대용량 데이터베이스에서 클러스터링을 이용한 빈발 패턴 생성

김의찬° 황병연
가톨릭대학교 컴퓨터공학과
{eckim°, byhwang}@catholic.ac.kr

Creation of Frequent Patterns using Clustering in Large Database

Euichan Kim° Byungyeon Hwang
Dept. of Computer Engineering, The Catholic University of Korea

요 약

데이터베이스에 저장되어 있는 데이터들을 통해서 의미있는 정보를 찾는 것이 데이터 마이닝이다. 많은 데이터 마이닝 기법들 중에 연관규칙을 다루는 연구가 많이 이루어지고 있다. 연관규칙 기법도 다양하게 연구되고 있는데 그 중 빈발 패턴 트리(FP-Tree)라는 방법을 이용하여 빈발 패턴을 찾아내는 연구가 활발히 진행되고 있다. 빈발 패턴 트리는 기존에 잘 알려져있는 연관규칙 생성 기법인 Apriori 기법보다 우수한 성능을 가지는 방법이다. 그러나 빈발 패턴 트리도 몇가지 문제점을 가지고 있다. 본 논문에서는 빈발 패턴 트리의 문제점 중 하나인 과도한 FP-Tree 생성을 줄이려 한다. 조건부 패턴 베이스를 통해 얻어지는 조건부 FP-Tree의 생성을 줄여 기존의 FP-Tree보다 더 나은 성능을 얻기 위해서 적절한 클리스터링을 이용하려 한다. 클러스터링 기법은 비트 트랜잭션을 이용한 클러스터링 방법을 이용한다.

1. 서 론

데이터 마이닝은 데이터베이스에 있는 간단하게 찾아볼 수 있는 데이터들을 통해서 함축적이고 의미 있는 정보를 찾는 것이다. 데이터 마이닝 기법에는 여러 가지 기법들이 있다[1]. 그 중 연관규칙이라는 기법은 지지도와 신뢰도라는 것을 이용하여 트랜잭션 데이터에 있는 아이템들 간의 어떠한 연관성을 찾아내는 것이다. 대표적인 알고리즘으로 Apriori 알고리즘[2]이 있고 아직도 변형된 형태의 유사 알고리즘이 많이 나오고 있다.

Apriori 알고리즘의 문제점은 전체 데이터베이스를 반복해서 검색하여야 한다는 것이다. 반복적인 검색을 통해 빈발 항목을 생성해 낼 수 있다. 또한 빈발 항목을 생성할 때 많은 조인으로 인한 성능 저하 문제도 발생하고 있다. 대용량 데이터베이스인 경우 너무 많은 트랜잭션을 가지게 되므로 이러한 문제점은 피해갈 수 없는 상황이다.

연관규칙과 관련된 또 하나의 기법이 빈발 패턴 트리(Frequent Pattern Tree: FP-Tree)이다[3]. FP-Growth라는 방법에서 사용된 이 트리는 기존의 Apriori 방법에서 찾았던 빈발 항목을 생성하지 않고 트리를 이용하여 빈발 패턴을 찾는 방법이다. 그러나 FP-Tree의 문제점은 대용량 데이터베이스일 경우 무수히 많은 FP-Tree 생성으로 인하여 주기억 장치를 기본으로 하는 FP-Tree 생성에 문제를 발생시킨다는 것이다. 따라서 본 논문에서는 대용량 데이터베이스의 트랜잭션 데이터를 적절한 클러스터링 기법을 이용하여 클러스터링을 하고 각 클러스터들을 통해 빈발 패턴을 생성함으로써 과도한 FP-Tree 생성 문제를 해결하려 한다. 클러스터링 방법은 기존에 제안하였던 비트를 이용한 트랜잭션 클러스터링 방법[4]을 이용할 것이다.

본 논문에서 제안하는 방법은 [4]에서 제시했던 방법보다도 더 나은 효과를 거둘 수 있다. 앞서 언급하였던 Apriori 알고리즘의 문제점을 해결하기 위하여 클러스터링을 이용해서 전체 데이터베이스 검색 문제를 해결하였는데 여기서의 문제점은 클러스터를 통해 얻는 후보 항목들이 기존 Apriori 방법을 통해 얻은 것보다 더 많이 나오기 때문에 최소 지지도가 낮은 경우 오히려 더 많은 시간이 걸리는 것이다. 그러므로 후보 항목을 생성하지 않는 FP-Tree 방법을 이용하게 되면 더 나은 성능을 보일 것이다.

2장에서는 관련 연구에 대해 기술하고, 3장에서는 본 논문에서 제시하는 방법에 대해 논의한다. 4장에서는 실험 예제를 통해 본 논문에서 제안한 방법을 확인하고, 5장에서 결론을 기술한다.

2. 관련 연구

기존의 Apriori 알고리즘 기법은 후보 항목 집합, 빈발 항목 집합을 생성하여 연관규칙을 찾아낸다. 그러나 상당한 크기의 후보 집합 생성이 필요할 수 있다. 이러한 후보 집합 생성을 하지 않고 빈발 항목 집합을 발견하기 위해 나온 방법이 분할-정보(divide-and-conquer) 기법을 사용하는 빈발 패턴 증가(Frequent Pattern Growth) 또는 간단히 FP-Growth라 하는 기법이다.

이 방법은 빈발 항목을 가지는 데이터베이스를 빈발 패턴 트리(Frequent Pattern Tree: FP-Tree)로 압축하여 사용한다.

FP-Tree 생성 시에 전체 데이터베이스를 2번 스캔하게 된다. 첫 번째 스캔을 통해 빈발 아이템 리스트를 찾

아낸다. 빈발 아이템 리스트는 주어진 최소 지지도 값보다 높은 아이템들을 모아둔 리스트이다. 이 리스트에 있는 빈발 아이템들의 지지도도 같이 저장하게 되는데 저장할 때는 지지도의 순서가 내림차순이 되도록 정렬을 시켜 저장한다.

빈발 아이템 리스트의 순서에 따라 두 번째 스캔을 통하여 트랜잭션을 검색하면서 재귀적 호출을 통해 FP-Tree를 생성하게 된다.

생성된 FP-Tree를 이용하여 마이닝을 하게 되는데 FP-Tree의 리프 노드부터 Bottom-up 방식으로 진행하게 된다. 이 때 생성되는 것이 조건부 패턴 베이스와 조건부 FP-Tree가 있다. 이 두 가지를 통해서 빈발 패턴을 생성하게 된다.

FP-Tree의 변형들도 많이 연구되고 있는데 데이터베이스를 두 번 스캔하는 FP-Tree와는 다르게 한 번 스캔으로 FP-Tree를 구성하여 업데이트 시에 유용한 방법을 제시한 [5, 6]이 있고, Bottom-up 방식이 아닌 Top-down 방식을 이용한 [7]이 있으며, 조건부 FP-Tree의 과도한 생성을 막기 위해 여러 개의 서브트리를 만들어 사용한 [8]과 최소지지도를 갖지 않고 Top-K 빈발 아이템을 찾는 [9]가 있다.

대용량 데이터베이스에서는 FP-Tree의 생성이 커지면서 그에 따른 조건부 패턴 베이스 및 조건부 FP-Tree가 무수히 많아지게 되므로 성능에 문제가 발생할 수 있기 때문에 본 논문에서는 이러한 문제의 해결책으로 클러스터링 기법을 이용하려는 것이다.

3. 클러스터링을 이용한 빈발 패턴 트리

3.1 비트 트랜잭션 클러스터링

본 논문에서 사용하는 클러스터링 방법은 [4]에서 제안하였던 클러스터링 방법을 사용한다. 각 트랜잭션에 있는 아이템들은 아이템의 유무에 따라서 0과 1의 비트형태로 변환하여 사용한다. 트랜잭션들의 유사도 값을 구하기 위하여 다음과 같은 유사도 식을 사용한다.

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

여기서, $t\_sim(t_i, t_j)$ 함수는 각 트랜잭션들 간의 유사도 값을 구하는 함수이며, t_i, t_j는 트랜잭션이 된다. 다음으로 xOR 연산을 이용하여 두 트랜잭션의 차이를 구한다. $|xOR(t_i, t_j)|$의 의미는 두 개의 트랜잭션이 서로 다른 아이템을 가지고 있는 개수가 몇 개가 되는지 나타내는 것으로 그 개수를 아이템의 총 개수로 나누게 된다. 다음으로 각 클러스터의 대표값을 구하기 위하여 사용되는 식은 다음과 같다.

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

여기서, $pos(i_i)$ 함수는 각 아이템들의 소유가능도를 구하는 함수이며, i_i는 하나의 아이템이 된다. $|i_i|$의 의미는 아이템을 소유하고 있는 즉, 1의 개수가 되고 $[i_i]$의 의미는 아이템을 소유하고 있거나, 소유하고 있지 않은 즉, 1과 0의 개수가 된다.

다음으로 위 $pos(i_i)$식을 통해 대표값을 구하는 방법을 설명한다. 클러스터링을 하기 위해서는 유사도 임계값과 소유가능 임계값이 필요하다. 대표값을 구할 때 소유 가능성 임계값을 사용하게 되는데 먼저 몇 가지 정의를 바탕으로 클러스터의 대표값을 찾아내게 된다.

정의 1. $pos(i_i)$ >= 소유가능 임계값 → 이 식을 만족한다면 이 아이템은 possessive item이라 한다.
정의 2. $pos(i_i)$ <= 1-소유가능 임계값 → 이 식을 만족한다면 이 아이템은 약한 소유 가능 아이템이라 한다.
정의 3. 1-소유가능 임계값 < $pos(i_i)$ < 소유가능 임계값 → 이 식을 만족한다면 이 아이템은 강한 소유 가능 아이템이라 한다.
정의 4. 해당 아이템이 possessive item이거나 강한 소유 가능 아이템인 경우에는 1로, 약한 소유 가능 아이템인 경우에는 0으로 간주한다.

3.2 빈발 패턴 트리

빈발 패턴 트리는 [3]에서 제안하였던 FP-Growth 방법을 이용하여 생성하게 된다. 3.1절에서 언급하였던 비트 트랜잭션 클러스터링을 이용하여 대용량 데이터베이스에 있는 전체 트랜잭션을 몇 개의 클러스터로 나누게 된다. 각각의 클러스터에는 트랜잭션이 중복되지 않게 들어가 있게 되며 클러스터의 생성 개수에 따라서 클러스터 안에 있는 트랜잭션의 개수는 현저하게 줄어들 것이다.

클러스터를 통해 빈발 패턴을 찾게 되면 기존의 전체 트랜잭션을 다루는 것보다 FP-Tree 생성도 줄어들 것이며 트리의 크기도 줄어들기 때문에 재귀적 호출도 그만큼 줄일 수 있다.

4. 실험 예제

표 1과 같은 데이터베이스가 있다고 가정하자. 표 2는 표 1의 내용을 아이템 유무에 따라 0 또는 1로 변환한 것이다.

표 1 예제 DB

	Items
T1	a, b, e
T2	b, d
T3	b, c
T4	a, b, d
T5	b, c
T6	a, b, c, e
T7	a, b, c

표 2 비트로 변환한 DB

	a	b	c	d	e
T1	1	1	0	0	1
T2	0	1	0	1	0
T3	0	1	1	0	0
T4	1	1	0	1	0
T5	0	1	1	0	0
T6	1	1	1	0	1
T7	1	1	1	0	0

유사도 임계값과 소유가능 임계값을 각각 0.5로 놓고 비트 트랜잭션 클러스터링을 이용하여 클러스터링을 하면 {T1, T4, T6}과 {T2, T3, T5, T7} 2개의 클러스터가 생성된다. 클러스터 생성과정을 간단히 살펴보면 다음과 같다.

T1과 T2의 유사도 값을 구하면 0.4이므로 유사도 임

계값보다 작다. 따라서 T1과 T2는 다른 클러스터로 들어가게 되고 T1과 T3의 유사도 값을 구하면 0.4, T2와 T3의 유사도 값을 구하면 0.6, 그러므로 T3는 T2와 같은 클러스터에 속하게 된다. 다음으로 두 번째 클러스터의 대표값을 구한다. 표 3과 같이 나오게 되는데 이는 앞서 언급하였던 정의 4가지를 이용하여 구할 수 있다.

표 3 C2 클러스터 대표값

	a	b	c	d	e
T2	0	1	0	1	0
T3	0	1	1	0	0
C2	0	1	1	1	0

나머지 트랜잭션들도 클러스터 대표값의 유사도를 구하여 해당 클러스터로 포함시키면 2개의 클러스터가 생성되는 것이다.

다음으로 클러스터의 빈발 패턴 트리를 만든다. 최소 지지도는 2라고 가정하자. 먼저 지지도 순서대로 내림차순으로 정렬하고 그것을 바탕으로 빈발 패턴 트리를 그려보면 각 클러스터(C1, C2)의 빈발 패턴 트리는 그림 1, 2와 같다.

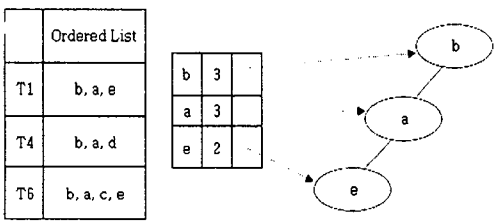

그림 1 C1에 대한 빈발 패턴 트리 생성

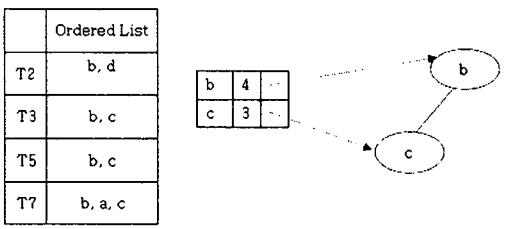

그림 2 C2에 대한 빈발 패턴 트리 생성

조건부 패턴 베이스와 조건부 FP-트리 마이닝을 통하여 얻어지는 빈발 항목들은 C1에서는 {(a, e), (b, e), (b, a, e), (b, a)}, C2에서는 {(b, c)}가 된다. 다만, 전체 트랜잭션을 통해 얻을 수 있는 (c, a)와 (d, b)는 최소 지지도를 더 낮게 주어야 찾을 수 있는 문제가 있다. 클러스터링을 하여 빈발 패턴을 찾을 때 성능 향상의 효과를 볼 수 있지만 약간의 빈발 패턴은 무시되어질 수 있다.

5. 결 론

본 논문에서 제안한 방법은 기존의 FP-Tree를 대용량 데이터베이스에서 사용할 경우 과도한 FP-Tree 생성으로 인한 성능저하 문제를 해결하기 위하여 비트 트랜잭션 클러스터링을 이용하였다. 클러스터링을 이용하면 기존의 FP-Tree 방법에서 보다 현저히 적은 조건부 FP-Tree가 생성되는 것을 실험 예제를 통해 확인할 수 있었다. 그러나 클러스터링을 하여 빈발 패턴을 얻는 경우 전체 트랜잭션을 통한 FP-Tree를 통해 얻어지는 빈발 패턴들 중 소량은 최소 지지도 값에 따라서 얻어지지 않는 경우도 발생할 수 있으므로 최적의 최소 지지도 값을 찾는 연구가 필요할 것이다.

* 참고 문헌

[1] M.S. Chen, J. Han, and P.S. Yu, "Data Mining: An Overview from a Database Perspective," IEEE Transactions 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," Proceeding of the 20th International Conference on Very Large Databases, Santiago de Chile, pp. 487-499, 1994.

[3] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proceeding of the 2000 ACM-SIGMOD International Conference Management of Data(SIGMOD '00), pp. 1-12, May 2000.

[4] 김의찬, 황병연, "트랜잭션 클러스터링을 이용한 연관 규칙 생성," 제 23회 한국정보처리학회 춘계학술대회 논문집, 제12권 제1호, pp. 15-18, 2005.

[5] H. Huang, X. Wu, and R. Relue, "Association Analysis with One Scan of Databases," Proceeding of the 2002 IEEE International Conference on Data Mining(ICDM '02), pp. 629-632, 2002.

[6] Y.L. Ruan, J.J. Zhang, Q.H. Li, and S.D. Yang, "A Fast Algorithm for Mining Frequent Patterns," Proceeding of the 3rd International Conference on Machine Learning and Cybernetics, Shanghai, pp. 1683-1686, Aug 2004.

[7] K. Wang, L. Tang, J. Han, and J. Liu, "Top Down FP-Growth for Association Rule Mining," Proceeding of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pp. 334-340, 2002.

[8] Y. Qiu, Y.J. Lan, and Q.S. Xie, "An Improved Algorithm of Mining From FP-Tree," Proceeding of the 3rd International Conference on Machine Learning and Cybernetics, Shanghai, pp. 1665-1670, Aug 2004.

[9] J. Wang, J. Han, Y. Lu and P. Tzvetkov, "TFP: An Efficient Algorithm for Top-K Frequent Closed Itemsets," IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 5, May 2005.