The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 19, No. 2, pp.127-133, Apr. 30, 2019. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2019.19.2.127
JIIBC 2019-2-18

데이터베이스에서 빈발패턴의 추출을 위한 메모리 향상기법

Memory Improvement Method for Extraction of Frequent Patterns in DataBase

박인규*

In-Kyu Park*

요 약 지금까지의 빈발 항목 추출에서는 FP-Tree에 대한 순회와 패턴의 탐색이 필수적인 과정이기 때문에 마이닝 데이터를 트리에 저장하는데 공간이 필요하고 탐색하는데 CPU시간이 필요하기 마련이다. 이러한 단점을 극복하기 위하여 본 논문에서는 조건부 FP-Tree의 의존하지 않고 트랜잭션 데이터의 각 항목들의 위치 정보를 부여하여 트랜잭션 데이터를 2차원의 위치정보 Look-Up테이블로 변환하여 시간과 공간적인 접근성을 용이하게 한다. 또한 항목과 항목의 위치에 대한 매핑배열을 병행하여 시간 복잡도를 줄이는 방법을 고려하는 알고리즘을 제안한다. 실험 결과를 통하여 제안된 방법은 FIMI 저장소 웹 사이트에서 얻은 데이터 세트를 기반으로 많은 실행 시간과 메모리 사용을 줄일 수 있음을 보였다.

Abstract Since frequent item extraction so far requires searching for patterns and traversal for the FP-Tree, it is more likely to store the mining data in a tree and thus CPU time is required for its searching.
In order to overcome these drawbacks, in this paper, we provide each item with its location identification of transaction data without relying on conditional FP-Tree and convert transaction data into 2-dimensional position information look-up table, resulting in the facilitation of time and spatial accessibility. We propose an algorithm that considers the mapping scheme between the location of items and items that guarantees the linear time complexity. Experimental results show that the proposed method can reduce many execution time and memory usage based on the data set obtained from the FIMI repository website.

Key Words : Frequent Itemset Mining, FP-Growth, Minimum Support, Pruning

Ⅰ. 서 론

연관 규칙 마이닝 (Association Rule Mining)은 데이터 마이닝과 지식 발견의 주요한 기능으로 트랜잭션 데이터베이스의 항목들 사이의 상관관계를 나타낸다. 이러한 관계의 응용분야는 로그인 분석, 침입 탐지를 비롯하여 많은 분야에 적용될 수 있다. 일반적으로 ARM에는 빈발 항목을 찾고 이를 토대로 규칙을 추출하는 과정이 필수적이다. 특히 빈발항목의 탐색에는 탐색시간과 저장공간이 많이 필요하므로 이를 극복하기 위해 다양한 알고리즘이 제안되었다[1,2].

지금까지의 여러 알고리즘의 공통적인 특징은 빈발 패턴을 찾기 위해 재귀적으로 조건부 FP-Tree를 작성하면 대부분의 CPU 시간이 소비되고 , 빈발하지 않은 항목

*정회원, 중부대학교 게임소프트웨어학과
접수일자 2019년 2월 11일, 수정완료 2019년 3월 11일
게재확정일자 2019년 4월 5일

Received: 11 February, 2019 / Revised: 11 March, 2019 / Accepted: 5 April, 2019
*Corresponding Author: ikpark@joongbu.ac.kr
Dept. of Computer Science, Chungwoon University, Korea

을 제거한 후 항목 빈도의 새 순서에 따라 조건부 FP-트리를 작성하기 때문에 많은 공간을 필요하게 된다[3,4]. 본 논문에서는 조건부 FP-Tree를 생성하지 않고 빈발 항목을 추출하기 위한 FP-Tree를 Look-Up Array로 변형한 알고리즘을 제안한다. 먼저 데이터베이스를 스캔하여 모든 빈발한 항목을 찾고 두 번째 스캔은 각 트랜잭션의 빈발한 항목을 FP-Tree용 Look-Up테이블을 생성하고 항목과 항목의 위치정보를 FP-Array에 매핑한다. 따라서 조건부 FP-Tree의 많은 후보를 발생하는 대신 FP-Tree탐색이 아닌 배열을 기반으로 조건부 FP-Tree에 해당하는 후보를 생성하는 새로운 데이터 구조를 제안한다.

II. Prefix-Tree

빈발패턴의 추출을 위한 자료구조로써 FP-Tree는 데이터베이스를 가장 효율적으로 나타낸다. 항목의 집합은 $I = \{a_1, a_2, \ldots, a_m\}$이고, 트랜잭션 데이터베이스는 $DB = T_1, T_2, \ldots, T_n$, 이다. 여기서, T_i ($i \in [1. \ldots n]$)는 트랜잭션으로 I 에 있는 항목들의 집합을 포함한다. 패턴 A의 지지도(Support)는 A가 항목 집합인 경우 DB에 A가 포함된 트랜잭션 수이다. 패턴 A는 A의 지지도가 미리 정해진 최소지지 임계값 ξ보다 작으면 빈발하다.

표 1은 임의의 트랜잭션 데이터베이스를 그림 1는 헤더 테이블과 FP-Tree를 나타낸다. FP-Growth 마이닝은 반복적인 조건부 FP-Tree를 구축하는 것이다. 예를 들어 "C"의 경우를 보면 "C"로 끝나는 경로가 세 가지가 탐색된다. 그림 2 와 같이 FP-Tree 헤더 테이블의 "C"를 포함하는 경로를 따라 표 2와 같은 새 헤더 테이블을 작성하는 항목의 새로운 빈도수를 결정된다. 여기서"A"와 "B"는 최소 지지도를 2 라고 가정했을 경우에 빈발항목이다. 항목 "C"를 포함하는 경로가 트리에 있는 해당 항목 집합을 찾기 위해 한 번 더 방문되고, 모든 빈발하지 않은 항목을 제거한 후 새로운 헤더 테이블에 새로운 순서에 따라 그림 3과 같이 조건부 FP-Tree가 생성된다[5,6].

표 3은 그림 1의 FP-tree에서 "E"로부터 "B"까지 링크에 따라서 분할정복에 의한 탐색에 의하여 얻어진 조건부 패턴과 조건부 FP-Tree를 나타낸다[7,8,9,10,11,12].

표 1. 트랜잭션과 항목의 위치
Table 1. Transaction and locations of items

TID	Items
1	{A B}
2	{B C D}
3	{A C D E}
4	{A D E}
5	{A B C}
6	{A B C D}
7	{B C}
8	{A B C}
9	{A B D}
10	{B C E}

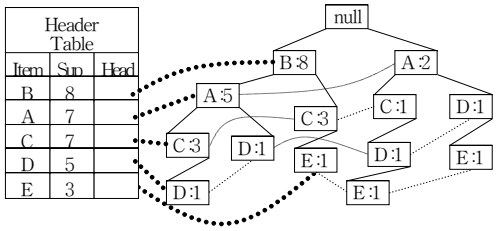

그림 1. 헤데 테이블과 FP-Tree
Fig. 1. Header table and FP-Tree

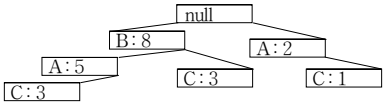

그림 2. "D" 로 끝나는 경로
Fig. 2. Paths ending in "C"

표 2. 새로운 헤더테이블
Table 2. New header table

Items	Supp
A	2
B	2

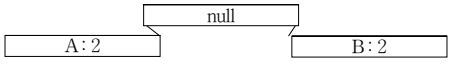

그림 3. "C" 에 대한 조건부 FP-Tree
Fig. 3. Conditional FP-Tree for surffix "C"

표 3. 조건부패턴베이스에 의한 패턴마이닝
Table 3. Mining patterns by conditional pattern base

항목	조건부 패턴	조건부 FP-Tree
E	{BC:1, ACD:1, AD:1}	{A:2, C:2, D:1}|E
D	{BAC:1, BA:1, BC:1, AC:1, A:1}	{A:4, B:3, C:3}|D
C	{BA:3, B:3, A:1}	{A:4, B:6}|C
A	{B:5}	{B:5}|A
B	∅	∅

결국 조건부 FP- 트리를 만들고 횡단함으로써 빈발한 패턴을 찾는 것은 많은 CPU 시간과 메모리를 수반한다. 기 수행된 실험에 따르면 CPU 시간의 거의 80 %가 FP-Tree를 탐색하는 시간이다. FP-Tree를 탐색하는 시간과 조건부 FP-Tree를 구성하는 과정을 Look-Up 테이블을 이용하여 시간과 공간의 제약을 극복하는 방법을 제안한다.

Ⅲ. 제안된 방법

1. Look-Up Array

제안하는 알고리즘은 FP-Growth처럼 헤더 테이블의 상향 탐색을 통하여 수행된다. 주어진 데이터 세트의 트랜잭션을 스캔하여 각 항목의 지지도를 계산하여 내림차순으로 헤더 테이블을 작성한다. 또한 지지도에 따라서 트랜잭션의 항목을 내림차순으로 정렬한다. 각 노드는 FP-Tree에서와 같이 노드 위치를 나타내기 위해 NodeID라는 특정 식별자를 표 4와 같이 부여하여 NodeID의 배열로 구성된 각각의 경로를 Look-Up Array를 구성한다.

Look-Up Array에 접근하기 위하여 FP-Array을 표 5와 같이 운영한다. 이는 노드의 NodeID에 해당하는 항목(Items)과 지지도(Supp) 그리고 다른 경로를 구성하는 같은 항목의 인덱스(Links)를 기록하는 데 사용되며 NodeID로 인덱싱되어진다. 또한 헤더 테이블의 항목과 동일한 항목에 해당하는 NodeID를 인덱싱할 수 있다. 따라서 동일한 항목을 가진 모든 항목의 모등 경로를 탐색할 수 있다. 이러한 데이터 구조는 조건부 FP- 트리를 생성하지 않고도 FP-Tree를 쉽게 탐색 할 수 있다. 다른 노드의 모든 위치를 유지하기 위해 FP-Array를 사용하여 NodeID를 기반으로 배열을 탐색하기 때문에 많은 메모리와 CPU 시간을 절약할 수 있다.

표 4. 트랜잭션의 Look-Up Array
Table 4. Look-Up table of transaction

TID	Node identification of Items
1	{B: 1, A: 2}
2	{B: 1, C: 3, D: 4}
3	{A: 5, C: 6, D: 7, E: 8}
4	{A: 5, D: 9, E: 10}
5	{B: 1, A: 2, C: 11}
6	{B: 1, A: 2, C: 11, D: 12}
7	{B: 1, C: 3}
8	{B: 1, A: 2, C: 11}
9	{B: 1, A: 2, D: 13}
10	{B: 1, C: 3, E: 14}

표 5. Look-Up Array을 위한 FP-Array
Table 5. FP-Array for Look-Up Array

Header Table			NodeID	Items	Supp	Links
Items	Supp	Head				
B	8	→	1	B	8	/
A	7	→	2	A	5	5
C	7	→	3	C	3	6
D	5	→	4	D	1	7
			5	A	2	/
			6	C	1	11
			7	D	1	9
E	3	→	8	E	1	10
			9	D	1	12
			10	E	1	14
			11	C	3	/
			12	D	1	13
			13	D	1	/
			14	E	1	/

2. 빈발 패턴의 발생

Look-Up Array와 FP-Array를 통하여 빈발 패턴을 추출한다. 표 6은 모든 경로들의 집합이 항목 E를 포함하는 경로의 Look-Up을 나타낸다. 경로를 스캐닝 한 후에 빈발하지 않은 3 개의 빈발한 항목, 즉 노드 C, A, D가 발견 될 수 있다. 표 7은 E로 끝나는 빈발 항목에 대한 새로운 헤더 테이블을 보여준다. Look-Up에 대한 또 다른 스캔은 그들의 지지도에 따라 New_H에서 자주 항목의 위치를 찾기 위해 수행된다. 표 7과 같이 항목에 레이블 E가 있는 둘 이상의 하위 항목이 있는 경우 하위 항목의 지지도가 표 8과 같이 하위 항목의 합계가 된다. 예를 들어 위치 "5"의 항목 A에는 두 개의 하위 항목이 있으므로 각 빈도수는 1로 A의 지지도는 "2"이다. 따라서 표9와 같이 "E"의 빈발 패턴이 추출된다.

표 6. "E" 로 끝나는 경로
Table 6. Paths ending in "E"

TID	Node identification of Items
3	{A:5}←{C:6}←{D:7}←{**E:8**}
4	{A:5}←{D:9}←{**E:10**}
10	{B:1}←{C:3}←{**E: 14**}

표 7. 헤더테이블
Table 7. header table

Items	Supp.
C	2
D	2
A	2

표 8. "E" 의 FP-Array
Table 8. FP-Array of "E"

NodeID	Items	Supp	Links
8	E	1	10
10	E	1	10
14	E	1	/

표 9. "E" 의 빈발패턴
Table 9. Frequent items of "E"

Items	NodeID of Items
C	{3:1, 6:1}
D	{7:1, 9:1}
A	{5:2}

표 10. "CE" 로 끝나는 경로
Table 10. Paths ending in "CE"

TID	Node identification of Items
3	{A:5}←{**C:6**}←{D:7}←{**E:8**}
10	{B:1}←{**C:3**}←{**E: 14**}

표 11. 헤더테이블
Table 11. header table

Items	Supp.
B	1
A	1

표 12. “DE” 로 끝나는 경로
Table 12. Paths ending in “DE”

TID	Node identification of Items
3	{A:5}←{C:6}←{**D:7**}←{**E:8**}
4	{A:5}←{**D:9**}←{**E:10**}

표 13. 헤더테이블
Table 13. header table

Items	Supp.
A	2
C	1

표 14. “AE” 로 끝나는 경로
Table 14. Paths ending in “AE”

TID	Node identification of Items
3	{**A:5**}←{C:6}←{D:7}←{**E:8**}
4	{**A:5**}←{D:9}←{**E:10**}

표 15. 헤더테이블
Table 15. header table

Items	Supp.

FP-Growth Algorithm
Input: FP-Tree, H is Header_table for FP-Tree
min_sup is the minimum support threshold
Output: the complete set of frequent patterns

```
for each item x in H do
  get the node n pointed by the link of x in H
  while n ≠ null
    find a path x1, x2, ..., xm from the parent node of n to the
child node of the root
    count the frequency for each item z along this path
    reset a new node n to the next node pointed by the link of
current node n
  end while
  for each counted item z
    if the frequency count of z≥min-sup then
     insert z into a frequent item list Fx
     add the set of locations for item z to the list Lx[z]
     print {zx}
    end if
  end for
 suffix = "x"
 FP-Growth2(Fx, Lx, LookUp, FP-Array, suffix)
end for
```

FP-Growth2(F_x, L_x, LookUp, FP-Array, suffix)
Input: F_x:the list fo frequent items traced from node x
Lx:the list of locations for the frequent items in F_x
LookUp:list of list of locations for transaction
FP-Array:list between items and locations in LookUp
Suffix-x:the previous frequent items with item x
min-sup:the minimum support threshold
Output: Full set of frequent patterns

```
for each element z in list Fx
  if z exists in one location in the tree
    let P is the single path for z and a=z∪suffix-x
    single_path(P, a);
  else if z exists in many locations
    use the list of locations for z(Lx[z]) to access LookUp
    find the frequent list of items from z nodes
    denote this list as F* and their locations as Lz
    for each frequent item such as y in Fz
      print {yz∪suffix-x}
    end for
      suffix-y = z∪suffix-x
    FP-growth2(Fz, Lz, LookUp, FP-Array, suffix-y)
  end if
end for
```

Look-Up을 스캔 한 후 Prefix {CE, AE, DE}를 얻을 수 있다. 표 10~표 14는 이러한 접미어를 포함하는 Look-Up을 나타낸다. 위치와 빈도수는 CE {3:1, 6:1}, DE {7:1, 9:1}, AE {5:2}로 기록되는 반면, {3,6}은 접미사 CE이고 {1, 1}은 해당 위치의 빈도를 나타낸다. 위치 목록을 사용하여 CE, AE 및 DE로 끝나는 모든 빈발한 항목을 재귀적으로 생성된다. CE로 끝나는 빈 항목을 얻으려면 위치 {3, 6}의 항목 CE를 포함하는 경로를 사용하여 이 경로를 따라 새 카운트를 결정합니다. 그림 6의 (a)는 주파수 "1"이 최소 지지도보다 작은 패턴 {ACE, BCE}를 도시한다. 그러나 접미어 DE에는 빈발한 항목이 하나 있습니다 (예 : ADE). 마지막으로 접미사 AE에 대해 부모가 null이기 때문에 빈발한 항목 계산을 중단합니다. 아래의 FP-Growth 알고리즘과 FP-GrowthPlus 알고리즘은 빈발 패턴을 생성하기 위해 제안한 방법의 의 코드를 나타낸다.

Ⅳ. 실험 및 결과

1. 실험환경 및 데이터 집합

제안된 알고리즘의 성능적 특징을 분석하기 위하여 표 16과 같은 데이터 집합을 사용하였다. 표 16의 chess, retail, connect 데이터 집합들은 빈발 패턴 탐색 알고리즘의 성능을 측정하기 위한 실질적인 데이터이다. 모든 실험은 Microsoft Windows 10(인텔 ® 코어 ™ i7-7700 CPU @3.6 GHz 및 16 GB 메모리)하에서 Java 프로그램 언어를 사용하여 구현하여 수행되었다.

표 16. 데이터 집합의 특징
Table 16. The characteristics of Datasets

Datasets	Size(MB)	No. of Trans	Distinct items	Avg. Trans. Len.
retail	0.56	8124	119	23
chess	0.34	3196	75	37
connect	30.5	67557	42	8.1
T1014D100K	3.83	100000	942	10.2

2. 성능분석

그림 4부터 그림 6은 Minimum Support가 다른 데이터 세트에서 제안된 알고리즘, FP-Growth, FP-Growth*, FP-Growth+ 및 CT-PRO 알고리즘 간의 실행 시간의 비교를 나타낸다. FP-Growth2는 제안된 알고리즘을 나

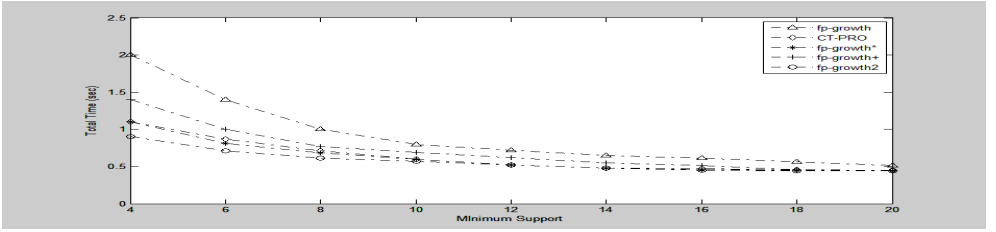

그림 4. retail 데이터의 실행시간
Fig. 4. Execution time on dataset retail

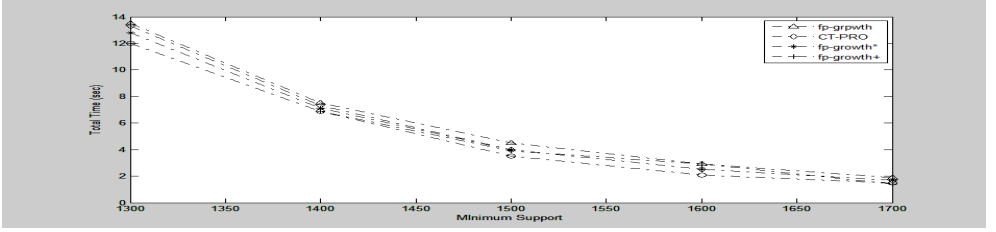

그림 5. chess 데이터의 실행시간
Fig. 5. Execution time on dataset chess

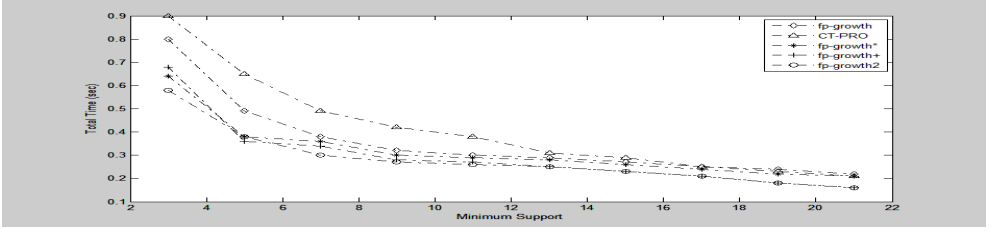

그림 6. T1014D100K 데이터의 실행시간
Fig. 6. Execution time on dataset T1014D100K

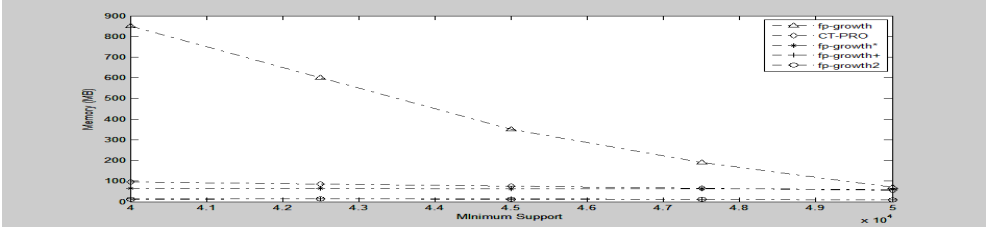

그림 8. connect 데이터의 메인 메모리 사용량
Fig. 8. Main memory usages on dataset connect

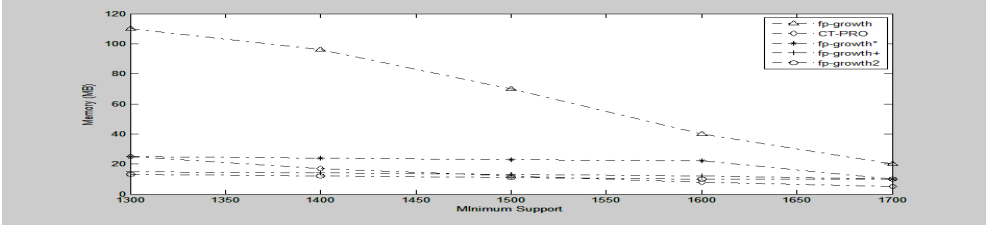

그림 9. chess 데이터의 메인 메모리 사용량
Fig. 9. Main memory usages on dataset chess

타낸다. 그림 4는 데이터 집합 retail에서 제안된 알고리즘의 계산 시간을 보여준다. 그림 4에서 제안된 방법의 계산 시간은 FP-Growth의 실행 시간보다 작다. min_sup이 4이면, FP-Growth2는 FP-Growth 방법과 비교하여 49 %의 향상을 보였다. 이 백분율은 min_sup이 20이 될 때까지 min_sup의 값이 증가 할 때 점차적으로 감소한다. 특히 min_sup이 작은 모든 경우에 FP-Growth2는 FP-Grwoth 방법보다 우수하다.

또한 FP-Growth+는 min_sup이 2에서 8일 때 FP-growth* 및 CT-PRO 에 대해서 매우 양호하다. min_sup이 8보다 큰 경우 FP-Growth+는 동일한 실행을 갖는다 FP-Growth* 및 CT-PRO 알고리즘과 같다.그림 5에서는 FP-Growth2가 최고의 실행 시간을 보였다. 모든 min_sup 경우의 실행 시간 개선은 FP-Growth 방법보다 7 % -10 %이다.

그림 6에서도 FP-growth2 는 min_sup이 4일 때 FP-Growth과 비교하여 CPU 시간을 최대 18 % 양호하였다. min_sup이 8을 초과하면 FP-Growth 및 FP-Growth+ 와 비슷하다. CT-PRO와는 매우 양호하고 FP-Growth*은 다른 접근 방식에 비해 최고의 실행 시간을 보였다.

메모리 사용량에 대한 비교 결과는 그림 7 ~그림 9에 나타나있다. FP-Growth2는 특히 min_sup이 작은 경우에 특히 다른 알고리즘보다 메모리 소비를 감소시키는데 매우 효과적이다. 또한 데이터 세트의 크기가 커지면 메모리는 증가하기 마련이다. 테스트 된 모든 데이터 세트에 대해 FP-Growth2가 모든 min_sup에 대해 더 적은 메모리를 소비하였다. min_sup이 더 높으면 min_sup이 증가 할 때 생성되는 조건부 FP-Tree의 수가 점차 감소하기 때문에 FP-Growth는 메모리를 덜 소모하였다. 이는 FP-Growth의 주 메모리 사용이 min_sup을 증가시킴으로써 점진적으로 감소함을 의미한다.

FP-Growth+ 방법은 FP-Tree를 순회하는 배열을 사용한다. 어레이 구조는 내용을 여러 번 덮어 씀으로써 추가 메모리 공간을 할당하지 않고 자주 재사용 할 수 있다. 또한 FP-Growth2는 FP-Tree를 생성하지 않고 Look-Up테이블을 접근하는 방법으로 탐색하기 때문에 기 때문에 FP-Growth+ 방법보다 빠르다. 데이터 세트 T10I4D100K의 경우에 FP-Growth2는 메모리 소비를 30 %에서 80 %로 줄일 수 있었다. 그림 9는 FP-Growth2가 가장 우수함을 보여준다. FP-growth2 방법은 min_sup이 1300일 때 chess 데이터 세트의 경우 80 %까지 줄였다. retail 데이터 세트에서 min_sup이 낮을 때 가장 양호하다. 이는 FP-Growth2가 특히 min_sup이 낮을 때 데이터 집합 크기를 늘려 더 많은 메모리를 절약함을 나타낸다.

V. 결 론

FP-Tree의 생성과 순회와 탐색에 의존하지 않고 Look-Up과 FP-Array 배열의 접근을 통하여 메모리의 효율성을 지향하는 알고리즘을 제안하였다. 제안된 방법은 배열 기반의 새로운 데이터 구조를 사용하여 조건부 FP-Tree의 많은 후보군을 만들지 않고 빈발 항목을 추출한다. 실험을 FIMI 저장소 웹 사이트의 많은 합성 및 실제 데이터 세트에 적용한 결과, 반복적으로 다수의 조건부 FP-Tree를 생성하는 대신 Look-Up Array를 순회하기 위한 배열을 사용하기 때문에 배열의 접근성과 메모리의 사용량을 고려할 때 실행 시간과 메모리 사용을 줄이기 위한 대안으로 효과적임을 보였다.

References

[1] Agrawal, R. and R. Srikant (1994). “Fast algorithms for mining association rule”. In Proceedings of the 20th International Conference on Very Large Data Bases, Morgan Kaufmann, Santiago, Chile, pp.487-499. https://doi.org/10.4018/978-1-59904-849-9.ch012

[2] Bashir, S. and A. Baig (2006). Ramp: High Performance Frequent Itemset Mining with Efficient Bit-vector Projection Technique. Advances in Knowledge Discovery and Data Mining, 3918, pp.504-50. https://doi.org/10.1007/11731139_59

[3] Grahne, G. and J. Zhu (2003). Efficiently Using Prefix-trees in Mining Frequent Itemsets. In Proceeding of the ICDM'03 international workshop on frequent itemset mining implementations (FIMI'03), Melbourne, FL, pp.123-132.

https://doi.org/10.5120/8767-2691

[4] Han, J., J. Pei and Y. Yin (2000). Mining frequent patterns without candidate generation. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. Dallas, TX: ACM Press, pp.1-12. https://doi.org/10.1109/fskd.2007.402

[5] Schlegel, B., R. Gemulla and W. Lehner (2011). Memory-Efficient Frequent-Itemset Mining", EDBT/ICDT '11 Proceedings of the 14th International Conference on Extending Database Technology, New York, NY, USA, pp.461-472. https://doi.org/10.1145/1951365.1951420

[6] Myung-Jae Lim, Jeong-Lae Ki, "Design And Implementation of a Speech Recognition Interview Model based-on Opinion Mining Algorithm", IIBC, VOL. 12 No. 1, February 2012, pp.225~230. https://doi.org/10.1109/iscid.2014.173

[7] Salah Alghyaline, Jun-Wei Hsieh, and Jim Z. C. Lai, "Efficiently Mining Frequent Itemsets In Transactional Databases", Vol. 24, No. 2, pp. 184-191 2016. https://doi.org/10.6119/jmst-015-0709-1

[8] SunYoung Kim, YuiSun Lee, "A Case Study on Big Data Analysis of Performing Arts Consumer for Audience Development", Vol..18, No.12, 2017, pp.286-299. http://dx.doi.org/10.5762/KAIS.2017.18.12.286

[9] SukJa Kim, "The Impact of the Motives of College Students for Choosing Majors on Career Decision-Making - Using Self-efficacy as a Mediation Effect", Vol. 11 No. 3, 2018, pp.221-228. https://doi.org/10.15703/kjc.14.4.201308.2525

[10] Y. S. Im, E. Y. Kang, "MPEG-2 Video Watermarking in Quantized DCT Domain," The Journal of The Institute of Internet, Broadcasting and Communication(JIIBC), Vol. 11, No. 1, pp. 81-86, 2011. https://doi.org/10.1109/tip.2006.873476

[11] I. Jeon, S. Kang, H. Yang, "Development of Security Quality Evaluate Basis and Measurement of Intrusion Prevention System," Journal of the Korea Academia-Industrial cooperation Society (JKAIS), Vol. 11, No. 1, pp. 81-86, 2010. https://doi.org/10.5762/kais.2010.11.4.1449

[12] J. S. Oh, B, S. Lee, "A Study for Lifespan Prediction of Expansion by Temperature Status,", The Journal of KISTI, Vol. 19, No. 10, pp. 424-429637, 2018. http://dx.doi.org/10.5762/KAIS.2018.19.10.424

저자 소개

박 인 규(정회원)

- 1987년 연세대학교 공학석사
- 1997년 원광대학교 공학박사
- 1997~현재 중부대학교 컴퓨터학과 교수
- 관심분야 : 소프트 컴퓨팅, 러프집합, 퍼지집합, 신경회로망

※ "이 논문은 2018년도 중부대학교 학술연구비 지원에 의하여 이루어진 것임"