단백질 이차구조의 검색을 위한 클러스터링된 세그먼트 인덱싱

서민구° 박상현
연세대학교 컴퓨터 과학과
{mkseo°, sanghyun}@cs.yonsei.ac.kr

Clustered Segment Indexing for Searching on the Secondary Structure of Protein

Minkoo Seo°, Sanghyun Park
Dept. of Computer Science, Yonsei University

요 약

바이오 인포메틱스에서의 데이터 검색은 DNA와 단백질 시퀀스에 대해서 주로 이루어지며, 지금까지의 연구는 주로 DNA와 단백질 1차 구조의 검색에 대해 이루어졌다. 단백질 2차구조는 1차구조 내 인접한 아미노산들의 공간적인 배열을 나타내며, 단백질의 기능을 예측하는데 중요한 3차구조의 지역적 아미노산의 특성을 나타낸다. 따라서 2차구조에 대한 검색은 단백질의 기능을 이해하는데 매우 중요한 역할을 한다[1]. 이 논문에서는 단백질 2차구조 및 질의 문자열을 세그먼트 단위로 나누고 검색하는 [4]의 방법을 개선하여 세그먼트를 조합한 클러스터 구조 및 Look Ahead를 사용해 Exact Matching 및 Wildcard Matching 질의를 효율적으로 처리할 수 있는 기법을 제시한다.

1. 서 론

바이오 인포메틱스에서 처리할 데이터의 양은 15~16개월에 2배씩 증가하고 있으며, 이러한 DNA 또는 단백질 시퀀스에 대한 질의는 기존에 잘 알려진 시퀀스로부터 미지의 시퀀스의 기능을 예측하는데 매우 중요한 역할을 한다. 이러한 시퀀스 질의에 대한 기존의 연구는 주로 DNA시퀀스 혹은 단백질 1차구조에 대해서만 이루어져왔으나, 실제로 단백질의 유사성에 대한 명확한 정의는 없는 실정이므로, 2차 구조의 구조적 유사성에 대한 정보가 더 유용한 생체고분자학적(biomolecular) 지식을 줄 수 있다[4]. 이 논문에서는 단백질 2차구조 시퀀스에 대한 Exact Matching 및 k 개의 와일드 카드가 포함된 질의 처리를 위해, 2차 구조 및 질의 문자열을 세그먼트단위로 분리하고, 세그먼트를 한데 묶은 클러스터를 사용한 인덱싱 기법을 제시한다.

2. 관련 연구

단백질 혹은 DNA시퀀스와 같은 문자열 검색 방법은 [5] 등에 제시되어 있으나, 이들 연구는 주로 edit distance를 유사성의 기준으로 사용하였으므로, 이 논문에서 다루는 wildcard문제에 직접적으로 적용이 곤란하다.

[4]에서는 질의 문자열 및 문자열 데이터를 세그먼트 단위로 나누어 저장 및 검색하는 방법을 제안하였다. 예를들어 eeehhll 과 같은 시퀀스는 eee/hh/ll 이라는 세 개의 세그먼트로 분리 및 저장한다. 그러나 단백질 2차구조에는 e, h, l 이라는 세가지 문자만 나타날 수 있다는 점을 고려한다면, 세그먼트만 사용해서는 동일한 문자와 동일한 길이를 가진 행이 매우 많이 존재하여 검색 속도가 느려질 수 밖에 없음을 예측할 수 있다.

이 논문에서는 [4]에서 제안한 방법을 개선하기 위하여, LookAhead 및 세그먼트 클러스터링 기법을 사용해 단백질 2차구조를 검색하도록 한다.

3. PROTEIN 테이블의 구성

단백질의 1차구조를 검색결과로 반환하기 위하여, [4]와 마찬가지로 단백질의 1차구조와 2차구조를 저장한 테이블을 작성한다.

[표 1] PROTEIN 테이블의 구조

필드 명	의 미
ID	단백질의 ID
Primary	이 단백질의 1차 구조
Secondary	이 단백질의 2차 구조

4. 세그먼트의 구성

단백질의 2차 구조에서는 e, h, l 세 개의 구조만 나타나며, 각 문자가 독립적으로 나타나기 보다는 연속적인 형태로 나타난다 [4]. 데이터 문자열과 질의 문자열은 다음에 설명할 세그먼트의 구조로 분할한다.

4.1. 세그먼트의 구조

여기서 제시하는 세그먼트의 구성은 [6]에서 구현된 방식과 유사한 방식을 사용해 [4]에서 제안한 세그먼트 테이블에, 검색 속도를 높이기 위한 LookAhead 필드를 추가한다.

[표 2] 세그먼트의 구조

필드 명	의 미
ID	단백질의 ID
Loc	단백질 2차 구조 내 세그먼트의 시작 위치
Type	2차 구조의 유형 (e, h, l 중 하나)
Len	2차 구조의 길이
LookAhead	이 세그먼트 뒤에 오는 세그먼트들의 단백질 2차 구조 중 처음 n개 세그먼트의 Type.

예를 들어 n=2인 경우, 단백질 1에 대한 eeehhll 와 같은 시퀀스는 다음과 같이 분할된다.

ID	Loc	Type	Len	LookAhead
1	0	e	3	hl
1	3	h	2	l
1	5	l	2	

세그먼트는 시퀀스를 분할하는 개념이며, 실제 저장은 세그먼트를 묶은 클러스터로 이루어진다.

5. 클러스터 테이블 구성

세그먼트만을 사용하여 인덱스를 구축한 [4]에서는 주로 인덱스의 정선도(selectivity)를 기준으로 검색 방법을 정했다. 그러나 단일 세그먼트만을 검색하면 전체 단백질에서 매우 많은 검색결과가 나타날 수 있다. 따라서, 인덱스 검색 시 랜덤 I/O가 너무 많아지고, 검색결과가 많아 검색 후 n-way merge-sort 시 조인의 부담이 너무 커지는 문제점이 있음을 예상할 수 있다. 따라서 이 논문에서는 다수의 세그먼트를 모아 클러스터를 구성함으로써 검색 속도를 높이는 방법을 제안한다.

이 때, 클러스터링 할 세그먼트의 개수는 [5]에서 사용한 range_search 알고리즘과 같이 2^k (단, k≥0인 정수) 형태로 증가하게 한다. 단, 저장공간의 한계를 감안하여 k의 상한을 정한다.

5.1. CLUSTER 테이블의 구조

연속된 2^k (단, k≥0인 정수)개의 세그먼트를 하나의 튜플로 묶어 클러스터 테이블을 구성한다.

[표 3] CLUSTER 테이블의 구조

필드 명	의 미
ID	표 2 참고
Loc	표 2참고
k	세그먼트의 개수가 2^k 일 때, k.
TypeStr	2^k개 세그먼트의 2차 구조 유형
TypeLen	2^k개 세그먼트의 2차 구조 총 길이
LookAhead	표 2 참고

예를 들어 n=2인 경우 단백질 eee/hh/ll/ee는 다음과 같이 CLUSTER 테이블 내에 저장된다.

ID	k	Loc	TypeStr	TypeLen	LookAhead
1	0	0	e	3	hl
1	0	3	h	2	le
1	0	5	l	2	e
1	0	7	e	2	
1	1	0	eh	5	le
1	1	3	hl	4	e
1	1	5	le	4	
1	2	0	ehle	9	

6. 검 색

검색을 위해 CLUSTER 테이블을 작성한다. 인덱스는 B+ tree를 사용하여 CLUSTER 테이블의 k, TypeStr, TypeLen 컬럼에 인덱스를 구성한다.

질의 처리는 클러스터를 이용한 검색과 검증의 두 단계로 이루어진다. 먼저 클러스터 테이블을 사용해 질의를 만족하는 연속된 세그먼트의 시작 위치를 검색한다. 그 뒤, PROTEIN 테이블에 직접 접근해 secondary 시퀀스가 일치하는지 확인한다. 검증 단계가 필요한 이유는 CLUSTER 테이블의 TypeLen에는 2차 구조의 개별 문자가 아닌 전체 문자열에 대한 길이만 저장되어있기 때문에 False Match 가 발생하기 때문이다.

6.1. Exact Matching

Exact Matching 질의가 들어온 경우 문자열을 길이에 따라 앞쪽에서부터 k 가 큰 수로부터 greedy로 잘라낸다. 예를 들어, 길이 35의 질의 문자열은 k의 상한이 4인 경우 2^4, 2^1, 2^0 인 네 개의 클러스터($2^4\times2 + 2^1\times1 + 2^0\times1$)로 자른다.

검색에는 미리 구성한 B+ tree 인덱스를 사용하고, 질의의 다음 클러스터와 CLUSTER 테이블에 미리 저장된 LookAhead를 비교함으로써 클러스터의 검색 결과를 줄이도록 한다.

CLUSTER 테이블의 검색 결과는 ID와 각 클러스터간 Loc, Len을 기준으로 sort-merge로 조인하며, 최종적으로 PROTEIN 테이블로 검증해 결과를 출력한다.

예를 들어 eeehhhhlll 에 대한 Exact Matching 질의에 대한 CLUSTER 테이블의 검색 단계는 다음 SQL 문과 동일한 방식으로 수행된다.

```
SELECT * FROM CLUSTER C1, CLUSTER C2
WHERE C1.ID = C2.ID
AND C1.Loc = C2.Loc - 7
AND C1.TypeStr = 'eh' AND C1.TypeLen = 7 AND C1.LookAhead
like 'l%' AND C1.K=1
AND C2.TypeStr = 'l' AND C2.TypeLen = 3 AND C2.K=0
```

6.2. Wildcard Matching

6.1 절과 유사한 방법으로 질의 문자열을 자르고 검색한다. 단, wildcard를 만나면 wildcard를 기준으로 질의를 둘로 나누어 각각 Exact Matching으로 검색하고 검색된 클러스터의 위치를 wildcard의 개수를 기준으로 조인한다. 그 뒤 PROTEIN 테이블을 통해 검증해 결과를 출력한다.

예를 들어, 질의 llleeehh???h 는 llleee / hh / h 로 나누어 3회의 Exact Matching 검색을 수행한 뒤, 검색결과의 Loc를 따져 조인한다. 따라서 CLUSTER 테이블의 검색 단계는 다음 SQL문과 동일한 방식으로 수행된다.

```
SELECT * FROM CLUSTER C1, CLUSTER C2, CLUSTER C3
WHERE C1.ID = C2.ID AND C2.ID = C3.ID
AND C1.Loc = C2.Loc - 6 AND C2.Loc = C3.Loc - 5
AND C1.TypeStr = 'le' AND C1.TypeLen = 6 AND C1.LookAhead
like 'h%' AND C1.K=1
AND C2.TypeStr = 'h' AND C2.TypeLen = 2 AND C2.K=1
AND C3.TypeStr = 'h' AND C3.TypeLen = 1 AND C3.K=0
```

7. 실 험

실험에 사용한 데이터는 PIR[7]의 단백질 1차 구조를 사용했다. 그 뒤 PIR 내 단백질 총 10,000개를 PREDATOR[2,3]를 사용하여 2차구조로 변환했다. 비교대상은 세그먼트만을 구성한 테이블을 Full Table Scan으로 검색한 경우와 [4]의 MISS에서 모든 세그먼트를 인덱스로 검색한 경우로 하였다.

아래 실험결과 중 n은 LookAhead에 저장한 2차 구조의 개수, k는 CLUSTER 테이블 작성에 사용한 k의 상한 값, |Q| 는 질의 문자열내 세그먼트의 개수를 의미한다.

실험 시 질의 중 절반은 무작위(random)로 생성하였으며, 나머지 절반의 질의는 실제로 데이터베이스 상에 존재하는 데이터를 사용하였다.

Wildcard Matching 실험 시에는 질의 내 1%의 확률로 wildcard가 존재하는 것으로 하였으며, 질의가 아무리 짧더라도 최소한 한 개의 wildcard가 존재하도록 질의를 작성하였다.

7.1. Exact Matching

① n=2 / k=4 / |Q|=2,6,9,10,50,80,100

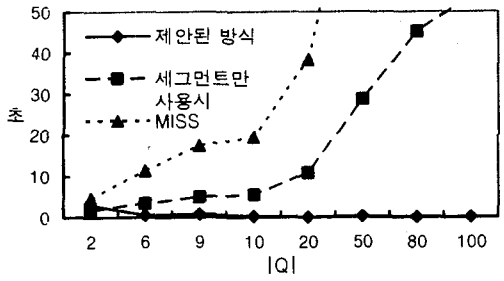

그림 1. Exact Matching의 성능 비교

전반적으로 제안된 방식이 우월한 결과를 보였으며, 질의의 길이가 길어질수록 성능 격차가 커짐을 알 수 있다.

② |Q|=2,6,9,10,20,50,80,100 / k=4 / n=1,4,16

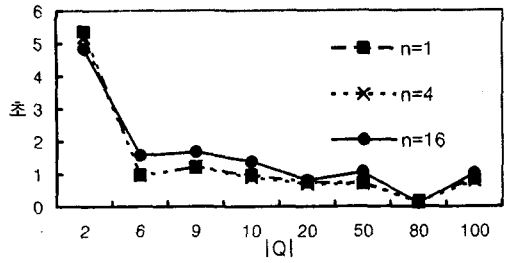

그림 2. look ahead의 개수와 수행시간의 상관관계

Look ahead의 개수를 바꿔가며 실험을 반복해보았지만 질의 처리 시간에 큰 차이가 보이지 않아 *look ahead*와 실행시간간의 명확한 상관관계는 드러나지 않았다. 이런 결과는 실험에서 사용한 데이터의 크기로 인해 빚어진 것으로 본다. 데이터의 크기가 영향을 주게 된 이유는 첫째, 테스트 데이터의 크기가 작기 때문에 중복 발생하는 클러스터가 적어 look ahead를 사용하더라도 사용하지 않은 경우에 비해 그다지 많은 데이터가 필터링 되지 않기 때문이다. 둘째, 검색된 결과가 작아 sort-merge 조인을 메모리 상에서 수행하게 되므로 검색된 행의 개수의 차이가 실행시간에 큰 영향을 주지 않았기 때문이다.

③ |Q|=2,6,9,10,20,50,80,100 / n=2 / k=1,3,5

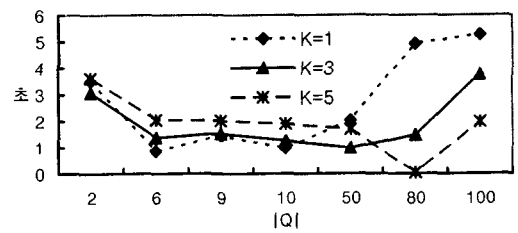

그림 3. 클러스터의 크기와 수행시간의 상관 관계

실험 결과, 질의의 길이가 길다면 K가 큰 경우에 명확히 득을 보게 됨을 알 수 있다. 그러나 K가 커지면 CLUSTER 테이블의 크기도 커지고 따라서 CLUSTER 테이블에 대한 인덱스 크기가 커진다는 단점이 있다. 따라서 질의가 짧고 k가 큰 경우, 더 큰 인덱스 크기로 인해 질의 처리 속도가 느려지는 점을 볼 수 있다. 만약 단 하나의 k를 선택해야 한다면, 2~100 사이의 질의에서는 K=3인 경우가 인덱스 크기와 클러스터링의 효과의 두 측면에서 적절한 타협점이 됨을 알 수 있다.

7.2. Wildcard Matching

① n=2, k=4, |Q| = 6,9,10, 50,80,100

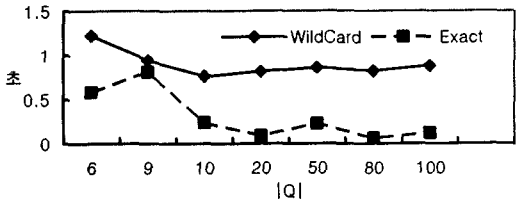

그림 4. Wild Card / Exact Matching 의 수행시간 비교

Wildcard 를 기준으로 질의가 좌우로 나뉘므로 수행시간은 질의가 길어지더라도 거의 일정하게 유지되었다.

8. 결 론

이 논문에서 제안한 방법은 2차 단백질 시퀀스를 세그먼트로 나누고, 세그먼트들을 클러스터로 묶어 질의를 처리하는 인덱싱 기법이다. 실험 결과 세그먼트만 사용하였던 기존 방식에 비해 성능이 향상되는 것을 확인할 수 있었다.

클러스터링이 성능향상의 주된 요인이었지만, 클러스터링 정도인 k가 너무 크면 질의의 길이가 짧을 때에는 인덱스 크기의 증대로 인한 성능이 저하가 관찰되었다. 따라서 향후 연구에서는 질의의 길이에 따라 동적으로 k 를 조정하여 검색하는 기법과, 질의 문자열을 오버랩핑 되도록 잘라 검색하는 방법의 적용을 통한 성능향상 기법을 찾고자 한다.

참고 문헌

[1] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson. Molecular Biology of the Cell, 3rd ed. Garland Publishing, Inc., 1994.

[2] D. Frishman and P. Argos, "75% accuracy in protein secondary structure prediction", in Proteins, 27:329-335, 1997

[3] D. Frishman and P. Argos, "Incorporation of long-distance interactions into a secondary structure prediction algorithm", in Protein Engineering, 9:133-142, 1996

[4] L. Hammel and J. M. Patel, "Searching on the secondary structure of protein sequences", in VLDB, 2002.

[5] T. Kahveci and A. K. Singh, "An efficient index structure for string databases", in VLDB, 2001.

[6] S. Park, D. Lee, and W. W. Chu, "Fast retrieval of similar subsequences in long sequence databases", in KDEX, 1999.

[7] C. H. Wu, L-S. L. Yeh, H. Huang, L. Arminski, J. Castro-Alvear, Y. Chen, Z-Z. Hu, Robert S. L., P. Kourtesis, B. E. Suzek, C. R. Vinayaka, J. Zhang, and Winona C. Barker. "The protein information resource", in Nucleic Acids Research, 31: 345-347, 2003.