빈발 항목의 탐색 시간을 단축하기 위한 알고리즘

윤소영* · 윤성대**

An Algorithm for reducing the search time of Frequent Items

So Young Yun* · Sung-Dae Youn**

요 약

최근 정보시스템의 활용도가 높아짐에 따라, 많은 데이터를 이용하여 필요한 상품을 빠르게 추출하는 방법들에 대한 연구가 활발히 이루어지고 있다. 숨겨진 패턴을 탐색하는 연관 규칙 탐색 기법들이 많은 관심을 받고 있으며, Apriroi 알고리즘은 대표적인 기법이다. 그러나 Apriori 알고리즘은 반복적인 스캔으로 인한 탐색시간 증가 문제를 가지고 있다. 본 논문에서는 빈발항목의 탐색시간을 단축하기 위한 알고리즘을 제안한다. 제안한 알고리즘은 트랜잭션 데이터베이스를 이용하여 매트릭스를 생성하고 매트릭스에서 트랜잭션들의 평균 항목 개수와 정의한 최소 지지도를 사용하여 빈발 항목을 탐색한다. 트랜잭션의 평균 항목 개수는 트랜잭션의 수를 줄이는데 사용되고 최소 지지도는 항목을 줄이는데 사용된다. 제안한 알고리즘의 성능 평가는 기존 알고리즘과의 탐색시간 비교와 정확도 비교로 이루어진다. 실험 결과는 제안한 알고리즘이 기존의 Apriori와 매트릭스 알고리즘보다 최종 빈발 항목의 추출에서 빠르고 효율적으로 탐색이 이루어지는 것을 확인하였다.

ABSTRACT

With the increasing utility of the recent information system, the methods to pick up necessary products rapidly by using a lot of data has been studied. Association rule search methods to find hidden patterns has been drawing much attention, and the Apriori algorithm is a major method. However, the Apriori algorithm increases search time due to its repeated scans. This paper proposes an algorithm to reduce searching time of frequent items. The proposed algorithm creates matrix using transaction database and search for frequent items using the mean number of items of transactions at matrix and a defined minimum support. The mean number of items of transactions is used to reduce the number of transactions, and the minimum support to cut down on items. The performance of the proposed algorithm is assessed by the comparison of search time and precision with existing algorithms. The findings from this study indicated that the proposed algorithm has been searched more quickly and efficiently when extracting final frequent items, compared to existing Apriori and Matrix algorithm.

키워드

데이터 마이닝, Apriori 알고리즘, 빈발 항목, 매트릭스

Key word

Data mining, Apriori algorithm, Frequent Item, Matrix

* 정회원 : 부경대학교 (ysmallzero@pknu.ac.kr)
** 정회원 : 부경대학교 컴퓨터공학과 교수 (교신저자)

접수일자 : 2010. 09. 01
심사완료일자 : 2010. 09. 28

Ⅰ. 서 론

정보통신기술이 급진적으로 발달함에 따라, 이용 가능한 데이터의 양도 급격히 증가하고 있으며, 기업들이 급증한 데이터로부터 유용한 정보를 추출하여 의사결정에 얼마나 빠르게 활용하느냐가 기업의 경쟁력이 되고 있다. 따라서 대량의 데이터로부터 숨겨져 있는 데이터 간의 패턴을 찾아 유용한 정보를 추출하기 위한 기술인 데이터 마이닝에 대한 관심이 꾸준히 증가하고 있다. 데이터 마이닝의 결과로 생성되는 유용한 지식은 기업의 의사결정에 직접적인 영향을 미칠 수 있어 다양한 데이터 마이닝 기법에 대한 연구가 활발히 이루어지고 있으며, 데이터 마이닝 기술들 가운데에서도 연관 규칙 탐색에 대한 연구가 다양하게 이루어지고 있다.

연관 규칙 탐색은 트랜잭션에서 미리 정의된 최소지지도를 만족하는 빈발 항목집합을 찾아내고, 이들 빈발 항목집합들 간의 연관성 정도를 반영하는 연관 규칙을 찾아내는 것이다[1]. 다양한 연관 규칙 알고리즘 중 1994년 Agrawel 등에 의해 제안된 Apriori 알고리즘[2]이 널리 사용되고 있다.

그러나 Apriori 알고리즘을 포함한 기존의 연관 규칙 알고리즘들은 방대한 양의 트랜잭션 데이터베이스를 각 빈발 항목마다 계속적으로 스캔해야 하는 문제점을 지니고 있다. 이러한 연관규칙에 관한 기존 알고리즘의 문제점을 해결하고 좀 더 효율적인 빈발 항목 탐색을 위한 다양한 알고리즘들에 대한 연구가 계속되고 있다.

본 논문에서는 Apriori 알고리즘의 단점인 빈발 항목 탐색을 위한 지속적인 스캔으로 인해 발생하는 탐색시간 증가 문제를 해결하기 위해 트랜잭션 데이터베이스를 활용하는 알고리즘을 제안한다. 제안하는 알고리즘은 좀 더 효율적인 빈발 항목 탐색을 위하여 트랜잭션 데이터베이스를 이용하여 매트릭스를 생성한다. 매트릭스 생성 후 최소 지지도 이상인 항목과 각 트랜잭션이 포함한 항목의 개수의 합을 이용하여 추가적인 매트릭스 생성 없이 빈발 항목을 찾아내는 방법으로 스캔 횟수를 줄임과 동시에 빠르게 빈발 항목을 추출하여 탐색속도를 향상시킬 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구인 연관 규칙 탐색과 Apriori 알고리즘, 매트릭스 알고리즘 등에 대해 기술하고, 3장에서는 트랜잭션 데이터베이스 매트릭스를 활용하여 탐색시간을 줄이는 제안 알고리즘에 대해 살펴본다. 4장에서는 실제 데이터를 대상으로 제안하는 알고리즘의 성능 평가를 하고 마지막 5장에서는 결론을 기술한다.

Ⅱ. 관련 연구

2.1 Apriori

연관 규칙이 Agrawel 등에 의해 처음 소개된 후[3], 데이터베이스 검색횟수를 줄이거나 주기억 장치의 한계를 없애는 등의 발전된 알고리즘들이 발표되어왔다. 연관 규칙 탐색 알고리즘 중 널리 사용되는 것이 Apriori 알고리즘이다.

Apriori 알고리즘은 1994년 Agrawel 등에 의해 연관 규칙의 특성에 기반해 제안된 알고리즘이다. Apriori 알고리즘은 각 패스에서 빈발 항목집합들의 후보 항목집합을 구성하고 난 후에 각 후보 항목집합의 발생 빈도수를 계산하고, 사용자가 정의한 최소 지지도를 기초로 하여 빈발 항목집합들을 결정한다. Apriori는 많은 수의 후보 항목집합을 생성하는 AIS[2]와는 달리 Apriori-gen 이라는 새로운 후보 항목집합 생성전략을 개발하여 후보 항목집합의 수를 줄이는데 성공하였으며 그 이후 대부분의 알고리즘에서 사용하게 되었다[1].

그러나 Apriori 알고리즘은 두 가지의 주요한 문제점을 가지고 있다. 첫째, 반복적인 데이터베이스 스캔과 대규모 후보 항목집합에 대한 패턴 매칭 검사가 필요하다. 둘째, 상당한 크기의 후보 항목집합들의 생성이 필요하며, 이 후보 항목집합들을 저장하고 계산하기 위해 많은 시간이 소모된다[3]. 빈발 항목집합의 원소수가 증가할수록 데이터베이스의 스캔 횟수가 증가 하고, 계산 복잡도 또한 증가한다[4].

데이터베이스 스캔 횟수를 감소시켜 Apriori 알고리즘의 효율성을 향상시키고 후보 항목의 수를 줄이기 위해 DIC(Dynamic Itemset Counting)[5], DHP(Direct Hashing and Pruning)[6], 분할(Partition)[7], 샘플링(Sampling)[8] 알고리즘 등이 제안 되었다. 이 알고리즘들은 데이터베이스의 스캔과 많은 후보항목을 생성하는 점을 부분적으로 보완했지만 여전히 공통적으로

많은 수의후보 항목집합들을 생성해야 하고 각 후보 항목에 대해 빈발 항목여부를 판단해야 하는 단점을 가진다[3].

2.2 매트릭스 알고리즘

Apriori 알고리즘의 문제점을 해결하기 위해 최근에는 매트릭스를 활용한 알고리즘들이 제안되고 있다. Feng WANG 등에 의해 제안된 알고리즘[9]은 연관 규칙 탐색의 중요한 원리인 한 항목 집합이 빈발 항목집합이면 공집합을 제외한 그것의 모든 부분 집합도 빈발하다는 것에 초점을 둔 연구이다. WANG 등의 연구[9]에서는 트랜잭션 데이터베이스를 사용하여 행에 항목들을 표시하고 열에 트랜잭션아이디(TID)를 표기한 후 매트릭스를 생성한다.

```
Input: affair database "D", minsupport;
Output: all the frequent itemsets in the D;
(1) [Produce the initial matrix Mat]
    Mat←Create (D, Minsupport);
    If Mat=NULL
        End, no frequent itemsets
(2) [Find out the largest frequent itemsets]
  MaxK←Max (Mat, 0) [MaxK is the number of items in
                an affair which contain the most items]
  While MaxK>0, repeat the following code
    NewMat ← SimplifyMat (Mat,MaxK,Minsupport)
                        [simplify the Mat]
  If NewMat≠NULL
     L_MaxK←Find_Max(NewMat, MaxK)
                  [find out the MaxK-frequent itemsets]
        If L_MaxK≠NULL [if find out the MaxK-frequent
                        itemsets, then break]
            Break
  MaxK←Max (Mat, MaxK)
    If L_MaxK=NULL
        End, no frequent itemsets
(3) [Find out all the other frequent itemsets]
    L_1←Find_1 (Mat) [the collection of items in the Mat is
                  1-frequent itemsets]
    While K from 2 to MaxK, execute
    Mat←SimplifyMat(Mat, K, Minsupport)
                                [simplify the Mat]
        C_k ← GenC(Mat, L_MaxK , L_K-1) [find out the
                  candidates of frequent itemsets]
        L_K←GenL(Mat, C_K, Minsupport)
                          [find out frequent itemsets]
(4) Answer←L_1∪L_2∪…∪L_K∪L_MaxK∪L
                          [L is subset of L_MaxK]
(5) [END]
```

그림 1. Wang의 algorithm
Fig 1. Wang's algorithm

매트릭스는 빈발 항목집합의 후보 항목의 지지도를 계산하기 위해 사용하며 한번 생성되면 다시 데이터베이스를 스캔할 필요가 없다.

그림 1은 WANG이 제안한 알고리즘이다. 이 알고리즘은 유용한 가장 큰 빈발 항목 집합을 찾을 때까지는 시간이 걸리지만 전체적으로 후보 항목을 생성하는데 걸리는 시간은 Apriori 알고리즘 보다 짧다. 또한 빈발하게 데이터베이스를 스캔하지 않아 I/O 소비시간이 감소되고 시간복잡도가 Apriori 알고리즘 보다 감소된다. 그러나 최소지지도가 큰 경우 Apriori 알고리즘과 생성되는 후보 항목의 수에서 차이가 거의 없다. 또한 최소지지도가 크면 WANG이 제안한 방식에서는 가장 큰 빈발 항목 집합을 찾아 NewMat을 생성하기 위해 걸리는 시간이 길어진다. 따라서 최소지지도가 작을 경우에는 적합하지만 최소지지도가 크고 조밀한 데이터에서는 탐색시간을 단축시키는 효과를 크게 기대할 수 없다.

Ⅲ. 빈발 항목의 탐색 시간을 단축하기 위한 알고리즘(RSFI: Reducing Search time of Frequent Items)의 제안

본 논문에서 제안하는 빈발 항목 탐색 시간 단축을 위한 알고리즘(RSFI)은 Apriori 알고리즘의 반복적인 스캔으로 인한 탐색시간 증가 문제를 해결하고자 매트릭스를 활용한다. RSFI 알고리즘은 탐색시간을 감소시키기 위한 방법으로 각 TID에 포함된 항목들의 개수를 이용한다. TID에 포함된 항목들의 개수를 이용하여 매트릭스에서 TID의 평균 항목 개수를 구한 후 이 값을 임계값으로 지정하고 TID의 항목 개수가 임계값 미만이면 해당 TID를 매트릭스에서 삭제하여 계산되는 트랜잭션의 수를 줄이는 방법을 사용한다. 항목을 줄이기 위한 방법은 기존 알고리즘들처럼 최소 지지도를 사용한다.

RSFI 알고리즘에서 매트릭스 생성은 기존의 매트릭스 알고리즘과 같이 트랜잭션 데이터베이스에서 전체 트랜잭션을 읽어 들여 행에는 TID를 열에는 항목을 위치시키고 각 TID에서 항목의 선택유무에 따라 항목값들을 1과 0으로 채우는 방법을 사용한다. 처음 매트릭스가 생성되면 이후에 추가적인 매트릭스의 생성 없이 계산을 통해 최소지지도를 만족하는 빈발 항목

을 추출한다. RSFI 알고리즘에서 빈발 항목을 추출하는 과정은 그림 2와 같이 4단계로 이루어지고 시간복잡도는 $O(n^2)$이다.

```
Algorithm  RSFI
Input : Transaction DB, minsupport ;
Output : frequent itemsets ;

step 1 :  Make a Matrix            // Fig 3
step 2 :  Delete Transactions      // Fig 4
step 3 :  Delete Items             // Fig 4
step 4 :  Select frequent item     // Fig 5
```

그림 2. RSFI 알고리즘
Fig 2. RSFI Algorithm

3.1 RSFI 알고리즘 1단계

1단계는 트랜잭션 데이터베이스를 이용하여 매트릭스를 생성하는 단계이다. 트랜잭션 데이터베이스의 TID를 행에 위치시키고 항목들을 열에 위치시킨 후 각 TID에서 선택된 항목 값은 1로 선택되지 않은 항목 값은 0으로 표기한다. 트랜잭션 수와 항목 수를 줄이는 계산을 위해 트랜잭션별, 항목별 개수의 합을 행과 열의 마지막에 각각 구하여 최종적으로 매트릭스를 생성한다.

```
Algorithm MakeMatrix
Input : Transaction DB ;
Output : Matrix M;

Make a Matrix with Transaction DB ;
// filling the final column
  for each transaction TID in Matrix do
  {
    for (j=0; j<m; j++) do
    {
      // m is the number of  Items
      find f_column;
      // f_column is the sum of number of each TID
      write f_column in final column ;
    }
  }
```

그림 3. 매트릭스 생성
Fig 3. Make matrix

그림 3은 RSFI 알고리즘에서 매트릭스를 생성하기 위한 1단계를 나타낸 것이다. 표 1은 20개의 트랜잭션과 7개의 항목으로 구성된 트랜잭션 데이터베이스 예제이다.

표 1. 트랜잭션 데이터베이스 예제
Table 1. Transaction database example

TID	Items
T1	A G
T2	B C D
T3	D E G
T4	B D E
T5	E F G
T6	B D
T7	C E G
≀	
T14	B C D
T15	D E G
T16	C D
T17	A C E G
T18	B D E F
T19	E
T20	B D E F

표 2는 표 1의 트랜잭션 데이터베이스를 사용하여 RSFI 알고리즘의 1단계인 매트릭스를 생성하고 행과 열의 합을 구한 것이다.

표 2. 매트릭스 생성
Table 2. Make Matrix

	A	B	C	D	E	F	G	SUM
T1	1	0	0	0	0	0	1	2
T2	0	1	1	1	0	0	0	3
T3	0	0	0	1	1	0	1	3
T4	0	1	0	1	1	0	0	3
T5	0	0	0	0	1	1	1	3
T6	0	1	0	1	0	0	0	2
T7	0	0	1	0	1	0	1	3
≀								

T14	0	1	1	1	0	0	0	3
T15	0	0	0	1	1	0	1	3
T16	0	0	1	1	0	0	0	2
T17	1	0	1	0	1	0	1	4
T18	0	1	0	1	1	1	0	4
T19	0	0	0	0	1	0	0	1
T20	0	1	0	1	1	1	0	4
SUM	3	10	6	14	14	5	9	

3.2 RSFI 알고리즘 2, 3단계

2단계는 트랜잭션의 수를 줄이기 위한 단계이다. 트랜잭션을 줄이기 위해 우선 1단계에서 생성된 매트릭스의 마지막 열에 계산된 각 트랜잭션의 항목 개수들의 평균을 계산한다. 계산된 평균을 이용하여 매트릭스에서 항목 개수가 평균 미만인 트랜잭션을 제거한다. 트랜잭션을 제거 한 후에는 나머지 열들의 합을 다시 구한다.

3단계는 항목의 수를 줄이는 단계이며 항목의 수를 줄이기 위해 지정한 최소지지도를 사용한다. 매트릭스에서 마지막 행에 계산된 각 항목의 개수가 최소 지지도 미만인 항목들을 제거한다. 그림 4는 RSFI 알고리즘에서 평균값 미만 트랜잭션을 제거하기 위한 2단계와 최소지지도 미만 아이템을 제거하기 위한 3단계를 나타낸 것이다. 표 3은 1단계에서 생성된 예제 매트릭스에 RSFI 알고리즘 2, 3단계를 적용한 결과를 나타내는 것이다. 표 2의 전체 TID의 항목 평균값은 3이고 최소지지도는 4를 적용한다. 6개 TID의 항목 개수가 평균값 3 미만으로 삭제되고 1개의 항목이 최소지지도 4미만으로 삭제된다.

```
Algorithm DelTransaction and DelItem
Input : Matrix  minsupport;
Output : Matrix ;

Compare value of final column with MeanC;
 // MeanC is the mean values of final column
Delete transaction lower than MeanC in Matrix;
Compute f_row again;

Compare value of final row with minsupport;
 // minsupport is minimum support
Delete Item lower than minsupport in Matrix;
```

그림 4. 트랜잭션과 항목 제거
Fig 4. Delete transaction and item

표 3. 트랜잭션, 항목제거
Table 3. Delete transaction, items

	B	C	D	E	F	G	SUM
T2	1	1	1	0	0	0	3
T3	0	0	1	1	0	1	3
T4	1	0	1	1	0	0	3
T5	0	0	0	1	1	1	3
T7	0	1	0	1	0	1	3
T8	1	0	1	1	1	0	4
T10	1	0	1	1	0	0	3
T12	1	1	1	1	0	0	4
T13	1	0	1	1	1	1	5
T14	1	1	1	0	0	0	3
T15	0	0	1	1	0	1	3
T17	0	1	0	1	0	1	3
T18	1	0	1	1	1	0	4
T20	1	0	1	1	1	0	4
SUM	9	5	11	12	5	6	

3.3 RSFI 알고리즘 4단계

알고리즘의 4단계는 항목의 개수가 가장 큰 항목을 빈발 항목으로 지정하는 단계이다. 4단계에서는 매트릭스에서 마지막 행의 값이 가장 큰 항목을 최대 항목으로 선택하고 해당 항목을 포함하지 않는 행을 제거한다. 제거된 항목을 행 계산에서 제외하고 항목들의 합을 다시 구한다. 행의 합을 구한 결과로 남은 항목이 없을 경우에는 마지막 빈발 항목까지를 모두 최종 빈발 항목집합으로 추출한다. 남은 항목이 1개이고 마지막 행의 값이 최소지지도 이상이거나 남은 항목이 2개이고 마지막 행의 값이 최소지지도 이상이며 값이 같으면 이전까지 구한 빈발 항목에 남은 항목을 하나씩 추가해서 최종적으로 빈발 항목 집합을 추출한다. 남은 항목이 1개 이상이고 그 값들이 최소지지도 미만일 경우 마지막으로 추출한 빈발 항목과 남은 항목들을 그 전까지 추출한 항목 값에 하나씩 추가해서 최종적으로 빈발 항목 집합을 추출한다.

그림 5는 RSFI 알고리즘에서 빈발 항목을 선택하는 4단계를 나타낸 것이다.

```
Algorithm SelFrequentItem
Input : Matrix ;
Output : frequent itemsets ;

for MaxI ≠ null do {
  Select MaxI in final row;
   // MaxI is item with the maximum value in final row
  Delete TID that value of MaxI is 0;
  Exclude MaxI and compute f_row again;
    if rc_items=1 and vr_items>= minsupport then
    {
      // rc_items are the count of remaining items,  vr_items are the values of f_row
      frequent itemsets = {MaxI1 ∪ ... ∪ MaxIk ∪ r_item};
        //MaxIk are selecting items until the k-th, r_item is remaining item
    }
    else if rc_items =2 and vr_items >= minsupport and rc_item1 = rc_item2 then
    {
      frequent itemsets = {MaxI1 ∪ ... ∪ MaxIk ∪ r_items };
    }
    else if rc_items > 1 and vr_items < minsupport then
    {
      frequent itemsets = {{MaxI1 ∪ ... ∪ MaxIk-1 ∪ MaxIk}, {MaxI1 ∪ ... ∪ MaxIk-1
                          ∪ r_items}};
    }
    else
    {
      frequent itemsets = ∪ MaxIk;
    }
}
```

그림 5. 빈발 항목 선택
Fig 5. Select frequent item

표 4는 표 3의 마지막 행의 값이 가장 큰 항목 E를 최대 항목으로 선택하고 항목 E를 포함하지 않은 트랜잭션들을 제거한 결과를 나타낸 것이다.

표 4. 트랜잭션, 항목제거
Table 4. Delete transaction, items

	B	C	D	E	F	G	SUM
T3	0	0	1	1	0	1	3
T4	1	0	1	1	0	0	3
T5	0	0	0	1	1	1	3
T7	0	1	0	1	0	1	3
T8	1	0	1	1	1	0	4
T10	1	0	1	1	0	0	3
T12	1	1	1	1	0	0	4
T13	1	0	1	1	1	1	5
T15	0	0	1	1	0	1	3
T17	0	1	0	1	0	1	3
T18	1	0	1	1	1	0	4
T20	1	0	1	1	0	0	4
SUM	9	5	11	12	5	6	

표 5는 표 4에서 항목 E를 빈발항목으로 추출하고 행과 열을 다시 계산한 결과를 나타낸 것이다.

표 5. 빈발 항목 선택
Table 5. Select frequency item

	B	C	D	F	G	SUM
T3	0	0	1	0	1	3
T4	1	0	1	0	0	3
T5	0	0	0	1	1	3
T7	0	1	0	0	1	3
T8	1	0	1	1	0	4
T10	1	0	1	0	0	3
T12	1	1	1	0	0	4
T13	1	0	1	1	1	5
T15	0	0	1	0	1	3
T17	0	1	0	0	1	3
T18	1	0	1	1	0	4
T20	1	0	1	1	0	4
SUM	7	3	9	5	6	

위의 4단계에서 종료조건을 만족할 때까지 3단계와 4단계의 과정을 반복한다.

RSFI 알고리즘은 트랜잭션이 포함하고 있는 항목들의 합을 사용하여 트랜잭션을 제거하는 방법으로 트랜잭션 수를 줄인 후 빈발 항목을 추출하는 과정을 거친다. 따라서 모든 항목에 대해 매번 스캔을 하여 빈발 항목을 추출하는 Apriori 알고리즘보다 빈발 항목을 빠르게 탐색할 수 있다. 그러나 트랜잭션을 제거하는 과정을 거치므로 너무 적은 수의 트랜잭션을 사용할 경우에는 적합하지 않을 수 있다.

Ⅳ. 실험 및 평가

4.1 실험데이터

본 논문에서 RSFI 알고리즘의 탐색시간과 정확도를 실험하기 위해 Mushroom 데이터와 MovieLens 데이터를 사용하였다.

RSFI 알고리즘은 트랜잭션의 수를 줄여 탐색시간을 개선하는 것이므로 재현율은 비교의 의미가 없어 사용하지 않았다.

Mushroom 데이터는 UCI 데이터 셋과 PUSM으로부터 Roberto Bayardo 에 의해 정제된 데이터 셋으로 8124개의 트랜잭션과 120개의 항목들로 구성된다. Mushroom 데이터는 조밀한 데이터이므로 실험을 위해 무작위로 25개의 항목과 8000개의 트랜잭션을 추출하였다.

MovieLens 데이터 셋은 943명의 유저들이 1682개의 영화들에 대해 평가한 100,000개의 데이터로 구성된다 [10]. 본 논문에서는 테스트를 위해 데이터 셋을 80%의 training 데이터 셋과 20%의 test 데이터 셋으로 나누어 실험을 하였다. 모든 실험은 2GB의 메모리와 Intel Core 2 Quard 2.4GHz의 컴퓨터에서 실험을 하였다. 이 컴퓨터의 운영체제는 Windows XP이고, 사용된 언어는 Microsoft Visual C++ 6.0으로 구현하였다.

4.2 실험과 분석

실험은 3가지로 이루어졌다. 실험 1, 2에서는 Mushroom 데이터에서 추출한 8000개의 트랜잭션을 사용하여 빈발 항목 탐색시간을 비교하였다. 실험 1에서는 트랜잭션 수를 2000~8000까지 변화를 주어 RSFI 알고리즘과 Apriori 알고리즘, WANG의 알고리즘의 탐색시간을 비교하였다.

실험 2에서는 전체 트랜잭션에서 최소지지도를 0.5%~2.5%로 변화를 주어 RSFI 알고리즘과 Apriori 알고리즘, WANG의 알고리즘의 탐색시간을 비교하였다. 실험 1, 2의 탐색시간은 빈발 항목이 추출되는 동안의 모든 시간을 포함한다. 그 결과는 다음과 같다.

표 6과 그림 6은 트랜잭션 수의 변화에 따른 탐색시간을 실험한 결과를 나타낸다. 최소 지지도는 트랜잭션 별로 0.5%로 하였다.

표 6. 트랜잭션 수의 변화에 따른 탐색시간 비교
Table 6. search time comparison under the different number of transaction

알고리즘 / 트랜잭션	RSFI 알고리즘	WANG의 알고리즘	Apriori 알고리즘
2000	0.844	0.875	4.573
4000	3.292	3.412	11.427
6000	7.384	7.610	22.125
8000	13.117	15.531	36.181

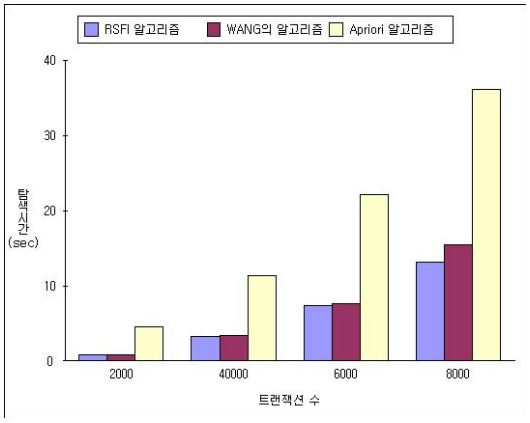

그림 6. 트랜잭션 수의 변화에 따른 탐색시간 비교
Fig 6. search time comparison under the different number of transaction

RSFI 알고리즘이 Apriori 알고리즘, WANG의 알고리즘 보다 탐색시간이 단축되는 것을 볼 수 있다. RSFI 알고리즘은 트랜잭션 별 항목의 평균 값 미만을 삭제한 나머지 트랜잭션과 최소지지도를 사용하여 빈발 항목을 추출하기 때문에 트랜잭션이 작을수록 탐색시간이 비교 알고리즘(Apriori)과 차이가 많이 난다. 트랜잭션이 2000일 때는 5배 정도 빠르고 8000일 때는 2.8배 정도 빠르다. 최소 지지도를 트랜잭션별로 0.5%로 작게 지정하여 비교하였으므로 트랜잭션이 적을 때는 WANG의 알고리즘 보다는 탐색시간이 짧지만 Apriori 알고리즘만큼의 차이는 보이지 않는다.

표 7. 최소지지도 변화에 따른 탐색시간 비교
Table 7. search time comparison under the different value of minsupport

알고리즘 / 최소지지도	RSFI 알고리즘	WANG의 알고리즘	Apriori 알고리즘
0.5%	13.116	13.569	39.713
1.0%	13.113	13.657	31.750
1.5%	13.100	13.728	30.138
2.0%	13.078	13.469	26.887
2.5%	13.066	14.637	25.759

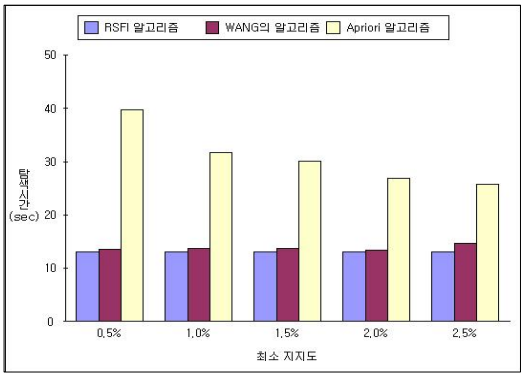

그림 7. 최소지지도 변화에 따른 탐색시간 비교
Fig 7. search time comparison under the different value of minsupport

표 7과 그림 7은 최소지지도의 변화에 따른 탐색시간을 실험한 결과를 나타낸다. 최소지지도는 0.5%부터 2.5%까지 0.5%씩 증가시켜 비교하였다. RSFI 알고리즘이 비교 알고리즘들 보다 지지도 변화에 따른 탐색시간도 더 빠르다. RSFI 알고리즘은 최소지지도의 변화에 크게 영향을 받지 않지만 Apriori 알고리즘은 최소지지도의 변화에 따라 탐색시간이 많은 영향을 받는 것을 볼 수 있다. WANG의 알고리즘은 K-최대 항목이 트랜잭션이 가진 최저 항목 수 이상일 경우에는 표 7과 같이 최소 지지도에 큰 영향을 받지 않지만 조밀한 데이터에서 최소지지도가 커져 K-최대 항목이 트랜잭션이 가진 최저 항목 수 보다 작을 경우 결과를 산출하지 못하는 경우가 발생한다.

정확성은 선택된 항목들이 적합할 확률로, 선택된 항목들의 수에 대한 적합한 항목들의 수의 비율로 정의된다. 다음의 식 (1)은 정확도를 구하는 식이다.

$$P = \frac{s}{s+ns} \times 100\% \qquad (1)$$

식 (1)에서 s는 추출된 빈발 항목 중 적합한 항목의 수이고 ns는 추출된 빈발 항목 중 적합하지 않은 항목의 수를 의미한다.

실험 3에서는 MovieLens 데이터 셋을 80%의 training 데이터 셋과 20%의 test 데이터 셋으로 나누어 사용하였다. training 데이터 셋을 사용하여 빈발 항목을 추출한 후 test 데이터 셋에서도 유효한 값인지를 판단하여 RSFI 알고리즘과 Apriori 알고리즘에서 추천의 정확성을 비교하였다. WANG의 알고리즘은 Apriori 알고리즘과 빈발 항목 탐색시간에서 많은 차이를 보이지만 추출결과가 Apriori 알고리즘과 같으므로 정확도의 결과는 Apriori 알고리즘과 같다.

표 8은 RSFI 알고리즘과 Apriori 알고리즘, WANG의 알고리즘의 정확도를 비교한 결과이다. 실험에서는 최소지지도를 다르게 하여 5회에 걸쳐 빈발 항목을 추출하여 그 결과를 비교하였다. 그 결과 RSFI 알고리즘은 평균 88.1%의 정확도를 나타냈고 WANG의 알고리즘과 Apriori 알고리즘은 83.2%의 정확도를 나타냈다. RSFI 알고리즘이 WANG의 알고리즘, Apriori 알고리즘보다 4.9% 정확도가 높았다.

표 8. 정확도 비교
Table 8. Precision Comparison

실험 \ 알고리즘	RSFI 알고리즘	WANG의 알고리즘	Apriori 알고리즘
TEST1	87.5%	82.61%	82.61%
TEST2	87.5%	81.97%	81.97%
TEST3	89.8%	82.05%	82.05%
TEST4	85.71%	82.97%	82.97%
TEST5	90.14%	86.5%	86.5%

V. 결 론

인터넷의 확산과 정보시스템의 발달로 인해 수집되는 데이터의 양이 급격히 증가하고 있다. 수집된 데이터로부터 필요한 정보를 추출하기 위해 대량의 데이터로부터 패턴을 찾아 유용한 정보를 추출하는 데이터 마이닝 기술에 대한 연구 또한 활발히 이루어지고 있다. 패턴을 탐색하는 대표적인 데이터 마이닝 기술은 연관 규칙 탐색이다. 연관 규칙 탐색을 위한 알고리즘 중에서도 Apriori 알고리즘이 많이 사용되고 있지만 Apriori 알고리즘은 방대한 양의 트랜잭션 데이터베이스를 각 빈발 항목마다 계속적으로 스캔해야 하기 때문에 시간적으로나 비용적으로 문제점을 가지고 있다. Apriori 알고리즘의 단점을 보완하기 위한 기존의 매트릭스 알고리즘도 트랜잭션 데이터베이스의 조밀성이나 최소 지지도의 크기에 많은 영향을 받는 문제점을 나타냈다.

본 논문에서는 Apriori 알고리즘의 단점인 반복적 스캔으로 인한 탐색시간 급증의 문제를 해결하기 위해 매트릭스를 활용하여 스캔 횟수를 줄임과 동시에 빠르게 빈발 항목을 추출함으로써 탐색속도를 향상시키는 알고리즘을 제안하였다.

제안하는 RSFI 알고리즘은 좀 더 효율적인 빈발 항목 탐색을 위하여 트랜잭션 데이터베이스를 이용하여 매트릭스를 생성한다. 이 때 마지막 행에는 각 항목들의 개수를 합으로 추가하고, 마지막 열에는 각 트랜잭션이 포함한 항목들의 개수를 합으로 추가한다. 생성된 매트릭스와 행과 열의 합을 이용하여 트랜잭션의 수를 줄이고 최소지지도 미만인 항목을 삭제하는 방법으로 추가적인 매트릭스 생성 없이 빈발 항목을 찾아낸다.

실험 평가를 통해 RSFI 알고리즘은 탐색시간에서는 Apriori 알고리즘보다 2배 이상 빠른 것을 알 수 있었다. WANG의 알고리즘과는 탐색시간에서 큰 차이가 없었지만 WANG의 알고리즘은 최소 지지도에 영향을 받아 최소 지지도가 작을 때에만 그 효과를 볼 수 있고 최소지지도가 클 경우에는 원하는 결과를 산출할 수 없는 경우가 발생한다. 정확도 비교에서도 RSFI 알고리즘이 Apriori 알고리즘, WANG의 알고리즘 보다 4.9% 높았다. 따라서 제안하는 RSFI 알고리즘은 빈발 항목들을 빠르게 추출함과 동시에 정확도가 높아 고객들에게 상품을 추천하는데 유용하다는 것을 알 수 있었다.

제안하는 RSFI 알고리즘은 트랜잭션을 제거하는 과정을 거치므로 너무 적은 수의 트랜잭션을 사용할 경우에는 적합하지 않을 수 있지만 실 데이터의 트랜잭션이 수십 개만 사용되지는 않으므로 실 데이터에서 사용할 경우에 문제가 되지 않는다.

향후 연구과제는 제안하는 알고리즘은 탐색시간을 줄이기 위해 트랜잭션을 제거하는 방법을 사용하였는데 이 때 사용하는 제거 조건에 대해 좀 더 다양한 연구가 필요할 것이다. 또한 더 다양한 실 데이터와 조건으로 실험하여 제안하는 알고리즘을 보강하고 정확도와 탐색시간을 향상시키는 방법을 연구하는 것이다.

참고문헌

[1] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules", *In Proceeding of the 20th VLDB Conference,* pp. 487-499, Santiago, Chile,1994.

[2] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large database", *Proceedings of the ACM SIGMOND Conference on Management of data,* pp. 207-216, Washington. DC, 1993.

[3] Jian Pei, *Pattern-Growth methods for frequent pattern mining,* the degree of doctor of philosophy, Simon Fraser University, pp. 9-12, 2002.

[4] Mehmed Kantardzic, *Data Mining : Concepts, Models, Methods and Algorithms,* Wiley-IEEE Press, 2002

[5] Sergey Brin, Rajeev Motwani, Jeffrey D. Ulman, and Shalom Tsur., "Dynamic Itemset Counting and Implication Rules for Market Basket Data," *In Proc. of ACM SIGMOD Conference on Management of Data (SIGMOD'97),* pp. 255-264,1997.

[6] Jung Soo Park, Ming-Syan Chen, and Philip S.Yu., "An effective hash-based algorithm for mining association rules," *In Proc. of ACM SIGMOD Conference on Management of Data(SIGMOD'95),* pp. 175-186, San Jose, California, May 1995.

[7] Ashok Savasere, Edward Omiecinski, and Shamkant Navathe, "An effective algorithm for mining association rules in large databases," *In Proc. of the 21st International Conference on Very Large Data Bases (VLDB'95),* pp. 432 -444, Zurich, Swizerland, 1995.

[8] Hannu Toivonen, "Sampling Large Data-base for Asso -ciation rules," *In Proc. of the 22nd International Conference on Very Large Data Bases (VLDB'96),* Mumbai(Bombay), India, 1996.

[9] Feng WANG, Yong-hua LI, "Am Improved Apriori Algorithm based on the matrix", *FBIE '08. International Seminar,* pp. 152-155, Dec. 2008.

[10] M.G. Vozalis, K.G. Margaritis, *Using SVD and demographic data for the enhancement of generalized Collaborative Filtering,* Information Sciences 177, pp.3017-3037, 2007.

저자소개

윤소영(So Young Yun)

2007. 2 부경대학교 경영대학원
경영학석사
2009. 8 부경대학교 대학원
공학박사 수료

※ 관심분야 : 전자상거래, 데이터마이닝, 추천시스템, M-commerce 등

윤성대(Sung-Dae Youn)

1980. 2 경북대학교 컴퓨터공학과
공학사
1984. 2 영남대학교 대학원
전자계산학과 공학석사
1997. 2 부산대학교 대학원 전자계산학과 이학박사
1981～1986 경남정보대학 전산과 조교수
1991～1992 MIT 방문교수
1989～현재 부경대학교 컴퓨터공학과 교수
※관심분야 : 병렬처리, 멀티캐스팅통신, 데이터 마이닝 등