XML 문서의 공통 구조를 이용한 클러스터링 기법
(A Clustering Technique using Common Structures of XML Documents)

황 정 희 † 류 근 호 ††
(Jeong Hee Hwang) (Keun Ho Ryu)

요 약 인터넷의 성장으로 인해 반구조적인 문서의 표준인 XML 문서의 사용이 증가하고 있고 이에 따라 XML 문서의 통합과 검색을 위한 연구가 많이 진행되고 있다. 효율적인 문서의 통합과 검색을 위한 기초 작업은 유사 구조의 XML 문서를 클러스터링 하는 것이다. 기존 연구의 XML 문서 클러스터링에서는 문서간의 구조적 유사도를 이용하여 클러스터를 생성한다. 그러나 이러한 방법은 문서간의 구조적 유사성의 정확한 측정 기준을 만들기 어렵고, 반복적인 유사도의 비교로 인해 처리 속도가 느리다는 단점이 있다. 이러한 문제점을 개선하기 위하여 이 논문에서는 많은 데이타에도 유연하게 적용할 수 있는 트랜잭션 데이타를 위한 클러스터링 알고리즘을 적용하는 새로운 클러스터링 방법을 제안한다. 이 논문에서 제안하는 클러스터링 방법은 하나의 DTD나 XML 스키마를 공유하는 문서 집합이 아닌 스키마가 없는 다양한 구조의 XML 문서들을 대상으로 공통 구조를 이용한다. 공통 구조를 이용하기 위하여 XML 문서의 트리 모델에서 구조를 분리하여 빈발 구조를 추출하고 이를 기반으로 클러스터링을 수행한다. 아울러, 기존 연구와의 비교 및 실험을 통해 제안 기법의 효율성을 보인다.

키워드 : 문서 클러스터링, XML 클러스터링, XML 문서, 구조적 유사성

Abstract As the Internet is growing, the use of XML which is a standard of semi-structured document is increasing. Therefore, there are on going works about integration and retrieval of XML documents. However, the basis of efficient integration and retrieval of documents is to cluster XML documents with similar structure. The conventional XML clustering approaches use the hierarchical clustering algorithm that produces the demanded number of clusters through repeated merge, but it have some problems that it is difficult to compute the similarity between XML documents and it costs much time to compare similarity repeatedly. In order to address this problem, we use clustering algorithm for transactional data that is scale for large size of data. In this paper we use common structures from XML documents that don't have DTD or schema. In order to use common structures of XML document, we extract representative structures by decomposing the structure from a tree model expressing the XML document, and we perform clustering with the extracted structure. Besides, we show efficiency of proposed method by comparing and analyzing with the previous method.

Key words : Document Clustering, XML Clustering, XML Document, Structural Similarity

1. 서 론

XML(External Markup Language)[1]은 문서와 데이타를 구조적으로 표현할 수 있는 메타언어이며 플랫폼에 독립적이라는 특성 때문에 인터넷을 비롯한 다양한 분야에서 정보 교환을 위한 표준으로 널리 사용되고 있다. 이에 따라 웹상에 분포되어 있는 XML 문서들 간의 통합과 검색을 위한 연구가 많이 진행되고 있다. 그러나 효율적인 문서의 통합과 검색을 위한 기초 작업은 유사한 구조의 문서를 클러스터링 하는 것이다[2-4]. 공통 구조에 의한 문서 클러스터링은 서로 다른 구조를 갖는 XML 문서들에 대해 구조적으로 유사한 문서들을 군집화하는 것으로써, 서로 다른 구조의 전체 문서를 대상으로 검색하는 것보다 유연하고 신속한 결과를 제공하므로 기존 문서와의 통합 및 분류, 그리고 체계적인 문서 관리에 효과적이다.

· 이 연구는 산업 자원부. 한국 산업 기술 평가원 지정 청주 대학교 정보통신 연구 센터의 지원에 의한 것임
† 학생회원 : 충북대학교 컴퓨터과학과
jhhwang@dblab.chungbuk.ac.kr
†† 정 회 원 : 충북대학교 전기전자컴퓨터공학부 교수
khryu@dblab.chungbuk.ac.kr
논문접수 : 2004년 12월 14일
심사완료 : 2005년 9월 6일

기존 연구의 XML 문서 클러스터링[4,5]에서는 문서 간의 유사도를 통해 단계적으로 계층 구조를 만들어 가는 계층적 클러스터링[6]을 일반적으로 사용한다. 그러나 유사도 계산에 의한 반복적 비교로 인해 처리 속도가 느리므로 많은 양의 문서에는 적용하기 어렵다. 또한 [7,8]에서는 클러스터링을 위해 분할기법의 대표적인 기법인 k-means 알고리즘[9]를 이용하였다. 이 방법은 문서의 특성을 객체로 간주하고 이를 벡터로 표현하여 대표 객체를 중심으로 유사거리가 근접한 객체들을 단위로 분할하고 평균값(means)을 계산하여 클러스터를 구성한다. 이 방법 또한 데이타의 특성을 고려하는 구조적 유사성 계산을 위한 측정 기준이 필요하고 K개의 클러스터 수를 요구한다.

XML 문서의 클러스터링을 위해 구조를 추출하는 것은 일반 텍스트(text) 문서에서 주제어를 선정하는 것과 유사하며 이것은 클러스터링의 결과에 많은 영향을 미친다. [10-12]에서는 트리 구조 집합에 대해 공통의 빈발 패턴 구조를 찾는 방법을 제안하였다. 그러나 이 연구들은 한정된 도메인 내의 유사 DTD나 스키마에서 구조를 추출하는 것에 초점을 두고 있으며, 복잡한 계산 과정에 의해 공통의 구조를 추출하므로 많은 양의 문서에는 적용하기 어렵다.

따라서 이 논문에서는 구조 정보가 없는 XML 문서에 대하여 문서의 구조적 특성을 나타내는 엘리먼트의 경로를 기반으로 하는 새로운 클러스터링 방법을 제안한다. 이를 위해 XML 문서를 트리로 표현하고, 트리 구조로부터 값을 갖는 엘리먼트를 기준으로 하여 구조를 분리한다. 그리고 분리된 구조에서 빈발하게 발생하는 경로 구조를 추출하고 이를 기반으로 공통 구조 중심의 클러스터링을 수행한다. XML 문서는 계층적 구조의 엘리먼트를 통해 문서의 특성을 표현하므로 구조에 의한 문서 분류가 가능하다.

이 논문에서 제안하는 트리 구조의 분리 방법은 기존 연구 [5]와 유사하다. 그러나 [5]는 트리구조의 속성까지 고려하므로 많은 널(null) 값을 포함하며 분리된 구조의 수 및 수행 과정도 많은 시간이 소요된다. 그러므로 이러한 문제점을 개선하기 위하여 이 논문에서는 XML 문서의 분류에 기반이 되는 구조적 특성에 많은 영향을 미치지 않는 속성을 고려하지 않고 구조를 분리하는 모델을 제안한다. 아울러, 기존의 XML 문서 클러스터링 방법과는 다르게 대량의 데이타에도 유연하게 적용할 수 있는 트랜잭션 데이타를 위한 알고리즘을 적용하여 클러스터링을 수행한다.

논문의 구성은 다음과 같다. 2장에서는 관련연구를 소개하고 3장에서는 XML 문서를 클러스터링하기 위한 전처리 과정으로써, 트리 구조의 분리 모델을 소개하고 분리 모델에 대하여 대표 구조를 추출하는 방법을 기술한다. 그리고 4장에서는 추출된 구조를 기반으로 하는 공통 구조 중심의 클러스터링 방법을 제시한다. 5장에서는 구조 분리 모델의 평가 및 클러스터링의 실험을 평가하여 비교 분석한다. 6장에서는 결론 및 향후 연구 과제를 기술한다.

2. 관련연구

공통 구조를 이용한 XML 문서의 클러스터링은 먼저 각 문서의 구조적 특성을 추출하고 이를 기반으로 클러스터링 한다[7,13]. 그러므로 XML 문서로부터 구조적 특성을 발견하는 것은 중요한 의미를 갖는다. 기존 연구에서 제안되었던 공통 구조의 추출 방법과 XML 문서의 클러스터링에 관한 연구에 대하여 기술한다.

[11]은 공통의 서브 트리 구조 추출을 목적으로 트리 구조에 중첩되어 있는 서브 트리의 구조발견을 위해 패턴 매칭 트리를 찾는 트리 마이닝 알고리즘(TREE-MINER)을 제안하였다. 이 알고리즘은 노드를 표현하는 범위(scope)를 리스트로 작성하고 이들에 대한 조인(join) 연산을 수행하여 빈발한 공통 구조를 추출한다. 이 알고리즘은 상세한 빈발 구조의 추출이 가능하다는 특징이 있다. [14]는 유사 도메인의 트리구조 집합에 대해 연관 규칙을 사용하여 함께 자주 발생하는 에지(edge)중심의 빈발 구조를 추출한다. 그리고 주어진 지지도를 만족하는 빈발 서브 트리를 포함하는 트리들을 그룹화 하여 최대 빈발 구조를 추출한다. 그러나 트리 구조를 구성하는 에지를 기반으로 하여 빈발 구조를 발견하므로 하나의 노드가 다수의 자식노드로 구성되어 있는 다중 관계의 빈발 트리 구조는 탐색되지 않는 단점이 있다. [13]은 주어진 목표 문서의 구조와 유사한 구조의 문서를 찾는 방식으로써, XML를 트리 구조로 표현하고 트리를 구성하는 경로들에 대한 공유 정도를 기준으로 문서 구조의 유사성을 측정하는 방법을 제안하였다. 그러나 이 연구는 주어진 트리 집합에 대하여 일정한 기준 문서가 주어지고 이와 유사한 구조를 찾는 것이 목적이므로 문서 집합에서 공통의 구조를 찾는 방법과는 차이가 있다. [3,15,16]에서는 마이닝 기법을 이용하는 XML 문서의 구조 추출 방법을 하나의 문서에서 구조를 발견하기 위한 Intra-Structured Mining과 여러 문서에서 구조적 연관성을 발견하기 위한 Inter-Structured Mining으로 분류하고 있다. 그러나 구체적인 마이닝 알고리즘과 적용 방법은 제시하지 않았다. 또한, XML의 구조적 유사성 측정을 위해 트리로 표현되는 노드의 매칭 조건에서 노드 레벨 및 노드의 위치 비교에 사용되는 연산 비용 등을 측정하고 매트릭스(matrix)를 구성하는 방법을 [17]에서 제시하였으나 유

사성 측정을 위한 계산이 복잡하다는 단점이 있다.

XML 문서의 클러스터링을 위해 [18]에서는 두 개의 XML 문서 트리에서 노드들 사이의 의미 있는 매칭 식별을 위해 edit distance를 이용하였다. 이것은 트리 구조의 변화를 탐색하기 위해 일반적으로 사용되는 연산(operation)으로써, insert, delete 등을 이용한다. 그러나 이 방법은 한정된 유사 도메인의 XML 문서에만 적용할 수 있다는 단점이 있다. [7]은 내용을 갖는 엘리먼트에 대해 ePath와 문서 id를 가지고 비트맵 인덱스를 구성하여 XML 문서를 클러스터링 하는 방법을 제시하였다. 그러나 대량의 문서에 적용할 경우 많은 공간을 차지한다는 문제점이 있다. [4]는 서로 다른 DTD들을 통합할 때 적당한 통합 구조의 DTD를 찾기 위한 방법으로 DTD를 클러스터링 하는 방법을 제안하였다. 그러나 구조가 제공되지 않는 스키마가 없는 XML 문서에는 적용할 수 없다. 또한 [19]에서도 XML 문서의 클러스터링 방법을 제안하고 있으나 트리에 대한 구조의 분리에 있어 값을 갖는 모든 엘리먼트의 경로를 개별적인 구조로 취급하여 구조를 추출하므로 하나의 부모 노드에 대하여 값을 갖는 여러 개의 자식 노드들이 있는 경우에는 개별적인 구조로 분리되는 단점이 있다.

이와 같이 기존의 XML 문서 클러스터링 방법은 유사 도메인의 DTD 또는 XML 문서를 대상으로 하는 적용범위의 제한이 있었고, 문서간의 구조 유사도에 의해 클러스터링을 수행하므로 유사도의 반복적인 비교로 인해 수행시간이 많이 소요되는 단점이 있었다. 그러므로 이러한 기존 연구들의 문제점을 해결하기 위하여 이 논문에서는 클러스터링의 효율적인 수행을 위해 트랜잭션 데이타에 이용하는 [20]의 클러스터 할당 기준 방식을 적용하고 [21]의 주요항목 개념을 추가적으로 이용하는 새로운 방식을 적용한다. 이것은 XML 문서를 트랜잭션으로 간주하고 문서의 특징 구조를 항목으로 간주하여 수행하는 것으로써, 적절한 클러스터의 조절 및 클러스터의 질을 향상시킬 수 있다는 특징이 있다.

3. 스키마 없는 XML 문서에 대한 구조 추출 알고리즘

XML 문서는 사용자에 의해 자유롭게 정의되는 엘리먼트들로 구성되어 있으므로 문서에 내재되어 있는 의미를 자동으로 해석하기 어렵다. 그러나 XML 문서를 구성하는 엘리먼트의 순서와 부여된 엘리먼트 그 자체는 XML 문서의 특성을 구분할 수 있도록 한다[12,13]. 아래의 그림 1은 두 개의 XML 문서 (a)movie.xml과 (b)actor.xml은 같은 태그들로 구성되어 있지만 서로 다른 구조로 인해 다른 내용의 문서를 표현하고 있는 예를 보여준다. 즉, 엘리먼트의 구조에 따라 문서의 내용이 다르게 구분된다. 그러므로 XML 문서에 대한 분류에서는 의미적 구조를 고려해야 한다. 의미적 구조란 엘리먼트의 순서와 구조적 특성을 말한다.

XML 문서의 구조 특성을 고려하는 구조 추출 방법은 XML 문서의 구조 분리 모델에 의하여 XML 문서의 구조를 분리하고, 분리된 구조들을 대상으로 구조 추출 알고리즘을 적용하여 각 XML 문서의 구조를 추출한다. 먼저, XML 문서의 구조 분리 모델에 대하여 기술한다.

3.1 XML 문서의 구조 분리 모델

XML 문서를 구성하는 정보 표현의 기본 단위는 엘리먼트이고, 각 엘리먼트에는 순서가 있으며(ordered), 레이블이 있는 노드(labeled node)들로 구성된 트리(tree)로 표현할 수 있다[22]. 이 논문에서는 XML 문서의 구조적 특성을 추출하기 위하여 트리에 대한 새로운 구조 분리 모델을 다음과 같이 제시한다.

XML 문서를 표현하는 트리에서 구조적 특성을 추출하기 위하여 값을 갖는 엘리먼트를 중심으로 경로의 정

```
<movie>
  <title>King Kong</title>
  <year>1933</year>
  <actor>
    <name>
      <first_name>Fay</first_name>
      <last_name>Wray</last_name>
    </name>
  </actor>
</movie>
```

(a) movie.xml

```
<actor>
  <name>
    <first_name>Fay</first_name>
    <last_name>Wray</last_name>
  </name>
  <movie>
    <title>King Kong</title>
    <year>1933</year>
  </movie>
</actor>
```

(b) actor.xml

그림 1 XML 문서 예

보를 표현하며, 이러한 분리 구조의 집합을 **Mine X_path**이라 하고 이를 *mX_path*로 표기 한다. 이에 대한 정의는 다음과 같다.

정의 1. *mX_path*

*mX_path*는 트리에서 값을 포함하는 리프노드 n에 대하여 2개의 tuple(PrefixPath(n), ContentNode(n))로 구성하여 표현하는 분리된 트리구조이다. 여기서 PrefixPath(n)는 암시적인 레벨 정보를 포함하며 트리의 루트노드로부터 값을 갖는 리프 노드까지의 경로에 대한 구조 정보를 표현하고, ContentNode(n)는 실제 content를 갖는 리프 노드의 엘리먼트들을 표현한다. 상세한 표현과 이에 설명은 다음과 같다.

PrefixPath(n)는 트리의 루트노드 n_1로부터 값을 갖는 리프노드 n의 전단계 노드 n_{i-1}까지의 경로(ordered sequence)들을 나타내며, 각 노드의 순서는 노드들의 레벨을 포함하는 계층적 관계를 표현한다.

$$PrefixPath(n) = \{n_1/n_2/.../n_{i-1}\}$$

여기서, 레벨에 의한 순차적인 경로 순서는 $n_1 > n_2 > ... > n_{i-1}$라는 것을 의미한다.

ContentNode(n)는 같은 PrefixPath를 갖는 리프노드로써, 값을 갖는 노드들에 대한 일련의 형제노드들을(sibling nodes) 의미한다. 그리고 이들의 순서는 고려하지 않는다.

$$ContentNode(n) = \{c_1, c_2, ... c_j\}$$

여기서 j는 노드 n과 같은 PrefixPath(n)를 갖는 형제 노드들의 수이다.

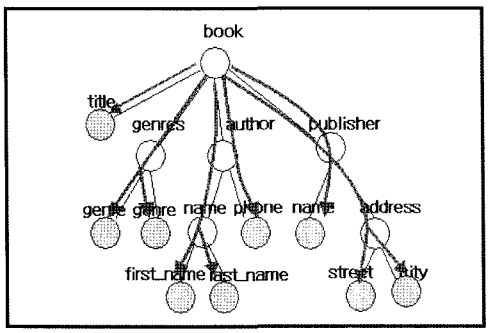
따라서 하나 이상의 서로 다른 부모 노드를 갖는 리프노드가 존재하는 XML문서, X_Doc_s는 여러 개의 *mX_path*로 구성된다. 이것을 표현하면 아래와 같고, 각 *mX_path*는 문서의 계층적 구조를 표현하는 것이며, 하나의 문서는 분리된 구조들의 집합을 표현하므로 *mX_path*의 순서는 고려하지 않는다.

$$X\_Doc_s = \{mX\_path_1, mX\_path_2, ..., mX\_path_n\}$$

하나의 XML문서를 여러 개의 *mX_path*로 분리하는 구조 분리 모델은 하나의 XML문서 전체에서 구조적 특성을 추출하는 것보다 더 빠르고 간단하게 문서들의 구조적 특성을 추출할 수 있도록 하는 장점을 제공한다. 그리고 분리된 구조, *mX_path*는 다중의 구조 관계 및 같은 부모 노드에 대한 자식 노드들의 관계를 쉽게 식별할 수 있으므로 구조적 정보를 손실하지 않으면서 구조적 특성을 신속하게 발견할 수 있도록 하는 기반이 된다.

3.2 구조 추출 알고리즘

각 문서에서 추출한 다수의 *mX_path*을 구성하는 엘리먼트들에 대한 시맨틱스(semantics)를 고려하기 위하여 wordnet 시소러스[23]를 이용하여 유사 의미의 엘리먼트는 같은 엘리먼트로 고려될 수 있도록 하는 엘리먼트 매핑 테이블을 생성한다. 그림 2의 (a)는 XML 문서를 트리 구조로 표현한 예이고, (b)는 (a)에 나타나는 엘리먼트들을 단순화하여 표현한 엘리먼트 매핑 테이블



(a) 예제 XML 문서 트리

element	rename	element	rename	element	rename
book	1	name	6	address	11
title	2	phone	7	street	12
genres	3	first_name	8	city	13
genre	4	last_name	9		
author	5	publisher	10		

(b) 엘리먼트 매핑 테이블

mX_path	Original paths	Transformed paths
mX_path_1	<{book}, {title}>	<{1}, {2}>
mX_path_2	<{book/genres}, {genre, genre}>	<{1/3}, {4,4}>
mX_path_3	<{book/author/name}, {first_name, last_name}>	<{1/5/6}, {8,9}>
mX_path_4	<{book/author}, {phone}>	<{1/5}, {7}>
mX_path_5	<{book/publisher}, {name}>	<{1/10}, {6}>
mX_path_6	<{book/publisher/address}, {street, city}>	<{1/10/11}, {12,13}>

(c) 단순화된 경로 시퀀스

그림 2 XML 문서 트리의 구조 분리 예

이다. 그리고 (a)로부터 분리된 구조, *mX_path*에 대하여 엘리먼트 매핑 테이블을 참조하여 단순화 시킨 것이 (c)이다.

하나의 XML 문서 구조에서 분리된 각각의 *mX_path*에 대하여 (c)와 같이 단순하게 부여된 경로 시퀀스는 문서의 구조 특성을 추출할 수 있는 기반이 되며, 이 논문에서는 문서의 구조적 특성을 추출하기 위하여 엘리먼트의 순서와 횟수를 함께 고려하는 순차 패턴 마이닝 알고리즘을 이용한다.

이 논문에서 제안하는 구조추출 알고리즘은 아래의 그림 3과 같이 *mX_path*를 구성하는 각 PrefixPath Sequence를 구조 시퀀스로 하고, 이들에 포함되어 있는 구성 엘리먼트들을 항목으로 간주하여 순차 패턴 알고리즘을 적용하는 첫 번째 단계와 값을 포함하는 리프노드 엘리먼트들의 시퀀스, ContentNode Sequence를 고려하는 두 번째 과정으로 이루어진다.

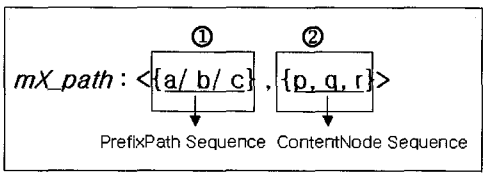


그림 3 구조 추출 처리 단계

그림 4는 제안하는 구조 추출 알고리즘을 보여준다. 분리된 구조 mX_path의 PrefixPath Sequence에 대해서는 순차 패턴 알고리즘[24]를 적용하여 빈발 패턴의 구조를 추출한다. 그리고 ContentNode Sequence에서 발생하는 엘리먼트의 빈발도에 따라 동일 엘리먼트가 같은 PrefixPath Sequence에 대하여 주어진 지지도를 만족하면 해당 엘리먼트를 포함하여 빈발패턴의 구조에 포함시키고, 동일 엘리먼트가 아닌 엘리먼트의 빈발도만이 지지도를 만족하면 해당 엘리먼트를 제외한 Prefix-Path Sequence를 빈발 패턴의 구조 집합, Fp_i에 포함한다.

각 문서로부터 mX_path_FrequentSearch 알고리즘에 의해 발견된 빈발 패턴의 구조에 대하여, 문서의 최대 빈발 구조 * 일정 길이 비율 σ (length_rate) ($0<\sigma\leq 1$)를 나타내는 최소 빈발 구조의 길이(min_length)를 만족하는 구조를 클러스터링을 위한 입력 데이타로 사용한다. 이것은 일정한 구조의 길이를 만족하지 않는 것은 문서의 특성을 나타내는 구조 항목이 될 수 없기 때문이다. 이에 대한 예는 다음과 같다.

(예) 그림 2에 대해 min_sup =2이고, length_rate = 0.6이라 가정할 때 구조 추출의 첫 단계인, *PrefixPath*에서 발견되는 빈발 구조 패턴(항목:지지도) Fp_i는 length_1 {1:6, 5:2, 10:2}, length_2{1/5:6, 1/10:2}이다. 그리고 *ContentNode*를 고려하는 두 번째 단계에서 새롭게 발견되는 빈발 구조는 {1/3/4:2, 1/5/6:2, 1/10/11:2}이므로 Fp_i에 포함된다. 예를 들면, *PrefixPath* 시퀀스에서 {1/3}은 *ContentNode*에 같은 엘리먼트(4, 4)를 두 개 포함하므로 min_sub를 만족하여 빈발항목에 추가된다. 또한 발견된 구조 패턴들 중에서 일정비율을 만족하는 구조 길이는 2(3 * 0.6 = 2)이므로 length_2 {1/5, 1/10}과 length_3{1/3/4, 1/5/6, 1/10/11}에 해당하는 구조가 문서의 대표 구조로 클러스터링에 입력된다.

4. 공통 구조를 이용한 XML 문서 클러스터링

이 논문에서는 각 문서에서 추출한 빈발 구조들에 대한 공통 구조를 이용하여 XML 문서를 클러스터링 한다. 이를 위해 우리는 XML 문서를 하나의 트랜잭션으로 가정하고 각 문서에서 추출된 빈발 구조들을 트랜잭션의 항목들로 취급하여, 트랜잭션 데이타에 대해 효율적으로 수행되는 [20]의 클러스터 할당 방식의 기본 개념을 이용한다. 또한 [20]에서 발견되는 클러스터의 관리 및 클러스터간의 유사성 문제를 해결하기 위하여 [21]의 주요항목(Large Item)개념을 추가적으로 이용하여 수행한다.

4.1 클러스터 할당

각 트랜잭션을 적당한 클러스터에 할당하기 위해 [20]에서 제시하는 클러스터 할당의 기본 개념을 그림 5를 이용하여 설명한다. 각기 다른 항목들로 구성되어 있는 5개의 트랜잭션, {(a, b), (a, b, c), (a, c, d), (d, e), (d, e, f)}에 대한 두 가지 방식의 클러스터링, (1){{ab, abc, acd}, {de, def}}, (2){{ab, abc}, {acd, de, def}}에 대한 비교는 공통 항목의 비율을 이용한다. 각 클러스터를 구성하는 개별 항목의 수(넓이)와 그에 대한 누적 항목의 수(높이)를 나타내는 그림 5의 히스토그램(histogram)에서 넓이에 대한 높이의 비율을 계산하여 클러스터의 질을 평가한다. 그러므로 개별항목에 대한 누적 항목의 비율이 높은 클러스터링 (1)의 (H = 8/4, H/W = 0.5)가 클러스터링 (2)의 (H=5/3, H/W = 0.32)보다 더 좋은 클러스터링 결과라고 볼 수 있다.

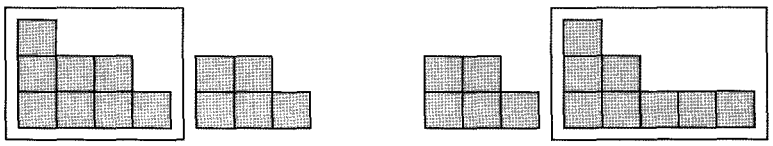
위와 같은 개념을 이용하는 클러스터 할당 방식은 하나의 클러스터에 많은 공통 항목을 포함하는 트랜잭션을 할당하도록 하는 것으로써 양질의 클러스터 생성을 유도하는 특징이 있다.

이러한 개념을 XML 문서 클러스터링에 적용하기 위하여 모든 트랜잭션에 포함되어 있는 빈발 구조 항목들의 집합을 $I = \{i_1, i_2, ..., i_n\}$, 클러스터 집합을 $C = \{C_1, C_2, ..., C_m\}$, 문서를 나타내는 트랜잭션 집합을 $T = \{t_1,$

Algorithm mX_path_FrequentSearch
Input: *PrefixPath* sequences database X of *mX_path* in an XML document(X_Doc_i), the minimum support threshold *min_sup*(the total number of *mX_paths* in a document * minimum_support Θ ($0 < \Theta \leq 1$))
Output: the complete set of frequent path Fp_i
/* *PrefixPath* sequence로부터 빈발 구조 추출 */
1. Find the set of frequent paths Fp_i in the set of *PrefixPath* sequence which is a part of each *mX_path*
   - call *PrefixSpanX(<>, 0, X)*
/* *ContentNode*를 고려하는 빈발 구조 추출 */
2. Find the set of frequent paths Fp_i considering the *ContentNode* of each *mX_path*
   for each *ContentNode* in each *mX_path*
   if the number of *ContentNode* ≥ min_sup
   if the number of the same name of *ContentNode* ≥ min_sup
   the *mX_path* including *ContentNode* is included in the set of Fp_i
   else
   the only *PrefixPath*, *mX_path* excluding *ContentNode*, is included in the set of Fp_i

Subroutine PrefixSpanX(*a*, *l*, *X/a*)
Parameter: *a*: a sequential pattern; *l*: the length of *a*; *X/a* : the *a*-projected database
output: the complete set of sequential patterns Fp_i
/**a*-projected database는 빈발 패턴의 구조 길이를 증가시키면서 *a*를 포함하는 시퀀스들로 구성된 데이터집합*/
1. Scan *X/a* once, find the set of frequent items *b* such that //빈발 항목 *b* 탐색
   - *b* can be added to the last element of *a* to form a sequential pattern
   - <*b*> can be appended to *a* to form a sequential pattern
2. for each frequent item *b* //빈발 항목 *b*를 기존의 빈발 패턴 *a*에 추가시킴
   - append it to *a* to form a sequential pattern *a'*
   - output sequential pattern *a'* //빈발 항목 *b*가 추가된 새로운 빈발 패턴 *a'* 생성
3. for each *a'*
   - construct *a'*-projected database *X/a'* //빈발 패턴 *a'*를 포함하는 데이터 집합 생성
   - call PrefixSpanX(*a'*, *l*+1, *X/a'*) //빈발 패턴의 길이를 증가시켜 빈발 패턴 구조 탐색

그림 4 빈발 구조 추출 알고리즘



클러스터링 (1) 클러스터링 (2)

그림 5 두 가지 방식의 클러스터링 히스토그램

t_2, ..., t_k}이라 표기한다. 그리고 클러스터에 트랜잭션을 할당하기 위한 기준이 되는 클러스터 할당 이익(Gain)을 다음과 같이 정의한다.

정의 2. 클러스터 할당 이익

클러스터 할당 이익은 전체 클러스터에 대해 각 클러스터를 구성하는 고유 항목에 대한 누적 항목 비율의 합이다[19]. 이것을 식으로 표현하면 다음과 같다.

$$Gain(C) = \frac{\sum_{i=1}^{m} G(C_i) \times |C_i|}{\sum_{i=1}^{m} |C_i|} = \frac{\sum_{i=1}^{m} \frac{T(C_i)}{W(C_i)^2} \times |C_i|}{\sum_{i=1}^{m} |C_i|}$$

여기서 $|C_i|$는 클러스터 C_i에 포함된 트랜잭션의 수이다. 그리고 G는 각 클러스터에서 개별항목 W에 대한 누적 항목의 비례를 나타내는 H를 나타내는 것으로써,

H = T(전체 항목 수) / W(개별 항목 수)이고 G = T/W^2이다. 클러스터 할당을 위한 기준 함수인 *Gain*은 전체적으로 각 클러스터에 대한 공통항목의 비율이 높게 구성될 때 큰 값의 클러스터 할당 이익이 산출되고, 최대의 *Gain* 값이 되도록 하는 클러스터에 트랜잭션을 할당하는 방식을 취한다. 그러나 [20]에서와 같이 개별 항목의 누적 빈도를 고려하지 않고 공통 항목의 비율인 *Gain* 만을 이용하여 클러스터에 할당할 경우, 새로운 클러스터에 대한 *Gain* = $\frac{\text{항목수}}{\text{항목수}^2}$(새로운 클러스터에는 다른 항목이 없는 상태이므로)가 되어 상당히 높은 값의 할당 이익 값이 산출된다. 따라서 이미 존재하는 클러스터에 할당하기보다는 새로운 클러스터를 생성하여 트랜잭션을 할당하게 되는 경우가 많아지므로 적당한 수보다 많은 클러스터가 생성되는 문제점이 있다. 우리는 이 문제를 개선하기 위하여 개별항목을 고려하기 위한 클러스터의 주요항목을 정의하고 이를 기반으로 하는 클러스터의 참여도를 다음과 같이 정의한다. 그리고 이를 클러스터 할당 과정에서 이용한다.

정의 3. 클러스터의 주요항목

클러스터 C_i에 대한 항목의 지지도는 C_i에서 항목 i_j(j <= n)를 포함하고 있는 트랜잭션의 수이고, 사용자가 지정한 최소 지지도, Θ(0 <Θ≤ 1)에 대해 C_i내에서 그 항목을 포함하고 있는 트랜잭션의 수가 항목의 지지도 Sup = Θ*|C_i| 이상이면 항목 i_j는 C_i의 주요항목이다[19]. 이것은 다음과 같이 표현한다.

$$C_i(L) \supset \{|C_i(i_j)| >= \text{Sup}\}$$

$|C_i(i_j)|$는 클러스터 C_i에 포함된 항목 i_j를 포함하고 있는 트랜잭션의 수를 의미한다.

정의 4. 클러스터 참여도

클러스터 참여도는 빈발 구조 항목으로 구성된 문서 t_k와 클러스터 C_j의 주요항목과의 공통 항목비율이고[19] 이것은 문서 t_k가 C_j에 속할 가능성의 정도를 나타내며 다음과 같은 식으로 표현한다.

$$p\_Allo(t_k \Rightarrow C_j) = \frac{|t_k \cap C_j(L)|}{|t_k|} \geq \omega$$

($0 < \omega < 1$: 최소 참여도)

여기서 $|t_k|$는 삽입되는 문서 t_k에 포함된 개별 항목의 수이다.

클러스터 참여도는 제안하는 클러스터링 방법에서 두 번 사용한다. 첫 번째는 할당 가능한 클러스터를 찾기 위해 모든 클러스터에 대한 *Gain*을 산출하고, 비교하는데 많은 시간이 소요되므로 이를 개선하기 위하여 사용한다. 즉, 모든 클러스터에 대한 *Gain*을 산출하지 않고 할당 가능성이 있는 클러스터에 대해서만 *Gain*을 산출하기 위하여 클러스터 참여도의 임계치 ω_1을 정의하여 사용한다. 그리고 이를 만족하는 클러스터에 대해서만 *Gain*을 산출하여 비교한다. 두 번째는 새로운 클러스터에 대한 *Gain*이 기존 클러스터에 대한 *Gain*보다 클 때 새로운 클러스터를 생성하기 이전에 주요항목과의 공통 비율을 고려하는 최소 참여도 ω_2(ω_1과는 다른 값이 주어질 수 있음)를 만족하는 기존 클러스터가 존재하는지 검사하고, 만약 존재하면 새로운 클러스터를 생성하지 않고 최대의 참여도를 갖는 기존 클러스터에 해당 문서를 할당하는 데 사용한다. 이것은 이미 존재하는 클러스터에 대한 할당 가능성을 검사하여 전체적인 클러스터의 응집도를 고려하면서 적정 수의 클러스터 생성을 유도하기 위함이다. ω_2가 크면 클러스터 생성 가능성이 줄어들고, ω_2를 작게 하면 클러스터의 생성 가능성이 커지는 반면 클러스터의 응집도는 작아질 수 있다.

또한, 클러스터링은 클러스터내의 유사도를 높이고 클러스터간의 유사도를 낮게 하여 그룹화하는 것이 목적이므로, 클러스터링의 정확성 결과를 측정할 수 있는 기준이 필요하다. 그러므로 클러스터내의 유사밀도를 나타내는 클러스터의 응집도와 클러스터간의 차별성을 나타내는 클러스터간의 유사도의 측정 기준을 다음과 같이 정의한다.

정의 5. 클러스터 응집도

클러스터 C_i의 응집도 $Coh(C_i)$는 클러스터 C_i에 포함된 전체 항목 $T(C_i)$에 대해 주요항목이 차지하는 비율이다[19]. 이것은 다음과 같이 계산하고 1의 값에 가까울수록 좋은 응집도를 나타낸다.

$$Coh(C_i) = \frac{C_i(L)}{T(C_i)}$$

클러스터에서 주요항목의 비율이 높다는 것은 같은 항목이 클러스터에 많다는 것이며, 이것은 공통의 구조를 갖는 문서들이 많이 할당되어 있다는 것을 의미한다. 그러므로 다른 클러스터와 비교하여 응집도가 크다는 것은 공통 구조를 포함하는 유사 구조의 문서가 더 잘 밀집되어 있는 클러스터를 말한다.

정의 6. 클러스터간의 유사도

클러스터 C_i, C_j에 포함되어 있는 주요 항목 중심의 유사도는 주요 항목 집합에 대한 공통의 주요항목의 비율을 클러스터 C_i, C_j의 유사도라 한다[19]. 이것은 다음과 같은 식에 의해 계산하고 0에 가까울수록 유사성이 거의 없는 클러스터이다.

$$Sim(C_i, C_j) = \frac{L(C_i \cap C_j) \times \frac{|L(C_i \cap C_j)|}{|L(C_i + C_j)|}}{C_i(L) + C_j(L)}$$

이 때 $L(C_i \cap C_j)$는 공통 항목에 대한 각 클러스터에서의 발생 횟수, $|L(C_i \cap C_j)|$은 공통의 주요 항목들의 누적 발생 횟수, $|L(C_i + C_j)|$는 주요항목의 전체 누적 횟수를

나타낸다. 클러스터간의 유사도는 각 클러스터간의 차별성의 정도를 의미하며, 낮은 클러스터간의 유사도는 전체적으로 양질의 클러스터 생성을 나타낸다.

4.2 점진적 클러스터링

초기 클러스터링의 결과에 대해 새로운 XML 문서가 삽입될 경우, 3장에서 제시한 방법에 의하여 구조 특성을 먼저 추출하고 이를 적당한 클러스터에 할당하기 위한 방법으로 기존 클러스터에 대하여 삽입되는 트랜잭션의 항목들에 대한 클러스터 할당 이익의 변화 정도를 아래의 식 (1)의 차분 연산을 이용하여 계산하고 가장 큰 차분의 클러스터에 할당하도록 한다. 이 논문에서는 하나씩의 XML 문서 삽입에 대한 클러스터링 처리를 가정한다.

$$Diff\_Gain(\Delta^+) = New\_Gain(C_i) - Old\_Gain(C_i)$$
$$= \frac{T(C_i)}{W(C_i)^2} \times (|C_i|+1) - \frac{T(C_i)}{W(C_i)^2} \times |C_i| \quad (1)$$

여기서 기존 클러스터에 대한 새로운 트랜잭션의 삽입에 대하여 고유 항목의 수, 총 항목의 수를 W′, T′로 표시하고, 식 (1)에 의하여 기존 클러스터 할당 이익에 대한 차분을 계산한다.

그림 6은 각 XML 문서에서 추출된 구조적 특성을 기반으로 차분 연산을 이용하여 XML 문서를 클러스터링 하는 절차를 나타낸 것이고, 이에 대한 알고리즘은 그림 7에서 보여 준다. 삽입되는 트랜잭션에 대한 클러스터의 할당은 주어진 클러스터 참여도(ω_1)를 만족하고, 이미 존재하는 클러스터들 중 가장 큰 차분 값을 갖는 클러스터에 할당한다. 그러나 만약 새로운 클러스터를 생성하는 경우에 대한 *Gain* 값이 더 클 경우에, 기존 클러스터에 대한 참여도가 클러스터의 응집도를 고려한

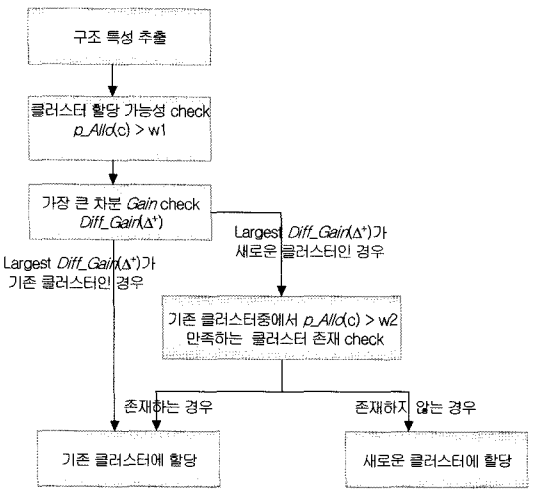


그림 6 차분 연산을 이용한 클러스터링 흐름도

■ **Insert transaction t**
extract representative structure using sequence pattern mining;
while not end of the existing cluster and $p\_Allo(C) >= w_1$ //(w_1=0.2)
  find a cluster(C_i) maximizing *Diff_Gain*(C);
  find a cluster(C_j) maximizing *p_Allo*(C);
  if new cluster *Gain*((C_k) > *Diff_Gain*(C_i)
    if $p\_Allo(C_j) >= w_2$ // (w_2=0.5)
      allocate t to an existing cluster C_j;
    else allocate t to an new cluster C_k;
  else allocate t to an existing cluster C_i;

그림 7 XML 문서의 삽입에 대한 점진적 클러스터링 알고리즘

기준치(ω_2)를 만족한다면 새로운 클러스터를 생성하지 않고 기존 클러스터에 할당하므로 생성되는 클러스터의 수를 조절할 수 있다.

클러스터의 신속한 할당과 더불어 각 클러스터의 핵심 항목이 되는 주요항목의 관리도 중요하다. 이를 위해 하나의 클러스터에 포함되어 있는 모든 항목에 대해 지지도를 유지할 수 있는 항목 해쉬 테이블(item hash table)과 주요항목만을 유지하는 별도의 주요항목 해쉬 테이블(large hash table)을 이용하여 각 클러스터의 주요항목 및 그에 대한 지지도 변화를 관리한다. 주요항목 해쉬 테이블을 별도로 관리하는 이유는 새로운 트랜잭션이 삽입되면 클러스터의 할당을 위해 각 클러스터의 주요항목과의 공통 비율을 고려하는 클러스터의 참여도를 계산해야 하는데 이 때 클러스터내의 모든 항목에 대한 지지도를 비교하지 않고 주요항목의 해쉬 테이블에 있는 항목만을 비교하면 신속하게 할당 여부를 결정할 수 있기 때문이다.

새로운 트랜잭션의 삽입에 따른 주요항목과 비주요항목의 변화는 해당 클러스터에 속하는 트랜잭션의 수에 따라 주요 항목이 변화하므로, 효율적인 주요항목의 관리는 주요항목과 비주요항목의 경계 항목, 즉 비주요항목에서 주요항목으로, 또는 주요항목에서 비주요항목으로 변화될 가능성이 있는 경계 항목(border item)에 대한 조사를 통해 주요항목의 변화를 빠르게 결정할 수 있다. 이에 대한 처리 과정은 다음과 같다.

1. 새로운 트랜잭션의 삽입에 따라 새로운 지지도(new_sup)의 값을 계산한다.
2. 지지도에 변화가 발생하면 기존 지지도(old_sup)와 같은 지지도를 갖는 경계 항목에 대하여 새롭게 추가되는 트랜잭션에 해당 항목이 존재하면 주요항목으로 유지되며, 존재하지 않으면 비주요항목으로 변화된다.
3. 지지도에 변화가 없으면 새롭게 삽입되는 트랜잭션의 항목에 대해서만 주요항목으로의 변화 가능성을 조사

한다.

4. 삽입되는 모든 항목에 대해 항목 해쉬 테이블의 지지도를 1씩 증가시킨다.

5. 실험 및 비교

XML 문서에서 구조를 추출하는 것은 구조적 클러스터링을 위한 기반이 되므로 트리 구조의 분리 방식은 중요하다. 그러므로 이 장에서는 제안하는 XML 문서의 트리 구조에 대한 분리 방식의 실험 및 클러스터링의 효율성을 측정하는 실험에 대하여 기술한다. 실험에 사용된 문서는 위스콘신의 XML 데이타 뱅크[25]에서 제공하는 스키마 없는 XML 문서 300개를 가지고 수행하였다.

먼저, 기존에 제시된 XML 문서의 트리에 대한 구조 분리방식 [5](S1_C)와[19](S2_C)를 이 논문에서 제안하는 mX_Path(XML_C)와 비교하기 위하여 DOM 파서를 이용하여 XML문서를 파싱하였고, 다음과 같은 두 가지의 실험을 수행하였다. 첫째, 문서의 수를 50, 100, 200, 300씩 증가시키면서 구조를 분리하는 데 걸리는 시간을 측정하였다. 둘째, 전체의 노드 수에 대하여 분리된 구조의 수에 대한 비율을 측정하였다.

그림 8은 구조를 분리하는 데 걸리는 시간을 측정한 결과이고, 그림 9는 전체 노드에 대한 분리된 구조의 비율을 보여준다.

그림 8의 실험 결과는 문서의 증가에 따른 처리 시간의 증가를 보이고 있으며 기존의 S2_C가 가장 빠르게

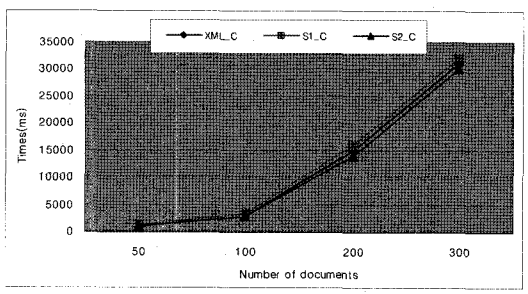


그림 8 XML 문서의 트리에 대한 구조 분리 수행시간

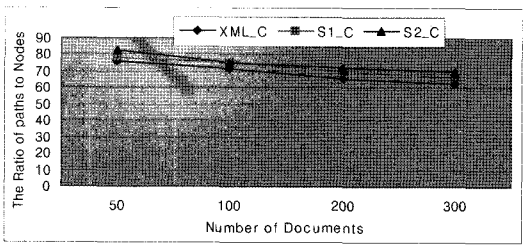


그림 9 전체 노드 수에 대한 분리된 구조의 비율

구조를 분리하는 것으로 나타난다. S2_C의 분리 방식을 그림 2의 (a)트리를 이용하여 예를 보이면 {book/author/name/first_name}, {book/author/name/first_name} 등과 같이 같은 조상의 경로를 갖는 리프노드에 대해서도 서로 다른 개별 구조로 분리한다. 그러므로 XML_C와 S1_C가 노드의 다중 관계 즉, 부모 노드가 두 개 이상의 자식 노드를 가질 경우에 대한 구조를 고려하는 것보다 더 빠르게 수행된다. 그러나 S2_C의 분리 방식은 중복 노드 경로들을 포함하는 많은 수의 구조로 분리되는 단점이 있다.

또한 S1_C가 XML_C보다 더 많은 시간이 소요되는 것은 XML_C가 엘리먼트를 나타내는 노드만을 고려하는 데 반해 S1_C는 엘리먼트와 속성을 함께 고려하기 때문이다. 실험에 사용된 300개의 XML문서를 나타내는 트리에서 전체 노드의 수에 대한 속성 노드의 비율은 0.157을 보였다.

그림 9는 전체 노드의 수에 대하여 분리된 구조의 비율을 나타낸 것으로써, XML_C가 S1이나 S2보다 분리되는 구조의 수가 가장 작게 나타난다. S2는 값을 갖는 리프노드에 대하여 루트 노드부터 해당 노드까지의 경로를 기본 구조로써 분리한다. 그러므로 임의의 부모 노드가 값을 갖는 두 개 이상의 자식 노드를 포함할 경우에 자식 노드의 수만큼 구조가 분리된다. 그러나 XML_C나 S1의 경우에는 다수의 리프 노드를 자식 노드로 포함 할 경우에도 하나의 경로로 구조가 분리될 수 있기 때문에 S2보다 분리되는 구조의 수가 적다. 또한 S1이 XML_C보다 더 많은 구조로 분리되는 이유는 S1은 구조를 분리하는 모델의 튜플은 3가지, {path, *attribute, content*}로 구성된다. 그러므로 임의의 노드가 속성과 하위의 자손 노드를 포함하는 부모노드가 같은 레벨에 존재하는 경우에는 루트로드로부터 속성까지의 경로는 하나의 구조로 분리되며, 이 때 content 부분에는 널 값으로 대치한다. 그리고 하위의 자손 노드를 포함하는 노드는 다른 구조로 분리된다. 그러므로 XML_C보다 많은 분리 구조가 생성되는 것이며, 속성을 고려하기 때문에 많은 널 값을 포함하게 되므로 저장 공간을 낭비하게 되는 단점이 있다.

위 실험을 통하여 이 논문에서는 제안하는 2개의 튜플{PrefixPath, ContentNode}로 구성되는 트리 구조의 분리 모델의 특성을 비교 설명하였다. 즉, 전체적인 트리 구조에 많은 영향을 주지 않는 속성을 고려하지 않으므로, 빠른 구조의 분리 및 분리되는 구조의 수도 작다는 특징을 보여주었다. 제안하는 분리 방식의 특징은 분리된 구조 자체를 통해 임의의 노드에 대한 다중 관계의 식별이 가능하고, 분리된 구조의 수가 작으므로 구조 추출의 시간 비용을 줄일 수 있다.

이 논문에서 제안하는 클러스터링의 효율성을 측정하기 위한 실험 내용은 다음과 같다.

실험에 사용된 300개의 XML 문서에서 구조적 특성을 추출하였다. 각 문서에서 추출된 최대 빈발 구조의 길이는 평균 5.3이고 이에 대한 최소 빈발 구조의 길이 min_length를 3(5.3* 0.5)으로 3이상의 빈발 구조를 문서의 대표 구조 항목으로 포함하였다. 실험 비교를 위한 알고리즘으로 기존 XML 문서의 클러스터링에서 많이 사용되는 HAC(Hierarchical Agglomerative Clustering)와, 제안하는 클러스터링과 가장 밀접한 관련이 있는 [20]의 CLOPE를, 제안하는 알고리즘 XML_C와 비교하는 실험을 수행하였다.

그림 10은 문서 수의 증가에 따른 클러스터링의 평균 수행시간을 비교한 것이다.

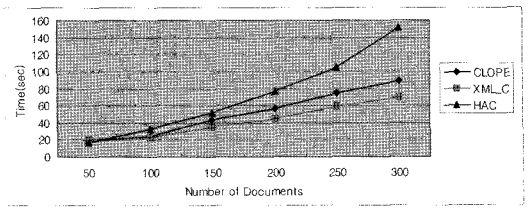


그림 10 수행시간

3개의 알고리즘 모두 문서의 수가 증가할수록 수행시간이 증가하는 것을 알 수 있다. 그러나 CLOPE이 XML_C보다 평균적으로 1.2배 더 많은 수행시간이 소요되는 것에 비해 HAC는 문서의 수가 증가할수록 XML_C와 급격한 차이를 나타내는 것을 확인할 수 있다. 이것은 문서의 수가 적은 실험의 초기에는 XML_C에서 클러스터에 대한 주요항목을 구성하는 시간과 HAC에서의 유사도 메트릭스의 구성 시간이 비슷하게 소요되는 것임을 알 수 있다. HAC에서의 유사도는 [5]에서와 같은 방식으로 측정하였다. 그러나 문서의 수가 증가할수록 XML_C는 클러스터의 전체 항목에 대한 비교가 아닌 주요항목에 의한 클러스터의 참여도를 이용하여 할당 가능한 클러스터를 검사한다. 그러므로 모든 클러스터에 대해 반복적인 비교를 수행하는 CLOPE, HAC보다 상대적으로 많은 수행시간을 줄일 수 있는 효과가 있다. 반면에 HAC는 유사도 매트릭스를 구성하는 시간과, 클러스터의 병합과정에서 매트릭스를 이용하여 문서의 유사도를 비교하는 과정에서 많은 시간이 소요되는 것으로 판단된다.

또한 실험 문서 300개에 대한 클러스터링의 결과를 측정하기 위하여 각 알고리즘을 수행한 후 클러스터의 유효성을 나타내는 정의 5의 클러스터의 응집도와 정의 6의 클러스터간의 유사도에 대한 측정치를 비교하였다. 클러스터 응집도와 클러스터간의 유사도의 측정에는 클러스터의 주요항목을 이용하는 데 CLOPE과 HAC에서는 주요항목 개념을 사용하지 않으므로, CLOPE과 HAC의 클러스터링 수행 결과로 생성된 클러스터들에 대하여 지지도를 만족하는 주요항목을 XML_C와 같은 방식으로 추출하고 이것을 정의 5의 응집도와 정의 6의 클러스터간의 유사도에 적용하여 정확도를 비교하였다. 그림 11과 그림 12는 최소 지지도의 변화에 따른 클러스터의 응집도와 클러스터간의 유사도를 나타낸 것이다.

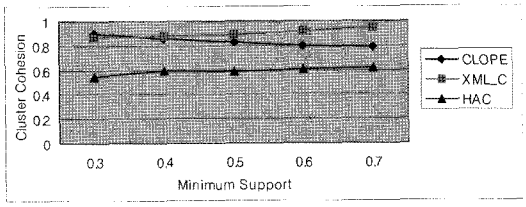


그림 11 클러스터 응집도

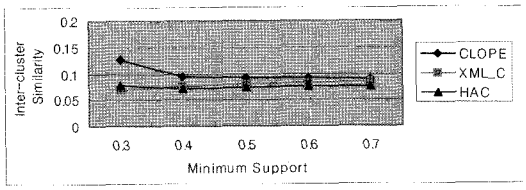


그림 12 클러스터간의 유사도

클러스터의 응집도를 나타내는 그림 11을 살펴보면 HAC는 XML_C과 CLOPE에 비해 응집도가 상대적으로 낮다는 것을 알 수 있다. 그리고 XML_C와 CLOPE를 비교해 보면 XML_C는 지지도가 커질수록 응집도가 증가하는 반면에 CLOPE은 점차로 감소하는 경향을 보인다. 그러므로 XML_C가 CLOPE에 비해 높은 클러스터의 응집도를 나타낸다고 할 수 있다. 그러나 그림 12의 클러스터간의 유사도에서는 HAC가 XML_C와 CLOPE에 비해 낮은 수치를 나타내는 좋은 결과를 보인다. 클러스터간의 유사도는 생성되는 클러스터의 수와 연관이 있다. 즉 생성되는 클러스터의 수가 많으면 클러스터 간의 주요항목이 중복적으로 나타날 가능성이 높다. HAC를 수행할 때 XML_C와 CLOPE에서 생성되는 클러스터의 수와 유사하게 클러스터가 생성되도록 하였으나 HAC의 특성상 클러스터가 병합되는 과정에서 XML_C와 CLOPE에 비해 클러스터가 적게 생성되었다. 이로 인해 클러스터간의 유사도에서는 낮은 수치가 나온 것으로 보인다. 따라서, 만약 HAC가 클러스터의 응집도에서도 높은 결과를 나타내고 클러스터간의 유사도에서 낮은 결과를 나타내었다면 양질의 클러스터를 생성하였다고 볼 수 있지만, 생성된 클러스터의 수가 작

아서 클러스터간의 유사도가 낮게 나온 것이므로 클러스터링의 결과가 좋다고 판단할 수 없다.

실험 결과를 요약하면 이 논문에서 제안하는 클러스터링 XML_C은 CLOPE과 HAC에 비해 높은 클러스터의 응집도를 나타내었고 수행시간에 있어서도 좋은 결과를 보였다.

6. 결 론

XML은 사용자가 임의로 엘리먼트를 정의할 수 있고 엘리먼트는 여러 하위 엘리먼트를 가질 수 있는 계층적 구조를 형성하여 잘 정의된 구조화 문서(well-structured documents)의 형식을 갖는다. 이 논문에서는 스키마가 없는 다양한 구조의 XML 문서들에 대해서 공통 구조를 이용하여 클러스터링 하는 방법을 제안하였다. 클러스터링을 위해 먼저 XML 문서를 나타내는 트리에서 구조를 분리하고, 이를 대상으로 순차패턴 알고리즘을 이용하여 문서의 특성을 나타내는 빈발 구조를 추출하였다. 그리고 각 문서로부터 추출한 구조를 중심으로 하나의 클러스터에 많은 공통 구조를 갖는 문서들을 할당하도록 하는 클러스터링 방법을 이용하였다. 트리에 대한 구조 분리 방법은 기존의 연구에서 속성을 고려하므로 써 발생하는 공간의 효율성 및 분리 시간 등의 문제를 해결하기 위하여 문서 분류를 위한 트리 구조에 많은 영향을 미치지 않는 속성을 제외하고 값을 갖는 엘리먼트의 경로를 중심으로 루트로부터 같은 부모노드의 값을 갖는 엘리먼트까지의 경로로 구조를 분리하는 방법을 제안하였다.

한편 기존 XML 문서의 클러스터링에서 많이 사용되는 HAC 알고리즘은 구조적 유사성 계산에 의해 매트릭스를 구성하여 클러스터링을 수행한다. 그러나 클러스터의 할당 및 병합 여부에 대해 반복적 비교가 이루어지므로 클러스터링을 위한 처리 시간이 많이 소요되는 단점이 있었다. 따라서 이 논문에서는 주요항목 개념을 추가한 트랜잭션 데이타를 위한 알고리즘을 이용하였고, 실험에 의한 비교에서 제안하는 클러스터링 알고리즘은 높은 클러스터의 응집도를 보였으며 수행시간도 많이 감소되는 것을 알 수 있었다.

XML 문서의 클러스터링은 점차 증가하고 있는 XML 문서에 대한 신속한 검색을 제공하기 위한 기반이 된다. 그러므로 이 연구는 XML 문서에 대한 구조 중심의 문서 분류에 유용하며 XML 문서의 통합 여부를 위한 구조 검색에 적용가능하다. 향후 연구에서는 기존의 XML 문서 클러스터링 결과와의 정확성 및 K-means 알고리즘에 대한 비교 실험을 추가적으로 수행할 것이며, XML 문서의 내용까지 함께 고려하는 클러스터링 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] W3C, Extensible Markup Language(XML) 1.1. http://www.w3.org/TR/xml11, W3C Working Draft. April 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] R. Nayak, R. Witt, A. Tonev, "Data Mining and XML Documents," International Conference on Internet Computing, 2002.

[4] 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.

[5] Y. Shen, B. Wang, "Clustering Schemaless XML Document," Proceedings of the 11th International Conference on Cooperative Inormation System, 2003.

[6] Jain, A.K., Mutty, M.N., Flynn, P.J., "Data Clustering: A review, ACM Computing Survey," Vol.31, 1999.

[7] 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.

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

[9] D.Hill, "A Vector Clustering Technique," Merchandise Information Storage, Retrieval and Dissemination, North-Holland, Amsterdam, 1968.

[10] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, "Efficient Substructure Discovery from Large Semi-structured Data," Proceedings of the SIAM International Conference on Data Mining, 2002.

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

[12] E. Kotasakis, "Structural Information Retrieval in XML Documents," ACM Symposium on Applied Computing(SAC), 2002.

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

[14] 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.

[15] J. Widom, "Data Management for XML: Research Directions," IEEE Computer Society Technical

Committee on Data Engineering, 1999.
[16] A. G. Buchner, M. Baumgarten, M. D. Mulvenna, R. Bohm, S. S. Anand, "Data Mining and XML: Current and Future Issues," Proceedings of WISE, 2000.
[17] Z. Zhang, R. Li, S. Cao, Y. Zhu, "Similarity Metric for XML Documents," Workshop on Knowledge and Experience Management(FGWM), 2003.
[18] F. D. Francesca, G. Gordano, G. Manco, R. Ortale, A. Tagarelli, "A General Framework for XML Document Clustering," Technical report, n(8), ICAR-CNR, 2003.
[19] J. H. Hwang, K. H. Ryu, "Structure-based Clustering for XML Document Retrieval," to be published in the Journal of KIPS.
[20] 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.
[21] K. Wang, C. Xu, "Clustering Transactions Using Large Items," Proceedings of ACM CIKM-99, 1999.
[22] S.Abiteboul, P. Buneman, D. Suciu, "Data On The Web: From Relational to Semistructured Data and XML," Morgan Kaufmann Publishers, San Francisco, California, 2000.
[23] http://www.cogsci.princeton.edu/~wn/
[24] 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.
[25] NIAGARA query engine. http://www.cs.wisc.edu/niagara/data.html.
[26] http://sourceforge.net/projects/javawn/

황 정 희

1991년 충북대학교 전산통계학과(이학사) 2001년 충북대학교 대학원 전자계산학과(이학석사). 2005년 충북대학교 대학원 전자계산학과(이학박사). 2005년 현재 정우씨스템(주) GIS 개발연구소장. 관심분야는 XML, 데이타 마이닝, 능동 데이타베이스, 시공간 데이타베이스

류 근 호

1976년 숭실대학교 전산학과(이학사) 1980년 연세대학교 공학대학원 전산전공(공학석사). 1988년 연세대학교 대학원 전산전공(공학박사). 1976년~1986년 육군군수 지원사 전산실(ROTC 장교), 한국전자통신 연구원(연구원), 한국방송통신대 전산학과(조교수). 1989년~1991년 Univ. of Arizona Research Staff (TempIS 연구원, Temporal DB). 1986년~현재 충북대학교 전기전자 컴퓨터공학부 교수. 관심분야는 시간 데이타베이스, 시공간 데이타베이스, Temporal GIS, 유비쿼터스와 스트림데이타 지식기반 정보검색 시스템, 데이타 마이닝 및 데이타베이스 보안, 바이오 인포메틱스