

시계열 데이터베이스에서 복수의 모델을 지원하는 모양 기반 서브시퀀스 검색

원 정 임* · 윤 지 희** · 김 상 욱*** · 박 상 현****

요 약

모양 기반 검색이란 실제 요소 값과 관계없이 질의 시퀀스와 유사한 모양을 갖는 시퀀스(서브시퀀스)를 데이터베이스 내에서 검색하여 내는 연산이다. 본 논문에서는 시계열 데이터베이스에서의 모양 기반 검색을 위한 유연성 있는 새로운 유사 모델을 정의하고, 이 유사 모델을 지원하기 위한 인덱싱 및 질의 처리 방안을 제시한다. 제안된 유사 모델에서는 정규화, 이동 평균, 타임 워핑 등 다양한 변환을 지원한다. 특히 최종 유사 정도를 계산하기 위하여 사용되는 L_p 거리 함수를 사용자가 임의로 지정하도록 함으로써 응용에서 선호하는 유사 모델을 반영할 수 있다. 또한 이러한 모양 기반 검색을 효과적으로 지원하기 위한 압축된 서브시퀀스 트리 구조를 제안하고, 이를 기반으로 하는 효율적인 질의 처리 기법을 제시한다. 실험 결과에 의하면 제안된 기법은 질의 시퀀스와 모양이 유사한 서브시퀀스들을 사용자에게 의하여 선택된 거리 함수를 사용하여 성공적으로 검색할 뿐 아니라, 순차 검색과 비교하여 거리 함수 선택에 따라 수 백배까지의 성능 개선 효과를 갖는 것으로 나타났다.

Shape-Based Subsequence Retrieval Supporting Multiple Models in Time-Series Databases

Jung-Im Won* · Jee-Hee Yoon** · Sang-Wook Kim*** · Sang-Hyun Park****

ABSTRACT

The shape-based retrieval is defined as the operation that searches for the (sub) sequences whose shapes are similar to that of a query sequence regardless of their actual element values. In this paper, we propose a similarity model suitable for shape-based retrieval and present an indexing method for supporting the similarity model. The proposed similarity model enables to retrieve similar shapes accurately by providing the combination of various shape-preserving transformations such as normalization, moving average, and time warping. Our indexing method stores every distinct subsequence concisely into the disk-based suffix tree for efficient and adaptive query processing. We allow the user to dynamically choose a similarity model suitable for a given application. More specifically, we allow the user to determine the parameter p of the distance function L_p when submitting a query. The result of extensive experiments revealed that our approach not only successfully finds the subsequences whose shapes are similar to a query shape but also significantly outperforms the sequence search.

키워드: 시퀀스 데이터베이스(Sequence Databases), 유사 검색(Similarity Search), 모양 기반 검색(Shape-Based Retrieval), 인덱싱(Indexing), 타임 워핑(Time Warping)

1. 서 론

시계열 데이터베이스(time-series 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, 제

품 판매량 데이터, 기업 성장률 데이터 등이 있다[2,3]. 유사 검색(similarity search)이란 주어진 질의 시퀀스(query sequence)와 변화의 패턴이 유사한 시퀀스들을 시퀀스 데이터베이스로부터 찾아내는 연산이다[1-3]. 이러한 유사 검색은 데이터 마이닝(data mining) 및 데이터 웨어하우스(data warehousing) 분야에서 중요한 연산으로 사용된다[4,5].

유사 검색에 관한 기존의 많은 연구에서는 길이 n 의 시퀀스를 n 차원 공간상의 한 점으로 간주한다. 또한, 두 시퀀스들간의 유사한 정도를 측정하기 위하여 두 점들간의 유클리드 거리(Euclidean distance) L_2 를 이용한다[1,3-7].

유클리드 거리만을 이용한 유사 검색을 통해서 사용하는 사용자

※ 본 논문은 2002년도 한림대학교 교비 연구비(HRF-2002-38), 정보통신부에서 시행한 2001년도 대학기초연구 지원사업의 지원을 받았습니다.

* 준 회원: 한림대학교 대학원 컴퓨터공학과

** 정 회원: 한림대학교 정보통신공학부 교수

*** 정 회원: 한양대학교 정보통신공학부 교수

**** 정 회원: 포항공과대학교 컴퓨터공학과 교수

논문접수: 2002년 11월 5일, 심사완료: 2003년 3월 28일

가 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. 참고문헌[1, 3, 8] 등에서는 변환을 지원하지 않았으나, 이후에는 정규화(normalization)[2, 6, 7, 9, 10], 이동 평균(moving average)[4, 5, 11], 타임 워핑(time warping)[12-16] 등의 다양한 변환을 지원하는 방법들이 제안되었다.

본 논문에서는 시계열 데이터베이스로부터 주어진 질의 시퀀스와 유사한 모양을 갖는 시퀀스를 검색하는 문제를 다루고자 한다. 본 논문에서는 이러한 검색 문제를 모양 기반 검색(shape-based retrieval)이라 정의한다. 본 연구에서는 모양 기반 검색을 지원하기 위하여 기존의 값 기반 검색에서 사용하던 다양한 형태의 변환들의 조합을 이용하고자 한다. 즉, 시프팅 변환, 스케일링 변환, 타임 워핑 변환, 이동 평균 변환 등 다양한 형태의 변환을 동시에 지원하는 유사 모델을 제안함으로써 효과적인 모양 기반 검색을 가능하도록 한다.

특히, 본 유사 모델에서는 최종 변환된 두 시퀀스간의 유사 정도를 측정할 때, 사용자가 원하는 다양한 L_p 거리 함수를 지원한다. 즉, 맨해튼 거리(Manhattan distance) L_1 , 유클리드 거리(Euclidean distance) L_2 , 대응되는 각 쌍의 거리 중 최대 거리인 L_∞ 중 사용자가 선호하는 것을 선택하면, 제안하는 기법에서는 선택된 거리 함수를 사용하여 모양 기반 검색을 수행한다. 응용에 따라 사용자의 유사 정도에 대한 관점이 다르므로, 이러한 다중 거리 함수의 지원은 매우 유용하다. 제안하는 기법의 중요한 특징은 사전에 구성된 단 하나의 인덱스를 이용하여 세 가지 서로 다른 거리 함수를 모두 지원할 수 있다는 점이다.

유사 검색은 다음과 같이 전체 검색(whole retrieval)과 서브시퀀스 검색(subsequence retrieval)으로 구분된다[3].

- 전체 검색 : 데이터베이스 D 내에 존재하는 시퀀스 $S_1, S_2, \dots, S_N$ 에 대하여 질의 시퀀스 Q와 허용치 ϵ 이 주어질 때, D로부터 Q와 유사한 시퀀스 S_i 를 검색한다. 이때, $S_1, S_2, \dots, S_N$ 및 Q의 길이는 동일해야 한다.
- 서브시퀀스 검색 : 데이터베이스 D 내에 존재하는 시퀀스 $S_1, S_2, \dots, S_N$ 에 대하여 질의 시퀀스 Q와 허용치 ϵ 이 주어질 때, D로부터 Q와 유사한 서브시퀀스를 포함하는 시퀀스 S_i 와 이 서브시퀀스가 S_i 내에서 시작하는 위치를 검색한다. 이때, $S_1, S_2, \dots, S_N$ 및 Q는 서로 다른 길이를 갖는 것이 허용된다.

모든 시퀀스들의 길이가 동일하다는 전제는 비현실적이므로, 서브시퀀스 검색은 전체 검색에 비하여 다양한 실제 분야에 적용될 수 있다.

본 논문에서는 모양 기반 서브시퀀스 검색을 위한 새로

운 기법을 제안한다. 본 논문의 공헌은 다음과 같다. ① 효과적인 모양 기반 서브시퀀스 검색을 위한 새로운 유사 모델을 정의한다. ② 이 모델을 기반으로 하는 모양 기반 유사 검색을 효율적으로 처리하기 위한 인덱싱 방안을 제안한다. ③ 제안된 인덱싱 방안을 이용한 질의 처리 방안을 제안한다. ④ 제안된 모양 기반 서브시퀀스 검색 기법의 우수성을 규명하기 위하여 다양한 실험들의 수행을 통하여 나타난 검색 결과의 질과 검색 성능을 제시한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로서 유사 검색 분야와 연관된 기존의 연구 결과를 간략히 기술한다. 제 3장에서는 논의 전개에 필요한 용어 및 기호, 그리고 본 논문에서 제시하는 유사 모델을 정의한다. 제 4장에서는 제안한 유사 모델을 기반으로 하는 질의를 효과적으로 처리하기 위한 인덱싱 기법을 제시한다. 제 5장에서는 제시한 인덱싱 기법을 기반으로 하는 질의 처리 기법을 제안한다. 제 6장에서는 제안하는 기법의 우수성을 규명하기 위한 성능 평가 결과를 제시한다. 끝으로, 제 7장에서는 본 논문을 요약하고, 결론을 내린다.

2. 관련 연구

본 장에서는 시계열 데이터베이스에서의 유사 검색에 관련된 기존의 연구 결과에 관하여 간략히 요약한다.

참고문헌[1]에서는 시계열 데이터베이스를 위한 전체 검색 기법을 제안하였다. 먼저, 길이 1인 각 데이터 시퀀스를 이산 푸리에 변환(discrete Fourier transform : DFT)을 이용하여 저차원 f -공간($f \ll 1$) 상의 점으로 변환하고, 이들을 R^* -트리[17]에 저장함으로써 인덱싱을 수행한다. 전체 검색을 위하여 길이가 1인 질의 시퀀스를 위와 동일한 방법으로 저차원 f -공간상의 한 점으로 변환하고, R^* -트리에 대하여 변환한 질의 점을 이용한 범위 질의를 수행함으로써 질의 시퀀스와 유사할 가능성이 큰 후보 시퀀스 집합(candidate sequence set)을 구성한다. 끝으로, 각 후보 시퀀스를 디스크로부터 액세스하여 질의 시퀀스와의 실제 유클리드 거리를 조사함으로써 최종 질의 결과를 반환한다.

참고문헌[3, 8]에서는 전체 검색 기법을 확장한 서브시퀀스 검색 기법을 제안하였다. 두 기법은 미리 지정된 고정된 길이 w 를 갖는 윈도우(window) 개념을 기반으로 다음과 같이 동작한다. 먼저, 인덱싱 구성을 위하여 각 시퀀스로부터 길이 w 의 윈도우들을 추출하고, 각 윈도우를 DFT 혹은 웨이블릿 변환(wavelet transform)을 이용하여 저차원 f -공간($f \ll w$) 상의 윈도우 점(window point)으로 변환한다. 효과적인 서브시퀀스 검색을 위하여 이러한 윈도우 점들을 R^* -트리에 저장한다. 서브시퀀스 검색을 위하여 길이 1인 질의 시퀀스로부터 길이 w 의 윈도우들을 추출하고, 각 윈도우를 저차원 f -공간상의 윈도우 점으로 변환한다. R^* -트

리에 대하여 변환된 각 윈도우 점을 이용하여 범위 질의를 수행함으로써 질의 시퀀스와 유사할 가능성이 높은 많은 후보 서브시퀀스 집합(candidate subsequence set)을 구성한다. 그 다음 후보 서브시퀀스를 포함하는 각 시퀀스를 디스크로부터 액세스하여 후보 서브시퀀스와 질의 시퀀스의 실제 유클리드 거리를 확인함으로써 최종 질의 결과를 반환한다.

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. 참고문헌[1, 3, 8] 등에서는 변환을 지원하지 않았으나, 이후에는 다양한 변환을 지원하는 방법들이 제안되었다.

참고문헌[2, 6, 7, 9, 10] 등에서는 유사 모델 상에서 정규화(normalization) 변환을 지원하는 방법들을 제안하였다. 정규화 변환을 지원하는 유사 모델에서는 시퀀스간의 절대적인 유클리드 거리에 관계없이 요소 값들의 상대적인 변화 패턴이 유사한 시퀀스들을 유사하다고 판정한다. 참고문헌[9]에서는 데이터베이스내의 전체 시퀀스들을 차례로 조사함으로써 유사한 시퀀스를 검색하는 전체 검색 방법을 제안하였다. 참고문헌[2, 7]에서는 R^* -트리를 이용하여 전체 검색의 속도를 크게 개선시키는 방법을 제안하였다. 또한, 참고문헌[2, 7, 9]에서는 전체 검색만을 다루었으나, 참고문헌[6, 11]에서는 이러한 개념을 확장하여 인덱스를 사용하여 서브시퀀스 검색을 효과적으로 처리할 수 있는 방법을 제시하였다.

참고문헌[4, 5, 10]에서는 유사 모델에서 이동 평균(moving average) 변환을 지원하는 방법들을 제안하였다. 이동 평균 변환은 시퀀스의 연속되는 k 개 요소 값들의 평균값을 순차적으로 나열하도록 하는 변환이다. 여기서, k 값을 이동 평균 계수(moving average coefficient)라 부른다. 참고문헌[4]에서는 컨벌루션 정리의(convolution theorem)[18]를 기반으로 하나의 다차원 인덱스만을 이용하여 임의 계수의 이동 평균 변환을 지원하는 전체 검색 방법을 제안하였다. 참고문헌[5]에서는 이 개념을 확장하여 다수의 이동 평균 계수를 동시에 지원하는 이동 평균 기반 유사 모델을 정의하고, 다차원 인덱스를 한번만 검색함으로써 전체 검색을 처리하는 방법을 제안하였다. 참고문헌[10]에서는 기존의 방법을 서브시퀀스에 직접 적용하는 경우의 문제점을 지적하고, 인덱스 보간법(index interpolation)이라는 개념을 이용하여 서브시퀀스 검색을 효과적으로 처리할 수 있는 방법을 제시하였다.

참고문헌[12-16]에서는 유사 모델에서 타임 워핑(time warping) 변환을 지원하는 문제를 다루었다. 타임 워핑은 시퀀스내의 각 요소 값을 임의의 수만큼 반복시키는 것을

허용하는 변환이며, 시퀀스들의 길이가 서로 달라서 유클리드 거리를 이용하여 유사 정도를 직접 측정할 수 없는 경우에 매우 유용하다. 참고문헌[12, 13]에서는 데이터베이스내의 전체 시퀀스들을 차례로 조사함으로써 유사한 시퀀스를 검색하는 전체 검색 방법을 제안하였다. 참고문헌[13-15]은 보다 빠른 전체 검색을 위하여 인덱스를 이용하는 방법에 관한 연구 결과를 제시하였다. 또한, 참고문헌[16]에서는 서브시퀀스 검색의 효율적인 처리를 위하여 참고문헌[15]의 기본 아이디어를 확장한 방법을 제안하였다.

시계열 데이터베이스 응용에서는 위와 같이 하나의 변환만을 지원하는 것이 아니라 실제 요소 값과 관계없이 주어진 질의 시퀀스와 유사한 모양을 갖는 시퀀스를 효과적으로 검색하는 것을 요구하는 경우가 빈번하게 발생한다. 본 논문에서는 이러한 검색 문제를 모양 기반 검색(shape-based retrieval)이라 정의한다.

참고문헌[19]에서는 모양 정의 언어(shape definition language: SDL)를 제안하고, 질의시 사용자가 이를 이용하여 특정한 패턴을 정의하도록 하였다. 질의 처리에서는 계층적인 인덱스 구조(hierarchical index structure)를 사용함으로써 SDL로 정의된 패턴을 만족하는 서브시퀀스들을 효과적으로 검색한다. 이 방식은 시퀀스의 값을 심볼의 형태로 바꾼 후, 원래의 값은 무시한 채 심볼들만을 대상으로 검색을 수행한다. 따라서 거의 유사한 두 값이 서로 다른 심볼들로 변환되는 경우, 이 두 값을 각각 포함하는 유사한 두 시퀀스가 유사하지 않은 것으로 간주되는 문제점을 갖는다.

참고문헌[20]에서는 유사-기반 패턴 질의 모델인 랜드마크 모델(landmark model)을 제안하였다. 랜드마크 모델은 두 시퀀스들로부터 중요한 의미를 갖는 요소인 랜드마크들을 각각 추출하고, 실제 시퀀스 요소 값이 아닌 랜드마크들을 대상으로 하여 유사한 정도를 측정한다. 이 모델을 이용함으로써 실제 요소 값에 관계없이 사람들이 생각하는 바와 같은 방식으로 모양이 유사한 시퀀스들을 찾아 낼 수 있다. 그러나 랜드마크들은 각 응용에 따라 다르므로, 해당 응용에 적합한 랜드마크를 찾아내는 것은 간단한 일이 아니다.

3. 유사 모델

본 장에서는 사용하는 용어 및 기호를 정의하고, 본 논문에서 채택하는 유사 모델(similarity model)과 모양 기반 유사 서브시퀀스 검색 문제를 정의한다.

3.1 기호 및 용어 정의

<표 1>은 본 논문에서 사용하는 기본적인 기호를 정리한 것이다. 데이터베이스 내에 저장된 시퀀스를 데이터 시퀀스라 하고, 질의에 주어지는 시퀀스를 질의 시퀀스라 한다.

<표 1> 기호 정의

기 호	정 의
$S = (s[i])$	데이터 시퀀스 ($0 \leq i < \text{Len}(S)$), $\text{Len}(S)$ 는 S에 포함되는 요소 값의 수
$X = (x[i])$	S에 포함되는 임의의 서브시퀀스 ($0 \leq i < \text{Len}(X) \leq \text{Len}(S)$)
$Q = (q[i])$	질의 시퀀스 ($0 \leq i < \text{Len}(Q)$)
ϵ	유사 허용치
$\text{First}(S)$	시퀀스 S의 첫 번째 요소 값 $s[0]$
$\text{Rest}(S)$	시퀀스 S에서 $s[0]$ 을 제외한 요소들로 구성된 시퀀스 ($s[1], s[2], \dots, s[\text{Len}(S)-1]$)
$\text{Max}(S)$	시퀀스 S내의 크기가 최대인 요소 값
$\text{Min}(S)$	시퀀스 S에서 크기가 최소인 요소 값
$()$	요소가 존재하지 않는 널 시퀀스(null sequence)
$\max(a, b)$	a와 b의 값 중 최대 값
$\min(a, b, c)$	a, b, c의 값 중 최소 값

정의 1

시퀀스 $S = (s[i])$ ($0 \leq i < \text{Len}(S)$)를 정규화(normalization) 변환한 시퀀스 $\text{Norm}(S) = (s'[i])$ ($0 \leq i < \text{Len}(S)$)는 다음과 같이 정의된다[2].

$$s'[i] = \frac{s[i] - \frac{\text{Max}(S) + \text{Min}(S)}{2}}{\frac{\text{Max}(S) - \text{Min}(S)}{2}}$$

□

정규화 변환의 정의로서 S내의 요소 값들의 평균 $\text{avg}(S)$ 와 표준 편차 $\text{std}(S)$ 를 이용하는 $s'[i] = (s[i] - \text{avg}(S))/\text{std}(S)$ 를 이용할 수도 있다[4, 7]. 그러나 본 연구에서는 정규화 변환 후의 요소 값의 상한과 하한을 각각 +1과 -1로 고정하기 위하여 정의 1을 사용한다. 변환 후의 모든 요소 값들이 +1과 -1 사이에만 분포하게 되므로, 제 4장에서 언급하는 도메인 분류(categorization)를 통한 서브시퀀스 트리의 압축 과정에서 좋은 효과를 얻을 수 있다.

정규화 변환을 통하여 해당 시퀀스가 갖는 요소 값의 절대적인 크기를 무시할 수 있다. 따라서 정규화 변환은 요소 값의 크기는 서로 다르지만 변화하는 패턴이 유사한 시퀀스들을 파악하는데 매우 유용하다.

정의 2

시퀀스 $S = (s[i])$ ($0 \leq i < \text{Len}(S)$)를 이동 평균 계수(moving average coefficient) k ($1 \leq k < \text{Len}(S)$)로 이동 평균(moving average) 변환한 시퀀스 $\text{MV}_k(S) = (s_k[j])$ ($0 \leq j < \text{Len}(S)-k+1$)는 다음과 같이 정의된다[21, 22].

$$\begin{aligned} s_k[j] &= \frac{1}{k} \times (s[j] + s[j+1] + \dots + s[j+k-1]) \\ &= \frac{1}{k} \times \sum_{i=j}^{j+k-1} s[i] \end{aligned}$$

□

정의 2에 나타난 바와 같이, 이동 평균 변환은 시퀀스의

연속되는 k개의 요소 값들의 평균값들을 순차적으로 나열하는 변환이다. 이동 평균 변환을 통하여 시퀀스 내에서 나타나는 잡음(noise)의 영향을 제거할 수 있다. 따라서 이동 평균 변환은 잡음의 영향 없이 전체적인 변화 경향이 유사한 시퀀스들을 파악하는데 매우 유용하다. 이동 평균 계수 k는 해당 응용에서 잡음의 영향을 줄이고자 하는 정도에 따라 적절하게 선택된다.

정의 3

길이 n을 갖는 두 시퀀스 S와 Q의 유사한 정도를 측정하기 위한 거리 함수 L_p 는 다음과 같이 정의된다. 여기서, L_1 은 맨하탄 거리(Manhattan distance), L_2 는 유클리드 거리(Euclidean distance), L_∞ 은 대응되는 각 쌍의 거리 중 최대 거리를 의미한다[23].

$$L_p(S, Q) = \left(\sum_{i=1}^n |s[i] - q[i]|^p \right)^{1/p}, \quad 1 \leq p \leq \infty$$

□

거리 함수 L_p 는 현재 많은 응용에서 널리 사용되고 있으나, 비교 대상이 되는 두 시퀀스의 길이가 같아야 한다는 제한이 있다[14, 15]. 타임 워핑은 시퀀스내의 각 요소 값을 임의의 수만큼 반복시킴으로써 서로 다른 길이의 두 시퀀스간의 유사한 정도를 측정할 수 있도록 허용하는 변환이다[13]. 예를 들어, 서로 다른 두 시퀀스 $S_1 = (20, 21, 21, 20, 20, 23, 23, 23)$ 과 $S_2 = (20, 20, 21, 20, 23)$ 가 타임 워핑 변환에 의하여 동일한 시퀀스 $S_3 = (20, 20, 21, 21, 20, 20, 23, 23, 23)$ 으로 변환된다.

정의 4

타임 워핑 변환 후의 두 시퀀스 S와 Q간의 거리인 타임 워핑 거리(time warping distance) D_{tw} 는 다음과 같이 재귀적으로 정의된다[24]. 여기서, 거리 함수 L_p 는 응용에 적합한 임의의 것을 선택하여 사용할 수 있다.

- ① $D_{tw}(), () = 0$,
- ② $D_{tw}(S, ()) = D_{tw}(), Q = \infty$,
- ③ $D_{tw}(S, Q) = (|L_p(\text{First}(S), \text{First}(Q))|^p + |\min(D_{tw}(S, \text{Rest}(Q)), D_{tw}(\text{Rest}(S), Q), D_{tw}(\text{Rest}(S), \text{Rest}(Q)))|^p|)^{1/p}$

□

타임 워핑은 데이터베이스내의 시퀀스들의 길이가 서로 달라서 유클리드 거리를 이용하여 유사 정도를 직접 측정할 수 없는 경우에 매우 유용하다. 타임 워핑 변환은 정의 4 ③에서와 같이 두 시퀀스의 거리 차를 최소화하기 위하여 한 시퀀스 내의 임의의 요소를 반복시킴으로써 이 요소가 다른 시퀀스의 다수의 요소들과 대응되는 것을 허용한다. D_{tw} 의 값은 동적 프로그래밍 기법(dynamic programming)에 의한

거리 축적 테이블(cumulative distance table)을 이용하여 효율적으로 계산될 수 있다[12, 24].

3.2 유사 모델의 정의

본 연구의 목적은 대량의 시계열 데이터베이스로부터 질의로 주어진 시퀀스와 유사한 모양을 갖는 서브시퀀스를 효과적으로 검색하는 기능을 지원하는 것이다. 본 연구에서는 이러한 모양 기반 유사 검색을 지원하기 위하여, 앞에서 언급한 정규화 변환, 이동 평균 변환, 타임 워핑 변환을 모두 지원하는 다음과 같은 유사 정도 측정 함수를 채택한다.

정의 5

두 시퀀스 혹은 서브시퀀스 S와 Q간의 유사한 정도 $D(S, Q)$ 는 다음과 같이 정의된다.

$$D(S, Q) = D_{tw}(\text{Norm}(\text{MV}_k(S)), \text{Norm}(\text{MV}_k(Q)))$$

□

정의 5에 나타난 바와 같이, S와 Q간의 유사한 정도 $D(S, Q)$ 는 S와 Q를 각각 ① k-이동 평균 변환, ② 정규화 변환, ③ 타임 워핑 변환을 차례로 거친 후의 거리로 정의된다. 본 논문에서 제안하는 유사 모델을 기반으로 한 모양 기반 유사 시퀀스 검색 기법은 시프팅(shifting), 스케일링(scaling), 타임 워핑(time warping) 등 다양한 변환을 지원할 뿐만 아니라, 이동 평균 변환을 지원하므로 시퀀스 내에서 발생하는 잡음의 영향도 최소화 할 수 있다.

특히, 본 연구에서는 이동 평균 및 정규화 변환된 두 시퀀스간의 타임 워핑 거리를 계산할 때, 기본 거리 함수로서 맨해튼 거리, 유클리드 거리, 대응되는 각 쌍의 거리 중 최대 거리 등 세 가지 형태의 L_p 를 지원한다. 이 세 가지 거리 함수는 다음과 같은 특성을 갖는다[1, 25, 26].

- 맨해튼 거리 L_1 : L_1 은 측정 오차가 additive I.I.D.(independent, identically distributed) Gaussian인 경우에 최적이라 알려져 있다. 이러한 특성으로 인하여 L_1 은 급격한 잡음에 큰 영향을 받지 않는다.
- 유클리드 거리 L_2 : L_2 는 측정 오차가 additive I.I.D.(independent, identically distributed) Laplacian인 경우에 최적이라 알려져 있다. L_2 는 현재 시계열 시퀀스 검색에 가장 널리 사용되고 있다.
- 대응되는 각 쌍의 거리 중 최대 거리 L_∞ : L_∞ 는 비교하고자 하는 시퀀스의 길이에 관계없이 일정한 크기의 유사 허용치를 사용할 수 있다는 것이 큰 장점이다. 질의 시퀀스와 전체적인 경향이 매우 유사한 시퀀스가 큰 잡음을 갖는 경우, L_∞ 는 이를 최종 질의 결과에서 제외시킬 수 있다. 그러나 제안하는 유사 모델은 L_∞ 를 이용하여 유사 정도를 계산하기 전에 미리 이동 평균 변환을 수행하므로 잡음으로 인한 이러한 문제가 완화된다.

각각의 L_p 거리 함수는 검색해 내는 대상이 조금씩 다르므로 응용 분야에 따라 선호하는 대상이 다르다. 또한, 동일한 응용인 경우에도 사용자의 성향에 따라 선호하는 검색 결과가 다를 수 있다[26]. 제 6장에서는 L_p 거리 함수에 따르는 결과 시퀀스들의 변화에 관한 실험 결과를 제시한다. 본 연구에서는 이러한 응용 분야 혹은 사용자의 선호도를 반영하기 위하여 위의 세 가지 L_p 거리 함수를 모두 지원하는 유사 모델을 제안한다. 즉, 위의 세 가지 거리 함수 중 선호하는 L_p 를 사용자가 질의 작성시 직접 선택하도록 하는 것이다. 이 결과, 사용자는 자신이 실제로 원하는 검색 결과를 구할 수 있다.

이와 같은 유사 모델을 기반으로 본 논문에서는 시계열 데이터베이스를 위한 모양 기반 유사 검색 문제를 다음과 같이 정의한다. 질의 시퀀스 Q, 유사 허용치 ϵ , 이동 평균 계수 k, L_p 거리 함수의 p가 주어지면, $D(X, Q) (= D_{tw}(\text{Norm}(\text{MV}_k(X)), \text{Norm}(\text{MV}_k(Q))))$ 의 값이 ϵ 이하인 데이터베이스 내의 서브시퀀스 X를 찾고, X를 포함하는 데이터 시퀀스 S와 S내에서의 서브시퀀스 X의 시작 위치를 반환한다.

4. 인덱싱 방안

본 장에서는 제 3장에서 제안한 유사 모델을 기반으로 하는 모양 기반 검색을 효과적으로 처리하기 위한 인덱싱 방안에 관하여 논의한다. 제 4.1절에서는 기본 인덱스 구조로서 사용하는 접미어 트리(suffix tree)에 대하여 설명하고, 제 4.2절에서는 인덱스의 크기를 줄이는 전략 및 구체적인 서브시퀀스 트리 구성 절차를 기술한다.

4.1 접미어 트리

접미어 트리는 다수의 시퀀스들을 인덱싱하기 위하여 사용되며, 주어진 질의 시퀀스와 정확하게 일치되는 서브시퀀스의 위치를 신속하게 찾도록 하는데 유용하다[27]. 접미어 트리는 인덱스의 구성시 어느 특정 거리 함수의 사용을 미리 가정하지 않으므로, 질의 처리 과정에서 사용자가 요구하는 임의의 거리 함수를 지원할 수 있다.

한 시퀀스에 속하는 각 접미어는 접미어 트리 내의 리프 노드(leaf node)와 대응된다. 예를 들어, 시퀀스 $S_i = (s[0], s[1], \dots, s[\text{Len}(S_i) - 1])$ 의 한 접미어 $(s[j], s[j+1], \dots, s[\text{Len}(S_i) - 1])$ 는 라벨 (S_i, j) 를 갖는 리프 노드로서 표현된다. 접미어 트리의 각 에지(edge)들도 라벨을 가지며, 트리의 루트로부터 라벨 (S_i, j) 를 가지는 리프 노드까지 경로상의 에지 라벨들을 모두 접합(concatenation)하면 접미어 $(s[j], s[j+1], \dots, s[\text{Len}(S_i) - 1])$ 가 된다. 루트로부터 한 내부 노드(internal node)까지 경로상의 에지 라벨들을 모두 접합한 결과는 그 내부 노드 아래에 존재하는 모든 리프 노드와 대응되는 접미어들의 최장 공통 접두어(longest common prefix)가 된다.

접미어 트리는 저장하는 접미어들이 많은 공통 접두어 서브시퀀스를 가질 때, 좋은 압축 효과를 갖는다. 그러나 시퀀스의 요소 값은 실수의 타입을 가지므로 접미어들이 공통의 접두어 서브시퀀스를 가질 가능성은 매우 낮다. 참고문헌[14]에서는 이러한 문제를 해결하기 위하여 도메인 분류(categorization)를 사용하는 방법을 제안한 바 있다. 도메인 분류는 요소 값을 심볼로 변환하기 위하여 요소 값의 도메인을 여러 범위로 분할하는 작업이다. 서로 다른 요소 값이라도 같은 범위에 속하게 되면, 동일한 심볼로 변환된다. 이러한 심볼로 변환된 시퀀스들을 대상으로 접미어 트리를 구성하는 경우, 접미어들이 공통의 접두어 서브시퀀스를 가질 가능성이 상대적으로 높아진다.

접미어 트리 내에 저장된 시퀀스 S의 한 접미어 S_a 와 S_a 의 임의의 접두어 서브시퀀스 S_b 를 고려해 보자. S_b 는 S_a 의 접두어이므로 S_b 의 모든 요소 값들은 S_a 내에 포함된다. 따라서 S_a 만을 참조해서도 질의 시퀀스 Q와 타임 워킹 거리가 유사 허용치 이하인 서브시퀀스 S_b 를 검색할 수 있다. 그러므로 유사 서브시퀀스 검색에서 변환을 지원하지 않는 경우에는 모든 시퀀스들의 가능한 접미어들만을 접미어 트리 내에 저장함으로써 임의의 유사한 서브시퀀스를 검색할 수 있다.

본 논문에서 채택하는 유사 모델에서는 정규화 변환을 지원한다. 정의 1에 나타난 바와 같이, 한 서브시퀀스 S를 정규화 변환하기 위해서는 $\text{Max}(S)$ 와 $\text{Min}(S)$ 를 사용한다. 위의 예에서 사용된 S_a 와 S_b 를 다시 고려하면, $\text{Max}(S_a)$ 와 $\text{Max}(S_b)$, 그리고 $\text{Min}(S_a)$ 와 $\text{Min}(S_b)$ 는 일반적으로 서로 다른 값을 갖는다. 이 결과, $\text{Norm}(S_b)$ 는 $\text{Norm}(S_a)$ 의 접두어가 아니므로 $\text{Norm}(S_a)$ 만을 참조하여 질의 시퀀스 Q와의 타임 워킹 거리가 유사 허용치 이하인 $\text{Norm}(S_b)$ 를 찾을 수 없다. 따라서 정규화 변환을 지원하는 경우에는 접미어 뿐만 아니라 발생 가능한 모든 서브시퀀스들을 개별적으로 인덱스 내에 저장해야 한다. 접미어 트리와 구조적으로 동일하며, 각 시퀀스 내의 모든 가능한 서브시퀀스들을 포함하는 인덱스를 서브시퀀스 트리(subsequence tree : ST)[28]라 정의한다.

4.2 압축 서브시퀀스 트리 구성

길이가 n인 시퀀스에 대하여 접미어 트리는 n개의 접미어들만을 인덱싱의 대상으로 하는 반면, 서브시퀀스 트리는 $n * (n + 1) / 2$ 개의 서브시퀀스들을 인덱싱의 대상으로 한다. 그러나 서브시퀀스 트리에서 같은 시퀀스에 속하는 서브시퀀스들은 많은 공통되는 접두어를 갖는다는 특성으로 인하여 서브시퀀스 트리는 접미어 트리에 비하여 크기가 지나치게 커지는 현상은 발생하지 않는다[28].

저장될 서브시퀀스 수를 줄일 수 있다면 서브시퀀스 트리의 크기를 줄이는 효과를 얻을 수 있으며, 이 결과 질의

처리 시간을 줄일 수 있다. 본 논문에서는 질의 처리 과정에서 필요로 하는 모든 정보를 손실 없이 유지하며, 서브시퀀스 트리에 저장하는 서브시퀀스의 수를 줄일 수 있는 다음과 같은 인덱스의 압축 표현 방식을 제안한다.

임의의 서브시퀀스 S_a 와 S_a 의 접두어 서브시퀀스 S_b 를 고려해 보자. $\text{Max}(S_a) = \text{Max}(S_b)$ 이며 $\text{Min}(S_a) = \text{Min}(S_b)$ 인 경우에는 $\text{Norm}(S_b)$ 는 $\text{Norm}(S_a)$ 의 접두어 서브시퀀스가 된다. 따라서 이 경우에는 $\text{Norm}(S_a)$ 만을 참조하여 질의 시퀀스 Q와의 타임 워킹 거리가 유사 허용치 이하인 $\text{Norm}(S_b)$ 를 찾을 수 있다. 즉 서브시퀀스 트리 구성시, 서브시퀀스 $\text{Norm}(S_a)$ 가 트리 내에 저장되어 있다면, $\text{Norm}(S_b)$ 를 트리에 저장할 필요가 없는 것이다. 정규화 변환된 시퀀스 S의 i번째 요소에서 j번째까지의 요소를 포함하는 서브시퀀스를 $S[i : j]$ 로 표현하면, 서브시퀀스 트리 구성시 $S[i : j]$ 의 삽입 여부를 결정하는 과정은 다음과 같다.

- ① insert $S[i : j]$ if $j = \text{Len}(S) - 1$
- ② insert $S[i : j]$ if $\max(S[i : j]) \neq \max(S[i : j+1])$ or $\min(S[i : j]) \neq \min(S[i : j+1])$

이렇게 함으로써 서브시퀀스 트리내에 저장해야 하는 서브시퀀스들의 수를 줄일 수 있으며, 이 결과 질의 처리 시간의 감소를 기대할 수 있다. 본 논문에서는 위 과정에 의하여 삽입된 서브시퀀스를 저장 서브시퀀스(stored subsequence)라 정의하고, 저장 서브시퀀스만을 포함하는 인덱스를 압축 서브시퀀스 트리(compact subsequence tree : CST)라 정의한다.

시계열 데이터베이스를 대상으로 압축 서브시퀀스 트리 CST를 구성하는 절차는 다음과 같은 다섯 단계로 이루어진다.

단계 1 : 이동 평균 변환

시계열 데이터베이스 내의 각 시퀀스 S에 대하여 k-이동 평균 변환을 수행한다. 여기서, 이동 평균 계수 k 값은 해당 응용에 적합한 것을 선정한다. S를 k-이동 평균한 시퀀스를 $MV_k(S)$ 라 하자.

단계 2 : 저장 서브시퀀스 추출

각 $MV_k(S)$ 의 모든 가능한 서브시퀀스로부터 저장 서브시퀀스 X를 선별한다. 이때, 추출될 서브시퀀스의 최소 길이 L을 지정할 수 있다. 즉, 너무 짧은 X가 질의 결과로서 큰 의미가 없는 경우에는 L 이상의 길이를 갖는 서브시퀀스만을 추출하도록 한다. 여기서, L 값은 해당 응용에 적합한 것을 선정한다.

단계 3 : 정규화 변환

각 저장 서브시퀀스 X에 대하여 고유의 $\text{Min}(X)$ 와 $\text{Max}(X)$ 값을 이용함으로써 정규화 변환을 수행한다. X를 정규화 변환한 것을 $\text{Norm}(X)$ 라 하자.

단계 4 : 도메인 분류를 이용한 심볼 변환

각 $\text{Norm}(X)$ 에 대하여 도메인 분류를 이용함으로써 심볼 변환을 수행한다. 즉, $\text{Norm}(X)$ 의 각 요소 값이 도메인 분류에 의하여 정의된 범위들 중 어느 곳에 속하는가 하는 것을 파악한 후, 이 범위와 대응되는 심볼로 변환시킨다. 여기서, 도메인 분류에 의하여 정의된 범위들의 수 C 는 해당 응용에 적합한 것을 선정한다.

단계 5 : 트리 구성

심볼로 변환된 저장 서브시퀀스들을 대상으로 CST를 구성한다. 심볼로 변환된 각 서브시퀀스는 루트 노드로부터 해당 리프 노드까지의 경로 상의 에지 라벨을 모두 접합시킨 것에 대응한다. 심볼로 변환된 각 서브시퀀스를 식별하기 위하여 (SID, i, j) 의 정보가 해당 리프 노드에 저장된다. 이때 SID 는 심볼화된 서브시퀀스가 추출된 원래의 시퀀스 식별자를 나타내며, i 는 시작점 오프셋(offset)을, j 는 종점 오프셋을 나타낸다. 대용량의 데이터베이스 환경에서 CST는 디스크에 존재하게 되므로, 효과적인 구성을 위하여 디스크 기반 접미어 트리 구성 알고리즘[14]을 활용한다.

5. 질의 처리 방안

본 장에서는 제 4장에서 제안한 인덱싱 전략을 기반으로 하는 모양 기반 유사 서브시퀀스 검색을 효과적으로 처리하는 알고리즘을 제시하고, 그 검색 성능을 분석한다.

5.1 질의 처리 알고리즘

질의에서는 다음의 두 가지 방식에 의하여 이동 평균된 시퀀스를 입력받는 것을 전제로 한다. 첫 번째는 사용자가 질의 패턴을 직접 작성하여 질의 시퀀스로 사용하는 방식이다. 이 경우, 사용자가 원하는 시퀀스의 모양을 직접 그리는 상황이므로, 실질적으로 잡음이 발생하지 않는다. 두 번째는 사용자가 데이터베이스 내에서 선택한 이동 평균된 시퀀스로부터 일부의 서브시퀀스를 추출하여 질의 시퀀스로 사용하는 방식이다. 두 가지 경우에서 모두 실질적으로는 이미 이동 평균된(잡음이 존재하지 않는) 질의 시퀀스를 대상으로 하므로, 이를 다시 정규화 변환하여 아래의 질의 처리 알고리즘을 적용한다.

(그림 1)은 압축 서브시퀀스 트리 CST를 이용하여 질의 시퀀스 Q 와 D_{tw} 가 ϵ 이내인 유사한 서브시퀀스들을 데이터베이스로부터 검색하는 알고리즘 Search-CST를 나타낸 것이다. 본 알고리즘에서는 질의 처리시 사용되는 L_p 거리 함수를 사용자가 임의로 지정할 수 있도록 허용하고 있으므로, Search-CST에는 L_p 거리 함수가 호출 함수의 인자로 지정되어 있음을 볼 수 있다. 또한 여기서 질의 시퀀스 Q 와 CST내의 모든 시퀀스들은 이미 이동 평균 변환과 정규화 변환을 모두 끝낸 것임에 유의해야 한다.

Algorithm 1 : Search-CST

Input : compressed subsequence tree CST, query sequence Q , tolerance ϵ , base distance metric L_p
Output : set of answers answerSet
1. candidateSet := VisitNode-and-FindAnswers-CST(
 rootNode (CST), Q , ϵ , emptyTable, L_p) ;
2. answerSet := PostProcess (candidateSet) ;
3. return answerSet ;

(그림 1) Search-CST : 압축 서브시퀀스 트리를 이용한 유사검색 알고리즘

제 4.2절에서 설명한 바와 같이 CST는 심볼로 변환된 서브시퀀스들을 대상으로 구성되어 있으므로, 서브시퀀스와 질의 시퀀스 사이의 타임 워핑 거리 D_{tw} 를 직접 계산할 수 없다. 따라서 Search-CST에서는 심볼로 변환된 서브시퀀스와 질의 시퀀스 Q 사이의 타임 워핑 거리 계산을 위하여, D_{tw} 의 하한 함수인 D_{tw-lb} 를 다음과 같이 재귀적으로 정의한다.

정의 6

심볼로 변환된 서브시퀀스 CS 와 질의 시퀀스 Q 사이의 D_{tw} 의 하한 함수인 $D_{tw-lb}(CS, Q)$ 는 다음과 같이 정의된다.

- ① $D_{tw-lb}(), () = 0$
- ② $D_{tw-lb}(CS, ()) = D_{tw-lb}(), Q = \infty$
- ③ $D_{tw-lb}(CS, Q) = ((D_{base-lb}(\text{First}(CS), \text{First}(Q)))^p + (\min(D_{tw-lb}(CS, \text{Rest}(Q)), D_{tw-lb}(\text{Rest}(CS), Q), D_{tw-lb}(\text{Rest}(CS), \text{Rest}(Q))))^p)^{1/p}$
- ④ $D_{base-lb}(A, b) = 0 \quad (\text{if } A.lb \leq b \leq A.ub)$
 $= b - A.ub \quad (\text{if } b > A.ub)$
 $= A.lb - b \quad (\text{if } b < A.lb)$

□

여기서, A 는 $\text{First}(CS)$ 에 해당하는 심볼을 나타내며, b 는 $\text{First}(Q)$ 에 해당하는 실제의 실수 값을 나타낸다. 또한 $A.lb$ 와 $A.ub$ 는 각각 A 가 속한 도메인의 최소 요소 값과 최대 요소 값을 나타낸다.

이와 같이, 제안된 기법에서는 인덱싱의 대상인 서브시퀀스들을 심볼로 변환하지만, 질의 시퀀스의 원래 요소 값을 기반으로 검색을 수행한다. 따라서 참고문헌[19]에서와는 달리 유사한 두 값이 서로 다른 심볼로 변환되는 경우에도 유사한 두 시퀀스가 유사하지 않은 것으로 간주되는 현상은 발생되지 않는다.

Search-CST의 실질적인 탐색 과정은 VisitNode-and-FindAnswers-CST(Line 1)의 수행을 통하여 실행된다. (그림 2)에 보이는 VisitNode-and-FindAnswers-CST는 질의 시퀀스 Q 와 타임 워핑 하한 거리(D_{tw-lb})가 ϵ 보다 작거나 같은 후보 서브시퀀스들을 검색한다. 알고리즘에서는 임의의 노드 N 이 주어지면, 새로운 후보 서브시퀀스 검색을 위하여 그 노드의 모든 자식 노드를 방문하며, 또한 각 자식 노드에서

그 이상의 깊이 탐색이 필요한가의 여부를 판별하게 된다.

예를 들어, 알고리즘이 노드 N 을 방문한 후, 그 자식 노드 CN_i 를 탐색하는 과정은 다음과 같다. 설명을 간략히 하기 위하여 여기에서는 각 예지가 단일 심볼의 예지 라벨정보만을 갖는다고 가정한다.

첫 번째 과정은 질의 시퀀스 Q 와 $label(N, CN_i)$ 을 위한 거리 축적 테이블 CT_i 를 생성하는 단계이다. 거리 축적 테이블 CT_i 에는, 질의 시퀀스 Q 와 $label(N, CN_i)$ 가 각각 X 축과 Y 축에 할당된다. 이때, 노드 N 이 루트 노드라면 테이블은 처음부터 생성되며, 만일 노드 N 이 루트 노드가 아니라면, 루트로부터 노드 N 에 대하여 이전에 생성된 테이블 CT_i 위에 $label(N, CN_i)$ 를 위한 새로운 행(row)이 쌓이게 된다. 여기에서 사용되는 함수 $Addrow$ (Line 3)는 주어진 L_p 거리 함수를 사용하여 D_{tw-lb} 계산에 의한 거리 축적 테이블 생성을 담당한다.

두 번째 과정은 새로운 후보 서브시퀀스 검색을 위하여 거리 축적 테이블에 새로이 추가된 행의 마지막 열(column)의 데이터 값을 조회하는 단계이다(Line 4). 만일 루트로부터 CN_i 노드까지의 경로에서 ϵ 이하의 타임 워핑 하한 거리가 얻어지면(Line 5), CN_i 이하의 리프 노드들로부터 서브시퀀스 식별자를 추출하여, 후보 서브시퀀스 집합에 추가한다(Line 6).

여기에서 주의할 점은 압축 서브시퀀스 트리에는 실제로 저장되지 않는 서브시퀀스가 존재한다는 점이다. 이 저장되지 않은 서브시퀀스는 트리의 루트 노드로부터 임의의 내부 노드 사이의 경로에 임베디드(embedded) 되어 있는 것으로 간주되며, 따라서 리프 노드가 아닌 임의의 내부 노드까지의 경로로부터 후보 서브시퀀스를 검색할 수 있다.

예를 들어 루트 노드에서 CN_i 노드까지의 경로에서 질의 시퀀스 Q 와의 타임 워핑 하한 거리(D_{tw-lb})가 ϵ 보다 크지 않은 경우, 이 경로에 임베디드 되어 있는 후보 서브시퀀스를 검색하는 과정은 다음과 같다. 루트 노드에서 CN_i 까지의 경로가 표현하는 모든 서브시퀀스에 관한 정보는 노드 CN_i 이하에 연결된 리프 노드들로부터 얻는다. LN 을 이 리프 노드 중 하나라고 가정할 때, LN 이 (S_k, j, j') 의 서브시퀀스 식별 정보를 갖고 있으며, 루트 노드에서 노드 CN_i 까지의 경로 길이가 l 이라면, 이 경로에 의하여 인식되는 서브시퀀스 중 하나로서 $(S_k, j, j'+l-1)$ 를 들 수 있다.

세 번째 과정은 현재 방문중인 노드에 대하여 이 이상 하향 탐색(자식 노드 탐색)을 수행할 필요가 있는지의 여부를 판단하는 단계이다. 만일 거리 축적 테이블에 새로이 추가된 행의 데이터 값 중, 적어도 하나의 열이 ϵ 보다 크지 않은 데이터 값을 갖는다면(Line 8), 새로운 후보 시퀀스 검색을 위하여 해당 노드로부터 자식 노드들에 대한 하향 탐색을 진행한다. 그러나 만일 그렇지 않다면, 해당 노드 N 의 다음 자식 노드에 대한 트리 탐색을 진행한다. 트리의 하향 탐색이 필요하다고 판정된 경우에는(Line 8), $VisitNo-$

de -and- $FindAnswers$ - CST 의 알고리즘이 재귀적으로 호출되는 형태를 갖는다(Line 9).

이와 같이 $VisitNode$ -and- $FindAnswers$ - CST 에 의하여 반환된 모든 결과는 하한 함수인 D_{tw-lb} 를 이용하여 얻어진 것이므로, 실제의 타임 워핑 거리가 ϵ 보다 큰 서브시퀀스가 결과에 포함될 수 있다. 이와 같은 서브시퀀스들을 착오 채택(false alarms)[13]이라 부른다.

따라서 $Search$ - CST 는 착오 채택된 서브시퀀스를 제거하기 위하여 후처리 과정(post processing)을 필요로 한다. 즉, 함수 $Postprocess$ 가 호출되어(Line 2), $candidateSet$ 에 저장된 모든 후보 서브시퀀스의 실제 서브시퀀스를 액세스하여, 이동 평균 변환과 정규화 변환을 거친 후, 타임 워핑 거리 D_{tw} 에 의하여 질의 시퀀스 Q 와의 거리를 계산한다. 그 결과 질의 시퀀스 Q 와의 실제 타임 워핑 거리가 ϵ 보다 크지 않은 서브시퀀스만이 최종 결과로 사용자에게 반환된다(Line 3).

Algorithm 2 : VisitNode-and-FindAnswers-CST

```

Input : node  $N$ , query sequence  $Q$ , tolerance  $\epsilon$ , cumulative distance
       table  $T$ , base distance function  $L_p$ 
Output : set of candidate answers candidateSet
1. candidateSet := {};
2. For (each child node  $CN_i$  of the node  $N$ ) do
3.    $CT_i$  := Addrow ( $T$ ,  $Q$ , label ( $N$ ,  $CN_i$ ),  $D_{tw-lb}$ ,  $L_p$ );
4.   Let dist be the last column value of the new row;
5.   if (dist ≤  $\epsilon$ ) then
6.     candidateSet := candidateSet ∪ {GetID (GetLeafNodes( $CN_i$ ))};
7.   Let mDist be the minimum column value of the new row;
8.   if (mDist ≤  $\epsilon$ ) then
9.     candidateSet := candidateSet ∪
       VisitNode-and-FindAnswers-CST( $CN_i$ ,  $Q$ ,  $\epsilon$ ,  $CT_i$ ,  $L_p$ );
10. return candidateSet;

```

(그림 2) VisitNode-and-FindAnswers-CST : 압축 서브시퀀스 트리 검색 알고리즘

5.2 알고리즘 분석

$Search$ - CST 의 이론적 계산량에 의한 질의 처리 비용을 순차검색 방식 Seq-Scan과 압축되지 않은 서브시퀀스 트리를 이용한 검색 방식 Search-ST와 비교한다.

우선, 순차 검색 방식 Seq-Scan에 의한 질의 처리 과정을 간략히 설명하면 다음과 같다. Seq-Scan에서는 데이터 베이스 내의 각 시퀀스 S 를 읽어들이며, S 에 대한 k -이동 평균 변환을 하여 $MV_k(S)$ 를 얻은 후, $MV_k(S)$ 로부터 모든 가능한 서브시퀀스 X 를 추출하여 이 들 각각의 X 에 대하여 정규화 변환을 수행한다. 다음, 주어진 질의 시퀀스 Q 와 각각의 X 를 대상으로 거리 축적 테이블을 생성하며, 이 때 거리축적 테이블의 생성 비용은 $O(|Q||X|)$ 이다. M 개의 k -이동 평균 변환된 시퀀스를 대상으로 하는 경우, M 개 시퀀스의 평균 길이를 L 이라 가정하면, 여기에서 $ML(L+1)/2$ 개의 서브시퀀스가 생성되고, 각 서브시퀀스의 평균 길이는 $(L+2)/3$ 이다. 따라서 순차 검색에 의한 질의 처리 비용은 $O(ML^3|Q|)$ 로 얻어진다.

Search-ST에서는 다음의 두 가지 이유에 의하여 순차 검색에 비하여 질의 처리 비용이 크게 감소한다. Search-ST에서는 공통 접두어를 갖는 다수의 서브시퀀스들이 거리 축적 테이블을 공유함으로써 얻어지는 효과를 얻을 수 있으며, 또한 트리 탐색 중의 가지치기(branch-pruning) 효과를 동시에 얻을 수 있다. Search-ST의 질의 처리 비용은 $O(ML^3|Q|/R_dR_p + nL|Q|)$ 이다. 여기에서 좌측 수식은 트리 탐색을 위한 비용이며, 우측 수식은 후처리를 위한 비용을 나타낸 것으로, $R_d(\geq 1)$ 는 거리 축적 테이블의 공유에서 얻어지는 감소 계수(reduction factor)를 나타내며, $R_p(\geq 1)$ 는 가지치기 현상에서 얻어지는 감소 계수를 나타낸다. 또한 n 은 후처리를 필요로 하는 서브시퀀스 개수를 나타낸다.

압축 서브시퀀스 트리를 이용한 유사 검색 알고리즘 Search-CST의 질의 처리 비용은 앞에서 보인 서브시퀀스 트리를 이용한 유사 검색 알고리즘 Search-ST의 질의 처리 비용으로부터 쉽게 유도될 수 있다. 압축 서브시퀀스 트리는 서브시퀀스 트리보다 그 크기가 작으므로 압축 서브시퀀스 트리의 탐색시간은 전체적으로 감소된다. 그러나 서브시퀀스 트리의 경우, 공통 접두어를 갖는 서브시퀀스를 다수 포함하므로 압축 서브시퀀스 트리의 경우와 비교할 때 비교적 높은 감소 계수(reduction factor) R_d 를 갖는다. 압축 서브시퀀스 트리의 트리 압축율을 $p(0 < p \leq 1)$ 로 표현하면, p 는 다음 수식, $p = (\text{the number of stored subsequences} / \text{the number of total subsequences})$ 로 정의된다. 따라서 이를 이용한 Search-CST의 질의 처리 비용은 $O(pML^3|Q|/R_d'R_p + nL|Q|)$ 으로 얻어진다.

6 성능 평가

본 장에서는 실험에 의한 성능 분석을 통하여 제안하는 기법의 우수성을 규명한다. 제 6.1절에서는 실험 환경을 설명하고, 제 6.2절에서는 실험 결과를 분석한다.

6.1. 실험 환경

본 연구에서는 합성 데이터 Syn_Data와 실제 데이터 S&P_Data를 이용한 실험을 통하여 성능을 분석한다. Syn_Data 내의 각 시퀀스 $S = \langle s_1, s_2, \dots, s_n \rangle$ 는 다음과 같은 랜덤 워크(random walk) 형태를 가진다.

$$s_i = s_{i-1} + z_i$$

여기서 z_i 는 구간 $[-0.1, 0.1]$ 사이에서 균일한 분포를 취하는 랜덤 변수이며, 시퀀스의 처음 요소 값 s_1 은 구간 $[1, 10]$ 사이의 임의의 값을 취한다. 실제 데이터 S&P_Data는 미국의 S&P 500 주식 데이터로서, 각 주식의 매일 종가를 기록한 시퀀스들의 집합이다.

성능 평가는 다음 세 가지 서로 다른 기법을 대상으로

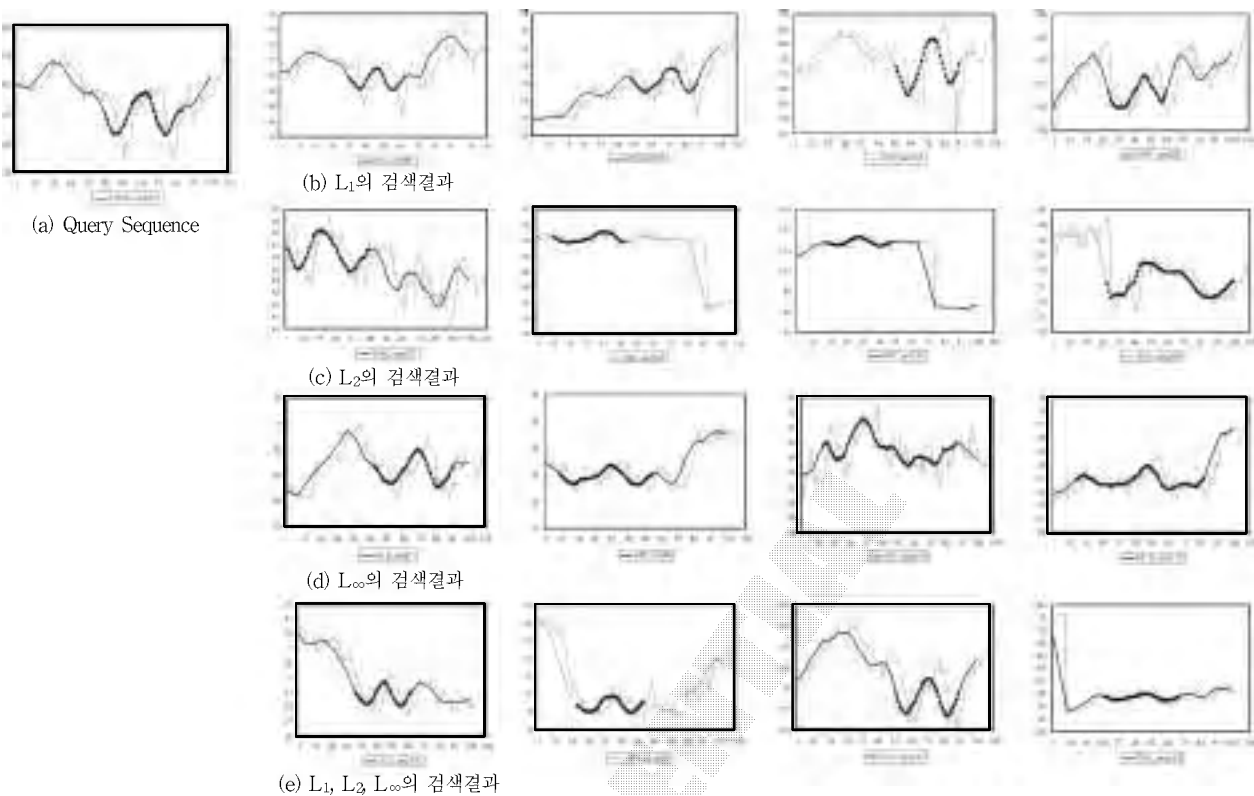
한다. Search-CST는 본 논문에서 제안한 압축 서브시퀀스 트리를 이용한 검색 방식이다. Search-CST는 동일한 서브시퀀스 트리를 이용하여 임의의 $L_p(p = 1, 2, \infty)$ 기반의 서브시퀀스 검색을 수행할 수 있으며, 이들을 각각 Search-CST- L_1 , Search-CST- L_2 , Search-CST- L_∞ 로 나타낸다. 성능 비교를 위한 기존의 기법으로는 순차 검색 방식 Seq-Scan과 압축되지 않은 서브시퀀스 트리를 이용한 검색 방식 Search-ST를 사용한다. Seq-Scan의 각 L_p -기반 검색을 Seq-Scan- L_1 , Seq-Scan- L_2 , Seq-Scan- L_∞ 로, Search-ST의 각 L_p -기반 검색을 Search-ST- L_1 , Search-ST- L_2 , Search-ST- L_∞ 로 나타낸다. 질의 시퀀스로는 제 5장에서 언급한 바와 같이, 이동 평균 변환을 취한 데이터 시퀀스로부터 임의로 추출한 서브시퀀스를 사용한다. 성능 평가 지수로서 UNIX의 time 명령어를 이용한 평균 질의 검색 시간을 측정한다. 실험을 위한 하드웨어 플랫폼으로는 Solaris 7을 운영체제로 사용하고 512MB의 주기억장치를 갖는 Sun Ultra-Sparc-10 워크스테이션을 사용한다.

6.2 실험 결과 및 분석

먼저, 실험 1에서는 본 논문에서 제안한 유사 모델에 의한 유사 서브시퀀스 검색 성능을 측정한다. 실험 데이터로는 10-이동 평균 변환된 S&P_Data를 이용하며, 평균 길이 100의 1000개의 시퀀스를 대상으로 한다. 질의 시퀀스로는 주식 데이터 분석에서 자주 사용되는 이중 바닥(double bottoms) 패턴의 서브시퀀스를 이용한다. 유사 허용치 ϵ 는 각 L_p -기반 검색에 대하여 평균 100개의 서로 다른 최종 결과를 얻기 위한 값부터 200개, 300개, 1000개의 최종 결과를 얻기 위한 값을 사용한다. 단, 본 실험에서는 검색된 결과 시퀀스의 상호 분석을 위하여 추출된 서브시퀀스 중 시점 오프셋 값과 종점 오프셋 값이 5이하의 오차만을 가질 경우, 동일 서브시퀀스로 간주한다.

(그림 3)은 본 실험에 의한 유사 서브시퀀스 검색 예의 일부를 보인 것이다. 각 그림 상의 가는 실선은 실제 데이터 시퀀스를 나타내며, 실선은 데이터 시퀀스를 이동 평균 변환한 시퀀스를 나타낸다. (그림 3)(a)의 이동 평균선 상에 굵은 선으로 표시된 서브시퀀스는 실험에 사용된 질의 시퀀스를 나타내며, (그림 3)(b), (그림 3)(c), (그림 3)(d)의 이동 평균선 상에 굵은 선으로 표시된 서브시퀀스는 거리 함수로서 각각 L_1 , L_2 , L_∞ 를 이용한 경우의 검색 예를 보인다.

이들 검색된 결과로부터 이중 바닥을 갖는 서브시퀀스들이 실제 요소 값에 관계없이 효과적으로 검색되었으며, 또한 지정된 거리 함수에 따라 거리 함수의 특징이 반영된 서브시퀀스들이 적절히 검색되었음을 볼 수 있다. 유사 서브시퀀스 검색을 위한 적절한 거리 함수의 선택은 응용 분야에 의존적인 문제로서, 응용 분야에 따라 사용자에게 의하여 직접 지정되는 것이 바람직하다.



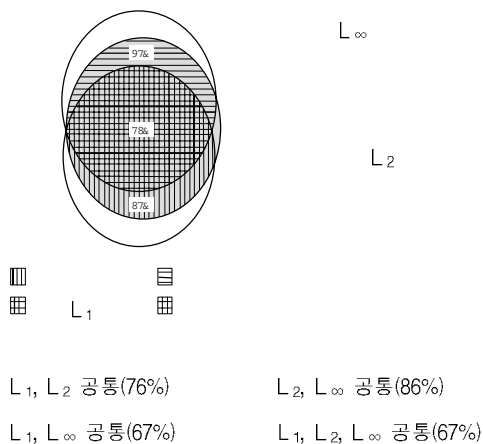
(그림 3) 유사 서브시퀀스의 검색 예

그러나 본 논문에서 제안한 유사 모델에서는 기본적으로 이동 평균 변화와 타임 위핑 변화를 지원하므로 주어진 유사 허용치(최종 결과 개수)에 대하여 각 L_p 거리 함수를 사용한 유사 검색 결과에는 많은 부분 공통된 서브시퀀스가 포함됨을 볼 수 있다. 예를 들어 (그림 3)의 (e)는 거리 함수로 L_1, L_2, L_∞ 를 각각 사용하여 검색된 결과 중에서 공통으로 검색된 서브시퀀스의 예를 나타낸 것이다. S&P_Data에 대한 공통 서브시퀀스 검색 정도를 비교하는 실험을 수행하여 그 결과를 (그림 4)에 보인다. (그림 4)의 벤 다이어그램은 유사

허용치 ϵ 로서 서로 다른 평균 100개의 최종 결과를 얻기 위한 값을 사용한 경우, 각 L_p -기반 유사 검색 결과의 중복 정도를 나타낸다. 최종 평균 검색 결과 개수를 200, 300, 1000으로 증가시킨 경우에도 이와 유사한 실험 결과를 얻는다.

실험 2에서는 본 논문에서 제안한 Search-CST의 성능을 Seq-Scan, Search-ST의 성능과 비교한다. <표 2>에 S&P_Data 데이터의 크기 변화에 따른 Search-CST와 Search-ST의 인덱스 크기 변화를 보인다. 실험 데이터로는 10-이동 평균 변환된 평균 길이 100의 시퀀스를 200개에서 1000개까지 사용한다. 또한 각 인덱스 구성에는 도메인 분류 개수로 60을 사용한다. 인덱스 구성시 도메인 분류 개수가 증가하면 인덱스 크기는 증가하지만, 평균 질의 처리 시간은 감소한다. 그러나 그 값이 어느 정도의 한계 값을 넘으면, 인덱스 크기와 질의 처리 시간의 변화율이 매우 적어진다. 이 한계 값을 최적의 도메인 분류 값으로 사용한다. 실험에 의하여 최적의 도메인 분류수를 측정하였으며, 이후의 실험에서 이 값을 사용한다.

실험 결과에 의하면 시퀀스의 개수 증가에 무관하게 압축 서브시퀀스 트리는 서브시퀀스 트리에 비하여 내부 노드수, 에지수, 리프 노드수가 약 50% 정도 감소하는 효과를 얻는다. 단, 압축 서브시퀀스 트리의 에지 노드에는 서브시퀀스 트리에 비하여 더 긴 에지 라벨 정보가 저장되므로 전체 인덱스 감소 효과는 약 36% 정도를 보인다.

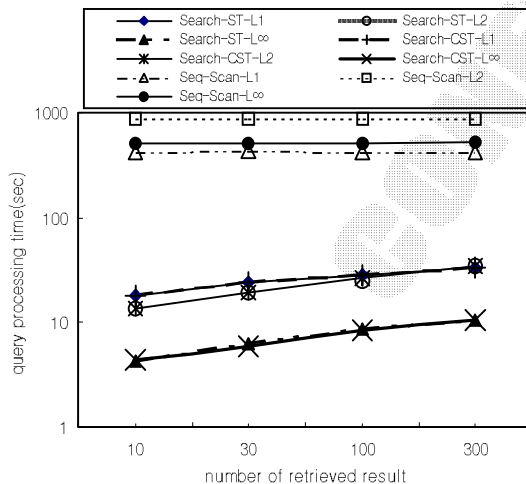


(그림 4) L_p -기반 유사 검색 결과의 중복 정도

<표 2> Search-CST와 Search-ST의 인덱스 크기 비교

시퀀스 개 수	알고리즘	인덱스의 내부노드 개수	인덱스의 에지 개수	인덱스의 리프노드 개수	인덱스의 크기 (KByte)
200	Search-ST	997,806	997,805	819,000	44,960
	Search-CST	537,176	537,175	350,300	28,483
400	Search-ST	1,992,098	1,992,097	1,638,000	90,534
	Search-CST	1,098,474	1,098,473	723,721	58,373
600	Search-ST	3,000,515	3,000,514	2,457,000	136,101
	Search-CST	1,649,482	1,649,481	1,081,029	87,433
800	Search-ST	3,998,892	3,998,891	3,276,000	181,170
	Search-CST	2,191,398	2,191,397	1,435,043	115,903
1000	Search-ST	5,006,675	5,006,674	4,095,000	227,021
	Search-CST	2,727,899	2,727,898	1,782,489	144,326

다음 (그림 5)은 유사 허용치 ϵ 를 기준으로 각 기법들의 성능을 비교한 실험 결과를 나타낸다. 실험 데이터는 평균 길이 100의 S&P_Data를 이용하였으며, 개수는 200개의 것을 이용하고, 질의 시퀀스의 평균 길이는 20으로 하였다. 또한, 질의 시퀀스 평균 길이의 절반 이하인 서브시퀀스들은 인덱싱의 대상에서 제외하였다. ϵ 는 평균 10개의 최종 결과를 얻기 위한 값부터 30개, 100개, 300개의 최종 결과를 얻기 위한 값을 사용하였다.



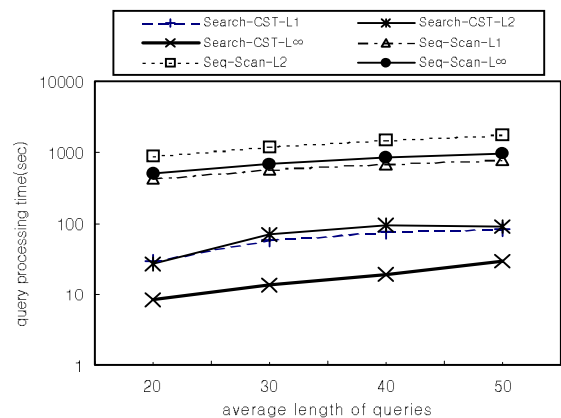
(그림 5) 유사 허용치 변화에 따른 질의 처리 시간의 비교

실험 결과에 의하면, Search-CST의 평균 질의 처리 시간에 관한 성능은 거리 함수 선택에 상관없이 Search-ST와 거의 같은 것으로 나타났다. 그러나 Search-CST는 Seq-Scan과 비교하여 매우 좋은 성능을 가지는 것으로 나타났다. 거리 함수로 L_{∞} 를 사용하는 경우, Search-CST- L_{∞} 는 Seq-Scan- L_{∞} 와 비교하여 유사 허용치의 변화에 따라 약 50배에서 117배까지 높은 성능을 보인다. 또한, 거리 함수로 L_2 를 사용하는 경우, Search-CST- L_2 는 Seq-Scan- L_2 와 비교

하여 약 26배에서 66배까지의 성능 향상을 보이고, 거리 함수로 L_1 를 사용하는 경우, Search-CST- L_1 는 Seq-Scan- L_1 과 비교하여 약 13배에서 23배의 성능 향상을 보인다. 거리 함수로 L_1 과 L_2 를 사용하는 경우, L_{∞} 를 거리 함수로 사용하는 경우에 비하여 검색시간이 다소 길어지는 이유는 동일 개수의 유사 서브시퀀스 검색을 위하여 L_1 과 L_2 의 경우, ϵ 의 값이 L_{∞} 의 경우에 비하여 커지기 때문이다.

실험 2의 결과에 의하여 인덱스 크기와 질의 처리 시간을 고려할 때 Search-CST는 Search-ST에 비하여 좋은 성능을 가짐을 알 수 있다. 따라서 이후의 실험에서는 성능 비교를 위하여 Search-CST와 Seq-Scan만을 비교 대상으로 한다.

실험 3에서는 질의 시퀀스의 평균 길이의 변화에 따른 평균 질의 처리 시간을 비교한다. 실험 데이터는 평균 길이 100, 개수 200의 S&P_Data를 이용하였으며, ϵ 는 질의 시퀀스별로 평균 100개의 최종 결과를 얻기 위한 값을 사용하였다. (그림 6)에 평균 질의 시퀀스 길이를 20부터 30, 40, 50으로 변화시키는 경우에 대한 Search-CST와 Seq-Scan의 성능 비교 결과를 보인다. 비교 결과로부터 Search-CST의 평균 질의 처리 시간에 관한 성능은 질의 시퀀스의 평균 길이가 변하여도 Seq-Scan와 비교하여 매우 좋은 성능을 가지는 것을 알 수 있다. 또한 각 거리 함수별 성능 향상 정도는 (그림 5)의 실험 결과와 거의 동일하다.

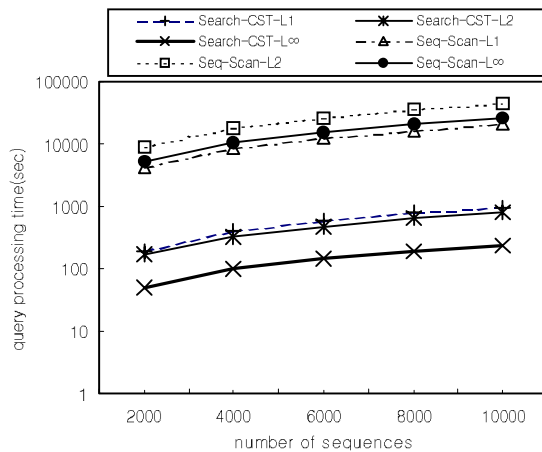


(그림 6) 질의 평균길이 변화에 따른 질의 처리 시간의 비교

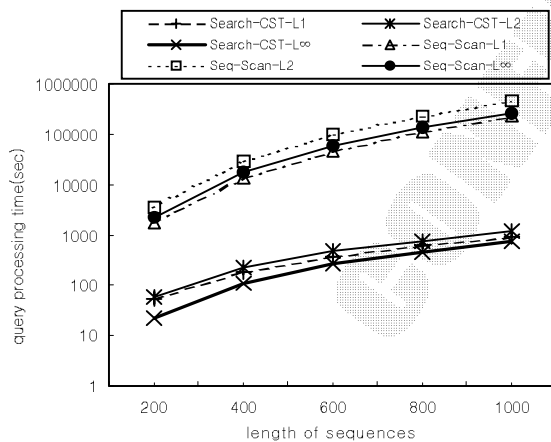
실험 4에서는 다양한 시퀀스의 수 및 시퀀스 길이를 갖는 대규모 데이터를 대상으로 하여 Search-CST의 성능을 Seq-Scan의 성능과 비교한다. 실험 데이터로는 Syn_Data를 이용하며, 평균 질의 시퀀스 길이를 20으로 한다.

(그림 7)는 시퀀스 개수의 변화에 따른 평균 질의 처리 시간의 비교 결과를 나타낸다. 시퀀스의 길이는 100으로 고정시키고, 시퀀스 개수를 2000에서 10000까지 증가시킨다. 각 거리 함수 별 ϵ 의 값은 시퀀스 개수 2000의 데이터에서 평균 약 100개의 최종 결과를 얻기 위한 값을 사용하였다. Search-

CST와 Seq-Scan는 모두 시퀀스 개수가 증가함에 따라 선형적으로 질의 처리 시간이 증가한다. 실험 결과로부터 Search-CST의 성능은 시퀀스 개수의 변화에 무관하게 Seq-Scan과 비교하여 매우 좋은 성능을 가지는 것을 알 수 있다. 또한 각 거리 함수별 성능 향상 정도는 (그림 5)의 실험 결과와 거의 동일하다.



(그림 7) 시퀀스 수 변화에 따른 질의 처리 시간의 비교



(그림 8) 시퀀스 평균 길이 변화에 따른 질의 처리 시간의 비교

(그림 8)은 시퀀스 길이 변화에 따른 평균 질의 처리 시간의 비교 결과를 나타낸다. 시퀀스 개수를 100으로 고정시키고 시퀀스 길이를 200에서 1000까지 증가시킨다. 각 거리 함수 별 ϵ 의 값은 시퀀스 길이 200의 데이터에서 평균 약 100개의 최종 결과를 얻기 위한 값을 사용하였다. 비교 결과로부터 시퀀스 길이가 증가함에 Seq-Scan의 질의 처리 시간은 Search-CST에 비하여 급격히 증가하고 있음을 볼 수 있다. 거리 함수로 L_{∞} 를 사용하는 경우, 시퀀스 길이의 변화에 따라 Search-CST- L_{∞} 는 Seq-Scan- L_{∞} 와 비교하여 약 102배에서 362배까지 높은 성능을 보인다. 또한 거리 함수

로 L_2 를 사용하는 경우, Search-CST- L_2 는 Seq-Scan- L_2 와 비교하여 약 61배에서 390배까지의 성능 향상을 보이고, 거리 함수로 L_1 을 사용하는 경우, Search-CST- L_1 은 Seq-Scan- L_1 과 비교하여 약 31배에서 253배의 성능 향상을 보인다. 또한, 이러한 성능 개선 효과는 시퀀스의 길이가 길어질수록 커지는 양상을 보인다.

7. 결 론

본 논문에서는 시계열 데이터베이스를 위한 모양 기반의 서브시퀀스 검색 문제에 대하여 논의하였다. 모양 기반 검색이란 주어진 질의 시퀀스의 요소 값에 상관없이, 질의 시퀀스와 모양이 유사한 시퀀스 혹은 서브시퀀스를 데이터베이스 내에서 검색하는 연산이다. 본 논문에서는 모양 기반 서브시퀀스 검색을 위한 유연성 있는 새로운 유사 모델을 정의하고, 이를 지원하기 위한 인덱싱 및 질의 처리 방안에 관하여 논의하였다.

제안된 유사 모델에서는 시프팅 변환, 스케일링 변환, 타임 워핑 변환, 이동 평균 변환 등 다양한 형태의 변환들의 조합을 동시에 지원하고, 특히 최종 변환된 두 시퀀스간의 유사 정도를 계산하기 위해 사용되는 거리 함수 L_p 를 사용자가 선택할 수 있도록 허용한다. 따라서 응용 분야 혹은 사용자가 선호하는 다양한 유사 모델을 유연성 있게 지원할 수 있다.

또한 본 논문에서는 효율적인 동시에 착오 기각 없는 모양 기반 검색을 지원하기 위한 압축 서브시퀀스 트리 구조를 제안하고, 이를 기반으로 하는 효율적인 질의 처리 기법을 제시하였다. 압축 서브시퀀스 트리는 서브시퀀스 트리의 저장 공간의 오버헤드를 줄일 수 있는 디스크 기반의 새로운 인덱스 구조이다. 제안된 인덱싱 및 질의 처리 기법의 중요한 특징은 사전에 미리 구성된 단 하나의 인덱스만을 이용하여 L_1 , L_2 , L_{∞} 의 서로 다른 거리 함수를 모두 동시에 지원할 수 있다는 점이다.

제안된 방안의 우수성을 규명하기 위하여 실제 데이터인 S&P 500 주식 데이터와 대용량의 합성 데이터를 이용한 다양한 실험들의 수행을 통하여, 나타난 검색 결과의 질과 검색 성능을 제시하였다. 실험 결과에 의하면 제안된 유사 모델을 기반으로 한 모양 기반 검색은 사용자의 응용 목적에 따라 선택된 거리 함수를 사용하여 질의 시퀀스와 유사한 모양을 갖는 서브시퀀스들을 성공적으로 검색하였으며, 기존의 순차 검색 기법과 비교하여 거리 함수 선택에 따라 수 십배에서 수 백배까지의 성능 개선 효과를 갖는 것으로 나타났다.

참 고 문 헌

- [1] R. Agrawal, C. Faloutsos and A. Swami, "Efficient Simi-

- larity 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] 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.
- [4] 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.
- [5] D. Rafiei, "On Similarity-Based Queries for Time Series Data," In *Proc. IEEE Intl. Conf. on Data Engineering*, pp. 410-417, 1999.
- [6] 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.
- [7] 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.
- [8] Y. S. Moon, K. Y. Whang, and W. K. Loh, "Duality-Based Subsequence Matching in Time-Series Databases," In *Proc. Int'l Conf. on Data Engineering*, IEEE ICDE, pp.263-272, 2001.
- [9] 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.
- [10] 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*, Vol.E84-D, No.1, pp.76-86, 2001.
- [11] W. K. Loh, S. W. Kim and K. Y. Whang, "Index Interpolation : An Approach for Subsequence Matching Supporting Normalization Transform in Time-Series Databases," In *Proc. ACM Intl. Conf. on Information and Knowledge Management(ACM CIKM)*, pp.480-487, 2000.
- [12] 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.
- [13] B. K. Yi, H. V. Jagadish and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences Under Time Warping," In *Proc. Int'l. Conf. on Data Engineering*, IEEE, pp.201-208, 1998.
- [14] 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.
- [15] S. W. Kim, S. H. Park and W. W. Chu, "An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases," In *Proc. Intl. Conf. on Data Engineering*, IEEE, pp.607-614, 2001.
- [16] S. H. Park, S. W. Kim, J. S. Cho and S. Padmanabhan, "Prefix-Querying : An Approach for Effective Subsequence Matching Under Time Warping in Sequence Databases," In *Proc. ACM Intl. Conf. on Information and Knowledge Management(ACM CIKM)*, pp.255-262, 2001.
- [17] 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.
- [18] F. P. Preparata and M. Shamos, *Computational Geometry : An Introduction*, Springer-Verlag, 1985.
- [19] R. Agrawal et al., "Querying Shapes of Histories," In *Proc. Int'l. Conference on Very Large Data Bases*, VLDB, pp. 502-514, Sept., 1995.
- [20] C. S. Perng et al., "Landmarks : A New Model for Similarity-Based Pattern Querying in Time Series Databases," In *Proc. Int'l. Conf. on Data Engineering*, IEEE, pp.33-42, 2000.
- [21] M. Kendall, *Time-Series*, 2nd Edition, Charles Griffin and Company, 1979.
- [22] C. Chatfield, *The Analysis of Time-Series : An Introduction*, 3rd Edition, Chapman and Hall, 1984.
- [23] K. S. Shim, R. Srikant and R. Agrawal, "High-dimensional Similarity Joins," In *Proc. Int'l. Conf. on Data Engineering*, IEEE, pp.301-311, Apr., 1997.
- [24] L. Rabiner and H. H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.
- [25] N. D. Sidiropoulos and R. Bros, "Mathematical Programming Algorithms for Regression-Based Non-Linear Filtering in R^N ," *IEEE Trans. on Signal Processing*, Mar., 1999.
- [26] B. K. Yi and C. Faloutsos, "Fast Time Sequence Indexing for Arbitrary L_p Norms," In *Proc. Int'l. Conf. on Very Large Data Bases*, VLDB, pp.385-394, 2000.
- [27] G. A. Stephen, *String Searching Algorithms*, World Scientific Publishing, 1994.
- [28] S. W. Kim, J. H. Yoon, S. H. Park, T. H. Kim, "Shape-Based Retrieval of Similar Subsequences in Time-Series Databases," In *Proc. ACM Intl. Symp. on Applied Computing(ACM SAC)*, pp.438-445, 2002.



원 정 임

e-mail : jiwon@hallym.ac.kr

1992년 한림대학교 전자계산학과(학사)
1997년 한림대학교 컴퓨터공학과(석사)
1999년 한림대학교 컴퓨터공학과 박사과정
2000년~현재 한림대학교 교양교육부
강의전담

관심분야 : 데이터베이스 시스템, 데이터 마이닝, XML 응용,
의료정보, 데이터베이스 보안



윤 지 희

e-mail : jhyoon@hallym.ac.kr

1982년 한양대학교 전자공학과(학사)
1985년 일본 구주대학교 정보공학과(석사)
1988년 일본 구주대학교 정보공학과(박사)
1998년~1999년 미국 UCLA대학교 전산
학과 방문교수

1988년~현재 한림대학교 정보통신공학부 교수

관심분야 : 시계열 데이터베이스, 데이터 마이닝, XML, 공간
데이터베이스/GIS



김 상 옥

e-mail : wook@ihanyang.ac.kr

1989년 서울대학교 컴퓨터공학과(학사)
1991년 한국과학기술원 전산학과(석사)
1994년 한국과학기술원 전산학과(박사)
1991년 미국 Stanford University, Com-
puter Science Department 방문
연구원

1994년~1995년 한국과학기술원 정보전자연구소 위촉연구원

1999년~2000년 미국 IBM T.J. Watson Research Center

Post-Doc

1995년~2003년 강원대학교 컴퓨터정보통신공학부 부교수

2003년~현재 한양대학교 정보통신대학 정보통신공학부 부교수

관심분야 : 데이터베이스 시스템, 데이터 마이닝, 멀티미디어 정
보 검색, 공간 데이터베이스/GIS, 주기억장치 데이터
베이스, 트랜잭션 관리, 바이오정보공학



박 상 현

e-mail : sanghyun@postech.ac.kr

1989년 서울대학교 컴퓨터공학과(학사)
1991년 서울대학교 컴퓨터공학과(석사)
2001년 UCLA대학교 전산학과(박사)
1991년~1996년 대우통신 연구원

2001년~2002년 IBM T. J. Watson Rese-
arch Center Post-Doctoral Fellow

2002년~현재 포항공과대학교 컴퓨터공학과 교수

관심분야 : 시공간 데이터베이스, 멀티미디어 데이터베이스, 데
이터 마이닝, 데이터 웨어하우징, XML, 분산 데이
터베이스

CONFIDENTIAL