단백질 이름 추출을 위한 패턴 매칭 오토마타

[1]박준형[O] [2]홍기호 [1]양지훈
[1] 서강대학교 컴퓨터학과 데이터마이닝 연구실
[2] LS산전(주) / 중앙연구소
pustar[O]@gmail.com, khhong1@isis.biz, yangjh@sogang.ac.kr

Pattern Matching Automata for the Extraction of Protein Names

[1]Junhyung Park[O], [2]Kiho Hong, [1]Jihoon Yang
[1] Datamining Laboratory, Department of Computer Science, Sogang University
[2] IT Agent Research Lab, LSIS R&D Center

요 약

텍스트마이닝(text mining) 기법을 통해 생물학 문헌으로부터 단백질 이름과 그들 간의 상호 관계를 추출하는 시스템이 제안된 바 있다[1]. 이 시스템에서 단백질 이름을 추출하는 과정을 패턴 일치 오토마타(PMA: Pattern Matching Automata)라는 방법을 이용하여 좀 더 유연하고 높은 성능을 가지도록 개선할 수 있었다. 본 논문은 예제를 통해 PMA의 학습, 테스트 과정과 결과를 설명함으로써 단백질 이름 추출 작업에서의 PMA의 가능성과 성능 향상을 위한 앞으로의 방안을 제시한다.

1. 서 론

생명 공학의 눈부신 발전과 더불어 생명 공학 관련 문헌들도 셀 수 없을 정도로 많이 찾아볼 수 있게 되었다. 이러한 많은 문헌들 속에 우리가 아직 모르는 귀중한 정보가 담겨있을 수 있으나 사람의 손으로 이 정보들을 다 처리하는 데에는 무리가 따른다. 따라서 이 분야에 데이터마이닝 기법을 적용하는 것은 매우 의미 있는 일이다.

일반적으로 생명 공학 관련 문헌에 나타나는 단백질 이름은 매우 불규칙하고 모호한 경우가 많다. 이러한 이유로 단백질 작명 규칙이 존재하기도 한다. Nature Genetics[1] 에서 제안하는 단백질 작명 기준이 그 예이다. 그러나 많은 경우 이러한 규칙은 지켜지지 않고 특수문자나 그리스 문자, 숫자, 로마 문자 등이 뒤섞여 쓰인다[2]. 따라서 고정된 단어사전(lexicon)을 가지고 말뭉치(corpus)를 태깅(tagging)하는 경우 약간의 표기법 차이로 단백질 이름이 제대로 태깅되지 않는 경우가 많이 생긴다. 이 논문은 2005년에 제안된, 단백질 이름과 그들의 연관관계를 추출하는 시스템[1]에서 단백질 이름을 추출하는 방식에 좀 더 효과적인 방법을 시도해보았다.

2. 패턴 매칭 오토마타 (Pattern Matching Automata)

단백질 이름들은 그 작명법에서 어느 특정한 패턴을 가진다. 예를 들어, MR-3, MRP-13, SRIF-2 라는 경우엔 알파벳 대문자 다음에 하이픈, 그리고 숫자가 나오는 형태를 가지고 있음을 확인해 볼 수 있다. 이전의 시스템에선 단순히 이 특수 문자나 숫자들을 각각 약속된 기호(예: 숫자는 &, 특수문자는 # 등등)로 바꾸어 단백질 이름 부분 매치를 사용하였고, 그 결과 양호한 결과를 얻었다. 이처럼, 단백질 이름이 그 자체로 특정한 성향을 가진다는 점에 착안하여 단백질 이름을 추정하는 간단한 패턴 매칭 오토마타(PMA:Pattern Matching Automata)를 설계하였다. 이 오토마타의 개략적인 모습은 아래 그림 1과 같다. 각각의 상태에 대한 설명은 표 1에 간략히 나와 있다.

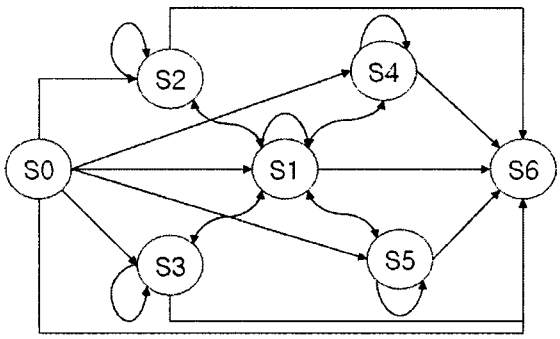

그림 1

표 1

상태	의미
S0	초기상태
S1	같은 문자 일치
S2	왼쪽 변수 포인터에 숫자 존재
S3	오른쪽 변수 포인터에 숫자 존재
S4	왼쪽 변수 포인터에 특수문자 존재
S5	오른쪽 변수 포인터에 특수문자 존재
S6	불일치. 초기상태로 돌아감

1) http://www.nature.com/ng/

이 오토마타는 두 개의 단백질 이름을 입력으로 받고, 두 입력에 대해 각각 전이 가능한 오토마타이다. 따라서 전이함수(transfer function)는 δ(A, B)와 같이 되며 A와 B에는 각각 치환된 단백질 이름이 오게 된다. 예를 들어 MRP-13과 MR-3을 입력으로 해서 비교할 경우를 생각해보자. 두 단백질 이름은 각각, MRP-13는 MRP#??, MR-3은 MR#?으로 치환된다.(앞으로 이렇게 치환된 단백질 이름을 이후 단백질 패턴이라 칭한다.) 그리고는 두 개의 단백질 이름을 가리키는 각각의 포인터에 위치한 문자들을 전이 함수에 따라 이동(shift) 혹은 소비(consume)시켜가면서 상태 전이를 시킨다. 이 과정을 간략하게 묘사하면 다음 표와 같다.

표 2

State	S0	S1	S1	S6	S6	S5	S1	…
A(left)	M	M	R	P	P	#	?	…
B(right)	M	M	R	#	#	#	?	…
Action	0→1	1→1	1→6	6→6	6→5	5→1	5→1	…

표 2에서 첫 행은 단백질 패턴 매칭 과정에서 거치게 되는 각각의 상태들을 보여준다. 두 번째, 세 번째 행은 각각의 단백질 패턴의 현재 포인터에 있는 문자들이다. 마지막 행인 Action 부분은 상태의 전이 과정을 보여준다. 다른 상태로 전이시, 각 상태와 상태를 잇는 간선(edge)에 가중치(weight)를 줘서 오토마타를 학습시키고, 마찬가지로 테스트 과정에서 해당하는 전이가 생길 때마다 가중치를 더해나가는 방법으로 전체 가중치의 합을 구할 수 있다. 좀 더 매칭 과정을 추상화하고 패턴 발견에 치중하기 위해서는 알파벳도 다른 문자로 치환하는 방법이 있다.(예: MRF-3을@@@#?으로 바꾼다. @은 알파벳에 대응하는 임의의 특수문자) 이어지는 장들에서 이 오토마타를 이용한 학습과 테스트 과정을 자세히 다룬다.

3. PMA 학습 단계

GenBank[2]와 Swiss-prot[3]의 데이터베이스에 존재하는 단백질들을 이용해서 단백질 리스트를 만들고 태거(tagger)의 단어사전에 이 리스트의 단백질들을 'GENE'이란 품사를 가지게 추가해준다. 그리고 태거에게 어휘 규칙(lexical rules)과 문맥 규칙(contextual rules)을 학습시키기 위해서 GENIA Corpus[3]를 이용한다. 태거로는 Brill's tagger[4]를 사용하였다. 이 태거를 이용해 우선 단백질 이름들을 문서에서 쉽게 추출할 수 있다. 이제 tagger로 학습하지 못한 단어들을 찾아내기 위해서 PMA를 학습시킨다. 학습 과정은 크게 다음과 같다.

1. 단백질 리스트에서 단백질 패턴들을 얻는다.
2. 단백질 리스트의 한 단백질마다에 대하여 단백질 리스트의 그 단백질을 제외한 모든 단백질들을 위의 PMA에 입력한다.
3. 단백질 리스트의 모든 단백질에 대해 같은 작업을 수행하며 PMA의 가중치를 누적시킨다.

이 과정을 거치게 되면 단백질 이름들이 서로 가지는 유사성 특성(similarity feature)이 PMA에 반영된다. 한 문자에 대한 전이시의 가중치는 다음과 같은 식으로 얻어진다.

$$W = \frac{1}{N_p \times L_p} \times C$$

식 1

여기서 W는 한 전이마다 더해지는 가중치, N_p는 단백질 패턴들의 총 길이, L_p는 단백질 패턴의 길이, 그리고 C는 임의의 상수이다. N_p는 단지 정규화 상수(normalization factor)이므로 이 식에서 의미 있는 부분은 비교하는 패턴의 길이인 L_p이다.

즉, 어떤 단백질 패턴들을 비교할 때 단순히 일치한 횟수를 보는 것이 아니고 그 단백질 패턴의 길이를 고려하는 것이다. 한 예로, 7문자로 이루어진 패턴이 완전히 다른 어떤 단백질 패턴과 일치한다면 매번 일치할 때마다 더해지는 가중치는 7의 역수일 것이다. 네 문자로 이루어진 단어에서 세 문자가 일치하는 경우와 열일곱 문자로 이루어진 단어에서 세 문자가 일치하는 경우, 전자가 더 높은 확률로 단백질 이름일수 있다는 가정을 생각해 볼 수 있다.

4. PMA 테스트 단계

위의 과정을 거쳐 학습된 PMA는 각 전이에 대한 가중치는 학습되어 있지만 어떤 단어를 넣어 얻은 가중치의 합이 어떤 의미를 가지는지는 알려주지 못한다. 즉, A라는 단어와 우리가 가진 단백질 리스트를 PMA에 입력했을 경우 얻어진 가중치의 합 그 자체로만은 이 A라는 단어가 단백질인지 아닌지를 말해주지 못한다. 따라서 단백질과 나머지로 구분해줄 문턱값(threshold value)이 필요하게 된다. 이 값을 구하기 위해, 우리는 다시 한 번 우리가 알고 있는 단백질들 사이의 가중치 합을 구해본다. 즉, PMA의 입력 값으로 A, B 단백질 패턴을 넣는데 이 A와 B는 우리가 이미 아는 단백질 리스트에 속하는 단백질 패턴이다. 문턱값은 다음 식을 사용해서 얻는다.

$$S_{total} = C_1 \cdot Avg + C_2 \cdot Max + C_3 \cdot Min$$

식 2

여기서 C값들은 상수이고 나머지 변수들은 각각 단백질들끼리 PMA를 통해 비교되었을 때 얻은 가중치의 평균값, 최대값 그리고 최소값이다. 이 상수 값들의 조정 역시 성능에 영향을 미친다. 이 상수들은 강화 학습(reinforced learning)등의 학습 알고리즘을 통해서 얻을 수 있다. C_2값을 제외한 다른 값을 0으로 하는 경우, PMA를 사용하지 않은 결과를 가져오며 C_3값만 사용하

2) http://www.ncbi.nlm.nih.gov
3) http://kr.expasy.org/sprot

는 경우에는 가장 넓은 범위에 PMA를 적용하는 경우가 된다. 위의 학습 과정을 거친 PMA에 단백질 이름인지 알아보고 싶은 단어들을 PMA의 첫 번째 입력 값으로 넣는다. 두 번째 입력 값들은 우리가 이미 알고 있는 단백질 이름들의 리스트가 된다. n개의 단백질 리스트가 있다고 할 때, n번의 수행을 거쳐서 각각의 가중치 합(sum-of-weights)을 구한다. 그 과정은 다음과 같다.

1. 모든 단백질 리스트의 단백질에 대해 비교하고자 하는 단어를 단백질 패턴으로 만들어서 비교한다.
2. 전이가 이루어질 때마다 그 전이의 가중치 값을 누적해 더한다.
3. 가중치 값과 문턱값을 비교해서 이 단어가 단백질 이름인지 아닌지를 알아낸다.

다음 장에서는 이와 같은 방법으로 실험한 결과와 성능 평가에 대해 다룬다.

5. 실험 결과

우리가 가지고 있는 97,018개의 단백질 리스트에서 29,163개의 단백질 패턴을 얻었다. 이 단백질 패턴들에 대해 위에 기술한 PMA를 적용하여 정확도(accuracy)와 재현율(recall)을 구해본 결과 각각 76.19%와 97.25%였다. PMA를 사용하지 않고 종전의 시스템에서 얻은 정확도와 재현율의 평균값은 각각 85.00%, 96.27%였다. 이 방법으로 정확도의 손실은 있었지만 재현율은 좀 더 올릴 수 있었음을 확인할 수 있었다. 정확도의 손실은 이와 같은 방법이 '단백질 이름과 비슷하나 단백질 이름은 아닌' 그러한 단어들을 많이 찾아내는 점에 기인한다. 비록 정확도는 떨어졌지만 이러한 단어들은 실제 단백질 이름의 일부이거나 혹은 단백질 이름과 관련이 깊은 단어라는 점에서 이것들을 찾아냈다는 것 자체도 의미 있는 작업이 될 수 있다고 할 것이다.

6. 성능 평가 및 결론

PMA가 실제적으로 이런 작업에서 높은 효율을 내기 위해서는 좀 더 사려 깊은 PMA 설계가 필요하다. 숫자와 특수문자의 위치뿐만 아니라 다른 여러 요소도 고려하는, 좀 더 표현력이 큰(예: 상태와 간선을 많이 가지는) PMA를 설계한다면 좀 더 높은 성능을 낼 수 있을 것이다. 그리고 PMA 자체가 스스로 학습되어 그 구조를 완성할 수 있게 할 수 있다면 설계 작업을 좀 더 유연하게 할 수 있을 것이다.

한 가지 더 생각해보아야 할 점을 든다면, 학습 단계에서 발생하는 오류(단백질이 아닌 단어에 PMA를 학습시킨 경우)가 가중치의 학습에 아무런 영향을 끼치지 못하는 점을 보완해야 한다는 점이 있다. 이러한 특성은 인공신경망(Artificial Neural Network)이 가지는 장점이지만 인공신경망 모델은 문자들의 순서성을 잘 반영하지는 못한다. 따라서 문자들의 순서가 중요한 의미를 가지는 이런 종류의 작업들에는 적절하지 못하다. 지금까지 제기된 이러한 점들을 고려한다면 PMA 방식이 생물정보문헌에서 단백질 이름들을 추출하는데 보다 더 유용하게 쓰일 수 있을 것이다.

7. 참고문헌

[1] K.H. Hong, J.H. Park, J.H. Yang and E.O. Paek. Automatic Extraction of Proteins and Their Interactions from Biological Text. In proceedings of DS-2005, pp. 322-329, 2005

[2] L. Tanabe and W. J. Wilbur. Tagging gene and protein names in full text article. In proceedings of Association for Computational Linguistics, pp. 9-13, 2004

[3] J.D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. Genia corpus - a semantically annotatedfor bio-textmining. Bioinformatics, 19, pp.180-192, 2002

[4] E. Brill. Some advances in transformation-based part of speech tagging, In proceedings of AAAI, 1994, pp. 722-727