서픽스 검사를 이용한 단계적 순차패턴 분할 탐사 방법

허용도[†] · 조동영[††] · 박두순[†††]

요 약

효율적인 순차패턴 마이닝을 위해서는 후보패턴의 생성 비용을 줄이고 동시에 생성된 후보패턴에 대한 탐색공간을 줄여야 한다. 그러나 이전에 개발된 알고리즘들은 이러한 문제들을 효율적으로 해결하지 못하고 있다. 특히 Apriori-like 방법들은 알고리즘은 단순하지만 많은 크기의 후보패턴 집합 생성, 대용량 데이터베이스의 반복적인 탐사 등의 문제점이 있고, PrefixSpan[2]은 단계별로 분할된 프레픽스 프로젝티드(prefix projected) 데이터베이스들을 구성하여 후보패턴의 지지도 계산을 위한 탐색공간을 줄이지만 프로젝티드 데이타베이스들의 구성비용이 크다는 문제점이 있다.

이러한 문제점들의 개선을 위해 본 논문에서는 새로운 순차패턴 마이닝 방법인 SuffixSpan(**Suffix** Checked Sequential **Pattern** mining)을 제안한다. SuffixSpan은 순차패턴 집합의 단계별 분할특성과 서픽스(suffix) 특성을 이용하여 적은 비용으로 작은 크기의 후보패턴 집합을 생성하고, 1-프레픽스 프로젝티드 데이타베이스를 구성하여 후보패턴 검사를 위한 탐색공간을 줄인다.

A Partition Mining Method of Sequential Patterns using Suffix Checking

Her Young Do[†], Cho Dong Young[††] and Park Doo Soon[†††]

ABSTRACT

For efficient sequential pattern mining, we need to reduce the cost to generate candidate patterns and searching space for the generated ones. Although Apriori-like methods like GSP[8] are simple, they have some problems such as generating of many candidate patterns and repetitive searching of a large database. PrefixSpan[2], which was proposed as an alternative of GSP, constructs the prefix projected databases which are stepwise partitioned in the mining process. It can reduce the searching space to estimate the support of candidate patterns, but the construction cost of projected databases is still high. To solve these problems, we proposed SuffixSpan(Suffix checked Sequential Pattern mining) as a new sequential pattern mining method. It generates a small size of candidate pattern sets using partition property and suffix property at a low cost and also uses 1-prefix projected databases as the searching space in order to reduce the cost of estimating the support of candidate patterns.

Key words: Sequential Pattern Mining, Apriori like method, GSP, PrefixSpan

1. 서 론

거대한 크기의 시퀀스 데이터베이스로부터 빈발한 시퀀스들을 탐색하는 순차패턴 마이닝은 데이터 마이닝의 중요한 연구영역으로 지금까지 많은 연구들이 있어 왔다[1-3,6,8]. 순차패턴 마이닝 문제는 기존 텍스트 패턴매칭 문제[5] 영역과는 달리 엄청난 용량의 데이터베이스를 문제 영역으로 가지기 때문에 데이터베이스에 내재되어 있는 모든 순차패턴들을 탐색해 내는 것은 물론 순차패턴들의 탐색과정에서 소요되는 시간과 메모리 요구량에서도 효율적이고 현실적이어야 한다.

[6,7,9]에서 처음으로 제안된 이후 순차패턴 마이닝에 대한 다양한 접근 방식의 연구들이 발표되고

접수일 : 2002년 5월 20일, 완료일 : 2002년 7월 9일
본 연구는 정보통신부의 ITRC 사업에 의해 수행되었음
[†] 건양대학교 IT 학부 부교수
[††] 전주대학교 정보기술컴퓨터공학부 부교수
[†††] 순천향대학교 정보기술공학부 교수

있다[1-3,6,8]. GSP[8]와 같은 전형적인 Apriori-like 방법들은 순차패턴 마이닝 과정에서 데이터베이스의 반복적인 탐사와 단계적인 후보패턴 생성 및 검사 방법을 사용한다. Apriori-like 방법들은 알고리즘의 단순함에도 불구하고 엄청난 크기의 후보패턴 집합 생성과 그에 따른 대용량 데이터베이스의 반복적인 탐사, 그리고 긴 순차패턴 탐사의 어려움 등의 문제점을 갖는다[2]. 최근에는 Apriori-like 방법들의 문제들을 개선하기 위한 방법들이 발표되고 있는데 [1,2,4], 특히 PrefixSpan[2]은 단계별로 분할된 프레픽스 프로젝티드(prefix projected) 데이터베이스들을 구성하여 후보패턴들의 지지도 계산을 위한 탐색공간을 줄인다. PrefixSpan의 성능은 단계별로 구성되는 프레픽스 프로젝티드 데이타베이스들의 구성비용에 의해 결정된다.

사용자에 의해 정의되는 순차패턴의 최소지지도 기준값(minimal support threshold)이 작으면 주어진 시퀀스 데이타베이스에서 얻어지는 순차패턴들의 평균 길이와 개수는 커진다. 그리고 시퀀스 데이터베이스의 순차패턴의 개수에 비해 순차패턴의 평균 길이가 커지면, GSP는 매 단계 수행되는 후보패턴들의 생성 및 탐색공간 검사비용이 크게 증가하게 되며, 특히 매 단계에서 생성된 후보패턴들에 대한 지지도 탐색공간이 시퀀스 데이터베이스 전체이기 때문에 마이닝 과정에 엄청난 시간이 요구된다. 반면 순차패턴의 개수에 비해 순차패턴의 평균 길이가 작게 되면 PrefixSpan의 매 단계에서 수행되는 분할된 프레픽스 프로젝티드 데이타베이스의 구성시간이 크게 증가하게 된다. 따라서, 효율적인 순차패턴 마이닝을 위해서는 후보패턴의 생성 비용을 줄이고, 동시에 생성된 후보패턴들에 대한 탐색공간을 줄일 수 있어야 한다.

이러한 목적을 달성하기 위해 본 논문에서는 GSP와 PrefixSpan의 접근방법을 부분적으로 결합한 새로운 순차패턴 마이닝 방법을 제안한다. 본 논문에서 제안하는 방법은 순차패턴 집합의 단계별 분할특성과 서픽스(suffix) 특성을 이용하여 적은 비용으로 작은 크기의 후보패턴 집합을 생성하고, 1-프레픽스 프로젝티드 데이타베이스를 구성하여 후보패턴들의 지지도 탐색공간을 줄인다.

본 논문의 구성은 다음과 같다. 2장은 순차패턴 마이닝 문제영역의 기본개념들을 정의하고, 본 연구와 관련있는 GSP 및 PrefixSpan의 문제점을 고찰한다. 그리고 3장은 GSP와 PrefixSpan을 결합한 접근방법을 기술하고, 4장에서는 제안된 방법의 성능요소를 GSP와 PrefixSpan과 비교한다. 5장은 본 연구의 요약과 결론을 기술한다.

2. 문제정의 및 관련연구

2.1 문제정의

이 절에서는 본 논문에서 사용되는 기본 개념들과 순차패턴 마이닝 문제를 정의한다. 본 논문에서 사용되는 개념들과 문제정의는 기본적으로 [1,2,4,7-9]와 동일하다.

$I=\{i_1,i_2,...,i_l\}$를 항목들의 집합이라 하고, I의 부분집합 X를 항목집합(itemset)이라고 한다. 시퀀스 s는 항목집합들의 순서리스트이고, 간단히 $s=<e_1e_2...e_n>$으로 표현한다. 그리고 각 $e_j(1\le j\le n)$를 시퀀스 s의 원소(element)라고 하고, 간단히 $e_j=(x_1x_2...x_m)$으로 표현한다. 여기서, 각 $x_i(1\le i\le m)$는 항목이고, 각 항목들은 사전식 순서로 정렬되는 것으로 가정한다. 특별히 $e_j=(x)$인 경우, 간단히 괄호를 생략하여 표현한다. 한 항목은 한 시퀀스에서 여러 번 표현될 수 있지만 시퀀스의 원소에서는 한번만 표현될 수 있다. 시퀀스 $s=<e_1e_2...e_m>$의 길이는 s의 모든 원소들의 항목개수의 합, 즉 $\sum_{i=1}^{m}|e_i|$ (여기서, $|e_i|$는 항목집합 e_i의 항목개수)이다. 길이 k를 갖는 시퀀스를 간단히 k-시퀀스(k-sequence)라고 한다. 시퀀스 $\alpha=<a_1a_2...a_n>$, $\beta=<b_1b_2...b_m>$가 $1\le j_1<j_2<...<j_n\le m$에 대해 $a_1\subseteq b_{j1}$, $a_2\subseteq b_{j2}$, ... , $a_n\subseteq b_{jn}$ 일 때 시퀀스 β는 시퀀스 α를 포함한다라고 하고, α를 β의 서브시퀀스(subsequence), β를 α의 수퍼시퀀스(supersequence) 라고 한다.

시퀀스 데이터베이스 S는 시퀀스 식별자 s_{id} 와 시퀀스 s 로 구성되는 튜플 $<s_{id}, s>$들의 집합이다. 어떤 시퀀스 s′가 s의 서브시퀀스이면, "s′는 $<s_{id}, s>$에 포함된다"라고 한다. 시퀀스 데이터베이스 S에서 시퀀스 s의 지지도는 S에서 s를 포함하는 튜플들의 개수이다. 그리고 사용자에 의해 정해진 S의 최소지지도 기준값 ξ에 대해, 어떤 시퀀스 s의 지지도가 ξ와 같거나 클 경우, s를 S에서의 순차패턴(sequential pattern) 또는 빈발패턴(frequent pattern)이라고 하고, 길이 k인 순차패턴을 간단히 k-순차패턴(k-sequential pattern)이라고 한다.

순차패턴 마이닝 문제는 주어진 시퀀스 데이터베이스 S와 지지도 기준값(ξ)에 대해, 시퀀스 데이터베이스 S에서 모든 순차패턴들의 집합을 찾아내는 것이다.

2.2 관련연구

이 절에서는 본 논문에서 제안하는 순차패턴 마이닝 방법과 비교되는 기존 연구들의 특성과 문제점을 고찰한다. 본 논문에서 제안하는 접근방법은 GSP와 FreeSpan[3], PrefixSpan의 접근방법들과 비교된다.

GSP는 ApioriAll[6]의 접근방법과 마찬가지로 "비빈발 시퀀스들의 슈퍼시퀀스들은 빈발하지 않다"라는 사실에 기본을 두고, 후보패턴 생성 및 검사 과정의 단계적 반복에 의해 모든 순차패턴들을 찾는다. GSP는 시간 제약조건(time constraints), 트랜잭션의 느슨한 정의(lossely defined transaction), 항목들 사이의 계층성(taxonomies) 등의 개념을 도입해 ApioriAll의 문제정의 영역을 확장하는데 공헌했지만, 매 단계 많은 크기의 후보패턴 집합생성과 그에 따른 대용량 데이터베이스의 반복적인 탐사 필요성 등의 문제점을 갖는다[1,2].

FreeSpan과 PrefixSpan은 후보패턴들의 지지도 탐색공간을 줄이기 위해 시퀀스 데이터베이스에 대한 프레픽스 프로젝션(prefix projection)을 반복적으로 수행한다. PrefixSpan은 FreeSpan의 개선된 방법이다. FreeSpan은 빈발 서브시퀀스들의 모든 발생가능성을 고려하여 시퀀스 데이터베이스를 프로젝션하고, PrefixSpan은 "모든 빈발 시퀀스들은 빈발한 프리픽스(prefix)들의 확장에 의해 발견할 수 있다"라는 사실에 기인하여 빈발한 프레픽스에 대해서만 데이타베이스 프로젝션을 수행한다. FreeSpan과 PrefixSpan의 성능은 단계별로 구성되는 프로젝티드 데이타베이스들의 구성비용에 의해 영향을 받는다.

효율적인 순차패턴 마이닝을 위해서는 후보패턴들의 생성비용과 후보패턴들의 지지도 계산을 위한 시퀀스 데이터베이스의 탐색공간을 줄여야 한다. PrefixSpan은 빈발한 프레픽스에 대한 분할된 프로젝티드 데이타베이스들을 구성해서 이 문제의 개선을 시도하지만 프로젝티드 데이타베이스들의 과다한 구성비용이 문제가 된다. GSP는 프로젝티드 데이타베이스들의 구성비용은 발생하지 않지만 후보패턴들의 지지도 계산을 위한 탐색공간이 거대한 시퀀스 데이타베이스 전체이기 때문에 그 성능에는 한계성을 갖는다. 본 연구는 PrefixSpan의 데이터베이스 프로젝션 개념을 GSP에 부분적으로 적용하고, GSP의 후보패턴 생성방법을 개선함으로써 효율적인 순차패턴 마이닝을 가능하게 한다.

3. 서픽스 검사를 이용한 순차패턴 분할 탐사

본 논문에서 제안하는 알고리즘의 기본 아이디어는 "모든 k-순차패턴들의 집합은 (k-1)-순차패턴들의 집합에 의해 분할될 수 있다"라는 순차패턴 집합의 단계별 분할특성과 "모든 k-순차패턴들의 서픽스는 (k-1)-순차패턴이다"라는 서픽스 특성을 이용하여 적은 비용으로 작은 크기의 후보패턴 집합을 생성하고, 1-프레픽스 프로젝티드 데이타베이스를 구성하여 후보패턴들의 지지도 탐색공간을 줄이는 것이다. 3.1절에서는 제안하는 알고리즘의 형식적 기술을 보이고, 3.2절에서는 예제를 가지고 제안하는 알고리즘의 순차패턴 마이닝 과정을 설명한다.

3.1 제안한 알고리즘의 형식적 접근

정의 1. 시퀀스 $s=\langle e_1e_2...e_n\rangle$에 대해서, s의 첫 번째 항목과 마지막 항목을 각각 first(s), last(s) 라고 표기하며,

1) 만일 $s_1=(x)$(즉,$|s_1|=1$)이면 first(s)=x로, $s_1=(x_1,x_2,...,x_m)$(즉,$|s_1|\geq 2$)이면 first(s)=x_1 로 정의한다.
2) 만일 $s_n=(x)$(즉,$|s_n|=1$)이면 last(s)=x로, $s_n=(x_1,x_2,...,x_m)$(즉,$|s_n|\geq 2$)이면 last(s)=x_m 로 정의한다.

I 는 항목들의 집합이다. 시퀀스 데이터베이스 S가 주어졌을 때, S에 있는 모든 순차패턴들의 집합을 F 라고 표기하고, 임의의 양수 k 에 대해서 길이가 k 인 순차패턴들의 집합을 F_k 라고 한다. 순차패턴 s가 F_1에 포함되어 있는 임의의 원소 x 에 대해서 ***first(s)*** = x 인 순차패턴들의 집합을 F_x 라고 하며, 또한 S에 있는 가장 긴 순차패턴의 길이를 max 라고 한다.

정의 2. 시퀀스 $s=\langle e_1e_2...e_n\rangle$와 $^{\forall}x\in I$에 대해서,
1) $s\wedge x = \langle e_1e_2...e_n\{x\}\rangle$라고 정의한다.

2) $s \vee x = \langle e_1 e_2 \ldots e_n \cup \{x\} \rangle$라고 정의한다. 이때 $x \notin s_n$ 이다.

정의 3. 항목집합 I의 부분집합 X에 대해서 $F_k \wedge X = \{s \wedge \{x\} | s \in F_k, x \in X\}$ 이고 $F_k \vee X = \{s \vee \{x\} | s \in F_k, x \in X\}$ 이다. X에 의한 F_k의 확장을 extension(Fk,X)으로 표기하며 extension(Fk,X) = $\{(F_k \wedge X) \cup (F_k \vee X)\}$ 으로 정의한다.

정의 4. 시퀀스 $s = \langle e_1 e_2 \ldots e_n \rangle$에 대해서.
1) 만일 $|e_1| = 1$ 이면 $\beta = \langle e_2 e_3 \ldots e_n \rangle$라고 하자
2) 만일 $|e_1| \geq 2$ 이면 $\beta = \langle e_1' e_2 \ldots e_n \rangle$라고 하자. 이때 $e_1' = e_1 - \{x_i\}$ 이고 $1 \leq i \leq m$ 이다.

1)과 2)의 경우 β를 s의 서픽스(suffix)라고 정의한다.

정의 5. 시퀀스 $s = \langle e_1 e_2 \ldots e_m \ldots e_n \rangle$에 대하여 $\alpha = \langle e_1 e_2 \ldots e_{m-1} e_{m'} \rangle$이고 $\beta = \langle e_{m''} e_{m+1} \ldots e_n \rangle$라고 하자. 단, $1 \leq m \leq n$ 이다. $x < y$ 일 경우 $^{\forall}x \in e_{m'}$, $^{\forall}y \in e_{m''}$ 에 대해서 $e_{m'} \subseteq e_m$ 이고 $e_{m''} = e_m - e_{m'}$ 일 경우 α를 S의 프레픽스(prefix) 라고 정의하며, $\beta = s|\alpha$ 로 표기한다.

정의 6. 시퀀스 $s = \langle e_1 e_2 \ldots e_n \rangle$와 $^{\forall}x \in I$에 대해서,
1) 시퀀스 데이터베이스 S의 원소이면서 프레픽스로서 α를 갖는 시퀀스 s'에 대하여, $s'|\alpha = s$ 이고 last(α)=x 일 경우 s 는 x-프로젝티드 순차패턴이라고 정의한다.
2) 시퀀스 데이터베이스 S에 대하여 x-프로젝티드 순차패턴들의 집합을 S|x으로 표기하며 x-프로젝티드 데이터베이스라고 정의한다.

정의 7. 시퀀스 $s = \langle e_1 e_2 \ldots e_n \rangle$에 대해서 $e1 = (\alpha_1, \alpha_2, .., \alpha_{m1})$, $e2 = (\beta_1, \beta_2, .., \beta_{m2}), .., en = (\gamma_1, \gamma_2, .., \gamma_{mn})$라고 하자. 시퀀스 s의 item_sequence는 $\alpha_1 \alpha_2 \ldots \alpha_{m1} \beta_1 \beta_2 \ldots \beta_{m2} \ldots \gamma_1 \gamma_2 .. \gamma_{mn}$로 정의하며 item_sequence(s)로 표기한다. 이 때, item_sequence(s)의 길이는 $\sum_{i=1}^{n} |ei|$ 이다.

예제. s1=<abc>, s2=<a(bc)>, s3=<(ab)c>일 경우 item_sequence(s1), item_ sequence(s2) 그리고 item_sequence(s3) 는 모두 abc로 동일하다.

정의 8. α를 길이가 k인 item_sequence 라고 하자.
1) item_sequence(s)=[α]이고 길이가 k인 모든 순차패턴들의 집합을 F[k,α]라고 정의한다. 즉. F[k,α]={s∈Fk|item_sequence(s)=[α]}이다.
2) F[k,α]의 x-extension을 F[k+1,αx]라고 정의한다.

정의 9. β를 길이가 k-1인 item_sequence 라고 하자. $X \subseteq F_1$에 대해서 extension[F[k-1, β],X] = $\bigcup_{x \in X}$ extension(F[k,βx) 라고 정의한다.

Lemma 1. α를 길이가 k인 item_sequence 라고 하자. 그러면 $^{\forall}x \in F_1$ 에 대해서 F[k+1,αx] 는 extension[F[k,α],{x}]와 같다.

Lemma 2. α와 β를 길이가 k인 item_sequences 라고 하자.
1) 만일 [α]≠[β] 이면, F[k,α] ∩ F[k,β]는 공집합이다.
2) 만일 F[k,α] ∩ F[k,β]이 공집합이면, $^{\forall}x \in F_1$에 대해서 F[k+1,αx] ∩ F[k+1,βx]는 공집합이다.

표 3. 순차패턴 집합의 단계적 분할

	F_1	F_2	F_3	F_4	F_5	...
F_a	F[1,a]	...	...	...	...	...
F_b	F[1,b]	F[2,ba]	...	...	...	...
		F[2,bb]				
		F[2,bc]	F[3,bca]	(A)	...	...
			F[3,bcb]	F[4,bcba]	F[5,bcbaa]	...
					F[5,bcbab]	...
					F[5,bcbac]	...
				F[5,bcbb]	...	...
				F[5,bcbc]	...	...
			F[3,bcc]	(B)	...	...
F_c	F[1,c]	...	...	...	...	...

Lemma 2로부터, 표 1에서 item_sequence bca과 bcc는 다르기 때문에 (A) 영역과 (B) 영역은 서로 분리된(disjoint) 영역이 된다. 이것은 F_3으로부터 F_4의 후보 패턴집합을 계산하는 과정에서 F[3,bca]는 F_4의 (B) 영역의 후보 순차패턴들을 구성하는 것과

는 관련성이 없다는 것을 의미한다.

Lemma 3. F 는 $\{F_x | x \in F_1\}$ 과 $\{F_1, F_2, ..., F_{max}\}$으로 각각 분할된다.

정리 1. F 는 $\{F_k \cap F_a | k=1,2,...,m \text{ and } a \in F_1\}$에 의해서 분할된다.

Lemma 3과 정리 1로부터, 표 1에서 보는 바와 같이 시퀀스 데이터베이스 S의 모든 순차패턴들의 집합은 서로 다른 부분집합들에 의해 분할될 수 있고, 단계-k의 각 분할집합은 이전 단계의 분할집합에 의해 다시 분할될 수 있다. 이러한 사실을 이용하여 단계적으로 후보 순차패턴들의 집합을 구성할 수 있다.

정리 2. $^{\forall}k \geq 3$, $^{\forall}x \in F_1$ 에 대해서 $F_k \cap F_x$ 는 extension($F_{k-1} \cap F_x, F_1$)에 의해서 분할된다.

정리 3. α를 길이가 k인 item_sequence 라고 하자. $\forall x \in F_1$, $k \geq 1$에 대해서 F[k,α] 가 공집합이면 F[k+1,αx] 도 공집합이다.

정리 4. $^{\forall}xi \in F_1$, $1 \leq i \leq k$ 에 대해서 $F[k, x_1x_2....x_k]$이 공집합이면, $^{\forall}y \in F_1$에 대해서 $F[k+1, x_1x_2....x_{k-1}\boldsymbol{y}x_k]$도 공집합이다.

정리 2로부터, extension(F_{k-1}, F_1)는 F_k의 후보시퀀스 집합($C_{k\text{-candidate}}$)이 된다. 특히 정리3과 정리4는 extension(F_{k-1}, F_1)에서 시퀀스 데이터베이스를 조사하지 않고도 순차패턴이 될 수 없는 시퀀스를 찾을 수 있는 방법을 제공한다. 예를 들면, 표-1에서 F_4의 F[4,bcba]가 공집합이면, 정리 3으로부터 F_5의 F[5,bcbaa], F[5,bcbab], F[5,bcbac]는 모두 공집합이 된다는 것을 알 수 있다. 그리고 만약 표-1에서 F_3의 F[3,bca]가 공집합이면, 정리 4로부터 F_4의 F[4,bcba]와 F[4,bcca]도 공집합이 된다는 것을 알 수 있다.

정리 5. 시퀀스 s의 길이가 k라고 하자. 만일 s가 순차패턴이면 s의 suffix(s)는 길이가 k-1인 순차패턴이다.

정리 5로부터, 길이 k인 시퀀스 s 가 순차패턴이기 위해서는 suffix(s)는 반드시 (k-1)-순차패턴이어야 한다. 그리고 정리 1로부터, suffix(s)가 k-순차패턴인지를 알기 위해서는 F_{k-1} 전체가 아닌 $F_{k-1} \cap F_{first(suffix(s))}$를 조사하면 된다. 제안된 순차패턴 마이닝 방법의 알고리즘은 다음과 같다.

Algorithm Our_*Mining_Method*

// S = a sequential database

// F_k = the set of all sequential patterns with length k

// F_x = the of all sequential patternes s such that first(s)=x, for each $x \in F_1$

1. Input S, min_sup;
2. By Scanning S, Construct F_1, F_x for each $x \in F_1$;
3. By scanning S,
   For each $x \in F_1$, Construct x-projected databases ;
   Construct F_2 using the same method of PrefixSpan;
   For each $s \in F_2$, $F_{first(s)} \leftarrow F_{first(s)} \cup \{s\}$;
4. // Construction of $F_k(k \geq 3)$
   $k \leftarrow 2$;
   while ($F_k \neq \varnothing$) {
   Call ***SuffixChecking***(min_sup,F_k,F_1,F_x for each $x \in F_1$);
   $k \leftarrow k+1$;
   }
6. Output $F_1 \cup F_2 \cup ... F_{K-1}$;

Subroutine *SuffixChecking*(min_sup, F_k, F_1, F_x for each $x \in F_1$)

1. $F_{K+1} \leftarrow \varnothing$;
2. By thorem-2,3,4, Construct $C_{(k+1)\text{-candidate}}$;
3. For each $s \in C_{k\text{-candidate}}$,
   if ($s \in F_{first(suffix(s))} \cap F_k$) {
   Compute support(s) by scanning first(s)-projected database;
   if (support(s) $\geq$ min_sup) {
   $F_{k+1} \leftarrow F_{K+1} \cup \{s\}$;
   $F_{first(s)} \leftarrow F_{first(s)} \cup \{s\}$;

}

}

4. Output F_{k+1}, F_x for each $x \in F_1$;

3.2 제안한 알고리즘의 적용 예

이 절에서는 예제를 가지고 제안한 알고리즘을 설명한다. 제안한 알고리즘은 먼저 시퀀스 데이터베이스의 탐색을 통해 길이 1의 모든 순차패턴들의 집합 F_1과 길이 2인 모든 순차패턴들의 집합 F_2를 생성한다. 그리고 F_{k-1}에 의한 F_k의 분할특성(정리 3, 정리 4)과 순차패턴들의 서픽스 특성(정리 5)을 이용하여 $F_{k-1}(k \geq 3)$로부터 F_k의 후보패턴들을 한다. 그리고 후보패턴들에 대한 탐색공간으로 후보패턴의 첫 번째 항목에 대한 프로젝티드 데이터베이스를 조사하여 F_k를 생성한다. 이러한 과정은 최대 길이의 순차패턴들의 집합 F_{max}가 생성될 때까지 반복한다. 표 2의 예제 시퀀스 데이터베이스에 대하여 제안한 알고리즘의 마이닝 과정을 설명하면 다음과 같다.

표 2. 시퀀스 데이타베이스(예제)

s_{id}	sequence
10	<a(abc)(ac)d(cf)>
20	<ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb
40	<eg(af)cbc)

표 3. 1-프레픽스 프로젝티드 데이타베이스

<x>	<x>-프로젝티드 데이타베이스
<a>	<(abc)(ac)d(cf)>, <(_d)c(bc)(ae)>, <(_b)(df)cb>, <(_f)cbc>
<b>	<(_c(ac)d(cf)>, <(_c)(ae)>, <(df)cb>, <c>
<c>	<(ac)d(cf)>, <(bc)(ae)>, <b>, <c>
<d>	<(cf)>, <c(bc)(ae)>, <(_f)cb>
<e>	<(_f)(ab)(df)cb>, <g(af)cbc>
<f>	<(ab)(df)cb>, <cbc>

단계 1 : 주어진 시퀀스 데이타베이스를 조사해서 F_1을 생성하고, 각 $x \in F_1$에 대해, F_x={<x>}를 구성한다. 표2의 예제 데이타베이스에서, 최소지지도를 2로 가정하면 F_1={<a>, <b>, <c>, <d>, <e>, <f>}와 F_a={<a>}, F_b={<b>}, F_c={<c>}, F_d={<d>}, F_e={<e>}, F_f={<f>}를 얻는다.

단계 2 : 시퀀스 데이터베이스로부터, F_1의 각 항목 x에 대해 x-프로젝티드 데이타베이스를 구성한다. 정리 1로부터, 이러한 프로젝티드 데이타베이스들은 마이닝 과정의 탐색공간을 줄인다. 표 3은 표 2의 시퀀스 데이터베이스로부터 구성된 프로젝티드 데이터베이스들이다.

단계 3 : 시퀀스 데이터베이스로부터 F_2를 구성한다. 그리고, F_2의 각 순차패턴 s는 항목 first(s)에 따라 $F_{first(s)}$에 추가된다. 단계 2와 단계 3은 한번의 시퀀스 데이터베이스 조사로 동시에 수행될 수 있다. 표 2로부터 이 단계에서 생성되는 결과는 다음과 같다.

$F_2 \cap F_a$
={<aa>,<ab>,<(ab)>,<ac>,<ad>,<af>}
$F_2 \cap F_b$ ={<ba>,<bc>,<(bc)>,<bd>,<bf>}

$F_2 \cap F_c$ ={<ca>,<cb>,<cc>}
$F_2 \cap F_d$ ={<db>,<dc>}
$F_2 \cap F_e$ ={<ea>,<eb>,<ec>,<ef>}
$F_2 \cap F_f$ ={<fb>,<fc>}

단계 4 : 제안한 알고리즘의 F_1과 F_2의 생성과정과 F_3 생성과정은 사실 PrefixSpan과 유사하다. 그러나 $F_k(k \geq 4)$를 얻는 과정부터는 PrefixSpan과는 다르다. 제안한 알고리즘에서 이전 단계에서 얻어진 F_{k-1} $(k \geq 3)$과 F_1으로부터, F_k를 생성하는 방법을 설명하면 다음과 같다.

$F_{k-1}(k \geq 3)$과 F_1으로부터 생성되는 F_k의 후보시퀀스 집합 $C_{k\text{-candidate}}$은 먼저 extension(F_{k-1},F_1)를 구성하고 난 후 정리 3과 정리 4를 이용하여 순차패턴이 될 수 없는 시퀀스들을 제거함으로써 구성된다. 예를 들면, 표 4의 Step-3에서 $C_{3\text{-candidate}}$는 extension(F_2,F_1)에서 extension($F_2 \cap F_a$,{<e>}), extension($F_2 \cap F_b$,{<b>, <e>}), extension ($F_2 \cap F_c$,{<d>,<e>,<f>}), extension ($F_2 \cap F_d$,{<a>,<d>,<e>,<f>}, extension($F_2 \cap F_e$,{<d>, <e>}), extension($F_2 \cap F_f$,{<a>,<d>,<e>,<f>})에 속하는 시퀀스들을 제거함으로써 구성된다. 그리고 정리 5에 의해, $C_{k\text{-candidate}}$의 각 시퀀스 s에 대해 suffix (s)가 $F_{k-1} \cap F_{first(suffix(s))}$에 존재하지 않으면 s는 무시되고, 그렇지 않으면 단계 2에서 구성된 first(s)-프로젝티드 데이타베이스를 조사하여 최소지지도를

만족하면 F_k와 $F_{first(s)}$에 추가한다. 예를 들어, 표 4의 단계 3에서, 순차패턴 s=<a(bc)>∈F_3 와 <a>∈ F_1를 고려해 보면, a-extension of <a(bc)>은 s_1=<a(bc)a>와 s_2=<a(bca)> 이다. 그리고 suffix(s_1)=<(bc)a> ∈ F_3 이지만, suffix(s_2)=<(bca)> ∉ F_3 이다. 따라서, 정리 5에 의해, s_2는 무시되고, s_1은 *a*-프로젝티드 데이타베이스를 조사하여 그 지지도를 계산하면, support(s_2)=2 ≥ min_sup 이므로 s_1은 F_4 에 속하게 된다. 또한 순차패턴 s=<a(bc)> ∈ F_3 와 <b> ∈ F_1를 고려해 보면, 정리 4에 의해 F[3,abb]=∅ 이므로 즉, 시퀀스 <a(b*c*)b>는 순차패턴이 될 수 없다. 이와 같은 F_{k-1}과 F_1 으로부터 F_k를 생성하는 과정은 F_k가 공집합이 될 때까지 반복한다.

4. 제안한 알고리즘과 기존연구들의 비교

본 논문에서 제안한 알고리즘의 단계적 순차패턴 구성과정은 GSP와 유사하지만 후보패턴들의 생성 방법과 후보패턴들에 대한 탐색공간으로 시퀀스 데이터베이스 대신 1-프레픽스 프로젝티드 데이타베이스를 이용한다는 점이 다르다. 또한, PrefixSpan과 같이 매 단계에서 데이터베이스 프로젝션 비용을 유발하는 대신 single_item 프로젝티드 데이타베이스만을 구성하기 때문에 초기의 데이터베이스 프로젝션 비용 이외에는 프로젝티드 데이터베이스 구성 비용이 발생하지 않는다.

대부분의 순차패턴 마이닝 알고리즘들과 마찬가지로 GSP, PrefixSpan과 SuffixSpan은 대용량 데이터를 다루기 때문에 많은 블록 I/O를 요구하며 알고리즘의 구현방법에 따라 그 성능에 차이를 가질 수 있다. 이것은 구현에 의한 세 알고리즘의 비교는 신뢰성 있는 비교결과를 얻는데 한계가 있다는 것을 의미한다. 따라서 구현에 의한 성능 비교보다는 해석적 접근에 의한 성능비교를 수행하는 것이 좋다.

순차패턴 마이닝 방법의 주요 성능요소는 시퀀스 데이터베이스의 조사횟수와 단계별 후보패턴 생성 비용 및 후보패턴 검사비용으로 구성된다. 세 알고리즘 모두 한번의 시퀀스 데이터베이스 조사를 통해 F_1을 생성한다. SuffixSpan과 PrefixSpan은 시퀀스 데이터베이스의 추가적인 조사를 통해 2-순차패턴들(F_2)을 생성하지만 GSP는 F_1으로부터 F_2를 생성한다. 이러한 주요 비용요소들을 고려한 각 알고리즘의 비용식은 다음과 같다.

(1) $C_{GSP} = \alpha + \sum_{i\geq 2}(\beta_{i\cdot GSP} + \gamma_{i\ GSP} * \alpha)$

(2) $C_{Prefix} = 2\alpha + C + \sum_{i\geq 3}(\beta(i,\beta(i-1)) + \gamma_{i\ prefix} * p_i \cdot \beta(i))$, where $0< p_i <1$

(3) $C_{Suffix} = 2\alpha + C + \beta(2) + \sum_{i\geq 3}(\beta_{i\ Suffix} + \gamma_{i\ Suffix} * q\cdot\alpha)$, where $0< q <1$

위 비용식에서 α는 시퀀스 데이타베이스 S의 1회 스캐닝 비용이고, C는 본 논문에서 제안한 방법과 PrefixSpan의 F_2 생성과정에서 요구되는 추가비용이다. 그리고 $\beta_{i\cdot GSP}$, $\beta(i,\beta(i-1))$, $\beta_{i\cdot Suffix}$는 각각 GSP,

표 4. 표 2에 대한 제안방법의 단계별 마이닝 결과

	Step-1 (F_1 생성)	Step-2 (F_2 생성)	Step-3 (F_3 생성)	Step-4 (F_4 생성)	Step-5 (F_5 생성)
F_a	<a>	<aa>, <ab>,<(ab)>, <ac>, <ad>, <af>	<aba>, <a(bc)>,<abc>, <(ab)c>, <(ab)d>, <(ab)f>, <aca>, <acb>, <acc>, <adc>	<a(bc)a>, <(ab)dc>	-
F_b	<b>	<ba>, <bc>, <(bc)>, <bd>, <bf>	<(bc)a>, <bdc>	-	
F_c	<c>	<ca>, <cb>, <cc>,	-		
F_d	<d>	<db>, <dc>	<dcb>	-	
F_e	<e>	<ea>, <eb>, <ec>, <ef>,	<eab>, <eac>, <ebc>, <ecb>, <efb>, <efc>	<efcb>	
F_f	<f>	<fb>, <fc>	<fbc>, <fcb>	-	

표 5. GSP, PrefixSpan 그리고 SuffixSpan의 비용식 요소 비교

Cost	GSP	PrefixSpan	SuffixSpan
F_1과 F_2의 생성	$\alpha + \beta_{2\text{-GSP}} + \gamma_{2\ \text{GSP}} * \alpha$	$2\alpha + C$	$2\alpha + C$
후보패턴 생성	joining on $F_{k-1} \times F_{k-1}$, scanning cts subsequences	-	partitioning F_{k-1}, suffix checking
프로젝티드 데이터베이스 구축	-	$\Sigma_i \beta(i)$, where $\beta$is descending	$\beta(2)$
후보패턴 검사	$\gamma_{i\ \text{GSP}} * \alpha$	$\gamma_{i\ \text{Prefix}} * p_i \cdot \beta(i, \beta(i-1))$, where $0 < p_i < 1$	$\gamma_{i\text{-Suffix}} * q \cdot \alpha$ where $0 < q < 1$

PrefixSpan, SuffixSpan에 의한 길이 i인 후보패턴 집합의 생성 비용이다. 그리고 $\gamma_{i\ \text{GSP}}$, $\gamma_{i\ \text{Prefix}}$, $\gamma_{i\ \text{Suffix}}$는 각각 GSP, PrefixSpan, 제안방법에 의한 길이 i인 후보패턴 개수이고, $p_i \cdot f(i)$, $q \cdot \alpha$는 각각 Prefix Span, 제안방법에 의한 길이 i인 후보패턴의 지지도 계산비용이다. p_i는 PrefixSpan에서 프로젝티드 database들의 크기 감소비율이고, q는 시퀀스 데이터베이스 S에 대한 1-prefix 프로젝티드 데이터베이스의 크기 감소비율이며, $p_3 = q$ 이다. $\beta(i, \beta(i-1))$는 PrefixSpan에서 길이 i-1인 프리픽스들에 의해 구성되는 프로젝티드 데이타베이스들의 구성비용으로, β는 단조 감소함수이고, 순차패턴의 최대길이 max에 대해, $\beta(\max+1, \beta(\max)) = 0$ 이다. 식 (3)의 $\beta(2)$는 제안방법에서 1-prefix 프로젝티드 데이터베이스의 구성비용이다.

일반적으로, $(\beta_{2\ \text{GSP}} + \gamma_{2\ \text{GSP}} * \alpha) > (\alpha + C)$ 이고, $\gamma_{i\ \text{prefix}} < \gamma_{i\ \text{Suffix}} \leq \gamma_{i\ \text{GSP}}$, 그리고 $\gamma_{i\ \text{prefix}} = |F_1|$이다. 따라서, 만약 $(\gamma_{i\ \text{Suffix}} * q \cdot \alpha)$가 $(\gamma_{i\text{-GSP}} * \alpha)$ 보다 작게 되면, SuffixSpan은 C와 $\beta(2)$의 추가비용을 고려하더라도 GSP보다 적은 비용을 갖는다. 그리고 $q \cdot \alpha$가 $\Sigma_{i \geq 3} \beta(i, \beta(i-1))$ 보다 충분히 작게 되면 $\gamma_{i\ \text{Suffix}} > \gamma_{i\ \text{prefix}}$ 이더라도 본 논문에서 제안한 방법은 Prefix Span 보다 적은 비용을 갖는다. 일반적으로 p_i, q는 시퀀스 데이터베이스에 종속해서 결정된다. 이상에서 설명한 내용을 정리하면 표 4와 같다.

5. 결 론

본 논문에서는 새로운 순차패턴 마이닝 알고리즘을 제안하고, 그 알고리즘에 대한 형식적 접근을 보였다. 본 논문에서 제안한 알고리즘은 GSP와 Prefix Span의 개념을 결합한 것으로, 그 기본개념은 순차패턴 집합의 단계별 분할특성과 suffix 특성을 이용하여 적은 비용으로 작은 크기의 후보패턴 집합을 생성하고, 1-프레픽스 프로젝티드 데이타베이스를 구성하여 후보패턴 검사를 위한 탐색공간을 줄인다.

일반적으로 순차패턴 마이닝이 효율적으로 수행되기 위해서는 후보패턴의 생성 비용을 줄이고, 동시에 생성된 후보패턴에 대한 탐색공간을 줄일 수 있어야 한다. SuffixSpan은 PrefixSpan의 프로젝티드 데이터베이스 개념을 GSP에 부분적으로 적용하고, 순차패턴의 서픽스 특성과 순차패턴 집합의 분할특성을 이용하여 GSP의 후보패턴 생성방법을 개선함으로써 이러한 목적을 달성한다. 또한, 본 논문에서 설명한 형식적 접근은 순차패턴 마이닝 문제에 대한 형식론을 제공한다.

향후 [8]에서 정의한 시간 제약조건(time constraints), 트랜잭션의 느슨한 정의(lossely defined transaction), 항목들 사이의 계층성(taxonomies)을 고려한 제안방법의 확장과 제안 방법에 대한 해석적 비용모델 개발과 응용시스템 개발에 관심을 갖는다.

참 고 문 헌

[1] M. J. Zaki. *SPADE: An Efficient Algorithm for Mining Frequent Sequences*. In Proc. of Machine Learning Journal, special issue on Unsupervised Learning (Doug Fisher, ed.), Vol. 42 Nos. 1/2, pages 31-60, Jan/Feb 2001.

[2] 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. Conf. Data Engineering (ICDE'01), pages 215-224, Heidelberg, Germany, April 2001.

[3] J. Han, J. Pei, B. Mortazavi-Asi, Q. Chen, U. Dayal, and M.-C. Hsu. *FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining.* In Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining, pages 355-359, 2000.

[4] J. Han, J. Pei, Y. Yin, *Mining Frequent Patterns without Candidate Generation*, Proc. 2000, ACM-SIGMOD Int. Conf. on Management of Data(SIGMOD'2000), pp. 1-12, Dallas TX May 2000

[5] M.Garofalakis, R. Rastogi, and K. Shim. *Spirit: Sequential pattern mining with regular expression constraints.* In Proc. 1999 Int. Conf. Very Large Data Bases(VLDB99), pages 223-234, Edinburgh, UK, Sept. 1999.

[6] R.Agrawal and R.Srikant. *Mining Sequential Patterns.* In Proc. Of the 11th Intl Conference on Data Engineering, Taipei, Tiwan, March 1995.

[7] R.Srikant and R.Agrawal. Mining Generalized Association Rules. In Proc. Of the 21st Intl Conference on Very Database, Zurich, Switzerland, September 1995.

[8] R.Srikant and R.Agrawal. *Mining Sequential Patterns: Generalizations and Performance Improvements.* Research Reports RJ 9994, IBM Almaden Research Center, San Jose, California, December 1995.

[9] R.Agrawal, S. Srikant, "Fast Algorithms for Mining Association Rules", Proc. 1994 Int. Conf. on Very Large Data Base, pp. 487-499, Santiago, Chile, Sept. 1994

허 용 도

1986년 고려대학교 수학과 (이학사)
1988년 고려대학교 대학원 이학석사(전산학 전공)
1993년 고려대학교 대학원 이학박사(전산학 전공)
1993년~현재 건양대학교 IT학부 부교수
관심분야 : 분산시스템, 컴퓨터 네트워크 보안, 데이터마이닝, 정보검색

조 동 영

1986년 고려대학교 수학교육학과(이학사)
1988년 고려대학교 대학원 이학석사(전산학 전공)
1992년 고려대학교 대학원 이학박사(전산학 전공)
1993년~현재 전주대학교 정보기술컴퓨터공학부 부교수
관심분야 : 데이타베이스, 데이터마이닝, 이동컴퓨팅, 멀티미디어

박 두 순

1981년 고려대학교 수학과 (이학사)
1983년 충남대학교 대학원 전산학전공(이학석사)
1988년 고려대학교 대학원 전산학전공(이학박사)
1992년 - 1993년 미국 U. of Illinois at Urbana-Chamaign CSRD 객원교수
2000년 현재 멀티미디어학회 논문지 정보처리 분과위원장
2000년 현재 순천향대학교 공과대학 학장
1985년 현재 순천향대학교 정보기술공학부 교수
관심분야 : 병렬처리, 멀티미디어 정보검색, 데이터마이닝, 컴퓨터 교육