DOI: 10.3745/KIPSTD.2009.16D.6.845

관계형 DBMS 기반의 XML 데이터를 위한 k-비트맵 클러스터링 기법

이 범 석[†] · 황 병 연[††]

요 약

웹2.0 환경의 발달과 함께 XML 데이터의 사용도 증가하였는데, 특히 블로그나 뉴스 피드의 정보 전달을 위한 RSS나 ATOM 포맷의 기반 기술로 사용되면서 그 장점과 가치를 인정받고 있다. XML 데이터의 인덱싱을 위한 여러 기법들 중 빠른 검색성능을 보인 비트맵 클러스터링은 관계형 DBMS를 기반으로 메모리에 인덱스를 유지하는 기법이다. 기존의 비트맵 클러스터링 기법을 이용하여 XML 데이터를 인덱싱할 때 너무 많은 클러스터가 생성되어 오히려 검색 효율과 결과의 품질이 저하되는 문제점이 있었다. 본 논문에서는 이 문제점을 해결하기 위해 사용자가 제시하는 k개의 클러스터를 생성하는 k-비트맵 클러스터링 기법과 대표비트를 생성할 때 배제된 단어를 검색하기 위한 역인덱스를 함께 유지하는 방법을 제안한다. 성능평가를 수행한 결과 제안하는 기법은 생성되는 클러스터의 수를 임의로 설정할 수 있을 뿐만 아니라 단일 단어 검색에서 높은 재현율을 보였고, 2개의 인덱스를 함께 유지할 때에는 질의에 대해 모든 관련된 문서의 반환을 보장하였다.

키워드 : XML, 비트맵, 클러스터링, 인덱싱, 관계형 데이터베이스 관리 시스템, 성능평가

k-Bitmap Clustering Method for XML Data based on Relational DBMS

Bum-Suk Lee[†] · Byung-Yeon Hwang[††]

ABSTRACT

Use of XML data has been increased with growth of Web 2.0 environment. XML is recognized its advantages by using based technology of RSS or ATOM for transferring information from blogs and news feed. Bitmap clustering is a method to keep index in main memory based on Relational DBMS, and which performed better than the other XML indexing methods during the evaluation. Existing method generates too many clusters, and it causes deterioration of result of searching quality. This paper proposes k-Bitmap clustering method that can generate user defined k clusters to solve above-mentioned problem. The proposed method also keeps additional inverted index for searching excluded terms from representative bits of k-Bitmap. We performed evaluation and the result shows that the users can control the number of clusters. Also our method has high recall value in single term search, and it guarantees the searching result includes all related documents for its query with keeping two indices.

Keywords : XML, Bitmap, Clustering, Indexing, Relational DBMS, Performance Evaluation

1. 서 론

웹2.0 환경의 발달과 함께 XML 데이터의 사용도 증가하였다. XML은 블로그와 뉴스의 피드를 위한 ATOM 포맷과 XML 기반의 가장 성공적인 포맷으로 소개되고 있는 사이트의 정보 전달을 위한 RSS의 기반기술로 사용되면서 그 장점과 가치를 인정받고 있다[1]. 지난 몇 년 동안 XML 데이터의 저장에 관한 연구는 반구조적인 데이터를 관계형 DBMS에 효율적으로 매핑하는 것에 중점을 두고 진행되었다. 이러한 연구들에는 빠른 비트 연산을 이용한 비트맵인덱스[2-3]와 문서 검색에 좋은 성능을 가진 역인덱스[4-5], 그리고 XML 데이터의 구조적 특징을 반영한 그래프 인덱스[6-8] 등이 있다. 특히 3차원 비트맵인덱스 기법을 적용한 BitCube[2]의 경우 빠른 검색속도 뿐만 아니라 XML 문서의 구조적 유사성을 기반으로 클러스터링을 수행하여 XPath 질의에 효율적으로 대응하는 결과를 보여주었다. 이처럼 다양한 구조를 가지는 XML 데이터의 처리는 이질적인 데이터의 교환이나, 웹서비스 통합 등의 분야에서 좋은 결과를

※ 이 논문은 2007년 정부(교육과학기술부)의 재원으로 한국연구재단(KRF-2007-521-D00400) 연구비와 2009년 가톨릭대학교 교비연구비의 지원으로 이루어졌음.
† 준 회 원 : 가톨릭대학교 컴퓨터공학과 박사과정
†† 종신회원 : 가톨릭대학교 컴퓨터정보공학부 교수
논문접수 : 2009년 6월 9일
수 정 일 : 1차 2009년 8월 3일, 2차 2009년 8월 31일
심사완료 : 2009년 8월 31일

가져올 수 있다.

하지만 최근 사용되는 XML 데이터는 RSS, ATOM과 같은 피드 표준을 비롯해 GML과 같은 지리데이터, CML과 같은 화학정보, SBML과 같은 생물정보데이터와 같이 XML을 기반으로 표준화된 스키마가 존재하는 경우가 많다. 이러한 경우에 검색 속도를 좌우하는 것은 특정한 경로에 존재하는 내용을 얼마나 잘 인덱싱 하느냐에 따라 결정될 수 있다.

비트맵 클러스터링은 관계형 DBMS를 기반으로 메모리에 인덱스를 유지하는 기법이다. 기존의 비트맵 클러스터링 기법은 사용자가 지정한 임계값을 기반으로 클러스터링이 수행되기 때문에 몇 개의 클러스터가 생성될 지 알 수 없고, 특히 임의의 데이터가 아닌 실제 블로그에서 수집한 20만개 이상의 웹 문서를 대상으로 내용기반 클러스터링을 하였을 때에는 아주 작은 임계값에서도 수 천 개의 클러스터가 생성되는 문제점이 발생하였다. 본 논문에서는 이러한 문제점을 해결하기 위해 사용자가 제시한 k개의 클러스터를 생성하는 k-비트맵 클러스터링 기법을 제안한다. 제안하는 방법은 검색품질의 향상과 안정적인 클러스터링이 수행되는 결과를 보였다. 또한 k-비트맵 클러스터링을 수행함과 동시에 대표 비트에서 배제된 모든 단어를 역인덱스로 생성해 두 개의 인덱스를 유지하는 방법도 함께 제안하였다. 이 방법은 질의에 대해 항상 모든 관련 문서의 반환을 보장한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 비트맵 클러스터링 기법을 설명하고, 그 문제점을 정리한다. 3장에서는 k-비트맵 클러스터링과 역인덱스를 이용한 2단계 인덱싱 기법을 소개한다. 4장에서는 구현 및 실험을 통해 기존의 방법과 제안한 방법의 성능평가 결과를 제시한다. 5장에서는 결론과 향후 연구 계획에 대해 정리한다.

2. 관련 연구

2.1 3차원 비트맵인덱스

XML 데이터의 효율적인 검색을 위한 클러스터링 기법은 경로 유사도에 기반한 클러스터링 기법[9]과 내용의 유사도에 기반한 클러스터링 기법[10]으로 나누어진다. 3차원 비트맵인덱스 기법은 문서, 경로, 내용을 축으로 3차원의 비트맵을 구성하고 경로 유사도를 기준으로 인덱스를 클러스터링한다. 이 방법은 비트 연산을 이용해 빠른 검색성능을 보장하였지만, 인덱스가 3차원으로 구성되기 때문에 저장되는 XML 문서가 많아지면 메모리의 점유가 급격히 증가하는 문제점이 발생하였고, 경로 비트맵인덱싱[11]은 이러한 문제점을 해결하기 위해 포인터를 이용하였다. (그림 1)은 3차원 비트맵인덱스와 경로 비트맵인덱스의 메모리 구조를 표현한 것으로 경로 비트맵인덱스의 메모리 점유가 상당히 낮아진 것을 알 수 있다.

2.2 비트맵 클러스터링

비트맵 클러스터링 기법[12]은 트랜잭션과 아이템을 축으로 2차원 비트맵을 생성하고 아이템을 공유한 정도에 따라 트랜잭션의 유사도를 계산하여, 유사도에 따라 트랜잭션을 클러스터링하는 방법이다. 비트맵 클러스터링을 위한 유사도는 다음의 식 1을 이용하여 계산된다.

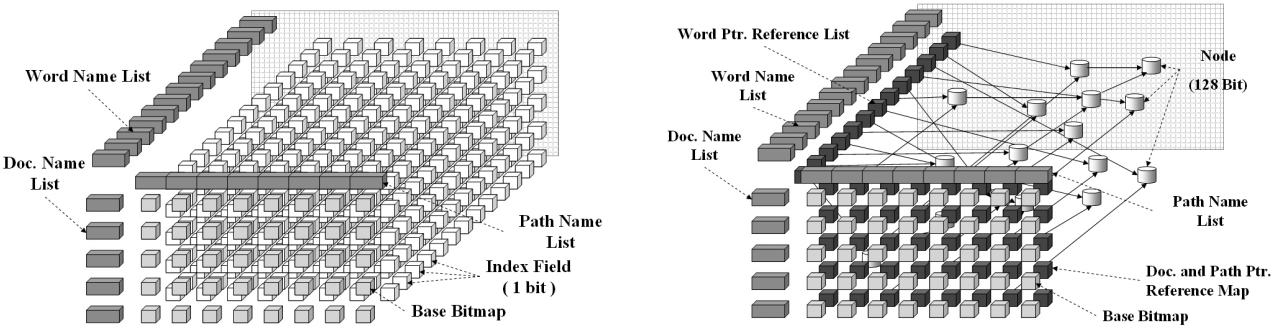
$$Similarity(T_i, T_j) = 1 - \left| \frac{\veebar(T_i, T_j)}{MAX([T_i],[T_j])} \right| \quad \text{식 (1)}$$

식 1에서 두 개의 트랜잭션 T_i와 T_j사이의 유사도 계산은 T_i와 T_j에 포함된 최대 아이템의 수에서 서로 다른 비트를 가지는 아이템의 수($\veebar$:bitwise exclusive OR)의 비율을 1에서 뺀 값이 된다. 트랜잭션의 아이템이 모두 같은 경우 유사도는 1이 되고, 모두 다른 경우에는 0의 유사도를 가진다. 이 클러스터링 기법을 XML 데이터에 적용하면 트랜잭션은 XML 문서가 되고, 아이템은 해당 문서에서 추출한 단어가 된다. (그림 2)는 본 논문의 설명을 위해 사용할 XML 데이터와 그것을 관계형 DBMS테이블에 파싱하여 저장한 모습을 보여준다. (그림 2)에서 문서 d_1과 d_2의 유사도는 1-4/10 =0.6이고, d_1과 d_3의 유사도는 1-3/10=0.7이다.

비트맵 클러스터링을 수행하기 위한 알고리즘은 다음과 같이 정리할 수 있다.

[알고리즘 1] 비트맵 클러스터링

1. 유사도 임계값과 대표비트 임계값을 초기화한다.
2. 문서 1개를 첫 번째 클러스터에 넣고, 대표비트를 생성



(그림 1) 3차원 비트맵인덱스(左)와 경로 비트맵인덱스(右)의 구조 비교

한다.

3. 다음 문서 1개를 불러들여 클러스터의 대표비트와 유사도를 비교한다. 유사도가 유사도 임계값 이상인 클러스터가 있으면 해당 클러스터에 넣고, 유사도 임계값 이상인 클러스터가 없으면 새로운 클러스터를 생성한다.
4. 갱신되거나 새롭게 생성된 클러스터의 대표비트를 생성한다.
5. 모든 문서를 클러스터링 할 때까지 3과 4의 과정을 반복한다.

(그림 3)은 (그림 2)의 예에 대해서 비트맵 클러스터링을 수행하기 전과 수행한 후의 결과를 보여준다. 클러스터링의 유사도 임계값은 0.67이고, 대표비트 임계값은 0.5로 지정하였다.

d_1	<element id="1">t_1 t_3 t_4 t_6 t_8 t_{10}</element>
d_2	<element id="2">t_2 t_3 t_4 t_6 t_8 t_9</element>
d_3	<element id="3">t_1 t_3 t_4 t_5 t_7 t_8 t_{10}</element>
d_4	<element id="4">t_1 t_3 t_4 t_5 t_{10}</element>
d_5	<element id="5">t_2 t_4 t_5 t_6 t_9</element>
d_6	<element id="6">t_2 t_4 t_5 t_6 t_7 t_8 t_9</element>

⇓

docID	elementID	content
d_1	1	t_1 t_3 t_4 t_6 t_8 t_{10}
d_2	2	t_2 t_3 t_4 t_6 t_8 t_9
...		
d_6	6	t_2 t_4 t_5 t_6 t_7 t_8 t_9

(그림 2) XML 데이터와 관계형 DBMS 테이블

【클러스터링 수행 전】

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
d_1	1	0	1	1	0	1	0	1	0	1
d_2	0	1	1	1	0	1	0	1	1	0
d_3	1	0	1	1	1	0	1	1	0	1
d_4	1	0	1	1	1	0	0	0	0	1
d_5	0	1	0	1	1	1	0	0	1	0
d_6	0	1	0	1	1	1	1	1	1	0

⇓ 【클러스터링 수행 후】

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
d_1	1	0	1	1	0	1	0	1	0	1
d_3	1	0	1	1	1	0	1	1	0	1
d_4	1	0	1	1	1	0	0	0	0	1
R_1	1	0	1	1	1	0	0	1	0	1
d_2	0	1	1	1	0	1	0	1	1	0
d_5	0	1	0	1	1	1	0	0	1	0
d_6	0	1	0	1	1	1	1	1	1	0
R_2	0	1	0	1	1	1	0	1	1	0

(그림 3) 비트맵인덱스 클러스터링

2.3 비트맵 클러스터링의 문제점

비트맵 클러스터링 기법의 첫 번째 문제는 실제 데이터에 적용할 때에는 (그림 3)에서 제시한 예에서처럼 적절한 수의 클러스터를 생성하기 위한 임계값의 설정이 어렵다. 적절한 임계값을 찾기 위해서는 반복적인 클러스터링의 수행이 필수적일 뿐만 아니라, 추출된 단어의 수가 많을 경우 임계값은 매우 작은 값을 가지게 된다.

다른 문제는 클러스터링 후의 재현율이 낮아지는 것이다. 재현율은 검색을 위해 입력한 키워드와 관련된 전체 데이터 중에서 검색 결과에 포함된 데이터의 비율을 의미($recall$ = ($retrieved\ data \cap related\ data$) / $related\ data$) 하는데, 너무 많은 클러스터가 생성된 후, 사용자의 검색 요구에 대해 특정 클러스터를 선택하여 결과를 반환하면 재현율이 낮아지게 된다.

3. k-비트맵 클러스터링

3.1 k-비트맵 클러스터링

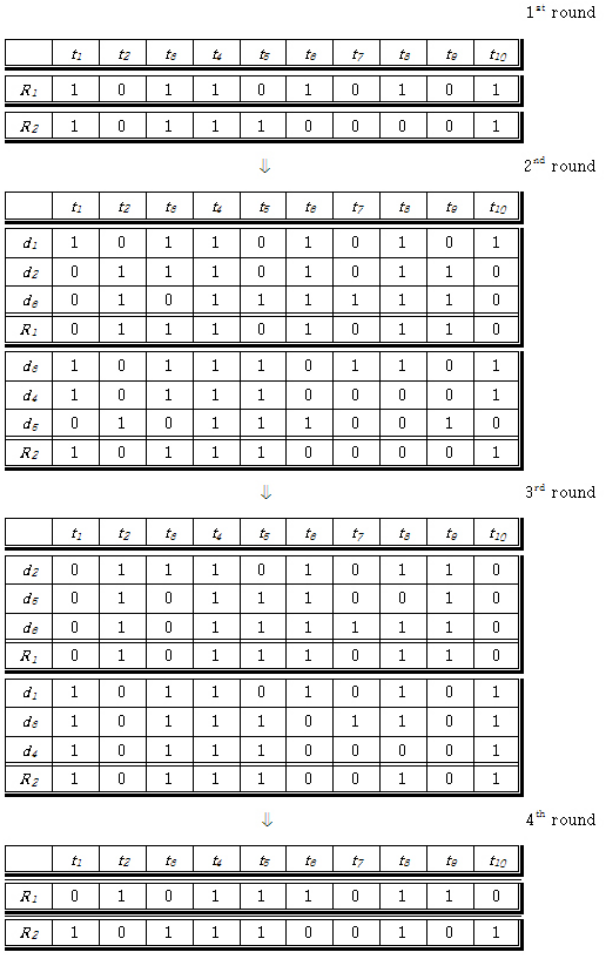
기존의 비트맵 클러스터링 기법은 사용자가 정해준 유사도 임계값에 따라 클러스터링이 수행되기 때문에 웹에서 실제로 사용되는 RSS나 ATOM과 같은 XML 기반의 피드 데이터에 적용할 경우 너무 많은 클러스터를 생성하여 검색 재현율이 낮아지는 문제점을 가지고 있었다. 또한 일정한 수준의 검색 품질을 보장하기 위해 어느 정도가 적절한 임계값인지 정하는 것은 반복적인 실험을 수행해야 한다. 본 논문에서 제안하는 k-비트맵 클러스터링 기법은 사용자가 원하는 k값을 입력하고, 전체 데이터를 유사도에 따라 k개의 클러스터를 생성한다. k-비트맵 클러스터링 알고리즘은 다음과 같다.

[알고리즘 2] k-비트맵 클러스터링

1. k개의 문서를 선택하여 k개의 클러스터에 할당한다.
2. k개의 클러스터에 대해 초기대표비트를 생성한다.
3. 모든 문서를 1개씩 불러들여 k개의 클러스터와 유사도를 비교하고 가장 유사도가 높은 클러스터에 추가한다.
4. 각 클러스터의 대표비트를 새로 생성한다.
5. 4의 대표비트를 기준으로 하여, 대표비트가 변하지 않을 때까지, 또는 적절한 임의의 값 이하로 변할 때까지 다시 3과 4를 수행한다.

(그림 3)의 원본 비트맵 데이터에 대해 k-비트맵 클러스터링 알고리즘을 수행하는 예(k=2, 대표비트 임계값=0.5)를 (그림 4)와 같이 단계별로 정리할 수 있다.

(그림 4)의 첫 번째 라운드에서는 d_1과 d_4의 값을 2개의 클러스터 C_1과 C_2에 초기대표비트로 할당하여 각각 R_1: 1011010101과 R_2: 1011100001의 대표비트를 가지게 되었다.



(그림 4) k-비트맵 클러스터링의 예

두 번째 라운드에서 d_1부터 d_6까지의 모든 문서를 한 개씩 불러들여 유사도를 계산하면 C_1에는 d_1, d_2, d_6, C_2에는 d_3, d_4, d_5 문서가 할당된다. 이렇게 생성된 클러스터의 대표비트를 계산하면 각각 R_1: 0111010110, R_2: 1011100001이다. 세 번째 라운드에서 이 대표비트를 기준으로 다시 클러스터링을 수행하면 C_1에는 d_2, d_5, d_6, C_2에는 d_1, d_3, d_4가 할당되고, 대표비트는 R_1: 0101110110, R_2: 1011100101로 변경된다. 네 번째 라운드를 수행했을 때의 대표비트가 세 번째 라운드의 대표비트와 동일하므로 반복을 종료한다.

3.2 2단계 인덱싱

k-비트맵 클러스터링은 사용자가 제시한 k개의 클러스터만 생성하므로 재현율의 향상에 기여하는 장점을 가진다. 하지만 클러스터링 기법은 일반적으로 전체 데이터를 검색하는 것보다는 재현율이 저하되는 문제점을 가진다. (그림 4)에서 t_5를 포함한 문서를 검색하기 위한 질의가 입력되면 먼저 대표비트를 확인하여 두 개의 클러스터를 모두 검색하여 결과 d_3, d_4, d_5, d_6을 반환한다. 이때는 재현율이 낮아지는 문제가 발생하지 않는다. 하지만 t_3을 검색한다면 대푯값에 t_3이 존재하는 클러스터 C_2의 데이터를 검색해서 d_1, d_3, d_4를 반환한다. 클러스터링을 수행하기 전에 비해 연산시간은 줄어들 수 있지만, 클러스터링 전에 1이었던 재현율은 클러스터링 후에 0.75로 낮아지는 문제점이 발생한다. 또한 t_7을 검색하면 d_3과 d_6이 결과에서 배제되고 아무런 결과도 얻을 수 없기 때문에 재현율은 0이 된다. 이 절에서는 재현율을 항상 1로 유지하기 위해 추가 역인덱스의 생성과정을 소개한다.

termID	*documentID*
t_3	d_2
t_6	d_1
t_7	d_3
t_7	d_6
...	...
t_i	d_j

(그림 5) 단어 역인덱스의 예

클러스터링을 수행한 후에 각 단어에 대해 역인덱스를 구성하는데, 이 인덱스는 대표비트에서 제외된 단어를 추가하여 생성한다. 역인덱스 생성 알고리즘은 다음과 같다.

[알고리즘 3] 역인덱스 생성

1. 모든 단어에 대해 Bitwise OR 연산을 수행한다.
2. 0 또는 1로 표현된 연산의 결과와 각 단어의 대표비트 값을 비교한다.
3. 두 값이 다를 경우 역인덱스에 단어 ID(termID)와 문서 ID(documentID)를 추가한다.
4. 모든 단어에 대해 1~3의 과정을 수행한 다음, 단어 ID를 기준으로 오름차순으로 정렬한다.

이렇게 생성한 추가 인덱스를 통해 재현율을 항상 1로 고정할 수 있다. (그림 4)의 예에서는 t_3, t_6, t_7의 세 단어가 그 대상이 된다. (그림 5)는 단어 역인덱스의 예를 보여준다. 단어 역인덱스에서는 n개의 *termID*가 정렬되어 있음을 가정할 때, 검색의 시간복잡도는 $O(\log n+s)$가 될 것이다. s는 반환되는 문서의 수이다.

4. 구현 및 성능평가

제안한 방법의 성능평가를 위해 웹 상에 존재하는 블로그의 RSS로부터 피드를 수집하고, 그 중에서 데이터 전처리 과정을 배제하기 위해 RSS피드에 언어 정보가 영어로 정의된 블로그만 선별하였다. 수집한 문서(article)의 수는 223,543개였고, 문서에서 추출한 단어의 수는 85,349개였다. 성능평가를 위한 프로그램은 Java 1.6.0_11 버전을 이용하여 구현하였고, Core2Quad 2.40CPU와 4GB RAM을 가진 PC환경에서 실험을 수행하였다. 데이터의 저장과 관리는 MySQL 5.0.67버전을 사용하였다. 본 논문의 성능평가에서 대표비트

〈표 1〉 유사도 임계값과 클러스터의 수

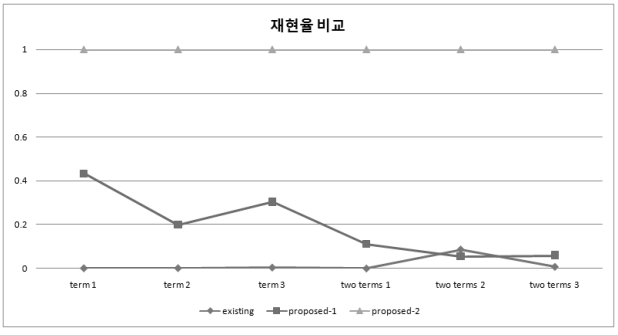
유사도 임계값	클러스터의 수
0.00001	1,794
0.0001	73,259
0.001	223,136
0.01	223,427
0.1	223,495

를 생성하기 위한 임계값은 0.0001로 설정하였다.

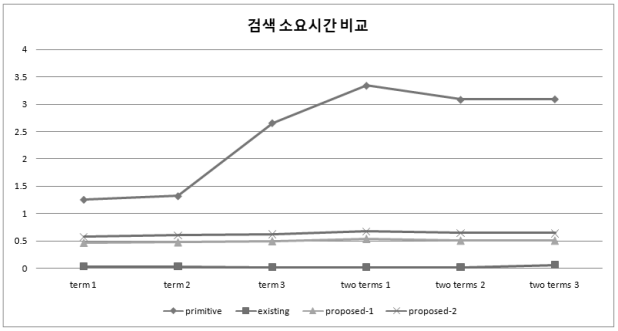
기존 방법을 실제 데이터에 적용하였을 때 유사도 임계값에 따라 얼마나 많은 클러스터가 생성되는지 확인하기 위해 첫 번째 성능평가에서는 기존의 비트맵 클러스터링 기법에 대해 유사도 임계값을 변형하며 생성되는 클러스터의 수를 측정해보았다. <표 1>은 기존의 비트맵 클러스터링 기법에서 유사도 임계값을 변화시키며 생성되는 클러스터의 수를 나타낸 것이다. 유사도 임계값의 범위가 0.1과 0.001 사이에서는 클러스터링이 거의 수행되지 않은 결과를 보였고, 0.0001에서 낮은 수준의 클러스터링이 수행됨을 보였다. 그리고 10만개 중 단 한 개의 단어만 공유해도 클러스터링이 될 수 있는 0.00001의 임계값에서도 1,794개의 클러스터가 생성되는 모습을 보여주었다. 제안하는 방법은 클러스터의 수를 임의로 설정할 수 있으므로 이 실험은 수행하지 않았다.

두 번째로 클러스터링의 품질을 평가하였는데, 대표적인 평가방법은 정확도(precision)와 재현율(recall), 그리고 F_1-measure가 있다. 이 실험에서 사용한 프로그램은 질의의 결과가 포함된 클러스터 전체를 반환하지 않고, 입력된 질의에 정확히 일치하는 것만 반환하도록 구현했기 때문에 정확도와 F_1-measure는 배제하고, 재현율만 측정을 하였다.

(그림 6)에서 실험 항목은 기존 비트맵 클러스터링에서 유사도 임계값 0.00001, 제안하는 k-비트맵 클러스터링, 그리고 여기에 역인덱스를 포함한 방법의 재현율을 비교하였다. 제안한 방법의 k값은 10으로 설정하여 10개의 클러스터를 생성하였다. 기존 방법(existing)은 모든 질의에 대해 0에 근접한 아주 낮은 재현율을 보였다. k-비트맵 클러스터링 기법(proposed-1)은 한 개의 단어를 검색할 때에는 항상 좋은 성능을 보였지만, 두 개의 단어를 검색할 때에는 질의에 따라 성능의 차이가 있었다. k-비트맵 클러스터링 후에 역



(그림 6) 재현율 비교 결과



(그림 7) 검색 소요시간 비교 결과

인덱스를 유지하는 방법(proposed-2)은 항상 1의 재현율을 보였다. 이는 클러스터의 대표비트를 생성할 때 배제된 항목들에 대해 추가된 역인덱스를 유지하면서 모든 관련문서를 반환하기 때문이다.

마지막 실험으로는 클러스터링 하지 않은 원시 데이터(primitive)와 앞의 각 실험 항목에 대해 검색 시간을 측정하였다. (그림 7)은 검색 소요시간을 비교하는 그래프이다. 클러스터링을 하지 않은 원시 데이터의 경우에는 단어의 수가 증가하면 검색 시간도 함께 증가하는 모습을 보였다. 클러스터링을 수행한 후에는 기존 방법이 가장 빠른 성능을 보였고, k-비트맵 클러스터링 기법과 역인덱스를 포함한 방법 모두 비교적 빠른 검색 속도를 보여주었다. 제안하는 기법의 검색 속도가 두 개의 인덱스를 검색하기 때문에 기존 기법보다는 느리지만 검색 재현율의 성능을 향상시킨다는 점에서 장점을 가진다.

5. 결론 및 향후 연구계획

본 논문에서는 XML 데이터의 내용기반 클러스터링 기법 중 기존에 좋은 성능을 보였던 비트맵 인덱스 클러스터링 기법의 장단점을 고찰하고, 임계값 설정이 어렵고 재현율이 낮아지는 문제점을 해결하기 위해 k-비트맵 클러스터링 기법을 제안하였다. 제안한 방법은 임의로 지정한 k개의 클러스터를 생성하기 때문에 유사도 임계값을 설정하지 않아도 클러스터링을 수행할 수 있다. 또한 k-비트맵 클러스터링을 수행함과 동시에 대표 비트에서 배제된 모든 단어를 역인덱스로 생성해 두 개의 인덱스를 유지하는 방법도 함께 제안하였다.

성능평가에서는 기존 방법으로 생성되는 클러스터의 수를 확인해보았고, 클러스터링 성능을 확인하기 위해 재현율과 검색 소요시간을 비교하였다. 기존 방법은 낮은 유사도 임계값에서도 매우 많은 수의 클러스터를 생성하였다. 재현율 확인에서는 제안한 방법이 단일 단어 검색에서 좋은 성능을 보였고, 두 개의 단어를 포함한 문서 검색에서는 기존의 방법과 k-비트맵 클러스터링 기법이 비슷한 수준의 성능을 보여주었다. k-비트맵 클러스터링 기법에 역인덱스를 추가하여 두 개의 인덱스를 유지할 때에는, 모든 검색 질의에 대해 관

련된 문서를 모두 반환하는 높은 재현율을 보였다. 검색 소요시간 비교에서는 기존 방법이 가장 빠른 성능을 보였지만, 제안한 두 개의 방법도 비교적 빠른 성능을 유지하였다.

본 논문에서 제안한 방법은 기존 방법에 비해 검색을 수행할 때 조금 더 많은 시간이 소요되지만, 생성되는 클러스터의 수를 임의로 조절할 수 있을 뿐만 아니라 2개의 인덱스를 함께 유지하며 높은 재현율을 보장하였다. 향후 연구로는 제안하는 k-비트맵 클러스터링 기법의 검색 소요시간을 줄이고, 검색 품질을 높일 수 있는 대표비트 임계값의 설정에 관한 연구가 필요하다.

참 고 문 헌

[1] M. Olson and U. Oqbuji, "The Python Web Service Developer: RSS for Python," http://www.ibm.com/developerworks/webservices/library/ws-pyth11.html, November, 2002.

[2] J. Yoon, V. Raghavan, V. Chakilam, and L. Kerschberg, "BitCube: A Three-Dimensional Bitmap Indexing for XML Documents," Journal of Intelligent Information System, Vol.17, pp.241-254, 2001.

[3] J. Yoon, V. Raghavan, and V. Chakilam, "BitCube: Clustering and Statistical Analysis for XML Documents," In Proc. of the 13th International Conference on Scientific and Statistical Database Management, Fairfax, Virginia, July, 2001.

[4] 민경섭, 김형주, "상이한 구조의 XML 문서들에서 경로 질의 처리를 위한 RDBMS 기반 역인덱스 기법", 정보과학회논문지, 제30권 제4호, pp.420-428, 2003.

[5] 서치영, 이상원, 김형주, "XML 문서에 대한 RDBMS에 기반을 둔 효율적인 역색인 기법", 정보과학회논문지, 제30권 제1호, pp.27-40, 2003.

[6] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, "Lore: A Database Management System for Semistructured Data," ACM SIGMOD Record, Vol.26, No.3, pp.54-66, 1997.

[7] C. Chung, J. Min, and K. Shim, "APEX: An Adaptive Path Index for XML Data," In Proc. of the International Conference on ACM SIGMOD, pp.121-132, Madison, Wisconsin, June, 2002.

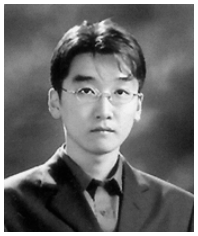
[8] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes, "Exploiting Local Similarity for Indexing Paths in Graph-Structured Data," In Proc. of the 18th IEEE International Conference on Data Engineering, pp.129-140, 2002.

[9] T. Dalamagas, T. Cheng, K. J. Winkel, and T. Sellis, "A Methodology for Clustering XML Documents by Structure," Information Systems, Vol.31, Issue 3, Elsevier Science Ltd., pp.187-228, May, 2006.

[10] T. Tran, R. Nayak, and P. Bruza, "Combining Structure and Content Similarities for XML Document Clustering," In Proc. of the 7th Australasian Data Mining Conference, pp.219-226, 2008.

[11] J. Lee and B. Hwang, "Path Bitmap Indexing for Retrieval of XML Documents," Lecture Notes in Computer Science, Vol.3885, Springer-Verlag, April, 2006.

[12] 김의찬, 황병연, "데이터 마이닝에서 비트 트랜잭션 클러스터링을 이용한 빈발항목 생성", 정보처리학회논문지D, 제13-D권, 제3호, pp.293-298, 2006.



이 범 석

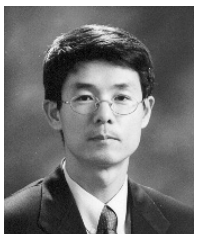
e-mail : leez79@gmail.com

2004년 가톨릭대학교 분자생물학과(이학사), 국사학과(문학사)

2006년 가톨릭대학교 컴퓨터공학과(공학석사)

2006년~현 재 가톨릭대학교 컴퓨터공학과 박사과정

관심분야 : XML, 웹2.0, 정보검색, 웹서비스, 생물정보



황 병 연

e-mail : byhwang@catholic.ac.kr

1986년 서울대학교 컴퓨터공학과(공학사)

1989년 한국과학기술원 전산학과(공학석사)

1994년 한국과학기술원 전산학과(공학박사)

1994년~현 재 가톨릭대학교 컴퓨터정보공학부 교수

1999년~2000년 University of Minnesota Visiting Scholar

2007년~2008년 California State University, Sacramento Visiting Scholar

관심분야 : XML 데이터베이스, 데이터마이닝, 지리정보시스템, 정보검색