# 쉬프팅을 지원하는 유사 시퀀스 검색

강석원*, 김수현*, 송준호**, 김상욱**+
*한양대학교 컴퓨터공학부
** 한양대학교 컴퓨터 소프트웨어학과
e-mail : *{kanggo1111, chriskim}@hanyang.ac.kr
**{junonjuno, wook}@hanyang.ac.kr

# Similar Sequence Search Supporting Shifting

Seok-Won Kang*, Su-Hyun Kim*, Junho Song**, and Sang-Wook Kim**+
*Dept. of Computer Science, Hanyang University
**Dept. of Computer and Software, Hanyang University

**요 약**

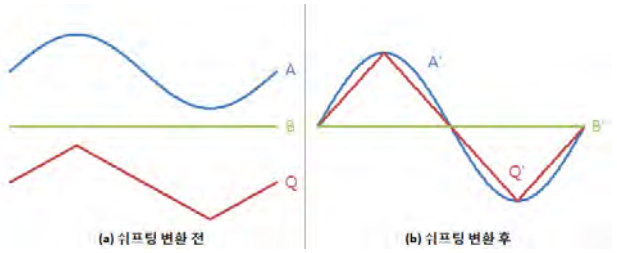
기존의 유사 시퀀스 검색 방법들은 시퀀스 간의 거리를 이용하여 유사도를 판단하였다. 그러나 이러한 방법은 시퀀스의 형태를 고려하지 못하는 문제가 있다. 본 논문에서는 시퀀스를 쉬프팅하여 형태를 고려하고 시퀀스 간의 거리를 이용해 유사한 시퀀스를 검색하는 방법을 제안하고자 한다.

## 1. 서론

시계열 데이터는 일정 주기의 시간에 따른 연속된 값의 집합으로 일반적으로 시퀀스 형태로 표현된다. 시계열 데이터로 구성된 데이터 베이스를 시계열 데이터 베이스라 하며 의료, 금융, 엔터테인먼트 분야 등에 널리 사용되고 있다[1, 2]. 데이터 베이스에 저장된 데이터 시퀀스 중에서 주어진 질의 시퀀스와 유사한 시퀀스를 검색하는 과정을 유사 시퀀스 검색이라고 한다[1, 2].

유사 시퀀스 검색에 관한 기존의 연구[3, 4]는 시퀀스 간의 거리를 기준으로 가까울 수록 유사도가 높다고 판단하여 질의 시퀀스와 유사한 시퀀스를 검색하였다. 그러나 시퀀스 간의 거리를 이용해 유사 시퀀스 검색을 시행할 경우, 시퀀스의 형태는 고려되지 않는다는 한계점이 있다.

[그림 1-(a)]에서 질의 시퀀스 Q 에 대해 데이터 시퀀스 A 와 B 의 유사도를 거리만을 이용하여 계산할 경우, 질의 시퀀스 Q 와 B 의 유사도가 시퀀스 A 와의 유사도보다 높을 것이다. 그러나 시퀀스의 형태를 고려하여 유사검색을 시행할 경우, 질의 시퀀스 Q 에 대해 시퀀스 A 가 시퀀스 B 보다 높은 유사도를 가질 것이다. [그림 1-(b)]의 시퀀스 Q' , A' , B' 는 각각 Q, A, B 를 각 시퀀스의 중간 값만큼 쉬프팅한 것이다. 시퀀스 A' 과 B' 를 대상으로 질의 시퀀스 Q' 에 대해 유사 시퀀스 검색을 시행할 경우, 시퀀스 A' 이 B' 보다 질의 시퀀스 Q' 와 더 유사하다고 판단 할 것이다.




(그림 1) 시프팅 변환 전 후 시퀀스.

이와 같이 쉬프팅 할 경우, 시퀀스 값의 범위 간의 차이가 줄어들어 시퀀스의 형태를 고려한 유사도 측정이 가능해진다. 따라서 본 논문에서는 시퀀스를 쉬프팅하여 유사 시퀀스를 검색하는 방법을 제안하고자 한다.

## 2. 제안하는 방법

본 장에서는 시퀀스를 중간 값만큼 쉬프팅하여 색인을 생성하고, 타임 워핑 거리를 이용하는 유사 시퀀스 검색 방법을 제안한다.

본 논문에서 제안하는 유사 시퀀스 검색은 대상이 되는 두 시퀀스의 길이가 서로 같은 전체 매칭(whole matching)의 경우로 한정하며, R-Tree 를 이용한 색인을 생성한다[4]. 또한 해당 유사 시퀀스 검색은 질의 시퀀스와 유사도가 가장 높은 K 개의 시퀀스 (Top-K)를 검색하는 K-NN(K-nearest neighbor) 알고리즘을 사용한다[4].

알고리즘 실행에 앞서 시퀀스 데이터베이스의 색인을 생성하기 위해 다음과 같은 과정을 거친다. 첫째, 모든 데이터 시퀀스 C 에 대해 각 시퀀스의 중간 값을 오프셋으로 쉬프팅하여 C' 를 생성한다. C 와 C' 와의 관계는 다음과 같다.

$$C'_i = C_i - C_m \ (C_m : C\text{의 중간 값})$$

둘째, R-Tree 를 이용한 색인 생성 시 차원 저주 현상을 피하기 위해 시퀀스 C' 를 PAA 를 이용해 저차원으로 변환한 뒤 색인을 형성한다.

질의 시, 주어진 질의 시퀀스 Q 또한 중간 값을 오프셋으로 쉬프팅하여 Q' 를 생성한다. Q 와 Q' 과의 관계는 다음과 같다.

$$Q'_i = Q_i - Q_m \ (Q_m : Q\text{의 중간 값})$$

유사 시퀀스 검색 알고리즘은 다차원 공간 구조에서 질의 시퀀스 Q 와 거리가 가까운 순으로 색인의 노드 및 객체를 조회하기 위해 우선순위 큐를 사용하며 다음과 같이 진행된다. 먼저 큐에 색인의 루트 노드를 넣는다. 그 후 큐의 항목이 모두 사라질 때까지 큐의 최상위 항목을 꺼내어 항목의 종류에 따라 다음과 같이 처리하는 과정을 반복한다. 최상위 항목이 리프 노드일 경우, 리프 노드에 포함된 각 저차원 시퀀스와 질의 시퀀스 간의 하한 거리(LB_PAA)를 구한다. 하한 거리가 임계 값보다 작을 경우, 저차원 시퀀스를 우선순위 큐에 넣는다. 만약 꺼낸 항목이 리프 노드가 아닌 다른 노드일 경우, 질의 시퀀스와 하한 거리(MINDIST)를 계산한다. 하한 거리가 임계 값보다 작을 경우, 해당 노드를 우선순위 큐에 넣는다. 최상위 항목이 저차원 시퀀스일 경우, 디스크에서 원본 시퀀스를 읽어온다.

---

+ 교신저자

본 연구는 (1) 미래창조과학부 및 정보통신기술진흥센터의 대학 ICT 연구센터육성 지원사업 (IITP-2015-H8501-15-1013), (2) 미래창조과학부 및 정보통신기술진흥센터의 서울어코드활성화지원사업 (IITP-2015-R0613-15-1149)의 연구결과로 수행되었음. 또한, 본 연구는 (3) 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발 사업의 일환으로 수행하였음.(B0101-15-0266, (딥뷰-1 세부) 실시간 대규모 영상 데이터 이해·예측을 위한 고성능 비주얼 디스커버리 플랫폼 개발)

읽어온 원본 시퀀스와 질의 시퀀스 간의 하한 거리(LB_Keogh)가 임계값 보다 작을 경우, 실제 타임 워핑 거리(DTW)를 계산하고, DTW 값이 임계 값보다 작을 경우에 한하여 우선순위 큐에 원본 시퀀스를 집어넣는다. 만약 꺼낸 항목이 원본 시퀀스 일 경우, 결과 리스트에 추가하고 결과 리스트의 수가 K 개이면 결과 리스트를 반환한다.

```
Index root: R-tree
Variable queue: Minimum Priority Queue;
Variable result: List
Variable t: threshold(=MAX_DOUBLE)
1. queue.push(root, 0);
2. while not queue.IsEmpty() do
3.      top = queue.pop();
4.      if top is Original Sequence
5.          result.insert(top);
6.          if |result| = k
7.              return result
8.      if top is leaf entry(=PAA data)
9.          C = top.orginalSequence;
10.          if LB_Keogh(C, Q) < t
11.              if DTW(C, Q) < t
12.                  queue.push(C, DTW(C, Q));
13.                  cnt = K- result.Count();
14.                  for each queue entry(=q)
15.                      if(q.type == original sequence)
16.                  cnt--;
17.                  if(cnt == 0)
18.                      t = q.distance;
19.                  break;
20.          if top is leaf node
21.              for each leaf entry(=E)
22.                  if LB_PAA(E, Q) < t
23.                      queue.push(E);
24.          if top is non-leaf node
25.              for each child node(=N)
26.                  if MINDIST(N, Q) < t
27.                      queue.push(N);
```

(그림 2) 검색 알고리즘.

## 3. 실험

본 논문에서 제안하는 유사 시퀀스 검색의 성능을 입증하기 위해 Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz 2.40GHz, 8GB RAM, Windows 7 64bit 환경에서 실험을 진행하였다.

본 실험에서 사용한 데이터 셋은 멘션, 해쉬태그, UCR Mix 이며, 모든 데이터 셋의 시퀀스 길이는 128 이다. 또한, 각 데이터 셋에 대해 차원 감소 길이는 8 과 16 으로 색인을 생성하였으며, 워핑 범위는 0.1 로 셋팅하였다.

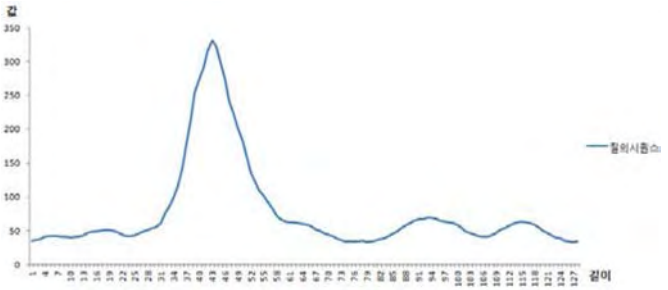
### 3.1 유사 시퀀스 검색 결과 형태 비교

유사 시퀀스 검색 시 쉬프팅의 성능을 검증하기 위해 다음과 같은 실험을 진행하였다. 원본 시퀀스 데이터 베이스와 쉬프팅하여 전처리 한 시퀀스 데이터 베이스의 색인을 생성하였다. 쉬프팅 전의 결과[그림 4]와 후의 결과[그림 5]를 비교 시, 쉬프팅 전의 시퀀스의 형태가 후의 시퀀스의 형태보다 질의 시퀀스와 유사한 것을 알 수 있었다. 따라서 제안하는 방법이 시퀀스의 형태를 고려한 유사 시퀀스 검색이 가능하다.

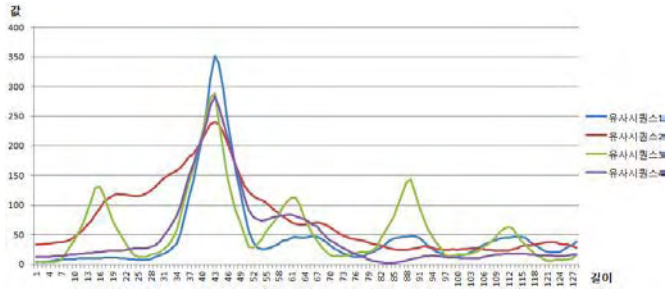
### 3.2 정확도 및 속도 비교

본 실험에서는 제안하는 방법의 정확도와 속도를 판단하기 위해 다음과 같은 실험을 진행하였다. 제안하는 방법(실험군)과 순차 검색(대조군)을 통해 도출한 유사 시퀀스 결과를 비교해 정확도를 측정하였으며, 모든 실험의 정확도는 100%였다.

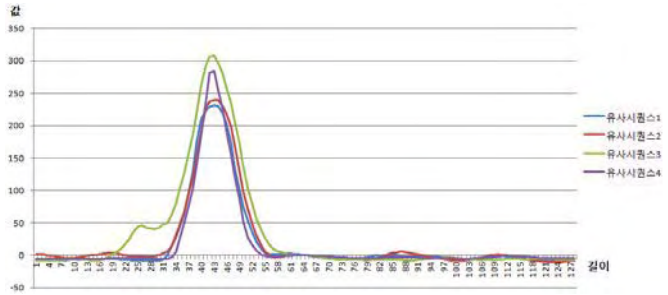
[그림 6]은 각 실험의 검색 연산 속도를 나타낸 것으로 평균시간은 각각 제안하는 방법과 순차 검색을 100 회 수행한 평균시간이다. 본 논문에서 제안하는 방법이 순차 검색에 비해 검색 속도가 최대 2.5 배 빠른 것을 알 수 있었다.



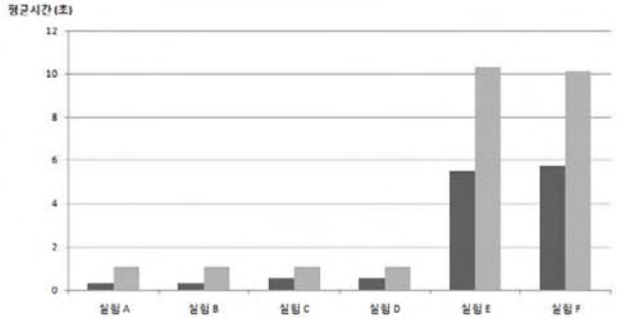

(그림 3) 질의 시퀀스.




(그림 4) 쉬프팅 전 Top-4 검색 결과.




(그림 5) 쉬프팅 후 Top-4 검색 결과.




(그림 6) 제안하는 방법 및 순차 검색 수행 시간.

## 4. 결론

기존의 연구에서는 시퀀스간의 거리를 기준으로 유사도를 판단하므로 시퀀스의 형태를 고려하지 못하였다. 본 논문에서는 시퀀스를 쉬프팅하여 유사도를 판단하는 방법을 제안하였다. 제안하는 방법은 기존의 방법의 단점을 보완하는 동시에 정확도 및 속도에서 우월하므로, 형태를 고려한 유사 시퀀스 검색에 유리하다.

## 5. 참고문헌

[1] R. Agrawal et al., “Efficient Similarity Search in Sequence DataBases,” In FODO, pp.69-84, Oct., 1993.

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

[3] R. Agrawal et al., “Efficient similarity search in sequence databases,” Springer Berlin Heidelberg, 1993.

[4] E. Keogh et al., "Exact indexing of dynamic time warping." In VLDB, pp. 406, 2002.