진화연산을 이용한 자연어 파싱

김동민○, 박성배, 장병탁
서울대학교 공과대학 컴퓨터공학부 바이오지능 연구실
{dmkim○, sbpark, btzhang}@bi.snu.ac.kr

Natural Language Parsing through Evolutionary Computation

Dongmin Kim○, Seong-Bae Park, and Byoung-Tak Zhang
Biointelligence Lab, School of CSE, Seoul National University

요 약

본 논문에서는 진화 연산 기법을 이용한 자연어 구문 분석 기법을 제시한다. 기존의 확률 문맥 무관 문법(PCFG)에 관한 연구는 차트 파싱 방법을 구문 분석을 위한 기법으로 가정하고 있다. 하지만, 차트 파싱은 문장의 길이가 늘어날수록 복잡도가 크게 증가하는 문제를 안고 있다. 따라서, 차트 파서의 대안으로서 진화 연산 기법을 사용하여 이 문제를 해결하였다. 진화 연산의 적합도 함수로는 생성된 파스 트리의 확률을 사용하였다. 작은 규모의 자연어 문제에 적용한 결과, 진화 연산이 파싱 문제를 성공적으로 해결할 수 있음을 확인할 수 있었다.

1. 서 론

자연어를 이용한 많은 응용 문제에서, 주어진 문장을 트리 형태로 분석하는 것은 이제 표준이 되었다. 이 트리의 각 노드는 문장 구성 성분을 나타내는데, 언어학적으로 보면 이들은 문장 내에서 서로 응집하는 단위이며 그 기능에 따라 명사구(NP), 동사구(VP) 등으로 이름지어 진다.

문장에 허용된 구조를 지정하기 위해서 일반적으로 문법(grammar)을 사용하며, 자연어 처리에서는 문맥 무관 문법(context free grammar)이 가장 널리 사용된다. 하지만, 문맥 무관 문법이 사람의 언어를 완벽히 설명하지 못하기 때문에 하나의 주어진 문장에 대해서 여러 개의 파스 트리가 만들어 질 수 있다. 이를 구조적 모호성(syntactic ambiguity)이라고 하며, 자연어 처리에서 가장 큰 열린 문제 중의 하나이다.

이러한 구조적 모호성을 해소하기 위해서, 문맥 무관 문법의 각 전개 규칙에 확률을 부여한 확률 문맥 무관 문법(PCFG)이 사용될 수 있다[1]. 확률 문맥 무관 문법을 사용하면, 각 파스 트리의 확률을 계산할 수 있으므로 가장 가능성(likelihood)이 높은 트리를 선택하는 것으로 구조적 모호성이 해소된다. Treebank와 같은 대규모의 트리 부착 말뭉치가 이용 가능해 짐에 따라 확률 문맥 무관 문법이 성공적으로 사용되고 있다.

하지만, 문맥 무관 문법을 이용한 파싱 알고리즘들은 n을 문장의 길이라고 할 때 $O(n^3)$의 복잡도를 가진다[2]. 특히 널리 쓰이는 차트 파싱의 경우, 문장이 길어질수록 활성 엣지(active edge)의 수가 크게 증가하여 분석이 느려지게 된다. 본 논문에서는 이 문제를 해결하기 위해서, 진화 연산 기법을 사용한다. 진화 연산으로 구문 분석을 할 때, 가장 중요한 이슈(issue)는 적합도 함수(fitness function)의 정의이다. 본 논문에서는 적합도 함수값으로 PCFG로 계산한 확률을 사용하였다.

2. 자연어 파싱을 위한 진화 연산 알고리즘

본 논문에서 제시하는 파싱 알고리즘은 파싱하고자 하는 문장에 이미 품사가 태깅되어 있고, 파싱을 위해 촘스키 표준형(Chomsky normal form)의 확률 문맥 무관 문법이 주어진 것을 가정한다. 이는 파스 트리의 구조를 매우 단순하게 하며, 간단한 작업을 통해 임의의 확률 문맥 무관 문법을 촘스키 표준형으로 바꿀 수 있으므로 적절한 가정이라 생각한다.

이제 위의 두가지 가정을 만족하면서 길이가 n인 문장을 파싱한 트리의 구조를 살펴보자. 이 트리는 기본적으로 이진(binary) 트리이며, 단말(terminal) 노드 개수는 n 개이다. 또, n - 1 개의 비단말(non-terminal) 노드를 가지며, 각각의 비단말 노드는 일괄적으로 2 개의 자식 노드를 갖는다. 따라서, 이러한 파스 트리는 각 비단말 노드로부터 자식 노드들로 이어지는 2 개의 엣지(edge)를 하나의 정수로 인코딩하여 n - 1 길이의 정수열로 나타낼 수 있게 된다. 그림 1에서 6 개의 품사를 가진 문장의 파스 트리와 정수열 표현을 예시하였다. 그림의 비단말 노드의 위치에 나타난 t1, t2, …, t6 등은 실제로 주어진 품사열에서 하나하나의 품사를 나타내는 것으로 이해할 수 있다. 주어진 정수열을 파스 트리로 변환하는 것은 이진 탐색 트리(binary search tree)를 구성하는 방법과 동일하며, 다만 일반적인 탐색 트리 구성에서 행해지는 트리의 높이(height)를 조정하기 위한 작업은 필요하지 않다.

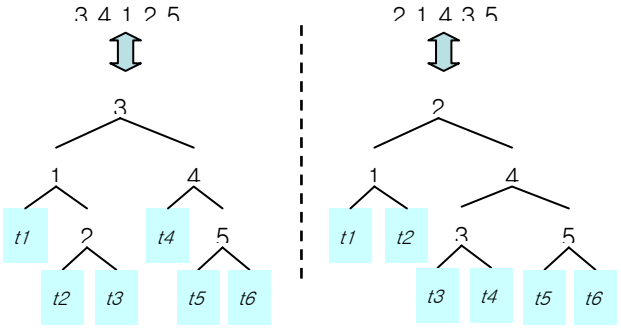


그림 1. 정수열과 파스 트리 변환 예제

이제 진화 연산에서 개체로 사용할 n - 1 길이의 정수열들을 생성할 수 있으며 각각의 정수열들은 n 개의 품사를 가진 문장의 서로 다른 파스 트리로 생각할 수 있다. 이와 같은 정수열 표현들을 통해 해(solution)를 진화시키는 진화 연산자(operator)들은 이미 많은 방법이 개발되어 있으며[3], 본 논문에서는 일점 교차 연산자(one point crossover)를 사용하였다. 교차 연산자는 주어진 문장의 파싱 과정 중 생성되는 파스 트리의 토폴로지(topology)를 변경하여 해당 문장에 대한 새로운 파스 트리를 구성하는 역할을 한다.

한가지 고려할 점은, 앞서 언급한 바와 같이, 어떤 파스 트리가 주어진 문장에 더욱 적합한 가를 가늠할 *적합도 함수*의 정의이다. 본 알고리즘에서는 여러 개의 유효한 파스 트리가 주어졌을 때, 각 노드의 확률을 곱하여 가장 높은 확률을 갖는 트리를 선택하는 확률 문맥 무관 문법의 파싱 방법을 이용하였다. 그림 2에서 그림 1의 왼쪽 파스 트리에 다음과 같은 2개의 문법 규칙을 적용하는 경우의 예를 보였다.

nt1 → t1 nt2 0.14
nt2 → t2 t3 0.18

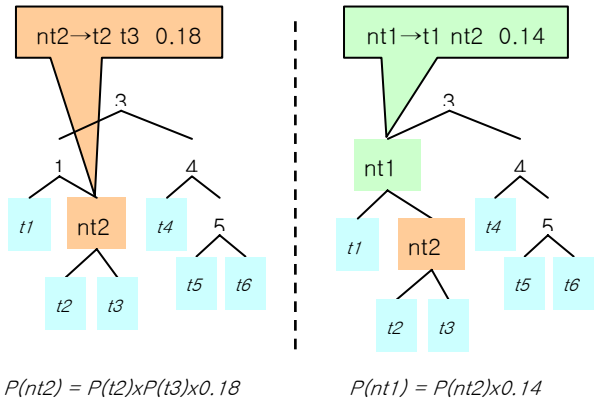


그림 2. 적합도 측정 예제

조금 더 구체적으로 살펴보면, 하나의 개체를 평가하기 위하여 위의 방법으로 파스 트리를 구성한 후, 먼저 단말 노드에 주어진 품사를 순서대로 할당하며 각 비단말 노드의 품사와 해당 노드에서 이루어지는 서브 트리(sub tree)의 확률을 상향식으로 계산한다.

이와 같이 계산된 루트 노드의 확률은 확률 문맥 무관 문법이 계산한 파스 트리의 확률과 같으며, 이를 직접적으로 해당 개체의 적합도로 이용하였다.

그림 3에서는 위의 과정을 행하는 적합도 함수의 유사 코드를 제시하였다.

```
function evaluate_tree (tree_node* curr, int index)
  if 현재 노드의 왼쪽 자식 노드가 없을 경우
    주어진 품사 중 위치에 적당한 품사 할당
    해당 품사의 확률 할당
    다음번 할당될 품사 index 조정
  else
    evaluate_tree (curr->left, index)   // recursion
    현재 노드의 왼쪽 자식의 품사 할당
    현재 노드의 왼쪽 자식의 확률 할당
  end if

  if // 오른쪽 자식 노드에 대하여 위의 과정을 반복
  else // recursion
  end if

  왼쪽 자식 노드 품사와 오른쪽 자식 노드 품사를 가지고
  파싱 규칙을 찾아내어 현재 노드의 품사와 확률을 계산
end function
```

그림 3. 적합도 함수의 유사 코드

진화 과정에서는 위의 적합도 함수에 의해 평가된 개체군(population)에서 적합도 비례 선택(proportional selection)을 통하여 부모를 선택하고 이들에게 교차 연산자를 적용하여 하나의 자식을 생성하는 방식으로 다음 세대 개체군을 구성하였다. 또, 서브 트리의 토폴로지가 결정되면 해당 서브 트리의 확률이 고정되는 적합도 함수의 특성을 고려할 때, 전체 진화의 과정에서 해(solution)가 지역 최적점(local minima)에 수렴할 가능성이 적다고 판단되어 변이(mutation) 연산자의 적용은 생략하였다. 같은 이유로, 엘리티즘(elitism)을 적용하여 수렴 속도를 향상시키고자 시도하였다.

3. 실험 및 결과

구현된 알고리즘을 이용하여 최적 파스 트리가 알려져 있는 다음과 같은 문장[1]을 파싱하였다.

astronomers saw stars with ears (1)

표 1. 확률 문맥 무관 문법

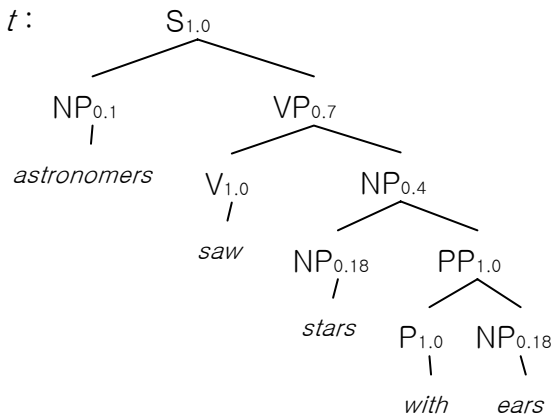
S→NP VP	1.0	NP→NP PP	0.4
PP→P NP	1.0	NP→*astronomers*	0.1
VP→V NP	0.7	NP→*ears*	0.18
VP→VP PP	0.3	NP→*stars*	0.18
P→*with*	1.0	NP→*telescopes*	0.1
V→*saw*	1.0		

1. 이 문장은 [4]에서 차용하였다.

문장 (1)의 파싱에 필요한 확률 문맥 무관 문법(PCFG)은 표 1에 나타내었으며, 문장 (1)과 주어진 문법으로부터 생성된 알고리즘의 실제 입력은 다음과 같다.

NP V NP P NP (2)

표 1의 문법을 가지고 본 알고리즘이 문장 (2)를 파싱한 결과는 그림 4와 같으며, 이 파스 트리가 [4]에서 제시한 최적해와 동일함을 확인하였다.



$P(t)$ = 1.0 x 0.1 x 0.7 x 1.0 x 0.4 x 0.18 x 1.0 x 1.0 x 0.18
= 0.0009072

그림 4. 최적 파스 트리

개체군 크기(population size)가 50일 경우에는 10세대 미만의 진화에서, 개체군 크기가 100일 경우에는 5세대 미만의 진화에서 대부분 최적해를 찾아내었다. 이는 작은 규모의 문제 해결에 있어서 성공적이라고 판단되나, 문장의 길이 n에 대하여 $O(n^n)$ 크기로 확대되는 탐색 공간을 고려할 때, 실제 크기의 문제 적용에서는 보다 큰 개체군 크기와 진화 세대수가 필요할 것으로 예상된다. 하지만, 현재의 알고리즘에서 각 개체의 길이는 문장 길이에 단순 비례하며, 대규모 문법 규칙 적용시에도 각 트리의 적합도 계산에 필요한 시간은 노드의 개수에만 의존하므로, 실제 규모의 문제를 해결할 가능성 역시 충분한 것으로 생각된다.

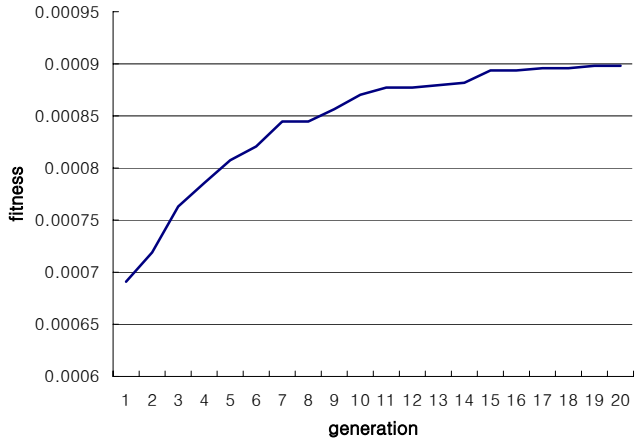


그림 5. 세대수에 따른 최적화 함수 평균값

그림 5는 개체군 크기 10일 때, 세대수 20으로 100회 수행한 적합도 함수 평균값의 변화 추이를 보여준다.

4. 결 론

자연어 파싱에 가장 많이 이용되는 차트 파서는 문장의 길이에 따라 그 복잡도가 기하급수적으로 증가됨으로서 실제 사용에 있어 큰 제약이 따르고 있다. 본 논문에서는 이러한 문제를 안고 있는 차트 파서의 대안으로서 진화 연산 기법을 이용한 자연어 파싱 알고리즘을 제안하였으며, 작은 규모의 자연어 파싱 문제를 성공적으로 해결하였다.

진화 연산을 통한 자연어 파싱의 장점은 파싱이 진행됨에 따라 가변적인 메모리 크기를 요구하는 차트 파서와 달리 문장 길이에 의해 고정된 크기의 메모리만을 사용한다는 점과, 차트 파싱에서 복잡도 감소를 위해 일반적으로 특정 문제에 특화된 휴리스틱(heuristics)을 사용하는 것과는 달리 파싱과정을 인위적으로 조작하지 않는다는 점을 들수 있다. 또 이러한 장점에 의해 파서의 구현이 쉽고 견고(robust)해 짐으로서 보다 큰 규모를 가진 문제로의 확장이 용이하다는 점을 꼽을 수 있다.

현재 본 논문의 알고리즘을 3000개 이상의 문법 규칙을 가진 현실적인 규모의 문제 해결을 위해 구현중이며, 이후로 이미 개발되어 사용되고 있는 자연어 파싱 알고리즘들과의 성능 비교를 해보고자 한다.

감사의 글

본 연구는 교육부 BK21 사업, 과기부 BrainTech 프로그램, 산자부 차세대신기술사업에 의하여 일부 지원되었음. 이 연구를 위해 연구 장비를 지원하고 공간을 제공한 서울대학교 컴퓨터신기술공동연구소에 감사드립니다.

참고 문헌

[1] E. Charniak, *Statistical Language Learning*, The MIT Press, 1993.

[2] M. Tomita, *Efficient Parsing for Natural Language*, Kluwer Academic Publishers, 1986.

[3] Thomas Back, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, 1996.

[4] Christopher D. Manning, and Hinrich Schutze, *Foundations of Statistical Natural Language Processing,* The MIT Press, 2002.