문맥 자유 문법 표현 방식을 사용한 유전 프로그래밍 응용의 한계성 : 자연 언어 문법 추론 문제

김강일O 장병탁
서울대학교
kangil.kim.01@gmail.com btzhang@snu.ac.kr

Limitation of genetic programming using Context Free Grammar representation : Inference of Grammars from Natural Language

Kim KangilO Zhang Byungtak
Seoul National University
kangil.kim.01@gmail.com btzhang@snu.ac.kr

요약

문법 추론은 자연 언어 처리 분야에서 오랫동안 연구되어오던 문제이다. 컴퓨팅 환경이 발전하면서, 결정적 알고리즘으로 풀기 힘든 문제들을 풀어내는 유전 알고리즘이 제시되었고, 이 문법 추론에도 사용되었다. 기존의 끈 타입이나, 트리 타입이 아닌, 더욱 복잡한 의존성을 가지고 있는 문법 표현 방식을 사용하기 때문에 다양한 방식의 연구가 진행되었지만, 아직 적용 문제의 범위는 간단한 문제들 정도로 제한되어있고, 자연 언어에서 문법을 추론하기 위한 연구는 많지 않다. 이 논문에서는 기존의 방식들을 살펴보고, 문법 표현 기반의 유전 프로그래밍 시스템을 큰 규모의 문제에 적용해봄으로써 어떤 문제들이 있는지 살펴보고, 앞으로의 연구 방향을 제시한다.

1. 서 론

John holland가 제시한 유전 알고리즘은, 컴퓨팅 환경이 급속하게 발전하면서, 기존의 결정적 알고리즘이 풀지 못하던 문제들을, 생물의 유전 과정과 흡사한 순환적 알고리즘으로 풀어내었고, 현실 생활에 많이 응용되어 왔다. 하지만 유전 알고리즘은 비교적 간단한, 끈 방식의 표현 형식을 사용하여, 개체의 내부 특성간의 긴밀한 연관 관계를 다루기에는 한계가 있었고, john koza는 트리 구조를 사용하여 조금 더 복잡한 실생활의 문제들을 풀어낼 수 있는 유전 프로그래밍을 제시하였다.

이런 진화적 연산들은 문법 추론에도 사용되었다. 주어진 문장들로부터 문법을 만들어내는 문제는 자연언어 처리에서 오랫동안 연구되어 오던 문제로, 결정적인 알고리즘을 사용해왔지만, 결국에는 이 알고리즘으로 해결하지 못할 수준의 복잡한 문제로 인식되었다.

Peter Wyard는 유전 알고리즘을 문맥 자유 문법 추론에 이용하였다. 그의 연구에서는 일반적인 유전 알고리즘을 문맥 자유 문법 형식에 맞게 고쳐 간단한 문법 추론 문제에 적용하는 것이었다. [1] ,

하지만, 그의 논문에서도 말하는 것처럼, 문법 표현 방식을 다루는 것은 쉬운 일이 아니다. 각 요소들간의 연관성이 기존의 끈 형식이나 트리 형식의 개체들에 비해서 훨씬 많다.

Mernik은 이런 연관성이 복잡한 개체들을 다루기 위해 유전 프로그래밍을 사용하였고, 특별한 연산자를 마련해 시스템을 정확도를 높이려는 시도를 했었다.[2]

이 연구들은 다분히 자연언어 처리나 문법추론을 연구하는 사람들의 입장에서 진행되어온 연구이다. 작은 규모의 문제들을 다루며, 문법 추론에서 문제가 되던,특정 형태의 문법들, 예를 들면, palindrom이나 recursive 형태의 문법을 잘 추론할 수 있는가에 대한 논의가 많이 이루어졌다.

이런 연구와는 다른 관점으로, 기계학습이나, 진화연산 분야에서는 각 요소들간의 연계성에 대한 확률적 학습을 통한 문법 추론을 연구하였다. Zhang은, 하이퍼 네트워크 모델을 기반으로 간단한 단어들의 조합 중에서 어떤 것들이 의미 있는 조합으로 이루어지는지를 학습하여, 일종의 네트워크 모델로 표현된 문법을 추론하

였다. [3]

학습 모델을 통한 문법 추론을 봤을 때 유전프로그래밍의 특징은, 매 순간 구조적인 학습이 대량으로 일어난다는 것이다. 모델 기반의 문법 추론은, 학습 자료를 통한 정보 획득, 수치적 학습, 구조적 학습을 하는데, 이 구조적 학습의 횟수나, 변화 정도가 영향을 미친다.

본 연구에서는 이런 관점에서 봤을 때 극단적인 구조 변화를 통해서 해를 찾아나가는 시스템을 구성한다. 이는 앞으로 진행될 구조 기반 학습 시스템을 이용한 연구를 위한 비교 대상으로써 의미를 가진다. 또한 위의 연구들을 바탕으로 어떤 것들이 문법 추론에 필요한 것인지 진화적 접근 방식에서 살펴 볼 것이다. 이를 위해서 문법 표현 방식을 사용하는 간단한 유전 프로그래밍 시스템을 구축하고, 자연 언어의 문법 추론에 응용해 본다.

2.1 장에서는 시스템의 구현, 2.2 장에서는 실험 설정. 2.3 장에서는 결론 및 앞으로의 연구방향에 대해서 논의한다.

2. 본 론

2.1 문맥 자유 문법 기반의 유전 프로그래밍 시스템

2.1.1.표현 방식

문맥 자유 문법을 각 개체의 표현 방식으로 사용한다. 문법은 Chomsky normal form 을 사용하였다. 이 문맥 자유 문법을 표현하는 양식은 GNF 나, BNF , CNF 등 여러가지가 있다. Wyard는 그 표현 양식의 변화에 따른 효과를 연구하였지만[3], 본 연구의 목적이 구조적인 변화와 연관성의 관찰이므로 조합하기에 편리하고, 작은 단위로 나누어져, 교배 연산을 쉽게 세밀하게 할 수 있는 Chomsky normal form을 사용하였다.

시스템은 모든 실험 문장들에서 사용되는 단어들로 구성된 테이블과 정해진 수의 nonterminal symbol set을 가진다. 각 개체는 starting symbol N0 - Ni의 임의의 심볼로 nonterminal set F를 만든다. 이 시스템은 모든 테스트 문장들에 사용되는 단어들을 테이블로 만들고 이 단어들을 production으로 가지는 nonterminal을 따로 생성한다. 이 nonterminal들은 T0 - Ti로 각 문법 개체들은 이 모든 단어들로부터 문법을 추론할 수 있도록 단어들을 T nonterminal에 할당한다.

2.1.2 진화 연산자

문법 표현 방식에서 진화 연산은 다른 표현 방식에 비해서 많이 고려해봐야 될 요소이다. 문법 표현 방식은 각 구성요소들간의 관계가 graph처럼 중복 상속을 허용하고, recursive link를 허용하며, 요소들이 production으로 한 구성요소가 되는 것이 가능하기 때문에 끈 표현 방식이나 트리 표현방식에 비해서 복잡하다. 기존의 유전 프로그래밍에서는 한 노드를 선택해서 교배를 할 경우, 그 노드의 자식들이 모두 같이 교배 상대와 교환되는 기본적인 유전 연산자를 사용해왔다. 트리 구조의 경우 명확하지만, 문법 구조에서는 빌딩 블록을 포함하는 의미 있는 경계를 정하는 것이 어렵다. 왜냐하면 복잡하게 얽혀있는 관계 때문에, 한 개체에서 의미 있는 빌딩 블록을 구성하던 요소가 다른 한 개체로 옮겨지는 순간 완전히 무의미하게 될 수 있기 때문이다.

mernik [2,4,5]은 연산자의 부족함을 보완하기 위해 기본적인 production 교환을 하는 교배 연산자에, symbol의 반복, empty symbol의 생성 연산 등을 추가하였다. 하지만 교배 연산자 자체는 production의 다양성을 주기 위한 역할을 제외하면, 빌딩블록의 생성 능력은 매우 떨어지는 것으로 보여, 본 연구에서는 위와 같은 연산자를 사용하지 않고, 강한 변이 연산만을 사용할 때와 다음 세 가지의 교배 범위가 다른 연산자들을 함께 사용할 때를 비교해 보았다.

a. Production 기반
   Production 기반은 각 production (LHS->RHS) 를 최소 단위로 하는 교배 연산
b. Nonterminal 기반
   한 nonterminal을 정하고 그 nonterminal의 production들을 교환하는 연산
c. Symbol 기반
   각 부모에서 서로 다른 nonterminal을 정하고 nonterminal의 모든 production을 교환하는 연산

변이 연산은, production내에서 하나의 요소를 다른 요소로 바꾸는 방식을 사용하였다.

2.1.3 적합도 평가

적합도는 주어진 문장들의 포함 여부를 바르게 판단한 경우의 합이다. 주어진 문장들은 긍정적인 문장뿐만 아니라 부정적인 문장들도 포함하고 있다. 사실, 유전 프로그래밍 시스템에서는, 특별히 일반화 수준에 대한 제한을 두지 않아, 긍정적인 문장만을 사용할 경우에는 , 매우 일반적인 문법을 생성한다. 예를 들면, 모든 단어를 포함하는 하나의 terminal과, 이 terminal과 자기 자신으로 구성된 production을 가지는 starting symbol만으로 이루어진 문법도 최적해로 나타날 수 있다.

2.1.4 초기 해집단 분포

초기 해집단은 임의적인 방식으로 채택하였다. 문법의 nonterminal 수와, terminal 수를 임의로 설정하고, production 수와, 그 내용 구성 역시 임의로 결정한다. 임의적이라고 해도, 적은 샘플을 사용하므로, 여러 번의 실험을 통해, 편향을 줄였다.

2.2 실 험

2.2.1 문제 설정

문제는 주어진 문장들을 잘 분류할 수 있는 성능 좋은 문맥 자유 문법으로 구성되어있는 파서를 찾아내는 것이다. 문장들은 Zhang의 연구에서 사용한, 드라마 friends의 자막을 한 문장씩 나눠 놓은 자료를 사용하였다.[3] 문장의 수는 5000문장을 사용하였다. 이 문장들은 모두 긍정적인 문장으로, 본 연구에서는 임의로 긍정적인 문장에서 위치를 정해 좌우의 단어들의 순서를 바꿔, 부정적인 문장을 생성하였다. 이 방식은 임의로 선택한 10문장에 대해서 20번 정도의 실행 결과 90%의 잘못된 문장을 발생시키는 것으로 나타났다.

Population size는 20으로, generation은 2000, mutation ratio 는 0.2로 설정하였다. 실험은 세 가지 연산자와 mutation만 사용하는 방법에 대해서 각각 20회씩 수행하였다. 문법 개체가 가질 수 있는 최대 nonterminal 수는 Nonterminal로 구성되는 production을 가지는 nonterminal용 10개와, Terminal을 production으로 가지는 nonterminal용 10개, 각각은 최대 한 nonterminal당 5개, 1000개씩의 production을 가질 수 있도록 설정하였다. 1000개의 production은 Terminal을 production으로 가지는 nonterminal들은 실험 자료에서 나타나는 모든 단어들을 분류할 수 있어야 기본적인 파싱이 되기 때문에, 단어 숫자 보다 큰 production들을 생성했다.

2.2.2. 실험 결과

표 1

	Mean	Std	Min	Max
Mutation Only	5641.55	40.13	5618.48	5664.62
a-crossover	5649.85	41.07	5626.24	5673.46
b-crossover	5658	44.80	5632.25	5683.76
c-crossover	5525.65	67.51	5486.84	5564.46

각 실행별 엘리트 개체들의 분포. Min, Max는 각각 t-test의 95% 신뢰구간의 Minimum, Maximum 값

결과를 보면, 각 방식은 거의 차이가 나지 않는다. 특히 c-crossover는 오히려 성능이 떨어지는데, b 와 c의 차이는 nonterminal과 production과의 연관성 유지에 있다. B가 그래도 조금 더 나은 결과를 보이는 것은, 하나의 production 단위로 보존이 되어야, building block이 파괴 되지 않기 때문이다. C의 경우는 거의 building block을 파괴하면서, 전혀 다른 production들을 생성함으로써, 엘리트의 분포도 다양해 높은 std 값을 가진다. Mutation만 쓰는 경우에 비해 a와 b와 같이 production들의 의미를 보존하는 정도가 높은 경우는 성능이 조금은 더 좋은 것으로 나타난다. 하지만 전체적으로 봤을 때 , 교배 연산자의 효과는 거의 없는 것으로 보인다. 잘해야 0.2%정도도 안되는 성능 향상이다.

3. 결 론

이 논문에서의 성과는 실제로 자연언어의 문법을 추론하는데, 극단적인 구조 변화만을 기반으로 하는 유전 프로그래밍 시스템이 어느 정도의 성능을 낼 수 있는지를 조사하는 것이었다. 실험 결과에 따르면 일반적인 유전 프로그래밍으로는 긍정적인 문장들을 포함하는 문법을 찾는 것은 매우 쉽지만, 부정적인 문장들까지 함께 분류해내는 것은 거의 불가능하다. 또한 문법 표현에서 일반적으로 생각하는 교배 연산자는 거의 의미가 없는 것으로 보여진다.

유전 프로그래밍에서 중요한 것은, 작지만, 점진적으로 해에 가까운 방향으로 다가가는 수렴성이다. 문법 표현 방식의 유전 프로그래밍에서는 단지 구조적인 정보만을 가지고, 합리적인 수렴 방향을 제시하기에는 구성요소간의 연관성이 너무 복잡하여, 적절한 방향을 제시하는 것이 어렵다.

앞으로의 연구에서는, 이런 수렴 방향을 확률적인 정보를 기반으로 하여 문법 구조를 변형시키는 것이 유용한지에 대해서 다루어 볼 것이다.

또한, 유전 프로그래밍 분야에서 모델 기반으로 학습을 하면서 구조적인 학습을 같이 하는 시스템과, 단계적인 해집단 생성 시스템의 성능을 체크해 보고, 이런 모델 기반 방식과 유전 프로그래밍에서의 구조 변화 정도가 학습 성능에 얼마나 영향을 미치는지, 어떤 방법으로, 합리적인 하위 구조 변화의 방향을 제시할 것인지에 대한 연구를 진행할 것이다.

4. 참고 문헌

1. Wyard, P., Representational Issues for Context-Free

Grammar Induction Using Genetic Algorithm in Proceedings of the 2nd International Colloquim on Grammatical Inference and Applications, Springer-Verlag Lecture Notes in
Artificial Intelligence, Vol 862, pp. 222-235, 1994.
2. Mernik, M., Gerli_ , G., Žumer, V., Bryant, B., Can a Parser be Generated from examples?, Proceedings of the ACM Symposium on Applied Computing, pp 1063 - 1067, 2003.
3. Self-assembling hypernetworks for cognitive learning of linguistic memory, B.-T Zhang and C.-H. Park, Proceedings of World Academy of Science, Engineering and Technology (WASET), p.134, 2008

4. F. Javed , B. R. Bryant , M. Črepinšek , M. Mernik , A. Sprague, Context-free grammar induction using genetic programming, Proceedings of the 42nd annual Southeast regional conference, April 02-03, 2004, Huntsville, Alabama
5. Crepinšek, M., Mernik, M., Bryant, B.R., Javed, F., and Sprague, A. Inferring Context-Free Grammars for Domain-Specific Languages, LDTA 2005,, pp. 64 -- 81, 2005