유사 구조 기반 XML 문서의 점진적 클러스터링

(Incremental Clustering of XML Documents based on Similar Structures)

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

요 약 XML은 정보 관리와 데이타 교환에서 점차로 더1) 중요해지고 있다. 효율적인 구조 검색과 문서 통합을 위한 기초 과정은 유사한 구조를 갖는 문서를 클러스터링 하는 것이다. 이것은 구조가 다른 전체 문서를 대상으로 검색하는 것보다 더 신속하고 유연성을 제공하기 때문이다. 따라서 이 논문에서는 XML 문서의 구조 검색과 통합에 유용한 유사 구조기반의 점진적 클러스터링 기법을 제안한다. 기존의 문서 클러스터링에서 벡터를 이용한 문서의 유사도에 의해 클러스터를 형성하는 것과는 다르게 우리는 대량의 데이타에 유연하게 적용할 수 있는 트랜잭션 데이타를 위한 클러스터링 알고리즘을 사용하였다. 제안 기법은 먼저 순차 패턴 알고리즘을 이용하여 XML 문서의 대표 구조를 추출한다. 그리고 문서를 하나의 트랜잭션으로, 문서의 대표구조를 트랜잭션의 항목으로 간주하여 유사 구조 항목 기반의 점진적인 클러스터링을 수행한다. 아울러, 클러스터의 응집도와 클러스터간의 유사도를 정의하였고, 이를 이용하여 기존 연구와의 실험에 대한 분석을 통해 제안 기법의 효율성을 분석하였다.

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

Abstract XML is increasingly important in data exchange and information management. Starting point for retrieving the structure and integrating the documents efficiently is clustering the documents that have similar structure. The reason is that we can retrieve the documents more flexible and faster than the method treating the whole documents that have different structure. Therefore, in this paper, we propose the similar structure-based incremental clustering method useful for retrieving the structure of XML documents and integrating them. As a novel method, we use a clustering algorithm for transactional data that facilitates the large number of data, which is quite different from the existing methods that measure the similarity between documents, using vector. We first extract the representative structures of XML documents using sequential pattern algorithm, and then we perform the similar structure based document clustering, assuming that the document as a transaction, the representative structure of the document as the items of the transaction. In addition, we define the cluster cohesion and inter-cluster similarity, and analyze the efficiency of the proposed method through comparing with the existing method by experiments.

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

1. 서 론

반구조적 언어인 XML(External Markup Language)[1]은 사용의 편리성과 표현의 유연성으로 인해 인터넷 정보 교환의 표준으로 자리 잡게 되었다. XML은 사용자가 임의로 엘리먼트를 정의할 수 있고 엘리먼트는 하위 엘리먼트를 가짐으로써 계층적 구조를 형성한다. 이러한 XML의 구조적 특징은 정보 검색, 문서 관리시스템, 그리고 데이타 마이닝 등에 커다란 영향을 미치고 있으므로 XML 문서에서 구조를 발견하는 연구는 반드시 필요하다[2-6].

최근 XML 문서의 구조 통합과 구조 검색에 대한 많은 연구들이 진행되고 있으며 이러한 대부분의 연구들

· 이 논문은 2003년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2003-002-D00280)
† 비 회 원 : 충북대학교 전자계산학과
jhhwang@dblab.chungbuk.ac.kr
†† 종신회원 : 충북대학교 전기전자컴퓨터공학부 교수
khryu@dblab.chungbuk.ac.kr
논문접수 : 2003년 12월 23일
심사완료 : 2004년 8월 12일

은 복잡한 트리 구조에서 공통의 구조를 찾기 위한 빈발 패턴의 구조 발견을 목적으로 한다. 이러한 빈발 패턴의 공통 구조 추출은 유전체 데이타, 웹 마이닝, 반구조 문서의 구조 검색 등의 응용 분야와 매우 밀접한 관련을 갖으며, 공통의 빈발 구조를 통해 관심 있는 유용한 정보를 더 빠르게 획득할 수 있도록 한다.

그리고 공통 구조 추출과 더불어 중요한 것은 유사한 구조의 문서들을 클러스터링하는 것이다. 유사 구조 문서의 클러스터링은 서로 다른 구조를 갖는 XML 문서들에 대해 구조적으로 유사한 문서들을 군집화하는 것으로써, 서로 다른 구조의 전체 문서를 대상으로 검색하는 것보다 더 신속하고 유연성을 제공하므로, 기존 문서와의 통합 및 분류, 그리고 체계적인 문서 관리에 효과적이다[3,7,8].

클러스터링 기법에는 점진적 클러스터링과 일괄처리 클러스터링 방식이 있다[9,10]. 점진적 클러스터링 방식은 입력 데이타들을 하나씩 입력받아 유사한 데이타들을 클러스터링 하는 방식이며, 일괄처리 클러스터링 방식은 입력 데이타들을 모두 입력받아 한 번에 처리하는 방식이다. 점진적 클러스터링 방식은 입력 데이타를 하나씩 처리하면서 유사한 입력 데이타들을 클러스터링 하므로 일괄처리 방식보다 효율적이며 방대한 데이타를 클러스터링 하는 데 적합하다.

기존 연구들[11,12]은 DTD구조가 같은 문서 또는 다른 구조의 문서들에만 적용될 수 있어 제한적이었다. 그러나 이 논문에서는 서로 다른 구조를 가지는 다중 문서에도 적용 가능한 유사 구조 기반의 점진적 클러스터링을 수행한다. 이를 위해, 먼저 순차 패턴 마이닝 기법[13]을 이용하여 각 문서에서 빈발한 패턴의 구조를 대표 구조로 추출한다. XML 문서를 구성하는 엘리먼트는 문서 내용에 대한 기본 정보 및 문서의 특성을 구별할 수 있는 의미 있는 단어의 계층 구조로 형성되므로 이 정보를 문서 분류를 위한 기초 자료로 사용하고, 추출된 빈발 패턴 구조를 이용하여 유사 구조 문서를 클러스터링 한다. 이 논문에서는 문서를 클러스터링하기 위해 [14]에서 제안한 CLOPE 알고리즘을 기반으로 하고, 점진적 클러스터링의 효율을 높이고자 [15]의 주요항목 개념을 도입하여 클러스터 할당을 위한 개별 항목의 특성을 고려한다. 또한 [14]에서 제시하지 않았던 클러스터의 응집도와 클러스터간의 유사성을 정의하고, 이를 이용하여 제안 기법의 성능을 분석한다.

논문의 구성은 다음과 같다. 먼저 2장에서는 XML의 구조 추출과 클러스터링에 대한 기존 연구 내용을 알아보고 3장에서는 순차패턴을 기반으로 하는 XML 문서의 구조 추출방법을 제시한다. 그리고 4장에서는 3장에서 추출된 구조를 가지고 유사 구조의 문서들을 분류하는 클러스터의 생성방법을 정의한다. 그리고 5장에서는 4장에서 정의된 클러스터의 구성 방법을 기반으로 하는 점진적 클러스터링 기법을 설명한다. 6장에서는 XML의 유사 구조 문서 클러스터링에 대한 기존 연구와의 실험을 통해 비교 분석하고 마지막으로 7장에서 결론을 맺는다.

2. 관련연구

다양한 구조를 갖는 XML 문서가 점차 증가하고 있으므로 XML 문서에 대해 유용한 정보의 추출 및 문서 통합을 위한 유사 구조 문서의 분류 기법에 대한 연구가 필요하다[4,5].

[16]에서는 XML의 구조간 유사성을 분석하기 위하여 XML구조를 트리로 보고 서로 다른 문서들이 트리의 경로를 얼마나 공유하고 있는지를 파악하고, 공유정도에 따른 구조간 유사성의 계산방식을 제안하였다. [17]에서는 점차 증가하는 XML 문서들에 대한 관리의 필요성을 언급하고 문서의 구조를 표현하는 태그(tag)와 일반 텍스트의 내용에 대한 클러스터링 기법을 제시하였다. 이 연구에서는 문서 클러스터링을 위해 거리 값을 기반으로 하는 K-means 알고리즘을 사용하였다. K-means 알고리즘은 문서의 양과 분포에 따라 달라질 수 있는 클러스터의 수를 요구하므로 클러스터링에 대한 유연성이 부족하다는 단점이 있다.

[18]의 DTD-Miner는 XML 문서의 구조를 나타내는 트리를 Spanning 그래프로 표현하고 서브모듈에 대한 반복적 그래프의 병합을 통해 모든 문서 트리를 병합한다. 그리고 경험적 규칙을 적용하여 Spanning 그래프로부터 생성되는 DTD를 생성한다. 이 방법에서는 구체적으로 마이닝 기법을 이용하지 않고, 사용자가 입력한 유사구조의 XML 문서 집합에 대해 자동으로 공통의 DTD 구조를 추출하여 사용자에게 제공한다.

[4,5,19]에서는 XML 문서의 구조추출을 위한 마이닝 기법은 문서 내에서의 구조 관계에 대한 Intra-structured mining과 여러 문서에 대한 Inter-structured mining으로 분류하고 있다. 그러나 구체적인 알고리즘을 제시하지는 않았다.

그리고 많은 트리 구조에서 빈발 서브 구조를 찾기 위한 기존의 연구[11,12,20]들이 있다. [11]는 서로 다른 DTD들을 통합할 때 적당한 통합 구조의 DTD를 찾기 위한 방법으로 DTD를 클러스터링 하는 방법을 제안하였다. 즉, DTD를 단순화하여 엘리먼트의 유사성을 기반으로 DTD를 클러스터링 한다. 그러나 같은 응용 도메인의 DTD를 대상으로 한다. [12]는 패턴 매칭 트리를 찾는 트리 마이닝 알고리즘을 사용하여 내장되어 있는 트리(embedded tree)구조까지도 발견할 수 있는 장점을

갖는다. 그러나 이 알고리즘은 공통의 구조 추출을 목적으로 하기 때문에 문서의 구조 특성에 의한 분류보다는 유사 트리 구조에서의 상세한 공통 구조 추출을 목적으로 한다. [20]은 레이블화되어 있는 트리구조의 집합에서 공통으로 발생하는 연관된 레이블의 쌍을 기반으로 연관규칙을 이용하여 그룹화하고 그것에 대해 최대 빈발 구조를 추출한다. 그러나 연관된 레이블의 쌍을 기반으로 하기 때문에 다중 관계의 트리 구조를 탐색할 수 없다는 단점이 있다. 또한 [21]에서는 비트 맵 인덱스를 이용하여 XML 문서를 클러스터링 할 수 있는 방법을 제시하였으나 대량의 데이타에 대해서는 너무 많은 공간을 차지한다는 문제점이 있다. 한편 [22]에서는 이 논문에서 제시하고 있는 방법과 유사한 순차 패턴 기반의 XML 문서 클러스터링 방법을 제시하였다. 그러나 클러스터내의 주요항목의 비율에만 초점을 두어 클러스터를 할당함으로써 전체적인 클러스터의 유사도를 실질적으로 반영하지 못하였고, 새로운 문서의 삽입이나 삭제로 인해 변화될 수 있는 클러스터의 할당 기준의 변경에 대한 방법론도 제시하지 않았다.

이와 같이 기존 연구에서는 XML 문서의 구조 추출을 위해 마이닝 기법을 일부 적용하고 있으나 다양한 구조를 갖는 많은 XML 문서를 분류하기 위한 마이닝 기법에 대한 연구는 아직 부족한 상태이다. 특히 마이닝을 이용한 XML 문서의 구조 추출에만 초점을 두고 있을 뿐 추출된 구조를 중심으로 클러스터링 하는 구체적인 방법은 거의 제시되지 않고 있다.

따라서 이 논문에서는 XML 문서들에 대해 빈발 패턴의 대표 구조를 추출하고 이를 기준으로 유사 구조 문서를 클러스터링 하는 방법을 제안한다. 제안하는 클러스터링 기법은 기존 연구에서 일반적으로 사용되는 문서의 특성을 벡터로 표현하고 문서간의 유사도를 통해 클러스터를 할당하는 방법을 사용하지 않고, XML 문서를 하나의 트랜잭션으로, 각 문서의 빈발 구조를 트랜잭션의 항목으로 간주하여 트랜잭션 데이타를 위한 클러스터링 알고리즘 CLOPE[14]를 이용한다. 이 CLOPE 알고리즘은 클러스터내의 공통 항목의 비율이 높으면 유사항목이 많은 양질의 클러스터를 생성할 수 있다는 개념을 이용한다. 그러므로 빠른 처리 속도 및 많은 데이타에도 유연하게 적용된다. 그러나 클러스터에 속하는 트랜잭션의 개별 항목을 고려하지 않기 때문에 클러스터간의 유사성이 높아질 수 있고 적정한 수의 클러스터 생성을 조절할 수 없다는 단점이 있다. 그러므로 우리는 이 문제를 개선하기 위하여 각 클러스터에 대한 주요항목을 유지하고 이를 이용한 점진적 클러스터링 수행 방법을 제안한다. 그리고 주요항목을 이용한 클러스터의 응집도와 클러스터간의 유사도를 정의하여, 기존 연구와의 실험에 대한 분석을 통해 제안 알고리즘의 효율성을 증명한다.

3. XML 문서의 대표 구조 추출

기존의 텍스트 문서의 분류에서는 문서의 특징을 추출하기 위해 단어의 발생빈도를 가지고 측정하였다[17]. 그러나 XML문서는 기존 문서들과는 다르게, 순차적이고 계층적인 구조의 엘리먼트로 구성되어 있다. 그러므로 XML 문서의 엘리먼트 순서와 부여된 엘리먼트 그 자체는 XML 문서의 형태와 종류를 구분할 수 있게 해주는 특징을 가지고 있다[16,17]. 아래 XML 문서의 일부는 XML 문서의 특성을 설명하기 위한 것이다. (a)와 (b) 문서는 같은 엘리먼트들로 구성되어 있지만 (a)는 학교 정보를 나타내는 문서이고 (b)는 학생 정보를 나타내는 문서이다. 이와 같이 엘리먼트의 구조와 순서는 문서의 내용을 구분할 수 있는 특성이 있다. 그러므로 이 논문에서는 엘리먼트를 기준으로 문서의 구조를 추출하고, 문서의 구조 추출에서는 엘리먼트의 발생횟수뿐만 아니라 발생 순서를 고려하는 순차패턴 마이닝을 이용한다.

```
<학교>
 <이름>○○대학교 </이름>
 <년도> 1953 </년도>
 <학생> 15000명 </학생>
</학교>
```

(a) 학교 정보 문서

```
<학생>
 <이름> 홍길동</이름>
 <년도> 1980 </년도>
 <학교> ○○대학교 </학교>
</학생>
```

(b)학생 정보 문서

3.1 엘리먼트 경로 시퀀스

XML 문서의 구조적 유사성은 XML 문서의 구조 경로에 대하여 서로 다른 문서들이 같은 구조 경로를 얼마나 공유하고 있는지를 파악하여, 이 공유 구조의 정도에 따라 구조간 유사성을 판별하는 것이다. 그러므로 구조의 유사성 판별을 위해 먼저 각 문서의 대표 구조를 추출한다.

XML문서는 계층적 구조로 이루어져 있으므로 하나의 문서에 포함되어 있는 경로는 다양하다. 다양한 경로의 XML 문서에서 의미 있는 구조를 추출하는 것은 문서의 주제어를 추출하는 것과 유사하다고 볼 수 있다. 즉, XML문서의 엘리먼트는 문서의 내용을 예측할 수 있는

문서의 특성을 나타내는 단어로 구성되므로 엘리먼트에 대한 구조 경로를 통해 계층적 구조를 이루는 문서의 의미 있는 대표 구조를 추출할 수 있다. 다음의 예제 문서(Actor.xml)는 문서의 빈발구조를 찾는 전처리 과정을 설명하기 위한 영화배우 정보 문서의 일부이다.

```
<Actor>
  <Name>
    <FirstName>Alfred</FirstName>
    <LastName>Abel</LastName>
  </Name>
  <Filmography>
    <Movie>
      <Title>Metropolis</Title>
      <Year>1927</Year>
      <Director>Fritz Lang</Director>
    </Movie>
  </Filmography>
</Actor>
```

위 문서는 레벨 1의 <Actor>부터 레벨 4에 해당하는 <Title>, <Year>, <Director> 등으로 구성되어 있다. 이렇게 여러 가지 엘리먼트로 구성되어 있는 문서에서 문서의 특성을 나타내는 구조를 추출하기 위해서는 각각의 엘리먼트를 쉽게 구별할 수 있는 재명명의 절차가 필요하다. 그러므로 위 예제 문서에서 추출될 수 있는 엘리먼트의 구조를 쉽게 식별하기 위하여 다음과 같은 엘리먼트의 매핑 테이블을 이용하여 알파벳으로 재명명한다.

그림 1과 같이 재명명된 문서의 구조를 토대로 실제적인 내용을 포함하는 엘리먼트들에 대한 문서 경로를 고유의 번호(X_id)를 갖는 시퀀스로 간주하고 주어진 최소 지지도를 만족하는 순차 패턴 마이닝 알고리즘을 적용하여 가장 빈발한 시퀀스 패턴, 즉 그 문서를 대표할 수 있는 엘리먼트의 구조 정보를 찾는다. 위 예제 문서에서 의미 있는 엘리먼트 경로의 시퀀스는 표 1과 같이 표현된다.

문서의 구조 특성을 추출하기 위한 의미 있는 엘리먼트의 고유한 경로 구조는 빈발패턴을 찾기 위한 입력 정보가 되며 하나의 경로에 포함되어 있는 엘리먼트들은 발생 시퀀스를 구성하는 하나의 항목으로 고려된다.

표 1 구조 경로 시퀀스

X_id	X_path
1	a/b1/c1
2	a/b1/c2
3	a/b2/c3/d1
4	a/b2/c3/d2
5	a/b2/c3/d3

3.2 순차패턴을 이용한 문서의 대표 구조

순차 패턴 마이닝 알고리즘은 연관 규칙과는 달리 트랜잭션의 발생 횟수와 발생 순서를 고려하므로 XML 문서의 구조 추출에 적합하다[6,16].

시퀀스들의 집합에 대한 빈발 구조를 추출하기 위하여 이 논문에서는 후보패턴을 생성하지 않는 [13]의 PrefixSpan 알고리즘을 이용한다. 이것은 루트노드에서 시작하여 깊이 우선 검색(Depth First Search)순으로 노드를 확장하면서 PrefixSpan 트리를 만들어 가는 방식이다. 노드를 확장할 때에 그 노드가 나타내는 빈번한 시퀀스를 포함하고 있는 시퀀스만을 모은 시퀀스(prefix) 이후의 부분만을 지정한 Project DB를 이용하는 방법으로써, 성능의 우수성이 [13]에서 증명되었다. 표 1에 대해 PrefixSpan 알고리즘을 이용하여 빈발구조를 추출하는 방법은 다음과 같고, 이를 위해 먼저 빈발 구조 최소 지지도(Frequent Structure Minimum Support)를 정의한다.

정의 1. 빈발 구조 최소 지지도(Frequent Structure Minimum Support)

빈발 구조 최소 지지도란 한 문서의 전체 경로 중에서 빈발 구조 비율을 만족하는 최소 빈발도이며, 이를 만족하는 구조 경로를 빈발 구조라 한다. 이것을 식으로 표현하면 다음과 같다.

FSMS = 빈발 구조 비율 * 문서 전체 경로의 수
(0 < 빈발 구조 비율< 1)

Level 1		Level 2		Level 3		Level 4	
element	rename	element	rename	element	rename	element	rename
Actor	a	Name	b1	FirstName	c1	Title	d1
		Filmography	b2	LastName	c2	Year	d2
				Movie	c3	Director	d3

그림 1 엘리먼트 매핑 테이블

각 엘리먼트의 경로로 구성된 시퀀스 집합에 대해 빈발 구조 비율을 0.3로 가정하면 FSMS는 2(0.3 *5)가 되고, 최소 지지도를 만족하는 크기(length)가 1인 엘리먼트의 빈발도는 a: 5, b1: 2, b2: 3, c3: 3 이다. 이러한 빈발 요소를 중심으로 시퀀스의 스캔비용과 Project DB 형성 비용을 줄일 수 있는 length-2 빈발 패턴에 대한 S-matrix는 그림 2와 같이 형성된다.

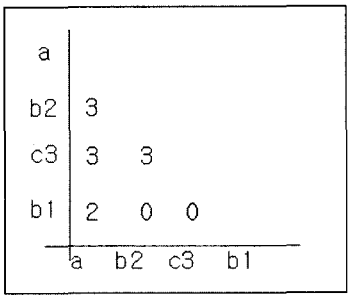


그림 2 Length-2 빈발 구조의 S-matrix

그림 2에서 지지도가 3인 M[a, b2]에 대한 시퀀스, 즉 (a, b2)를 prefix로 하는 projection_DB의 구성요소는 <c3, d1>, <c3, d2>, <c3, d3>이고 이 시퀀스를 기초로 다시 빈발 항목에 대한 Project DB의 매트릭스를 그림 2와 같은 방식으로 형성하여 점차로 빈발구조의 크기를 늘려가고 더 이상 Projected DB를 만들 필요가 없을 때까지 이 과정을 반복한다(상세한 알고리즘은 [13]참고).

위의 예에서 발견된 최대 빈발 구조인 <a/b2/c3>는 전체 문서 경로에 대해 약 60%(3/5)의 비율로 발생하는 구조이다.

일반적으로 순차패턴 마이닝에 의해 발견된 최대 빈발패턴은 문서에서 가장 공통적으로 사용되는 구조로서 중요한 의미를 갖지만 이 논문에서는 최대 빈발패턴의 구조가 아니더라도 최대 빈발 구조에 대한 일정 비율이상을 만족하는 구조(예, 최대빈발구조 길이 3 * 80% = 빈발 구조 길이 2)도 중요한 의미로 간주하고, 중복되지 않는 구조에 대해 클러스터링을 위한 기초 입력 자료에 포함한다. 이것은 하나의 문서에 여러 가지 주제가 함께 나타날 수 있는 경우에 최대 빈발 패턴의 구조만이 그 문서를 대표하는 유일한 구조가 될 수 없기 때문이다. 그리고 우리는 빈발 구조 자체를 문서를 클러스터링하기 위한 구조 항목으로 간주하기 때문에 문서의 구조 정보에 대한 손실을 줄이기 위함이다.

이렇게 순차 패턴 알고리즘을 통해 찾아진 빈발패턴은 문서에서 많은 하위노드를 포함하는 엘리먼트일수록 발생 빈도수가 많이 나타나게 되고 이것은 문서에서 그 엘리먼트가 차지하는 비중이 크다는 것을 의미한다. 따라서 순차패턴에 의한 빈발 구조 추출은 임의문서의 내용을 예측할 수 있는 경로 구조 특성을 잘 반영하는 구조적 분류 기준이 된다.

4. XML 문서 클러스터링

순차패턴 마이닝에 의해 추출된 각 문서의 빈발구조는 전체 문서에서 미리 주어진 최소 지지도에 대해 만족하는, XML 문서의 엘리먼트 순서에 기반하여 추출된 구조이다. 우리는 클러스터링을 수행하기 위해 각 XML 문서를 하나의 트랜잭션으로 가정하고 각 문서에서 추출된 빈발 구조들을 트랜잭션의 항목들로 취급하여 유사한 항목기준의 그룹으로 문서를 클러스터링 한다.

이 논문에서 제안하는 클러스터링은 많은 공통 항목을 포함하도록 유도하는 [14]의 클러스터 할당 방식을 기반으로 하고, 효율적인 클러스터 관리를 위해 [15]의 주요항목(Large Item)개념을 이용한다.

모든 트랜잭션에 포함되어 있는 빈발 구조 항목들의 집합 I = $\{i_1,i_2,...,i_n\}$하고, 클러스터 집합 C = $\{C_1,C_2,...,C_m\}$, 문서를 나타내는 트랜잭션 집합 T = $\{t_1,t_2,...,t_k\}$이라 표기한다. 먼저 적합한 클러스터에 트랜잭션을 할당하기 위한 기준으로 클러스터 할당 이익을 정의한다.

정의 2. 클러스터 할당 이익(Gain)

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

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

여기서 G는 각 클러스터에서 고요항목 W에 대한 누적 항목의 비례를 나타내는 H를 나타내는 것으로써, H = T(전체 항목 수) / W(고유항목수)이고 G = T/W^2이다.

*Gain*은 하나의 트랜잭션에 대하여 클러스터 할당을 위한 기준이 되며, 각 클러스터에 대한 공통항목의 비율이 높을수록 큰 값의 클러스터 할당 이익이 산출된다. 그러므로 최대의 *Gain* 값이 되도록 하는 클러스터에 할당하여 공통 항목들이 많은 클러스터의 구성을 유지한다. 다음 예제는 *Gain*을 이용하여 클러스터링 형태를 비교한 것이다.

예제 1. 각 문서(t1, t2, t3, t4, t5, t6 {ti ∈ T})에 대한 빈발 구조 항목을 알파벳으로 간단히 표기한 t1 = {a, b, c}, t2 = {a, b, c, f}, t3 = {a, c, d, e}, t4 = {c,

d, e}, t5 = {g, h, a, e}, t6 = {g, h, a, f}를 고려할 때 두 가지 형태의 클러스터링에 대한 클러스터 할당 이익의 비교이다.

① C = {C1={t1,t2,t,3,t4}, C2={t5,t6}}일 경우

C1의 고유 항목과 그에 대한 공통 항목의 누적은 {c:4, a:3, b:2, d:2, e:2, f:1}이고 C2는 {g:2, h:2, a:2, f:1, e:1}이다. 이것을 고유 항목에 대한 공통 항목의 비율을 나타내는 클러스터 할당 이익은 Gain = $\frac{\frac{14}{6^2}\times 4+\frac{8}{5^2}\times 2}{6}$ = 0.365 이다.

② C = {C1={t1,t2}, C2={t3,t4}, C3={t5,t6}}일 경우

C1의 고유 항목과 그에 대한 공통 항목의 누적은 {a:2, b:2, c:2 f:1}이고 C2는 {a:1, c:2, d:2, e:2}, C3는 {g:2, h:2, a:2, e:1, f:1}이다. 이것에 대한 클러스터 할당 이익 Gain = $\frac{\frac{7}{4^2}\times 2+\frac{7}{4^2}\times 2+\frac{7}{4^2}\times 2}{6}$ = 0.437

①과 ②에 대한 클러스터 할당 이익은 ②가 ①보다 크다. 그러므로 ②와 같은 방법으로 클러스터링되는 것이 바람직하다.

그러나 개별 항목을 고려하지 않고 공통 항목의 비율인 *Gain* 만을 이용하면 다음과 같은 문제점이 발생한다.

예제 2. 각각 3개의 트랜잭션을 포함하고 있는 클러스터 C1 = {a:3, b:3, c:1}, C2 = {d:3, e:1, c:3}상태에서 t4 = {f, c}의 구조 항목이 삽입된다고 가정하면, C1 또는 C2에 할당되었을 경우의 클러스터 할당 이익은 $\frac{\frac{9}{4^2}\times 4+\frac{7}{3^2}\times 3}{7}$ = 0.654이고, 새로운 클러스터를 생성하여 할당했을 경우의 클러스터 할당이익은 $\frac{\frac{7}{3^2}\times 3+\frac{7}{3^2}\times 3+\frac{2}{2^2}\times 1}{7}$ = 0.738이다. 그러므로 *Gain*의 비교를 통해, 존재하는 C1, C2 클러스터에 할당하기보다는 새로운 클러스터를 생성하게 된다. 이것은 하나의 트랜잭션이 새로운 클러스터에 할당되는 경우, 클러스터 할당 이익 *Gain* = $\frac{\text{항목수}}{\text{항목수}^2}$가 되므로 상당히 높은 값의 할당 이익 값이 산출된다. 이로 인해 적정 크기 이상의 많은 클러스터가 생성되는 문제점이 발생한다. 우리는 이 문제를 개선하기 위한 방법으로 클러스터의 개별 항목을 고려하기 위해 주요항목과 클러스터 참여도를 다음과 같이 정의한다.

정의 3. 주요항목(Large Item)

클러스터 Ci에 대한 항목의 지지도는 Ci에서 항목 i_j(j <= n)를 포함하고 있는 트랜잭션의 수이고, 사용자가 지정한 최소 지지도, θ(0 <θ≤ 1)에 대해 Ci내에서 그 항목을 포함하고 있는 트랜잭션의 수가 항목의 지지도 Sup = θ*|Ci(Tr)| 이상이면 항목 i_j는 Ci의 주요항목이다.

$$Ci(L)i_j = |Ci(Tr)i_{j\in I})| >= Sup$$

이 때 |Ci(Tr)|는 클러스터 Ci에 포함된 전체 트랜잭션의 수이며, $|Ci(Tr)i_{j\in I})|$는 클러스터 Ci에 포함된, 항목 i_j를 포함하고 있는 트랜잭션의 수를 의미한다.

하나의 클러스터에 포함되어 있는 모든 항목은 주요항목과 비주요항목(Small Item)으로 나뉘어지며 비주요항목은 주요항목에 포함되지 못하는 후보항목이다. 임의의 클러스터에 새로운 문서가 할당되면 그에 따라 주요항목이 될 수 있는 임계치가 변화하고 이에 따라 주요항목과 비주요항목에도 변화가 발생한다. 즉, 비주요항목은 공통의 구조 항목이 삽입되면 주요항목으로 변화될 수 있는 후보 항목이다.

정의 4. 클러스터 참여도

빈발 구조 항목으로 구성된 문서 t_k와 클러스터 Cj의 주요항목과의 공통 항목비율이고, 이것은 문서 t_k가 Cj에 속할 가능성의 정도를 나타내며 다음과 같은 식으로 표현한다.

$$p\_Allo(t_k \Rightarrow Cj) = \frac{|tk \cap Cj(L)|}{|tk|} >= \omega$$

(0 < ω< 1: 최소 참여도)

$|t_k|$는 삽입되는 문서 t_k에 포함된 항목의 수이고, 클러스터 참여도는 (예제 2)와 같이 클러스터 할당이익의 계산에 의하여 새로운 클러스터를 생성해야 하는 경우에 주어진 최소 참여도를 만족하는 클러스터가 있으면 새로운 클러스터를 생성하지 않고 최대의 참여도를 갖는 클러스터에 해당 문서를 할당한다. 즉, 이미 존재하는 클러스터에 대한 할당 가능성을 검사하여 적정 수의 클러스터 생성을 유도한다. ω를 작게 하면 클러스터 생성 가능성이 줄어들고, ω를 크게 하면 클러스터의 생성 가능성이 커지는 반면 클러스터의 응집도는 작아질 수 있다. 이 논문에서는 ω를 0.5로 하고 이것을 만족하는 클러스터 중에서 가장 높은 값을 갖는 클러스터에 할당한다.

클러스터링은 클러스터내의 유사도를 높이고 클러스터간의 유사도를 낮게 하여 그룹화하는 것이 목적이므로, 이를 측정할 수 있는 기준이 필요하다. 그러므로 클러스터내의 유사밀도를 나타내는 클러스터의 응집도와 클러스터간의 차별성을 나타내는 클러스터간의 유사도를 다음과 같이 정의한다.

정의 5. 클러스터 응집도

클러스터 Ci의 응집도 Coh(Ci)는 클러스터 Ci에 포함된 전체 항목 T(Ci)에 대한 주요항목이 차지하는 비율이다. 이것은 다음과 같이 계산하고 1의 값에 가까울수

록 좋은 응집도를 나타낸다.

$$Coh(\mathrm{Ci}) = \frac{Ci(L)}{T(Ci)}$$

다른 클러스터와 비교하여 응집도가 크다는 것은 유사 구조의 문서가 더 잘 밀집되어 있는 클러스터를 의미한다.

정의 6. 클러스터간의 유사도

클러스터 Ci, Cj에 포함되어 있는 주요 항목 중심의 유사도는 주요 항목 집합에 대한 공통의 주요항목의 비율을 클러스터 Ci, Cj의 유사도라 한다. 이것은 다음과 같은 식에 의해 계산하고 0에 가까울수록 유사성이 거의 없는 클러스터이다.

$$Sim(\mathrm{Ci},\ \mathrm{Cj}) = \frac{L(Ci \cap Cj) \times \frac{|L(Ci \cap Cj)|}{|L(Ci + Cj)|}}{Ci(L) + Cj(L)}$$

이 때 L(Ci∩Cj)는 공통 항목에 대한 각 클러스터에서의 발생 횟수, |L(Ci∩Cj)|은 공통의 주요 항목들의 누적 발생 횟수, |L(Ci+Cj)|는 주요항목의 전체 누적 횟수를 나타낸다.

클러스터간의 유사도는 각 클러스터간의 차별성의 정도를 의미하고, 낮은 클러스터간의 유사도는 전체적으로 양질의 클러스터 생성을 의미한다.

예제 3. 사용자 정의 최소 지지도 0.6일 때 (예제 1)의 ①, ②에 대한 클러스터 응집도와 클러스터간의 유사도 산출은 다음과 같다.

① C = {C1={t1,t2,t,3,t4}, C2={t5,t6}}일 경우(C1의 주요항목(0.6*4 = 3)은 a, c이고 C2의 주요항목(06*2 = 2)은 g,h,a), 응집도는 $\frac{\frac{7}{14}+\frac{6}{8}}{2}$ = 0.625이고, 클러스터간의 유사도는 공통 항목 a, 발생 수 2, 누적 5이므로 $\frac{2\times\frac{5}{13}}{5}$ = 0.153이다.

② C = {C1={t1,t2}, C2={t3,t4}, C3={t5,t6}}일 경우, 응집도는 $\frac{\frac{6}{7}+\frac{6}{7}+\frac{6}{8}}{3}$ = 0.821이고, 클러스터간의 유사도는 공통 항목 a, c이고, 이들의 클러스터에서의 발생 수는 4, 누적 수는 8이므로 $\frac{4\times\frac{8}{18}}{9}$ = 0.197이다.

산출된 클러스터의 응집도와 클러스터간의 유사도를 비교하면 응집도에서는 ②가 ①보다 더 좋은 결과를 나타내고, 클러스터간의 유사도에서는 ①의 경우가 더 낮은 좋은 클러스터링 결과를 나타낸다. 그러나 만약 지지도를 0.5로 하면 응집도와 유사도에서 ①의 경우가 모두 좋게 나타난다. 그러므로 적당한 지지도의 결정은 중요하다.

5. 점진적 클러스터링

점진적인 XML 문서 클러스터링은 삽입되는 하나씩의 XML 문서에 대해 구조적 유사성을 기준으로 이미 존재하는 클러스터들에 대한 차분 연산을 수행하여 적합한 클러스터를 찾거나 새로운 클러스터를 생성한다. 그리고 삭제되는 문서로 인해 해당 문서를 포함하는 클러스터가 주어진 최소의 응집도를 만족하지 못할 경우에는 클러스터내의 문서들을 적합한 다른 클러스터로 이동하여 양질의 클러스터를 유지하도록 한다.

5.1 차분 연산

새로운 문서 삽입에 대한 클러스터링은 전체 문서에 대한 클러스터링보다 기존 클러스터의 할당 이익에 대한 차분 연산을 계산하여 적합한 클러스터를 할당하는 것이 더 효율적이다. 이 논문에서는 하나의 문서에 대한 삽입 및 삭제가 이루어지는 점진적 클러스터링을 고려하고 이에 대한 차분 연산의 수행을 위해 다음과 같이 정의한다.

정의 7. 차분 연산

기존 클러스터들에 대하여 삽입 또는 삭제되는 트랜잭션의 항목들에 대한 클러스터 할당 이익의 변화 정도에 대한 연산을 차분 연산이라 하고 삽입 차분($add\_Gain(\Delta^+)$)과 삭제 차분($del\_Gain(\Delta^-)$)을 식으로 나타내면 다음과 같다.

$$add\_Gain(\Delta^+) = New\_Gain(\mathrm{Ci}) - Old\_Gain(\mathrm{Ci})$$
$$= \frac{T'(Ci)}{W(Ci)^2} \times (|Ci(Tr)| + 1) - \frac{T(Ci)}{W(Ci)^2} \times |Ci(Tr)| \quad (1)$$

$$del\_Gain(\Delta^-) = Old\_Gain(\mathrm{Ci}) - New\_Gain(\mathrm{Ci})$$
$$= \frac{T(Ci)}{W(Ci)^2} \times |Ci(Tr)| - \frac{T'(Ci)}{W(Ci)^2} \times (|Ci(Tr)| - 1) \quad (2)$$

기존 클러스터들에 대한 고유 항목의 수, 총 항목의 수, 해당 트랜잭션의 수를 W, T, |Ci(Tr)|로 표시하고 트랜잭션의 삽입 및 삭제 시점에 대하여 W′, T′, |Ci′(Tr)|로 표시하면 클러스터 할당 이익의 삽입 차분은 식 (1), 삭제 차분은 식 (2)에 의해 구할 수 있다. 그리고 새롭게 삽입되는 트랜잭션이나 트랜잭션의 삭제로 인해 다른 클러스터로 이동해야 하는 경우에 가장 큰 삽입 차분의 클러스터에 할당한다.

5.2 문서의 삽입 및 삭제

빈발 구조 항목을 갖는 문서들이 클러스터에 할당될 경우에 모든 클러스터에 대한 할당이익을 산출하고 이를 비교하여 가장 큰 값을 나타내는 클러스터에 문서를 할당하는 과정은 많은 시간이 소요된다. 이를 개선하기 위한 방법으로 우리는 정의 4의 클러스터 참여도를 다시 이용한다.

클러스터의 참여도는 삽입되는 항목들에 대한 클러스터의 주요항목과의 공통 비율을 의미하므로 해당 클러스터에 대한 참여도를 통해 할당 가능성을 미리 예측하고, 최소의 참여도 w를 만족하는 클러스터에 대해서만 클러스터 할당 이익을 계산하여 할당 여부를 결정한다.

클러스터의 할당은 (정의 7)의 차분 연산을 이용하여 문서의 삽입과 삭제로 인해 변화되는 클러스터 할당 이익의 정도를 비교하고, 삽입의 경우에는 가장 많은 차분의 이익이 발생하는 클러스터에 트랜잭션을 할당하고, 삭제의 경우에는 해당 클러스터에서 트랜잭션을 제거하고 이로 인하여 주어진 최소의 클러스터 응집도를 만족하지 못하는 경우에는 클러스터에 포함되어 있는 트랜잭션에 대한 클러스터의 이동 여부를 결정한다. 다른 클러스터로 이동할 경우에는 새로운 클러스터에서 발생하는 삽입 차분 $Add_Gain(\Delta^{+})$과 해당 클러스터에서 트랜잭션이 삭제될 경우의 삭제 차분 $Del_Gain(\Delta^{-})$의 값을 비교하여 삽입 차분이 삭제 차분보다 큰 경우에만 이동이 가능하다. 트랜잭션의 삽입과 삭제에 대한 알고리즘은 그림 3과 같다.

트랜잭션의 삽입과 삭제가 발생하면 해당 클러스터의 주요항목에도 변화가 발생한다. 그러므로 클러스터에 할당된 트랜잭션 수의 증가 또는 감소에 따라 새롭게 추가되어야 하는 주요항목이나 제거되어야 하는 주요 항목에 대한 조사가 반드시 이루어져야 한다. 트랜잭션의 삽입과 삭제는 1회에 하나의 트랜잭션에 대한 변화를 고려하므로 각 해당 항목에 대한 지지도는 +1, -1의 변화에 대한 주요항목 재조정이 이루어진다.

6. 실험 및 비교

이 장에서는 이 논문에서 제안하는 알고리즘에 대한 클러스터의 효율성을 측정하기 위해 기존 알고리즘 [14]와의 비교에 대한 실험을 기술한다.

실험에 사용된 문서는 위스콘신의 XML 데이타 뱅크 [23]에서 제공하는 XML 문서를 이용하였고 각 분야별 다른 DTD로 구성되어 있는 문서, 즉 책, 클럽, 연극, 경매, 회사, 학과, 배우, 영화 등의 8개 분야에 대한 총 400개 문서를 가지고 클러스터링의 수행에 대한 비교 실험을 하였다.

먼저, 유사구조에 기반하는 클러스터링을 수행하기 위하여 각 문서에 대해 3장에서 설명된 순차 패턴을 이용하여 문서 전체 경로의 수에 대한 빈발 구조 비율 0.2로 하여 대표 구조를 추출하였다. 각 문서의 대표 구조로 추출된 결과를 보면 최대 빈발 패턴 구조의 평균 길이는 5.4이고 한 문서에서의 빈발 구조는 평균 4.9개의 빈발 패턴 구조가 추출되었다. 그리고 이러한 최대 빈발 구조와 더불어 중복되지 않는 최대 빈발구조 길이의 80% 이상의 구조도 클러스터링을 위한 빈발 구조 항목으로 포함시켜 점진적인 클러스터링을 수행하였다. CLOPE을 수행할 때에도 동일한 방식의 실험 비교를 위해 우리가 제안한 차분 연산을 이용하여 점진적으로 클러스터링하였다. 다음의 실험들은 우리가 제안한 알고리즘 XML_C와 기존 [14]의 CLOPE 알고리즘에 대한

● **Insert transaction t**
  extract representative structure using sequence pattern mining;
  while not end of the existing cluster and *p_Allo*(C) >= *w1* // (w=0.2)
    find a cluster(Ci) maximizing *add_Gain*(C);
    find a cluster(Cj) maximizing *p_Allo*(C);
  if new cluster *add_Gain*((C_k) > *add_Gain*(Ci)
   if *p_Allo*(Cj) >= *w2* // (w=0.5)
     allocate t to an existing cluster Cj;
    else allocate t to an new cluster C_k;
   else allocate t to an existing cluster Ci;

● **Delete transaction <t_i, Ci>**
  repeat
    read transaction t_j belong to Ci;
    find a cluster(Cj) maximizing (*add_Gain*(C) - *del_Gain*(C));
    move transaction t_j to cluster Cj;
    until *Coh*(Ci) >= *min_Coh*;

그림 3 차분 연산을 이용한 XML 문서 클러스터링 알고리즘

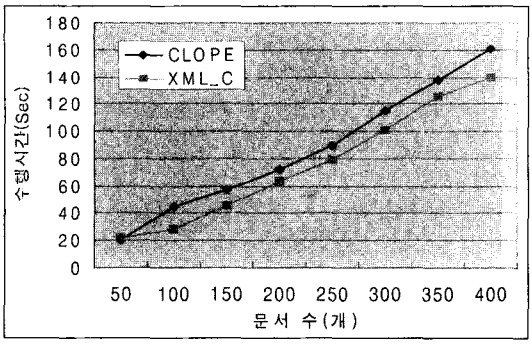


그림 4 수행시간

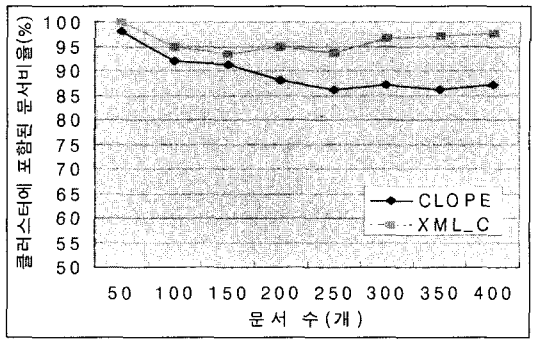


그림 5 클러스터에 포함된 문서 비율

실험 결과의 비교이다.

먼저, 그림 4는 문서의 수에 따른 클러스터링의 평균 수행시간을 비교한 것이다.

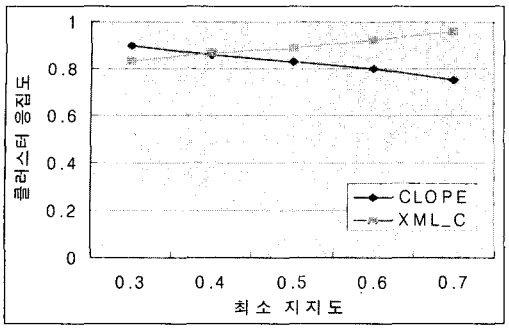
기존 알고리즘과 제안 알고리즘은 문서의 수가 많아질수록 선형적인 증가를 보인다. 그리고 CLOPE은 XML_C보다 평균적으로 1.2배 더 많은 수행시간이 소요되는 것을 알 수 있었다. 반면에 문서의 수가 적은 실험의 초기에는 작은 차이지만 XML_C이 더 많은 시간이 소요되는 것을 보여준다. 즉, 소량의 데이타를 포함하는 각 클러스터에 대한 주요항목의 구성시간은 주요항목의 구성없이 각 클러스터의 *Gain*만을 비교하여 클러스터링을 수행하는 CLOPE보다 많은 시간 비용을 요구한다는 것을 알 수 있다. 그러나 문서의 수가 많아지면 많아질수록 클러스터의 주요항목은 전체적인 수행시간을 줄일 수 있는 효과가 있다는 것을 확인할 수 있다. 이것은 주요항목을 바탕으로, 모든 클러스터에 대한 할당 이익을 비교하지 않고 주요항목을 고려하는 클러스터 참여도를 이용하여 할당 가능한 클러스터에 대해서만 차분의 할당 이익을 비교하므로 써 클러스터의 할당 시간을 줄일 수 있기 때문이다.

다음은 빈발 구조 항목의 문서의 삽입에 따라 생성되는 클러스터들에 대해 어느 정도의 문서가 클러스터에 포함되어 생성되는지에 대한 실험을 하였다. 이 실험은 적정 수의 클러스터 생성과 관련이 있으며, 클러스터에 포함된 문서의 수를 통해 클러스터의 유효성을 검사하는 것이다. 이 실험에서 CLOPE는 XML_C에 비해 평균적으로 1.3배 많은 클러스터가 생성되었다. 그리고 실험에 사용된 8개 분야의 XML 문서는 분야별 다른 DTD를 사용하기 때문에 같은 DTD로 구성되어 있는 같은 분야의 문서들은 대부분 같은 클러스터에 할당되는 것을 알 수 있었다. 그림 5는 이에 대한 실험 결과이며 생성되는 클러스터 중에서 전체 문서 수의 3%미만의 수가 할당된 클러스터는 노이즈로 간주하여 실험 결과에 포함하지 않았다.

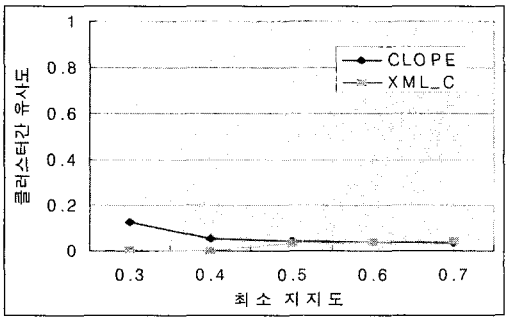
그림 5의 실험 결과를 통해 XML_C는 CLOPE보다 평균적으로 적은 수의 클러스터를 생성하고 대체적으로 95%이상의 많은 문서를 포함하는 높은 밀도의 클러스터가 생성된다는 것을 알 수 있었다. 이것은 기존 알고리즘이 단지 클러스터에 포함되는 항목들의 공통 비율만을 가지고 클러스터를 생성하므로 예제 2에서 언급되었던 것과 같이 새로운 클러스터에 대한 *Gain*이 기존 클러스터에 할당하는 경우의 *Gain*보다 더 클 경우가 많기 때문에 클러스터의 생성 수가 증가하는 것이다. 그러나 XML_C에서는 최소의 클러스터 참여도를 0.5로 하고 이를 이용하여 응집도를 고려하는 클러스터의 생성을 조절하므로 써 클러스터에 많은 문서를 포함하도록 하는 효과를 나타내었다.

전체 400개의 실험 문서에 대한 클러스터링 결과에 대해 클러스터의 정확도를 나타내는 클러스터의 응집도와 클러스터간의 유사도를 측정하였다. 클러스터의 응집도와 클러스터간의 유사도의 측정은 클러스터의 주요항목을 이용한다. 그러나 CLOPE에서는 주요항목을 이용하지 않기 때문에 우리는 CLOPE를 수행한 후 생성된 클러스터들에서 주어진 지지도를 만족하는 주요항목을 XML_C와 같은 방식으로 추출하고 이것을 응집도와 유사도 공식에 적용하여 XML_C의 수행 결과와 비교하였다. 그림 6은 최소 지지도의 변화에 따른 클러스터의 응집도(a)와 클러스터간의 유사도(b)를 나타낸 것이다.

클러스터의 응집도를 나타내는 (a)에서 지지도가 0.4일 때 제안 알고리즘은 0.87의 응집도를 나타내었고 기존 알고리즘은 0.86의 응집도를 보였다. 그리고 지지도가 커질수록 응집도의 차이가 점차로 증가하는 것을 확인할 수 있으므로, 평균적으로 XML_C가 높은 응집도의 클러스터링 결과를 나타낸다. 그리고 (b)의 클러스터간의 유사도에서도 가장 적절한 지지도의 0.3과 0.4에서 CLOPE은 0.12, 0.05, XML_C는 0.007, 0.001의 결과를 보였다. 즉, XML_C는 CLOPE과 비교하여 더 낮은 수치의 결과를 보이므로 질적으로 좋은 클러스터가 생성



(a) 클러스터 응집도



(b) 클러스터간의 유사도

그림 6 클러스터 응집도와 클러스터간의 유사도

되는 것을 확인할 수 있었다. 이것은 CLOPE이 XML_C보다 많은 클러스터를 생성하고 이들 클러스터에 포함되는 문서의 수도 적기 때문에 클러스터의 응집도가 낮아지는 것이며, 많은 클러스터에 각 항목들이 낮은 밀도의 분포로 포함되어 있으므로 클러스터간의 유사도는 높아지는 것임을 알 수 있다.

XML_C은 이 실험 데이타에서 클러스터 응집도와 클러스터간의 유사도 두 가지를 모두 고려할 때 지지도 0.4이하에서 클러스터의 응집도가 높으면서 클러스터간의 유사도가 낮은 좋은 결과를 보인다. 또한 응집도 및 클러스터간의 유사도 관계를 살펴보면 응집도와 클러스터간의 유사도가 비례하는 것으로 나타난다. 즉, 응집도가 커지면 클러스터간의 유사도 또한 높아지고 응집도가 낮아지면 클러스터간의 유사도도 낮아지는 것을 알 수 있다. 이것은 지지도가 커질수록 클러스터 유지를 위해, 주요항목을 포함하는 문서가 더 많이 요구되므로 응집도의 유지를 위한 적정 크기 이상의 클러스터를 생성하게 되는 것으로 판단된다. 그러므로 높은 클러스터의 응집도와 낮은 클러스터간의 유사도를 위해서는 적절한 지지도가 주어져야 전체적으로 양질의 클러스터를 생성할 수 있음을 알 수 있다.

위의 실험 결과와 더불어 생성된 클러스터를 살펴보면 각 분야의 문서에서 공통으로 사용되는 엘리먼트 즉, <name>, <year>, <title>, <address> 등이 많이 존재하지만 문서별로 루트 엘리먼트가 다르게 선택되어 구성되면, 문서의 부분 경로가 같은 엘리먼트로 구성되어 있을 지라도 문서의 전체 경로 구조는 다르게 구별된다. 그러므로 루트 엘리먼트를 제외한 구조 경로의 엘리먼트들을 대상으로 할 것인지, 동일 의미의 엘리먼트를 같은 엘리먼트로 취급할 것인지에 대한 결정이 중요하고 문서의 분류에 많은 영향을 준다.

7. 결 론

이 논문에서는 다양한 구조를 가지는 XML 문서의 경로 구조를 중심으로 빈발 구조에 대한 유사성 기반의 점진적 클러스터링 기법을 제안하였다. 이 제안 기법은 문서 구조의 특성에 의해 그룹화 하므로 문서의 분류 및 저장, 그리고 사용자의 XML 문서 검색에 대한 결과를 효율적으로 제공하기 위한 기반 연구이다. 제안 기법에서는 먼저, XML 문서를 구성하는 엘리먼트의 순서와 발생 빈도를 동시에 고려할 수 있는 순차패턴을 이용하여 일정한 지지도를 만족하는 빈발 구조 패턴을 추출하였다. 그리고 추출된 구조를 기준으로 유사 구조 문서를 그룹화 하여 주요항목 기반의 클러스터를 생성하고, 클러스터 할당 이익에 대한 차분 연산을 통해 점진적 클러스터링을 수행하였다.

기존 연구에서는 단지 공통 항목의 비율만을 가지고 클러스터링을 수행하므로 써 적정 크기 이상의 클러스터 생성과 클러스터간의 유사도가 커질 수 있는 문제점이 있었다. 이러한 문제점을 개선하기 위하여 이 논문에서는 클러스터내의 개별항목을 고려하는 주요항목을 구성하고 이를 이용하는 클러스터의 참여도에 의해 할당 가능한 클러스터에 대해서만 할당 이익을 비교하여 클러스터링 하였고, 적정 크기의 클러스터 생성을 조절할 수 있도록 하였다.

기존 연구와의 비교 실험에서 이 논문의 제안기법은 수행시간을 많이 감소시킬 수 있었음을 알 수 있었고, 클러스터링 결과에서도 기존 방법보다 평균적으로 더 높은 클러스터의 응집도와 더 낮은 클러스터간의 유사도를 나타내었다.

이 연구는 XML 문서의 계층적 엘리먼트 경로를 이용한 구조적 특성에 기반하므로 구조 중심의 문서 분류에 사용될 수 있으며, XML 문서의 전체적인 형식에 의한 분류, XML 문서의 주요 엘리먼트에 의한 분류 및 문서의 병합 여부를 위한 구조 검색과 유사 구조 문서의 저장 관리에 효율적으로 적용 가능하다.

향후에는 이 연구를 기반으로 XML 문서의 효율적인 구조 검색의 적용 및 구조 검색을 위한 질의 처리 연구가 진행될 것이다.

참 고 문 헌

[1] W3C, Extensible Markup Language(XML) 1.1. http://www.w3.org/TR/xml11, W3C Working Draft. April 2002.

[2] P. Kotasek, J. Zendulka, "An XML Framework Proposal for Knowledge Discovery in Database," European Conference on Principles and Practice Knowledge Discovery in Databases, 2000.

[3] K. Wang, H. Liu, "Discovery Typical Structures of Documents: A Road Map Approach," In Proceedings of ACM SIGIR Conference on Information Retrieval, pp.146-154, 1998.

[4] J. Widom, "Data Management for XML: Research Directions," IEEE Computer Society Technical Committee on Data Engineering, pp.44-52, 1999.

[5] R. Nayak, R. Witt, A. Tonev, "Data Mining and XML Documents," International Conference on Internet Computing, pp.660-666, 2002.

[6] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, "Efficient Substructure Discovery from Large Semi-structured Data," In Proceedings of SIAM International Conference on Data Mining, pp.158-174, 2002.

[7] A. P. Asirvatham, K. K. Ravi, "Web Page Classification based on Document Structure," In National Level Student Paper Contest, conducted by IEEE India Council, 2001.

[8] J. T. Wang, D. Shasha, G. J. S. Chang, "Structural Matching and Discovery in Document Databases," In Proceedings of International Conference of ACM SIGMOD on Management of Data, pp.560-563, 1997.

[9] W. Chiu, A. Wai-chee, "Incremental Document Clustering for Web Page Classification," In Proceedings of IEEE 2000 International Conference on Information Society in the 21st Century: Emerging Technologies and New Challenges, 2000.

[10] M. Ester, H. P. Kriegel, J. Sander, M. Wimmer, X. Xu, "Incremental Clustering for Mining in a Data Warehousing Environment," In Proceedings of International Conference on VLDB, pp.323-333, 1998.

[11] M. L. Lee, L. H. Yang, W. Hsu, X. Yang, "XClust: Clustering XML Schemas for Effective Integration," In Proceedings of ACM International Conference on Information and Knowledge Management, pp.292-299, 2002.

[12] M. Zaki, "Efficiently Mining Frequent Tree in a Forest," In Proceedings of ACM SIGKDD International Conference, pp.71-80, July 2002.

[13] J. Pei, J. Han, B. M. Asi, H. Pinto, "PrefixSpan: Mining Sequential Pattern Efficiently by Prefix-Projected Pattern Growth," In Proceedings of International Conference of Data Engineering(ICDE), pp.215-224, 2001.

[14] Y. Yang, X. Guan, J You, "CLOPE : A fast and effective clustering algorithm for transaction data," In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.682-687, 2002.

[15] K. Wang, C. Xu, "Clustering Transactions Using Large Items," In Proceedings of ACM CIKM International Conference, pp.483-490, 1999.

[16] J. W. Lee, K. Lee, W. Kim, "Preparation for Semantics-Based XML Mining," In Proceedings of IEEE International Conference on Data Mining (ICDM), pp.345-352, 2001.

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

[18] C. H. Moh, E. P. Lim, W. K. Ng, "DTD-Miner: A Tool for Mining DTD from XML Document," In Proceedings of International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS), pp.144-151, 2000.

[19] A. G. Buchner, M. Baumgarten, M. D. Mulvenna, R. Bohm, S. S. Anand, "Data Mining and XML: Current and Future Issues," In Proceedings of WISE International Conference, pp.131-135, 2000.

[20] A. Termier, M. C. Rouster, M. Sebag, "TreeFinder: A First Step towards XML Data Mining," In Proceedings of IEEE International Conference on Data Mining (ICDM), pp.450-457, 2002.

[21] J. Yoon, V. Raghavan, V. Chakilam, "BitCube: Clustering and Statistical Analysis for XML Documents," In Proceedings of International Conference on Scientific and Statistical Database Management, pp.241-254, 2001.

[22] J. H. Hwang, K. H. Ryu, "XML Document Clustering Based on Sequence Pattern," Journal of KIPS, (D), Vol. 10, No. 7, pp.1093-1102, 2003.

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

황 정 희

1991년 충북대학교 전산통계학과(이학사) 2001년 충북대학교 대학원 전자계산학과(이학석사). 2001년~현재 충북대학교 대학원 전자계산학과 박사과정. 관심분야는 XML, 데이타 마이닝, 능동 데이타베이스, 시공간 데이타베이스

류 근 호

정보과학회논문지 : 데이타베이스
제 31 권 제 1 호 참조