시계열 데이타베이스에서 인덱스 보간법을 기반으로 정규화 변환을 지원하는 서브시퀀스 매칭 알고리즘 †

노 웅 기*°, 김 상 욱**, 황 규 영*
*한국과학기술원 전산학과, 첨단정보기술 연구센터 **강원대학교 컴퓨터정보통신공학부
{woong,kywhang}@mozart.kaist.ac.kr wook@cc.kangwon.ac.kr

An Index-Based Subsequence Matching Algorithm Supporting Normalization Transform in Time-Series Databases

Woong-Kee Loh*°, Sang-Wook Kim**, and Kyu-Young Whang*
*Department of Computer Science and Advanced Information Technology Research Center (AITrc)
Korea Advanced Institute of Science and Technology (KAIST)
**Division of Computer, Information, and Communications, Kangwon National University

요 약

본 논문에서는 시계열 데이타베이스에서 정규화 변환을 지원하는 서브시퀀스 매칭 알고리즘을 제안한다. 정규화 변환은 시계열 데이타 간의 절대적인 유클리드 거리에 관계 없이, 구성하는 값들의 상대적인 변화 추이가 유사한 패턴을 갖는 시계열 데이타를 검색하는 데에 유용하다. 제안된 알고리즘은 몇 개의 질의 시퀀스 길이에 대해서만 각각 인덱스를 생성한 후, 이를 이용하여 모든 가능한 길이의 질의 시퀀스에 대해서 탐색을 수행한다. 이때, 착오 기각이 발생하지 않음을 증명한다. 본 논문에서는 이와 같이 인덱스가 요구되는 모든 경우 중에서 적당한 간격의 일부에 대해서만 생성된 인덱스를 이용한 탐색 기법을 **인덱스 보간법**이라 부른다. 질의 시퀀스의 길이 256 ~ 512 중 다섯 개의 길이에 대해 인덱스를 생성하여 실험한 결과, 탐색 결과 선택률이 10^{-3}일 때 제안된 알고리즘의 탐색 성능이 순차 검색에 비하여 평균 14.6 배 개선되었다.

1 서 론

시계열(time series) 데이타는 일정한 시간 주기마다 얻어진 연속된 실수 값들로 이루어진 데이타이다 [2, 4]. 시계열 데이타는 새로운 데이타베이스 분야에서 점차 중요성이 더해가고 있으며, 시계열 데이타 간의 유사성 문제는 그러한 분야에서 가장 관심을 끄는 문제 중의 하나이다 [2]. 시계열 데이타베이스에 저장된 시계열 데이타를 **데이타 시퀀스**(*data sequence*)라고 부르며, 질의 시퀀스와 유사한 데이타 시퀀스를 검색하는 연산을 **유사 시퀀스 매칭**(*similar sequence matching*)이라고 한다 [1, 2, 4, 8].

본 논문에서는 정규화 변환 [5, 8]을 지원하는 서브시퀀스 매칭 알고리즘에 관하여 논의한다. 정규화 변환을 지원하는 유사 시퀀스 매칭은 데이타 시퀀스 간의 절대적인 유클리드 거리에 관계없이, 구성하는 값들의 상대적인 변화 추이가 유사한 패턴을 갖는 데이타 시퀀스들을 탐색한다 [5, 8]. 문제를 해결하기 위하여 고려할 수 있는 방법 중의 하나는 기존의 서브시퀀스 매칭 알고리즘 [2, 4]을 단순히 응용하는 것이다. 그러나, 이러한 방법은 질의 처리 결과에 포함되어야 할 서브시퀀스를 모두 찾아내지 못하는 착오 기각(false dismissal)이 발생한다. 문제 해결을 위하여 고려할 수 있는 다른 방법은 정규화 변환을 지원하는 기존의 전체 매칭 알고리즘 [5]을 서브시퀀스 매칭에 그대로 적용하는 것이다. 전체 매칭 알고리즘이 고정 길이의 질의 시퀀스만을 지원하므로, 임의 길이의 질의 시퀀스를 처리하기 위해서는 모든 가능한 질의 시퀀스 길이 각각에 대하여 하나씩의 다차원 인덱스를 생성해야 한다. 따라서, 인덱스 저장 공간과 데이타 시퀀스 삽입 및 삭제의 부담(overhead)이 매우 심각하다.

본 논문에서는 이러한 문제점들을 극복할 수 있는 효율적인 **정규화 변환** 서브시퀀스 매칭 알고리즘을 제안한다. 제안된 알고리즘에서는 모든 가능한 질의 시퀀스 길이 중에서 적당한 간격으로 일부에 대해서만 인덱스를 생성하고, 임의 길이의 질의 시퀀스에 대하여 적절한 인덱스를 선택하여 **정규화 변환** 서브시퀀스 매칭을 수행한다. 이때, 질의 처리 결과에서 착오 기각이 발생하지 않음을 증명한다. 제안된 알고리즘이 모든 가능한 질의 시퀀스 길이들 중 일부만을 선택하여 인덱스를 생성하고 탐색을 수행하므로, 본 논문에서는 제안된 탐색 기법을 **인덱스 보간법**(*index interpolation*)이라 부른다.

제안된 알고리즘의 우수성을 규명하기 위하여 다양한 실험을 통하여 탐색 성능을 분석한다. 실험 결과, 적은 개수의 인덱스만을 이용하는 경우에도 모든 가능한 길이의 질의 시퀀스에 대하여 인덱스를 구성하는 경우에 비하여 탐색 성능이 크게 저하되지 않는 것으로 나타났다. 질의 시퀀스의 길이 256 ~ 512 중 다섯 개의 길이에 대해 인덱스를 생성한 경우, 제안된 알고리즘은 순차 검색(sequential search)과 비교하여 탐색 성능이 평균 14.6 배까지 개선되었다.

2 문제 정의

정의 1 길이 n (≥ 1)인 시퀀스 $\vec{X} = (x_i)$ $(0 \leq i < n)$를 정규화 변환한 시퀀스 $\nu(\vec{X}) = (\bar{x}_i)$는 다음과 같이 정의한다

† 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음.

[5, 8]:

$$\tilde{x}_i = \frac{x_i - \mu(\vec{X})}{\sigma(\vec{X})}$$

여기에서, $\mu(\vec{X})$와 $\sigma(\vec{X})$는 각각 시퀀스 $\vec{X}$의 평균(mean)과 표준 편차(standard deviation)이다. □

정규화 변환을 지원하는 서브시퀀스 매칭 문제는 다음과 같이 정의한다. 질의 시에 질의 시퀀스 $\vec{T}$와 탐색 영역 ϵ이 주어지면, 정규화 변환한 질의 시퀀스 $\nu(\vec{T})$를 비교 대상으로 삼는다. 데이타베이스에 저장된 데이타 시퀀스 $\vec{S}$ 내에 질의 시퀀스 $\vec{T}$와 같은 길이를 갖는 임의의 서브시퀀스 $\vec{X}$를 정규화 변환한 시퀀스 $\nu(\vec{X})$가 $\nu(\vec{T})$와 유사하면, 즉 다음의 공식 (1)을 만족하면, 데이타 시퀀스 $\vec{S}$와 $\vec{S}$ 내에서의 서브시퀀스 $\vec{X}$의 위치를 반환한다.

$$d\left(\nu(\vec{X}), \nu(\vec{T})\right) \le \epsilon \tag{1}$$

3 정규화 변환 서브시퀀스 매칭 기법

본 논문에서는 참고문헌 [5]의 알고리즘을 확장하여 문제를 해결한다. 참고문헌 [5]의 알고리즘을 확장 없이 단순히 응용하여 정규화 변환 서브시퀀스 매칭을 수행하기 위한 방법은 모든 가능한 질의 시퀀스 길이에 대하여 하나씩의 인덱스를 생성하는 것이다. 그 이유는 하나의 정해진 길이 n_0에 대하여 생성된 인덱스를 이용하여 임의 길이 n ($> n_0$)의 질의 시퀀스에 대한 정규화 변환 서브시퀀스 매칭을 수행할 때 착오 기각이 발생하기 때문이다. 그러나, 이러한 방법은 저장 공간 및 데이타 시퀀스 삽입/삭제의 부담이 매우 심각하다.

본 논문에서는 이러한 문제를 극복하기 위하여, 미리 선택된 몇 개의 질의 시퀀스 길이 w에 대해서만 인덱스를 생성하고, 이를 이용하여 임의 길이 n ($\ge w$)의 질의 시퀀스에 대한 탐색을 수행하는 새로운 탐색 기법을 제안한다. 본 논문에서는 길이 w의 질의 시퀀스에 대하여 생성된 인덱스를 w-**인덱스**(w-*index*)라 정의한다. 주어진 질의 시퀀스의 길이 n에 대하여, 인덱스를 생성한 질의 시퀀스 길이 w 중에 같은 값이 있으면 그 값에 대한 w-인덱스를 이용하고, 같은 값이 없으면 w-인덱스 중에서 다음의 공식 (2)에 의하여 하나를 선택하여 탐색을 수행한다.

$$\omega = \max\{w | w < n\} \tag{2}$$

이때, 공식 (2)에 의해 선택되어 질의 처리에 사용되는 인덱스를 ω-**인덱스**라 정의한다.

질의 시퀀스의 길이 n이 ω 값과 같지 않은 경우의 탐색을 위하여 다음의 정리 1이 필요하다. 정리 1은 탐색 영역 ϵ을 대치하여 탐색 결과에 착오 기각이 발생하지 않음을 보장하는 새로운 탐색 영역 ϵ'을 제시한다.

정리 1 길이 n (≥ 1)인 임의의 시퀀스 $\vec{X} = (x_i)$, $\vec{T} = (t_i)$ ($0 \le i < n$)에 대하여 다음이 성립한다 ($0 \le s \le f < n$):

$$d\left(\nu(\vec{X}), \nu(\vec{T})\right) \le \epsilon \Rightarrow d\left(\nu(\vec{X}[s,f]), \nu(\vec{T}[s,f])\right) \le \epsilon' \tag{3}$$

여기에서, ϵ'은 다음과 같으며, ω는 윈도우 $\vec{T}[s,f]$의 길이이다 ($\omega = f - s + 1$). $\vec{T}[s,f]$는 시퀀스 $\vec{T}$ 내의 $t_s, \ldots, t_f$ 값으로 구성한 윈도우이다.

$$\epsilon' = \sqrt{2\omega - 2\sqrt{\omega^2 - \omega \cdot \epsilon^2 \cdot \frac{\sigma^2(\vec{T})}{\sigma^2(\vec{T}[s,f])}}} \tag{4}$$

증명: 생략. □

공식 (4)에서 안쪽의 제곱근 내의 값이 0 이상이어야 한다. 즉, ϵ'을 계산함에 있어서 다음 공식 (5)의 조건을 검토하여야 한다. 공식 (4)에서 바깥쪽의 제곱근 내의 값은 항상 0 이상이다.

$$\omega > \epsilon^2 \cdot \frac{\sigma^2(\vec{T})}{\sigma^2(\vec{T}[s,f])} \tag{5}$$

질의 시퀀스의 길이 n이 공식 (2)에 의하여 얻어진 ω와 같지 않으면, ω-인덱스를 이용하여 공식 (3)의 결론부의 식인 다음의 공식 (6)을 만족하는 모든 서브시퀀스 $\vec{X}$로 후보 집합을 구성한다:

$$d\left(\nu(\vec{X}[s,f]), \nu(\vec{T}[s,f])\right) \le \epsilon' \tag{6}$$

윈도우 $\vec{T}[s,f]$의 길이가 ω이므로, 공식 (6)을 이용한 탐색을 수행하기 위하여 질의 시퀀스 $\vec{T}$로부터 추출 가능한 윈도우 $\vec{T}[s,f]$의 개수는 $(n - \omega + 1)$ 개이다. 이들 중에서 새로운 탐색 영역 ϵ'을 이용한 탐색의 성능을 최대화하기 위하여 다음의 공식 (7)에 의하여 윈도우 $\vec{T}[s,f]$를 결정한다 ($0 \le s' \le f' < n, f' - s' = \omega - 1$):

$$\vec{T}[s,f] = \left\{ \vec{T}[s',f'] \,\middle|\, \sigma(\vec{T}[s',f']) \text{ is maximum.} \right\} \tag{7}$$

탐색 영역 ϵ'이 클수록 공식 (6)을 이용한 탐색에 의한 착오 채택(false alarm)의 가능성이 커지므로, 가급적 ϵ'을 작게 설정하여야 한다. 공식 (4)에서 ϵ' 값을 변화시킬 수 있는 인자는 $\sigma(\vec{T}[s,f])$ 뿐이며, $\sigma(\vec{T}[s,f])$ 값이 최대가 될 때 ϵ' 값이 최소가 된다. 따라서, 공식 (7)에 의해 결정된 윈도우 $\vec{T}[s,f]$를 이용하여 탐색을 수행할 때 최고의 탐색 성능을 얻을 수 있다.

탐색 영역 ϵ'에 대해 ω-인덱스를 이용하여 공식 (6)을 만족하는 탐색을 수행할 때, 정리 1을 통하여 착오 기각이 발생하지 않음을 보인다. 정리 1에서 제시한 공식 (3)의 전제부의 조건을 만족하는 모든 서브시퀀스 $\nu(\vec{X})$에 대하여, 윈도우 $\nu(\vec{X}[s,f])$가 항상 결론부의 조건을 만족한다. 즉, 공식 (3)의 결론부의 조건을 만족하는 $\left(\nu(\vec{X}[s,f]),\ \nu(\vec{T}[s,f])\right)$ 쌍의 집합은 전제부의 조건을 만족하는 대응되는 $\left(\nu(\vec{X}),\ \nu(\vec{T})\right)$ 쌍의 집합을 포함한다. 따라서, 결론부의 식에 의하여 탐색을 수행할 때 착오 기각이 발생하지 않는다.

4 성능 평가

본 실험의 목적은 몇 개의 w-인덱스만을 이용하더라도 모든 질의 시퀀스 길이 n에 대하여 인덱스가 생성되어 있는 경우에 비해 제안된 알고리즘의 탐색 성능이 크게 저하되지 않음을 보이는 것이다.

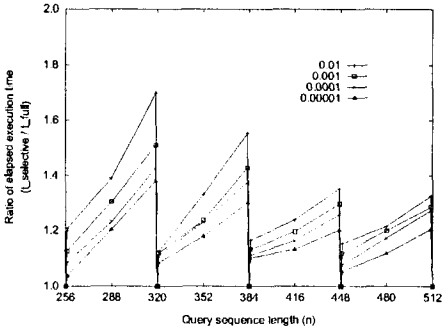

그림 1: 실행 시간 비율 ($t_{selective}/t_{full}$).

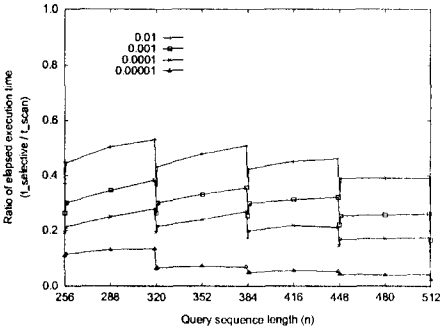

그림 2: 실행 시간 비율 ($t_{selective}/t_{scan}$).

본 실험에 사용한 데이타베이스는 1994년 11월부터 1998년 5월까지 길이 1024의 한국 주가 데이타 620 개 종목의 데이타 시퀀스로 구성된다. 제안된 알고리즘의 실험을 위하여 선택된 질의 시퀀스 길이 $w = 256, 320, 384, 448, 512$에 대하여 w-인덱스를 생성하였고, 모든 질의 시퀀스 길이에 대하여 인덱스를 생성하여 참고문헌 [5]의 알고리즘을 확장 없이 응용하는 경우의 실험을 위하여 질의 시퀀스 길이 $n = 256 + 32i$ $(i = 0, \ldots, 8)$인 경우와 $n = w \pm 1$인 경우에 대하여 인덱스를 생성하였다. 본 실험에서는 인덱스의 차원을 줄이기 위하여 DFT 변환 [7]을 사용하였다. 인덱스 차원을 여러 가지로 변화하며 실험하여 가장 탐색 성능이 우수한 $f = 6$ 차원을 선택하였다. 탐색 영역 ϵ은 128 개의 질의 시퀀스 각각에 대하여 선택률(selectivity)을 기준으로 결정하였고, 실험에 사용된 선택률 값들은 0.00001, 0.0001, 0.001, 0.01이다.

첫번째 실험은 주어진 질의 시퀀스의 길이 n에 대한 w-인덱스가 생성되어 있지 않은 경우와 생성되어 있는 경우의 탐색 알고리즘의 실행 시간(elapsed execution time)을 비교하는 실험이다. 이 실험의 목적은 질의 시퀀스 길이 n에 대한 w-인덱스가 생성되어 있지 않은 경우에 제안된 알고리즘의 실행 시간이 얼마나 더 걸리는지 비교하기 위한 것이다. 데이타베이스 프로그램의 전체 실행 시간은 CPU 작업 시간과 디스크 액세스 시간의 합이다. 이 실험에서는 실제의 디스크 I/O를 보장하기 위하여 OS에 의한 버퍼링(buffering)을 수행하지 않는 루틴들을 사용하였다 [6].

그림 1은 첫번째 실험에 의한 결과를 보인 것이다. 세로축은 인덱스가 없는 경우의 실행 시간 $t_{selective}$를 인덱스가 있는 경우의 실행 시간 t_{full}로 나눈 비율 값을 나타낸다. 이 값은 128 개의 질의 시퀀스에 대하여 얻어진 실행 시간 비율 값들을 평균한 것이다 (최대 약 1.70 배).

두번째 실험은 제안된 알고리즘과 순차 검색 알고리즘의 실행 시간을 비교하는 실험이다. 이 실험의 목적은 질의 시퀀스 길이 n에 대한 w-인덱스가 생성되어 있지 않은 경우에 제안된 알고리즘이 순차 검색에 비하여 얼마나 성능이 개선되는가를 보이기 위한 것이다. 이 실험에서도 첫번째 실험과 마찬가지로 OS에 의한 버퍼링을 수행하지 않는 루틴들을 사용하였다 [6].

그림 2는 두번째 실험에 의한 결과를 보인 것이다. 세로축은 제안된 알고리즘이 w-인덱스를 이용하여 탐색에 걸린 실행 시간 $t_{selective}$를 순차 검색 알고리즘의 실행 시간 t_{scan}으로 나눈 비율 값을 나타낸다. 이 값은 128 개의 질의 시퀀스에 대하여 얻어진 실행 시간 비율 값들을 평균한 것이다. 제안된 알고리즘의 탐색 성능이 순차 검색에 비하여 개선된 배율을 선택률 0.00001에 대하여 평균하면 14.6 배로 나타났다. 이때, 사용하는 w-인덱스의 개수를 늘림으로써 탐색 성능을 더욱 향상시킬 수 있다.

참고문헌

[1] Agrawal, R. et al., "Efficient Similarity Search in Sequence Databases," In *Proc. Int'l Conf. on Foundations of Data Organization and Algorithms*, pp. 69-84, Chicago, Illinois, Oct. 1993.

[2] Agrawal, R. et al., "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases," In *Proc. Int'l Conf. on Very Large Data Bases*, pp. 490-501, Zurich, Switzerland, Sept. 1995.

[3] Beckmann, N. et al., "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 322-331, Atlantic City, NJ, June 1990.

[4] Faloutsos, C. et al., "Fast Subsequence Matching in Time-Series Databases," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 419-429, Minneapolis, Minnesota, June 1994.

[5] Goldin, D. Q. and Kanellakis, P. C., "On Similarity Queries for Time-Series Data: Constraint Specification and Implementation," In *Proc. Int'l Conf. on Principles and Practices of Constraint Programming*, pp. 137-153, Cassis, France, Sept. 1995.

[6] Hart, J. M., *Win32 System Programming*, Addison-Wesley Developers Press, 1997.

[7] Kreyszig, E., *Advanced Engineering Mathematics*, 7th Ed., John Wiley & Sons, 1993.

[8] Rafiei, D. and Mendelzon, A., "Similarity-Based Queries for Time Series Data," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 13-25, Tucson, Arizona, June 1997.