 트리 분리 모델을 이용하는 클러스터링 방법을 제시하였다. 그러나 트리모델에서 속성까지 고려하므로써 구조 분리의 복잡도가 증가하고 속성을 포함하지 않는 구조들은 null 값을 저장하는 공간의 낭비가 발생하는 단점이 있다. 이 논문에서는 구조에 따른 문서 분류에 많은 영향을 미치지 않는 속성은 고려하지 않는다.

XML 문서의 구조적 특성을 추출하기 위하여 트리에서 값을 갖는 엘리먼트들을 중심으로 분리하며, 분리된 구조들은 mX_path 집합이라 칭한다. 그러므로 각 XML문서는 여러 개의 mX_path로 구성된다. 하나의 mX_path는 트리에서 값을 갖는 리프노드 n_i에 대하여 2개의 tuple(PrefixPath(n_i), ContentNode(n_i))로 구성된다. 여기서 PrefixPath(n_i)는 암시적인 레벨 정보를 포함하며 트리의 루트노드부터 값을 갖는 리프노드까지의 경로정보를 포함하고, ContentNode(n_i)는 실제 content를 갖는 엘리먼트들을 나타낸다.

PrefixPath(n_i)는 트리의 루트노드 n_1로부터 값을 갖는 리프노드 n_i의 전단계 노드 n_{i-1}까지의 경로(ordered sequence)들을 나타내며, PrefixPath(n_i) = {$n_1/n_2/.../n_{i-1}$}으로 표현한다. ContentNode(n_i)는 같은 PrefixPath를 갖는 리프노드로써, 값을 갖는 노드들에 대한 일련의 형제 노드들을 의미하고, ContentNode(n_i) = {$c_1,c_2,...,c_j$}으로 표현한다.

각 문서에서 분리된 구조, mX_path 집합에 포함되어 있는 엘리먼트들은 사용자 정의에 의해 다양한 용어로 구성되어 있다. XML은 사용자마다 자유롭게 엘리먼트를 정의할 수 있는 특징으로 인해 같은 의미를 나타내는 경우에도 서로 다른 용어가 사용될 수 있다. 그러므로 엘리먼트의 의미를 식별하고 이에 대한 시맨틱스를 고려하기 위하여 WordNet 시소러스[13]을 이용하여 유사어(synonym)를 판별하고 유사 의미의 엘리먼트는 같은 엘리먼트로 고려한다.

그림 1은 XML 문서 트리를 분리하는 방법을 보여주기 위한 예이며, 그림 2는 XML 문서를 트리로 표현한 것이다.

```
<book>
  <preface number = "0">
    <author key = "CS-01-03">
      <name>John Philips</name>
      <name>Willam Moore</name>
    </author>
  </preface>
  <chapter number = "1">
    <content>...</content>
  </chapter>
  <chapter number = "2">
    <author key = "IR-01" refno = "interal-02">
      <name>Frank Allern</name>
    </author>
    <content>....</content>
  </chapter>
</book>
```

그림 1 XML 문서의 예

트리로 표현된 XML 문서에서 구조를 분리하는 것은 값을 갖는 리프 노드의 엘리먼트를 기준으로 한다. 즉, 트리 구조에서 같은 레벨에 있는 자손 노드들의 순서 및 속성은 고려하지 않고 문서의 구조에 직접적으로 연관되어 있는 노드의 상하 관계를 파악하는 경로 구조를 기준으로 추출한다. 표 1은 그림 2의 트리 구조에서 분리해 낸 엘리먼트들의 구조 경로 리스트이다.

표 2는 표 1의 분리된 구조 리스트에서 사용된 엘리먼트들을 단순화시킨 엘리먼트 매핑 테이블이다.

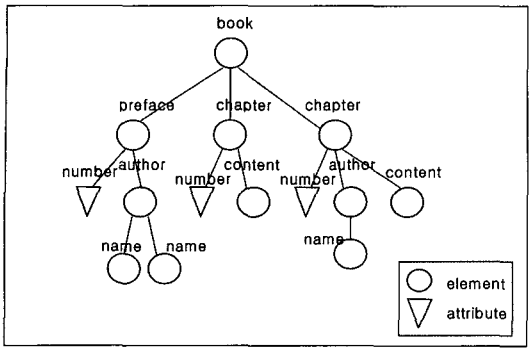


그림 2 XML 문서의 트리구조

표 1 분리된 구조 리스트

mX_paty	Original paths
mX_paty_1	<{book/preface/author}, {name, name}>
mX_paty_2	<{book/chapter}, {content}>
mX_paty_3	<{book/chapter/author}, {name}>
mX_paty_4	<{book/chapter}, {content}>

표 2 엘리먼트 매핑 테이블

Level 1		Level 2		Level 3		Level 4	
element	rename	element	rename	element	rename	element	rename
book	a1	preface	b1	author	c1	name	d1
		chapter	b2	content	c2		

트리에서 분리된 구조, mX_path의 엘리먼트들에 대하여 표 2의 엘리먼트 매핑 테이블을 참조하여 구조를 단순화시켜 표현한 것이 표 3이다. 단순화된 구조는 빈발 구조를 추출하는 데 따른 복잡도를 줄인다.

표 3 단순화된 구조 경로 리스트

mX_paty	Transformed paths
mX_paty_1	<{a1/b1/c1}, {d1, d1}>
mX_paty_2	<{a1/b2}, {c2}>
mX_paty_3	<{a1/b2/c1}, {d1}>
mX_paty_4	<{a1/b2}, {c2}>

단순화된 구조 경로로부터의 빈발 구조 추출은 먼저 mX_path를 구성하는 각 Prefix_path Sequence를 구조 시퀀스로 간주하고, 구성 엘리먼트들을 항목에 대한 순차 패턴 알고리즘[14]를 적용하여 빈발 구조를 추출한다. 그리고 Content_node Sequence를 추가적으로 고려하여 빈발구조 집합을 완성한다. 각 문서로부터 발견된 빈발 구조 패턴에 대해서, 문서의 최대 빈발 구조에 대한 일정 비율을 만족하는 구조만을 클러스터링의 입력 데이타로 사용한다.

예 1. 표 3에서 발견된 빈발 구조 항목 집합 Fp_i는 length_1의 {a1:4, b2:3, c1:2}, length_2의 {<a1/b2>:3, <a1/c1>:2}, length_3의 {<a1/b1/c1>:2}, 그리고 length_4의 {<a1/b1/c1/d1>:2}이다(구조: 빈발도를 표시한다). 이들에 대해 length_rate = 0.6 라 가정하면 발견된 빈발 구조 패턴들 중에서 일정비율을 만족하는 구조 길이 (4 * 0.6 = 2)이므로 length_2{<a1/b2>, <a1/c1>}, length_3의 {<a1/b1/c1>}, length_4{<a1/b1/c1/d1>}이 문서를 대표하는 구조로써 클러스터링을 위한 기초 데이타로 입력된다.

4. 히스토그램 기반 XML 문서 클러스터링

동일한 빈발 항목의 구조를 많이 포함하고 있는 XML 문서들을 그룹화하기 위해 각 클러스터의 항목에 대한 누적분포를 나타내는 히스토그램을 이용한다.

4.1 기본 정의

그림 3은 주어진 5개의 트랜잭션, t1 = { a, d, c }, t2 = { a, d }, t3 = { a, c, b }, t4 = { b, f, g }, t5 = { b, g } 에 대해 생성된 클러스터의 항목에 대한 누적분포를 나타내는 히스토그램이다. 히스토그램의 가로축은 클러스터를 구성하는 항목들을 의미하고 세로축은 각 항목에 대한 누적 항목의 수를 의미한다.

그림 3의 (a), (b)의 히스토그램을 비교해 보면 (a)의 오른쪽과 (b)의 왼쪽 히스토그램은 같은 형태이므로 이에 대한 비교는 생략하고 (a)의 왼쪽과 (b)의 오른쪽 그림에서 항목 수(W)에 대한 누적 항목의 수(H)의 비율

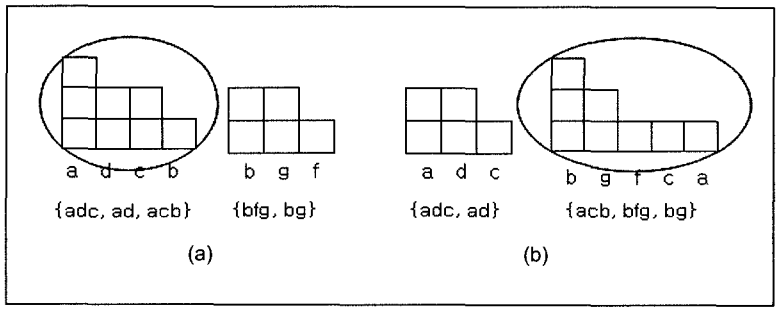


그림 3 클러스터 히스토그램

(HR = H/W)은 (a)는 8/4=2이고 (b)는 8/5 = 1.6이다. 이의 비교를 통해 알 수 있는 것은 항목 수에 대해 누적 항목의 수가 상대적으로 많아야 양질의 클러스터를 형성한다는 것이다. 즉, 밀도 기반의 클러스터링과 유사한 방식으로 동일 항목에 대한 누적 분포를 나타내는 히스토그램을 통해 양질의 클러스터 생성을 유도한다. 히스토그램을 이용하는 클러스터링을 수행하기 위해 할당 기준이 되는 할당도를 정의하고 각 클러스터의 대표 항목들을 나타내는 주요항목을 다음과 같이 정의한다. 정의에 사용되는 기호 표현으로 클러스터의 집합을 C = {C_1,C_2...,C_m}, 그리고 문서를 나타내는 트랜잭션 집합을 D = {d_1,d_2,..,d_m}으로, 빈발 구조 항목들의 집합을 F = {f_1,f_2,..,f_r}이라 표기한다.

정의 1. 클러스터 할당도(Cluster Profit)

전체 클러스터에 대해 각 클러스터를 구성하는 항목의 수에 대한 누적 항목의 비율을 합산한 값을 클러스터 할당도라고 한다. 이것은 다음의 식으로 표현한다.

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

위 식에서 |C|는 생성된 클러스터의 수를 의미한다. 클러스터 할당도는 각 클러스터에 있는 항목 수 W(C_i)와 항목 수에 대한 누적 항목수 H(C_i)의 비율 HR(C_i)이다. HR(C_i) = H(C_i)/W(C_i)이므로 클러스터 할당도는 다음과 같이 나타낸다.

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

클러스터 할당도는 클러스터링을 처리하기 위한 가장 기본이 되는 클러스터 할당을 위한 척도로 사용된다. 그러나 각 클러스터의 개별항목에 대한 중요도를 고려하기 위해 주요항목과 비주요항목을 다음과 같이 정의한다.

정의 2. 클러스터의 주요항목(Large Items)

클러스터 C_i에 포함된 트랜잭션 수에 대해 주어진 최소 지지도 $\Theta(0 < \Theta \le 1)$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 만족하는 주요항목 집합은 다음과 같이 표현한다.

$$C_i(L) = \{f_1, f_2, ..., f_j\} >= \text{sup}$$

클러스터의 주요항목은 각 클러스터를 구성하는 대표 항목을 의미하고 클러스터의 특징을 반영하는 핵심 항목들의 집합이다.

정의 3. 클러스터의 비주요항목(Small Item)

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

$$C_i(S) = \{f_1, f_2, ..., f_j\} < \text{sup}$$

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

클러스터 생성의 기준함수인 클러스터 할당도만을 이용하면 같은 값의 할당도가 산출될 때에 할당 판단이 어렵다. 그러므로 양질의 클러스터 생성과 전체적인 응집도를 함께 고려할 수 있도록 주요항목과 비주요항목에 대한 중요도를 다음과 같은 가중치를 부여하여 고려한다. 비주요항목에 대한 가중치는 1보다 작은 수치로 정한다.

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

위와 같은 가중치의 부여는 클러스터 할당에 있어 비주요항목보다 주요항목의 중요도를 우선적으로 고려하기 위함이다. 가중치를 고려하면 두개 이상의 클러스터에 대해 같은 값의 클러스터 할당도가 산출되는 경우에 가중치에 의해 주요항목이 많은 클러스터에 할당된다. 그러므로 가중치의 적용은 클러스터의 할당을 위한 추가적인 기준 정보이다. 이에 대한 적용 예를 아래에서 보여준다.

예 2. 생성된 두 개의 클러스터 C_1 = {a:3, b:3, c:1}, C_2 = {d:3, e:1, c:3}의 항목 및 항목 빈발도에 대한 각 클러스터의 주요항목 집합은 $C_1(L)$ = {a, b}, $C_2(L)$ = {d, c}이다. 새로운 트랜잭션 d_4 = {f, c}가 클러스터 C_1에 삽입되었을 때 C_1 = {a:3, b:3, c:2, f:1}이고, 삽입되는 항목 c, f는 비주요항목이므로 가중치를 고려한 클러스터 할당도는 $\frac{\frac{7+(0.8\times2)}{4^2}+\frac{7}{3^2}}{2}$ = 0.657(주요항목의 가중치는 1, 비주요항목의 가중치는 0.8을 고려; 가중치*항목 수)이다. 트랜잭션 d_4 = {f, c}가클러스터 C_2에 할당되었을 경우 C_2 = {d:3, c:4, e:1, f:1}이고 삽입되는 항목 c는 비주요항목이고 f는 주요항목이므로 클러스터 할당도는 $\frac{\frac{7}{3^2}+\frac{7+(1+0.8)}{4^2}}{2}$ = 0.663이다. 그리고 새로운 클러스터 C_3을 생성하여 할당했을 경우에 대한 클러스터 할당도는 $\frac{\frac{7}{3^2}+\frac{7}{3^2}+\frac{(0.8\times2)}{2^2}}{3}$ = 0.651이다. 따라서 가장 큰 할당도의 값을 나타내는 C_2에 트랜잭션 d_4

를 할당한다.

4.2 점진적 평가에 의한 클러스터링

지속적으로 삽입 또는 삭제되는 XML 문서들을 고려하기 위해 할당도의 차분 연산 알고리즘을 이용한다.

• 클러스터에 새로운 트랜잭션이 삽입되었을 경우

변화된 클러스터의 개별 항목의 수 W'는 삽입 이전에 존재하는 항목들에서 새로운 트랜잭션의 삽입으로 인해 추가된 항목이 있는가를 검사하고 추가된 항목의 수만큼 증가시킨다. 그리고 발생 빈도의 합 H'는 클러스터내의 개별 항목들에 대한 누적 발생 빈도의 합을 나타내므로 삽입된 트랜잭션이 포함하고 있는 항목의 수만큼 증가시킨다. 이에 대한 알고리즘은 다음과 같고, 가장 큰 삽입 차분을 나타내는 클러스터에 트랜잭션을 할당한다.

```
Int Add_Profit_C(W, H, |C|)
{
   read the new_transaction;
   H' = H + new_t.count;        //누적 항목 수 증가
   for (i=0; i < new_t.count; i++)
     {
       if ( new_t.item[i] not exist in W)
          W' = W + 1;          //새로운 항목 수 증가
     }
  return add_update = (Σ_{i=1}^{n} H(Ci)/W(Ci)^2)/|C| - (Σ_{i=1}^{n} H(Ci)/W(Ci)^2)/|C|;
}
```

그림 4 삽입 차분($Add\_Profit\_C(\Delta^+)$) 알고리즘

• 클러스터에 이미 할당된 트랜잭션이 삭제되었을 경우

갱신된 클러스터의 개별 항목의 수 W'는 트랜잭션의 삭제 이전에 존재하는 개별 항목들의 각각에 대한 빈발도에 대해서 삭제되는 트랜잭션이 포함하고 있는 항목들에 대한 빈발도를 하나씩 감소시킨 후 지지도가 0이 되면 그 항목을 포함하고 있는 트랜잭션이 존재하지 않는 것이므로 개별항목의 수 W'를 1만큼 감소시킨다. 그리고 발생 빈도의 합 H'는 기존의 H에서 삭제되는 트랜잭션의 항목 수만큼 감소시킨다. 이에 대한 알고리즘은 다음 그림 5와 같다.

이와 같은 방법으로 하나의 클러스터에 계속적으로 삽입되거나 삭제되는 트랜잭션으로 인해 영향을 받게 되는 개별 항목의 수 또는 개별 항목에 대한 누적 분포를 Add_Profit_C과 Del_Profit_C 알고리즘에 의해 클러스터 할당도의 증가분과 삭제분을 얻을 수 있다. 그리고 새롭게 삽입되는 트랜잭션에 대한 클러스터의 할당이나 이미 클러스터에 할당되어 있는 트랜잭션의 삭제로 인해 해당 클러스터에 포함되어 있는 임의의 트랜잭션을 다른 클러스터로 이동해야 하는 경우에 가장 큰

```
Int Del_Profit_C(W, H, |Ci|)
{
   read the del_transaction;
   H' = H - del_t.count;        //누적 항목 수 차감
   for (i=0; i < del_t.count; i++)
     {
       new_sup = Sup(del_t.item[i]) - 1;
       if ( new_sup == 0)
          W' = W - 1;          // 항목 차감
     }
  return del_update = (Σ_{i=1}^{n} H(Ci)/W(Ci)^2)/|C| - (Σ_{i=1}^{n} H'(Ci)/W(Ci)^2)/|C|;
}
```

그림 5 삭제 차분 ($Del\_Profit\_C(\Delta^-)$) 알고리즘

삽입 차분을 나타내는 클러스터에 트랜잭션을 할당한다.

4.3 클러스터의 주요항목 관리

클러스터의 신속한 할당을 위해서는 각 클러스터의 핵심 항목이 되는 주요항목의 관리가 중요하다. 삽입 트랜잭션에 대한 주요항목의 관리 방법과 삭제 트랜잭션에 대한 주요항목의 관리 방법은 다음과 같다.

• 삽입 트랜잭션의 경우

하나의 트랜잭션이 클러스터에 할당될 때 발생하는 주요항목의 변화를 관리하는 과정은 다음과 같다.

1. 새로운 트랜잭션의 삽입에 따라 새로운 지지도(new_sup)의 값을 계산한다.
2. 지지도에 변화(지지도 증가)가 발생하면 기존 지지도(old_sup)와 같은 지지도를 갖는 경계 항목에 대하여 새롭게 추가되는 트랜잭션에 해당 경계 항목이 존재하는지 검사하여 존재하면 주요항목으로 유지되고, 경계 항목이 존재하지 않으면 비주요 항목으로 변화된다.
3. 지지도에 변화가 없으면 새롭게 삽입되는 트랜잭션의 항목들에 대해서만 주요항목으로 변화될 가능성의 항목이 있는지 검사한다.
4. 주요항목의 변화 여부를 결정하고 나서 항목 해쉬 테이블에 삽입되는 트랜잭션이 포함하고 있는 모든 항목들의 지지도를 1씩 증가시킨다. 이 때 기존에 존재하지 않는 항목 즉, 새롭게 삽입되는 개별 항목의 지지도는 1이 된다.

• 삭제 트랜잭션의 경우

이미 할당되어 있는 하나의 트랜잭션이 해당 클러스터에서 삭제 될 때 발생하는 주요항목의 변화를 관리하는 과정은 다음과 같다.

1. 해당 클러스터에서 트랜잭션의 삭제에 따른 새로운 지지도(new_sup)의 값을 계산한다.
2. 지지도에 변화(지지도 감소)가 발생하면 기존 지지도(old_sup)보다 적은 지지도(old_sup-1)를 갖는 경계 항목에 대하여 삭제되는 트랜잭션에 해당 경

계 항목이 존재하는지를 검사하여 존재하지 않으면 주요항목으로 변화되고, 경계 항목이 존재하면 비주요 항목으로 유지된다.

3. 지지도에 변화가 없으면 삭제되는 트랜잭션의 항목들에 대해서만 주요 항목에서 비주요 항목으로 변화될 항목이 있는지 검사한다.
4. 주요항목의 변화 여부를 결정하고 나서 항목 해쉬 테이블에서 삭제되는 트랜잭션이 포함하고 있는 모든 항목들의 지지도를 1만큼 감소시킨다. 이 때 감소된 지지도가 0이 되면 그 항목은 개별 항목에서 제거된다.

각 클러스터의 주요항목은 트랜잭션의 삽입과 삭제에 따라 변화한다. 그러므로 해당 클러스터에 트랜잭션의 할당이 결정된 후에 주요항목에 대한 관리가 이루어진다.

5. 실험 및 분석

이 논문에서 제안하는 XML 문서의 클러스터링에 대한 효율성을 측정하기 위하여, 위스콘신의 XML 데이타 뱅크[15]와 ACM SIGMOD[16]에서 제공되는 XML 문서, 그리고 해당 사이트의 DTD를 가지고 IBM's Alpa-Words XML generator를 이용하여 2000개를 실험 문서를 생성하였다. 실험에서는 먼저 트리분리 모델에 대한 비교를 수행하고 클러스터링 결과를 비교한다.

5.1 트리 분리 모델 비교

XML 문서로부터 구조를 추출하기 위한 첫 번째 과정은 각 XML 문서를 나타내는 트리 구조를 분리하는 것이다. 이 논문에서 제안하는 트리 구조의 분리 방법, New_IC(mX_Path)를 기존 연구 [9]의 S_C(mPath)와 비교하기 위하여 문서의 수를 증가시키면서 다음과 같은 두 가지의 실험을 수행하였다. 첫째, 트리 구조를 분리하는 데 소요되는 시간을 측정하였다. 둘째, 하나의 XML 문서 트리에 존재하는 노드 수에 대해 분리된 구조 수의 비율을 측정하였다.

그림 6은 트리 구조를 분리하는 데 걸리는 시간을 측정한 결과이고, 그림 7은 트리의 노드 수에 대해 분리된 경로 구조의 수에 대한 비율을 보여준다.

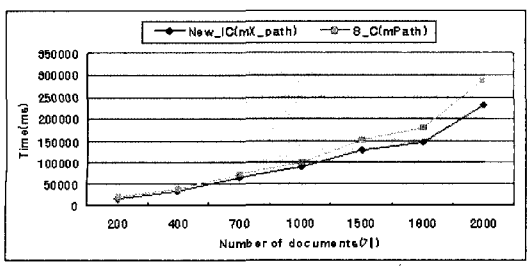


그림 6 트리 분리 처리 시간

그림 6의 실험 결과에서 기존의 분리 방식인 S_C는 New_IC보다 문서가 증가할수록 더 많은 소요 시간의 차이를 나타내는 것을 확인할 수 있다. 이것은 New_IC가 내용을 포함하는 단말 노드 중심의 엘리먼트 경로만을 고려하는 데 반해 S_C는 엘리먼트와 속성을 함께 고려하기 때문에 문서의 수가 증가할수록 속성 비율도 증가하므로 더 많은 처리 시간이 소요되는 것으로 판단된다. 실험에 사용된 2000개의 XML 문서를 나타내는 트리에서 전체 노드의 수에 대한 속성 노드의 비율은 0.184를 보였다.

그림 7은 New_IC가 S_C보다 분리된 경로의 수가 평균적으로 적게 나타난다. S_C는 트리의 분리 모델에서 튜플을 {path, attribute, content}로 구성하기 때문에 더 많은 분리 구조를 형성하기 때문이다. 즉, 속성 노드까지 고려하기 때문에 속성이 없는 구조에서는 null값으로 대치한다. 그러므로 많은 저장 공간과 시간을 필요로 한다. New_IC는 구조를 분리하는 속도가 빠르고 분리되는 구조의 수가 적으므로 구조 기반의 문서를 분류하는 데 효율적이다.

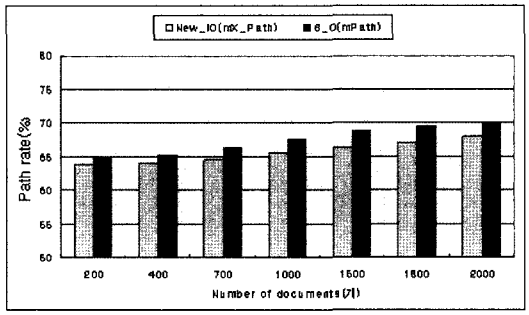


그림 7 분리된 노드의 수

5.2 클러스터링의 효율성

실험에 사용된 각 XML 문서에서 추출된 최대 빈발 구조의 평균 길이는 5.5이고, 이에 대해 클러스터링을 위한 빈발 구조의 최소 길이는 length_rate 0.5로 하는 min_length 3(5.5 * 0.5) 이상의 빈발 구조들을 문서의 대표 구조로 하여 실험하였다. 실험에서 비교 할 클러스터링 알고리즘은 [12]에서 제안한 알고리즘 Old_C, 그리고 기존의 XML 문서 클러스터링에서 많이 사용되고 있는 HAC(Hierarchical Agglomerative Clustering)와 K-Means 알고리즘을 이용하였다. 실험 결과에서는 이 논문에서 제안하는 클러스터링 방법은 New_IC로 표기한다.

그림 8은 문서의 수의 증가에 따른 각 클러스터링의 평균 수행시간을 비교한 결과이다.

Old_C는 이 논문에서 제안한 New_IC보다 작은 차이

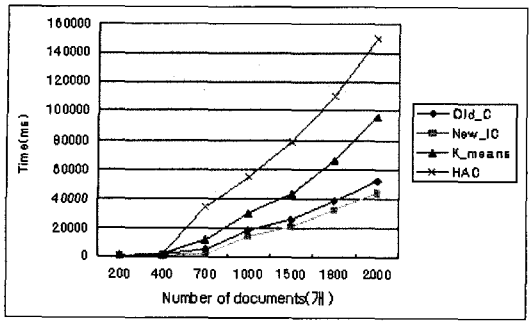


그림 8 클러스터링 수행시간

지만 더 많은 시간이 소요되는 것을 알 수 있다. Old_C는 주요항목을 이용한 클러스터의 참여도 및 클러스터 할당 이익을 비교하는 두 단계의 과정을 거쳐 클러스터링을 수행한다. 반면에 New_IC는 각 클러스터의 주요항목에 대한 가중치를 이용하는 클러스터 할당도에 의해 클러스터를 생성하므로 빠른 수행속도를 보이는 것으로 판단된다. 또한 HAC는 K-Means알고리즘보다 많은 수행시간의 차이를 보였고, 수행된 알고리즘 중에서 가장 느린 속도를 보였다.

XML 문서에 대한 구조 중심의 클러스터링은 유사한 구조를 갖는 XML 문서들을 동일한 그룹으로 그룹화하는 것이므로 서로 다른 그룹의 XML 문서들의 구조는 차별성이 있어야 한다. 그러므로 이 논문에서는 제안하는 클러스터링 방법의 평가를 위해 클러스터의 응집도와 클러스터간의 유사도에 대한 실험을 비교하였다. 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. 기존의 HAC와 K-Means에서는 주요항목 개념을 사용하지 않으므로 HAC과 K-Means알고리즘에 의해 생성된 클러스터들로부터 이 논문에서 이용하는 주요항목 개념에 의해 각 클러스터에서 주요항목을 추출하여 실험 결과를 비교하였다. 그림 9는 각 알고리즘을 수행하고 이들에 대한 클러스터 응집도를

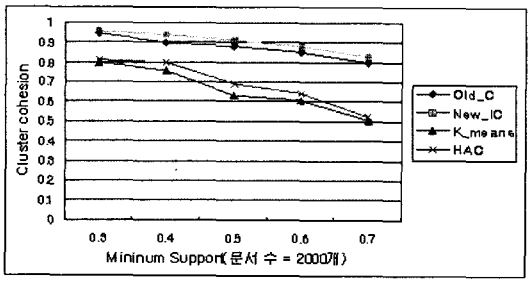


그림 9 클러스터 응집도

비교한 결과이다.

클러스터의 응집도는 유사한 구조의 문서들에 대한 분포 정도를 의미한다. 실험 결과를 통해 HAC와 K-Means는 이 논문에서 제안하는 New_IC나 Old_C에 비해 상대적으로 낮은 결과를 보였다. 클러스터 응집도의 값은 1에 가까울 수록 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_IC는 Old_C에 비해 클러스터 응집도가 더 좋은 결과를 보여 주었다. 이는 주요항목에 대한 가중치를 이용하는 것이 클러스터링 수행에 효율적인 영향을 미치는 것으로 판단된다.

K-Means 알고리즘과 HAC는 클러스터링의 수행을 종결시키기 위해 미리 주어지는 클러스터의 수 k의 값이 필요하다. 다른 알고리즘과의 동일한 조건에서 클러스터 응집도와 클러스터간의 유사도 비교를 위해 실험에서는 New_IC나 Old_C에 의해 자동으로 생성되는 평균적인 클러스터의 수 8-12를 적용하여 반복 실험하였다. 그림 10은 클러스터간의 유사도를 비교한 실험 결과이다. 클러스터간의 유사도 계산은 각 클러스터의 주요항목들에 대한 중복분포정도를 측정하였다.

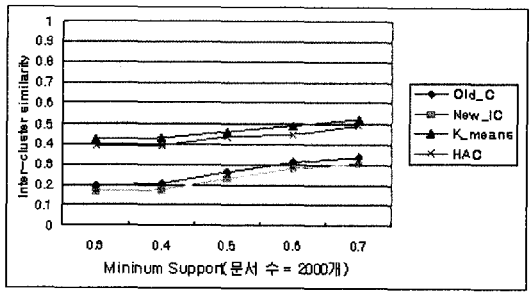


그림 10 클러스터간의 유사도

클러스터간의 유사도는 각 클러스터간의 차별성을 고려하는 것으로, 낮은 클러스터간의 유사도는 전체적으로 양질의 클러스터 생성을 의미한다. 그림 10에 의해 이 논문에서 제안하는 New_IC가 다른 알고리즘에 비해 가장 낮은 값의 클러스터간 유사도를 보여준다. 그리고 HAC는 클러스터 응집도에서와 마찬가지로 K-Means 보다 더 좋은 결과의 클러스터간의 유사도 결과를 나타내었다.

6. 결 론

XML은 인터넷을 비롯한 다양한 분야에서 정보 교환을 위한 표준으로 널리 사용되고 있다. 이 논문에서는

XML 문서의 구조를 트리로 표현하여 트리로부터 빈발 구조를 추출하였다. 빈발 구조를 추출하기 위해 구조의 순서와 연관성을 함께 고려하는 순차패턴 마이닝 기법을 이용하였다. 내용 값을 갖는 엘리먼트 즉, 트리의 단말 노드를 기준으로 루트부터의 구조경로 정보에 대하여 주어진 지지도를 만족하는 빈발 구조들을 추출한다. 추출된 빈발 구조는 XML 문서의 구조 특성을 나타내며 클러스터링을 위한 입력 데이타가 된다. 클러스터링의 수행은 각 클러스터에 대한 항목들의 누적 분포를 나타내는 히스토그램을 기반으로 트랜잭션 데이타를 취급하는 클러스터링 알고리즘을 적용하였다. 각 클러스터들의 빈발 구조항목인 주요항목 위주로 클러스터를 생성하기 위하여 주요항목에 대한 가중치를 부여하고 이를 중심으로 클러스터를 생성하였다. 또한 지속적으로 삽입, 삭제되는 XML 문서들을 고려하기 위하여 점진적인 평가에 의한 클러스터링 기법을 제안하였다. 이 논문에서 제안한 클러스터 히스토그램을 이용하여 주요항목의 가중치를 고려하는 클러스터링 방법은 생성되는 각 클러스터들에 대한 응집도를 전체적으로 고려하므로 양질의 클러스터링 결과를 유도한다는 것을 실험을 통해 알 수 있었다.

향후 연구에서는 클러스터링 결과를 효율적으로 이용할 수 있는 검색 및 통합 방법에 대한 연구를 수행할 것이고, XML 문서들로부터 유용한 지식을 추출하기 위한 연관 규칙 및 분류 규칙에 대한 연구를 계속할 것이다.

참 고 문 헌

[1] D. Braga, A. Campi, S. Ceri, M. Klemettinen, and P. Lanzi, "A Tool for Extracting XML Association Rules from XML Documents," Proceedings of IEEE-ICTAI 2002, USA, November 2002.

[2] 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.

[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] R. Nayak, R. Witt, A. Tonev, "Data Mining and XML Documents," International Conference on Internet Computing, 2002.

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

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

[8] 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.

[9] Y. Shen and B. Wang, "Clustering Schemaless XML Documents," International Conference on Ontologies, Databases and Applications of SEmantics(ODBASE), 2003.

[10] J. W. Lee, K. Lee, W. Kim, "Preparation for Semantics-Based XML Mining," Proceedings of IEEE International Conference on Data Mining (ICDM), 2001.

[11] T. Dalamagas, T. Cheng, K. J. Winkel, and T. Sellis, "Clustering XML Document by Structure," The 3rd Helenic Conference on AL. SETN, 2004.

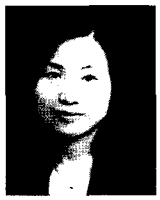
[12] J. H. Hwang, K. H. Ryu, "A Clustering Technique using Common Structures of XML Documents," KISS, Vol.32, No.6, 2005.

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

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



황 정 희

1991년 충북대학교 전산통계학과(이학사) 2001년 충북대학교 대학원 전자계산학과(이학석사). 2005년 충북대학교 대학원 전자계산학과(이학박사). 2001년 8월~2006년 2월 정우씨스템(주) GIS 연구소장. 2006년~현재 남서울 대학교 전임강사. 관심분야는 XML, 데이타 마이닝, 유비쿼터스 컴퓨팅, 능동 데이타 베이스, 시공간 데이타베이스