논문 2010-47CI-5-2

비트벡터에 기반한 XML 문서 군집화 기법

(XML Documents Clustering Technique Based on Bit Vector)

김 우 생*

(Woosaeng Kim)

요 약

XML은 점점 데이터 교환과 정보 관리에서 중요하게 여겨진다. 따라서 XML 문서들을 접근, 질의, 저장하는 효율적인 방법들을 개발하기 위한 많은 노력이 진행되고 있다. 본 논문은 XML 문서들을 효율적으로 군집화 하는 새로운 기법을 제안한다. XML 문서를 군집화하기 위해 문서를 대표하는 비트 벡터를 제안한다. 두 XML 문서의 유사도는 대응하는 두 비트 벡터간의 bit-wise AND 연산에 의해서 측정된다. 실험 결과 XML 문서의 특징으로 비트 벡터가 사용되었을 때 군집화가 제대로 그리고 효율적으로 형성됨을 알 수 있다.

Abstract

XML is increasingly important in data exchange and information management. A large amount of efforts have been spent in developing efficient techniques for accessing, querying, and storing XML documents. In this paper, we propose a new method to cluster XML documents efficiently. A bit vector which represents a XML document is proposed to cluster the XML documents. The similarity between two XML documents is measured by a bit-wise AND operation between two corresponding bit vectors. The experiment shows that the clusters are formed well and efficiently when a bit vector is used for the feature of a XML document.

Keywords: bit vector, hierarchical clustering, XML, XML clustering.

I. 서 론

인터넷의 성장은 전 세계에 존재하는 모든 데이터와 정보의 접근을 쉽게 만들면서 많은 데이터들이 다양한 형태의 정보로 생성되는데 이바지하고 있다. 인터넷이 점점 성장하고 발전할수록, 더 많은 정보들은 XML과 같이 구조적으로 풍부한 문서 형태로 존재하게 되었다. 웹에서 문서가 많아질수록, 이와 같이 구조적으로 풍부한 문서들을 자동적으로 검색하고 관리하는 응용들이 요구된다.

XML은 원소들을 포함하며 각 원소는 시작 태그 종료 태그, 두 태그 사이의 텍스트 데이터로 구성된다. 사용자가 태그를 자유롭게 정의할 수 있기 때문에, 태그는 데이터를 표현할 뿐 아니라 데이터의 의미를 표현할 수 있다. XML의 원소들은 내포 구조로 구성되기 때문에, XML은 정렬된 라벨 트리로 모델링 될 수 있다[1]. 이 트리에서 각 노드는 문서의 태그와 대응하며 태그의 이름으로 라벨 된다. 트리의 부모와 자식 노드간의 관계는 문서에서 내포 관계에 대응한다.

데이터 군집화의 일반적인 목적은 다음 단계의 데이터 처리를 위해 여러 종류의 데이터들로부터 연관된 정보를 추출하는 것이다. 군집화 된 데이터들은 데이터들 간에 일종의 경향 또는 규칙성을 보이고 심지어 주목할 가치가 있는 관련 지식을 보여주기까지 한다. XML 문서들에 대한 군집화는 유사한 문서들의 그룹을 만들어 특정한 카타고리 안에서 검색과 처리를 용이하게 하기 위함이다. XML 문서에 대한 적절한 군집화는 체계적인 문서 관리와 문서 저장을 위해서도 효율적이다. 또한

* 정회원, 광운대학교 컴퓨터소프트웨어학과
(Dept. of Computer Software, Kwangwoon Univ.)
※ 본 논문은 2010년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음.
접수일자: 2010년4월5일, 수정완료일: 2010년8월31일

일반적이지 않은 문서가 쉽게 발견되기에 시스템 보호에도 도움이 된다.

본 논문은 XML 문서들을 효율적으로 군집화하기 위해서 XML 문서의 특징으로 비트벡터를 제안한다. 비트벡터는 XML 문서에 대응하는 트리의 경로를 통해서 만들어진다. 두 문서의 유사도는 대응하는 두 비트벡터들간의 bit-wise AND 연산을 통하여 측정된다. 본 논문은 같은 DTD들에서 생성된 문서들을 유사한 문서로 간주하여 같은 그룹으로 군집화를 시킨다. 본 논문은 비트벡터를 사용하여 XML 문서들의 DTD 정보가 존재할 때와 존재하지 않을 때 문서들을 군집화 하는 두 가지 다른 방법들을 제안한다. 실험 결과는 제안하는 방법들을 통해 군집화가 잘 이루어지고 효율적임을 보인다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 XML 군집화 관련 연구에 대해 기술하고, Ⅲ 장은 XML 문서로부터 비트 벡터를 생성하는 방법을 제안한다. Ⅳ 장은 XML 문서가 DTD 정보를 갖고 있을 때, 문서를 군집화하기 위한 문서 비트 벡터와 DTD 비트 벡터를 제안한다. Ⅴ 장은 XML 문세가 DTD 정보가 없을 때, 문서를 군집화하기 위하여 계층 군집화 기법에 비트 벡터를 적용하는 방법을 제안한다. Ⅵ 장은 실제 데이터를 통한 실험을 통하여 제안한 방법이 효율적인지를 조사한다. 마지막으로 Ⅶ 장은 결론을 낸다.

Ⅱ. 관련 연구

다양한 XML의 증가로 인해 XML 데이터를 조직하고 군집화 하는 필요성이 증대되고 있다. 최근에 XML 문서들의 구조 뿐 아니라 내용을 고려하는 군집화 기법들이 연구되고 있다. 벡터 공간 모델에 표현되는 XML 문서들에 K-means 군집화 기법을 적용하는 연구가 있었다[2]. 이 표현에서 각 문서는 N 차원 벡터로 표현되며, 이 때 N은 문서의 특징들로 텍스트 특징들, 태그 특징들, 그리고 이 둘을 합한 특징들로 구성된다. 이 방법은 오직 XML 문서의 내용만을 고려한다. 문서간의 구조에 대한 공통 구조의 존재 여부를 0, 1로 표현하는 비트를 이용하여 비트맵 인덱스에 기반한 군집화 기법이 제안되었다[3~4]. BitCube는 3 요소 즉, 문서, 경로, 단어의 3 차원 비트맵 인덱스로 표현된다. BitCube 인덱스들은 문서들을 분할해서 군집화하기 위해 bit-wise 거리 척도를 활용한다. 그러나 이 방법은 비트맵 인덱스를 생성하기 위해서 수작업을 필요로 한다. XML 데이터를 위한 특징들로 문서로부터 추출한 내용 정보와 태그 경로들로부터 유도되는 구조 정보들을 사용하는 방법이 제안되었다[5]. 이 방법은 XML 문서 트리들을 트랜잭션 데이터 즉, 분류별의 속성들을 가진 객체들로 사상하는 것을 허용하는 XML 표현 모델의 정의 안에서 트리 투플이라는 개념을 소개하며 군집화 기법이 XML 트랜잭션의 영역 안에서 개발되고 적용되었다. 반면에 XML 문서의 구조를 이산 함수로 변환하는 방법이 연구되었다[6]. 이산 함수는 FFT에 의해서 주파수 영역으로 변환된다. FFT의 결과는 x와 y의 값들을 포함하는 복소수 쌍들이며 n 차원 벡터들로 간주되어 유클리디안 거리 척도를 사용하여 비교된다. 이 방법은 오직 문서들의 구조만을 고려한다. 반면 XML 문서내의 특징들의 발생에 근거해서 문서 트리들을 다차원 유클리디안 공간의 벡터들로 변환한 후 주성분 분석(PCA)을 적용하여 차원을 줄인 후 줄어든 차원 공간에서 K-평균 군집화 알고리즘을 사용하여 적절한 분류를 시도하는 연구가 있었다[7]. 다양한 구조를 가지는 XML 문서의 경로 구조를 중심으로 빈발 고조에 대한 유사성 기반의 점진적 클러스터링 기법을 제안되었다[8]. XML 문서를 구성하는 원소의 순서와 발생 빈도를 동시에 고려할 수 있는 순차패턴을 이용하여 일정한 지지도를 만족하는 빈발 구조 패턴을 추출하여 유사 구조 문서를 그룹화 하여 주요 항목 기반의 클러스터를 생성하고, 클러스터 할당 이익에 대한 연산을 통해 점진적 클러스터링을 수행하였다.

Ⅲ. 유사한 XML 문서와 비트벡터

같은 DTD에서 생성된 XML 문서들은 유사한 문서들이기에 이들의 군집화를 시도한다. 예를 들어, 그림 1의Actor DTD에 의해 생성된 그림 2(a), 2(b), 2(c)는 모두 유사한 문서들이다. 본 논문에서는 XML 문서를 대응하는 트리의 루트로부터 리프 노드들의 모든 경로들로 표현한다. 따라서 그림 2(a)의 문서 A1은 actor/name/lname, actor/filmography/movie/title, actor/filmography/movie/year 3개의 경로들로 표현된다. 반면에 그림 2(b)의 문서 A2는 actor/name/fname, actor/name/lname, actor/filmography/movie/title, actor/filmography/movie/year 4개의 경로들로 표현된다. 한편 그림 2(c)의 문서 A3은 그림 2(b)와 구조가 다르지

```
Actor
<!ELEMENT actor (name, filmography*)>
<!ELEMENT name (fname?, lname)>
<!ELEMENT fname (#PCDATA)>
<!ELEMENT lname (#PCDATA)>
<!ELEMENT filmography (movie)>
<!ELEMENT movie (title, year)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA)>
```

그림 1. Actor DTD
Fig. 1. Actor DTD.

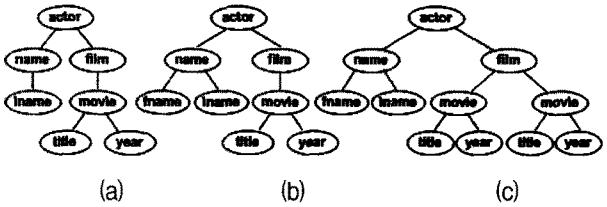

그림 2. Actor DTD에 대응하는 트리들
Fig. 2. Trees correspond to Actor DTD.

표 1. 문서 비트벡터들
Table 1. Document Bit Vectors.

	P1	P2	P3	P4
A1	0	1	1	1
A2	1	1	1	1
A3	1	1	1	1

만 그림 2(b)와 같은 4개의 경로들로 표현된다.

따라서 문서 A1, A2, A3에 의해 생성되는 4개 경로actor/name/fname, actor/name/lname, actor/filmography/movie/title, actor/filmography/movie/year를 각각 P1, P2, P3, P4라고 표기할 때, 3개의 문서들에 대해 만들어지는 문서 비트벡터들은 표 1과 같다. 본 논문은 이러한 문서 비트벡들을 사용하여 문서들을 군집화 하는 방법을 제안한다.

Ⅳ. DTD 정보가 있을 때의 군집화 기법

XML 문서에 DTD가 있을 경우에는 먼저 DTD에 대응하는 트리의 경로들을 통해 DTD 비트벡터를 만든다. 이 때 DTD에 포함될 수 있는 특수 기호인 ?, +, *의 경우에는 해당 원소(element)가 1번만 발생하는 것으로 간주한다. 따라서 그림 1의Actor DTD는 actor/mame/fname,actor/name/lname,actor/filmography/movie/title, actor/filmography/movie/year의 4개 경로들로 표현된다. 이와 같은 방법으로 주어진 DTD들에 대해서 DTD 비트벡터들을 만든다. 예를 들어서 Star_Fan DTD가 그림 3과 같을 때 대응하는 트리로부터 생성되는 7개의 경로들은 다음과 같다: actor/name/fname, actor/name/lname.actor/club/name,actor/club/tel, actor/club/addr,actor/club/fan/name,actor/club/fan/tel. 따라서 Actor DTD와 Star_Fan DTD 의 9개의 경로들을 P1=actor/mame/fname, P2=actor/name/lname, P3=actor/filmography/movie/title,P4=actor/filmography/ movie/year, P5=actor/club/name, P6=actor/club/tel,P7=actor/club/addr,P8=actor/club/fan/name, P9=actor/club/fan/tel로 표기할 때 2개의 DTD에 의해 만들어지는 DTD 비트벡터들은 표 2와 같다.

다음으로 문서들은 Ⅲ장에 나온 것과 같은 방법으로 각 문서에 대응하는 트리의 경로들로부터 문서 비트벡터들을 만든다. 마지막으로 군집화하고자 하는 문서의 문서 비트벡터를 모든 DTD 비트벡터들과 bit-wise AND 연산을 통해서 값이 가장 큰 DTD의 문서들로 군집화 한다. 예를 들어, 그림 2(a)의 문서 A1은 다음과

```
Star_Fan
    <!ELEMENT actor (name, club*)>
    <!ELEMENT name (fname?, lname)>
    <!ELEMENT fname (#PCDATA)>
    <!ELEMENT lname (#PCDATA)>
    <!ELEMENT club (name, tel, addr, fan*)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT tel (#PCDATA)>
    <!ELEMENT addr (#PCDATA)>
    <!ELEMENT fan (name, tel)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT tel (#PCDATA)>
```

그림 3. Star_Fan DTD
Fig. 3. Star_Fan DTD.

표 2. DTD 비트벡터들
Table 2. DTD Bit Vectors.

	P1	P2	P3	P4	P5	P6	P7	P8	P9
Actor	1	1	1	1	0	0	0	0	0
S_F	1	1	0	0	1	1	1	1	1

같은 방법으로 군집화가 된다. (i) A1의 문서 비트벡터가 만들어진다. 9개의 경로들로 구성된 문서 비트벡터에서 A1에 대응하는 트리의 경로들이 P2, P3, P4이므로, 이 경로들은 1의 값을 갖고 나머지 경로들은 0의 값을 갖는다. 따라서 A1 문서 비트벡터는 011100000 된다. (ii) A1 문서 비트벡터를 표 2의 Actor와 Star_Fan DTD 비트벡터들과 bit-wise AND 연산을 수행한다. (iii) Actor DTD 비트벡터와의 bit-wise AND 연산의 결과는 3이지만 Star_Fan DTD 비트벡터와의 bit-wise AND 연산의 결과는 1이므로, A1 문서는 Actor 문서들로 군집화가 된다.

V. DTD 정보가 없을 때의 군집화 기법

본 논문은 DTD가 없는 XML 문서의 군집화를 위해서 계층 군집화 기법을 사용한다. 계층 군집화 기법은 중첩(nested) 군집들의 계층 구조를 생성한다. 만약 R_1 내의 각 군집이 R_2의 진부분 집합이면, k개의 군집을 포함하는 군집 R_1은 r ($<k$)개의 군집을 포함하는 군집 R_2에 중첩된다고 말한다. 계층 군집화 기법은 군집을 형성하는 방향에 따라서 응집형(agglomerative)과 분리형(divisive)으로 구분된다. 모든 패턴의 숫자를 n이라 가정할 때 일반적인 응집형 군집화 기법은 다음과 같다.

응집형 군집화 기법에서는 군집 간 유사도를 결정하는 방법에 따라 단일 연결(single-linkage), 완전 연결(complete-linkage) 알고리즘 등이 존재한다. 단일 연결 알고리즘에서는 두 군집 사이의 거리는 각 군집에 있는 패턴 간 거리 중에서 가장 짧은 거리로 정의한다. 만약 C_i와 C_j가 군집이라면, 그들 간 거리 d_{SL}은 다음과 같이 정의된다. 여기서 $d(X,Y)$는 패턴 X와 Y 사이의 거리를 나타낸다.

$$d_{SL}(C_i, C_j) = \min_{X \in C_i, Y \in C_j} d(X, Y) \quad (1)$$

① 각 한 개의 패턴으로 구성된 n개의 군집들로부터 시작한다.
② 단계 ③을 n-1 번 반복한다.
③ 가장 유사한 군집 쌍 Ci 와 Cj 를 하나의 군집으로 합친다. 만약 하나 이상의 쌍이 존재한다면 첫 번째 쌍을 합친다.

그림 4. 응집형 군집화 기법
Fig. 4. Agglomerative clustering method.

반면 완전 연결 알고리즘은 두 군집 사이의 거리를 각 군집에 있는 패턴 간 거리 중에서 가장 긴 거리로 정의한다. 만약 C_i와 C_j가 군집이라면, 그들 간 거리 d_{CL}은 다음과 같이 정의된다. 여기서 $d(X,Y)$는 패턴 X와 Y 사이의 거리를 나타낸다.

$$d_{CL}(C_i, C_j) = \max_{X \in C_i, Y \in C_j} d(X, Y) \quad (2)$$

군집화 과정을 위해 중요한 하나의 이슈는 패턴간의 유사도를 어떻게 정량화하는가에 있다. 패턴간의 유사성 (또는 비유사성)을 측정하는 가장 일반적인 척도는 거리, 특별히 유클리디안 거리를 사용한다. 그러나 본 논문에서는 문서를 비트벡터로 표현하기 때문에, 비트벡터를 사용해서 문서간의 유사도를 나타내고자 한다. 같은 DTD에 의해 생성되는 유사한 문서들은 많은 공통의 경로들을 갖기 때문에 그들 간의 bit-wise AND 연산은 큰 값을 갖는다. 즉, 두 문서가 유사하면 대응하는 문서 비트벡터간에 공유하는 1의 비트가 많아지고 bit-wise AND 연산은 큰 값을 갖게 된다. 따라서 본 논문에서는 유사도를 두 문서 비트벡터간의 bit-wise AND 연산의 값으로 정량화한다. 두 XML 문서 X와 Y의 대응하는 문서 비트벡터들을 V_x 와 V_y로 표기할 때, 두 문서의 유사도 $f(X, Y)$는 식 (3)과 같이 정의된다.

$$f(X, Y) = bit-wise\ AND(V_X, V_Y) \quad (3)$$

예를 들어, 2개의 Star_Fan 문서 S1과 S2가 있고 S1과 S2에 대응하는 트리의 경로들은 다음과 같다고 가정한다. S1= {actor/fname, actor/lname, actor/club/name, actor/club/tel, actor/club/addr}. S2 = {actor/fname, actor/lname,actor/club/name,actor/club/tel, actor/club/addr,actor/club/fan/name, actor/club/fan/tel}. 또한 군집화를 수행하는 문서들이 A1, S1, A2, A3, S2의 순서로 들어온다고 가정한다. 이 문서들에 의해 생성되는 경로들을 A1={P1, P2, P3}, S1={P4, P1, P5, P6, P7}, A2 = {P4, P1, P2, P3}, A3 = { P4, P1, P2, P3}, S2 = {P4, P1, P5, P6, P7, P8, P9}라고 표기 할 때 5개의 문서에 의해 만들어지는 문서 비트벡터들은 표 3과 같다.

표 3에서 가장 유사한 2개의 군집은 S1과 S2이다. 왜냐하면 두 군집간의 bit-wise AND 연산의 결과는 5로 다른 어떤 2개의 군집간의 bit-wise AND 연산의 결과보다 크기 때문이다. 따라서 {S1, S2}를 하나의 군집으

표 3. 문서 비트벡터들
Table 3. Document Bit Vectors.

	P1	P2	P3	P4	P5	P6	P7	P8	P9
A1	1	1	1	0	0	0	0	0	0
S1	1	0	0	1	1	1	1	0	0
A2	1	1	1	1	0	0	0	0	0
A3	1	1	1	1	0	0	0	0	0
S2	1	0	0	1	1	1	1	1	1

로 형성한다. 마찬가지 방법으로 다음으로 가장 유사한 2개의 군집은 bit-wise AND 연산의 결과가 4인 A2와 A3이기 때문에 {A2, A3}을 하나의 군집으로 형성한다. 마지막으로 가장 유사한 2개의 군집은 bit-wise AND 연산의 결과가 3인 A1과 {A2, A3}이기 때문에 {A1, A2, A3}을 하나의 군집으로 형성한다. 따라서 5개의 문서들에 대해 2개의 군집화를 수행하면 {A1, A2, A3}과 {S1, S2}로 제대로 군집이 이루어짐을 알 수 있다.

VI. 실험 및 분석

본 연구는 위스콘신 대학의 XML 데이터 뱅크에서 제공하는 데이터들을 사용하여 제안하는 방법의 효율성을 실험하였다[9]. 이 데이터 뱅크는 bibliography, club, company profiles, stock quotes, department, personal information, movies, actor와 같은 8개의 DTD를 제공한다. XML 문서들의 군집화를 실험하기 위해 각 DTD에 대하여 10개씩의 문서들을 생성하여 모두 80개의 문서들을 사용하였다.

1. DTD의 정보가 있을 때의 군집화 결과

DTD의 정보가 있는 XML 문서들을 군집화하기 위하여 각 DTD에 대응하는 DTD 비트벡터와 각 문서에 대응하는 문서 비트벡터를 만들어 사용하였다. 그림 5는 80개의 문서들에 대한 군집화의 결과이다. 각 DTD에 대하여 제대로 군집화 된 문서들의 수(NCC)와 잘못 군집화 된 문서들의 수(NIC)를 계산하였다. 그림에서 보는 것처럼 모든 문서들이 제대로 군집화가 되는 것을 알 수 있다. 이것은 당연한 결과로서 서로 다른 DTD들에 의해 생성되는 문서들간에 경로들이 같은 경우가 일부 있을 수 있으나 대부분은 서로 다르기 때문에, 주어진 문서의 문서 비트벡터는 대응하는 DTD 비트벡터와

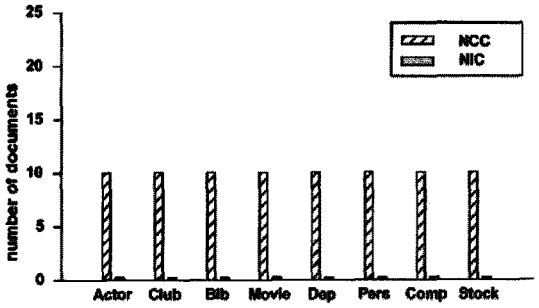

그림 5. 군집화된 문서들의 숫자
Fig. 5. Number of Clustering documents.

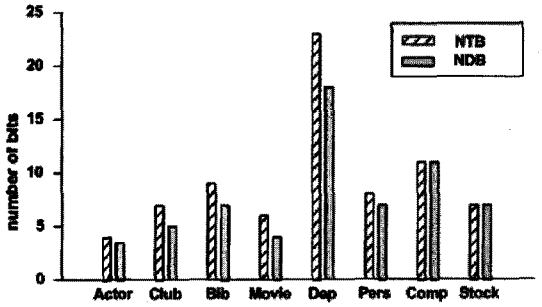

그림 6. DTD와 문서 비트벡터들의 1의 비트 수
Fig. 6. Number of 1's bits of DTD and document bit vectors.

가장 큰 bit-wise AND 연산 값을 갖기 때문이다.

그림 6은 각 DTD 비트벡터의 1의 비트 수(NTB), 즉 경로들 수와 해당 그룹에 속한 문서들의 문서 비트벡터들의 1의 평균 비트 수(NDB)이다. 그림에서 Department가 가장 큰 값을 갖는 이유는 Department DTD가 가장 많은 #PCDATA를 갖고 있기 때문이다. 즉, Department DTD에 대응하는 트리가 가장 많은 리프 노드들을 갖기 때문이다. 그림에서 Company와 Stock의 DTD 비트벡터의 1의 비트 수와 해당 그룹에 속한 문서들의 문서 비트벡터들 1의 평균 비트 수가 같은 이유는 Company나 StockDTD에는 특수 기호인 ?나 *가 포함되지 않았기 때문에 DTD에 표기된 모든 원소가 DTD에 의해 생성되는 문서에도 나오기 때문이다.

2. DTD의 정보가 없을 때의 군집화 결과

DTD의 정보가 없는 XML 문서들을 군집화하기 위하여 각 문서에 대응하는 문서 비트벡터를 만들어 사용하였다. 그림 7은 80개의 문서들에 대해 계층 군집화의 단일 연결 알고리즘을 적용한 성장 그래프이다. 그림에서 가로축은 문서이며 세로축은 유사도 척도이다. 각 문서는 이해를 돕기 위하여 다음의 번호로 표기하였다: Actor (1~5), Club (6~10), Bibliography(11~15),

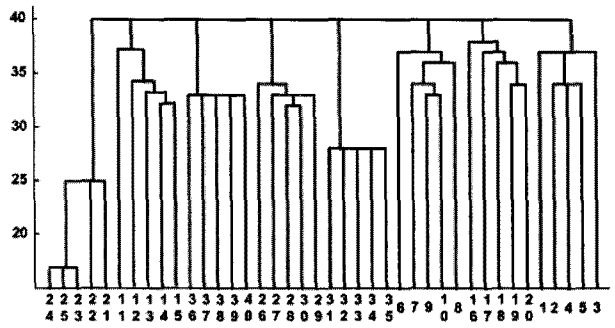

그림 7. 단일연결 알고리즘에 의한 성장 그래프
Fig. 7. Dendrogram by single-linkage algorithm.

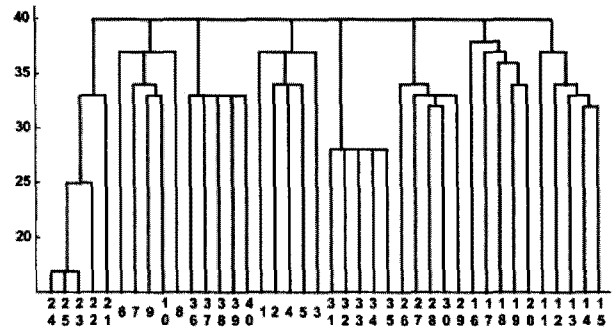

그림 8. 완전연결 알고리즘에 의한 생성 그래프
Fig. 8. Dendrogram by complete-linkage algorithm.

Movie (16~20), Department (21~25), Personal (26~30), Company (31~35), Stock (36~40). 그림에서 보듯이 모든 문서들은 제대로 군집화가 되었음을 알 수가 있다. 그림에서 Department 문서들(21~25)이 가장 먼저 군집화가 되는 것을 알 수 있다. 이것은 Department DTD가 #PCDATA를 가장 많이 갖고 있기 때문에 해당 문서들의 문서 비트벡터들은 1비트를 가장 많이 포함하고 있기 때문이다.

그림 8은 80개의 문서들에 대해 계층 군집화의 완전연결 알고리즘을 적용한 성장 그래프이다. 그림에서 보듯이 군집화는 제대로 되지만 군집화 되는 순서는 그림 7과는 약간 다른 것을 알 수 있다. 예를 들어, 그림 7에서는 모든 Department 문서들(21-25)이 가장 먼저 군집화 되나 그림 8에서는 Department 문서 21의 경우 Company (31-35) 문서들이 다 군집화 된 후에 Department의 다른 문서들과 군집화 되는 것을 알 수가 있다. 그 이유는 완전 연결 알고리즘의 경우 두 군집 사이의 거리는 각 군집에 있는 패턴 간 거리 중에서 가장 긴 거리로 정의하기 때문이다. Department 문서들의 경우 교직원, 학부, 대학원 정보로 구성되나 문서에 따라 학부 또는 대학원 정보만 가질 수도 있다. 일부 Department 문서들(23~25)의 경우는 학부와 대학원 정보를 모두 갖고 있기 때문에 공유하는 1의 비트가 많아 먼저 군집화가 된다. 그러나 문서 22는 대학원 정보만 갖고 있고 문서 21은 학부 정보만 갖고 있기 때문에 문서 21이 가장 먼 거리에 있는 문서 22와 군집화를 할 때에는 공유하는 1의 비트가 작아 Company 문서들보다 늦게 군집이 수행됨을 알 수 있다.

Ⅶ. 결 론

우리는 XML 문서들을 군집화하기 위한 문서의 특징으로 비트벡터를 제안하였다. 두 문서의 유사도는 대응하는 두 비트벡터들간의 bit-wise AND 연산에 의하여 측정된다. 본 논문은 DTD 정보가 있는 XML 문서들의 경우에는 문서 비트벡터와 DTD 비트벡터를 사용하여 군집화 하는 방법을 제안하였다. 반면 DTD 정보가 없는 XML 문서들의 경우에는 계층 군집화 기법에 문서 비트벡터를 적용하는 방법을 제안하였다. 실험 결과 본 논문에서 제안하는 방법은 군집화를 제대로 형성하며 효율적으로 수행하였음을 보여 주었다.

참 고 문 헌

[1] R. Behrens, "A Grammar based model for XML schema integration" *Proc. of the 17th British National Conf. on Databases*, pp.172-190, 2000.

[2] A. Doucet, H. Ahonen-Myka, "Naive clustering of a large XML document collection" *Proc. 1st Annual Workshop of the Initiative for the Evaluation of XML retrieval*(INEX), Germany, pp.81-88, 2002.

[3] J. Yoon, V. Raghavan, V. Chakilam, "BitCube: clustering and statistical analysis for XML documents": *Proc. of the 13th Int. Conf. on Scientific and Statistical Database Management*, Fairfax, Virginia, 2001.

[4] J. Yoon, V. Raghavan, V. Chakilam, L. Kerschberg, "BitCube: a 3-D bitmap indexing for XML documents" *Journal of Intelligent Information Systems*, Vol. 17, pp.241-254, 2001.

[5] A. Tagarelli, A. Greco, "Toward semantic XML clustering" *6th SIAM International Conference on Data Mining*(SDM '06), pp. 188-199. Bethesda, Maryland, USA, 2006.

[6] H. Lee, "An Unsupervised clustering technique of XML documents based on function transform and FFT" *Journal of Korea Information*

Processing Society, 2007.
[7] J. Liu, T. Jason, L. Wang, W. Hsu, K.G. Herbert, "XML clustering by principal component analysis" *Proc. of the 26th IEEE International Conference on Tools with Artificial Intelligence*, 2004.
[8] 황정희, 류근호 "유사 구조 기반 XML 문서의 점진적 클러스터링" 정보과학회 논문지-데이터베이스 제 31권 제 6호, 2004. 12
[9] Niagara Query Engine, http://www.cs.wisc.edu/niagara/data.html

저 자 소 개

김 우 생(정회원)-교신저자
대한전자공학회논문지,
제46권 CI편 제2호 참조