생물학 서열 데이타베이스에서 부분 문자열의 선택도 추정

(Estimation of Substring Selectivity in Biological Sequence Database)

배 진 욱 † 이 석 호 ††
(Jinuk Bae) (Sukho Lee)

요 약 지금까지 문자열 데이타에 대한 선택도 추정은 문자열들의 등장 회수에 대한 정보를 저장하고 있는 '카운트 서픽스 트리'를 생성한 뒤, 이 트리를 이용하여 부분 문자열들의 선택도를 추정하는 방법으로 이루어졌다. 그런데, 문자열 데이타가 생물학 서열처럼 매우 길어질 경우 카운트 서픽스 트리를 생성하는 일은 거의 불가능해진다는 문제점이 발생한다. 이 논문에서는 길이가 q인 부분 문자열들만을 삽입한 '카운트 큐그램 트리'를 제안한다. 카운트 큐그램 트리는 서열 내의 길이가 q 이하인 모든 부분 문자열(큐그램)들의 정확한 등장 회수를 저장하고 있으며, 문자열의 전체 길이 N에 상관없는 크기로, O(N) 시간에 생성 가능하다. 또한, 이 논문에서는 카운트 큐그램 트리를 이용한 'k번째 최대겹침' 추정 방법을 제시한다. 이 추정 방법은 질의 문자열을 길이 q인 부분 문자열로 나눌 때 부분 문자열들의 겹치는 정도 k를 선택할 수 있도록 한 방법으로 이전 연구에서 제시한 '최대겹침' 방법을 확장하였다. q와 k를 변화시키며 진행한 실험을 통해 대부분의 경우에 매우 정확하게 선택도를 추정할 수 있음을 확인하였다.

키워드 : 생물학 서열 데이타베이스, DNA 서열, 부분 문자열 선택도 추정, 카운트 서픽스 트리, 카운트 큐그램 트리, k번째 최대겹침 추정 방법

Abstract Until now, substring selectivities have been estimated by two steps. First step is to build up a count-suffix tree, which has statistical information about substrings, and second step is to estimate substring selectivity using it. However, it's actually impossible to build up a count-suffix tree from biological sequences because their lengths are too long. So, this paper proposes a novel data structure, count q-gram tree, consisting of fixed length substrings. The Count q-gram tree retains the exact counts of all substrings whose lengths are equal to or less than q and this tree is generated in O(N) time and in size not subject to total length of all sequences, N. This paper also presents an estimation technique, k-MO. k-MO can choose overlapping length of splitted substrings from a query string, and this choice will affect accuracy of selectivity and query processing time. Experiments show k-MO can estimate very accurately.

Key words : Biological Sequence Database, DNA Sequence, Estimation of Substring Selectivity, Count Suffix Tree, Count Q-gram Tree, k-MO

1. 서 론

질의 최적화 단계에서는 WHERE절에 주어진 조건에 따라 결과가 될 투플들의 선택도(selectivity)를 추정한 뒤, 그 값에 따라 질의 처리 과정을 달리한다. 이 때, WHERE절에 주어진 조건이 숫자 애트리뷰트에 대한 범위 질의일 경우, 샘플링[1]이나 히스토그램[2] 등을 이용하여 선택도를 계산하는 문제가 오랫동안 연구되어왔다. 반면에 그림 1의 질의처럼 와일드문자가 포함된 문자열이 조건으로 주어졌을 때 투플들의 선택도를 추정하는 문제는 [3]에서 처음으로 제기되었다. 이 문제에 대해 [4]에서는 보다 정확한 추정 방법을 제시하였고, 다차원 애트리뷰

· 본 연구는 두뇌한국21 사업에 의하여 지원받았음
† 학생회원 : 서울대학교 전기·컴퓨터공학부
oblody@db.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
shlee@cse.snu.ac.kr
논문접수 : 2001년 12월 28일
심사완료 : 2003년 1월 30일

트들에 대해 선택도를 추정[5,6,7]하거나, 가지 모양의 질의에 대해 선택도를 추정하는 문제[8]로 확장 연구되었다. 최근에는 XML 문서에서 경로 문자열들에 대한 선택도를 추정하는 문제[9] 또한 대두되었다.

```
select *
from part
where color like '*green*';
```

그림 1 와일드문자가 사용된 질의

[3,4]에서 사용한 방법은 문자열 데이타에 대한 정보를 저장하는 단계와 이 정보를 이용하여 선택도를 추정하는 단계로 구성된다. 문자열 데이타에 포함된 서픽스(suffix)들의 발생 회수를 서픽스 트리[10]의 변형인 카운트 서픽스 트리(Count Suffix Tree)에 저장한다. 이 때, 히스토그램의 크기에 제한이 있는 것과 마찬가지로 이 트리의 크기도 메모리 내에 들어올 수 있는 작은 크기이어야 한다. 선택도를 추정할 때는 문자열 패턴을 부분 문자열들로 쪼갠 뒤 각 부분 문자열의 카운트 값을 조합하여 추정한다. 그런데, [3,4]에서 사용한 카운트 서픽스 트리는 염색체 서열과 같이 매우 긴 문자열들에 대해서는 적용하기가 어렵다. 염색체 서열들의 서픽스들을 트리에 삽입하기도 어렵거니와 설사 삽입했다고 할지라도 공간상의 제약으로 나중에 가지치기될 가능성이 높기 때문이다. 그러므로, 생성도중의 트리 크기가 대단히 크고, 생성 시간이 대단히 오래 걸린다는 문제점이 생긴다.

이 논문에서는 매우 긴 문자열들로 이루어진 데이타베이스가 주어졌을 때 질의 문자열의 선택도를 추정하는 문제를 해결하기 위해, 긴 서픽스들을 트리에 삽입하는 대신 일정한 길이의 부분 문자열(q-gram)들만을 슬라이딩 윈도우 방식으로 트리에 삽입하는 카운트 큐그램 트리를 제안한다. 큐그램이란 길이가 q인 문자열을 말하는데, 문자열 패턴 매칭 분야[11,12,13]에서 다양하게 사용되어왔다. 임의의 길이의 질의로부터 길이가 q인 부분 문자열들을 추출하여 주어진 질의와 유사한 문자열들을 찾아내는 방법으로 사용되거나, 최근에는 유사 문자열 조인 문제[14]에서도 사용되었다.

카운트 큐그램 트리(Count Q-gram Tree)는 어차피 가지치기될 긴 서픽스의 뒷부분을 처음부터 삽입하지 않는다는 의미를 담고 있다. 즉, 문자열 "AAACTTA"에 대해 카운트 서픽스 트리는 "AAACTTA", "AACTTA", ..., "A"를 삽입하지만, q가 4인 카운트 큐그램 트리는 'AAAC", "AACT", "ACTT" 등을 삽입한다. 이를 통해 가지치기 과정없이 빠른 트리 생성이 가능하다. 선택도 추정 단계에서는 양극단의 [3]과 [4]의 추정 방법을 포괄하는 k번째 최대겹침 방법을 제시한다. 이 추정 방법은 카운트 큐그램 트리의 완전성(길이가 q 이하인 모든 부분 문자열들에 대해 정확한 카운트 값을 가지고 있음)에 기반한 방법이다.

이 논문의 구성은 다음과 같다. 2절에서 관련 연구들을 자세히 살펴보고, 3절에서 이 논문에서 다루고자 하는 문제를 설명한다. 4절과 5절에서는 3절에서 정의된 문제를 해결하기 위한 새로운 자료구조인 카운트 큐그램 트리에 대한 설명과, 이 트리를 이용한 추정 방법에 대해 살펴본다. 그리고 6절에서 실험 평가를 하고 7절에서 결론을 짓는다.

2. 관련 연구

문자열 애트리뷰트에 대한 선택도를 추정하는 문제는 [3]에서 처음 제기되었고, [4]에서 발전되었다. [3]과 [4] 모두 두 단계로 문제를 해결하고 있는데, 첫 번째는 문자열들을 읽어들이며 카운트 서픽스 트리를 생성하는 단계이다. 이 트리는 부분 문자열들이 나타나는 회수가 기록되지만, 공간적 제약으로 인해 임계값 p 미만의 노드들은 가지치기된다. 두 번째 단계는 질의 처리시 카운트 서픽스 트리를 이용하여 질의 문자열을 포함한 투플들의 수를 추정하는 단계이다. 만약, 카운트 서픽스 트리가 질의 문자열을 포함한다면 정확한 선택도를 알 수 있지만, 그렇지 않다면 트리에 포함된 정보만을 가지고 추정을 해야 한다. 2.1에서는 카운트 서픽스 트리에 대해서 설명하고, 2.2에서는 이를 이용한 추정 방법에 대해서 설명한다.

2.1 카운트 서픽스 트리

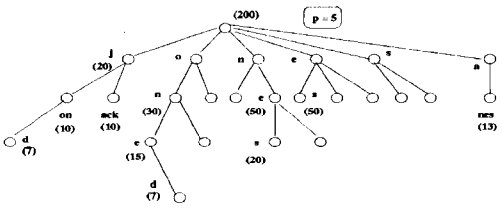


그림 2 5 이하의 노드가 가지치기된 o-Suffix 트리

[3]에서는 전통적인 데이타베이스의 선택도를 추정하는 문제 그대로, 전체 투플들 중에서 'like "*pattern*"'을 만족하는 투플들의 비율을 계산하는 문제를 다루었는데, 카운트 서픽스 트리를 자료구조로 사용하였다. 이

트리는 서픽스 트리[10]와 두 가지 면에서 다르다. 서픽스 트리는 각 단말 노드에서 해당 문자열이 나타나는 지점을 가리키는 포인터를 가지고 있지만, 카운트 서픽스 트리는 이 포인터들을 가지고 있지 않다. 그리고 카운트 서픽스 트리는 각 노드(중간 노드와 단말 노드 모두)에 문자열이 나타나는 회수를 기록하는 카운트 항목을 추가하였다. 그림 2에서 제일 왼쪽 경로는 "j"는 20번 나타났고, "jon"은 10번 나타났음을 보여준다. [4]는 이를 확장시켜 다음과 같이 두 가지 형태의 문제를 해결하는 트리[1]를 정의하였다.

- p-Suffix 트리 : 데이타베이스의 문자열들 중에서 부분 문자열 α를 포함하는 문자열들의 수를 저장하는 카운트 서픽스 트리
- o-Suffix 트리 : 데이타베이스의 문자열들 중에서 부분 문자열 α가 나타나는 회수를 저장하는 카운트 서픽스 트리

이 두 트리는 카운트 값을 처리하는 부분에서만 달라진다. 즉, 한 문자열에서 이미 등장한 부분 문자열이 다시 등장할 경우 p-Suffix 트리는 이 부분 문자열을 무시하지만, o-Suffix 트리는 카운트 값을 1 증가시킨다.

2.2 선택도 추정

[3]은 카운트 서픽스 트리에 나타나지 않는 패턴에 대한 선택도 추정 기법으로 독립기반 전략, 자식기반 전략, 깊이기반 전략을 제시하였다. [4]는 독립기반 전략을 KVI라고 이름붙인 뒤, 자신들이 제안하는 최대겹침(MO; Maximal Overlap), 제약조건 최대겹침(MOC), 제약조건 격자 최대겹침(MOLC) 방법과 비교하였다. 독립기반 전략이란 '패턴의 부분 문자열들이 등장하는 확률은 서로 독립이다'라는 가정 아래, 패턴을 카운트 서픽스 트리에 나타나는 문자열로 나눈 후 각각의 확률을 곱하는 방법이다. 예를 들면 "jones"의 선택도를 추정하기 위해, 그림 2의 카운트 서픽스 트리에 등장하는 "jon"과 "es"로 나누고, 각각의 확률의 곱을 "jones"의 선택도로 추정한다. 이 때, Cjon은 카운트 서픽스 트리에 저장된 "jon"의 카운트 값, N은 모든 문자열 길이의 합이다.

$$\begin{aligned} Pr(jones) &= Pr(jon)\times Pr(es|jon) \\ &\approx Pr(jon)\times Pr(es) \\ &= (C_{jon}/N)\times(C_{es}/N) \end{aligned}$$

반면에 [4]의 추정 방법들은 최대겹침 추정 방법에 기반하고 있는데, 패턴의 부분 문자열들을 서로 겹치지 않게 나눈 [3]과 달리, 최대한 겹치도록 나눈다는 점이 특징이다. "jones"는 "jon"과 "one"와 "nes"로 나누어지고 각각의 조건부 확률을 계산하여 보다 정확한 추정을 하게 된다. 이 때, Pr(*e*|*jon*)은 "jon" 다음에 "e"가 나올 확률이다.

$$\begin{aligned} Pr(jones) &= Pr(jon)\times Pr(e|jon)\times Pr(s|jone) \\ &\approx Pr(jon)\times Pr(e|on)\times Pr(s|ne) \\ &= (C_{jon}/N)\times(C_{one}/C_{on})\times(C_{nes}/C_{ne}) \end{aligned}$$

또, [4]에서는 가지치기로 노드가 제거된 후 남아있는 조건들을 최대한 활용하여 선택도가 만족해야 하는 제약 조건을 정의하였다. 예를 들면, "jes"를 추정하기 위해서는 KVI와 최대겹침 추정 방법 모두 "j"와 "es"로 나누고 선택도를 계산한다. 그러나, 그림 2의 트리를 살펴보면 가지치기되기 전이라고 하더라도 결코 "jes"는 나타날 수 없다. 왜냐하면, "j"의 발생 빈도는 20인데, 그 자식들인 "on"과 "ack"가 각각 10으로 이 두 값을 더하면 부모 "j"의 발생 빈도 20이 나온다. 즉, "j"의 자식들 중에는 결코 가지치기된 노드가 없다는 뜻이다. 그러므로, 이 경우 "jes"의 선택도는 0이 된다. 이와 같이 부모 노드의 값과 자식 노드들의 값을 분석하여 질의 문자열이 가질 수 있는 최대값들을 계산하는 제약조건들을 도입한 것이 제약조건 최대겹침 추정 방법과 제약조건 격자 최대겹침 추정 방법이다.

3. 생물학 서열에 대한 질의 패턴의 선택도 추정

3절에서는 [3,4]에서 다루었던 문제들을, DNA 서열처럼 길이가 매운 긴 문자열 데이타베이스가 주어졌을 때 질의 패턴의 선택도를 추정하는 문제로 확장시킨다.

3.1 생물학 서열 데이타베이스

생물학 서열 데이타베이스 BSD는 $\{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_n\}$이고, α_k는 생물학 서열을 나타낸다. $|\alpha_k|$는 α_k의 문자열 길이로 정의하고, BSD에 있는 모든 문자열들의 길이의 합을 N이라 한다. 서열 α에 대해 $\alpha[i:j]$(단, $i \le j$)는 i번째 글자부터 j번째 글자로 이루어진 부분 문자열을 뜻한다. 각 서열에 나타날 수 있는 글자들의 집합을 도메인이라 하고, 도메인의 크기를 D라 한다. 예를 들어, DNA 서열의 경우 {A, C, G, T}로만 이루어지고, 이 때 D = 4이다. 그림 3은 생물학 서열 데이타베이스의 한 예를 보여준다. 그리고, 2.1에서 설명한 두 가지 형태의 문제에 대해, p-Suffix 트리를 사용하는 문제를

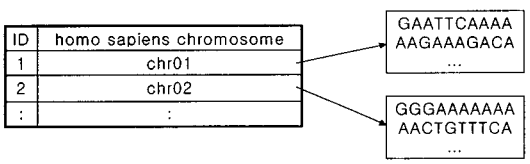


그림 3 생물학 서열 데이타베이스

1) p/o-Suffix 트리의 p는 presence, o는 occurrence의 약자이다.

p-선택도 문제, o-Suffix 트리를 사용하는 문제를 o 선택도 문제라 정의한다.

이 논문에서는 생물학 서열 데이타베이스와 질의 패턴이 주어졌을 때, 질의 패턴의 선택도를 추정하는 문제를 다룬다. 이 경우 p-선택도 문제보다는, 긴 서열의 경우 동일한 부분 문자열이 대단히 많이 반복되는 경향이 있으므로, o-선택도 문제가 더 의미가 있다고 판단된다. 정리 3.1의 문자열 패턴 pt의 최대 등장 회수를 이용하여, 정의 3.1과 같이 p-선택도와, o-선택도를 정의한다.

[정리 3.1] BSD에서 길이가 l인 문자열 패턴 p가 나타날 수 있는 최대 등장 회수는 $N-nl+n$이다.

[증명] 하나의 서열 α_i에 대해 p가 나타날 수 있는 최대 등장 회수는 $|\alpha_i|-l+1$이다. 예를 들어, "aaaaaa"에서 "aaa"가 나타나는 회수는 3(=5-3+1)이다. 이를 n개의 서열에 대해 적용하면, $\sum_{i=1}^{n}(|\alpha_i|+l-1)$이다. 이 때, $\sum_{i=1}^{n}|\alpha_i|$이 N이므로, $N-nl+n$이 된다. □

[정의 3.1] p-선택도와 o-선택도를 다음과 같이 정의한다.

(1) $p-$선택도 $=\dfrac{\text{문자열 패턴이 포함된 투플의 수}}{\text{모든 투플의 수}}$

(2) $o-$선택도 $=\dfrac{\text{문자열 패턴이 나타나는 회수}}{N-nl+n}$,

(l: 문자열 패턴의 길이)

3.2 카운트 서픽스 트리를 이용한 방법

카운트 서픽스 트리 생성 방법은 카운트를 계산한다는 점을 제외하고는 서픽스 트리와 동일하다. 서픽스 트리를 생성할 때, 각 노드의 문자열 저장 방법은 두 가지이다. 노드마다 실제로 문자열을 저장하거나, 전체 문자열에서의 시작 위치와 마지막 위치만을 저장[15]한다. 첫 번째 방법으로 생물학 서열에 대해 서픽스 트리를 만든다면 디스크 기반 서픽스 트리를 만든 후 가지치기 과정을 거치게 될텐데, 이는 필연적으로 생성 시간이 오래 걸리고, 생성 중의 트리 크기가 대단히 커지는 결과를 가져온다. 그리고, 두 번째 방법의 경우, 전체 문자열이 메모리에 있어야 하므로 제한된 크기를 만족하기 어렵게 된다.

4. 카운트 큐그램 트리

4.1 구조

카운트 큐그램 트리는 삽입하는 문자열이 다르다는 점을 제외하고는 카운트 서픽스 트리와 동일하다. 카운트 큐그램 트리는 모든 서픽스를 트리에 삽입하는 대신, 큐그램들을 슬라이딩 윈도우 방식으로 삽입하여 트리를 구성한다. 그림 4는 삽입 예를 보여준다. 문자열 "AAACTTA"에 대해 카운트 서픽스 트리를 구성하면 "AAACTTA", "AACTTA", ..., "A"가 삽입된다. 반면에 q가 4인 카운트 큐그램 트리는 문자열의 마지막에 q-1개의 종료 문자(#)를 덧붙인 뒤, 큐그램들("AAAC", "AACT", ..., "A###")을 삽입한다. 실제로 구현을 할 때는 # 문자를 이용한 변환 과정이 필요없지만, 설명을 쉽게 하기 위해 첨가하였다. 하나의 큐그램이 삽입되는 과정은 서픽스 트리와 마찬가지로 루트로부터 동일한 문자들을 따라오다가 동일한 문자가 없어지는 순간에 새로운 노드를 만들게 된다. 이 과정 중 만나는 노드들마다 카운트 값을 1 증가시키고, 새로 만들어지는 노드는 1로 설정한다.

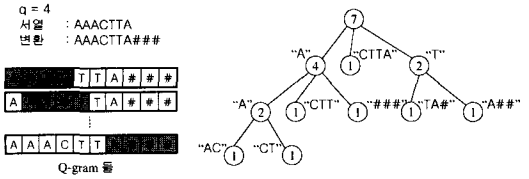


그림 4 카운트 큐그램 트리

4.2 완전성

카운트 큐그램 트리는 길이가 q 이하인 모든 부분 문자열들의 정확한 카운트 값을 가지고 있다는 특징이 있다. 이 특징은 정리 4.1과 정리 4.2에서 설명한다.

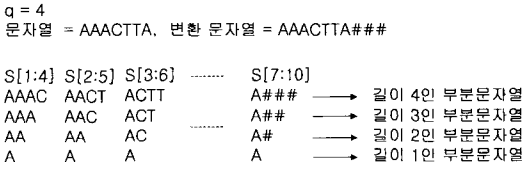


그림 5 카운트 큐그램 트리의 카운트 값들

[정리 4.1] 길이가 q인 부분 문자열 $\alpha[i:i+q-1]$을 카운트 큐그램 트리에 삽입하면, $\alpha[i:i+q-1]$의 모든 프리픽스(prefix)들의 카운트가 1 증가한다.

[증명] 큐그램을 삽입할 때, 루트로부터 한 문자씩 따라 내려오며 카운트 값을 1 증가시키는 카운트 큐그램 트리 생성 알고리즘과, 카운트 큐그램 트리에서 카운트 값을 읽는 방법에 의해 정리 4.1이 성립한다. 예를 들어 "ACTT"를 트리에 삽입하면, 루트에서 'A'를 찾아 1 증가시키고, 그 아래에 있는 'C'(즉, 루트로부터 생각하면 "AC")를 찾아 1 증가시킨다. 이와 같은 과정이 "ACTT"에 도달할 때까지 반복된다. □

[정리 4.2] (카운트 큐그램 트리의 완전성) 서열 α가 주어졌을 때, 카운트 큐그램 트리는 α에 있는 길이가 q 이하인 모든 부분 문자열들에 대해 정확한 카운트 값을 가지고 있다. 이는 하나 이상의 서열에 대해 성립한다.

[증명] 정리 4.1을 그림 5와 같이 모든 길이 q인 부분 문자열에 대해 적용하면 길이가 q이하인 모든 부분 문자열들에 대해 정확한 카운트 값을 갖게 된다. □

4.3 큐그램의 길이와 트리의 크기

카운트 큐그램 트리에서 단말 노드에 이르는 모든 경로는 q개의 문자열로 구성된다. 이 때, 매 글자마다 노드가 생기고 모든 글자들에 대해 분기가 생기면 트리 크기가 최대가 된다. 이 트리는 높이가 q인 완전 D-nary 트리로 크기는 다음과 같다.

$$\frac{D^q-1}{D-1}\times I + D^q \times L,$$

(I : 중간 노드의 크기, L : 단말 노드의 크기)

이 때, 정해진 메모리의 크기가 M일 때, 트리는 M보다 작거나 같아야 한다(식 (1)). 식 (1)을 q에 관해 정리하면 식 (2)를 얻게 되는데, 식 (2)를 만족하는 최대 값으로 q를 설정하면 된다.

식 (1) : $\frac{D^q-1}{D-1}\times I + D^q \times L \le M$

식 (2) : $q \le \log_D \frac{M+I}{LD-L+I}$

식 (2)에서 주목할 점은 카운트 큐그램 트리의 크기는 문자열의 길이 N과 상관없다는 점이다. 즉, 아무리 문자열 길이가 길고, 그러한 문자열들의 개수가 많아도 트리의 크기는 O(Dq)로 한정될 수 있다. 만약, 카운트 값이 큰 노드들 위주로 저장하고 싶다면, q를 식 (2)를 만족하는 값보다 크게 설정한 후 가지치기를 하는 방법도 생각할 수 있다. 이 경우 카운트 큐그램 트리의 완전성이 사라지게 된다.

4.4 구현

카운트 큐그램 트리를 구현할 때, [3]에서 설명한 카운트 서픽스 트리의 구현 방법이나 서픽스 트리의 압축 기법[16]을 사용할 수 있다. 그러나, 서픽스 트리 계열은 많은 포인터들이 필요하고, 각 가변 길이 문자열들을 저장하고 있어야 하며, 문자열 삽입이나 추정을 위해 트리를 탐색할 때마다 가변 길이 문자열들과 비교하는 연산이 필요하다.

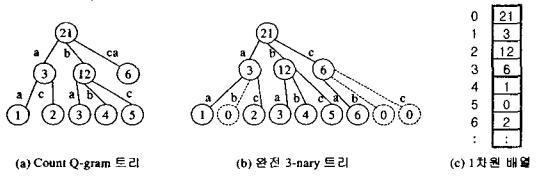


그림 6 카운트 큐그램 트리의 구현

DNA 서열처럼 도메인의 크기가 작으며 길이가 긴 서열들을 대상으로 카운트 큐그램 트리를 생성할 경우 완전 트리 또는 이에 가까운 트리가 생겨나기 쉽다. 이 성질을 이용하면 1차원 배열로 작고 빠르게 트리를 생성할 수 있다. 그림 6은 1차원 배열을 이용한 구현 모습을 보여준다. 부모 노드의 위치가 m일 때, 자식 노드들은 $mD+i$이다. 이 때, i는 각 글자의 도메인 내의 순서로, $1 \le i \le D$이다. 그림 6.(b)에서 "ac"는 'a'의 위치 1에 3을 곱한 뒤 'c'에 해당하는 3을 더한 6에 카운트 값이 저장된다. 1차원 배열 구현 방식에서는 문자열 공간과 포인터 공간이 절약되며, 문자열 비교가 없으므로 탐색이 빨라진다. 만약, 완전 트리에 가깝지 않다면 그림 6.(b)의 "ab"나 "cb"처럼 카운트 값이 0이 되는 노드들이 많이 생겨서, 배열의 많은 부분이 낭비되는 문제점이 발생할 수 있다.

1차원 배열에 의한 트리 생성 시간은 큐그램들마다 q번의 배열 갱신이 이루어지므로 $O(q \times N)$이다.

5. k번째 최대겹침 선택도 추정 기법

5절에서는 KVI[3]와 최대겹침 추정 방법[4]의 일반적인 형태인 k번째 최대겹침 추정 방법을 정의한다. 카운트 큐그램 트리는 완전성을 만족하므로, 길이가 q를 초과하는 질의 문자열에 대해, 트리에 존재하는 문자열들로 쪼개는 방법은 다양하다. 이 다양한 방법들을 일반화시킨 것이 k번째 최대겹침 추정 방법이다.

질의 문자열의 길이가 q 이하일 경우 카운트 큐그램 트리에서 정확한 선택도를 알 수 있다. 그러나, q보다 긴 질의 문자열은 큐그램들로 나눈 후, 각 카운트 값을 이용하여 선택도를 추정해야 한다. 예를 들어 q=4이고 최대겹침 추정 방법에 의해 "AACTTA"의 선택도를 추정하면, "AACT", "ACTT", "CTTA"로 나눈 뒤, 각 카운트 값을 이용하게 된다. 추정식은 다음과 같다. 단, 정의 3.1에 따라 N 대신 N-5n을 분모로 이용하였다.

$$\begin{aligned} \Pr(\text{AACTTA}) &= C_{\text{AACT}} \times \Pr(\text{T}|\text{AACT}) \times \Pr(\text{A}|\text{AACTT})/(N-5n) \\ &\approx C_{\text{AACT}} \times \Pr(\text{T}|\text{ACT}) \times \Pr(\text{A}|\text{CTT})/(\text{N}-5\text{n}) \\ &= C_{\text{AACT}} \times (C_{\text{ACTT}}/C_{\text{ACT}}) \times (C_{CTTA}/C_{CTT})/(N-5n) \end{aligned}$$

이 때, 질의로부터 나누어진 각 문자열들은 서로 q-1 글자씩 겹치게 된다. 질의 문자열의 길이가 q 보다 커지면 커질수록 질의 문자열은 더 많은 문자열들로 나누어지기 때문에, 카운트 큐그램 트리를 더 많이 접근해야 한다. 만약, 빠른 선택도 추정이 중요하다면 q-1 글자씩 겹치는 대

신, 겹치는 글자의 수를 줄이는 방법을 생각해볼 수 있다. 즉, "AACTTA"를 "AACT"와 "CTTA"로 나눈 뒤 선택도를 추정한다.

$$\begin{aligned} Pr(AACTTA) &= C_{AACT} \times Pr(TA|AACT)/(N-5n) \\ &\approx C_{AACT} \times Pr(TA|CT)/(N-5n) \\ &= C_{AACT} \times (C_{CTTA}/C_{CT})/(N-5n) \end{aligned}$$

위의 내용을 일반화한 것이 정의 5.1이다.

[정의 5.1] 질의 문자열을 q-k 글자씩 겹치도록 나누어 선택도를 추정하는 방법을 k번째 최대 겹침이라 정의한다. 이 때, 1번째 최대겹침이 최대겹침이고, q번째 최대겹침은 KVI이다.

6. 실 험

실험은 크게 두 부분으로 구성된다. 첫째, 호모사피엔스 염기 서열들에 대해 카운트 큐그램 트리가 O(N) 시간에 생성가능함을 실험으로 확인한다. 더불어 사용된 공간의 크기와 카운트 값들의 분포를 살펴본다(6.2절). 둘째, 생성된 카운트 큐그램 트리를 이용하여, k번째 최대겹침 문자열 선택도 추정 방법의 정확성을 측정한다(6.3절). 이 실험은 PIII-1GHz CPU, 40GB HDD, 768MB 메모리가 장착되고, WOWLiNUX Release 7.1이 운영체제로 설치된 컴퓨터에서 C++로 구현하였다.

6.1 데이타셋

사용한 데이타셋은 두 종류로 다음과 같다.

- 데이타셋 1 : 미 국립 생명공학 정보센터[17]가 제공하는 호모사피엔스 염기 서열 1과 2로 구성되어 있다. 이 염기 서열들의 도메인의 크기(D)는 5로, 기본적인 염기 A, C, G, T와 미확인 염기를 의미하는 N으로 구성되어 있다. 각 염기들의 등장 빈도는 표 1에 보여진다. 이 때, N의 등장 빈도가 다른 염기들에 비해 대단히 낮은 값을 가진다.
- 데이타셋 2 : 데이타셋 1에서 등장 빈도가 낮은 N을 제거한 염기 서열들이다. 즉, 도메인의 크기는 4이다.

표 3 데이타셋 1에 나타난 염기의 등장 빈도

염기	A	C	G	T	N
등장 빈도	136,512,829	94,518,252	94,464,939	136,210,212	1,400,357

6.2 카운트 큐그램 트리

표 4 사용 공간(바이트)

큐그램의 길이(q)	1	2	3	4	5	6	7	8	9	10	11
데이타셋 1	24	244	624	3K	15K	78K	390K	1.9M	9M	48M	244M
데이타셋 2	20	84	340	1K	5K	21K	87K	0.3M	1.3M	5M	22M

표 5 생성 시간(초)

큐그램의 길이(q)	1	2	3	4	5	6	7	8	9	10	11
데이타셋 1	58	66	73	80	91	102	113	148	184	237	303
데이타셋 2	60	66	74	82	90	102	113	131	180	222	276

1차원 배열로 구현한 카운트 큐그램 트리의 사용 공간과 생성 시간은 표 2와 표 3에 각각 보여진다. 사용 공간의 정확한 값은 배열의 크기 과 카운트의 데이타형의 크기(정수, 4 $(D^{q+1}-1)/(D-1)$바이트)를 곱하여 계산할 수 있다. 이 때, 데이타형의 크기를 $\lceil \log_2 N \rceil$ 비트로 압축하면 사용 공간은 줄어든다(N : 전체 염기 수). 생성 시간은 $O(qN)$으로 N이 고정되었을 때, q값에 비례한다.

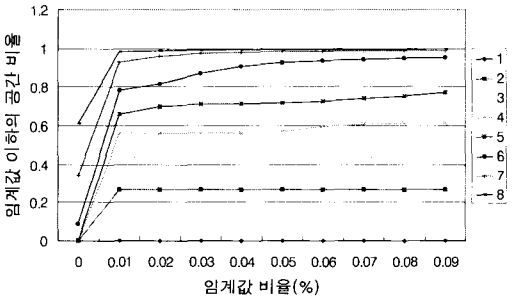


그림 7 데이타셋 1의 데이타 분석

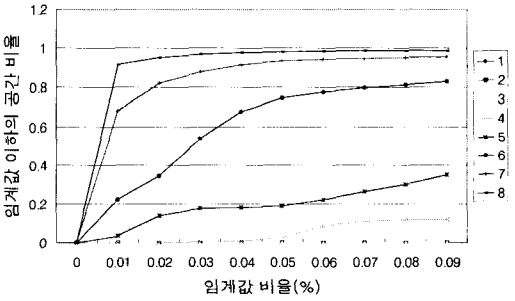


그림 8 데이타셋 2의 데이타 분석

1차원 배열로 구현시 임계값 이하의 카운트가 차지하는 공간의 비율은 그림 8과 그림 9에 보여진다. 범례는 카운트 큐그램 트리로, q 값에 따라 1부터 8까지 8개의 트리를 나타낸다. x축은 임계값 비율(임계값*100/N)이고, y축은 전체 배열 공간에서 임계값 이하의 배열 공간이 차지하는 비율을 나타낸다. q가 커질수록 임계값 이하의 공간 비율이 증가하였는데, 이는 전체 배열 공간

이 기하급수적으로 증가하기 때문이다. 그림 8은 q=1일 때만 x축에 붙어있지만, 그림 9는 q=1, 2, 3일 때 x축에 붙어있고 동일한 q 값에 대해 임계값 이하의 공간 비율이 높았다. 이는 데이타셋 1이 염기 N 때문에 임계값 이하의 공간 비율이 더많이 생성되었기 때문이다. 또한, 그림 8과 그림 9는 염기 서열 데이타셋들의 데이타 특성을 보여주는 것으로 의미있는 정보를 저장하기 위해 q 값이 더 클 필요는 없음을 알 수 있다.

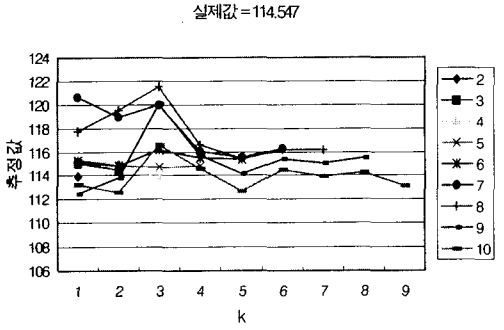


그림 9 데이타셋 1의 추정값

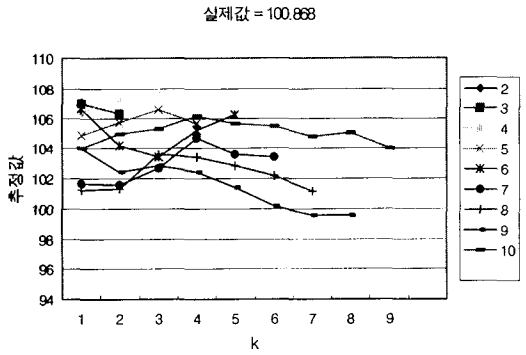


그림 10 데이타셋 2의 추정값

6.3 k번째 최대겹침 선택도 추정

k번째 최대겹침 방법의 선택도 추정의 정확성을 평가하기 위해, 각 데이타셋에 대해 길이가 11인 질의 문자열 1000개를 임의로 생성하였고, q 값을 변화시키며 카운트 1-그램 트리에서 카운트 10-그램 트리까지 생성하였다. 그리고, 각 트리마다 질의 문자열의 겹치는 정도를 의미하는 k를 변화시켜 선택도의 정확성을 평가하였다. 이 때, k는 $1 \le k \le q-1$이다.

그림 10과 그림 11은 각 데이타셋에 대한 선택도 추정 결과를 보여준다. 범례는 q 값이 2부터 10까지인 카운트 큐그램 트리를 의미하며, x축은 각 트리에서 질의의 선택도 추정시 겹치는 문자열의 정도를 나타내는 k이고, y축은 추정값이다. 실제값과 추정값 모두 1000개의 질의 문자열에 대한 평균값이다. q 값이 클수록, 그리고 k 값이 작을수록 보다 정확하게 추정할 것으로 예상한 것과는 달리, 카운트 큐그램 트리의 종류와 k 값에 따라 조금씩의 차이는 있었지만, 대부분의 경우에서 거의 실제값에 가깝게 추정하고 있음을 보여주었다.

7. 결 론

수년 전부터 문자열 데이타에 대한 선택도 추정 문제가 연구되어왔다. 지금까지 선택도 추정을 하는 방법으로는 문자열 데이타로부터 카운트 서픽스 트리를 생성한 다음, 이 트리를 이용하여 선택도를 추정하게 된다. 그런데, 카운트 서픽스 트리는 본질적으로 DNA 서열처럼 길이가 매우 긴 문자열 데이타들로 이루어진 데이타베이스에는 적합하지 않다. 수백 메가바이트에 달하는 서열에 대해 모든 서픽스를 삽입하는 과정 자체가 공간과 시간 모두 비효율적인 연산을 가져오게 된다. 그래서, 이 논문에서는 모든 서픽스를 삽입하는 대신, 길이가 q인 부분 문자열들만을 삽입하는 카운트 큐그램 트리를 제안하였다. 카운트 큐그램 트리는 길이가 q 이하인 모든 부분 문자열들의 정확한 등장 회수를 저장하고 있는 완전성을 지니고 있으며, 일정한 길이의 부분 문자열들만을 삽입하기 때문에 적은 메모리 한계 내에서도 빠른 생성이 가능하다. 즉, 카운트 서픽스 트리가 긴 서픽스들을 삽입한 후 카운트 값에 의한 노드 가지치기라고 한다면, 카운트 큐그램 트리는 긴 서픽스들을 삽입 전 길이에 의해 가지치기를 먼저 한다는 의미를 담고 있다. 삽입 전 길이에 의한 가지치기에 의해 기존의 서픽스 트리들이 겪고 있는 구현상의 문제를 해결한 1차원 배열에 의한 트리 구현이 가능해졌다. 1차원 배열에 의한 구현은 문자열과 포인터를 저장할 필요가 없으므로, 빈번한 동적 메모리 할당없이 공간을 절약할 수 있으며 빠른 카운트 연산이 가능하였다.

이 논문에서는 또한, 카운트 큐그램 트리가 만족하는 완전성에 의해 임의의 길이의 질의 문자열을 트리에 있는 문자열들로 쪼갤 때, 겹치는 정도를 조절할 수 있는 k번째 최대겹침 방법을 제시하였다. 이 방법은 [3]과 [4]에서 제시한 KVI와 최대겹침 방법을 통합하는 방법이다.

실험 결과 도메인이 커지고 q가 커질수록 트리의 크기는 증가하였지만, 수메가 또는 수십메가 이내에서 DNA 서열들에 대한 충분한 통계적 정보를 저장할 수 있었다. 또한, 다양한 q와 k에 대해 매우 정확하게 추정할 수 있음을 보여주었다.

참 고 문 헌

[1] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In Proceedings of VLDB, pages 311-322, 1995

[2] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In Proceedings of ACM SIGMOD, 1996

[3] P. Krishnan, J. S. Vitter, and B. Iyer. Estimating alphanumeric selectivity in the presence of wildcards. In Proceedings of ACM SIGMOD, pages 282-293, 1996

[4] H. V. Jagadish, R. T. Ng, and D. Srivastava. Substring selectivity estimation. In Proceedings of ACM Symposium on Principles of Database Systems, June 1999

[5] H. V. Jagadish, O. Kapitskaia, R. T. Ng, and D. Srivastava. Multi-dimensional substring selectivity estimation. In Proceedings of VLDB, 1999

[6] H. V. Jagadish, R. T. Ng, and D. Srivastava. On effective multi-dimensional indexing for strings. In Proceedings of ACM SIGMOD, pages 403-414, 2000

[7] P. Ferragina, N. Koudas, S. Muthukrishnan, and D. Srivastava. Two-dimensional substring indexing. In Proceedings of ACM Symposium on Principles of Database Systems, 2001

[8] Z. Chen, H. V. Jagadish, F. Korn, N. Koudas, S. Muthukrishnan, R. T. Ng, and D. Srivastava. Counting twig matches in a tree. In Proceedings of ICDE, pages 595-604, 2001

[9] A. Aboulnaga, A. R. Alameldeen, and J. F. Naughton. Estimating the selectivity of XML path expressions for internet scale applications. In Proceedings of VLDB, pages 591-600, 2001

[10] E. M. McCreight. A space-economical suffix tree construction algorithm. Journal of the ACM, Volume 23, pages 262-272, 1976

[11] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. Theoretical Computer Science, Volume 92, pages 192 211, 1992

[12] G. Navarro and R. Baeza-Yates. A practical q-gram index for text retrieval allowing errors. CLEI Electronic Journal, Volume 1, Number 2, 1998.

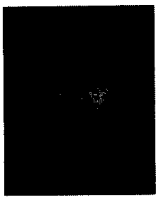
[13] G. Navarro, E. Sutinen, J. Tanninen, and J. Tarhio. Indexing text with approximate q grams. In Proceedings of Combinatorial Pattern Matching, pages 350-363, 2000

[14] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In Proceedings of VLDB, 2001

[15] E. Hunt, M. P. Atkinson, and R. W. Irving. A database index to large biological sequences. In Proceedings of VLDB, 2001

[16] R. Grossi and J. S. Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. ACM Symposium on Theory of Computing, pages 397-456, 2000

[17] ftp://ncbi.nlm.nih.gov/genomes/H_sapiens/, 2001



배 진 욱

1996년 서울대학교 컴퓨터공학과 졸업. 1998년 서울대학교 컴퓨터공학과 석사학위 취득. 1998년 3월~현재 서울대학교 전기컴퓨터공학부 박사과정. 관심분야는 데이타마이닝, 문자열 데이타베이스, XML 데이타베이스 등

이 석 호

1964년 연세대학교 정치외교학과 졸업. 1975년, 1979년 미국 텍사스대학교 전산학 석사와 박사학위 취득. 1979년~1982년 한국과학원 전산학과 조교수. 1982년~1986년 한국정보과학회 논문 편집위원장. 1986년~1988년 한국정보과학회 부회장. 1988년~1989년 미국 IBM T.J. Watson연구소 객원교수. 1988년~1990년 데이타베이스연구회 운영위원장. 1989년~1991년 서울대학교 중앙교육연구전산원 원장. 1994년 한국정보과학회 회장. 1997년~현재 한국학술진흥재단 부설 첨단학술정보센터 소장. 1982년~현재 서울대학교 컴퓨터공학부 교수. 관심분야는 데이타베이스, 멀티미디어 데이타베이스