산업기술연구(강원대학교 산업기술연구소 논문집), 제20권, 2000.
Journal of Industrial Technology, Kangwon Nat'l Univ., Korea, No. 20, 2000.

시퀀스 데이터베이스를 위한 타임 워핑 기반 유사 검색

A Method for Time Warping Based Similarity Search in Sequence Databases

김 상 욱* 박 상 현**
Kim, Sang-Wook Park, Sang-Hyun

Abstract

In this paper, we propose a new novel method for similarity search that supports time warping. Our primary goal is to innovate on search performance in large databases without false dismissal. To attain this goal, we devise a new distance function $D_{tw\text{-}lb}$ that consistently underestimates the time warping distance and also satisfies the triangular inequality. $D_{tw\text{-}lb}$ uses a 4-tuple feature vector extracted from each sequence and is invariant to time warping. For efficient processing, we employ a multidimensional index that uses the 4-tuple feature vector as indexing attributes and $D_{tw\text{-}lb}$ as a distance function. We prove that our method does not incur false dismissal. To verify the superiority of our method, we perform extensive experiments. The results reveal that our method achieves significant speedup up to 43 times with real-world S&P 500 stock data.

Keywords : sequence database, similarity search, time-warping

I. 서 론

시퀀스 데이터베이스(sequence database)란 객체의 변화되는 값들의 연속으로 구성된 데이터 시퀀스(data sequence: 이후부터 간략히 시퀀스라 칭함)들의 집합이다[1]. 대표적인 예로는 주가 데이터, 환율 데이터, 기온 데이터, 제품 판매량 데이터, 기업 성장률 데이터 등이 있다[2][9]. 유사 검색(similarity search)이란 주어진 질의 시퀀스(query sequence)와 변화의 패턴이 유사한 시퀀스들을 시퀀스 데이터베이스로부터 찾아내는 연산이다[1][2][9]. 이러한 유사 검색은 데이터 마이닝(data mining) 및 데이터 웨어하우징(data warehousing) 분야에서 중요한 연산으로 사용된다[6][17].

유사 검색에 관한 기존의 많은 연구에서는 길이 n의 시퀀스를 n 차원 공간상의 한 점으로 간주하고, 두 시퀀스들간의 유사한 정도를 측정하기 위하여 두 점들간의 유클리드 거리(Euclidean distance)를 이용한다[1][7][9][11][17].

유클리드 거리만을 이용한 유사 검색을 통해서는 사용자가 원하는 시퀀스들을 검색하지 못하는 경우가 빈번하게 발생한다. 따라서 응용 분야에 적합한 유사 모델(similarity model)을 적절하게 정의할 수 있도록 변환(transform)을 지원하기도 한다. 초기의 연구인 참고 문헌 [1][9] 등에서는 변환을 지원하지 않았으나, 이후에는 스케일링(scaling) [2][7], 시프팅(shifting) [2][7], 정규화(normalization)[8][11][13], 이동평균(movingaverage)[14][17], 타임 워핑(time warping) [4][15][20] 등의 다양한 변환을 지원하는 방법들이 제안되었다.

이들 중 타임 워핑은 시퀀스내의 각 요소 값을 임의의 수만큼 반복시키는 것을 허용하는 변환이다[20].

* 강원대학교 컴퓨터정보통신공학부 조교수
** UCLA 대학 컴퓨터 과학과 박사과정

예를 들어, 타임 워핑에 의하여 두 시퀀스 S = <20, 21, 21, 20, 20, 23, 23, 23> 와 Q = <20, 20, 21, 20, 23>를 동일한 시퀀스 <20, 20, 21, 21, 20, 20, 23, 23, 23>으로 변환시킬 수 있다. 타임 워핑 후의 두 시퀀스들 간의 거리를 타임 워핑 거리(time warping distance)라 정의한다. 타임 워핑은 데이터베이스내의 시퀀스들의 길이가 서로 달라서 유클리드 거리를 이용하여 유사 정도를 직접 측정할 수 없는 경우에 매우 유용하다.

기존의 연구에서는 효율적인 유사 검색을 위하여 다차원인덱스(multidimensionaindex)[3][5][18]를 사용한다[1][2][9]. 대부분의 인덱스들은 채택하는 거리 함수가 삼각형 부등식 성질(triangle inequality)[16]을 만족한다는 것을 전제로 한다. 만일, 이 성질을 만족하지 못하는 거리 함수를 이용하는 경우에는 유사 검색 시 착오 기각(false dismissal)이 발생된다[20]. 착오 기각이란 실제 질의 결과로 반환되어야 할 질의 시퀀스와 유사한 시퀀스를 올바르게 찾아내지 못하는 현상이다[1][9]. 참고 문헌 [20]에서는 타임 워핑 거리가 삼각형 부등식 성질을 만족하지 못함을 증명하고, 착오 기각을 허용하지 않는 응용에서 타임 워핑을 지원하는 유사 검색을 처리할 때에는 거리 함수 기반 인덱스를 사용할 수 없다고 주장한 바 있다.

참고 문헌 [4]와 [20]에서는 인덱스 없이 시퀀스들을 모두 액세스함으로써 타임 워핑 지원 유사 검색을 처리하는 방법을 제안하였다. 그러나 대규모의 데이터베이스 환경에서는 이와 같이 인덱스를 사용하지 않는 경우, 검색 성능이 심각하게 저하된다. 참고 문헌 [20]에서는 FastMap[10]을 이용하여 k(<<n) 차원 공간내의 점들로 변환된 시퀀스들을 대상으로 다차원 인덱스를 구성함으로써 검색 성능을 개선하는 방식을 제안하였다. 그러나 이 방식은 착오 기각을 유발시킨다는 심각한 문제점을 가지므로, 이를 허용하는 제한된 응용에 한해서만 사용될 수 있다. 참고 문헌 [15]에서 우리는 거리 함수를 기반으로 하지 않는 서픽스 트리(suffix tree)[19]를 사용함으로써 착오 기각을 허용하지 않으면서 부분 매칭 시의 검색 성능을 개선시킬 수 있는 방식을 제안하였다. 그러나 이 방식은 좋은 성능을 보장하는 분류 작업(categorization)이 매우 복잡하며, 또한 전체 매칭 시에는 트리의 크기가 매우 커지므로 검색 성능이 저하된다는 문제점을 갖는다.

본 논문에서는 타임 워핑을 지원하는 유사 검색을 처리하기 위한 효율적인 방법에 관하여 논의한다. 본 연구의 목표는 착오 기각 발생의 방지와 빠른 검색 성능을 동시에 보장하는 것이다. 본 연구에서는 새로운 거리 함수를 고안하고, 이 거리 함수를 기반으로 구성된 다차원 인덱스를 이용하여 타임 워핑을 지원하는 유사 검색을 빠르게 처리할 수 있는 새로운 기법을 제안한다.

제안된 기법의 견고성(robustness)를 규명하기 위하여 유사 검색에서 착오 기각이 발생되지 않음을 증명한다. 또한, 다양한 실험에 의한 성능 분석을 통하여 제안된 기법의 우수성을 제시한다.

II. 용어 정의

시퀀스 데이터베이스는 다양한 길이를 갖는 시퀀스들의 집합으로 구성된다. 시퀀스 S(= <s_1, s_2, ..., $s_{|S|}$>)는 실수인 요소 값들의 연속이다. 여기서 |S|는 시퀀스의 길이이며, s_i는 S의 i번째 요소를 의미한다. First(S)와 Last(S)는 각각 S의 첫 번째 요소 s1과 마지막 요소 $s_{|S|}$를 의미한다. Rest(S)는 s_1을 제외한 S의 나머지 요소들로 구성되는 시퀀스 <s_2, ..., $s_{|S|}$>를 의미한다. <>은 요소가 존재하지 않는 널 시퀀스(null sequence)를 의미한다. 데이터베이스 내에 저장된 시퀀스를 데이터 시퀀스라 하고, 유사 검색 질의에 주어지는 시퀀스를 질의 시퀀스라 한다.

길이 n을 갖는 두 시퀀스 S와 Q의 유사한 정도를 측정하기 위하여 다음과 같은 거리 함수 Lp가 널리 사용된다. L_1은 맨하탄 거리(*Manhattan distance*), L_2는 유클리드 거리(Euclidean distance), L_∞은 대응되는 각 쌍의 거리 중 최대 거리를 의미한다. 거리 함수 Lp는 대상이 되는 두 시퀀스의 길이가 같아야 한다는 제한이 있다.

$$L_p(S,Q) = \left(\sum_{i=1}^{n} | s_i - q_i |^p \right)^{1/p}, \quad 1 \le p \le \infty.$$

두 시퀀스 S와 Q간의 타임 워핑 변환을 기반으로 한 타임 워핑 거리(time warping distance) D_{tw}는 다음과 같이 재귀적으로 정의된다.

정의 1:

(1) $D_{tw}(<>, <>) = 0$,

(2) $D_{tw}(S, <>) = D_{tw}(<>, Q) = \infty$,

(3) $D_{tw}(S, Q) = D_{base}(First(S), First(Q)) +$
MIN($D_{tw}(S, Rest(Q))$, $D_{tw}(Rest(S), Q)$,
$D_{tw}(Rest(S), Rest(Q))$)

여기서, MIN은 인자들 중 가장 작은 값을 가지는 것을 취하는 함수이며, D_{base}는 기본 거리 함수로서 Lp 중 임의의 것을 선택하여 사용할 수 있다. 타임 워핑 변환에서는 정의 1 (3)에서와 같이 요소 반복(stuttering)을 사용한다[20]. 요소 반복이란 두 시퀀스의 거리 차를 최소화하기 위하여 한 시퀀스 내의 임의의 요소를 반복시킴으로써 이 요소가 다른 시퀀스의 다수의 요소들과 매치되는 것을 허용하는 연산이다.

이러한 특성으로 인하여 D_{tw}는 요소 추출의 주기가 다르거나 길이가 다른 두 시퀀스의 유사 정도를 측정하는 응용에서 널리 사용된다. 이러한 응용에서는 질의 시퀀스와의 D_{tw}가 사용자에 의하여 주어진 값 ε 이하인 시퀀스들은 질의 시퀀스와 유사하다고 간주된다. 본 논문에서는 이와 같이 D_{tw}를 거리 함수로 사용하여 질의 시퀀스와 유사한 데이터 시퀀스들을 데이터베이스로부터 찾아내는 과정을 간략히 타임 워핑 지원 유사 검색(similarity search supporting time warping)이라 정의한다.

III. 제안하는 기법

3.1. 유사 검색 모델

본 연구에서는 서로 다른 길이를 가지는 두 시퀀스들의 유사한 정도를 나타내는 척도로서 정의 1에 나타난 타임 워핑 거리 D_{tw}를 사용한다. 특히, 요소 반복을 통하여 변환된 두 시퀀스간의 거리 함수 D_{base}로서 L_∞를 사용한다. 이를 위하여 타임 워핑 거리의 정의는 다음 정리 2와 같은 형태로 변형된다. 즉, 두 시퀀스간의 타임 워핑은 요소 반복을 통하여 변환된 두 시퀀스들의 서로 대응되는 요소 쌍 간의 거리를 최소화하기 위한 변환이며, 타임 워핑 거리는 요소 쌍의 거리들 중 최대 값을 의미한다. 질의 시퀀스 Q와 허용치 ε이 주어지는 유사 검색 질의에서 $D_{tw}(S, Q)$의 값이 ε 이하인 데이터 시퀀스 S들은 Q와 유사한 시퀀스로서 간주된다. 이는 S의 타임 워핑 변환된 시퀀스의 각 요소가 Q의 타임 워핑 변환된 시퀀스의 대응되는 요소의 일정 범위 ε 내에 존재함을 의미한다.

정의 2:

(1) $D_{tw}(<>, <>) = 0$,

(2) $D_{tw}(S, <>) = D_{tw}(<>, Q) = \infty$,

(3) $D_{tw}(S, Q) = MAX(|First(S) - First(Q)|,$
$MIN(D_{tw}(S, Rest(Q)), D_{tw}(Rest(S), Q),$
$D_{tw}(Rest(S), Rest(Q))))$

D_{base}로서 L_1을 사용하는 참고 문헌 [4][15][20]과 달리 본 연구에서 L_∞를 사용하는 주된 이유는 사용자의 질의 작성의 부담을 덜도록 하기 위해서이다. 정의 1에서 나타난 바와 같이 L_1을 사용하는 경우, 타임 워핑 거리는 변환된 두 시퀀스의 각 대응되는 요소 쌍의 거리들의 합으로 나타나므로 질의 시퀀스와 데이터 시퀀스의 길이에 큰 영향을 받는다. 따라서 질의를 작성하는 사용자가 해당 데이터베이스 특성에 맞는 적절한 ε을 결정한다는 것은 매우 어려운 일이다. 특히, 동적인 환경에서는 시퀀스의 길이가 계속 변경되므로 올바른 ε를 결정하는 것이 사실상 불가능하다. 반면, L_∞를 사용하는 경우, 시퀀스의 길이에 영향을 받지 않고 일관된 ε을 사용할 수 있으므로 사용자가 질의 작성의 부담을 덜 수 있다.

L_∞를 이용함으로써 얻을 수 있는 부가적인 장점은 빠른 질의 처리가 가능하다는 것이다. 타임 워핑 지원 유사 검색은 시퀀스 하나가 요소 반복을 통하여 많은 시퀀스들로 변환되므로 CPU 비용이 매우 크다. 따라서 타임 워핑 거리 계산 도중 최종 결과에 포함되지 않을 시퀀스들을 가능한 빨리 파악하는 것이 필요하다. L_1의 경우, 여러 요소 쌍들의 거리 합을 축적하여 이 값이 ε을 초과해야 이 시퀀스를 필터 아웃(filter out)하므로 많은 요소 쌍의 거리를 계산해야 한다. 반면, L_∞의 경우, 각 요소 쌍의 거리에 의하여 필터 아웃이 가능하므로 상대적으로 작은 CPU 비용으로 처리가 가능하다.

3.2. 인덱싱 전략

유사 검색에서는 정확한 질의 결과를 보장하기 위하여 착오 기각[1][9]을 방지하는 것이 매우 중요하다. 대부분의 인덱스 구조들은 사용되는 거리 함수가 삼각형 부등식 성질(triangle inequality)을 만족한다고 가정하며, 이 성질을 만족하지 못하는 거리 함수를 사용하는 경우 착오 기각을 유발하게 된다[20]. 참고 문헌 [20]에서는 타임 워핑 거리가 삼각형 부등식 성질을 만족하지 못함을 보였으며, 착오 기각을 허용하지 않는 응용에서는 타임 워핑 지원 유사 검색의 처리를 위하여 인덱스를 사용할 수 없음을 주장한 바 있다. 그러나 대용량의 데이터베이스 환경에서 이와 같이 인덱스를 사용하지 않는 경우, 검색 성능이 심각하게 저하된다.

본 연구에서는 이에 대한 해결 방법으로서 삼각형 부등식 성질을 만족하는 타임 워핑 거리의 하한 함수(lower bound function)를 고안하고, 이를 기반으로 인덱스를 구성하는 전략을 사용한다.

하한 함수를 정의하기 위해서는 이 함수에서 인자로 사용될 시퀀스의 특징들을 먼저 추출해야 한다. 특징 추출이 어려운 이유는 타임 워핑 거리를 계산하기 위하여 같은 시퀀스가 질의 시퀀스에 따라 다양한 형태로 변환되기 때문이다. 즉, 요소 반복을 적용하는 위치나 횟수에는 특별한 제약이 없으므로, 비교되는 질의 시퀀스에 따라 같은 시퀀스라도 다양한 길이와 요소 값을 갖는 새로운 시퀀스로 변환될 수 있다. 그러나 시퀀스로부터 추출되는 특징은 인덱스 구성을 목적으로 하므로 질의 시퀀스와 독립적인 고유의 성질을 가져야 한다. 이것은 특징 추출이 시퀀스를 인자로 하는 함수(function)의 형태로 표현되어야 함을 의미한다.

본 연구에서는 하한 함수를 위한 인자로서 사용될 시퀀스 S의 특징들로서 첫 값인 First(S), 마지막 값인 Last(S), 요소들 중 최대 값인 Greatest(S), 요소들 중 최소 값인 Smallest(S)를 선정한다. 이들은 주어진 질의 시퀀스와의 타임 워핑 거리 계산을 위한 어떠한 형태의 요소 반복에도 변하지 않는 고정된 특징들이다. 시퀀스 S의 네 특징들로 구성되는 4-티플 레코드를 Feature(S)라 표기한다. 이러한 특징들을 인자로 사용하는 타임 워핑 거리 D_{tw}의 하한 함수 D_{tw_lb}는 다음과 같이 정의된다.

정의 3:

$D_{tw\_lb}(S, Q) = L_\infty(Feature(S), Feature(Q))$ 여기서 Feature(S) = <First(S), Last(S), Greatest(S), Smallest(S)>, Feature(Q) = <First(Q), Last(Q), Greatest(Q), Smallest(Q)> 이다.

다음에는 정리 1과 정리 2을 이용하여 함수 D_{tw_lb}가 타임 워핑 거리 D_{tw}의 하한 함수인 동시에 삼각형 부등식을 만족함을 보이고자 한다. 정리 1의 증명을 위하여 다음의 보조 정리 1과 보조 정리 2를 이용한다.

보조 정리 1:

임의의 두 시퀀스 $S = <s_1, s_2, ..., s_n>$, $Q = <q_1, q_2, ..., q_m>$ 에 대하여 다음이 항상 성립한다.

$D_{tw}(S, Q) \geq L_\infty(<First(S), Last(S)>, <First(Q), Last(Q)>)$

증명 : 생략

보조 정리 2:

임의의 두 시퀀스 $S = <s_1, s_2, ..., s_n>$, $Q = <q_1, q_2, ..., q_m>$ 에 대하여 다음이 항상 성립한다.

$D_{tw}(S, Q) \geq L_\infty(<Greatest(S), Smallest(S)>, <Greatest(Q), Smallest(Q)>)$

증명 : 생략

정리 1:

임의의 두 시퀀스 $S = <s_1, s_2, ..., s_n>$, $Q = <q_1, q_2, ..., q_m>$ 에 대하여 다음이 항상 성립한다

$D_{tw}(S, Q) \geq D_{tw\_lb}(S, Q)$

증명 : 생략

정리 1을 이용하여 다음의 따름 정리 1을 쉽게 유도해 낼 수 있다.

따름 정리 1:

임의의 두 시퀀스 $S = <s_1, s_2, ..., s_n>$, $Q = <q_1, q_2, ..., q_m>$ 와 임의의 값 ε에 대하여 다음이 항상 성립한다.

$D_{tw}(S, Q) \leq \varepsilon \Rightarrow D_{tw\_lb}(S, Q) \leq \varepsilon$

정리 2:

임의의 세 시퀀스 X, Y, Z에 대하여 다음이 항상 성립한다.

$D_{tw\_lb}(X, Z) \leq D_{tw\_lb}(X, Y) + D_{tw\_lb}(Y, Z)$

증명:

$D_{tw_lb}(S, Q)) = L_\infty(Feature(S), Feature(Q))$이며, 거리 함수 L_∞은 항상 삼각형 부등식 성질을 만족하므로 [16] 정리 2는 항상 성립한다.

따름 정리 1은 유사 검색 질의를 처리할 때, D_{tw} 대신 D_{tw_lb}를 사용하는 경우에도 착오 기각이 발생하지 않음을 의미하는 것이다. 정리 2는 유사 검색 질의를 처리할 때, 삼각형 부등식 성질의 만족하는 D_{tw_lb}를 거리 함수로 하는 인덱스를 사용할 수 있음을 의미하는 것이다. 따라서 위의 두 정리들은 새로운 거리 함수 D_{tw_lb}를 기반으로 구성된 인덱스를 이용하여 타임 워핑 지원 유사 검색 질의를 착오 기각 없이 처리할 수 있음을 증명하는 것이다.

3.3. 알고리즘

3.3.1. 인덱스 구성

인덱스 구성을 위하여 각 시퀀스의 네 개의 특징을 사용하므로 각 시퀀스는 사차원 유클리드 공간상의 점으로 표현된다. 따라서 이러한 특징들을 효과적으로 검색하기 위한 인덱스 구조로는 R-트리[12], R+-트리[18], R*-트리[3], X-트리[5] 등 다차원 인덱스를 고려할 수 있다. 인덱스 구성 알고리즘은 우선 데이터베이스내의 각 시퀀스 S를 액세스하여 First(S), Last(S), Greatest(S), Smallest(S)를 구한 후, 이 특징들과 시퀀스 식별자(*identifier*)로 구성되는 엔트리를 주어진 다차원 인덱스 내에 삽입함으로써 인덱스를 구성한다.

3.3.2. 질의 처리

그림 3.2에 나타난 TW_Sim_Search는 사차원 인덱스를 이용하여 질의 시퀀스 Q와의 D_{tw}가 ε 이내인 유사한 시퀀스들을 데이터베이스로부터 검색하는 알고리즘을 나타낸 것이다. 단계 (1)에서는 질의 Q로부터 네 가지 특징들을 추출하고, 단계 (2)에서는 사차원 인덱스를 이용한 정사각형 형태의 영역 질의를 수행한다. 이때, 단계 (1)에서 구한 네 특징들은 영역

TW_Sim_Search(query sequence Q, tolerance ε)

(1) Get query features First(Q), Last(Q), Greatest(Q), Smallest(Q);
(2) Perform a range search on the four dimensional index using these features, ε, D_{tw_lb} as a query point, a range, and a distance function, respectively;
(3) Make a candidate set CandSet consisting of returned entries from Step (2);
(4) **FOR** each entry in CandSet **DO**
(5) Read the corresponding sequence S from the database;
(6) **IF** $D_{tw}(S, Q) \leq \varepsilon$, **THEN** return S;

그림 3.2. 타임 워핑 지원 유사 검색 질의 처리 알고리즘

질의의 중심점이 되며, ε은 질의의 범위가 된다. 또한, 거리 함수로는 D_{tw_lb}가 사용된다. 단계 (3)에서는 영역 검색의 결과로 반환된 엔트리들로 후보 집합을 구성한다. 단계 (4)~(6)은 후보 집합에 포함된 각 엔트리와 대응되는 시퀀스들에 대하여 최종 질의 결과로서의 적합성을 판정하는 것이다. 단계 (5)에서는 대응되는 시퀀스를 직접 데이터베이스로부터 읽어들이며, 단계 (6)에서는 이 시퀀스와 질의 시퀀스 Q간의 실제 D_{tw}가 ε이하이면 이를 최종 결과로 반환한다.

Ⅳ. 성능 분석

4.1. 실험 환경

본 연구에서는 실제 데이터 S&P_Data를 이용한 실험을 통하여 성능을 분석한다. S&P_Data는 미국의 S&P 500 주식 데이터이며, 평균 길이가 231인 545개의 시퀀스들로 구성된다.

질의 시퀀스는 데이터베이스로부터 임의로 하나의 시퀀스를 선택한 후, 각 요소 값에 적절한 범위[2] 내의 임의의 값을 선택하여 더하는 방식으로 변형하여 생성한다. 각 데이터에 대하여 100개의 유사 검색 질의를 수행한 후, 나타난 평균 수행 시간(elapsed time)을 성능 평가 지수로 사용한다.

성능 평가는 다음의 네 가지 서로 다른 기법들을 대상으로 한다. Ours는 본 논문에서 제안된 기법으로서 시퀀스의 네 가지 특징들을 대상으로 구성된 다차원 인덱스를 이용하는 방식이다. 다차원 인덱스로는 현재 시퀀스 데이터베이스 분야에서 가장 널리 채택되고 있는 R*-트리[3]를 사용하여 실험한다. 사용된 R*-트리는 Maryland 대학의 Faloutsos 교수 팀에서 개발한 R*-tree Version 2.0이며, 페이지 크기로서 1KB를 사용한다. 또한, 성능 비교를 위한 기존의 기법으로서 Naive_Scan[4], LB_Scan[20], ST_Filter[3] [15]의 세 가지를 사용한다. FastMap 방식[20]은 착오 기각을 유발한다는 기능상의 문제를 가지므로, 성능 비교의 대상에서 제외한다.

4.2. 실험 결과 및 분석

먼저, 실험 1에서는 후보 비율(candidate ratio)을 기준으로 각 기법들의 성능을 비교한다. 후보 비율이란 전체 시퀀스 수에 대한 후보 시퀀스 수로 정의된다. ST_Filter와 Ours에서는 서픽스 트리와 R*-트리의 탐색 후 나타나는 후보 시퀀스들을 대상으로 하며, LB_Scan에서는 하한 함수 D_{lb}에 의하여 후보로 채택되는 시퀀스들을 대상으로 한다. 단, Naive_Scan의 경우에는 별도의 필터링 작업이 존재하지 않으므로, 최종 결과로 반환되는 시퀀스들을 대상으로 표기한다. 이 실험의 목적은 착오 채택(false alarm)[1][10]의 경향을 관찰함으로써 각 기법의 필터링 효과를 비교하기 위한 것이다.

그림 5.1는 S&P_Data에 대하여 각 기법들을 각각 적용한 실험 결과를 나타낸 것이다. 가로축은 허용치 ε을 나타내며, 세로축은 필터링 비율을 나타낸다. 허용치가 2에서 6까지 변화함에 따라 전체 시퀀스에 대한 최종 결과로 반환되는 시퀀스의 비율은 0.2%(≒ 1.1개)에서 1.7%(≒ 9.3개)까지 변화함을 볼 수 있다.

2) 선택된 시퀀스 내에 속하는 요소 값들의 표준 편차를 std라 할 때, 이 범위는 [-std/10, std/10]이다.

3) ST_Filter를 위한 최적의 도메인 분류를 위하여 다수의 선행 실험을 수행하였으며, 이 결과 각 도메인이 100개의 등 간격 구간(equal-length interval)을 갖도록 설정하였다.

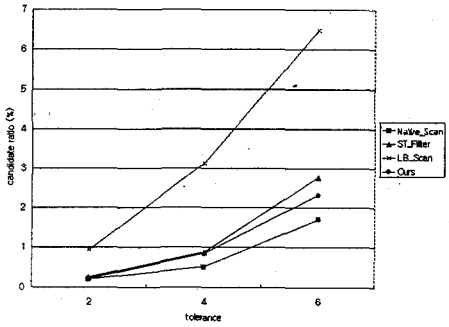

그림 5.1. S&P_Data를 이용한 필터링 효과의 비교

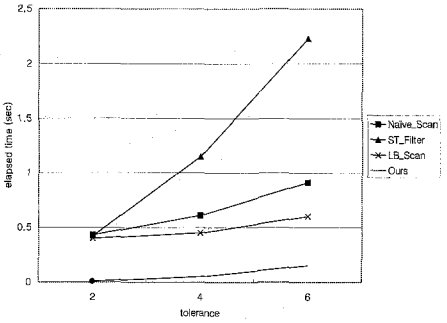

그림 5.2. S&P_Data를 이용한 실행 시간의 비교

실험 결과에 의하면, 제안하는 기법의 필터링 효과가 가장 뛰어난 것으로 나타났으며, ST_Filter의 필터링 효과가 제안하는 기법의 필터링 효과에 근접하는 것으로 나타났다. 반면, LB_Scan의 필터링 효과는 제안하는 기법 및 ST_Filter와 비교하여 상당히 떨어지는 것으로 나타났다.

실험 1에서 관찰한 필터링 효과가 전체 검색 성능과 완전히 일치하지는 않는다. 그 이유는 Naive_Scan과 LB_Scan에서는 디스크내의 전체 시퀀스들을 액세스하는 비용이 고려되어야 하며, 제안하는 기법과 ST_Filter에서는 R*-트리와 서픽스-트리를 탐색하는 비용이 고려되어야 하기 때문이다. 따라서 실험 2에서는 각 기법의 질의 처리를 위한 전체 실행 시간(elapsed time)을 비교한다. 그림 5.2는 실험 1과 동일한 데이터 및 질의 시퀀스들을 사용하여 수행한 실험 결과를 나타낸 것이다. 가로축은 허용치 ε을 나타내며, 세로축은 실행 시간을 나타낸다.

실험 결과에 의하면, ST_Filter는 Naive_Scan보다도 떨어지는 성능을 가지는 것으로 나타났다. 그 근본적인 이유는 ST_Filter가 공통 심볼들이 많이 발생하는 서브시퀀스 환경을 대상으로 고안된 기법이기 때문이다. 즉, 많은 공통 심볼들을 포함하는 경우 ST_Filter에서 사용하는 서픽스 트리의 크기는 작아지나, 그렇지 않은 경우에는 그 크기가 매우 커진다. ST_Filter는 질의 처리 시 트리 내 많은 경로들을 점검하게 되므로 트리가 커질수록 검색 성능이 크게 저하된다. 따라서 ST_Filter는 서브시퀀스 매칭에는 유용하나, 전체 매칭에서는 적합하지 않다.

기존의 기법들 중에서는 LB_Scan이 가장 좋은 성능을 가지는 것으로 나타났다. 전체 시퀀스들을 액세스한다는 점에서는 LB_Scan과 Naive_Scan가 동일하지만, LB_Scan는 하한 함수를 사용함으로써 CPU 비용을 절감할 수 있기 때문이다. S&P_Data가 약 850KB의 소규모이므로, 이러한 CPU 비용 절감 효과가 전체 성능에 반영된 것이다. 제안된 기법은 LB_Scan에 비교하여 허용치에 따라 약 4배에서 43배까지 나은 성능을 보였다. 이것은 제안된 기법이 데이터의 4% 미만의 작은 R*-트리의 극히 일부분만을 탐색하며, 이 탐색에 의한 필터링 효과가 매우 뛰어남을 의미하는 것이다. 또한, 이러한 성능 개선 효과는 허용치가 작아질수록 더욱 두드러짐을 볼 수 있다. 실제 응용에서 요구하는 질의 결과의 수가 일반적으로 작다는 것을 고려할 때, 이러한 경향은 매우 바람직한 것이다.

V. 결론

본 논문에서는 착오 기각 발생의 방지와 빠른 검색 성능을 동시에 보장하는 새로운 타임 워핑 지원 유사 검색 처리 기법을 제안하였다. 제안된 기법은 먼저 새롭게 고안된 거리 함수 Dtw_lb를 이용하여 거리 함수 기반 다차원 인덱스를 구성하고, 이를 이용하여 타임 워핑 지원 유사 검색을 빠르게 처리한다. Dtw_lb가 Dtw의 하한 함수인 동시에 삼각형 부등식 성질을 만족한다는 것을 보임으로써 제안된 기법에서 착오 기각이 발생하지 않음을 증명하였다. 제안된 기법은 거리 함수를 기반으로 하는 최초의 인덱스 기반 타임 워핑 지원 유사 검색 기법이라는 점에서 큰 의미가 있다.

제안된 기법의 성능을 평가하기 위하여 기존의 기법들과의 다양한 실험을 통한 성능 비교를 수행하였다. 실험 결과에 의하면, 제안된 기법은 기존의 기법들과 비교하여 실제 주식 데이터를 대상으로 하는 경우 4배에서 43배까지의 성능 개선 효과를 보였으며, 대규모의 합성 데이터를 대상으로 하는 경우 19배에서 720배까지의 성능 개선 효과를 보였다. 이러한 성능 개선 효과는 (1) 시퀀스들의 수가 많아질수록, (2) 시퀀스의 길이가 길어질수록, (3) 질의에서 사용되는 허용치가 작아질수록 더욱 증가하는 것으로 나타났다. 실제 데이터베이스 특성을 고려할 때, 이들은 제안된 기법의 유용성을 보여주는 바람직한 경향

이다.

제안된 기법의 기본 아이디어를 서브시퀀스 매칭에도 그대로 적용할 수 있다. 제안된 기법은 시퀀스의 수가 많을수록 성능 개선 효과가 커지므로, 서브시퀀스 매칭의 경우 그 효용성이 더욱 커질 것으로 예상된다.

Acknowledgment

본 논문은 정보통신부에서 주관하는 정보통신 우수시범학교 지원사업에 의하여 수행되었음.

참고 문헌

[1] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases," In Proc. Int'l. Conference on Foundations of Data Organization and Algorithms, FODO, pp. 69-84, Oct. 1993.

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

[3] N. Beckmann 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, May 1990.

[4] D. J. Berndt and J. Clifford, "Finding Patterns in Time Series: A Dynamic Programming Approach," Advances in Knowledge Discovery and Data Mining, pp. 229-248, 1996.

[5] S. Berchtold, D. A. Keim, and H.-P. Kriegel, "The X-tree: An Index Structure for High-Dimensional Data," In Proc Int'l. Conf. on Very Large Data Bases, VLDB, pp. 28-39, 1996.

[6] Chen, M. S., Han, J., and Yu, P. S., "Data Mining: An Overview from Database Perspective," IEEE Trans. on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 866-883, 1996.

[7] K. K. W. Chu, and M. H. Wong, "Fast Time-Series Searching with Scaling and Shifting," In Proc. Int'l. Symp. on Principles of Database Systems, ACM PODS, pp. 237-248, May 1999.

[8] G. Das, D. Gunopulos, and H. Mannila, "Finding Similar Time Series," In Proc. European Symp. on Principles of Data Mining and Knowledge Discovery, PKDD, pp. 88-100, 1997.

[9] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast Subsequence Matching in Time-series Databases," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp. 419-429, May 1994.

[10] C. Faloutsos and K. I. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp. 163-174, 1995.

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

[12] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," In Proc. Int'l. Conf. on Mangement of Data, ACM SIGMOD, pp. 47-57, 1984.

[13] W. K. Loh, S. W. Kim, and K. Y. Whang, "Index Interpolation: A Subsequence Matching Algorithm Supporting Moving Average Transform of Arbitrary Order in Time-Series Databases," IEICE Trans. on Information and Systems, 2000. (accepted to appear)

[14] W. K. Loh, S. W. Kim, and K. Y. Whang, "Index Interpolation: An Approach for Subsequence Matching Supporting Normalization Transform in Time-Series Databases, 2000. (submitted for publication)

[15] S. H. Park et al., "Efficient Searches for Similar Subsequences of Difference Lengths in Sequence Databases," In Proc. Int'l. Conf. on Data Engineering, IEEE, pp. 23-32, 2000.

[16] F. P. Preparata and M. Shamos, Computational Geometry: An Introduction, Springer-Verlag, 1985

[17] D. Rafiei and A. Mendelzon, "Similarity-Based Queries for Time-Series Data," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp. 13-24, 1997.

[18] T. K. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+-Tree: A Dynamic Index for

Multi-Dimensional Objects," In Proc. Int'l. Conf. on Very Large Data Bases, VLDB, 507-518, 1987.

[19] G. A. Stephen, String Searching Algorithms, World Scientific Publishing, 1994.

[20] B. K. Yi, H. V. Jagadish, and C. Faloutos, "Efficient Retrieval of Similar Time Sequences Under Time Warping," In Proc. Int'l. Conf. on Data Engineering, IEEE, pp. 201-208, 1998.