단백질 부분 구조를 위한 효율적인 오류 허용 알고리즘

김계형° 김민호 허준호 R.S. Ramakrishna
광주과학기술원 정보통신공학과
{kgh4001°, mhkim, jhher, rsr}@kjist.ac.kr

An Efficient Fault Tolerant Apriori Algorithm for Local Protein Structures

Gye Hyung Kim° Minho Kim Junho Her R.S. Ramakrishna
Dept. of Information & Communications, K-JIST

요 약

단백질 부분 구조는 일종의 단백질 패턴으로써 진화적인 성질을 띄고 있다. 본 논문에서는 단백질 간의 열 안정성과 이러한 단백질 부분 구조 간의 관련성에 대해서 알아보고자 한다. 또한 오류 허용 알고리즘 (FT-Apriori)의 성능을 향상시킬 수 있는 효과적인 기법을 제안한다. 이러한 기법을 단백질 부분 구조에 적용시킴으로써 실제 단백질 데이터에서 그 효용성을 알아본다.

1. 서 론

최근 들어 데이터 마이닝에 대한 관심이 급증하고 있다. 특히 게놈 프로젝트 이후 발견된 단백질의 수가 기하급수적으로 증가하고 있는 요즘은 더욱 그 관심이 높아지고 있다. 더욱이 실제의 데이터들은 정제 되어 있지 않고 어느 정도의 오류를 내포하고 있다. 이렇게 오류를 포함하고 있는 데이터들에서 우리가 원하는 유용한 정보(연관관계, 패턴)를 정확하게 추출하는 데는 어려움이 따른다. 그래서 이러한 문제를 해결하기 위한 많은 노력이 있었고 그 중 주목할 만한 것으로 오류 허용 알고리즘 (Fault-Tolerant Frequent Pattern Mining)을 들 수 있다. 이 알고리즘 (FT-Apriori)[3]은 δ라는 오류 허용 정도를 두어 최대 δ개의 오류까지를 허용하고 있다. 즉, 최대 δ개의 오류까지는 같은 item으로 생각해서 연관규칙을 발견하는 것이다. 이 알고리즘은 기존의 알고리즘에 비해 일반적이면서도 훨씬 길이가 긴 pattern을 발견할 수 있다는 장점을 갖고 있다. 하지만 이런 장점에 비해 FT-Apriori는 너무나 방대한 양의 candidate itemsets을 만들어 내서 메인 메모리 내에 탑재가 불가능 할 뿐만 아니라 실행 시간 또한 증가 하는 등의 문제점을 안고 있다.

본 논문에서는, 이러한 문제를 해결하면서 FT-Apriori의 능률을 향상시킬 수 있는 기법을 제안한다. 또한 FT-Apriori를 실제 부분 구조 단백질에 적용하여 그 효용성을 알아보고자 한다. 부분 구조 단백질[1]에 대한 연구는 단백질 전체의 성질을 이해하는데 많은 도움이 된다. 또한 단백질의 성질 중 열 안정성은 매우 중요한 문제이다. 그래서 본 논문에서 우리는 고온에서 생존하는 *Thermus thermophilus* 라는 박테리아와 가장 실험에 많이 이용되는 대장균 박테리아를 대상으로 단백질의 열 안정성에 대한 부분 구조적인 차이점을 살펴보고자 한다.

2. FT-Apriori Scheme Using (δ+1)-index Lists

2.1 FT-Apriori

실제의 데이터들은 어느 정도의 오류를 포함하고 있다. 서론에서 기술하였듯이 FT-Apriori는 최대 δ개의 오류를 허용한다. 따라서 좀 더 길면서도 일반적인 정보를 얻을 수 있게 된다. 그림 1에서는 이러한 FT-Apriori 알고리즘을 기술한다.

Input: *Transaction database TDB, frequent-item support* threshold min_sup^{item}, FT-support threshold min_sup^{FT}, fault tolerance δ and length threshold *min_l*
Output: The complete set of FT-patterns.
Method:
1. Scan TDB once, find the set F_1 of global frequent items. *An item x is global frequent iff sup(x)* ≥ min_sup^{item}
2. Let $C_{\delta+1}$ be the set of all length-(δ+1) subsets of F_1. Let i=δ+1
3. Do {
   (a) Scan TDB. Check candidate itemsets in C_i;
   (b) Let F_i be the set of FT-patterns in C_i;
   If(i ≥ *min_l*) then output patterns in F_i;
   (c) If F_i is not empty, generate C_{i+1} from F_i.

A length-(i+1) itemset X is in C_{i+1} iff every length-i subset of X is in F_i;
(d) i = i+1;
} until either F_{i-1} or C_i is empty.

그림 1. FT-Apriori 알고리즘

2.2 (δ+1)-index Lists

대부분 단순 Apriori[2] 알고리즘은 빠른 접근을 위해 해쉬 트리(hash tree)를 이용한다. 이는 [4][5]에 자세히 설명 되어 있다. 하지만 FT-Apriori에서 해쉬 트리는 효과적이지 못하다. 그림 2를 통해서 보자. δ=1인 경우 candidate itemset " 457" 은 트랜잭션 " 12357" 에 의해 체크 되어져야 한다. 그러나 해쉬 트리의 경우에는 트랜잭션 " 12357" 은 " 457" 에 접근할 수 없어 확인이 불가능하다. 이렇듯 해쉬 트리는 모든 leaf node를 가로지르지 않는 한 FT-Apriori에는 적합하지 않다.

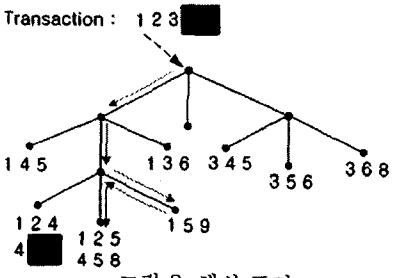

그림 2. 해쉬 트리

위의 그림에서 보았듯이 δ=1인 경우 candidate itemset " 457" 은" 45" ," 47" ," 57" 을 포함하는 경우를 다 고려해 봐야 한다. 이 세 가지 경우를 모두 확인하기 위해서는 3-candidate itemset의 처음과 두번째 item을 가리키는 index lists가 필요하다. 즉, (δ+1)개의 index lists가 필요한 것이다. δ=2인 경우에도 그림3의 ii)를 통해 이 같은 사실을 알 수 있다.

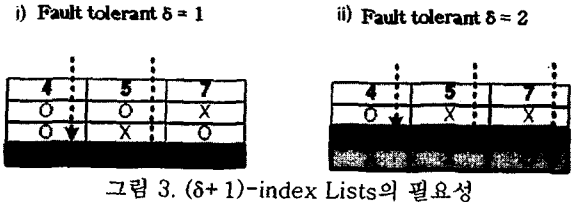

그림 3. (δ+1)-index Lists의 필요성

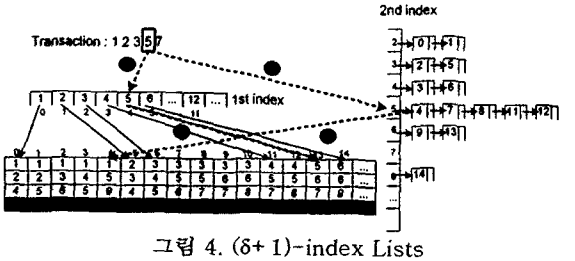

그림 4. (δ+1)-index Lists

그림 4는 (δ+1)-index lists를 보여주고 있다.

3. A process of Discovering Association Rules

본 논문에서는 유기체 간의 연관 규칙과 구성 요소(frequent pattern)를 찾아 비교함으로써 그들 유기체 간에 예상하지 못했던 유사성과 차이점을 찾아내고자 한다. 먼저, 비교하기 위해서는 각 유기체의 데이터베이스가 필요하다. 우선 *Thermus thermophilus* 의 PDB ID[6]들을 모은다. BLAST 알고리즘[7]을 이용해 *Thermus thermophilus*와 유사한 대장균(*Escherichia coli*)의 단백질들을 찾는다. 그 후 I-Sites[8]를 이용해 단백질 부분 구조를 찾아 데이터베이스를 구축한다. 이렇게 만들어진 데이터베이스를 이용해 마이닝을 하고 그 결과를 분석한다. 전체적인 과정은 다음과 같다.

1. Find all the PDB IDs of one organism from the PDB
2. Find similar proteins in the organism under comparison by applying the BLAST algorithm.
3. Search the local protein structures from I-sites
4. Construct databases consisting of local protein structures of each organism with high confidence (greater than 0.5)
5. Mine local protein structures
6. Discover the association rules among organisms

4. 실험 결과

4.1 Composition Comparison of Proteome vs. Proteome

각 유기체에서 가장 많이 포함된 부분 구조들의 구성 요소를 살펴보자. 다음 그림 5, 6에서 그 구성 요소를 볼 수 있다. 각 부분 단백질 구조에 대한 이름은 [8]를 참조한다. 각 유기체는 구성 성분은 비슷하지만 비율에 있어 차이가 있음을 알 수 있다. 이러한 구성 요소의 비율 간의 차이점에 대한 지속적인 연구는 앞으로 다른 유기체들을 구분하는데 있어 큰 도움이 될 수 있을 것이다.

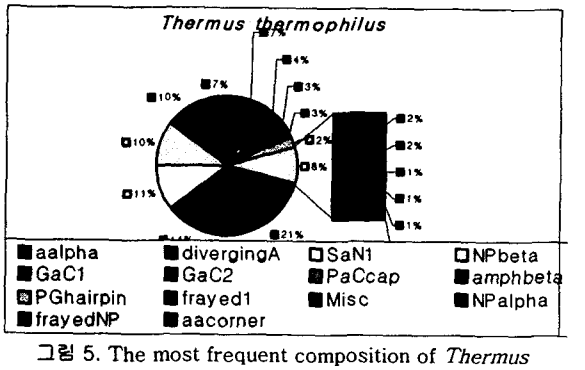

그림 5. The most frequent composition of *Thermus thermophilus*

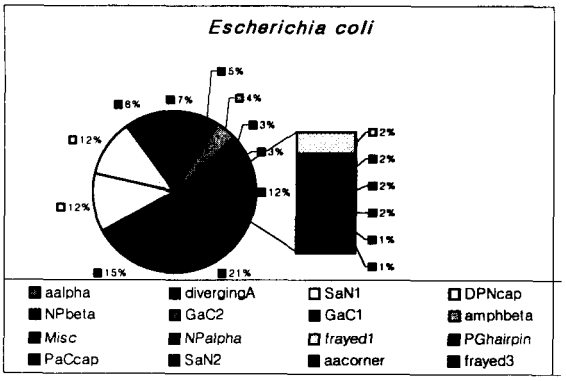

그림 6. The most frequent composition of *Escherichia coli*

4.2 Comparison of FT-Apriori and Apriori

이번 장에서는 단백질 구조와 관련해서 FT-Apriori 와 Apriori의 결과에 대해서 비교해 보고자 한다. 다음 표에서는 Apriori와 FT-Apriori의 결과를 보여준다. 실험을 통해 많은 연관 규칙이 발견되었으나 여기서는 각 유기체에서 공통적으로 발견되는 연관 규칙을 살펴본다. 다음 표 1과 2에서는 각 유기체에 대한 Apriori와 그에 대응되는 FT-Apriori의 결과를 보여준다. 이 표에서도 볼 수 있듯이 FT-Apriori가 더 긴 연관규칙을 찾음을 알 수 있다.

표 1. The result of Apriori and FT-Apriori working on *Thermus thermophilus*

Apriori	FT-Apriori
SaN1 aalpha ⇒ GaC2	*NPbeta DPNcap SaN1 aalpha GaC1 ⇒ GaC2*

표 2. The result of Apriori and FT-Apriori working on *Escherichia coli*

Apriori	FT-Apriori
SaN1 aalpha ⇒ GaC2	*NPbeta divergingA GaC1 SaN1 aalpha ⇒ GaC2*

5. 결론

본 연구에서는 같은 fold 구조를 갖는 enzyme에서의 연관 규칙들을 찾아보았다. 각 유기체는 한가지 공통 연관 규칙을 제외하고는 다른 연관 규칙들을 갖고 있었다. 이는 진화적 차이에 그 원인이 있을 가능성이 있다. 또한 이런 차이가 각 유기체의 열 안정성의 원인이 될 수 있을 것이다. 하지만 아직까지 모든 종에 대한 genome 연구가 끝나지 않았기 때문에 모든 종에 대한 연구가 끝나고 난 후 종간의 비교를 통해서 만이 이런 사실을 더욱 확실히 알 수 있을 것이다. 또한 각 유기체는 특징적인 연관 규칙을 갖고 있기 때문에 이는 다른 종이나 유기체를 구분할 수 있는 일종의 틀이 될 수 있다. 이렇듯이 FT-Apriori는 매우 효과적일 뿐만 아니라 *homo sapiens*와 같은 대용량의 단백질에 대해서는 더욱 효과적일 것이다.

참고 문헌

[1] C. Bystroff and D. Baker, Prediction of Local Structure in Proteins Using a Library of Sequence-Structure Motifs, J. Mol. Biol., 281:565--577, 1998.
[2] R. Agrawal, R. Srikant, Fast Algorithms for Mining Association Rules, Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, Sept. 1994.
[3] Jian Pei, Anthony KH Tung and Jiawei Han, Fault-Tolerant Frequent Pattern Mining: Problems and Challenges, Proc. ACM-SIGMOD Int, 2001.
[4] J. Han, M. Kamber, Data Mining Concepts and techniques, Academic Press, 2001.
[5] R. Agrawal, J.C. Shafer, Parallel Mining of Association Rules, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, December 1996.
[6] http://www.rcsb.org/pdb/index.html.
[7] http://www.ncbi.nlm.nih.gov/BLAST/.
[8] http://www.bioinfo.rpi.edu/~bystrc/lsites/by_motif.html