시간에 따른 가변성을 고려한 상대적인 빈발항목 탐색방법

박태수*, 이주홍*, 박선*
*인하대학교 컴퓨터정보공학과
e-mail:{taesu, sunpark}@datamining.inha.ac.kr, juhong@inha.ac.kr

Search Method of the time sensitive frequent itemsets

Tae-Su Park*, Ju-Hong Lee*, Sun Park*
*Dept. of Computer Science & Information Engineering, Inha University

요 약

최근 유비쿼터스 컴퓨팅 및 인터넷 서비스에 대한 관심이 증대되면서, 대용량의 데이터에 내재되어 있는 정보를 빠른 시간 내에 처리하여 새로운 지식을 창출하려는 요구가 증가하고 있다. 데이터 마이닝 기법을 이용하여 데이터 스트림에서 빈발항목을 탐색하는 기존의 연구는 시간을 고려하지 않고 단순히 집계를 통하여 빈발항목을 탐색하기 때문에 정확성을 보장하지 못한다. 따라서 본 논문에서는 데이터 스트림에서 시간적 측면을 고려하여 상대적인 빈발항목을 탐색하기 위한 새로운 알고리즘을 제안하고자 한다. 논문에서 제안하는 알고리즘의 성능은 다양한 실험을 통해서 검증된다.

1. 서론

최근 들어 인터넷의 발달 및 유비쿼터스 환경으로 인하여 주어진 시간 내에 처리해야할 데이터의 양이 방대해지고 있다. 또한, 주식 시장 정보나 웹 서비스, 센서 데이터, 네트워크 보안등 많은 응용분야에서 대용량의 데이터가 발생되고 있으며, 이러한 방대한 양의 데이터에서 가치있는 정보를 추출하기 위한 많은 노력들이 여러 분야에 걸쳐 이루어지고 있다. 데이터 마이닝 기법을 통한 데이터 스트림에서의 가치 있는 정보 추출은 주요 연구 분야 중 하나이다.

데이터 스트림은 매우 빠른 시간 내에 지속적으로 데이터가 증가되는 특성을 가지고 있다. 따라서 빈발항목이 아니었던 단위 항목들이 시간의 흐름에 따라 빈발항목으로 변경될 수 있는 가변성이 존재하기 때문에 단위 항목의 빈발도수를 동적으로 갱신하고 저장해야 한다. 이러한 이유로 기존의 데이터 마이닝 기법을 데이터 스트림에 적용하는 것은 적합하지 않다. 현재 데이터 스트림에서 빈발항목을 찾는 새로운 알고리즘들이 연구되고 있다.[2][3][4]

하지만 데이터 스트림에서의 빈발항목 탐색 방법들은 시간을 고려하지 않고, 단순히 빈발항목의 집계를 통해 빈발항목을 탐색하거나 또는 일정한 크기의 슬라이딩 윈도우를 임의로 설정하여 그 시간, 즉 구간에 국한된 빈발항목을 탐색하고 있으며, 시간이 흐름에 따라 총체적으로 빈발항목을 집계하기 때문에 현 시점에서 빈발항목의 변화 및 시간이 흐름에 따른 빈발항목의 변화를 탐색하지 못하고 지나쳐 빈발항목 탐색의 정확도를 보장하지 못한다.

본 논문에서는 이러한 문제점을 개선하기 위해서 데이터 스트림에서 시간적 요소를 고려하여 상대적인 빈발항목을 효율적으로 탐색하기 위한 새로운 마이닝 기법을 제안한다.

논문의 구성은 다음과 같다. 제 2장에서는 기존의 관련 연구들을 기술하고, 제 3장에서는 제안하는 빈발항목의 탐색 방법에 대해 상세히 기술한다. 제 4장에서는 다양한 실험을 통하여 제안된 방법의 성능을 평가하며 제 5장에서는 결론 및 향후 개선해야 할 과제에 대하여 기술한다.

2. 관련 연구

기존의 데이터 마이닝 기법에서의 빈발항목 탐색은 단순히 데이터베이스의 트랜잭션을 탐색하여 사전에 미리 정의된 최소지지도 이상의 지지도를 가지는 최대 항목들

1)본 연구는 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음

을 탐색하는 것이다. 이전에 연구된 빈발항목에 대한 알고리즘들은 대부분 Apriori원칙에 기반을 두고 있다[1]. 이 원칙은 빈발항목의 모든 부분집합은 반드시 빈발항목이었어야 한다는 것이다.

분할-정복기법을 사용하는 FP-growth는 후보 집합을 생성하지 않는다. FP-growth는 길거나 짧은 빈발 항목을 마이닝하는데 매우 효율적이고, 확장성을 가지며 Apriori 알고리즘보다 속도 측면에서 한 차원 앞선다는 것을 보여주고 있다. 하지만 위에서 기술된 두 기법 모두 데이터 집합을 다중 탐색해야하며 새로운 트랜잭션이 발생하였을 때 전체를 재탐색해야 한다. 또한 데이터 집합이 지속적으로 빠르게 증가하면 가용 메모리의 한정성으로 인하여 성능이 낮아진다.

최근 들어 저장장치의 발전과 네트워크의 발전으로 지속적으로 대량의 데이터가 생성이 되고 있으며 이 데이터 스트림에서의 빈발항목 탐색방법들은 다양하게 연구되고 있다.[2][3][4]

Lossy Counting 알고리즘은 최소 지지도와 최대 허용오차 조건이 주어졌을 때 데이터 스트림에서 발생한 빈발항목들의 집합을 찾는다[2]. 이 알고리즘은 시간을 고려하지 않고 단순히 빈발항목을 탐색하는데 중점을 두고 있다.

한정적인 가용 메모리 공간을 사용하는 슬라이딩 윈도우에 대한 빈발항목 탐색뿐만 아니라 빈발항목에 근접한 항목들까지 탐색하는 Moment 알고리즘은 CET (Closed enumeration tree) 라는 prefix tree와 유사한 트리 구조를 사용한다[4]. CET에서는 최대 빈발항목과 빈발항목, 근접 빈발항목을 저장하기 때문에 시간이 흐름에 따른 빈발항목의 변화를 알아낼 수 있으며, 메모리를 효율적으로 관리하며 빈발항목을 탐색할 수 있다.

FP-stream알고리즘은 FP-growth를 데이터 스트림에 적절하게 변형시킨 알고리즘이다[3]. 이 알고리즘은 데이터 집합을 최소 지지도와 최대 허용오차를 통하여 크게 빈발 항목, 부분 빈발 항목, 빈발하지 않은 항목으로 구분하며 Pattern tree와 tilted time window를 사용한다. 또한 Pattern tree에 빈발항목을 저장하고 tilted time window라는 고정된 크기의 윈도우를 사용하여 현 시점까지의 빈발항목을 축적하여 최근의 빈발항목의 변화를 용이하게 파악한다. 하지만 시간이 고정되어 있어 유동적으로 빈발항목을 탐색하지 못한다.

3. 데이터 스트림에서 빈발항목 탐색

3.1 상대적인 빈발항목 탐색

데이터 스트림에서 $S_N = \{T_1, T_2, T_3, ..., T_N\}$는 현재까지 발생한 트랜잭션의 집합, 즉 전제 데이터 집합을 의미하며 현재의 트랜잭션은 $T_t = \{I_1, I_2, I_3, ..., I_n\}, (t = 1, ..., n)$로 표현되고 각각의 트랜잭션은 TID라고 하는 고유한 식별인자를 가진다. 그리고 트랜잭션의 구성요소인 단위 항목에 대한 집합은 $I = \{i_1, i_2, i_3, ..., i_n\}$로 나타낸다.

본 논문에서는 FP-stream[3]알고리즘에서와 같이 사전에 사용자에 의해 정의되는 최소 지지도($S_{min} \in (0,1)$)와 최대 지지도 오차($e \in (0, S_{\min})$)를 이용하여 단일 항목을 빈발항목, 부분 빈발항목, 빈발하지 않은 항목으로 구분하였다. 출현 빈도가 최소 지지도 이상의 값을 가질 경우는 빈발항목으로 간주하고 최소 지지도보다는 작지만 최대 지지도 오차 이상의 값을 가질 경우는 부분 빈발항목이라고 정의한다. 최대 지지도 오차보다 작은 경우는 빈발하지 않은 항목으로 간주하여 처리하지 않는다.

하지만 기존의 연구에서는 빈발항목과 부분 빈발항목에 대해 단지 집계 값의 비교를 통해서 빈발항목을 탐색한다. 따라서 시간에 민감한 상대적인 빈발항목을 간과하고 지나칠 수 있다. 즉 현재까지 집계된 값이 현재의 빈발항목보다는 작지만 상대적인 출현빈도가 현재의 빈발항목에 비해 상대적으로 클 경우의 항목에 대한 고려가 필요한 것이다. 따라서 본 논문에서는 다음과 같이 상대적인 빈발항목에 대하여 정의한다.

상대 빈발항목 집합 R_t은 현재의 빈발항목의 출현빈도보다 출현빈도가 작지만 상대적인 출현빈도가 현재 빈발항목의 상대적인 출현빈도보다 큰 항목들의 집합으로 정의된다. 여기서 f는 현재의 빈발 항목이다.

$$R_t = \{ x | F(f) > F(x), E_m(x) > E_m(f) \}$$

상대적인 출현빈도란 유동적인 트랜잭션의 개수 m을 출현 빈도 간격에 대한 차의 합으로 나눈 것이며, 상대적인 빈발항목을 탐색하기 위한 기준이 된다.

$$E_m(x) = \frac{m}{C_m(x)}$$

또한, 출현빈도는 지속적으로 입력되는 트랜잭션에 대해서 항목이 출현했는지 출현하지 않았는지를 파악하여 집계한 값이다.

$$A_t(x) = \begin{cases} 1 & x \in T_t \\ 0 & otherwise \end{cases}$$

$$F(x) = \sum_{i=1}^{t} A_i(x)$$

그리고 출현 빈도 간격에 대한 차의 합을 구하는 것은 상대적인 빈발항목이 발생하는 시점을 파악하기 위한 것으로 현재 출현한 시점과 이전에 출현했던 시점 사이에 대한 시간차의 합으로 정의된다. 여기서 y_t는 항목 x를 포함하는 트랜잭션 T_t가 발생한 시점을 의미한다.

$$C_m(x) = y_t - y_{t-m+1}$$

위의 내용을 바탕으로 상대적인 빈발항목에 대하여 살펴보도록 하겠다. (그림 1)은 상대적인 빈발항목에 대한

개념도이다. 여기서 각각의 I_1, I_2, I_3는 최소지지도 이상의 값을 가지는 빈발항목과 최소지지도 보다는 작지만 최대 지지도 오차보다는 큰 부분 빈발항목을 나타내고, 각각의 화살표의 점은 출현 빈도를 나타낸 것이다.

<표 1> 상대적인 빈발항목 탐색을 위한 요소

기 호	의 미
S_{min}	최소 지지도
e	최대 지지도 오차
f	빈발항목
R_t	상대 빈발항목
$A_t(x)$	항목 x에 대한 출현 판단 함수
$F(x)$	항목 x의 출현빈도
$C(x)$	항목 x에 대한 출현 간격의 총합
$E_m(x)$	항목 x의 상대적인 출현빈도

또한, T_t는 현재까지의 트랜잭션을 의미한다. 첫 번째 줄의 I_1을 살펴보면 현재까지의 출현빈도가 28로 가장 높은 것을 알 수 있다. 즉 빈발항목을 의미한다. 그리고 두 번째 줄과 세 번째 줄은 부분 빈발항목을 의미한다. (그림 1)을 살펴보면 첫 번째 항목 즉 빈발항목에서는 초반에 출현빈도가 높고 중간으로 갈수록 출현간격이 점점 벌어지는 것을 알 수 있다. 하지만 두 번째 줄은 중간 부분에서 급격하게 출현빈도가 높아진 것을 볼 수 있다. 그리고 상대적으로 첫 번째 항목보다 더욱 빈번하게 출현하고 출현간격 또한 첫 번째 항목 보다 좁기 때문에 중간 부분에서는 두 번째 항목을 빈발항목으로 지정해야 정확한 것이라고 볼 수 있다. 즉 두 번째 항목이 중간부분에서는 첫 번째 항목보다는 상대적인 빈발항목이 되는 것이다.

여기서 중요한 요점은 상대적인 빈발항목으로 변경되는 시점에 대한 기준을 찾는 것이다. 그래서 본 논문에서는 다음과 같은 상대적인 빈발항목에 대한 기준을 제시한다.

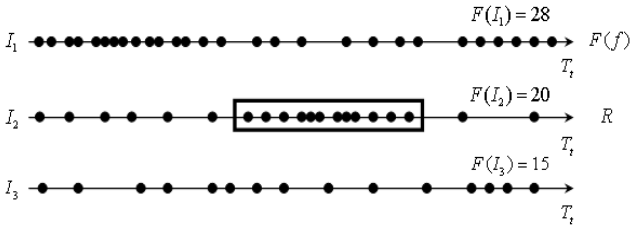
1. 상대적인 출현빈도가 빈발항목에 대한 상대적인 출현빈도보다 커야한다.

$$E_m(f) < E_m(x) \text{ and } E_m(f) > 0$$

2. 상대적인 빈발항목의 출현 시점은 바로 이전의 상대적인 출현빈도의 값에서 현재의 상대적인 출현빈도의 값을 뺀 결과가 0보다 작아지는 시점부터 0보다 커지는 시점까지이다.

$$E_m(z) - E_m(x) \leq 0 \qquad (z \in T_{t-1})$$
$$until\ E_m(z) - E_m(x) > 0$$

즉, 빈발항목에 대하여 상대적인 출현빈도가 높아지고 출현간격이 빈발항목에 비해 급격히 좁아졌을 때를 의미



(그림 1) 상대적인 빈발항목 개념도

하는 것이다. 본 논문에서는 수치를 일반화시키고 계산을 간편하게 하기 위하여 상대적인 출현빈도를 계산할 경우 소수점 첫째 자리까지 만을 고려하였다.

3.2 FP-Tree를 이용한 저장기법

본 절에서는 모든 빈발항목과 상대적인 빈발항목들을 메모리에서 효율적으로 유지, 관리하는 prefix-tree구조의 FP-Tree를 이용한 저장기법에 대하여 설명하도록 하겠다.

데이터 스트림에서 데이터는 무한집합이라고 간주한다. 그렇기 때문에 모든 항목들을 저장하는 것은 사실상 불가능하다. 따라서 FP-Tree에서는 빈발항목과 상대적인 빈발항목을 효율적으로 유지, 관리하기 위하여 (items, 출현빈도, TID)의 3가지 정보만을 저장한다.

FP-Tree를 이용한 저장기법은 크게 4단계로 구성 된다. 트랜잭션이 추가될 때마다 4단계를 반복적으로 수행하게 된다.

제 1단계는 데이터 스트림의 트랜잭션을 탐색하여 각 항목에 대한 출현 빈도를 갱신하는 단계로 새로운 트랜잭션이 추가되면 전체 데이터집합 $|S_N|$의 크기는 1씩 증가된다. 그리고 새로 추가된 트랜잭션에서 출현한 항목들이 FP-Tree에 해당하는 노드에 존재하면 출현 빈도와 TID 값을 갱신한다.

제 2단계는 부분 빈발항목이 추가되는 단계로 새롭게 추가된 트랜잭션에 출현하는 항목들이 FP-Tree에 존재하는 노드들 중에 없는 경우, 최소 지지도보다는 작지만 최대 지지도 오차보다는 큰 값을 가지는 항목을 부분 빈발항목으로 구분하여 FP-Tree의 노드로 삽입한다. 여기서 최대 지지도 오차보다 작은 항목들은 빈발항목이 될 가능성이 희박하기 때문에 제거된다. 그렇기 때문에 메모리 사용공간을 줄이고 FP-Tree의 노드에 삽입하는 추가적인 작업이 없으므로 수행시간이 짧다.

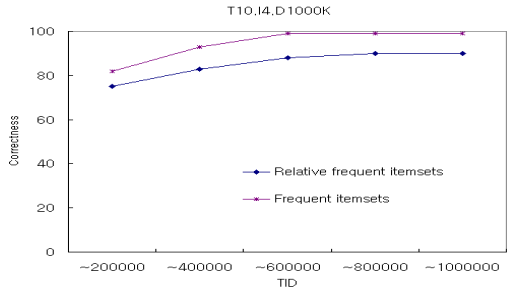
제 3단계는 현재의 빈발항목을 찾는 단계로 사용자의 요구에 의해 현재까지 총 빈발도수가 가장 크고 최소 지지도($S_{\min}$)이상의 지지도를 갖는 항목의 (items, 출현 빈도, TID) 정보를 출력해 준다.

제 4단계는 상대적인 빈발항목을 찾는 단계로 빈발항목의 상대적인 출현빈도가 커졌을 때, 즉 출현 간격이 좁아졌을 경우에 현재의 빈발항목보다 상대적으로 빈번하게 출현하는 항목을 찾는 것이다.

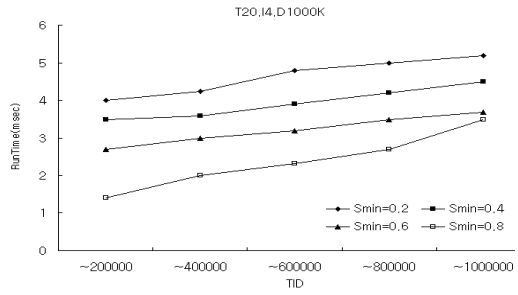
4. 실험 결과

본 절에서는 논문에서 제안된 상대적인 빈발항목과 FP-Tree에 대한 성능을 다양한 실험을 통하여 검증한다. 데이터 집합은 [1]에서 제안된 데이터 생성 방법에 따라 T10.I4.D1000K, T10.I4.D100K, T15.I6.D1000K의 데이터 집합을 생성하여 사용하였다. 각 데이터 집합에서 T는 트랜잭션의 평균적인 길이를 의미하며, I는 잠재적인 최대 빈발 항목의 평균적인 길이를 의미한다. 또한 D는 데이터 집합에 대한 트랜잭션의 총수를 의미한다.

실험은 크게 3가지 부분으로 나누어져 실행된다. 첫 번째는 빈발항목 및 상대적인 빈발항목 탐색의 정확도에 대한 검증이다. 두 번째는 빈발항목과 상대적인 빈발항목을 탐색하는 수행 시간에 대한 검증이다. 마지막으로 세 번째는 빈발항목과 부분 빈발항목을 관리하는 FP-Tree에 대한 메모리 사용량에 대한 검증한다.



(그림 2) 빈발항목의 정확도



(그림 3) 최소지지도에 따른 평균 수행시간

(그림 2)는 T10.I4.D1000K 데이터 집합을 이용한 상대적인 빈발항목과 빈발항목에 대한 정확도를 보여준다. 빈발항목이 상대적인 빈발항목보다 다소 높은 정확도를 보여주고 있다. 그 이유는 상대적인 빈발항목은 빈발항목보다 시간의 영향을 많이 받아 가변성이 크므로 빈발항목보다 탐색하는 것이 더욱 어렵기 때문이다.

(그림 3)은 최소지지도를 다양하게 변화시켰을 때 각 데이터 집합에 대한 평균 수행시간을 비교한 것이다. 평균 수행시간은 새로운 트랜잭션이 추가되었을 때 빈발항목 및 상대적인 빈발항목을 탐색하는데 소요되는 평균적인 시간을 의미한다. 최소지지도가 낮을수록 평균 수행시간이 커진다. 그 이유는 최소지지도가 낮을수록 빈발항목과 부분빈발항목에 대한 허용범위가 넓어져서 빈발항목을 탐색하기 위한 비교 횟수가 증가되기 때문이다. 각 데이터 집합에 따른 FP-Tree에서의 메모리 사용량에 대한 차이는 크지 않다. 그 이유는 FP-Tree알고리즘에서 최소지지도와, 최대 지지도 오차를 이용하여 빈발항목과 부분 빈발항목만을 관리함으로 메모리의 사용량이 작기 때문이다.

5. 결 론

데이터 스트림에서 가장 기본적인 문제는 스트림에서 발생하는 빈발적인 항목들을 탐색하는 것이다. 하지만 데이터 스트림은 시간에 따른 빈발항목의 가변성이 존재한다. 또한, 방대한 양의 데이터가 발생되기 때문에 데이터 스트림 내의 모든 항목들을 저장하는 것은 불가능하다. 본 논문에서는 위와 같은 문제를 해결하기 위하여 상대적인 빈발항목이라는 개념과 FP-Tree를 이용한 저장기법을 제안하였다.

제안된 방법에서는 각 단일 항목의 빈발도수와 빈발 간격에 따른 상대적인 빈발도수를 계산하고, 시간이 경과함에 따른 각 단일 항목의 상대적인 빈발도수를 비교하여 간과하고 지나칠 수 있는 상대적인 빈발항목을 탐색한다. 또한, 한정적인 메모리 공간을 효율적으로 사용하기 위하여, FP-Tree를 이용하여 빈발항목과 부분 빈발항목에 대한 (items, 출현 빈도, TID)의 3가지 정보만을 저장한다. 그러므로 시간에 민감한 상대적인 빈발항목을 탐색할 수 있으며, 빈발항목 및 상대적인 빈발항목 탐색에 대한 정확도도 향상시킬 수 있고, 한정적인 메모리를 효율적으로 사용할 수 있다.

참고문헌

[1] Rakesh Agrawal,, & Ramakrishnan Srikant, Fast Algorithms for Mining Association Rules. Proc. 20th Int. Conf. Very Large Data Bases, 487-499, 1994

[2] Manku, G. S., & Motwani, R., Approximate frequency counts over data streams, In Proc. of the 28th Conference on Very Large Databases, 2002

[3] Giannella, C., Han, J., Pei, J., Yan, X., & Yu, P. S.. Mining Frequent Patterns in Data Streams at Multiple Time Granularities, Next Generation Data Mining, AAAI/MIT, 2003

[4] Yun, Chi., Haixun, Wang., Philip, S., Yu, Richard, R., Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window .Proceedings of the IEEE International Conference on Data Mining, 2004