시계열 데이터베이스에서의 모양 기반 서브시퀀스 매칭

김태훈[0*] 윤지희* 김상욱** 박상현***
*한림대학교 컴퓨터공학과
**강원대학교 컴퓨터 정보통신공학부
***IBM T.J. 왓슨 연구소 데이터 매니지먼트 그룹
thkim@cie.hallym.ac.kr, jhyoon@sun.hallym.ac.kr, wook@cc.kangwon.ac.kr, sanghyun@us.ibm.com

Shape-Based Subsequence Matching in Time-Series Databases

Tae-Hoon Kim[0*] Jeehee Yoon* Sang-Wook Kim** Sanghyun Park***
*Dept. of Computer Science, Hallym Univ.
**Dept. of Computer, Information and Communications Engineering, Kangwon National Univ.
***Data Management Group, IBM T.J. Watson Research Center

요 약

모양 기반 검색은 주어진 질의 시퀀스의 요소 값에 상관없이, 모양이 유사한 시퀀스 혹은 부분시퀀스를 찾는 연산이다. 본 논문에서는 시프트, 스케일링, 타임 워핑 등 동일 모양 변환의 다양한 조합을 지원할 수 있는 새로운 모양 기반 유사 검색 모델을 제안하고, 효과적인 유사 부분 시퀀스 검색을 위한 인덱싱과 질의 처리 방법을 제안한다. 또한 실세계의 증권데이터를 이용한 다양한 실험 결과에 의하여, 본 방식이 질의 시퀀스와 유사한 모양의 모든 서브시퀀스를 성공적으로 찾는 것은 물론 순차검색 방법과 비교하여 매우 빠른 검색 효율을 가짐을 보인다.

1. 서 론

시계열 데이터베이스(time-series database)는 객체의 변화되는 값들의 연속으로 구성된 데이터 시퀀스들의 집합으로 주가, 기온, 제품 판매량 데이터 등을 그 예로 들 수 있다. 유사 검색(similarity search)은 주어진 질의 시퀀스(query sequence)와 변화 패턴이 유사한 시퀀스들을 시퀀스 데이터베이스로부터 찾아내는 연산이다.

유사 검색은 응용 목적에 따라 질의 시퀀스의 요소 값의 변화에 기반한 값 기반(value-based) 검색과 질의 시퀀스의 요소 값에 상관없이, 모양에 기반한 모양 기반(shape-based) 검색으로 나눌 수 있다. 기존의 많은 연구들에서 길이 n인 시퀀스를 n차원 상의 한 점으로 간주하고, 점으로 간주된 두 시퀀스의 유사도를 측정하기 위해 유클리드 거리를 사용하며[1], 응용 분야에 적합한 유사 모델(similarity model)을 정의하기 위해 정규화(normalization)[2], 이동 평균(moving average)[3][1], 타임 워핑(time warping)[4][5] 등의 변환(transform)을 지원하는 방식 등이 제안되어 있다.

본 논문에서는 시계열 데이터베이스로부터 주어진 질의 시퀀스와 유사한 모양을 갖는 모든 서브 시퀀스를 검색하는 문제를 다루고자 한다. 이 문제를 해결하기 위해 기존의 값 기반 검색에서 사용하던 다양한 형태의 변환들의 조합을 이용하여, 동일 모양으로 간주되는 이들 변환을 동시에 지원하는 유사 검색 모델을 제안함으로써 효과적인 모양 기반 검색이 가능하도록 한다. 기존 연구에서 이러한 변환을 개별적으로 지원하는 기법들이 다수 제안된 바 있으나, 이들의 다양한 조합을 지원하는 모양 기반 검색 기법은 제안된 바 없다.

본 논문에서는 모양 기반 서브시퀀스 검색을 위한 유사검색 모델을 정의하고, 이를 효과적으로 처리하기 위한 인덱스 구조와 질의 처리 방법에 대하여 논의한다. 또한, 제안된 방안의 우수성을 규명하기 위해 다양한 실험을 수행하여 검색 결과의 질과 성능을 제시한다.

2. 모양 기반 유사 서브시퀀스 검색

2.1 기호 및 용어 정의

유사 모델을 정의하기 전에 본 논문에서 사용되는 용어 및 기호들을 정의하면 다음과 같다. 표 1은 기본적인 기호를 정리한 것이다.

표 1 기호 정의

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

정의 1: 시퀀스 S = (s[i]) (0≤i<Len(S))를 **정규화 변환한** 시퀀스 Norm(S) = (s'[i]) (0≤i<Len(S))는 다음과 같이 정의된다[2].

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

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

정의 2: 시퀀스 S = (s[i]) (0≤i<Len(S))를 이동 평균 계수(moving average coefficient) k(1≤k<Len(S))로 이동 평균 변환한 시퀀스 $MV_k(S)=(s_k[j])$ (0≤j<Len(S)-k+1)는 다음과 같이 정의된다[6].

$$s_k[j] = \frac{1}{k} \times (s[j] + s[j+1] + \cdots + s[j+k-1]) = \frac{1}{k} \times \sum_{i=j}^{j+k-1} s[i] \quad \square$$

이 변환을 통해 시퀀스 내 잡음(noise)의 영향을 제거할 수 있으므로 잡음의 영향 없이 전체적인 변화 경향이 유사한 시퀀스를 파악하는데 유용하다. k값은 응용 분야에 따라 적절하게 선택된다.

정의 3: 길이 n의 두 시퀀스 S와 Q의 유사한 정도를 측정하기 위한 거리 함수 L_p는 다음과 같이 정의된다. L_1은 맨하탄 거리(Manhattan distance), L_2는 유클리드 거리(Euclidean distance), L_∞은 대응되는 각 쌍의 거리 중 최대 거리를 의미한다[7].

* 본 논문은 한국학술진흥재단 선도연구자 연구비의 지원을 받았습니다. (과제번호: KRF-2000-041-E00258)

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

거리 함수 L_p는 현재 많은 응용에서 널리 사용되고 있으나, 비교 대상인 두 시퀀스의 길이가 같아야 한다는 제한이 있다[4].

정의 4: 두 시퀀스 S와 Q 간의 타임 워핑 거리(time warping distance) D_{tw}는 다음과 같이 재귀적으로 정의된다[8]. D_{base}는 기본 거리 함수로서 L_p 중 응용에 적합한 것을 선택하여 사용한다.

(1) D_{tw} = (⟨⟩,⟨⟩) = 0,
(2) D_{tw}(S,⟨⟩) = D_{tw}(⟨⟩,Q) = ∞,
(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))) □

타임 워핑 변환은 시퀀스 내의 각 요소의 값을 임의의 수만큼 반복시켜 서로 다른 길이의 두 시퀀스를 동일한 길이로 변환한다[5]. 이 변환은 비교 대상인 두 시퀀스의 길이가 서로 달라서 유클리드 거리로 유사 정도를 측정할 수 없는 경우에 유용하다.

2.2 유사 모델

본 연구에서는 모양 기반 유사 검색을 지원하기 위해, 앞에서 언급한 정규화 변환, 이동 평균 변환, 타임 워핑 변환을 모두 지원하는 다음의 유사 정도 측정 함수를 채택한다.

정의 5: 두 시퀀스 혹은 서브 시퀀스 S와 Q 간의 유사한 정도 D(S,Q)는 다음과 같이 정의된다.

D(S,Q) = D_{tw}(Norm(MV_k(S)), Norm(MV_k(Q))) □

정의 5에 나타난 바와 같이, D(S, Q)는 S와 Q를 각각 (1) k-이동 평균 변환, (2) 정규화 변환, (3) 타임 워핑 변환을 차례로 거친 후의 거리로 정의된다. 특히, 타임 워핑 거리 계산에서는 거리 함수 D_{base}로서 L_∞를 사용한다. 이를 위해 정의 4에 나타난 타임 워핑 거리는 다음 정의 6과 같은 형태로 변형된다.

정의 6:

(1) D_{tw} = (⟨⟩,⟨⟩) = 0,
(2) D_{tw}(S,⟨⟩) = D_{tw}(⟨⟩,Q) = ∞,
(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_{tw}의 값은 동적 프로그래밍 기법(dynamic programming)에 의한 거리 축적 테이블(cumulative distance table)을 이용하여 효율적으로 계산될 수 있다[8].

정의 5를 이용하여 시계열 데이터베이스를 위한 모양 기반 유사 검색 문제를 다음과 같이 정의한다.

질의 시퀀스 Q, 유사 허용치 ε, 이동 평균 계수 k가 주어지면, D(X,Q)(=D_{tw}(Norm(MV_k(X)), Norm(MV_k(Q))))의 값이 ε 이하인 데이터베이스 내의 서브시퀀스 X를 찾고, X를 포함하는 시퀀스 S와 S내에서 X의 시작 위치를 반환한다. D(X,Q)의 값이 ε 이하라는 의미는 Norm(MV_k(X))를 타임 워핑 변환한 시퀀스의 각 요소가 Norm(MV_k(Q))를 타임 워핑 변환한 시퀀스의 대응되는 요소의 일정 범위 ε 내에 존재함을 의미한다.

본 논문에서 제안하는 유사 모델을 기반으로 한 모양 기반 유사 검색 기법은 시프팅(shifting), 스케일링(scaling), 타임 워핑 등 다양한 변환을 지원할 뿐만 아니라, 이동 평균 변환을 지원하므로 시퀀스 내에서 발생하는 잡음의 영향도 최소화 할 수 있다.

D_{base}로서 L_1을 사용하는 참고 문헌[4][5]과 달리, L_∞를 사용하는 이유는 다음과 같다. L_1을 사용하는 경우, 타임 워핑 거리는 변환된 두 시퀀스의 각 대응되는 요소 쌍의 거리의 합으로 나타나므로 시퀀스와 질의 시퀀스의 길이에 큰 영향을 받게 된다. 더구나 본 논문에서 고려하는 서브시퀀스 매칭 문제에서는 같은 시퀀스로부터 추출되는 서브시퀀스들 조차도 길이 차가 매우 크기 때문에 이 영향은 더욱 커지게 된다. 따라서 질의를 작성하는 사용자가 해당 데이터베이스의 특성에 맞는 적절한 ε를 결정하기가 어렵다. 반면, L_∞를 사용하면, 시퀀스의 길이에 영향을 받지 않고 일관된 ε를 사용할 수 있어 이러한 질의 작성 부담을 덜 수 있다.

3. 인덱싱 및 질의 처리

3.1 서브시퀀스 트리

2장에서 정의한 유사 모델을 기반으로 하는 모양 기반 서브시퀀스 검색을 효과적으로 처리하기 위하여 서브시퀀스 트리 인덱스 구조를 사용한다. 서브시퀀스 트리는 시퀀스 내의 모든 서브시퀀스를 리프 노드로 포함하는 일종의 접미어 트리(suffix tree)[9]로 볼 수 있다.

접미어 트리는 다수의 시퀀스들을 인덱싱하기 위해 주로 사용되며, 주어진 질의 시퀀스와 정확히 일치하는 서브시퀀스의 위치를 신속히 찾는 데 유용하다. 참고문헌[4]에서는 타임워핑 변환을 지원하는 효과적인 서브시퀀스 검색 방법이 제안되어 있다. 접미어 트리는 저장되는 접미어들이 많은 공통 접두어 서브시퀀스를 가질 때, 좋은 압축 효과를 갖는다. 그러나 시퀀스의 요소 값은 실수 타입을 가지므로, 공통의 접두어 서브시퀀스를 가질 가능성이 매우 낮다. 참고 문헌[4]에서는 이를 해결하기 위해 도메인 분류 방법을 제안하고 있다. 도메인 분류는 요소 값을 심볼로 변환하기 위하여 요소 값의 도메인을 여러 범위로 분류하는 작업이다. 즉, 서로 다른 요소 값이라도 같은 범위에 속하면 동일한 심볼로 변환되기 때문에 접미어들이 공통의 접두어 서브시퀀스를 가질 가능성이 상대적으로 높아진다.

접미어 트리에서는 인덱싱 대상이 되는 모든 시퀀스의 가능한 접미어만을 저장하여, 유사한 서브시퀀스 검색이 가능하다[4]. 그러나 본 연구에서 채택한 유사 모델에서는 정규화 변환을 지원하므로 다음과 같이 모든 시퀀스의 모든 가능한 서브시퀀스를 트리 내에 저장하여야 한다.

정의 1에 나타난 바와 같이 한 서브시퀀스 S를 정규화 변환하기 위해서 Max(S)와 Min(S)을 사용한다. S의 한 접미어 S_a와 S_a의 접두어 서브시퀀스 S_b를 예로 들면, Max(S_a)와 Max(S_b), Min(S_a)과 Min(S_b)가 일반적으로 서로 다른 값을 가지게 되기 때문에 Norm(S_b)는 Norm(S_a)의 접두어가 되지 않을 수 있다. 따라서 Norm(S_a)만을 참조하여 Norm(Q)와의 타임 워핑 거리가 ε 이하인 Norm(S_b)를 찾을 수 없게 된다. 그러므로 정규화 변환을 지원하는 경우에는 접미어 외에도 발생 가능한 모든 서브시퀀스들도 인덱스 내에 저장해야 한다.

본 연구에서는 접미어 트리와 구조적으로 동일하며, 각 시퀀스 내의 모든 서브시퀀스들을 포함하는 인덱스를 서브시퀀스 트리(subsequence tree)라 정의하고 이를 인덱스 구조로 사용한다. 서브시퀀스 트리 구성 알고리즘을 그림 1에 보인다.

단계 1: 이동 평균 변환
이동 평균 계수 k 값은 해당 응용에 적합하게 선택하여 시계열 데이터베이스 내의 각 시퀀스 S에 대해 k-이동평균 변환을 수행하여 MV_k(S)를 구한다.
단계 2: 서브 시퀀스 추출
각 MV_k(S)로부터 모든 서브시퀀스 X를 추출한다. 이 때 너무 짧은 길이의 X가 결과로서 큰 의미가 없을 경우 추출될 서브시퀀스의 최소 길이 L를 지정해서 제한할 수 있다.
단계 3: 정규화 변환
각 X에 대해 고유의 Max(X)와 Min(X) 값을 이용하여 정규화 변환을 수행하여 Norm(X)를 구한다.
단계 4: 도메인 분류를 이용한 심볼 변환
각 Norm(X)에 대해 도메인 분류를 이용해 심볼 변환을 수행하여 Sym(X)를 구한다.
단계 5: 트리 구성
Sym(X)들을 대상으로 서브시퀀스 트리를 구성한다.

그림1 서브시퀀스 트리 구성 알고리즘.

3.2 질의 처리

위에서 제안한 인덱싱 전략을 기반으로 하는 모양 기반 서브시퀀스 검색 방식은 다음과 같다. 질의는 이동 평균된 질의 시퀀스를 대상으로 한다. 그림 2는 서브시퀀스 트리를 이용해 질의 시퀀스 Q와 D_{tw}가 ε 이하인 유사 서브시퀀스들을 데이터베이스로부터 검색하는 알고리즘 S-ST를 나타낸다. 단, 여기서 서브시퀀스 트리는 심볼로 변환된 서브시퀀스들을 대상으로 구성되어 있으므로, 서브시퀀스와 질의 사이의 타임 워핑 거리 D_{tw}를 직접 계산할 수 없다. 따라서 D_{tw}의 하한 함수인 $D_{tw\text{-}lb}$를 다음과 같이 재귀적

으로 정의하여 사용한다. 그 결과 착오채택(false alarms)이 발생할 수 있으며, 이를 제거하기 위하여 함수 PostProcess()에 의한 후처리를 수행한다.

정의 7: 여기서, CS는 심볼로 변환된 서브시퀀스를 의미한다. 또한 A는 First(CS)에 해당하는 심볼을 나타내며, b는 First(Q)에 해당하는 실제의 실수 값을 나타낸다. A.lb와 A.ub는 각각 A가 속한 도메인의 최소 요소 값과 최대 요소 값을 나타낸다.

(1) $D_{tw\text{-}lb}((),()) = 0$

(2) $D_{tw\text{-}lb}(CS,()) = D_{tw\text{-}lb}((),Q) = \infty$

(3) $D_{tw}(CS,Q) = \max(D_{base\text{-}lb}(First(CS),First(Q)),$
$\min(D_{tw\text{-}lb}(CS,Rest(Q)), D_{tw\text{-}lb}(Rest(CS), Q), D_{tw\text{-}lb}(Rest(CS), Rest(Q))))$

(4) $D_{base\text{-}lb}(A,b) = 0$ (if $A.lb \le b \le A.ub$)
$= b - A.ub$ (if $b > A.ub$)
$= A.lb - b$ (if $b < A.lb$) □

```
Algorithm S-ST
candidate-ans ← VisitNode-and-FindAnswers(root R, Q, ε, emptyTable);
actual-ans ← PostProcess(candidate-ans);
return actual-ans;
End

Algorithm VisitNode-and-FindAnswers(Node N, Q, ε, cumDistTable T)
For i ← 1 to |CN| do {
    for j ← 1 to |label(N, CN)| do {
        CTi ← Addrow(T, Q, label(N, CN)[j], D_tw-lb());
        Let mDist be the minimum column value of the new row;
        if mdist > ε goto next-CN;
    }
    Let dist be the last column value of the new row;
    if (CN is a leaf node && dist ≤ ε) then {
        answerSet ← FindAnswer(CN);
        goto next-CN;
    }
    answerSet ← answerSet ∪ VisitNode-and-FindAnswers(CN, Q, ε, CTi)
    label : next-CN;
}
return answerSet;
End
```

그림2 S-ST: 서브시퀀스 트리를 이용한 유사검색 알고리즘.

4. 성능 평가

실제 데이터인 미국의 S&P500 주식 데이터를 이용한 실험을 통하여 서브시퀀스 트리를 이용한 모양 기반 유사 서브시퀀스 검색 알고리즘의 성능을 측정한다.

그림 3은 제안된 기법을 이용한 유사 서브시퀀스 검색의 예를 보인다. 질의로서는 처음 그림의 이동 평균선 상의 굵은 선으로 표시된 이중 바닥(double bottoms) 패턴의 서브시퀀스를 이용하였으며, 나머지 그림들의 이동 평균선 상의 굵은 선으로 표시된 부분이 검색된 서브시퀀스를 나타낸다. 사용자가 원하는 이중 바닥을 갖는 서브시퀀스들을 성공적으로 검색하였음을 볼 수 있다.

S-ST의 검색 성능을 평가하기 위하여 순차 검색 방식을 그 비교 대상으로 하며, 평균 질의 처리 시간을 비교 측정하였다. 이를 위해 다음의 실험을 실시하였다. 시퀀스 데이터는 기본적으로 10-이동 평균 변환된 길이 100의 시퀀스 200 개를 사용하였고 질의 시퀀스의 평균 길이는 20으로 하였다. 도메인 분류 개수는 실험을 통하여 인덱스 크기와 질의 처리 시간 사이의 변화율을 고려한 최적의 값을 선정하였으며, 이 실험에서는 60이 선택되었다.

그림 4는 유사 허용치 ε의 변화에 따른 질의 처리 시간의 비교 결과를 나타낸다. ε는 평균 10개의 최종 결과를 얻기 위한 값부터 30개, 100개, 300개의 최종 결과를 얻기 위한 값을 사용하였다. 실험 결과, S-ST는 순차 검색 방식에 비하여 유사 허용치의 변화에 따라 약 20배에서 35배까지의 높은 성능 향상을 보이고 있다.

그림 5는 시퀀스 개수 변화에 따른 질의 처리 시간의 비교 결과이다. ε는 시퀀스 길이 160의 데이터에서 평균 약 100개의 최종 결과를 얻기 위한 값(ε=0.104471)을 사용하였다. 실험 결과, S-ST는 순차 검색 방식에 비해 시퀀스 길이의 변화에 따라 약 25배에서 66배까지의 높은 성능을 보이고 있다.

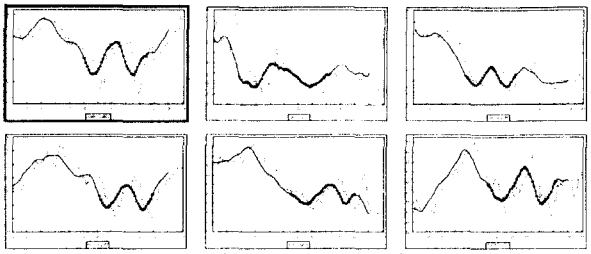

그림 3 유사 서브시퀀스 검색 예.

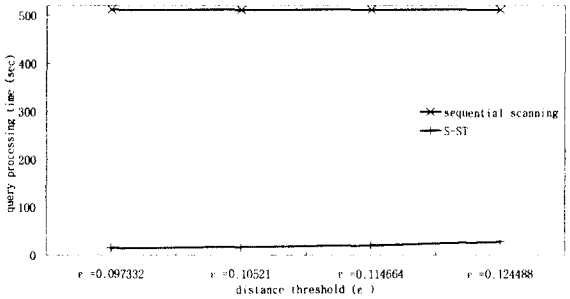

그림 4 유사 허용치 변화에 따른 질의 처리 시간의 비교.

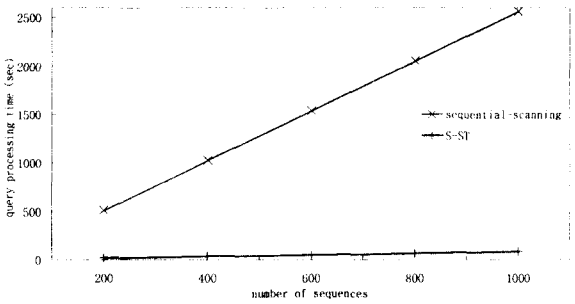

그림 5 시퀀스 수 변화에 따른 질의 처리 시간의 비교.

참고 문헌

[1] 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.

[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 DataBases*, VLDB, pp. 490-501, Sept. 1995.

[3] 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. Int'l. Conf. on Information and Knowledge Management*, ACM CIKM, 2000.

[4] S. Park, W. W. Chu, J. Yoon, and C. Hsu, "Efficient Searches for Similar Subsequences of Difference Lengths in Sequence Databases," In *Proc. Int'l. Conf. on Data Engineering*, IEEE, pp. 23-32, 2000.

[5] 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.

[6] C. Chatfield, *The Analysis of Time-Series: An Introduction*, 3rd Edition, Chapman and Hall, 1984.

[7] 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.

[8] L. Rabiner and H. H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

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