데이터 마이닝에서 비트 트랜잭션 클러스터링을 이용한 빈발항목 생성

김 의 찬† · 황 병 연††

요 약

데이터베이스에는 많은 데이터들이 저장되어 있다. 무수히 많은 데이터들로부터 어떠한 정보를 얻기 위해서는 질의문을 사용하면 된다. 질의문을 통해 얻는 정보들은 기본적이고 단순한 정보들이다. 데이터 마이닝은 데이터베이스를 통해서 얻을 수 없는 정보를 얻게 해주는 기법이다. 데이터 마이닝 기법에는 여러 가지가 있지만 본 논문에서는 클러스터링과 연관규칙을 찾아내는 기법을 다룬다. 기존의 연관규칙 기법에서의 문제점을 보완하고 더 나은 규칙들을 찾아내기 위한 방법을 제시한다. 여기에 클러스터링 방법을 적용하게 되는데 기존의 거리기반이나 범주 기반 등의 클러스터링이 아닌 연관규칙에 적합한 클러스터링 기법을 제안하여 적용하게 된다. 각 클러스터의 연관규칙들을 찾게 되면 기존의 전체 데이터베이스에서 찾아진 연관규칙 뿐만 아니라 클러스터들의 특징이 될 규칙들도 찾을 수 있게 된다. 본 연구를 통해 대용량 데이터베이스의 많은 트랜잭션 접근을 줄이고 소집단의 연관성도 찾을 수 있다.

키워드 : 데이터 마이닝, 비트 트랜잭션, 클러스터링, 연관규칙

Frequent Itemset Creation using Bit Transaction Clustering in Data Mining

Eui-Chan Kim† · Byung-Yeon Hwang††

ABSTRACT

Many data are stored in database. For getting any information from many data, we use the query sentences. These information is basic and simple. Data mining method is various. In this paper, we manage clustering and association rules. We present a method for finding the better association rules, and we solve a problem of the existing association rules. We propose and apply a new clustering method to fit for association rules. It is not clustering of the existing distance basis or category basis. If we find association rules of each clusters, we can get not only existing rules found in all transaction but also rules that will be characteristics of clusters. Through this study, we can expect that we will reduce the number of many transaction access in large databases and find association of small group.

Key Words : Data Mining, Bit Transaction, Clustering, Association Rules

1. 서 론

우리가 사용하는 데이터베이스 내에는 많은 양의 데이터들이 들어 있으며 그러한 데이터들은 실세계의 데이터와 일치하는 데이터들이다. 데이터베이스로부터 정보를 얻기 위하여 우리는 질의를 하게 되고 그로부터 얻어내는 정보 또한 실세계의 데이터와 일치하는 것들이다. 질의 검색을 통해 기본적이고 일반적인 정보를 데이터베이스로부터 얻어내는 것과는 다르게 데이터베이스로부터 얻어내긴 하지만 기본적이고 일반적인 정보가 아닌, 단순한 질의 검색을 통해 얻어 낼 수 없는 함축적이고 암시적인 정보를 얻어내는 방법이 데이터 마이닝(Data Mining)이다. 데이터 마이닝 방법에는 여러 가지가 있다. 연관규칙(Association Rules), 분류(Classification), 일반화(Generalization), 클러스터링(Clustering) 등 다양하다[1]. 본 논문에서는 기존의 연관규칙 방법의 문제점을 보완하려 한다.

연관규칙 기법은 많이 연구되고 여러 분야에 적용되고 있는 데이터 마이닝 기법이다. 데이터베이스에 저장되어 있는 기본적인 데이터들을 바탕으로 기본 질의문을 통해서는 얻을 수 없는 규칙을 찾아내는 기법이다. 연관규칙을 생성할 때 사용하는 알고리즘으로는 매우 잘 알려진 Apriori 알고리즘이 있다[2]. 그러나 이 알고리즘은 무수히 많은 트랜잭션을 다루기에는 문제점이 있다. 왜냐하면 데이터베이스의 모든 트랜잭션을 반복해서 검사해야하기 때문에 트랜잭션이 많으면 많을수록 성능은 저하되는 것이다. 본 연구에서는 이러한

※ 본 연구는 2006년도 가톨릭대학교 전공특성화 사업비 지원으로 이루어졌음.
† 준 회 원 : 가톨릭대학교 컴퓨터공학과 박사과정
†† 종신회원 : 가톨릭대학교 컴퓨터정보공학부 교수
논문접수 : 2005년 9월 1일, 심사완료 : 2006년 2월 22일

점을 보완하기 위하여 클러스터링을 이용한다.

클러스터링 기법은 무수히 많이 연구되고 있으며 계속적으로 연구되고 있는 기법들 중 하나이다[3]. 또한 여러 다양한 분야에서 클러스터링 기법을 사용하고 있다. 클러스터링 기법은 주어진 객체들 중에서 유사한 객체들을 몇 개의 클러스터로 묶어 각 클러스터의 성격이나 특징을 파악하는 방법이다[4]. 본 논문에서 사용하는 방법은 기존의 클러스터링 방법인 거리 기반이나 밀도 기반으로 한 것이 아니라 아이템의 유무와 유사도, 아이템 소유가능도를 기반으로 한 클러스터링 방법을 제시하고 이것을 사용한다. 클러스터링을 이용함으로써 불필요한 트랜잭션 접근 수를 줄이고 기존의 방법으로 찾을 수 없었던 규칙도 찾아내어 각 클러스터들의 특징들을 찾아낼 수 있게 되는 것이다.

2장에서는 본 논문에서 다루게 되는 2가지 데이터 마이닝 기법에 대하여 살펴볼 것이고, 3장에서는 본 논문에서 제시하는 방법에 대하여 자세히 논의할 것이다. 4장에서는 실험결과를 바탕으로 제안했던 부분을 확인하며, 5장에서 결론 및 향후 연구 방향 계획에 대하여 기술할 것이다.

2. 관련 연구

2.1 클러스터링

클러스터링을 하는 데이터에는 크게 수치형 값과 범주형 값으로 나누어 볼 수 있다. 범주형 값의 경우는 클러스터를 형성하는 척도로 거리라는 개념을 적용하기가 쉽지 않다. 빈발 항목 개념을 이용한 트랜잭션 클러스터링 알고리즘은 범주형 값 중에서도 장바구니 데이터를 Apriori 알고리즘에서 사용한 빈발 항목 개념을 도입하여 빈발하는 아이템끼리만 클러스터링 한다[5-7].

범주형 값들을 비교하는 방법에는 [8]에서도 언급되었던 범주형 값을 수치형 값으로 바꾼 다음 좌표값으로 인식하여 두 값 사이의 거리를 구하는 방법이 있다. 좌표값으로 인식하기 위해서 클러스터링을 하기 전에 각 트랜잭션의 아이템 종류를 보고 해당하는 아이템이 있으면 1, 없으면 0으로 표현한다. 이렇게 변환된 좌표를 유클리디안 거리법을 이용하여 인접한 것끼리 클러스터링을 하는 것이다.

하지만 본 연구에서는 범주형 값들 사이의 거리를 구하는 것이 유사도를 구하는 데에 적합한 방법이 아니라고 본다. 왜냐하면 아이템들 간의 거리 값을 구하려 할 때 근사값이 나오기도 하는데 이는 아이템을 나타내는데 적합하지 않기 때문이다. 따라서 본 논문에서는 유사도 식과 소유가능도 식을 제시하여 해당 아이템의 유무에 따라 클러스터링 하는 방법을 제시한다.

2.2 연관규칙

연관규칙은 대용량 데이터베이스 내의 단위 트랜잭션에서 빈번하게 발생하는 사건의 유형을 발견하는 것이다[4]. 예를 들어, "전체 고객 중에 빵과 버터, 그리고 우유를 구매한 고객이 10% 이상이고, '빵과 버터'를 구매한 고객의 50%가 우유도 함께 구매한다." 이것이 하나의 발견된 사건의 유형, 즉 하나의 규칙이 된다. 여기서 10%는 연관규칙의 지지도(support)가 되고, 50%는 신뢰도(confidence)가 된다.

연관규칙을 찾는 전체 과정을 간단하게 살펴보면, 전체 데이터베이스에서 먼저 후보 아이템 항목 집합을 찾고, 이 후보 아이템 항목 집합에서 미리 제시된 최소 지지도 값을 넘는 빈발 항목 집합을 찾아낸다. 빈발 항목 집합을 찾을 때 전체 데이터베이스의 트랜잭션을 반복적으로 검색하면서 조인연산을 계속해서 사용하게 된다. 최종적으로 나오는 빈발 항목 아이템 집합에서 최소 신뢰도 값을 넘는 연관규칙을 찾아내게 되는 것이다. 여기서 지지도(S)란, 전체 사건 또는 거래 중에서 어떤 아이템 X와 아이템 Y를 동시에 포함하는 사건 또는 거래가 어느 정도 되는가 하는 것이다. 이것을 식으로 표현하면 다음과 같다.

$$S = \frac{|X \cap Y|}{N} \quad (\text{N은 전체 트랜잭션의 개수})$$

신뢰도(C)는 어떤 아이템 X를 포함하는 사건이나 거래 중에서 Y가 포함된 사건이나 거래가 어느 정도인가 하는 것이다. 이것을 식으로 표현하면 다음과 같다.

$$C = \frac{|X \cap Y|}{|X|}$$

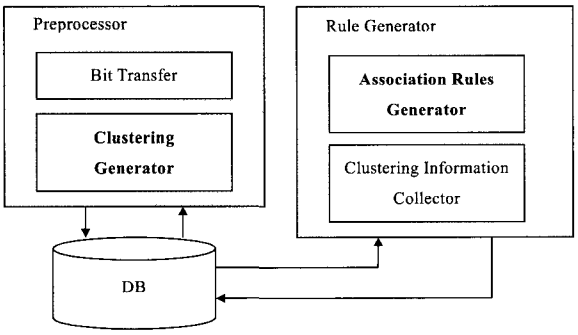
지지도를 통해 나온 빈발항목들에서 신뢰도를 통해 최종 연관규칙을 얻어내는 것이다. 대표적인 연관규칙 알고리즘으로는 [4]에서 제안한 Apriori 알고리즘이 있다. 이 알고리즘은 조인(join) 단계와 가지치기(prune) 단계로 나누어지는데 앞서 살펴본 지지도 계산에서 각 항목의 발생 빈도수를 세어 빈발 항목 집합을 찾아내게 된다. 이 과정에서의 문제점은 데이터베이스의 전체 트랜잭션을 반복적으로 검색해야하기 때문에 그만큼 수행속도는 느려지게 된다는 점이다. 본 논문에서는 이러한 문제점을 해결하기 위한 방법으로 먼저 클러스터링을 실시하고 클러스터링을 통해 나온 각 클러스터에서 연관규칙을 찾게 되면 전체 트랜잭션보다 클러스터 내에 있는 더 적은 수의 트랜잭션을 검색하기 때문에 기존의 방법보다 성능은 더 나아지게 될 것이다. 또한, 기존의 전체 데이터베이스를 검색해서 얻어내는 연관규칙뿐만 아니라 각 클러스터별로 기존의 방법으로는 찾을 수 없는 규칙 및 클러스터의 특징을 알 수 있는 연관규칙을 찾아낼 수 있다는 장점도 있다.

기존 연구되었던 방법들 중에는 후보 항목집합의 크기를 줄이는데 해시기반을 사용한 기법[9]과 주어진 데이터베이스에서 임의의 샘플을 취하여 빈발항목 집합을 탐색하는 방법[10], 후보집합 생성이 없는 빈발항목 집합 마이닝 방법[11] 등이 있다.

3. 클러스터링을 이용한 빈발항목 생성

본 논문에서 제시되는 내용은 아이템을 다루는데 적합한 클러스터링 방법의 제안과 제안한 클러스터링 방법을 이용하

여 기존의 연관규칙 생성 방법을 보완하는 것이다. 이 시스템의 기본 구조는 (그림 1)과 같다. 시스템은 전처리 작업 부분과 규칙 생성 부분으로 나누어지며, 전처리 작업 부분에는 비트로 변환시키는 변환기와 클러스터 생성기가 있고 규칙 생성 부분에는 클러스터 정보 수집기와 연관규칙 생성기가 있다.



(그림 1) 시스템 구조도

3.1 전처리 작업

먼저 전처리 작업으로 데이터베이스 내에 해당 트랜잭션에 있는 아이템들을 비트로 변환시켜 저장한다. 비트로 변환시킨다는 것은 예를 들어 각 트랜잭션을 고객이라고 가정하고 상품을 아이템이라 가정한다면, 고객이 상품을 샀을 경우 즉, 소유하게 되었을 경우 1로, 사지 않은 경우 즉, 소유하지 않은 경우를 0으로 변환시킨다는 것이다. 이렇게 미리 비트로 변환하게 된다면 클러스터링을 할 때보다 빠르게 처리할 수 있게 된다. 본 논문에서는 랜덤생성기를 이용하여 0과 1로 이루어진 비트 트랜잭션들을 생성해서 실험하였다. 비트 변환으로 데이터베이스에 트랜잭션 아이디와 비트 값을 갖는 항목들로 이루어진 테이블이 생성되게 된다. 이 테이블을 이용하여 전처리 작업의 두 번째인 클러스터링을 하게 된다. 클러스터링에 대한 내용은 3.2 절에서 좀 더 자세히 살펴보기로 한다.

3.2 비트 트랜잭션 클러스터링

클러스터링을 하기 위해서는 각 트랜잭션의 유사성을 확인할 필요가 있다. 본 논문에서는 기존의 거리 기반을 통한 유사성 확인이 아니라 [12, 13]에서 사용하였던 방법을 수정하였다. 각각의 트랜잭션의 유사도를 구할 수 있는 본 논문에서 제안하는 식을 이용하였다. 그 식은 다음과 같다.

$$t\_sim(t_i,t_j)=1-\frac{|xOR(t_i,t_j)|}{total\ of\ items}$$

여기서, t_i, t_j는 트랜잭션을 의미하며 |xOR(t_i, t_j)|는 t_i와 t_j 트랜잭션을 비교하는데 해당 아이템의 유, 무 즉 0과 1을 비교하여 같지 않은 것의 개수를 세는 부분이다. 그것을 전체 아이템의 개수로 나누어 1에서 뺀 값이 두 트랜잭션의 유사도 값이 된다.

클러스터링을 하기 위해서는 유사도 식을 통하여 각 트랜잭션들 간의 유사성을 구하고 클러스터를 생성하게 된다. 유사도를 비교하는 과정에서 클러스터와 하나의 트랜잭션을 비교하기 위해서는 그 클러스터를 대표할 수 있는 대푯값이 있어야 비교가 가능하다. 본 논문에서는 그 대푯값을 구하기 위하여 소유가능도를 계산하여야 한다. 여기서, 몇 가지 용어가 정의되는데 첫째로 소유가능도라 함은 각 트랜잭션 별로 해당하는 아이템을 소유할 가능성을 나타내는 것으로 정의하고, 소유가능 임계값은 소유가능도 값을 구하기 위해 필요한 사용자 정의 입력값이라고 정의할 수 있다(단, 소유가능 임계값 >= 0.5). 소유가능도를 구하기 위한 식은 다음과 같다.

$$pos(i_i)=\frac{|i_i|}{[i_i]}$$

여기서 i_i는 해당 아이템이 되며, $|i_i|$는 해당 아이템의 값이 1인 개수, $[i_i]$는 해당 아이템의 1과 0의 총 개수가 된다. 소유가능도 식을 통해 소유가능도 값을 구하고 난 다음 아래 정의를 통해 해당 아이템의 대푯값을 정하게 된다.

[정의 1] 소유가능 아이템이라고 하는 것은 소유가능도 (pos(i_i)) 값이 소유가능 임계값보다 크거나 같은 아이템이다.

pos(i_i) >= 소유가능 임계값
→ 소유가능 아이템(possessive item)

[정의 2] 약한 소유가능 아이템이라고 하는 것은 소유가능도 (pos(i_i)) 값이 1-소유가능 임계값보다 작거나 같은 아이템이다.

pos(i_i) <= 1-소유가능 임계값
→ 약한 소유가능 아이템(weaken possessive item)

[정의 3] 강한 소유가능 아이템이라고 하는 것은 소유가능도 (pos(i_i)) 값이 1-소유가능 임계값보다 크고 소유가능 임계값보다 작은 아이템이다.

1- 소유가능 임계값 < pos(i_i) < 소유가능 임계값
→ 강한 소유가능 아이템(strengthen possessive item)

[정의 4] 1-bit 아이템은 해당 아이템이 소유가능 아이템이거나 강한 소유가능 아이템으로 간주되었을 때의 아이템으로 값을 1로 표시하고, 0-bit 아이템은 약한 소유가능 아이템으로 간주되었을 때의 아이템으로 값을 0으로 표시한다.

위의 4가지 정의를 이용하여 각 클러스터의 대푯값을 구하게 된다. 4가지 정의를 적용한 예를 살펴보기로 한다[11]. <표 1>과 같은 데이터베이스가 있다고 가정하자. 아이템들의 유무를 비트로 변환시키면 <표 2>와 같이 변환시킬 수 있다.

<표 2>와 같이 비트로 변환시킨 데이터베이스를 가지고 클러스터링을 하게 된다. 먼저 각 트랜잭션의 유사도를 구한다. 여기서 유사도 임계값이 필요한데 유사도 임계값은 실험

〈표 1〉 예제 데이터베이스

TID	Item Set
T1	a, b, c, f
T2	a, b, d, e
T3	a, b, c, d, f, g
T4	a, b, d, f
T5	g, h, i, j
T6	d, e, g, h, j

〈표 2〉 비트로 변환한 데이터베이스

TID	a	b	c	d	e	f	g	h	i	j
T1	1	1	1	0	0	1	0	0	0	0
T2	1	1	0	1	1	0	0	0	0	0
T3	1	1	1	1	0	1	1	0	0	0
T4	1	1	0	1	0	1	0	0	0	0
T5	0	0	0	0	0	0	1	1	1	1
T6	0	0	0	1	1	0	1	1	0	1

〈표 3〉 첫 번째 클러스터의 대푯값

TID	a	b	c	d	e	f	g	h	i	j
T1	1	1	1	0	0	1	0	0	0	0
T2	1	1	0	1	1	0	0	0	0	0
C1	1	1	1	1	1	1	0	0	0	0

에서 언급하겠지만 입력되는 값으로 구현하였고 이 예에서는 0.6으로 가정한다. 트랜잭션 T1, T2의 유사도를 구하는 과정은 다음과 같다. |xOR(t_1, t_2)|의 값은 서로 다른 아이템 비트값의 수를 구하는 것이므로 <표 2>에서 살펴보면 4가 나온다. 그리고 아이템의 총 개수는 10이므로 1-4/10=0.6 이라는 유사도 값이 나오게 된다. 이는 유사도 임계값인 0.6보다 크거나 같은 값이므로 T1과 T2는 같은 클러스터 내에 있는 트랜잭션으로 간주하게 된다. 다음 트랜잭션을 비교하기 전에 클러스터내에 2개 이상의 트랜잭션이 들어 있으므로 대푯값을 구해야 한다. 즉, T1과 T2 트랜잭션이 있는 클러스터의 대푯값을 구해야 한다. 대푯값을 구하는 과정은 다음과 같다.

먼저 소유가능도 임계값을 입력받아야 한다. 이 예에서는 0.6으로 가정한다. 각 아이템별로 소유가능도를 구하여야 한다. <표 3>에 나와 있는 것을 보면 a라는 아이템은 T1, T2 모두 소유하고 있다. 소유가능도 식을 통해 살펴보면 $|i_a|$는 소유하고 있는 즉, 1의 개수가 되므로 2라는 값이 되며 $[i_a]$는 1과 0의 개수라고 했으므로 2가 된다. 계산하여 보면 2/2 = 1 이라는 소유가능도 값이 나오게 된다.

다음으로 살펴보아야 할 부분이 정의 부분이다. 소유가능도 값이 1이 나왔고 이것은 소유가능 임계값 0.6보다 크거나 같으므로 정의 1에 해당한다. 이는 소유가능 아이템에 해당하므로 정의 4에 의해 a의 대푯값이 1이 된다. 아이템 c의 경우에는 소유가능도 식을 통해 계산하면 1/2 = 0.5가 나오게 되는데 이는 정의 3에 의해서 강한 소유가능 아이템에 해당하며 정의 4에 의해서 c의 대푯값이 1이 된다. 이러한 방법으로 다른 아이템들을 모두 계산하게 되면 <표 3>과 같은 결과가 나오게 된다. T3 트랜잭션은 <표 3>에 나와 있는 C1과 유사도 값을 비교하여 C1 클러스터에 포함을 시킬 것인지 새로운 클러스터 C2로 만들 것인지 결정하게 된다.

(그림 2)는 이와 같은 과정을 나타낸 B-Trans(Bit Transaction) 클러스터링 알고리즘이다. 1~2행은 유사도 임계값(sth), 소유가능 임계값(pth)을 입력받는 부분이고, 3행은 하나의 트랜잭션이 다른 트랜잭션들과 유사성을 비교하였을 때 기존 클러스터에 포함이 되었는지, 새로운 클러스터로 생성되었는지를 판별하기 위한 변수이다. 4행은 첫 번째 트랜잭션을 첫 번째 클러스터로 포함을 시키는 부분이며, 5~9행은 두 번째 트랜잭션을 첫 번째 클러스터와 비교하여 입력받은 유사도 임계값보다 크거나 같으면 첫 번째 클러스터에 포함시키고, 그렇지 않으면 새로운 클러스터로 생성하는 것이다. 10~30까지의 행은 남은 트랜잭션의 개수만큼 반복하여 클러스터링 과정을 수행하는 부분이다. 11~14행은 판별 변수가 1인 경우 소유가능도 함수를 통해 클러스터의 대푯값을 구하는 부분으로 각 클러스터의 대푯값들은 13행에 있는 cluster_center_table에 저장되어 있고 이 테이블을 새로 업데이트 하게 된다. 15~19행은 클러스터의 개수만큼 반복되는 부분으로 다음 트랜잭션과 cluster_center_table에 저장되어 있

```
Input: transactions, sim__threshold, pos__threshold
Output: clusters

1:  sth = sim_threshold;
2:  pth = pos_threshold;
3:  flag = 0;
4:  c1 ⊃ transaction1;
5:  If(sim_function(next_transaction, c1_center) >= sth){
6:    c1 ⊃ next_transaction;
7:    flag = 1; }
8:  else
9:    new_cluster ⊃ next_transaction;
10: while(transactionID <= transaction_total){
11:  if(flag == 1){
12:   pos_function(pth);
13:   update cluster_center_table;
14:  }
15:  while(maxcluster){
16:   if(sim_function(next_transaction, cluster_center) >= sth)
17:     temp = sim_function(next_transaction, cluster_center);
18:     largest_sim = find_largest_sim(temp);
19:   }
20:   if(largest_sim != NULL){
21:     largest_sim_cluster ⊃ next_transaction;
22:     flag = 1;
23:   }
24:   else{
25:     new_cluster ⊃ next_transaction;
26:     update maxcluster;
27:     flag = 0;
28:   }
29:   next_transaction++;
30: }
```

(그림 2) B-Trans 클러스터링 알고리즘

는 각 클러스터들의 대푯값과 유사성을 비교하는 부분이다. 20~28행은 15~19행에서 수행된 결과 값에 따라서 유사도가 가장 큰 클러스터에 해당 트랜잭션을 포함시키는 부분이다. 만일 입력받은 유사도 임계값보다 크거나 같은 클러스터가 없다면 새로운 클러스터로 생성하게 된다.

클러스터링이 완료되면 클러스터 정보를 가지고 있는 테이블이 생성되게 되며 이 테이블은 (그림 1)의 시스템 구조도에 나와 있는 클러스터 정보 수집기에서 가져와 사용하게 된다. 가져오는 정보의 내용은 클러스터 아이디와 트랜잭션 아이디이다.

3.3 빈발항목 생성

연관규칙을 생성하는 알고리즘은 기존의 Apriori 알고리즘을 이용한다. 그러나 전체 트랜잭션을 검색하여 빈발항목을 생성하는 것이 아니라 원하는 클러스터 내에 있는 트랜잭션만을 검색하여 빈발항목을 생성하게 되는 것이다.

4. 실험 및 결과

본 연구에서는 앞서 살펴보았던 클러스터링 방법을 이용하여 클러스터를 생성해 내고 클러스터 아이디와 클러스터 내에 있는 트랜잭션 아이디 정보를 추출해 내어 해당 트랜잭션들의 빈발항목을 찾아내게 된다. 연관규칙을 생성하기 위해 최소 지지도와 최소 신뢰도 값이 필요하다. 실험에서는 최소 지지도만을 사용하여 빈발항목을 구하는 실험만을 하였다. 최소 신뢰도를 통한 연관규칙 생성부분은 전체 연관규칙 기법의 성능면에서 큰 비율을 차지하지 않고 최소 지지도를 통한 빈발항목을 생성하는 부분이 성능의 대부분을 좌우하기 때문에 빈발항목을 구하는 실험까지만 하였다. 실험은 랜덤 생성기를 통해 만든 데이터를 이용하여 실험하였다.

4.1 실험 환경 및 방법

CPU는 Pentium 4 2.4GHz, Ram 512M, 운영체제는 Windows 2000 Server, 데이터베이스는 MS SQL Server 2000을 사용하였으며, 프로그램 구현은 Microsoft Visual Studio .Net C#으로 하였다. 실험 방법은 랜덤 생성기를 통하여 트랜잭션 데이터를 생성한 후 본 논문에서 제시한 방법으로 클러스터링을 한 다음 최대 트랜잭션 수를 가지고 있는 클러스터의 빈발항목을 찾는 것으로 하였다. 앞서 언급하였듯이 빈발 항목을 통하여 연관규칙을 찾는 것은 전체 성능에 영향을 주지는 않는다. 빈발항목을 찾아내는 데까지가 성능을 좌우하는 부분이므로 본 실험에서는 빈발항목을 찾는 데까지로 실험의 초점을 맞추었다.

4.2 랜덤 데이터 생성 및 임계값 설정

랜덤 생성기는 [5]에 나와 있는 것을 참고하여 만들었다. 데이터를 랜덤하게 생성하였을 때 너무 무작위로 만들어지면 빈발항목이 하나도 생성되지 않는 경우가 있어서 실험에 대한 의미가 없어질 수 있으므로 조금이라도 빈발항목이 생성될 수 있도록 2가지 변수를 만들어 사용하였다. 첫 번째로 트랜잭션 내에 아이템을 어느 정도 소유하고 있도록 1의 비율 설정과 두 번째로 1-빈발항목의 아이템 수가 어느 정도 유지될 수 있도록 설정하였다. 트랜잭션의 개수는 10000개, 아이템의 개수는 50개로 설정하였고 클러스터링을 하기 위한 유사도 임계값과 소유가능도 임계값은 각각 0.6으로 고정하여 실험하였다.

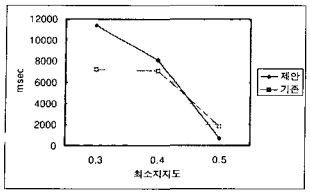
유사도 임계값이 낮은 경우 클러스터의 생성 개수가 줄어들며 그렇게 되면 클러스터를 하지 않은 상태와 비슷해지기 때문에 어느 정도 클러스터를 만들어내기 위해서 0.6으로 설정하였으며 소유가능도 임계값 또한 낮게 책정하면 대푯값 자체가 모두 1이 되어버리는 현상이 발생하는 것을 처음 실험을 통하여 확인할 수 있었기 때문에 유사도와 같이 0.6으로 고정하였다. 또한 유사도 임계값이 큰 경우에는 클러스터의 생성 개수가 너무 많아지고 클러스터 내에 트랜잭션들이 너무 적어지기 때문에 0.6으로 고정하게 되었다.

4.3 실험 결과 분석

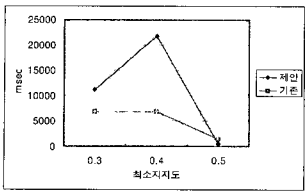
빈발 항목의 정확성 측정은 매 실험을 통하여 확인하였는데 기존 방법의 빈발항목 내에 들어있는 아이템들이 각 클러스터들을 통하여 나온 빈발항목에 포함되어있는 것을 확인할 수 있었다. 따라서 전체 데이터베이스 트랜잭션을 검색하는 Apriori 알고리즘을 통해 얻어낼 수 있는 빈발 항목 및 규칙들은 본 연구에서 제안한 방법으로 얻어내는 빈발 항목 및 규칙들에 포함되고 있으므로 본 연구에서 제안한 방법에 문제가 없고 클러스터 별로 특징이 될 수 있는 더 정밀한 규칙들을 찾아낼 수 있다는 것을 확인하였다.

성능 측면은 위에 언급하였던 변수들과 최소 지지도 값을 변경하면서 측정하였다. 최소 지지도 값을 0.6 이상으로 하였을 때 랜덤 데이터를 생성하는데 너무 오래 걸리며 1000회를 돌려도 실패하는 경우가 있었기에 0.5, 0.4, 0.3으로 나누어서 측정하였다. 랜덤 데이터에서 1의 비율과 1-빈발 항목 비율을 각각 0.2, 0.36으로 나눠서 실험하였다. 1-빈발 항목 비율 0.2는 50개 중에 10개, 0.36은 18개 이상의 아이템이 소유되고 있음을 의미한다.

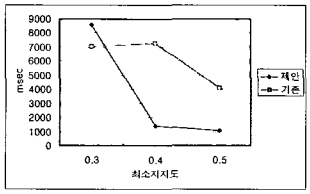
실험한 결과 각 최소지지도 및 1의 비율, 1-빈발 비율 변화에도 거의 비슷한 개수의 클러스터가 만들어졌다. 클러스터의 최대 트랜잭션 개수도 293~340 개 사이의 값으로 나타났다. 빈발 항목 생성 시간 비교는 클러스터내의 트랜잭션 개수가 가장 많은 것과 비교하였다. 빈발 항목 생성 시간을 살펴보면 본 연구에서 제안한 방법이 최소지지도가 높을수록 빈발항목 생성 시간이 단축됨을 알 수 있다. 그리고 최소 지지도가 0.3인 경우 기존의 방법보다 빈발항목 생성 시간이 더 걸리는 경우도 있는데 이는 빈발항목의 수가 많아지기 때문이다. 기존의 방법으로는 전체 트랜잭션을 검색하고 빈발항목을 생성할 경우 빈발 항목 집합이 적게 나와 그만큼 조인의 비용이 덜 들고, 제안한 방법으로는 비슷한 것끼리 묶인 클러스터이기 때문에 빈발 항목 집합이 많이 나와 조인의 비용이 더 많이 들게 되기 때문에 생성 시간이 오래 걸리게 된다. (그림 3)~(그림 6)은 1의 비율과 1-빈발 비율에 따라 빈발 항목 생성 시간을 그래프로 나타낸 것이다.



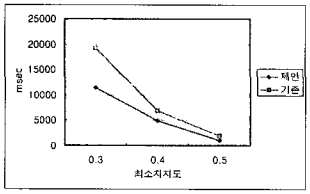
(그림 3) 비율(0.2, 0.2)



(그림 4) 비율(0.2, 0.36)



(그림 5) 비율(0.36, 0.2)



(그림 6) 비율(0.36, 0.36)

5. 결 론

본 논문에서는 아이템을 다루는 범주형 데이터의 경우에 적합한 클러스터링 방법을 제안하고 제안된 클러스터링 방법을 이용하여 빈발항목을 생성하였다. 그리고 기존의 연관규칙 방법보다 본 논문의 방법이 빈발항목을 생성하는 시간 측면과 정확한 규칙을 얻어내는 측면에서 더 나은 성능을 보인다는 것을 실험을 통하여 확인할 수 있었다.

클러스터링을 할 때는 본 논문에서 제시된 유사도 식, 소유가능도 식과 정의를 통해서 클러스터를 생성하게 되며 이 클러스터를 이용하여 빈발 항목을 찾아낸다. 데이터를 다룰 때에는 아이템의 소유 유무에 따라 0 또는 1의 비트 값으로 바꾸어 클러스터링의 효율성을 높였다. 실험 결과에서 알 수 있듯이 기존의 방법에서 얻어내는 빈발 항목을 포함하면서 더 많은 빈발 항목을 얻어낼 수 있었다. 그러나 본 연구에서의 한계점은 최소 지지도가 낮은 경우에 너무 많은 빈발 항목을 얻게 되어 기존 방법의 성능보다 오히려 저하되는 현상을 보이고 있다. 최소 지지도가 높은 경우에는 본 연구에서 제안한 방법이 더 나은 성능을 보인다.

향후 최소 지지도의 경우에도 좋은 성능을 보일 수 있도록 조인 연산 부분에 대한 보완과 빈발 항목의 최대 개수 조절, 다양한 트랜잭션의 개수를 이용한 성능 분석이 필요하며 랜덤 데이터가 아닌 실제 데이터를 비트 트랜잭션으로 변화시켜 본 연구에서 제안한 방법을 적용하여 실험하는 것도 필요하겠다.

참 고 문 헌

[1] M.S. Chen, J. Han, and P.S. Yu, "Data Mining: An Overview from a Database Perspective," IEEE Trans. on Knowledge and Data Engineering, Vol.8, No.6, pp.866-883, Dec., 1996.
[2] R. Agrawal and R. Srikant, "Fast Algorithm for Mining Association Rules in Large Databases," Proc. of Int. Conf. on Very Large Databases, pp.487-499, 1994.
[3] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," ACM Computing Surveys, Vol.31, No.3, pp.264-323, 1999.
[4] H. Wang, W. Wang, J. Yang, and P.S. Yu, "Clustering by Pattern Similarity in Large Data Sets," Proc. of ACM SIGMOD, pp.394-405, Jun., 2002.
[5] E.H. Han, G. Karypis, V. Kumar, and B. Mobasher, "Clustering Based On Association Rule Hypergraphs," Workshop on Research Issues on Data Mining and Knowledge Discovery, 1997.
[6] W.A. Kosters, E. Marchiori, and A.J. Oerlemans, "Mining Clusters with Association Rules," Proc. of Intelligent Data Analysis, pp.39-50, 1999.
[7] K. Wang, C. Xu, and B. Liu, "Clustering Transactions Using Large Items," Proc. of Int. Conf. on Information and Knowledge Management, pp.483-490, Nov., 1999.
[8] S. Guha, R. Rastogi, and K. Shim, "ROCK: a Robust Clustering Algorithm for Categorical Attributes," Proc. of Int. Conf. on Data Engineering, 1999.
[9] J.S. Park, M.S. Chen, and P.S. Yu, "An Effective hash-based Algorithm for Mining Association Rules," Proc. of ACM SIGMOD, pp.175-186, May, 1995.
[10] H. Toivonen, "Sampling Large Databases for Association Rules," Proc. of Int. Conf. on Very Large Databases, pp.134-145, Sep., 1996.
[11] J. Han, J. Pei and Y. Yin, "Mining Frequent Patterns without candidate generation," Proc. of ACM SIGMOD, pp.1-12, May, 2000.
[12] J. Yoon, V. Raghavan, and V. Chakilam, "BitCube: Clustering and Statistical Analysis for XML Documents," Proc. of Int. Conf. on Scientific and Statistical Database Management, Jul., 2001.
[13] 김의찬, 이재민, 황병연, "XML 문서 클러스터링을 이용한 개선된 연관규칙," 한국정보과학회 추계학술대회논문집, 제31권 제2호, pp.181-183, 2004.
[14] 김의찬, 황병연, "트랜잭션 클러스터링을 이용한 연관규칙 생성," 한국정보처리학회 춘계학술대회논문집, 제12권 제1호, pp.15-18, 2005.

김 의 찬
e-mail : eckim@catholic.ac.kr
1999년 가톨릭대학교 전산학과(이학사)
2001년 가톨릭대학교 컴퓨터공학과(공학석사)
2001년~현재 가톨릭대학교 컴퓨터공학과 박사과정
관심분야 : 데이터 마이닝, 공간 데이터베이스, 전자상거래, XML 데이터베이스

황 병 연
e-mail : byhwang@catholic.ac.kr
1986년 서울대학교 컴퓨터공학과(공학사)
1989년 한국과학기술원 전산학과(공학석사)
1994년 한국과학기술원 전산학과(공학박사)
1994년~현재 가톨릭대학교 컴퓨터정보공학부 교수
1999년~2000년 University of Minnesota Visiting Scholar
관심분야 : XML 데이터베이스, 데이터 마이닝, 지리정보시스템, 정보검색