LSI 유사도를 이용한 효율적인 빈발항목 탐색 알고리즘

고윤희† · 김현철†† · 이원규††

요 약

본 논문에서는 frequent itemset을 빠르게 발견해내기 위한 효율적인 vertical 마이닝 알고리즘을 제안한다. 본 알고리즘은 frequent itemset을 구하기 위해 아이템들을 Least Support Itemset(LSI)과의 유사도에 의해 올림차순으로 정렬하여 탐색 트리를 구축하여 보다 빠르고 효율적으로 frequent itemset을 찾아낸다. 또한, 트리를 탐색 시, 2가지의 휴리스틱 방법을 사용하여 탐색의 초기에 많은 후보 아이템들을 탐색 트리로부터 제거함으로써 탐색 공간을 크게 줄인다. 본 논문에서 제안하는 알고리즘은 이전의 알고리즘들과 비교해, long pattern을 가지는 데이터 베이스에서 보다 빠르게 frequent itemset을 발견해 냄을 실험을 통해 발견하였다.

Frequent Itemset Search Using LSI Similarity

Younhee Ko† · Hyeoncheol Kim†† · Wongyu Lee††

ABSTRACT

We introduce a efficient vertical mining algorithm that reduces searching complexity for frequent k-itemsets significantly. This method includes sorting items by their LSI(Least Support Itemsets) similarity and then searching frequent itemsets in tree-based manner. The search tree structure provides several useful heuristics and therefore, reduces search space significantly at early stages. Experimental results on various data sets shows that the proposed algorithm improves searching performance compared to other algorithms, especially for a database having long pattern.

1. 서 론

수많은 양의 데이터 속에서 의미 있는 패턴을 발견하거나 특정 아이템들간의 상관관계를 발견해 내는 연관 규칙 마이닝을 위한 많은 연구가 진행되어 왔다. 이러한 연관규칙 마이닝은 일반적으로 지지도를 만족시키는 frequent itemsets을 찾아내는 과정과 이를 이용해 연관 규칙을 생성해 내는 2개의 단계로 나누어진다. 하지만 실제로 frequent itemset을 찾아내기 위한 과정이 전체 마이닝에 필요한 시간의 대부분을 차지함에 따라 효율적으로 frequent itemset을 찾아내기 위한 많은 알고리즘이 연구되어 왔다. 이러한 알고리즘의 가장 대표적인 것이 Apriori 알고리즘이다. 이 알고리즘은 다음과 같은 성질을 이용하여 빈발항목 탐색의 효율성을 높였다: "Frequent itemsets의 모든 부분집합 또한 반드시 frequent itemset이 된다."[14,15]. 그러나 이러한 Apriori 알고리즘은 각 패스에서 생성된 후보항목집합(candidate itemset)들 중에서 frequent itemset을 찾아내기 위해 매번 계속적으로 방대한 양의 트랜잭션 데이터베이스를 스캔해야 하는 문제점을 가지고 있었다. 따라서 이러한 문제점을 해결하

† 준 회 원: 고려대학교 컴퓨터교육과 석사과정
†† 종신회원: 고려대학교 컴퓨터교육과 교수
* 본 논문은 2002년 고려대학교의 교내특별연구비에 의하여 지원되었음.

여 효율성을 향상시킨 많은 알고리즘들이 제안되어 왔다.[2,8,6,19,18,5,1,7,13].

최근에는, frequent itemset을 효율적으로 구하기 위해 MFI(Maximal Frequent Itemset)의 개념을 도입하여 보다 빠르게 frequent itemset를 찾아내려는 방법들이 연구되어 왔다[3,4]. 이러한 연구는 모든 frequent itemset을 찾아내는 대신, MFI만을 발견해 내는 것이 훨씬 효율적이라는 것을 보였다[3,4]. Pincer-search 알고리즘[4]은 상향식 탐색 방식과 하향식 탐색 방식을 모두 사용하여 MFI를 효율적으로 발견해 내게 된다. 또한 Mafia[3]알고리즘은 MFI를 찾아내기 위하여 깊이 우선 탐색 방식을 사용함과 동시에 효율적인 pruning 방법을 사용함으로써 빠르게 모든 MFI를 발견해 내었다.

이러한 기존의 연관규칙 마이닝 연구에서는, 대부분의 알고리즘이 horizontal 구조를 사용하여, 각 트랜잭션별로 발생한 아이템의 리스트를 유지하고 있으며 이러한 트랜잭션이 하나의 row를 이루어 전체 트랜잭션 데이터베이스를 구성하게 된다. 이 방법은 매 단계마다 아이템의 지지도를 구하기 위해 방대한 크기의 트랜잭션 데이터베이스에 계속적으로 접근해야 하는 오버헤드를 가지고 있으며, 이는 frequent itemset의 길이가 길어짐에 따라 비례하여 접근 비율도 증가한다는 단점을 가진다.

따라서, 이러한 문제점을 해결하기 위해 vertical 마이닝 방법이 등장하게 되었다. 기존의 horizontal 마이닝이 한번에 발생한 트랜잭션을 기준으로 각 트랜잭션별로 발생한 아이템의 리스트를 유지하는 구조를 가지는 반면, vertical 마이닝에서는 모든 트랜잭션이 고유의 ID를 가지며, 각 아이템을 기준으로 이 아이템이 발생한 모든 트랜잭션의 tid-list를 유지하는 구조를 가지게 된다.

이에 따라 많은 효율적인 vertical 마이닝 알고리즘들이 제안되어 왔다. [14,10,11,20] 이러한 vertical 구조는 트랜잭션 데이터베이스에 계속적으로 접근할 필요 없이 트랜잭션 리스트의 교집합만으로 frequent itemset을 구해냄으로써 I/O의 수를 현저히 줄여주었다. 그러나 이러한 장점에도 불구하고, frequent itemset의 크기가 길어짐에 따라 트랜잭션 벡터의 교집합을 위한 많은 연산이 요구될 뿐 아니라 이들의 중간 계산 결과의 리스트를 유지하기 위한 추가적인 공간을 요구하는 등의 한계점을 가지고 있었다.

본 논문에서는 frequent itemset을 구하기 위해 아이템들을 Least Support Itemset(LSI)과의 유사도에 의해 올림차순으로 정렬하여 탐색 트리를 구축하여 보다 빠르고 효율적으로 frequent itemset을 찾아내는 방법을 제안한다. 본 알고리즘은 vertical transaction vector(VTV)의 구조를 사용함으로써, 반복적으로 방대한 양의 트랜잭션 벡터에 불필요하게 접근하는 것을 막고, 트랜잭션 벡터들간의 AND 연산만을 통해 모든 frequent itemset을 발견해 낸다. LSI와의 유사도에 의한 아이템의 재정렬은 발생 가능한 많은 후보집합을 초기에 탐색 공간으로부터 제거할 뿐 아니라 MFI만을 찾아냄으로써 보다 빠르게 모든 frequent itemset을 발견해 내게 된다.

2. 문제 정의

연관 규칙과 MFI(Maximal Frequent Itemset): I = {i1, i2,⋯, im}를 데이터 아이템들의 집합이라고 가정하면, 트랜잭션 T는 I에 있는 아이템들의 부분집합으로 정의된다. 트랜잭션 데이터베이스 D는 n개의 트랜잭션들의 집합으로 구성되며, 각 트랜잭션은 고유의 TID(Transaction Identifier)를 가지고 구분 지어진다. 임의의 아이템 집합 X의 지지도(support)는 supp(X)라 정의하고, D의 트랜잭션 중에서 X를 부분집합으로 포함하고 있는 트랜잭션의 수를 의미한다. 최소 지지도(Minimum support)는 S_{min}이라 정의하고, supp(X) >= S_{min}을 만족하는 집합 X를 frequent itemset이라 한다. 또한, 만약 아이템 집합 X 가 frequent itemset이고 동시에 X의 어떤 수퍼셋(superset)도 frequent itemset이 아니라면, 이러한 X를 maximal frequent라 하고 이러한 아이템들의 집합을 maximal frequent itemset(MFI)이라 정의한다. 일반적으로 연관 규칙은 R: X->Y 와 같은 형태로 표현된다.(단, X, Y ⊆ I, X∩Y = ∅, Y≠∅) 즉, X라는 사건이 일어난다면 Y 또한 일어난다는 것을 이용해 특정 사건 혹은 아이템들 간의 연관성을 발견하는 것이다. 일반적으로 연관 규칙을 발견하기 위해서는 지지도(Support)와 신뢰도(Confidence)가 중요한 척도로써 사용된다. 연관 규칙 R에 대한 지지도는 supp(X∪Y)로 표현되며,

이는 이 규칙이 얼마나 자주 발생될 수 있는 지를 알려주는 척도가 된다. 또한 아이템 X를 포함하는 트랜잭션의 c%가 아이템 Y도 포함할 때, 연관 규칙의 신뢰도는 c라 말하고, 이는 supp(X∪Y)/supp(X)로 정의된다. 이는 그 룰 자체가 얼마나 신뢰성을 가지는 지를 보여준다.

Generic set-enumeration tree: 본 알고리즘에서 사용한 탐색 트리는 Rymon's generic set-enumeration tree search framework을 사용한다[17]. 이는 skewed 트리의 형태로 아이템들을 탐색 트리에 추가시켜 나가는 방법으로, 현재의 아이템들로 표현될 수 있는 모든 아이템들의 조합을 포함하게 된다. 이러한 탐색 트리의 핵심은 불필요한 가지를 제거하는 pruning 방법에 있다[16]. 또한, 모든 MFI를 찾아내기 위해, 본 알고리즘에서는 GenMax 알고리즘에서 사용했던 backtracking 탐색 방법을 사용한다[9]

3. LSI와의 유사도를 이용한 Vertical 마이닝

3.1. LSI 유사도에 기반한 탐색 트리

본 알고리즘은 horizontal 구조로 생성된 트랜잭션 D를 스캔하여 각 아이템별로 발생한 트랜잭션 리스트를 비트 벡터의 형태로 바꾸어 VTV 구조로 변환하면서 시작된다.<그림 1>:(a),(b). VTV 구조는 각 아이템이 발생한 트랜잭션을 0과 1로 구성된 일련의 스트림 형태의 비트 벡터로 표현하며, 이들은 각각 아이템들의 발생 여부를 의미하게 된다. 이 과정에서 사용자에 의해 주어진 S_{min}을 만족시켜 주지 못하는 1-itemset의 후보 집합들은 탐색 트리로부터 바로 제거된다. 이는 메모리의 낭비를 막고, 불필요한 연산을 줄여준다. VTV구조에서, 각 아이템의 지지도는 각각의 트랜잭션 벡터의 크기에 의해 정의된다. LSI(Least Support Itemset)는 최소의 지지도를 가지는 frequent 1-itemset으로써, 다음과 같이 정의된다.

$$LSI = Y \text{ such that } supp(Y) = \underset{1<=i<=n}{MIN}(supp(Xi))$$

(X 와 Y 는 frequent 1-itemsets, supp(Y)는 아이템 Y에 대한 지지도이며 n 은 아이템들의 총 수를 의미한다.)

이렇게 하여 발견된 LSI를 중심으로 다른 아이템들과의 유사도를 통해 아이템을 재정렬하게 된다. 임의의 두개의 비트 벡터의 유사도는 두 비트벡터 들간의 내적값에 의하여 정의된다. 즉, 임의의 아이템과 LSI와의 유사도는 LSI와 그 아이템이 얼마나 유사한가를 의미한다. 즉, 낮은 유사도는 두 아이템이 같은 트랜잭션에서 거의 함께 나타나지 않음을 의미한다. 예를 들면, <그림 1>:(c)는 LSI를 결정하고 이와의 유사도에 의해 아이템이 재정렬되는 과정을 보여준다. 아이템 C와 F가 유사도 1을 가진다는 것은 두 아이템이 오로지 한 트랜잭션에서만 함께 나타난다는 것을 의미한다. 즉, 2-itemset {C, F}의 지지도는 이의 유사도 1과 같으며, 이는 아이템 C를 부분집합으로 가지는 모든 2-itemset들 중에서 가장 작은 지지도를 가지는 아이템이 된다. 또한 이는 이 아이템이 현재의 탐색 트리로부터 가장 infrequent itemset이 될 가능성이 높은 아이템이 됨을 의미하게 된다.

Transaction ID	Items
T1	A B C E F
T2	A B C E G
T3	B D F
T4	A B E F
T5	A B F I
T6	B C E H

(a) Transaction dataset in horizontal

Item	bit-vector	support
A	1 1 0 1 1 0	4
B	1 1 1 1 1 1	6
C	1 1 0 0 0 1	3
E	1 1 0 1 0 1	3
F	1 0 1 1 1 0	4

(b) Dataset in VTV format

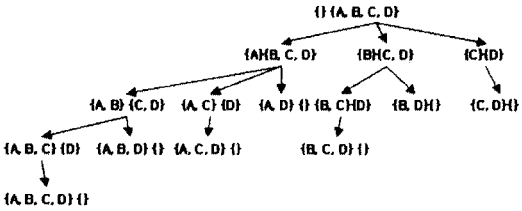
Item	bit-vector	similarity
C	1 1 0 0 0 1	LSI
F	1 0 1 1 1 0	1
A	1 1 0 1 1 0	2
B	1 1 1 1 1 1	3
E	1 1 0 1 0 1	3

(c) Sorted Itemset with LSI similarity

<그림 1> 트랜잭션 데이터베이스 (a) horizontal 구조 (b) VTV 구조 (c) LSI유사도에 의한 재정렬

정렬된 아이템들을 가지고 형성된 트리의 각 노드는 두 개의 아이템 집합으로 구성된다: Possible itemset(P(i))와 Applicant itemset(A(i)). 트리의 루트 노드의 possible itemset은 공집합이고 application itemset은 모든 frequent 1-itemset을 가진 상태에서 시작된다. 즉 각 레벨별로 A(i)의 아이템을 하나씩 P(i)로 추가하면서 다음 가지를 형성한다. <그림 2>는 트리의 루트 노드로부터 다음 가지들이 어떤 식으로 형성되는

지를 보여준다. <그림 2>에서 보이듯, balanced tree를 구성하는 대신, 트리의 한쪽 가지에 후보 집합들을 많이 배치시켜 놓은 형태의 트리를 구축하여, 초기에 많은 pruning이 이루어 질 수 있도록 아이템들을 재배치하였다. 즉, 이는 트리의 특성상 왼쪽 가지에 많은 후보 집합이 존재하게 되고, 결국 왼쪽가지에서 불필요한 가지의 제거가 전체적 효율성을 좌우하는 중요한 요소가 된다. 따라서, LSI를 중심으로 유사도가 적은 순으로 아이템들을 정렬하여 초반에 frequent itemset이 될 가능성이 적은 것들을 리스트의 앞쪽에 배치시켜 초기에 많은 pruning이 이루어 질 수 있도록 하였다.



<그림 2> 탐색 트리의 생성

깊이 우선 탐색시, 각 노드의 지지도는 possible itemset에 있는 모든 아이템의 비트벡터들간의 AND연산에 의해 구해진다. 예를 들면, itemset {C, A, B}의 지지도는[1 1 0 0 0 1] AND [1 1 0 1 1 0] AND [1 1 1 1 1 1]에 의해 2가 된다. 만약 이의 자식 노드의 지지도가 S_{min}을 만족시키지 못할 경우, 아이템 집합 {C,A,B}는 MFI의 정의에 의해 MFI가 된다.

이런 식으로 일단 MFI가 발견된 후에는, 다음의 몇가지 휴리스틱 방법을 사용하여 좀 더 빠르게 모든 MFI들을 찾아내게 된다.

▸발견된 MFI의 서브트리(Sub-tree)는 탐색 공간으로부터 제거되어 진다. 즉, MFI의 모든 수퍼셋(superset)은 탐색 트리로부터 제거된다. 예를 들면, 만약 아이템 {C, F}가 frequent itemset이 아니라면, 이것을 부분집합으로 가지는 모든 집합 {C,F,A}, {C,F,B}, {C,F,A,B}들 또한 탐색 공간으로부터 제거된다.

▸MFI의 부분 집합 또한 frequent이므로, 이들 역시 탐색 트리로부터 제거된다. 예를 들어 possible itemset을 P(i)라 하고, applicant itemset을 A(i)라 할 때, $(P(i) \cup A(i)) \subseteq$ MFI를 만족하다면, 이 노드는 탐색 공간으로부터 바로 제거되어 진다. 즉, <그림 4>에서, 아이템 집합 {F, A, B, E} 가 MFI로 발견된다면, 트리의 노드 [{A}, {B, E}]과 [{B, E}{}]은 모두 탐색 트리로부터 제거된다. 노드 [{A}, {B, E}]에서, P(i) = {A}, A(i) = {B, E}, $(P(i) \cup A(i))$ = {A, B, E} $\subseteq$ MFI = {F, A, B, E}이므로, 이 노드를 포함한 아래 가지는 모두 탐색 트리로부터 즉시 제거된다. 이러한 휴리스틱 방법은 long pattern을 가지는 트랜잭션 데이터베이스에서 보다 효과적으로 사용되어진다.

본 알고리즘의 탐색 트리는 주어진 트랜잭션 데이터 베이스의 아이템들로 만들어 질 수 있는 모든 가능한 조합을 포함한다는 점에서 completeness를 가진다고 말할 수 있다. 또한, 제안된 알고리즘의 탐색 방법은 탐색공간으로부터 infrequent itemset만을 정확히 제거함으로써 모든 frequent itemset을 발견해 낸다는 점에서 soundness를 가진다. 이 알고리즘은 탐색의 초기 단계에서 frequent itemset이 될 수 없는 것들을 재빠르게 제거해 냄으로써 보다 효율적으로 frequent itemset을 찾아내게 된다

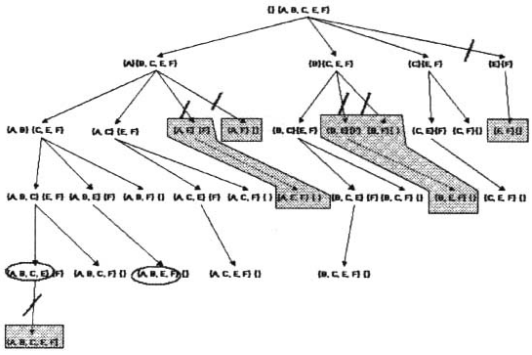
3.2. 예 제

<그림 3>과 <그림 4>는 <그림 1>의 데이터 셋으로 구성된 두개의 다른 탐색 트리를 보여준다. <그림 4>의 탐색 트리는 LSI와의 유사도에 의해 오름차순으로 재정렬된 아이템을 가지고 형성된 트리이다. 마찬가지로 <그림 3>에 나타난 탐색 트리는 임의의 순서로 구성된 아이템 즉, <그림 1>:(b)에 나타난 데이터 셋으로 구성되어졌다. 그림에서 보여지듯, <그림 4>에 보여지는 탐색 트리와 탐색 방법은 <그림 3>과 비교해서 탐색 공간을 크게 줄여준다. 특히, LSI와의 유사도에 따른 정렬 방법을 사용함으로써, 본 논문에서 제안하는 알고리즘은 트리의 왼쪽 가지에서 최대의 pruning이 이루어 지도록 하며, infrequent itemset X가 frequent itemset Y와 결합하여 불필요한 연산을 만들어 내는 것을 막는다. (Lemma 2)

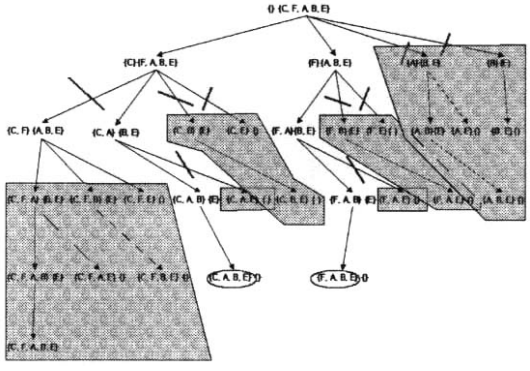
Lemma 1. 만약 itemset X 가 MFI라면, X의

모든 부분집합들 또한 frequent itmesets이다.

Lemma 2. 만약 itemset X 가 infrequent itemset 이고, itemset Y 가 frequent itemset이라면, itemset (X∪Y)는 항상 infrequent itemset이다.



<그림 3> 정렬 전의 set-enumeration 트리



<그림 4> LSI와의 유사도에 따라 정렬된 set-enumeration 트리 (트리의 어두운 영역은 탐색 과정에서 pruning되는 부분을 나타낸다.)

<그림 1>-(C)에서 아이템 C가 LSI이고, 이것과의 유사도에 의해 오름차순인 F, A, B, E의 순으로 아이템들이 재정렬된다. 초기에 루트 노드는 공집합인 possible set 과 {C,F,A,B,E}를 가지는 applicant set으로 구성된다. 이로부터 루트 노드는 application itemset으로부터 하나씩을 possible set으로 옮기면서 4개의 자식 노드를 분기한다. 아이템 C는 LSI와의 유사도가 가장 작으므로, pruning될 가능성이 가장 큰 아이템이 되고, 이들을 앞에 배치함으로써 되도록 많은 가지를 탐색 없이 제거하게 된다. 가장 왼쪽의 첫번째 노드인 {C,F}가 가장 먼저 테스트 되어지고, 이것의 지지도가 아이템 C와 F의 트랜잭션 백터의 AND연산에 의해 계산되어진다. 이 노드의 지지도 값은 1이고,(i.e. [110001] AND [101110]=1) 이는 최소 지지도 Smin을 만족시키지 못하므로, 이 노드를 포함하는 전체 서브트리 또한 탐색 트리로부터 제거된다.

또한 탐색 과정에서 노드 {C,A,B,E}의 application set이 공집합이며, 동시에 S_{min}을 만족시켜주므로이는 MFI로 발견된다. 따라서, 이의 모든 부분집합 또한 frequent itemset이 되므로 이들 또한 탐색 공간으로부터 별도의 연산 없이 제거될 수 있다(by Lemma 1). 노드 [{C, A, E}{}], [{C, B}{E}], [{C, E},{}], [{A}{B, E}], [{B},{E}]들 또한 마찬가지로 탐색 트리로부터 즉시 pruning되어진다. 같은 방식으로, MFI인 {F, A, B, E}는 노드 [{F, A, E}{}], [{F, B}{E}], [{F, E}{}]들을 탐색 트리로부터 pruning해 낸다.<그림 1>과 <그림 4>의 예에서, 본 알고리즘은 두개의 MFI을 발견하기 위해 전체 31개의 노드 중에서 단 7개의 노드들만을 조사한다.(i.e., {C,A,B,E}과 {F,A,B,E}). 반면, <그림 3>과 같이 아이템의 정렬없이 만들어진 탐색 트리의 경우, 두개의 MFI을 발견하기 위해,총 18개의 노드를 조사하게 된다. 그리고 이처럼 발견된 MFI의 모든 부분집합이 frequent itemset이 된다

4. 실험 결과

본 실험에서는 frequent itemset을 찾아내기 위한 가장 대표적인 알고리즘인 VIPER, MaxCliqu와 알고리즘의 성능을 비교하였다. VIPER(Vertical Itemset Partitioning for Efficient Rule-extraction) 는 "snakes"라고 불리우는 압축된 비트벡터의 형태로 데이터를 저장하고 이들을 통해 frequent itemset을 구하는 알고리즘으로써, "snakes"의 효과적인 생성과 연산을 위해 다양한 방법들을 사용한다[12]. MaxClique 또한 clustering 기술을 사용하여 효과적으로 모든 MFI을 찾아내는 효과적인 알고리즘이다[11]. 이 논문에서는 IBM의 Quest group에서 제공하는 Synthetic Data Generation을 이용하여 랜덤하게 생성된 데이터 셋을 이용해 실험하였으며, 실제

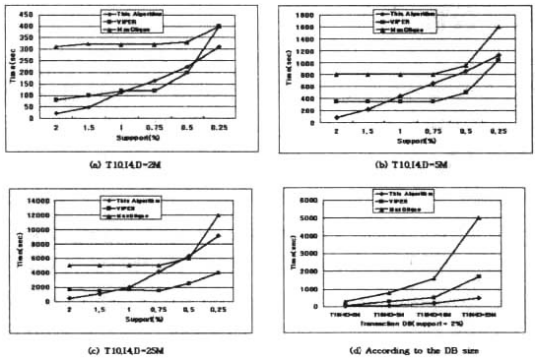
트랜잭션 당 발생 아이템의 수인 |T|와 MFI의 평균 발생 아이템의 수인 |I| 및 발생 가능한 MFI의 수인 |L|를 변화시켜가면서, 여러 가지 데이터 셋에 따른 알고리즘의 성능 평가를 하였다. 데이터 생성시 사용된 변수들과 기본 값들은 <표 1>과 같다.

Parameter Symbol	Parameter Meaning	Default Value
|D|	Number of transactions	2M ~ 25M
|T|	Mean transaction length	10
|I|	Mean frequent itemset length	4
|L|	Number of maximal frequent itemsets	2000
N	Number of Items	1000

<표 1>. 파라미터 정의

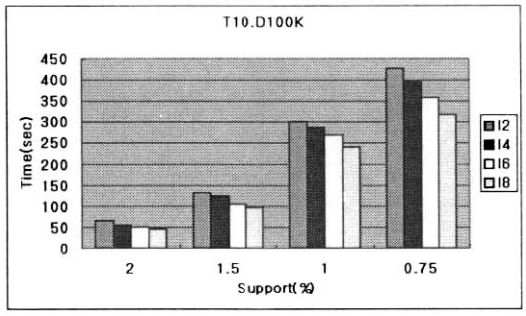
알고리즘을 수행시키기 위한 컴퓨터 시스템은 OS는 Windows 2000, 메모리 256MB, 하드15GB의 사양을 사용했으며 알고리즘의 구현은 Java를 사용하였다.

알고리즘에 따른 실험평가 요소는 지지도를 2%에서 0.25%로 변화시킴에 따른 알고리즘의 속도 변화를 비교하였다. 우선, T10I4로 트랜잭션 당 아이템의 수와 MFI의 평균 크기를 고정한 상태에서 트랜잭션 데이터 베이스의 크기를 변화시켜가면서 알고리즘의 속도 변화를 측정하였다. 이의 실험 결과는 <그림5>:(a),(b),(c)에 보여진다.



<그림 3> 데이터 베이스 크기에 따른 알고리즘의 속도 비교

그래프에서 보이듯, 본 알고리즘은 MaxClique와 VIPER에 비해 평균적으로 좋은 성능을 보여준다. 특히, 다른 알고리즘에 비해, 본 알고리즘은 지지도 값이 작아짐에 따라 알고리즘의 수행시간에 있어 완만한 증가를 보여준다. 즉, 지지도가 작아짐에 따라 발견되는 frequent itemset의 수가 크게 증가한다는 점을 고려할 때, 본 알고리즘은 발견되어지는 frequent itemset의 크기에 비교적 적게 영향을 받는다는 사실을 알 수 있다. 반면, Viper와 MaxClique 는 지지도 값이 작아짐에 따라 알고리즘의 속도가 상당히 떨어지는 것을 알 수 있다. 또한, <그림 5>:(d)에서 보이듯, 지지도 값이 2%, 1.5%일 때, 본 알고리즘은 트랜잭션 데이터베이스의 크기에 매우 적게 영향을 받고, 이에 따른 실행 시간의 증가가 비교적 적다는 것을 알 수 있다. 반면 MaxClique와 Viper는 트랜잭션 데이터베이스의 크기가 커짐에 따라 상당한 정도로 수행 속도가 느려짐을 알수 있다.



<그림 6> MFI의 크기에 따른 수행 시간 비교

본 실험에서는 같은 크기의 트랜잭션 데이터베이스에 대해, MFI의 크기를 변화시켜 가면서, 그 속도 변화를 비교하였다. MFI의 평균 길이인 I를 2에서 8까지 변화시킬 때, 제안된 알고리즘의 성능은 점점 향상된다는 것을 실험을 통해 확인하였다.

5. 결 론

본 논문에서는, LSI와의 유사도에 의해 아이템들을 재정렬하여 set-enumeration tree를 만들어 frequent itemset을 찾아내는 효율적인 vertical 마이닝 알고리즘을 제안했다. 본 알고리즘은 LSI를 중심으로 유사도에 따른 정렬을 통해 탐색 트리를 구축하여, 탐색의 초기에 되도록 많은 가지

를 제거하여 탐색 공간을 줄임과 동시에, 여러 가지 휴리스틱 방법을 사용해, maximal frequent itemse만을 발견해 냄으로써, 보다 빠르게 frequent itemset을 발견해 낸다. 특히, 본 논문에서 제안하는 알고리즘은, 높은 빈발도의 frequent itemset을 가지는 데이터 베이스나 long pattern을 가지는 데이터 베이스에서의 경우, 많은 후보 집합들을 탐색 공간으로부터 초기에 제거함으로써, 불필요한 연산을 최소화하여 빠른 속도로 모든 frequent itemse을 발견해 내는 것을 실험적으로 확인하였다

참 고 문 헌

[1] A, Mueller., "Fast sequential and parallel algorithms for association rule mining : A comparison", Technical Report No, CS-TR-3515 of CS Department, University of Maryland-College Park.

[2] 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.(partition).

[3] D. Burdick, M.Calimlim, and J,Gehrge. "MAFIA : a maximal frequent itemset algorithm for transactional databases", In Inl, Conf, on Data Engineering, Apr. 2001.

[4] D. I. Lin and Z. M. Kedem. "Pincer-search: A new algorithm for discovering the maximum frequent set". Proc, of he 6th Int'l Conference on Extending Database Technology(EDBT), Valencia, Spain, 1998.

[5] H. Mannila, H. Toivonen, and A. I. Verkamo. "Improved methods for finding association rules", In Proc, AAAI Workshop on Knowledge Discovery, July 1994.

[6] Hannu Toivonen, "Sampling Large Database for Association rules," In Proc. of the 22nd International Conference on Very Large Data Bases (VLDB'96), Mumbai(Bombay), India, 1996.

[7] J. Han, Y. Fu, "Discovery of multiple-level association rules from large databases", In 21st VLDB, Sept.1995.

[8] 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.

[9] K. Gouda and M. J. Zaki Efficiently Mining Maximal Frequent Itemsets. In 1stIEEE International Conference on Data Mining, November 20001.

[10] M. J. Zaki and K, Gouda. "Fast vertical mining using Diffsets" TR 01-1, CS Dept., RPI, Mar.2001.

[11] M. J. Zaki, S. Parasarathy, M. Ogihara, and W. Li. "New algorithms fast discovery of association rules". In Proc, of 3rd Inl. Conf, on Knowledge Discovery and Data Mining(KDD), August 1997.

[12] P. Shenoy, J. R. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa, and D. Shah, "Turbo-charging Vertical Mining of Large Databases", SIGMOD Conference 2000

[13] R. Agrawal and J. C. Shafer, "Parallel Mining of association rules : Design, Implementation and Experience", IBM Research Report RJ10004, Feb.1996

[14] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," In Proc. of the 20th International Conference on Very Large Data Bases (VLDB94), pp. 487-499, Santiage, Chile, September 1994.

[15] R. Agrawal, T. Imielinski, and A. Swamy. Mining association rules between sets of items in large databases". In Proc, of ACM SIGMOD Intl. Conf. on Management of Data, May 1993.

[16] R. J. Bayardo. Efficiently mining long patterns from databases. In ACM SIGMOD

Conf., June 1998.

[17] Rymon, R. 1992, Search through Systematic Set Enumeration. In Proc. of Third Int'l Conf. On Principles of Knowledge Representation and Reasoning, 539-550

[18] R. Srikant and R. Agrawal., "Mining generalized association rules", In 21st VLDB, Sept. 1995.

[19] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, 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

[20] 고윤희 · 김현철. TID List를 이용한 빈발 항목의 효율적인 탐색 알고리즘. 정보과학회 학술발표논문집 제29권 1호, pp136-138 (2002)

고 윤 희

2001 고려대학교 컴퓨터교육과(이학학사)
2001~현재 고려대학교 컴퓨터교육과 석사과정
관심분야: 데이터마이닝, CAT
E-Mail: unygo@comedu.korea.ac.kr

김 현 철

1988 고려대학교 전산과학과 졸업
1990 미조리 주립대학(Rolla) (전산학석사)
1998 플로리다 대학(전산학박사)
1998 GTE Data Services, Inc. 시스템 분석가
1998 삼성 SDS 책임컨설턴트
1999~현재 고려대학교 컴퓨터교육과 교수
관심분야: 인터넷기반 학습이론, 데이타마이닝
E-Mail: hkim@comedu.korea.ac.kr

이 원 규

1985 고려대학교 영어영문학과 졸업
1989 츠쿠바대학 전자정보공학전공(공학석사)
1993 츠쿠바대학 전자정보공학전공(공학박사)
1993~1995 한국문화예술진흥원 책임연구원
1996~현재 고려대학교 컴퓨터교육과 교수
관심분야: 컴퓨터교육, 정보검색, 데이타베이스
E-Mail: lee@comedu.korea.ac.kr