퀀터티가 있는 순차 패턴을 찾는 깊이 우선 탐색 알고리즘*

김철연° 심규석
서울대 전기컴퓨터공학부 서울대 전기컴퓨터공학부
cykim@kdd.snu.ac.kr shim@ee.snu.ac.kr

An Efficient Depth First Algorithm for Mining Sequential Patterns with Quantities

Chulyun Kim° Kyuseok Shim
School of EECS, SNU School of EECS, SNU

요 약

순차 패턴을 찾는 것은 데이타 마이닝 응용분야에서 중요한 문제이다. 기존의 순차 패턴 마이닝 알고리즘들은 아이템으로만 이루어진 순차 패턴만을 취급하였으나 Apriori-QSP에서는 새롭게 퀀터티 정보에 대한 처리의 개념을 도입하였다. 전체 순차 패턴을 찾는 알고리즘들은 너비 우선 탐색과 깊이 우선 탐색 기법으로 분류할 수 있는데, 이러한 분류에서 Apriori-QSP 알고리즘은 너비 우선 탐색 기법으로 분류할 수 있다.

본 논문에서는 퀀터티 정보를 처리하는 깊이 우선 탐색 기법을 제안하였다. Apriori-QSP에서 제안되었던 후보패턴 생성에 대한 필터링과 샘플링 기법을 깊이 우선 탐색의 탐색 기법으로 적용하였으며, 다양한 실험결과들이 깊이 우선 탐색에서도 이러한 기법이 효율적임을 보여 주고 있다. 또한 길이가 긴 순차 패턴 마이닝의 경우 너비우선 탐색에 비해 향상된 성능을 보임을 확인하였다.

제 1 절 서론

1.1 배경 및 문제점

데이타 마이닝 분야 중 하나인 순차 패턴(sequential pattern)을 찾는 알고리즘은 현재까지 크게 두 가지로 분류할 수 있다. IBM 연구소에서 개발한 Apriori 알고리즘[1][2]으로 대표되는 너비 우선 탐색 기법과 캐나다의 Simon Fraser대학에서 개발한 PrefixSpan 알고리즘[3]으로 대표되는 깊이 우선 탐색 기법이다. 최초로 퀀터티 정보가 있는 순차패턴을 찾는 알고리즘인 Apriori-QSP[4]는 너비 우선 탐색기법으로 분류된다.

너비 우선 탐색기법은 순차 패턴을 검색하기 위해 많은 후보패턴들을 생성하여야 하기 때문에 이를 위해 많은 메모리를 사용하여야 하며, 이러한 후보패턴의 검사로 인한 성능저하가 가장 큰 문제점으로 지적된다[3] 깊이 우선 탐색은 후보패턴을 생성하지 않기 때문에 많은 양의 메모리나 후보패턴에 대한 검사가 필요로 하지 않으므로, 특히 길이가 긴 순차패턴의 탐색의 경우에 너비 우선 탐색에 비해 좋은 성능을 보인다.[3] 따라서 깊이 우선 탐색 기법의 특성을 가지는 퀀터티 정보를 처리하는 효율적인 알고리즘이 필요하기에 연구를 시작하게 되었다.

제 2 절 초보적인 접근법

2.1 확장된 문제 정의

이 장에서는 기존 문제를 확장 하여 퀀터티를 다루는 문제를 정의한다. 퀀터티를 추가함으로써 기존 문제 정의와 달라지는 부분은 아이템이 아이템과 퀀터티의 쌍(pair) 으로 확장된다는 점이다. $I = \{i_1, i_2, ..., i_m\}$는 모든 아이템의 집합(set of all items)이다. $i \in I$인 아이템 i와 정수의 집합 N[1] 의 원소인 정수 n과의 쌍인 $[i,n]$을 **확장아이템**(extended item)이라고 한다. 이때 i를 **확장아이템의 아이템 부분**, n을 확장아이템의 **퀀터티 부분**(혹은 **정수 부분**)이라고 한다. 확장아이템들의 집합인 EI는 $\{[i,n]|i \in I \wedge n \in N_i\}$로 정의 한다. **확장아이템집합**(extended itemset)은 EI의 부분집합(subset)중에 아이템 부분이 같은 확장아이템이 동시에 존재하지 않는 집합을 말한다. 이러한 확장아이템집합 eis과 EI와의 관계를 $eis \subseteq_e EI$로 정의하자. 확장아이템집합간의 부분집합(subset), 포함집합(superset)을 정의하면 확장아이템집합 $eis_1 = \{a_1, a_2, ..., a_n\}$ 와 확장아이템집합 $eis_2 = \{b_1, b_2, ..., b_m\}$ 이 있을 때, $1 \le p \le n$인 eis_1의 모든 확장아이템 a_p의 (1)a_p의 아이템 부분인 i_{a_p}와 같은 아이템을 갖는 $1 \le q \le m$인 b_q가 존재하며, (2)a_p의 퀀터티 부분인 n_{a_p}가 b_q의 퀀터티 부분인 n_{b_q}보다 작거나 같은 경우, $eis_1 \subseteq eis_2$라고 하고 eis_1이 eis_2의 부분집합(subset), eis_2가 eis_1의 포함집합(superset)이라고 한다.

시퀀스(sequence)는 $\langle s_1 s_2 ... s_l \rangle$로 나타낼 수 있는데, 여기서 $s_j \subseteq_e EI$, 즉 모든 $1 \le j \le l$를 만족하는 j에 해당되는 s_j는 확장아이템집합이 된다.

2.2 PrefixSpan 알고리즘의 변형

PrefixSpan 알고리즘에서 확장된 문제를 해결하기 위해서는, 길이가 1 또는 2인 순차 패턴을 찾는 과정과 시퀀스 데이터베이스에 대한 프로젝션 과정을 확장해야 할 것이다. 먼저 순차 패턴을 찾는 과정에서 확장해야 할 점은 단순히 아

*본 연구는 대학 IT연구센터 육성지원사업의 연구결과로 수행되었음

[1]실수로 확장될 수도 있으나, 문제정의에서는 정수의 집합으로 한다.

이템이 같은 지를 살피는 과정 대신 아이템과 아이템의 퀀터티까지도 확장된 문제 정의의 포함관계를 사용하여 판별하면 된다. 그 외의 과정은 기존의 과정과 동일하다. 다음으로 프로젝션 과정에서도 순차 패턴을 찾는 과정과 동일하게 확장된 문제 정의의 포함관계를 사용하여 프로젝션을 하면 된다. 앞으로 이 알고리즘을 초보적인(naive) 확장 PrefixSpan 알고리즘이라고 부르기로 한다.

제 3 절 통합된 접근법

3.1 시퀀스 여과과정

퀀터티가 있는 순차 패턴 마이닝의 결과들을 관찰하면 퀀터티가 있는 순차 패턴 마이닝을 하기 전에, 퀀터티가 없는 결과를 먼저 구한 후 그 결과를 이용할 수 있게 된다. 이렇게 퀀터티 없이 얻은 결과를 이용하는 것을 **여과과정**(filtered counting)이라고 하기로 한다.

길이가 1인 순차 패턴 단위로 재귀적 호출이 일어나는 Level-by-Level의 경우를 생각해 보자. 순차 패턴 f의 아이템을 $f.i$로 나타내기로 하자. 퀀터티 없는 순차 패턴의 집합을 F라고 하고, 퀀터티 없는 순차 패턴 중 1개 이상의 순차 패턴을 재귀적 호출의 결과로 가지는 집합을 G라고 하고, 아이템 i와 관련된 퀀터티를 L_i개라고 하면, 여과과정이 없는 경우는 해당 Level에서 $\sum_{f\in F} L_{f.i}$번의 재귀적 호출을 통해 결과를 얻게 되지만, 여과과정을 넣은 경우는 여과과정에서 $|F|$번의 재귀적 호출, 그 이후의 과정에서 $\sum_{f\in G} L_{f.i} - |G|$ 번의 재귀적 호출을 하게 되어, 총 $|F| + \sum_{f\in G} L_{f.i} - |G|$ 번의 재귀적 호출을 하게 된다. 이 경우에 $|G| \leq |F|$이기 때문에, 여과과정이 있는 경우의 재귀적 호출의 수는 여과과정을 사용하지 않은 경우의 재귀적 호출의 수보다 항상 작거나 같다.

정리 1 여과과정을 사용한 경우에 불리는 재귀적 호출의 수는 여과과정을 사용하지 않은 경우의 재귀적 호출의 수보다 항상 작거나 같다.
증명: *[5] 참조* □

3.2 샘플링 기반 알고리즘

샘플링 기반 알고리즘은 퀀터티가 있는 순차 패턴 마이닝에서 최대의 빈번한 패턴만을 얻기 위한 방법이다. 즉, 빈번한 패턴중에서 다른 빈번한 패턴의 부분시퀀스가 되지 않는 패턴들을 얻기 위한 방법이다.

기존 방법에서는 모든 퀀터티에 대해 재귀적 호출을 불렀지만, 샘플링 기반 알고리즘에서는 퀀터티마다 일정 간격(interval)를 두고, 그 간격에 해당하는 순차 패턴에 대해서만 재귀적 호출을 부른 후, 이 중 한개 이상의 순차패턴을 재귀적 호출의 결과로 갖는 순차 패턴에 대해서만, 그 다음 단계에서 간격 사이의 순차 패턴에 대해 모두 재귀적 호출을 부르는 방법을 사용한다. 즉, 1단계에서는 듬성듬성한(sparse grained) 재귀적 호출을 통해서 결과를 얻게 되고, 1단계에서 1개 이상의 순차 패턴을 재귀적 호출의 결과로 갖는 순차 패턴 중에 다른 패턴의 진 부분시퀀스(proper subsequence[2])인 것들을 지운 뒤, 남아 있는 패턴들을 2단계에서 세밀한 재귀적 호출을 부르는데에 사용한다.

길이가 1인 순차 패턴 단위로 재귀적 호출이 일어나는 Level-by-Level의 경우를 생각해 보자. 기존의 방법을 사용할 경우는 $\sum_{f\in F} L_{f.i}$ 번의 재귀적 호출을 통해 결과를 얻게 되지만, 샘플링 기반 알고리즘을 사용하면 1단계에서 $k|F|$ 번의 재귀적 호출을 부르게 되고, 진 부분시퀀스 제거과정을 거친 후의 결과의 집합이 $G_1(0 \leq |G_1| \leq k|F|)$라면 2단계에서는 $\sum_{f\in G_1}(\frac{L_{f.i}}{k} - 1)$ 번의 재귀적 호출을 부르게 된다. 따라서 총

$$k|F| + \sum_{f\in G_1}(\frac{L_{f.i}}{k} - 1)$$

번의 재귀적 호출을 부르게 된다. 이 역시 기존의 방법에서의 재귀적 호출의 수보다는 항상 작거나 같게 된다.

정리 2 샘플링 기반 알고리즘을 사용한 경우의 재귀적 호출의 수는 샘플링 기반 알고리즘을 사용하지 않은 경우의 재귀적 호출의 수보다 항상 작거나 같다.
증명: *[5] 참조* □

3.2.1 정확성(correctness)증명

정리 3 1단계에서 1개 이상의 순차 패턴을 재귀적 호출의 결과로 갖는 순차 패턴중에 진 부분시퀀스를 지워도 최대의 빈번한 패턴을 잃지 않는다.
증명: *[5] 참조* □

하지만 1단계의 결과중에 진 부분시퀀스가 아닌 부분시퀀스의 관계에 있는 순차 패턴[3]은 지우지 말아야 한다. 그 패턴들을 이용해서 2단계에서 호출하게 되는 재귀적 호출의 결과는 최대의 패턴이 될 가능성이 있기 때문이다.

3.2.2 여과과정과 샘플링 기반 알고리즘을 같이 사용하는 경우

정리 4 여과과정과 샘플링 기반 알고리즘을 동시에 사용하는 경우 불려지는 재귀적 호출의 수는 여과과정만을 사용한 경우나 샘플링 기반 알고리즘만을 사용한 경우보다 항상 작거나 같다.
증명: *[5] 참고* □

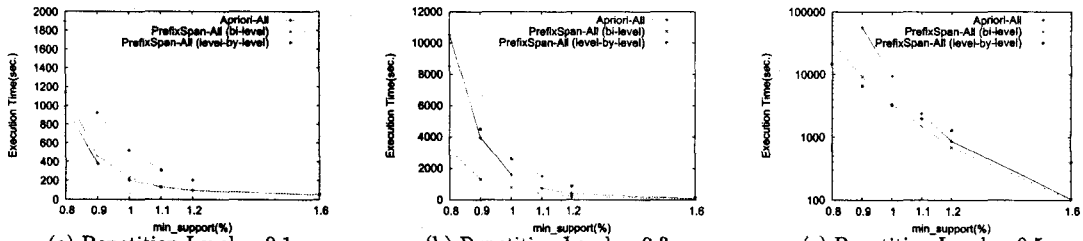
제 4 절 성능 평가

모든 실험은 LINUX 운영체제에 메모리 512MB인 Pentium 4 PC에서 실행되었다.

합성 데이타를 만드는 데에는 Apriori-QSP[4]에서와 동일한 데이터 생성기를 사용하였다. 본 논문에서 사용된 실험 데이타 들은 시퀀스당 평균 트랜잭션 수 10, 트랜잭션당 평균 아이템 수 2.5, 총 아이템 수 100,000개의 파라미터로 생성된 데이타들이다.

[2]아이템 부분은 모두 같고 퀀터티 부분은 모두 작은 부분시퀀스

[3]모든 아이템부분은 같고, 퀀터티부분도 같은 경우가 존재하는 부분시퀀스



(a) Repetition Level = 0.1 (b) Repetition Level = 0.3 (c) Repetition Level = 0.5

그림 2: Apriori-All과 Prefix-All의 수행시간 비교

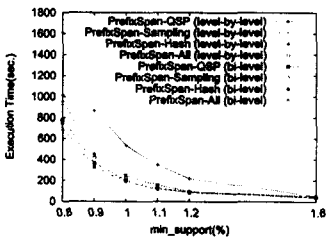


그림 1: PrefixSpan-QSP 수행시간 평가

실험에서는 아래의 알고리즘의 실행시간을 비교하였다.

- 초보적인 확장 PrefixSpan 알고리즘(PrefixSpan-QSP)
- 여과과정 기반 알고리즘(PrefixSpan-Hash)
- 샘플링 기반 알고리즘(PrefixSpan-Sampling)
- 여과과정+샘플링 기반 알고리즘(PrefixSpan-All)
- 여과과정+샘플링 기반 Apriori 알고리즘(Apriori-All)

실험 결과를 통해 본 논문이 제시하는 알고리즘이 초보적으로 확장된 알고리즘에 비해 50% 이상의 수행시간 단축을 보여 준다는 사실을 알 수 있으며, 길이가 긴 순차 패턴의 경우 Apriori-All 보다 뛰어난 성능을 보임을 확인할 수 있다.

4.1 합성 데이타 집합에 대한 결과

그림 1은 각각의 알고리즘들의 최소 지지도별 수행속도를 보여준다. 본 논문에서 제시하는 아이디어의 유무에 따른 수행시간의 차이는 최소 지지도가 낮아질 수록 더욱 커짐을 알 수 있다.

마지막으로 본 논문에서 제시한 알고리즘과 Apriori-All 알고리즘과의 수행시간에 관한 비교 결과를 살펴보자. 그림 2는 Repetition Level을 변화시키면서 알고리즘 간의 성능의 변화를 살펴본 그래프이다. Repetition Level이 커질 수록 순차 패턴의 길이가 길어지며, 순차 패턴의 길이가 길어 질 수록 점차 PrefixSpan-All이 더욱 좋은 성능을 보임을 확인할 수 있다.

제 5 절 결론

본 논문에서는 기존의 깊이 우선 순차 패턴 마이닝을 확장하여 퀀터티 정보까지도 포함한 깊이 우선 순차 패턴 마이닝 알고리즘을 제시하였다. 이러한 깊이 우선 순차 패턴 마이닝 알고리즘은 특히 길이가 긴 순차패턴에 대한 검색의 경우 너비 우선 탐색 기법에 비해 좋은 성능을 보인다.

퀀터티가 있는 순차 패턴을 찾는 깊이 우선 알고리즘으로 PrefixSpan 알고리즘의 초보적인 확장 알고리즘은 재귀적 호출의 수를 크게 늘리게 되어 결과적으로 수행 시간도 크게 증가하였다. 따라서 결과를 얻는데에 필요한 재귀적 호출의 수를 줄임으로써 수행 시간을 개선할 수 있는 방법을 제안하였다. 즉, 여과과정(filtered counting)과 샘플링 기반 알고리즘(sampling based algorithm)이라는 아이디어를 사용하여 최대의 패턴이 될 수 없는 재귀적 호출들을 미리 전지함으로써 알고리즘의 수행속도를 매우 개선하였다.

참고 문헌

[1] R. Agrawal and R. Srikant. "Mining sequential patterns," In *Proc. 1995 Int'l Conf. Data Engineering (ICDE'95)*, pages 3-14, Taipei, Taiwan, March 1995.

[2] R. Agrawal and R. Srikant. "Mining sequential patterns: Generalizations and performance improvements," In *Proc. 5th Int'l Conf. Extending Database Technology (EDBT'96)*, pages 3-17, Avignon, France, March 1996.

[3] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," In *Proc. 2001 Int'l Conf. Data Engineering (ICDE'01)*, pages 215-224, Heidelberg, Germany, April 2001.

[4] J.-H. Lim, K. Shim, C. Kim. "An Efficient Algorithm for Mining Sequential Patterns with Quantities," In *Proceedings of The 30th KISS Spring Conference*, pages 569-571, Jeju, Korea, Apr 2003.

[5] C. Kim, J.-H. Lim, R. Ng, and K. Shim. "SQUIRE: Sequential pattern mining with quantities. Technical report," Seoul national University, 2003.