KIPS Tr. Software and Data Eng.
Vol.6, No.11 pp.507~520 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2017.6.11.507

An Index-Based Search Method for Performance Improvement of Set-Based Similar Sequence Matching

Juwon Lee† · Hyo-Sang Lim††

ABSTRACT

The set-based similar sequence matching method measures similarity not for an individual data item but for a set grouping multiple data items. In the method, the similarity of two sets is represented as the size of intersection between them. However, there is a critical performances issue for the method in twofold: 1) calculating intersection size is a time consuming process, and 2) the number of set pairs that should be calculated the intersection size is quite large. In this paper, we propose an index-based search method for improving performance of set-based similar sequence matching in order to solve these performance issues. Our method consists of two parts. In the first part, we convert the set similarity problem into the intersection size comparison problem, and then, provide an index structure that accelerates the intersection size calculation. Second, we propose an efficient set-based similar sequence matching method which exploits the proposed index structure. Through experiments, we show that the proposed method reduces the execution time by 30 to 50 times then the existing methods. We also show that the proposed method has scalability since the performance gap becomes larger as the number of data sequences increases.

Keywords : Set Similarity, Similar Sequence Matching, Set Index

집합 유사 시퀀스 매칭의 성능 향상을 위한 인덱스 기반 검색 방법

이 주 원† · 임 효 상††

요 약

집합 유사 시퀀스 매칭 방법은 유사한 정도를 나타내는 척도로 교집합을 기반으로 한 유사도를 사용한다. 그러나 교집합 크기를 계산하는 과정에 시간이 오래 걸릴 뿐만 아니라, 유사한 시퀀스를 찾기 위해서 수많은 집합 간 교집합 크기를 구해야 하므로 수행 시간이 오래 걸리는 성능상의 문제가 있다. 본 논문에서는 이러한 성능상의 문제를 해결하기 위해 인덱스 기반의 검색 방법을 사용하여 집합 기반 유사 시퀀스 매칭을 빠르게 수행하는 방법을 제안한다. 제안하는 방법은 크게 두가지로 구분된다. 첫 번째로 집합 시퀀스 유사도 문제를 교집합의 크기 비교 문제로 정형적으로 변환하고, 교집합의 크기를 빠르게 찾을 수 있는 인덱스 구조를 제안한다. 두 번째로 제안한 인덱스 구조를 사용하여 집합 기반 유사 시퀀스 매칭을 효율적으로 수행할 수 있는 방법을 제안한다. 성능 평가 결과, 제안하는 방법이 기존 방법에 비해 최대 30배에서 50배의 수행 시간 단축이 있음을 보인다. 또한 데이터 시퀀스의 개수가 증가할수록 수행시간의 차이가 점점 커지므로, 대용량 데이터 처리에 적절함을 보인다.

키워드 : 집합 유사도, 유사 시퀀스 매칭, 집합 인덱스

1. 서 론

1.1 연구 배경 및 동기

최근 스마트폰의 보급과 소셜 네트워크 서비스의 발전으로 인해 다양한 데이터가 스마트폰 사용자들로부터 데이터 스트림의 형태로 끊임없이 실시간으로 생성되는 빅데이터 환경이 도래하였다. 데이터스트림이란 시간의 흐름에 따라 순차적으로 무한하게 생성되는 데이터를 말한다[1]. 데이터 스트림의 특징으로 데이터가 시간 순서대로 빠르고 무한하게 동적(dynamic)으로 생성된다는 것을 들 수 있다. 데이터 스트림의 예로는 각종 센서 데이터, 네트워크 패킷 데이터 등이 존재한다.

유사 시퀀스 매칭이란 수많은 데이터 시퀀스(data sequence)들 중에서 사용자에 의해 주어지는 질의 시퀀스(query sequence)와 유사한 데이터 시퀀스를 검색하는 문제이다. 유사 시퀀스 매

※ 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신진흥센터의 지원을 받아 수행된 연구임(No.R7117-17-0214, 데이터 스트림 정제를 위한 지능형 샘플링 및 필터링 기술 개발).
† 비 회 원 : 연세대학교 전산학과 석사과정
†† 종신회원 : 연세대학교 컴퓨터정보통신공학부 교수
Manuscript Received : June 30, 2017
Accepted : July 26, 2017
* Corresponding Author : Hyo-Sang Lim(hyosang@yonsei.ac.kr)

칭의 예로, 여러 사람들의 DNA 시퀀스가 저장되어 있을 때, 특정 질병을 유발하는 DNA 시퀀스를 가지고 있는 사람들을 찾아내는 것이 있다.

시퀀스를 구성하는 데이터로 연속성을 가지는 수치 데이터를 사용하여 유사 시퀀스 매칭을 수행할 경우, 비슷한 형태를 가지는 유사한 데이터 시퀀스를 찾을 수 있다. 이것은 연속성을 가지는 데이터의 경우, 데이터 값 간의 거리로 유사한 정도를 계산할 수 있기 때문에 가능하다. 하지만 범주형 데이터에 대한 유사 시퀀스 매칭은 각 범주간의 유사한 정도가 수치적으로 정해져 있지 않기 때문에 시퀀스 간의 데이터 값들이 시간 순서에 따라 정확하게 일치해야만 유사하다고 판단한다. 이로 인해 실제로 몇몇 데이터의 순서만 바뀌었을 뿐인 유사한 데이터 시퀀스가 유사하지 않다고 판단 될 수 있다. 즉, 시간 기준이 엄격하기에 실제로 유사한 시퀀스이지만 유사하지 않은 시퀀스로 판단되는 시간 불일치(time mismatch) 문제가 발생한다. Fig. 1은 범주형 데이터 시퀀스에 대해 유사 시퀀스 매칭을 수행할 시 발생하는 유사 시퀀스와 시간 불일치 문제가 발생하는 시퀀스를 그림으로 표현한 것이다.

Fig. 1의 (a)에서 시퀀스 S_1과 시퀀스 S_2는 시간 순서에 따라 5개의 데이터가 일치하는 것을 확인할 수 있다. 하지만 Fig. 1의 (b)에서 시퀀스 S_1과 시퀀스 S_3는 시퀀스를 구성하는 데이터들 중 5개가 일치하지만, 시간 순서가 달라 전혀 유사하지 않은 시퀀스로 판단된다. 이와 같이 일치하는 데이터들이 많지만, 순서에 따라 유사하지 않은 시퀀스로 판단되는 문제를 시간 불일치 문제라고 한다.

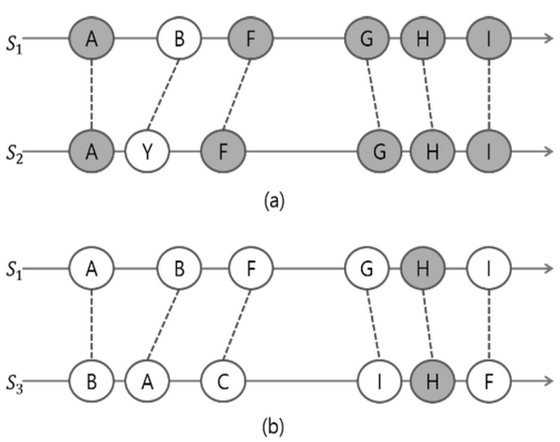


Fig. 1. Time Mismatch Problem

이러한 시간 불일치 문제를 해결하기 위해 집합 기반 유사 시퀀스 매칭이 제안되었다. 집합 기반 유사 시퀀스 매칭은[2-4] 시퀀스의 데이터들을 일정한 크기 단위인 집합으로 나누어 집합의 시퀀스로 표현한다. 집합의 시퀀스로 표현할 경우, 집합으로 묶인 간격 안에서는 각 데이터의 순서가 정확히 일치하지 않아도 유사하다고 판단할 수 있다. 집합 기반 유사 시퀀스 매칭을 그림으로 나타내면 Fig. 2와 같이 표현된다. Fig. 1에서 시퀀스 S_1과 시퀀스 S_3은 순서가 달라서 유사하지 않은 시퀀스라고 판단되었다. 하지만 Fig. 2와 같이 일정 크기의 집합으로 구분할 경우, 각각의 집합들에 대해 2개씩 일치하며 시퀀스 전체로 보면 총 4개가 일치한다는 것을 알 수 있다. 즉, 집합 기반 유사 시퀀스 매칭은 기존 유사 시퀀스 매칭의 엄격한 시간 기준을 완화시키는 효과를 얻을 수 있어 시간 불일치 문제를 해결할 수 있다.

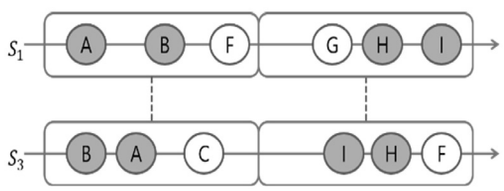


Fig. 2. Solution of Time Mismatch Problem

이러한 기존의 집합 기반 유사 시퀀스 매칭 연구는 시간 불일치 문제의 해소에 중점을 두었으나 성능적인 측면에 대해서는 고려되지 않았다. 집합 기반 유사 시퀀스 매칭은 시퀀스를 집합으로 나누고 각 집합 간에 얼마나 유사한지, 즉 공통으로 가지는 원소가 얼마나 많은 지를 계산해야만 한다. 따라서 수행 시간이 오래 걸리는 성능상의 문제가 있다. 집합 기반 유사 시퀀스 매칭을 보다 유용하게 사용하기 위해서는 집합 간 교집합의 크기를 계산하는 과정의 성능을 향상시키는 것이 필요하다.

1.2 연구 내용 및 목적

본 논문에서는 집합 간 유사도를 계산하는 과정에서 발생하는 성능 문제를 해결한다. 1.1절에서 설명한 바와 같이 집합 유사 시퀀스 매칭에서의 시퀀스 간 유사도는 집합 간의 유사도, 즉 두 집합 간의 교집합 크기를 이용하여 계산할 수 있다. 본 논문의 저자들은 간단한 집합 인덱스 구조를 사용하여 교집합 계산을 빠르게 수행할 수 있는 기초적인 방법을 제시한 바 있다[5]. 본 연구는 이러한 기초 연구를 확장하여 특정 집합과 유사한 집합을 찾는 집합 간의 유사도 비교 문제를 교집합 크기 비교 문제로 변환할 수 있음을 보다 정형적으로 보인다. 그리고 집합 인덱스 구조를 개선하고 불필요한 데이터 접근을 줄이기 위한 다양한 최적화 방법을 제시한다. 또한 이렇게 새롭게 제시한 방법이 착오기각(false dismissal) 없이 모든 유사 시퀀스를 빠짐없이 찾을 수 있음을 정형적으로 증명한다.

1.3 논문의 구성

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로서 집합 유사도와 유사 시퀀스 매칭에 대한 기존 연구들에 대해 살펴본다. 제 3장에서는 본 논문에서 제안하는 방법과 제안하는 방법을 집합 유사 시퀀스 매칭에 적용하는 방법에 대해 설명한다. 제 4장에서는 제안한 방법에 대한 성능 평가 결과를 제시한다. 마지막으로 제 5장에서 결론을 맺는다.

2. 기본 지식(Preliminaries)

본 장에서는 제안하는 방법을 이해하기 위해서 필요한 기본 지식을 설명한다. 제 2.1절에서는 집합 간 유사도를 표현하는 방법들에 대해 설명하고, 제 2.2절에서는 유사 시퀀스 매칭에 대해서 설명한다. 제 2.3절에서는 본 논문에서 다루는 집합 기반 유사 시퀀스 매칭에 대해 설명한다.

2.1 집합 유사도

집합 간 유사도를 표현하는 방법은 자카드 계수(Jaccard coefficient)[6], 소렌센-다이스 계수(Sorensen-Dice coefficient)[7] 등이 있다. 자카드 계수는 집합(set) 간의 유사도를 표현하는 대표적인 방법으로, 공통되는 원소의 비율이 높을수록 유사하다고 판단한다. 그리고 유사한 정도를 0부터 1사이의 값으로 표현한다. 즉, 값이 0인 경우는 두 집합의 유사성이 전혀 없다는 것을 의미하고, 1인 경우는 두 집합이 완전히 같다는 의미이다. 집합 A, B에 대한 자카드 계수는 Equation (1)처럼 표현할 수 있다.

$$Jaccard(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

소렌센-다이스 계수는 집합 뿐만 아니라 중복 집합(bag) 간의 유사도를 표현하는 방법으로, 자카드 계수와 유사한 형태 및 의미를 지닌다. 본 논문에서는 집합 간 유사도를 표현하는 방법으로 자카드 계수를 사용한다.

2.2 유사 시퀀스 매칭[8, 9]

유사 시퀀스 매칭은 데이터 시퀀스들 중에서 사용자에 의해 주어지는 질의 시퀀스와 유사한 것을 찾는 문제이다. 유사 시퀀스 매칭은 전체 매칭과 서브시퀀스 매칭으로 구분된다[8]. 전체 매칭은 데이터 시퀀스와 질의 시퀀스의 길이가 동일한 경우, 질의 시퀀스와 유사한 데이터 시퀀스를 찾는 문제이다. 반면 서브시퀀스 매칭은 질의 시퀀스와 데이터 시퀀스의 길이가 다른 경우, 질의 시퀀스와 유사한 서브 시퀀스를 가지고 있는 데이터 시퀀스와 그 위치를 찾는 문제이다. 서브시퀀스 매칭은 길이에 대한 제약이 없이 수행되므로, 전체 매칭 문제를 일반화 했다고 할 수 있다. 따라서 본 논문에서는 보다 일반화된 문제인 서브시퀀스 매칭 문제를 다룬다.

유사한 시퀀스를 찾기 위해 두 시퀀스의 유사한 정도를 유클리디안 거리를 사용하여 정형적으로 표현할 경우, 길이 n인 두 시퀀스 X={X[1],X[2],…,X[n]}, Y={Y[1],Y[2],…Y[n]}의 유클리디안 거리는 다음과 같이 정의된다.

$$D(X,Y) = \sqrt{\sum_{i=1}^{n}(X[i]-Y[i])^2} \quad (2)$$

두 시퀀스 사이의 거리인 D(X,Y)가 사용자가 제시한 허용치 이하이면 두 시퀀스는 유사하다고 판단한다. 그리고 이때의 허용치가 ε일 경우, 두 시퀀스가 ε-매치(ε-match)한다고 한다. 따라서 유사 시퀀스 매칭은 질의 시퀀스와 ε-매치하는 데이터 시퀀스를 찾는 문제라고 할 수 있다. 이를 수식으로 표현하면 다음과 같다.

$$D(X,Y) \leq \varepsilon \quad (3)$$

이때 ε을 크게 한다는 것은 유클리디안 거리가 비교적 먼, 덜 유사한 시퀀스라도 유사하다고 판단하겠다는 것이다. 반대로 ε을 작게 한다는 것은 거리가 비교적 가까운 시퀀스들만 유사하다고 판단하겠다는 것이다.

기존의 서브 시퀀스 매칭 방법들은 시퀀스 전체 길이에 대한 유사도를 계산하여 유사한지 유사하지 않은 지 판단하므로, 많은 계산과 시간을 필요로 했다. 이를 보다 효율적으로 수행하기 위해 시퀀스를 일정한 길이 단위인 윈도우로 나누어, 윈도우 간의 유사도 계산으로 서브시퀀스 매칭을 수행하는 방법인 Dual-Match[9]가 제안되었다.

Dual-Match는 시퀀스를 일정한 길이 단위인 윈도우로 나누고, 각 윈도우들 간의 유사도 계산을 통해 특정 시퀀스와 유사한 시퀀스를 찾는 방법이다. Dual-Match 알고리즘은 i) 후보 매칭 단계와 ii) 후처리 단계로 이루어진다. 후보 매칭 단계는 윈도우 간의 유사도 비교를 통해 특정 시퀀스와 유사할 것이라고 예상되는 시퀀스들을 후보로 선정하는 단계이다. 그리고 후처리 단계는 후보로 선정된 시퀀스들에 대해 특정 시퀀스와의 전체길이 시퀀스 유사도 계산을 수행하여 실제로 유사한 데이터 시퀀스를 찾는 단계이다.

Dual-Match에서 사용한 개념인 윈도우는 디스조인트 윈도우와 슬라이딩 윈도우로 구분된다. 디스조인트 윈도우는 시퀀스의 시작 엔트리부터 크기가 ω인 윈도우를 구성하고, 윈도우 간에 겹치지 않도록 시퀀스를 윈도우로 나눈 것이다. 이때, 크기가 ω보다 작은 윈도우는 생성되지 않도록 한다. Dual-Match에서는 데이터 시퀀스를 디스조인트 윈도우로 구성하며, 이는 Fig. 3의 (a)로 표현된다. 그리고 슬라이딩 윈도우는 시퀀스의 시작 엔트리부터 크기가 ω인 윈도우를 구성하여 하나의 엔트리씩 슬라이딩 시키며 윈도우를 구성한 것이다. Dual-Match에서는 질의 시퀀스를 슬라이딩 윈도우로 구성하며, Fig. 3의 (b)와 같다.

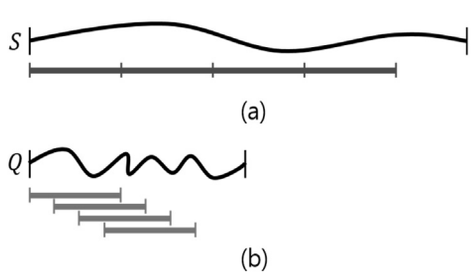


Fig. 3. Window Structure of Dual-Match

Dual-Match는 시퀀스의 일부인 윈도우 간의 유사도를 계산하는 방식으로 유사 시퀀스 매칭을 수행한다. 이에 Dual-Match

에서는 윈도우 간의 유사도 계산 만으로 유사한 시퀀스를 찾더라도 착오 기각이 발생하지 않음을 증명하였다. 자세한 내용은 [9]를 참고하도록 한다.

2.3 집합 기반 유사 시퀀스[2-4]

집합 기반 유사 시퀀스 매칭은 범주 데이터에 대해 유사 시퀀스 매칭을 수행할 시에 발생하는 시간 불일치 문제를 해결하기 위한 방안으로 제안되었다. 수치 데이터에 대한 유사 시퀀스 매칭의 경우, 2.2절에서 설명했듯 각 데이터 값 간의 거리로 유사한 정도를 계산할 수 있다. 그러나 범주형 데이터에 대한 유사 시퀀스 매칭은 각 범주간의 유사한 정도가 수치적으로 정해져 있지 않기 때문에 시퀀스 간 데이터 값들이 시간 순서에 따라 정확하게 일치해야만 유사하다고 판단한다. 이로 인해 몇몇 데이터의 순서만 바뀌었을 뿐인 유사한 데이터 시퀀스가 유사하지 않은 시퀀스로 판단되는 시간 불일치 문제가 발생한다. 이러한 경우에 집합 개념을 적용할 경우, 집합으로 묶인 일정 시간 간격 안에서는 데이터의 순서가 일치하지 않아도 유사하다 판단이 가능하므로 시간 불일치 문제를 해결할 수 있다.

본 논문에서는 집합 기반 유사 시퀀스 매칭 기법인 SSM (Set-based Subsequence Matching)[4]의 성능을 향상시키는 문제를 다루기 때문에, SSM에 대해서 좀 더 자세히 설명한다. SSM은 기존 유사 시퀀스 매칭 방법인 Dual-Match에서 발생하는 시간 불일치 문제를 해결하기 위해 기존의 시퀀스들을 집합 개념의 시퀀스들로 모델링한다. SSM에서 집합에 대한 정의는 다음과 같다.

정의 1 (집합)[2-4] 집합은 시퀀스에서 시간 순서대로 집합의 크기인 σ개의 원소들을 하나로 묶어서 구성한다. 시퀀스 P_i의 k번째 엔트리인 $P_i[k]$를 시작 엔트리로 하는 크기 σ인 집합은 $P_i[k]$부터 $P_i[k+\sigma-1]$까지 시간 순서대로 σ개의 원소들의 집합이며 $SP_{i,k}=\{P_i[k],\cdots,P_i[k+\sigma-1]\}$로 표현된다.

기본적으로 SSM은 시퀀스를 일정 크기의 집합으로 나누어 집합들의 시퀀스로 모델링한다. 정의 1에서 정의한 집합 개념을 이용하여 집합 시퀀스를 다음과 같이 정의한다.

정의 2 (집합 시퀀스)[2-4] 집합 시퀀스는 집합들을 시간 순서대로 정렬한 것이다. 시퀀스 P_i의 집합 시퀀스는 $SP_i=(sp_{i,1},sp_{i,(1+\sigma)},\cdots)$로 표현된다. 그리고 SP_i의 k번째 엔트리는 $SP_i[k]$로 표현된다.

그리고 고정된 개수의 집합들로 윈도우를 구성한다. 즉, 집합 시퀀스는 집합들을 시간 순서대로 나열한 시퀀스를 의미하고 윈도우는 집합의 개수가 ω가 되도록 구성한 서브시퀀스를 의미한다. 집합 시퀀스에서의 윈도우는 다음과 같이 정의된다.

정의 3 (집합 시퀀스의 윈도우)[2-4] 집합 시퀀스에서의 윈도우의 개념은 윈도우 크기인 ω개의 집합들로 구성된 서브 시퀀스이다. 집합 시퀀스 SP_i의 k번째 엔트리인 $SP_i[k]$를 시작 엔트리로 하는 크기 ω인 윈도우는 $SP_i[k]$부터 $SP_i[k+\omega-1]$까지 집합들의 시퀀스이며 $sp_i^k=(sp_{i,k},sp_{i,(k+\sigma)},\cdots,sp_{i,(k+\sigma\times(\omega-1))})$로 표현된다. 그리고 $sp_i^k[j]$는 k번째 윈도우의 j번째 집합을 의미한다.

SSM은 Dual-Match와 동일하게 질의 시퀀스를 슬라이딩 윈도우로 나누고, 데이터 시퀀스는 디스조인트 윈도우로 나누어 구성한다. 다만, SSM에서는 최근 시점을 더 중요하다고 생각하므로 최근 시점부터 윈도우가 구성된다는 차이점이 있다. Fig. 4는 SSM의 데이터 구조 및 윈도우 구성을 나타낸다. Fig. 4에서 각각의 원들은 시퀀스의 원소를 의미하며, 원소를 세 개씩 묶은 사각형은 집합을 의미한다. Fig. 4의 (a)는 σ가 3일때 ω가 2인 데이터 집합 시퀀스 S의 디스조인트 윈도우를 나타낸 것으로, 각각의 윈도우는 두 개의 집합으로 구성되어 있다. (b)는 σ가 3일 때 ω가 2인 질의 집합 시퀀스 Q의 슬라이딩 윈도우를 나타낸다.

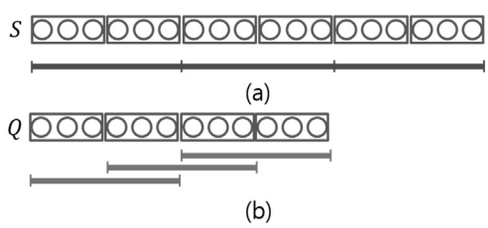


Fig. 4. Window Structure of SSM

S-Match 알고리즘은 Dual-Match와 동일하게 i) 후보 매칭 단계와 ii) 후처리 단계로 이루어진다. 후보 매칭 단계는 윈도우 간의 유사도 계산을 통해 후보 데이터 시퀀스를 선정하는 단계로, 윈도우 간의 유사도 비교를 통해 특정 시퀀스와 유사할 것이라고 예상되는 시퀀스들을 후보로 선정한다. 그리고 후처리 단계는 전 단계에서 얻은 후보 시퀀스에 대해 검증을 수행하는 단계로, 후보 시퀀스들에 대해 특정 시퀀스와의 전체길이 시퀀스 유사도 계산을 수행하여 실제로 유사한 데이터 시퀀스를 찾는다. 하지만 SSM은 집합의 개념을 이용하므로 두 집합 시퀀스 또는 두 윈도우가 유사한지 판단하기 위한 모델로 유클리디안 거리를 집합으로 확장한 유클리디안 집합 거리를 사용한다. 기존 시퀀스에서 유클리디안 거리가 두 값 사이의 차이의 합이라면, 집합 시퀀스에서 유클리디안 집합 거리는 두 집합 사이의 차이의 합을 의미한다. 유클리디안 집합 거리의 정의는 다음과 같다.

정의 4 (유클리디안 집합 거리)[3, 4] 유클리디안 집합 거리는 두 집합 시퀀스의 유사도를 측정하는 모델로, 집합 간의 차이의 합으로 표현된다. 두 집합 시퀀스 SP_i와 SP_j가 존재할 때, SP_i와 SP_j의 유클리디안 집합 거리는 다음과 같으며, 이때 $SP_i[k]$는 집합 시퀀스 SP_i의 k번째 집합을 의미한다.

$$D(SP_i,SP_j)\sum_{(k=1)}^{|SP|}(1-Jaccard(SP_i[k],SP_j[k])) \quad (4)$$

유사 시퀀스 매칭에서는 두 집합 시퀀스 사이의 거리가 허용치 ε 이하인 경우 두 시퀀스가 유사하다고 판단한다. 따라서 SSM에서도 두 집합 시퀀스의 유클리디안 집합 거리가 ε 이하일 경우 유사하다고 판단하고, 이를 ε-매치한다고 표현한다. 그리고 Dual-Match와 동일하게 집합 시퀀스를 구성하는 윈도우들 간의 유사도 계산을 통해 유사한 후보를 선정하므로 그 과정에서 착오기각이 발생하지 않는다는 것이 보장되어야 한다. 이를 위해 SSM에서는 정리1를 통해 착오기각이 발생하지 않음을 증명하였다.

정리 1 [3, 4] 데이터 집합 시퀀스 SP_m와 질의 집합 시퀀스 SP_a가 동일한 길이이며 이를 각각 ρ개의 디스조인트 윈도우 sp_m^i, $sp_a^i(0 \le i \le \rho)$로 나누었을 때, 두 시퀀스 SP_m과 SP_a가 ε-매치하면, 적어도 하나 이상의 (sp_m^i, sp_a^i)쌍이 ε/ρ-매치한다. 이때 ρ는 질의 집합 시퀀스가 가질 수 있는 최소한의 윈도우 개수를 의미하는 것으로 $\rho = [(Len(Q)+1)/\omega - 1] - 1$와 같이 계산된다. 즉, 다음 조건식이 성립한다.

$$D(SP_m, SP_a) \le \varepsilon \Rightarrow \vee_{(i=0)}^{(\rho-1)} D(sp_m^i, sp_a^i) \le \frac{\varepsilon}{\rho} \qquad (5)$$

결국 두 시퀀스 SP_m과 SP_a가 ε-매치하면, 적어도 하나 이상의 디스조인트 윈도우 $sp_m^i(0 \le i \le \rho)$와 sp_a^i가 ε/ρ-매치한다.

정리 1은 윈도우 간의 유사도 값인 유클리디안 집합 거리가 ε/ρ 이하인 해당 시퀀스를 후보로 삼으면 착오기각이 발생하지 않는다는 것을 의미한다. SSM의 더 자세한 착오기각에 대한 증명은 [3, 4]를 참고하도록 한다.

3. 집합 역인덱스 기반 검색 방법을 사용한 집합 기반 유사 시퀀스 매칭 방법

본 장에서는 기존 집합 유사 시퀀스 매칭의 성능 상의 문제를 해결하기 위해 집합 역인덱스 기반의 검색 방법을 사용하여 집합 유사 시퀀스 매칭을 빠르게 수행하는 방법에 대해 설명한다. 제 3.1절에서는 집합 역인덱스를 사용한 경우에 유사한 집합을 효율적으로 찾을 수 있도록 하는 교집합 크기 비교 문제에 대해 설명한다. 3.2절에서는 집합 역인덱스 구조와 교집합 크기를 효율적으로 계산 할 수 있는 방법에 대해 설명한다. 3.3절에서는 집합 역인덱스와 교집합 크기 비교 문제를 적용한 집합 유사 시퀀스 매칭 방법에 대해 설명한다.

3.1 교집합 크기 비교

본 절에서는 집합 역인덱스에서 검색의 결과로 선정될 집합을 결정하는데 사용되는 척도인 교집합 크기에 대해서 설명한다. 여러 개의 대상 집합들 중에서 특정 집합과 유사한 집합을 찾기 위해선, 모든 대상 집합들과 특정 집합 간의 집합 유사도 계산이 필요하다. 이때 집합 유사도는 제 2장의 Equation (1)에서 설명한 바와 같이 Jaccard 유사도를 사용한다. 보다 구체적으로는 정의4에서 사용된 것과 같이 거리 개념을 사용하여 Equation (6)과 같이 사용한다. 유사도는 값이 클수록 유사한 것을 나타내는 반면, 거리 개념은 가까울수록, 즉 값이 작을수록 더 유사하다고 판단한다.

$$Distance(A,B) = 1 - Jaccard(A,B) \qquad (6)$$

이렇게 거리 개념으로 변환된 집합 유사도를 사용하여 두 집합이 ε-매치 한다는 것으로 표현하면 Equation (7)과 같다.

$$Distance(A,B) \le \varepsilon \qquad (7)$$

그러나, Jaccard 계수를 구하기 위해서는 두 집합 간의 합집합 크기와 교집합 크기를 알아야 하기 때문에, Equation (7)의 조건이 만족되는지 판단하기 위해서는 매번 그 두 크기를 계산해야 하는 문제가 있다. 본 논문에서는 이러한 문제를 해결하기 위해서 Equation (7)에서 정의한 집합간 ε-매치 문제를 정리 2의 교집합 크기 비교 문제로 변환한다.

정리 2 두 집합 A,B가 다음의 조건을 만족하면, A와 B는 ε-매치한다.

$$|A \cap B| \ge \frac{(|A|+|B|)(1-\varepsilon)}{(2-\varepsilon)} \qquad (8)$$

증명: 부록A 참고

결과적으로 교집합의 크기가 특정 임계값을 넘는지만 확인하면 ε-매치 여부를 판단할 수 있는 것이다. 이때 A,B가 ε-매치할 수 있는 교집합 크기 기준값, 즉 Equation (8)의 우변값을 교집합 임계값이라 한다. 앞으로는 이와 같이 교집합 크기를 구하여 임계값과 비교하는 방식으로 집합 유사 시퀀스 매칭을 수행한다.

3.2 집합 역인덱스 구조 및 인덱스 기반 검색 방법

제 3.1절에서는 집합 간 유사도 계산 문제를 집합 간 교집합 크기 비교 문제로 변환하는 방법을 다루었다. 본 절에서는 이러한 변환에 입각하여, 집합 유사 시퀀스 매칭을 보다 빠르게 수행 할 수 있도록 교집합 크기가 임계값 이상의 유사한 집합을 빠르게 찾는 인덱스 기반 검색 방법을 설명한다. 인덱스 기반 검색 방법을 설명하기에 앞서 집합 유사 시퀀스 매칭의 데이터 구조 와 본 논문에서 제안하는 집합 역인덱스 구조에 대해 설명한다.

1) 집합 역인덱스 구조

집합 역인덱스는 집합에 대한 역인덱스를 기반으로 구성된다. 역인덱스는 텍스트 문서에서 많이 사용되는 구조로,

특정 단어가 어떤 문서에 위치하고 있는 지 나타내는 매핑 정보를 저장하여 검색을 빠르게 수행할 수 있도록 하는 인덱스 구조이다. 집합에 대해서 역인덱스를 구성할 때는 특정 원소가 어떤 집합에 들어있는지 나타내는 매핑 정보를 저장한다. 다시 말하면, 각 원소에 대해서 그 원소를 포함하는 집합들의 리스트를 유지하는 형태로 이루어지고 이를 사용하여 특정 원소가 어떤 집합에 포함되어 있는지 빠르게 확인할 수 있다. 집합 역인덱스 구조를 개념적으로 표현하면 Fig. 5와 같다. 먼저 전체 원소들의 집합 $U=\{e_1,\cdots e_M\}$이 존재할 때, 원소 e_1,e_2,e_3로 이루어진 집합 s_i(초록색 영역으로 표현)는 U의 부분 집합이다. 그리고 집합 역인덱스 엔트리 IE_M은 e_M이라는 원소를 포함하는 집합들 s_i의 식별자 i를 오름차순으로 정리한 리스트이다. 그리고 집합 역인덱스 IV_U는 전체 원소들에 대한 모든 집합 역인덱스 엔트리 $IE_j(e_j \in U)$의 집합이다.

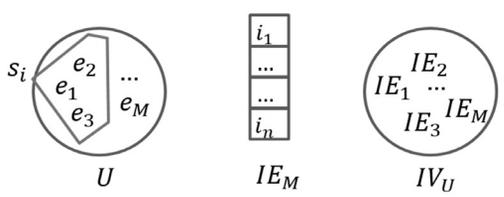


Fig. 5. Set inverted index structure

다음의 예 1은 영화 제목을 원소로 하는 집합 시퀀스에 대한 집합 역인덱스의 예이다.

예 1 Fig. 6은 집합 시퀀스에 대한 집합 역인덱스를 구성하는 예를 보여준다. 전체 영화 이름의 집합 U에 대해서 집합 $s_1=\{Ted, Avatar, Sing\}$, 집합 $s_2=\{Sherlock, Avatar, Sing\}$, 집합 $s_3=\{Sing, Alive, Avatar\}$, 집합 $s_4=\{Avatar, Ted, Alive\}$로 구성되어 있을 때, 이에 대한 집합 역인덱스 IV_U는 $IE_{Alive}, IE_{Avatar}, IE_{Sherlock}, IE_{Sing}, IE_{Ted}$로 구성된다. 이때, IE_{Alive}는 Alive를 포함하고 있는 집합 식별자의 오름차순 리스트인 3,4로 구성된다.

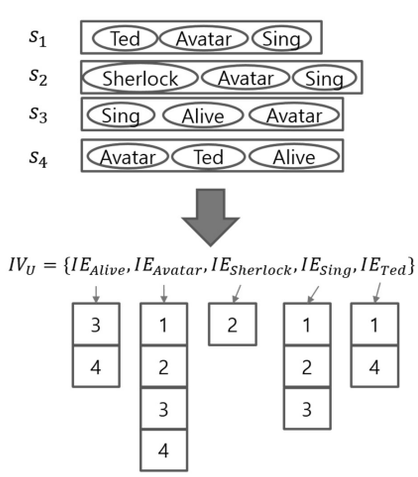


Fig. 6. Example of Set Inverted Index Structure

본 논문에서는 이러한 집합 역인덱스를 사용하여 유사한 집합을 찾기 위해 모든 집합들 간의 교집합 크기를 계산할 필요 없이, 특정 원소가 포함된 집합들에 대해서만 교집합 크기를 계산함으로써 보다 효율적으로 유사한 집합을 찾는다. 다음으로 집합 역인덱스를 사용하여 교집합 크기를 계산하는 방법에 대해 설명한다.

2) 집합 역인덱스를 사용한 교집합 크기 계산

주어진 모든 집합들에 대해서 집합 역인덱스 IV_U가 만들어져 있을 때, 또 다른 집합 s_q와의 교집합 크기를 구하는 방법에 대해서 설명한다. 우선 s_q에 포함된 각각의 원소들에 대한 집합 역인덱스 엔트리들을 읽어온다. 다음으로, 집합 식별자에 대해서 오름차순으로 정렬되어 있는 각 집합 역인덱스 엔트리들을 앞에서부터 순차적으로 읽으면서 특정 집합 s_i가 전체 엔트리에서 몇 번 나오는지 빈도수를 계산한다. 이때의 빈도수는 각 집합 s_i에 집합 s_q에 속한 원소들이 몇 개나 들어있는지를 뜻하는 것으로, 결국 s_i와 s_q의 교집합 원소 개수가 된다. 이러한 교집합 크기 계산 방법을 알고리즘으로 정형적으로 표현하면 다음과 같다.

알고리즘 교집합 계산
입력 : (1) 데이터 집합 s_i들로부터 구축된 집합 역인덱스 IV_U
(2) 질의 집합 s_q
출력 : s_q와 s_i들의 교집합 크기 저장 테이블 *Result*
알고리즘 :
1. s_q에 포함된 원소들 e_j에 대한 집합 역인덱스 엔트리 $IE_j(e_j \in s_q)$들을 가져옴
2. 읽어온 IE_j들에서 첫번째 원소들을 읽음
3. ***FOR EACH*** IE_j들의 모든 원소들을 읽을 때까지
4. IE_j들에서 읽어온 원소들 중에서 가장 작은 값인 *CurrentMin*을 찾음
5. IE_j들에서 읽어온 원소들 중에서 *CurrentMin*과 동일한 값을 가지는 원소의 개수인 *Count*를 계산
6. 계산한 *Count* 값이 s_q와 $s_{CurrentMin}$의 교집합 크기이므로, *Count* 계산 결과와 *CurrentMin*을 *Result*에 저장
7. *CurrentMin*을 가지고 있는 IE_j들에 대해서만 다음 원소를 읽고, 나머지 IE_j들은 기존에 읽은 원소를 유지함
8. ***END EACH***

Fig. 7. Calculate Intersection Algorithm

먼저 질의 집합에 포함된 원소들에 대한 집합 역인덱스 엔트리들을 모두 읽어온다(step 1). 다음으로 읽어온 집합 역인덱스 엔트리들의 첫번째 원소들을 읽는다(step 2). 그리고 집합 역인덱스 엔트리들의 모든 원소를 읽을 때까지 교집합 크기 계산 과정을 반복 수행한다(step 3~8). 교집합 크기 계산 과정은 집합 역인덱스 엔트리에서 읽어온 원소들 중에서 가장 작은 값인 *CurrentMin*을 찾고(step 4), 찾은 *CurrentMin*과 동일한 값을 가지는 원소가 몇 개인지 나타내는 Count를 계산한다(step 5). 이때 계산한 *Count*값은 질의 집합과 *CurrentMin*을 식별자로 갖는 집합이 공통적으로 가지는 원소의 개수이므로, 두 집합의 교집합 크기라고 할 수

있다. 그러므로 *Count*값과 *CurrentMin*을 교집합 크기 저장 테이블 *Result*에 기록한다(step 6). 마지막으로 *CurrentMin*을 가지고 있던 집합 역인덱스 엔트리에 대해서만 다음 원소를 읽어온다(step 6). 이와 같이 *CurrentMin*을 가지고 있던 집합 역인덱스 엔트리에 대해서만 다음 원소를 읽어오는 이유는 모든 집합 역인덱스 엔트리를 한번의 검색으로 모든 원소들을 확인하기 위함이다. 다음의 예2는 특정 집합과 유사한 집합을 찾고자 하는 환경에서의 교집합 크기 계산 방법에 대한 예이다.

예2 집합 s_q에대한 교집합 크기 계산 방법은 다음과 같다. 먼저 s_q={*Ted,Avatar,Alive*}로 구성되어 있을 때, s_q에 포함되어 있는 원소들 *Ted,Avatar,Alive*에 대한 집합 역인덱스 엔트리 $IV_{Ted}, IV_{Avatar}, IV_{Alive}$를 읽어온다. 다음으로 집합 역인덱스 엔트리 각각의 첫번째 원소들인 1,1,3을 읽는다. 그리고 읽어온 원소들 1,1,3 중에서 가장 작은 값인 *CurrentMin*을 계산하고, 이 시점에서의 *CurrentMin*인 1을 가지는 원소의 개수인 *Count*를 계산한다. 이때 *CurrentMin*인 1의 *Count*는 2이므로, s_q와 s_1의 교집합 크기는 2가 된다. 그리고 *CurrentMin*인 1을 가지고 있던 IV_{Ted}와 IV_{Avatar}에서 다음 원소를 읽어와 집합 역인덱스 엔트리의 원소들 4,2,3에 대해 *CurrentMin*과 *Count* 계산 과정을 수행한다. 이때 *CurrentMin*은 2이고 *Count* 1이 되어서 s_q와 s_2의 교집합 크기는 1이 된다. 이 과정을 반복하면 s_q와 s_1,s_2,s_3,s_4 간의 교집합 크기 계산이 완료된다.

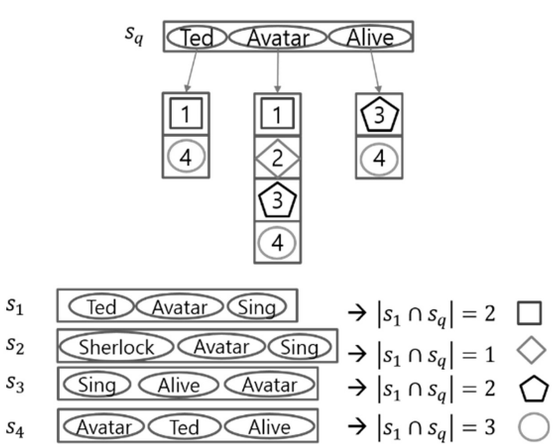


Fig. 8. Example of Calculate Intersection

3.3 집합 역인덱스 및 교집합 크기 비교 문제를 적용한 집합 유사 시퀀스 매칭 방법

본 절에서는 3.1절, 3.2절에서 설명한 집합 역인덱스를 사용한 교집합 크기 계산 방법을 집합 유사 시퀀스 매칭에 적용한다. 본 논문에서 제안하는 방법은 SSM에서 ε/ρ-매치하는 윈도우를 찾는 과정에 집합 역인덱스를 사용한 검색 방법을 적용함으로써 집합 유사 시퀀스 매칭을 빠르게 수행한다. 기존의 SSM에서는 ε/ρ-매치하는 윈도우를 찾기 위해서 윈도우를 구성하는 집합 시퀀스들 간의 유사도를 일일이 계산해야 해서 속도가 느리다. 그러나, 제안하는 방법은 윈도우를 구성하는 집합들 간의 교집합 크기가 특정 값 이상인 집합들을 집합 역인덱스를 사용하여 검색하고, 이러한 조건을 만족하는 집합을 포함한 윈도우에 대해서만 유사도를 계산함으로써 빠른 계산이 가능하게 한다.

제안하는 인덱스 기반 집합 유사 시퀀스 매칭의 전체 과정을 정리하면 다음과 같다. 1) 집합 역인덱스를 사용하여 질의 집합 시퀀스의 한 윈도우와 ε/ρ-매치할 가능성이 있는 데이터 집합 시퀀스의 윈도우에 포함된 집합(이를 후보 집합(candidate sets)이라 부른다)을 모두 찾는다. 2) 후보 집합이 포함된 데이터 집합 시퀀스의 윈도우와 질의 집합 시퀀스의 윈도우 간의 유사도 계산을 통해 실제로 ε/ρ-매치하는 윈도우인지 검증하고, 실제로 ε/ρ-매치하는 데이터 집합 시퀀스의 윈도우(이를 후보 윈도우(candidate windows)라 부른다)를 모두 찾는다. 3) 후보 윈도우가 포함된 데이터 집합 시퀀스와 질의 집합 시퀀스의 시퀀스 전체 유사도 계산을 통해 해당 데이터 집합 시퀀스가 ε-매치하는 집합 시퀀스인지 검증한다.

제안한 방법의 첫 번째 부분은 후보 집합을 찾는 과정에서 질의 시퀀스의 집합과 교집합의 크기가 특정 값 이상인 데이터 시퀀스의 집합들을 모두 찾는데 이때 사용하는 교집합 크기 값(이후 교집합 임계값이라 부른다)을 어떻게 정하는지에 대해서 설명한다. 이러한 교집합 임계값을 정할 때 중요한 것은 착오기각이 발생하지 않도록 해야 한다는 것이다. 즉, 교집합 임계값 이상의 집합들을 선택했을 때, ε/ρ-매치하는 모든 집합 윈도우를 찾을 수 있어야 한다. 다음의 정리3은 착오 기각이 발생하지 않도록 하기 위해서 윈도우와 집합 간에 어떤 조건을 만족해야 하는지 그 관계를 밝힌 것이다.

정리 3 두 집합 시퀀스 SP_m,SP_a의 i번째 디스조인트 윈도우 sp_m^i,sp_a^i가 ω 개의 집합으로 이루어져 있을 때, 두 디스조인트 윈도우가 ε/ρ-매치하면 적어도 하나의 집합 쌍 $sp_m^i[j]$,$sp_a^i[j]$이 $\varepsilon/(\rho \cdot \omega)$-매치한다. 즉 다음의 조건식이 성립한다.

$$D(sp_m^i, sp_a^i) \le \frac{\varepsilon}{\rho} \Rightarrow \vee_{(k=0)}^{(\omega-1)} D(sp_m^i[j], sp_a^i[j]) \le \frac{\varepsilon}{\rho\omega} \quad (9)$$

결국 두 윈도우를 구성하는 집합의 쌍 중 하나라도 $\varepsilon/(\rho \cdot \omega)$-매치하면, 두 윈도우는 ε/ρ-매치할 가능성이 있다.

증명: 부록B 참고

정리 3에 의해 집합 간 유사도 비교를 통해 후보 윈도우를 선정하더라도 착오기각이 발생하지 않는다는 것이 증명된다. 다음은 교집합 크기 비교를 통해 후보 윈도우를 선정

하더라도 착오 기각이 발생하지 않도록 하는 교집합 임계값을 구하기 위해 정리 3을 교집합 비교 문제로 변환한다.

정리 4 윈도우를 구성하는 집합 쌍 중 하나라도 다음의 식을 만족하면 두 집합 쌍은 $\varepsilon/(\rho \cdot \omega)$-매치한다.

$$|sp_m^i[j] \cap sp_a^i[j]|$$

$$\geq \frac{(|sp_m^i[j]|+|sp_a^i[j]|)(1-\frac{\varepsilon}{\rho \cdot \omega})}{2-\frac{\varepsilon}{\rho \cdot \omega}} = \frac{2 \cdot \sigma \cdot (1-\frac{\varepsilon}{\rho \cdot \omega})}{2-\frac{\varepsilon}{\rho \cdot \omega}} \quad (10)$$

증명: 부록C 참고

정리 3과 정리 4에 따라 집합 역인덱스를 사용하여 질의 집합 시퀀스와 데이터 집합 시퀀스 사이에 교집합의 크기가 교집합 임계값, 즉 $(|sp_{i,k}|+|sp_{j,k}|)(1-\frac{\varepsilon}{\rho \cdot \omega})/(2-\frac{\varepsilon}{\rho \cdot \omega})$ 이상인 모든 집합을 찾으면 ε/ρ 매치할 가능성이 있는 모든 윈도우를 빠르게 찾을 수 있다. 다음은 이러한 방법을 적용한 집합 유사 시퀀스 매칭 방법인 집합 역인덱스 기반 집합 유사 시퀀스 매칭의 수행 과정에 대해 설명한다.

집합 역인덱스 기반 집합 유사 시퀀스 매칭의 수행 과정은 a) 교집합 비교 단계와 b) 후보 매칭 단계, c) 후처리 단계로 구성된다. 교집합 비교 단계에서는 질의 시퀀스들의 모든 집합들과 데이터 시퀀스들의 집합들 간의 교집합 크기 비교를 통해 유사한 윈도우를 후보로 선정한다. 후보 매칭 단계에서는 선정한 후보 윈도우에 대한 검증을 수행하여 윈도우가 실제로 유사한 경우, 이 윈도우가 포함된 시퀀스를 후보로 선정한다. 마지막으로 후처리 단계에서는 후보 시퀀스가 실제로 유사한 지 검증을 수행한다.

a) 단계 1: 교집합 비교 단계

교집합 비교 알고리즘은 데이터 집합 시퀀스들 로부터 구축된 집합 역인덱스와 질의 집합 시퀀스 및 유사도 허용치를 입력 받아 유사할 것으로 생각되는 후보 집합을 선정한다. 먼저 유사도 허용치를 정리 4를 이용하여 교집합 임계값으로 변환한다. 그리고 질의 집합 시퀀스에 포함된 각각의 집합에 대해 3.2절에서 설명한 집합 역인덱스를 이용한 교집합 계산 알고리즘을 이용하여 교집합 크기를 계산한다. 마지막으로 계산한 집합 간 교집합 크기가 교집합 임계값 이상일 경우, 임계값 이상의 교집합 크기를 가지는 집합들과 윈도우에서 해당 집합의 위치를 기록한다. 교집합 비교 알고리즘은 다음의 Fig. 9와 같다.

본격적인 교집합 비교 알고리즘을 수행하기 앞서 유사도 허용치 ε을 정리 4를 이용하여 교집합 임계값 *Thresh*로 변환함으로써, 집합 간 유사도 비교 문제를 교집합 비교 문제로 변환한다(step 1). 그리고 교집합 비교 알고리즘은 입력 받은 질의 집합 시퀀스 SP_a에 포함된 모든 집합들에 대해 수행한다(step 2~12). 먼저 SP_a에 포함된 하나의 집합 $sp_{a,k}$

알고리즘 교집합 비교
입력 : (1) 데이터 집합 시퀀스 SP_i들로부터 구축된 집합 역인덱스 IV_U
(2) 질의 집합 시퀀스 SP_a와 허용치 ε
출력 : 후보 집합 저장 테이블 candiSet
알고리즘 :
1. 허용치 ε과 집합 크기 σ, 윈도우 크기 ω, 최소 포함 윈도우 개수 ρ를 이용하여 교집합 임계값 Thresh 계산
2. ***FOR EACH*** 질의 집합 시퀀스 SP_a의 모든 집합들에 대해 반복 수행
3. 집합 $sp_{a,k}$를 구성하는 원소들 $e_j(e_j \in sp_{a,k})$에 대한 집합 역인덱스 엔트리 IE_j를 모두 읽어옴
4. 읽어온 IE_j들에서 첫번째 원소들을 읽음
5. ***FOR EACH*** IE_j의 모든 원소를 읽을 때까지
6. IE_j들에서 읽어온 원소들 중에서 가장 작은 값인 CurrentMin(i,k)을 찾음
7. IE_j들에서 읽어온 원소들 중에서 CurrentMin과 동일한 값을 가지는 원소의 개수인 Count를 계산
8. ***IF*** (Count > Thresh)
9. CurrentMin과 SP_i의 디스조인트 윈도우에서 $sp_{CurrentMin(i,k)}$의 위치 및 $sp_{a,k}$의 식별자를 candiSet에 저장
10. CurrentMin을 가지고 있는 IE_j들에 대해서만 다음 원소를 읽고, 나머지 IE_j들은 기존에 읽은 원소를 유지함
11. ***END EACH***
12. ***END EACH***

Fig. 9. Comparison Intersection Algorithm

와 유사한 데이터 집합 시퀀스들의 집합을 찾기 위해 $sp_{a,k}$를 구성하는 원소들에 대한 집합 역인덱스 엔트리 IE_j를 모두 읽어온다(step 3). 그리고 읽어온 IE_j들의 각각의 첫번째 원소들을 읽어온다(step 4). 다음으로 읽어온 IE_j들의 원소를 모두 읽을 때까지 교집합 크기 계산을 반복한다(step 5~11). 교집합 크기 계산을 하기 위해 읽어온 원소들 중에서 가장 작은 값인 *CurrentMin*(*i,k*)를 찾는다(step 6). 이때 i는 데이터 집합 시퀀스의 식별자이고, *k*는 데이터 집합 시퀀스의 집합 식별자이다. 그리고 IE_j에서 읽어온 원소들 중에서 *CurrentMin*(*i,k*)와 동일한 값을 가지는 원소의 개수인 *Count*를 계산한다(step 7). 이때 계산한 *Count*는 $sp_{a,k}$와 $sp_{CurrentMin(i,k)}$가 가지고 있는 동일한 원소의 개수를 나타내므로, 교집합 크기라고 할 수 있다. 다음으로 *Count*값과 *Thresh*값을 비교했을 때 *Count*값이 더 크다는 것은 두 집합이 유사하다는 것을 의미하므로, *CurrentMin*과 SP_i의 디스조인트 윈도우에서 $sp_{CurrentMin(i,k)}$의 위치 및 교집합 크기를 계산한 $sp_{a,k}$의 식별자를 후보 집합 테이블인 *candiSet*에 저장한다(step 8~9). 마지막으로 *CurrentMin*(*i,k*)를 가지고 있던 IE_j들에 대해서만 다음 원소를 읽고, 나머지 IE_j들은 기존에 읽은 원소를 유지하여 한번의 검색 과정을 통해 IE_j의 모든 원소를 확인할 수 있도록 한다(step 10).

b) 단계2: 후보 매칭 단계

후보 매칭 알고리즘은 단계1에서 선정한 후보 집합에 대한 검증을 수행하여 실제로 질의 집합 시퀀스의 윈도우와 데이터 집합 시퀀스의 윈도우가 실제로 ε/ρ 매치하는지 검증하는 알고리즘이다. 이 알고리즘은 단계1에서 계산된 후보 집합과 질의 집합 시퀀스 및 허용치, 데이터 집합 시퀀

스를 입력 받아 실제로 ε/ρ 매치하는 데이터 시퀀스의 윈도우를 출력한다. 이를 위해 후보 집합이 포함된 데이터 집합 시퀀스의 디스조인트 윈도우와 후보 집합과 유사하다고 판단됐던 질의 시퀀스 집합이 포함된 슬라이딩 윈도우 간의 유사도 계산을 수행한다. 이때, 두 집합이 각 윈도우에서 차지하는 위치가 동일해야 하는데, 그 이유는 이 경우에만 두 집합 간의 유사도가 윈도우 간 유사도 계산에 반영되기 때문이다. 그리고 윈도우 간 유사도 계산 결과가 ε/ρ 이하인 경우, 이 윈도우가 포함된 데이터 집합 시퀀스와 질의 집합 시퀀스는 실제로 ε-매치 할 가능성이 있다. 따라서 해당 윈도우 및 윈도우 간 매칭 시점을 저장하여 이를 후처리 과정에 사용한다. 후보 매칭 알고리즘은 다음과 같다.

```
알고리즘 후보 매칭
입력 : (1) 후보 집합 저장 테이블 candiSet
       (2) 데이터 집합 시퀀스 SP_i
       (3) 질의 집합 시퀀스 SP_a와 허용치 ε
출력 : 후보 윈도우 저장 테이블 candiWin
알고리즘 :
1.  FOR EACH 질의 집합 시퀀스 SP_a의 슬라이딩 윈도우들에
    대해 반복 수행
2.      FOR EACH SP_a의 모든 집합 sp_a,k들에 대해 반복 수행
3.          FOR EACH candiSet에 포함된 모든 집합 sp_i,k들에
                대해 반복 수행
4.              IF(SP_i의 디스조인트 윈도우에서 sp_i,k의 위치
                    와 SP_a의 슬라이딩 윈도우에서 sp_a,k의
                    위치가 동일한 경우)
5.                  sp_i,k와 sp_a,k가 포함된 윈도우들에 대해
                    윈도우 간 유클리디안 집합 거리인
                    winDist를 계산
6.                  IF(winDist ≤ ε/ρ)
7.                      sp_i,k를 포함하는 윈도우 및 sp_a,k를
                        포함하는 윈도우와의 매칭 시점을
                        candiWin에 저장
8.              END IF
9.          END EACH
10.     END EACH
11. END EACH
```

Fig. 10. Candidate Matching Algorithm

후보 매칭 알고리즘은 질의 집합 시퀀스의 모든 슬라이딩 윈도우에 대해 수행된다(step 1~11). 또한 슬라이딩 윈도우에 포함된 모든 집합들에 대해 순차적으로 수행된다(step 2~10). 그리고 교집합 비교 알고리즘의 결과인 *candiSet*에 포함된 모든 집합 $sp_{i,k}$들이 포함된 윈도우들에 대해서 검증 작업을 수행한다(step 3~9). 윈도우가 실제로 ε/ρ 매치하는지 검증하는 작업은 *candiSet*에 포함된 집합인 $sp_{i,k}$와 질의 집합 시퀀스에 포함된 집합인 $sp_{a,k}$가 각 윈도우에서 차지하는 위치가 동일한 경우에 대해서만 수행된다(step 4~8). 이 조건을 만족할 경우, $sp_{i,k}$가 포함된 윈도우와 $sp_{a,k}$가 포함된 윈도우 간 유클리디안 집합 거리인 *winDist*를 계산한다(step 5). 이때, *winDist*가 ε/ρ 이하인 경우, 해당 윈도우 및 매칭 시점을 *candiWin*에 저장한다.

c) 단계3: 후처리 단계

마지막 단계인 후처리 알고리즘은 단계2에서 계산된 후보 윈도우와 질의 시퀀스 및 데이터 시퀀스들을 입력 받아 질의 시퀀스와 ε-매치하는 데이터 시퀀스들만 출력한다. 이 과정은 후보 매칭 알고리즘의 결과인 ε-매치 할 것으로 생각되는 데이터 집합 시퀀스의 후보 윈도우들 및 매칭 시점들 중에서 실제로 ε-매치하지 않는 후보들을 제거한다. 이를 위해 후보 윈도우들이 포함된 데이터 집합 시퀀스와 질의 집합 시퀀스를 매칭 시점을 이용하여 시퀀스 간 유사도 계산을 수행한다. 그리고 시퀀스 간 유사도 계산 결과가 ε 이하일 경우, 해당 데이터 집합 시퀀스를 출력한다. 후처리 알고리즘은 다음과 같다.

```
알고리즘 후처리
입력 : (1) 후보 윈도우 저장 테이블 candiWin
       (2) 데이터 집합 시퀀스 SP_i
       (3) 질의 집합 시퀀스 SP_a와 허용치 ε
출력 : SP_a와 ε-매치하는 SP_i
알고리즘 :
1.  FOR EACH candiWin 에 포함된 모든 후보 윈도우들 및
    매칭 시점에 대해 반복 수행
2.      후보 윈도우가 포함된 SP_i와 SP_a를 매칭 시점에
        맞추어 시퀀스 간 유클리디안 집합 거리인 seqDist를
        계산
3.          IF(seqDist ≤ ε)
4.              조건을 만족하는 SP_i 및 매칭 시점을 출력함
5.  END EACH
```

Fig. 11. Post-Processing Algorithm

후보 매칭 알고리즘은 후보 윈도우 저장 테이블 *candiWin*의 모든 후보 윈도우들 및 매칭 시점에 대해 수행된다(step 1~5). 먼저 *candiWin*의 후보 윈도우가 포함된 데이터 집합 시퀀스 SP_i와 질의 집합 시퀀스 SP_a를 매칭 시점에 맞추어 시퀀스 간 유클리디안 집합 거리인 *seqDist*를 계산한다(step 2). 그리고 *seqDist*가 ε 이하일 경우 해당 SP_i는 SP_a와 ε-매치한다는 것을 의미하므로, SP_i와 두 시퀀스의 매칭 시점을 출력한다(step 3~4).

본 논문에서 제안한 집합 역인덱스 기반 집합 유사 시퀀스 매칭은 유사한 집합을 효율적으로 찾음으로써 기존 방법보다 데이터 집합 시퀀스에 대한 교집합 크기 계산 수행 횟수를 줄일 수 있다. 기존 방법은 ε/ρ-매치 하는 윈도우를 찾기 위해 모든 데이터 집합 시퀀스의 모든 디스조인트 윈도우와 질의 집합 시퀀스의 모든 슬라이딩 윈도우 간의 유사도 계산을 수행해야 했다. 하지만 본 논문에서는 이 과정에 집합 역인덱스 기반의 검색 방법을 적용하여 교집합 임계값 이상의 집합이 포함된 윈도우에 대해서만 유사도 계산을 수행하도록 하여, 기존 방법에서 불필요하게 수행해야만 했던 윈도우 간 유사도 계산 횟수를 줄인다.

4. 실 험

본 장에서는 제안하는 방법들을 적용한 집합 유사 시퀀스에 대한 성능 평가 결과를 제시한다. 제 4.1절에서는 실험 데이터에 대해 설명하고, 4.2절에서는 실험 환경에 대해서 설명한다. 마지막으로 4.3절에서는 실험 결과를 설명한다.

4.1 실험 데이터

본 논문에서는 실험 데이터로 합성 데이터를 사용한다. 이렇게 합성 데이터를 사용하여 실험을 수행한 이유는 기존 연구[4]에서 사용한 데이터와 실제로 구할 수 있는 데이터들이 시퀀스 개수에 제한이 있고, 시퀀스 개수에 비해 데이터의 종류가 상당히 많아 특정 질의 시퀀스와 유사한 정답 시퀀스들이 극히 드물기 때문이다. 따라서 본 논문에서는 대용량의 데이터 처리 성능 비교와 유사 시퀀스 비율 등에 따른 성능 비교를 위해서 합성 데이터를 사용한다. 데이터 생성에 관련된 표기에 대한 정의 및 의미는 Table 1과 같다. 먼저 n은 데이터 시퀀스의 개수로, 몇 개의 데이터 시퀀스를 생성할지에 대한 기준이 된다. $avgLen$은 데이터 시퀀스의 평균 길이를 나타내며, 데이터 시퀀스들이 길이는 $avgLen$에서 일정 수준이상 벗어나지 않도록 생성된다. 그리고 $qlen$은 질의 시퀀스의 길이로, $avgLen$의 비율로 표현이 된다. N과 U는 관계가 밀접하다고 할 수 있는데, N은 생성할 전체 원소(element)들의 개수로 n과 $avgLen$의 곱으로 표현된다. 그리고 U는 데이터의 희귀 비율로서, 몇가지의 데이터를 생성할 지에 대한 척도가 된다. 예를 들어 N이 30,000개이고 U가 0.1이라면 3,000가지의 원소를 이용하여 데이터 시퀀스를 생성하겠다는 것을 의미한다. 그리고 C는 특정 질의 시퀀스와 유사한 데이터 시퀀스를 얼마나 생성할 지에 대한 파라메터로 n과 β의 곱으로 표현된다. 이때 β는 정답 데이터 시퀀스 집합 비율에 대한 파라메터로, 0부터 1 사이의 실수 값을 가진다. 마지막으로 δ와 D는 데이터를 변이하기 위한 파라메터로 얼마만큼 변이할 지를 나타내는 척도이다.

Table 1. Summary of Notation

Symbols	Definitions	Default
n	데이터 시퀀스의 개수	2,500/5,000/ **10,000**/20,000
$avgLen$	데이터 시퀀스의 평균 길이	**1,000**
$qlen$	질의 시퀀스의 길이 qlen=avgLen × α	**200** (α=0.2)
U	데이터의 희귀(unique) 비율 U= k/N(k:#of kind of data)	0.003 (k=7,500/15,000/ 30,000/60,000)
C	정답(유사) 데이터 시퀀스 집합 개수 C=n× β	250/500/**1000**/2000 (β=0.1)
δ	변이율 x≤δ≤y 사이의 실수(0≤x,y≤1)	u[0.1, 0.5]
D	데이터 변이 개수 D=qlen × δ	u[20, 100]

D개 만큼의 데이터를 임의로 변이 시키는 이유는 임계값에 따라 선택율(selectivity)이 다르게 나오게끔 생성하기 위함이다. 그리고 임의의 위치에 삽입하는 이유는 유사 시퀀스 매칭 수행 시에 동일한 위치에 있는 질의 시퀀스와 유사한 서브시퀀스만을 찾는 것이 아니라, 질의 시퀀스와 유사한 서브시퀀스가 어디에 있던 정답으로 찾을 수 있음을 증명하기 위함이다. 위와 같은 방법으로 총 C개 만큼의 정답 집합을 생성하고, 기준이 되는 데이터 시퀀스를 새로 생성하여 위 과정을 반복 수행한다.

본 논문에서는 실험을 위해 데이터 시퀀스의 개수가 2,500개부터 20,000개까지 2배씩 증가시키며 총 4개의 데이터셋을 생성하였다. 이때 데이터 시퀀스의 길이는 평균적으로 1,000이 되게끔 생성하였고, 질의 시퀀스는 200의 길이를 가지게끔 생성하였다. 또한 정답 집합의 개수는 250개부터 2,000개까지 2배씩 증가하게 생성하였고, 데이터 시퀀스의 임의의 위치에 삽입되는 데이터는 질의 시퀀스 길이 기준 10%에서 50% 사이의 임의의 개수만큼 변이되도록 설정하였다.

4.2 실험 환경

실험은 제안하는 방법의 우수성을 검증하기 위해 기존 집합 유사 시퀀스 방법인 SSM과 SSM에 제안하는 방법을 적용한 방법 $Indexed$에 대해 동일한 데이터, 동일한 사양의 기기를 이용하여 두 방법에 대한 성능 비교를 공정하게 수행하였다. 실험을 위해 이용한 기기의 사양은Intel® Core™ i7-4790k, 8GB의 램을 장착한 Windows 10 PC이며 C언어 기반으로 작성된 코드를 이용하여 실험을 수행하였다.

Table 2. Summary of Experimental Parameters

Symbols	Default
σ	**10**
ω	**4**
ε	10(ɣ=0.5), **8**(ɣ=0.6), 6(ɣ=0.7), 4(ɣ=0.8)

Table 2는 실험에 사용된 매개 변수를 정리한 것이다. 실험에서 집합의 크기 σ는 10으로 하였고, 윈도우 크기 ω는 4로 하였다. 그리고 허용치 ε은 Equation (11)에 의해 기준 유사도 ɣ가 0.5에서 0.8까지 0.1씩 증가시키는 것과 대응하여 10에서 4까지 2씩 감소시키며 실험을 수행하였다. 이때 기준 유사도 ɣ는 유사하다고 판단하기 위한 유사도의 기준을 말하며, 만약 ɣ가 0.6일 경우 두 집합 시퀀스가 전체 시퀀스 중에서 60% 이상 같아야 유사하다고 판단한다.

$$\varepsilon = \frac{|qlen|}{\sigma} \times (1-\gamma) \tag{11}$$

Equation (11)은 두 집합 시퀀스를 유사하다고 판단하기 위한 허용치 ε을 집합 시퀀스 환경에 맞도록 변환한 것이다. 이때, $\frac{|qlen|}{\sigma}$은 질의 시퀀스를 구성하는 집합의 개수를 의미하고, $(1-\gamma)$은 두 집합 간의 유사도 허용치를 의미한다. 두 집합 시퀀스 간의 유사도를 나타내기 위해 사용하는 유클리디안 집합 거리는 정의 4와 같이 집합 시퀀스를 구성하는 집합들의 유사도의 합으로 표현된다. 따라서 집합 시퀀

스 간 유사한지 유사하지 않는지 판단하기 위한 허용치 ε은 Equation (11)과 같이 집합 시퀀스를 구성하는 집합의 개수에 집합 간 유사도 허용치를 곱한 값으로 표현할 수 있다.

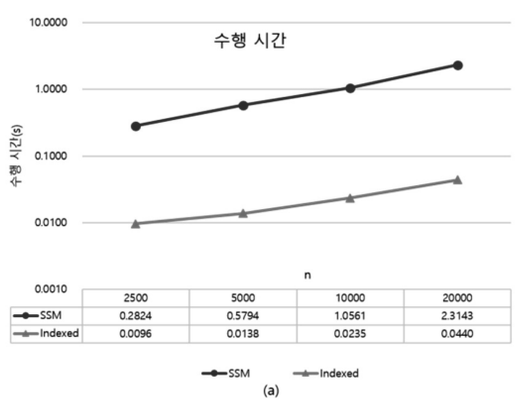
4.3 실험 결과

실험은 제안하는 방법의 성능상 우수함을 보이기 위해서 집합 기반 유사 시퀀스 매칭의 수행 시간을 측정하였다. 먼저 데이터 시퀀스의 개수 n을 증가시키며 수행 시간 및 결과를 확인함으로써 제안하는 알고리즘이 대량의 데이터 시퀀스에 대해서 빠른 처리가 가능함을 보였다. 다음으로 허용치 ε을 변화시켜가며 수행 시간 및 결과를 확인함으로써 결과 크기 증가, 즉 후보 증가로 인한 연산량 증가에 따라 제안하는 알고리즘이 어떤 성능상의 이점을 가지는지 보였다. 이와는 별도로 n의 증가에 따른 인덱스 구축 시간을 확인함으로써 인덱스를 사용하는데 필요한 사전 비용이 어느 정도 인지를 확인하였다.

1) n을 변화시키며 수행 시간을 측정

첫 번째 실험은 ε을 8, *qlen*은 200, σ는 10, ω는 4로 고정한 상태에서 n을 2,500부터 20,000까지 2배씩 증가 시키며 수행 시간을 확인하였다. 이 실험은 제안하는 방법의 확장성(scalability)을 보여주고, 집합 인덱스를 사용함으로써 검색 공간(search space)을 줄여 속도가 향상될 수 있음을 증명하기 위한 실험이다.

실험 결과는 Fig. 12와 같다. Fig. 12의(a)는 두 방법에 대해 수행시간을 측정한 결과로, 제안하는 방법들이 성능 향상의 효과가 있음을 보여주고 있다. 이는 집합 인덱스를 사용함으로써 검색 공간(search space)을 줄여 속도 향상이 있었음을 증명해주는 결과이다. 그리고 기존 방법의 경우 n이 증가함에 따라 수행 시간 또한 약 2배씩 비례하여 증가하는 것을 확인할 수 있는 반면, 제안하는 방법의 경우 약 1.5배씩 증가하여 기존 방법보다 완만하게 증가하는 것을 확인



n	2500	5000	10000	20000
SSM	0.2824	0.5794	1.0561	2.3143
Indexed	0.0096	0.0138	0.0235	0.0440

(a)

n	2500	5000	10000	20000
Indexed/SSM	0.034	0.024	0.022	0.019

(b)

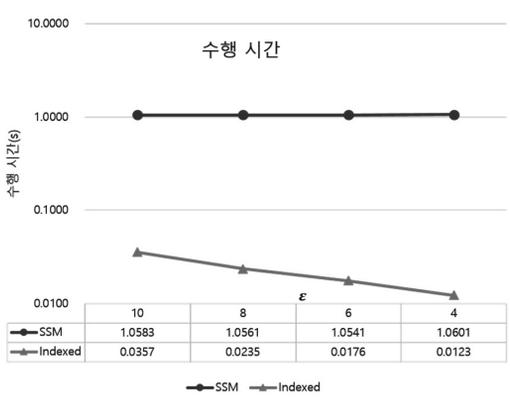
Fig. 12. Experimental Results
(changing number of data sequence)

할 수 있다. 이러한 현상은 n이 증가 할수록 제안하는 방법의 성능 향상 정도가 증가한다는 것을 의미하므로 제안하는 방법의 확장성이 기존 방법에 비해 높다는 것을 나타낸다. Fig. 12의 (b)는 SSM에 비해 제안하는 방법의 성능이 얼마나 향상되었는지 확인하기 위해 비율로 나타낸 결과로, 제안하는 방법이 최소 30배에서 최대 50배 정도의 성능 향상을 보인다는 것을 확인할 수 있다.

2) ε을 변화시키며 수행 시간을 측정

두 번째 실험은 n을 10,000으로, *qlen*은 200, σ는 10, ω는 4로 고정 시킨 후, ε를 변화시키며 실험한 결과에 대해 설명한다. 이 실험은 ε에 따라 후보 집합 및 실제로 유사하다고 판단된 집합 시퀀스들의 개수가 변하는 점이 실제 성능에 미치는 영향을 분석하기 위해 수행한다.

유사도 허용치 ε의 변화에 따른 수행 시간 측정 결과는 Fig. 13과 같다. Fig. 13의 (a)는 두 방법에 대해 수행 시간을 측정한 결과이다. 첫 번째 실험과 동일하게 제안하는 방법들이 성능 향상의 효과가 있음을 보여주고 있다. 그리고 제안하는 방법의 경우 ε이 감소함에 따라 수행시간이 확연하게 줄어드는 것을 확인할 수 있다. 이러한 현상은 ε이 감소함에 따라 윈도우 간 유사도 계산을 수행할 윈도우들을 교집합 비교 단계에서 확연하게 줄이기 때문에 발생한다. Fig. 13의 (b)는 제안하는 방법의 성능이 얼마나 향상되었는지 확인하기 위해 비율로 나타낸 결과이다. ε이 감소 할 수록, 제안하는 방법의 성능이 최소 30배에서 최대 90배까지 증가하는 것을 확인할 수 있다. 즉, ε이 작아질수록 제안하는 방법이 보다 효율적이라는 것을 나타낸다.



ε	10	8	6	4
SSM	1.0583	1.0561	1.0541	1.0601
Indexed	0.0357	0.0235	0.0176	0.0123

(a)

ε	10	8	6	4
Indexed/SSM	0.033	0.022	0.016	0.011

(b)

Fig. 13. Experimental Results
(changing similarity rate)

3) n의 변화에 따른 인덱스 구축 시간 비교

본 논문에서 제안하는 방법은 기존의 *SSM*에 비해서 역 인덱스를 미리 구축해 두어야 한다는 차이가 있다. 이러한 인덱스 구축은 사전에 단 한번 수행하는 작업이지만, 필요

한 시간이 어느 정도인지 확인하는 것은 제안하는 방법의 현실 적용성을 살펴본다는 측면에서 중요하다. 인덱스 구축 시간은 n에 직접적인 영향을 받으므로 n을 2,500에서 20,000까지 2배씩 증가시키며 확인한다. Fig. 14에서 n이 20,000일 때 621초(약 10.4분) 정도가 소요되어 초기에 한 번 수행하는 작업임을 감안하였을 때 현실적인 시간 내에 완료할 수 있을 알 수 있다. 또한 n이 증가함에 따라 인덱스 구축 시간 또한 비례하여 증가하는 것을 확인 가능하다. 각각에 대한 데이터를 살펴보면, n이 두배로 증가할 때 인덱스 구축 시간은 약 4배정도 증가하는 것을 확인할 수 있는데 이유는 다음과 같다. 집합 역인덱스는 데이터 상에 존재하는 모든 데이터의 종류마다 생성이 된다. 즉 n 뿐만 아니라 데이터의 모든 종류 개수 k와도 연관이 있다. k는 데이터 생성 시에 사용되는 파라메터로, 표1의 데이터 희귀 비율 U에 의해 결정된다. 데이터 희귀 비율 U를 고정시켜 데이터를 생성하는 경우, n이 2배씩 증가하면 k 또한 2배씩 증가한다. 집합 역인덱스는 n개 만큼의 데이터에 대해 k개 만큼의 역인덱스 엔트리가 생성되므로 n과 k가 각각 2배씩 증가한다면 인덱스 구축 시간은 총 $n \cdot k$ 배인 4배만큼의 시간이 증가하게 된다.

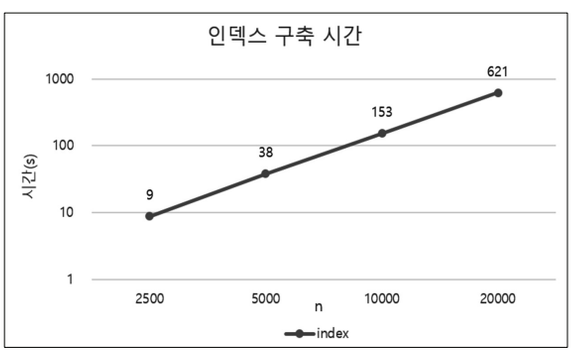


Fig. 14. Result of Index Build Time

5. 결 론

본 논문에서는 데이터스트림 환경에서의 집합 유사 시퀀스 매칭의 성능 향상을 위한 인덱스 기반 검색 방법을 제안하였다. 집합 유사 시퀀스 매칭 방법은 데이터 각각에 대해서 유사도를 계산하는 대신에 여러 개의 데이터를 집합으로 묶어서 집합 단위로 유사도를 계산한다는 특징을 가진다. 이 때, 집합 간의 유사한 정도를 나타내는 척도로 교집합을 기반으로 한 유사도를 사용한다. 그러나 기존의 방법은 교집합을 구하는 과정에 시간이 오래 걸릴 뿐만 아니라, 유사한 집합을 찾기 위해서는 수 많은 집합 간 교집합 크기를 구하는 과정을 수행해야 하므로 수행 시간이 많이 걸리는 성능상의 문제가 있다.

본 논문에서는 이러한 성능상의 문제를 해결하기 위해서 인덱스 기반의 검색 방법을 사용하여 집합 유사 시퀀스 매칭을 빠르게 수행하는 방법을 제안하였다. 제안하는 방법은 크게 두 가지로 구분된다. 첫 번째로 집합 시퀀스 유사도 문제를 교집합의 크기 비교 문제로 정형적으로 변환하고, 교집합의 크기를 빠르게 찾을 수 있는 인덱스 구조를 제안하였다. 이 방법은 집합 시퀀스 간의 유사한지 유사하지 않은 지 판단하기 위해서 시퀀스 내의 모든 집합을 비교할 필요 없이 특정 정도 이상의 유사도를 가지는 집합들 만을 선별적으로 찾음으로써 결과적으로 유사 시퀀스를 매우 빠르게 찾을 수 있는 기반을 제공한다. 두 번째로 제안한 인덱스 구조를 사용하여 집합 기반 유사 시퀀스 매칭을 효율적으로 수행할 수 있는 방법을 제안하였다. 기존의 윈도우를 사용한 집합 유사 시퀀스 매칭 방법에 인덱스 구조를 적용하고, 이때 착오기각이 발생하지 않음을 증명하였다.

실험 결과, 제안하는 인덱스 기반 집합 유사 시퀀스 매칭 방법은 기존의 방법에 비해서 최대 30배에서 50배의 수행시간 단축이 있음을 보였다. 또한 데이터 시퀀스의 개수가 증가할 수록 수행 시간의 차이가 점점 커지므로, 대용량 데이터 처리에 적절함을 보였다.

References

[1] Babcock, Brian et al., “Models and issues in data stream systems,” *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Databased Systems*, ACM, 2002.

[2] Eunji, Yeo, Juwon, Lee, and Hyo-Sang, Lim, “A Similar Data Stream Matching Method by Using the Concept of Item-Set Time Series,” *Korea Computer Congress*, pp.237-239, 2016.

[3] Eunji, Yeo, “A Data Stream Similar Sequence Matching technique Using the Concept of Item Set and Hierarchy,” M. Sc. thesis, Yonsei University, 2016.

[4] Eunji,Yeo, Juwon, Lee, and Hyo-Sang, Lim, “Set-based Subsequence Matching,” *KIISE SIGDB*, Vol.32, No.3, pp.152-169, 2016.

[5] Juwon, Lee, Daewon, Kim, and Hyo-Sang Lim, “An index Technique for Efficiently Measuring Set Similarities,” *KIISE Winter Conference*, pp.214-216, 2016.

[6] Jaccard, Paul, “Etude comparative de la distribution florale dans une portion des Alpes et du Jura. Impr. Corbaz, 1901.

[7] Sørensen, Thorvald, “A method of establishing groups of equal amplitude in plant sociology Indexedd on similarity of species and its application to analyses of the vegetation on Danish commons,” *Biol. Skr.*, 5, pp.1-34, 1948.

[8] Faloutsos, Christos, Mudumbai Ranganathan, and Yannis Manolopoulos, “Fast subsequence matching in time-series database,” ACM, Vol.23, No.2, 1994.

[9] Yang-Sae, Moon, Kyu-Young Whang, and Woong-Kee Loh, “Efficient time-series subsequence matching using duality in constructing windows,” *Information Systems*, Vol.26, No.4, pp.279-293, 2001.

부록 A: 정리2의 증명

두 집합 A, B가 ε-매치한다는 것을 수식으로 표현하면 다음과 같다.

$$Distance(A,B) = 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \leq \varepsilon$$

이를 교집합 크기 기준으로 다시 정리하기 위해 좌변을 정리한다.

$$\frac{|A| + |B| - 2|A \cap B|}{|A| + |B| - |A \cap B|} \leq \varepsilon$$

위 수식에서 좌변의 분모를 우변으로 넘겨 정리한다.

$$|A| + |B| - 2|A \cap B| \leq \varepsilon\,(|A| + |B| - |A \cap B|)$$

교집합 크기를 포함하고 있는 변수를 좌변으로 이동 시키고, 이를 교집합 크기 기준으로 정렬하면 다음과 같다.

$$|A \cap B|(2-\varepsilon) \geq (|A| + |B|)(1-\varepsilon)$$

$$|A \cap B| \geq \frac{(|A| + |B|)(1-\varepsilon)}{(2-\varepsilon)}$$

부록 B: 정리3의 증명

모순 증명을 이용하여 증명한다. 만약 집합 시퀀스의 i번째 윈도우를 구성하는 모든 집합 쌍들($sp_m^i[j], sp_a^i[j]$)의 거리가 $\frac{\varepsilon}{\rho\omega}$을 초과하는 값을 가진다면, 집합 쌍들이 포함된 윈도우 쌍(sp_m^i, sp_a^i)은 $\frac{\varepsilon}{\rho\omega}$을 초과하는 값을 가진다. 즉, 다음의 수식들이 성립한다.

먼저 윈도우 쌍의 거리는 윈도우를 구성하는 집합 쌍들의 거리 합이므로 다음과 같이 표현된다.

$$D(sp_m^i, sp_a^i) = \sum_{j=i\times\omega+1}^{(i+1)\times\omega} (D(sp_m^i[j], sp_a^i[j])$$

이때 모든 집합 쌍들의 거리 $D(sp_m^i[j], sp_a^i[j])$가 $\frac{\varepsilon}{\rho\omega}$을 초과한다면, ω개의 집합으로 이루어진 윈도우 간의 거리 $D(sp_m^i, sp_a^i)$는 반드시 $\frac{\varepsilon}{\rho}$을 초과한다. 즉, 다음의 식이 성립한다.

$$D(sp_m^i, sp_a^i) = \sum_{j=i\times\omega+1}^{(i+1)\times\omega} (D(sp_m^i[j], sp_a^i[j]) > \frac{\varepsilon}{\rho}$$

따라서, 두 디스조인트 윈도우 sp_m^i, sp_a^i가 ε/ρ-매치하면 적어도 하나의 집합 쌍 $sp_m^i[j], sp_a^i[j]$이 $\varepsilon/((\rho \cdot \omega))$-매치해야만 한다.

부록 C: 정리4의 증명

집합 쌍($sp_m^i[j], sp_a^i[j]$)의 거리가 $\varepsilon/((\rho \cdot \omega))$-매치 한다는 수식을 교집합 기준으로 정리함으로써 증명한다. 먼저 집합 쌍($sp_m^i[j], sp_a^i[j]$)의 거리가 $\varepsilon/((\rho \cdot \omega))$-매치 한다는 것을 수식으로 표현하면 다음과 같다.

$$D(sp_m^i[j], sp_a^i[j])$$

$$= 1 - \frac{|sp_m^i[j] \cap sp_a^i[j]|}{|sp_m^i[j]| + |sp_a^i[j]| - |sp_m^i[j] \cap sp_a^i[j]|} \leq \frac{\varepsilon}{\rho \bullet \omega}$$

좌변을 분모로 묶으면 다음과 같다.

$$\frac{|sp_m^i[j]| + |sp_a^i[j]| - 2|sp_m^i[j] \cap sp_a^i[j]|}{|sp_m^i[j]| + |sp_a^i[j]| - |sp_m^i[j] \cap sp_a^i[j]|} \leq \frac{\varepsilon}{\rho \bullet \omega}$$

그리고 좌변의 분모를 우변으로 넘겨 정리하면 다음과 같다.

$$|sp_m^i[j]| + |sp_a^i[j]| - 2|sp_m^i[j] \cap sp_a^i[j]|$$

$$\leq \frac{\varepsilon}{\rho \bullet \omega}(|sp_m^i[j]| + |sp_a^i[j]| - |sp_m^i[j] \cap sp_a^i[j]|$$

마지막으로 우변에서 교집합을 포함하고 있는 변수들을 좌변으로 넘기고, 이를 교집합 기준으로 정리하면 다음과 같다. 그리고 집합들의 크기는 모두 동일하게 σ이므로 σ로 치환하여 나타낸다.

$$|sp_m^i[j] \cap sp_a^i[j]|(2 - \frac{\varepsilon}{\rho \bullet \omega})$$

$$\geq (|sp_m^i[j]| + |sp_a^i[j]|)(1 - \frac{\varepsilon}{\rho \bullet \omega})$$

$$|sp_m^i[j] \cap sp_a^i[j]|$$

$$\geq \frac{(|sp_m^i[j]| + |sp_a^i[j]|)(1 - \frac{\varepsilon}{\rho \bullet \omega})}{2 - \frac{\varepsilon}{\rho \bullet \omega}}$$

$$= \frac{2 \bullet \sigma \bullet (1 - \frac{\varepsilon}{\rho \bullet \omega})}{2 - \frac{\varepsilon}{\rho \bullet \omega}}$$

이 주 원

e-mail : juwonlee@yonsei.ac.kr
2015년 연세대학교
컴퓨터정보통신공학부(학사)
2015년～현 재 연세대학교 전산학과
석사과정
관심분야 : 데이터베이스, 정보보안,
GPU를 사용한 데이터 처리

임 효 상

e-mail : hyosang@yonsei.ac.kr
1998년 연세대학교 컴퓨터과학과(학사)
1999년 한국과학기술원 전산학과(석사)
2007년 한국과학기술원 전산학과(박사)
2011년 Purdue Univ.Computer Science
Department(박사후과정)
2011년～현 재 연세대학교 컴퓨터정보통신공학부 교수
관심분야 : 모바일 데이터베이스, 데이터스트림, 정보보안,
데이터 신뢰도 분석, GPU를 사용한 데이터 처리