JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 2, pp. 43-51, February 2022
https://doi.org/10.9708/jksci.2022.27.02.043

Improved Method for Learning Context-Free Grammar using Tabular representation

Soon-Ho Jung*

*Professor, Dept. of Computer Engineering, Pukyong National University, Busan, Korea

[Abstract]

In this paper, we suggest the method to improve the existing method leaning context-free grammar(CFG) using tabular representation(TBL) as a chromosome of genetic algorithm in grammatical inference and show the more efficient experimental result. We have two improvements. The first is to improve the formula to reflect the learning evaluation of positive and negative examples at the same time for the fitness function. The second is to classify partitions corresponding to TBLs generated from positive learning examples according to the size of the learning string, proceed with the evolution process by class, and adjust the composition ratio according to the success rate to apply the learning method linked to survival in the next generation. These improvements provide better efficiency than the existing method by solving the complexity and difficulty in the crossover and generalization steps between several individuals according to the size of the learning examples. We experiment with the languages proposed in the existing method, and the results show a rather fast generation rate that takes fewer generations to complete learning with the same success rate than the existing method. In the future, this method can be tried for extended CYK, and furthermore, it suggests the possibility of being applied to more complex parsing tables.

▸ **Key words**: Tabular Representation, Context-free Grammar, Genetic Algorithm, Partitioning, Fitness

[요 약]

이 논문은 문법적 추론에서 유전자 알고리즘의 진화대상으로 테이블 표현(Tabular representation: TBL)을 이용한 문맥자유 문법(Context-free grammar: CFG)을 학습하는 기존의 방법을 개선하여 더 효율적인 결과를 얻은 그 방법과 실험 결과를 제시한다. 이 논문에서 소개하는 개선된 점은 두 가지로, 첫째는 적합도 함수를 긍정과 부정의 예들에 대한 학습 평가를 동시에 반영하도록 수식을 개선하고 둘째는 긍정적 학습 예들로부터 생성된 TBL들에 대응되는 파티션(partition)들을 학습 문자열의 크기별로 분류하여 부류별 진화 과정을 진행하며 그 성공률에 따라 구성 비율을 조정하여 다음세대에 생존에 연계하는 학습 방법을 적용한다. 이 개선점들은 학습 예들의 크기에 따른 TBL의 크기가 여러 개체들 사이의 교배와 일반화 단계에서 복잡성과 어려움을 해결하여 기존 방법보다도 좋은 효율을 제공한다. 이 연구는 기존 방법에서 제안된 언어들로 실험하고 그 결과는 기존 방법보다 같은 성공률을 갖는 상태에서 학습 완성의 평균 세대수가 적게 걸리는 다소 빠른 세대속도의 결과를 보여준다. 앞으로 이 방법은 확장된(extended) CYK에 시도할 수 있으며 더 나아가 좀 더 복잡한 파싱 테이블(parsing table)에도 적용할 가능성을 제시한다.

▸ **주제어**: 테이블 표현, 문맥-자유 문법, 유전자 알고리즘, 파티셔닝, 적합도

• First Author: Soon-Ho Jung, Corresponding Author: Soon-Ho Jung
*Soon-Ho Jung (snow@pknu.ac.kr), Dept. of Computer Engineering, Pukyong National University
• Received: 2022. 01. 27, Revised: 2022. 02. 23, Accepted: 2022. 02. 23.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

문법적 추론(Grammatical Inference: GI)은 학습 데이터들을 입력으로 사용하여 형식적인 언어들을 학습하는 절차를 의미한다. 이 문법적 추론은 기계학습과 컴퓨터의 학습 이론, 특별히 음성인식, 컴퓨터 생물학, XML 기술, 압축, 컴퓨터 언어학 등에서 자연언어 처리, 의미적 파싱, 자연언어의 이해 등의 구체 분야를 위해 많은 적용 기술들이 연구, 개발되고 있다.

문법적 추론은 유한개의 샘플 표현들을 점진적으로 학습해 나가는 컴퓨터 모델을 구축한다. 학습 데이터는 모델이 학습하는 과정에서 사용되는 긍정적인 예들과 부정적인 예들로 구성되는데 이 두 개의 예들을 통해서 모델의 인식의 범위를 정확하게 형성하고 새로이 제시되는 예들에 대하여서도 정확한 응답을 하도록 모델을 구축하는 것이다. 이러한 추론으로 결정적 유한오토마타 (Deterministic Finite Automata: DFA)는 증명이 가능한 수렴 알고리즘으로 학습될 수 있다[3]. 그러나 문맥 자유 문법(Context-free Grammar: CFG)을 학습하는 알고리즘은 현재에도 해결해야 할 중요한 문제 중에 하나로 되어 있다[1].

CFG를 추론하는 접근 방법으로는 CFG의 대안적 표현을 형식화하거나 CFG의 부분적 부류로 제안하거나 통계적 방법을 사용하기 위해 사용되어 왔다[4]. 최근에는 진화적인 방법을 사용하여 CFG 또는 동등한 PDA (Push-down Automata)를 추론하나 주로 인위적인 언어에 적용하고 있다[5]. 최근에는 유전적 알고리즘을 이용하여 유전되어 진화되는 대상으로 CFG의 생성규칙을 개체로 지정하여 진화함으로서 CFG를 추론을 수행하는 방법도 있다. 이 방법에서는 CNF(Chomsky Normal Form)로 표현된 CFG를 추론하기 위한 TBL(tabular representation)과 알고리즘을 이용하여 CYK(Coke-Younger-Kasami) 파싱 알고리즘의 파싱 테이블과 유사한 테이블 표현을 학습 예로 사용하여 CFG 학습을 수행한다. 이 TBL은 다항식 차수의 테이블에서 기하급수적 수의 모든 가능한 문법적 구조들을 동적 프로그래밍으로을 사용하여 효과적으로 저장하여 처리한다[3]. 이 표현을 사용함으로써 CFG 학습의 문제는 비단말 노드(nonterminal)들의 분할하는 문제로 축소된다. TBL 알고리즘은 유전자 알고리즘을 사용하여 NP-hard의 파티셔닝(partitioning) 문제를 해결한다.

이 논문은 기존의 방법을 2가지 점을 개선하여 적용하여 실험하였다. 첫째는 유전자 알고리즘의 적합도 함수의 개선이고 둘째는 유전자 학습과정에서 입력 문자열 크기별로 분류하여 경쟁 후 선택, 조정하는 방법을 개선하여 앞선 방법보다도 학습능력이 우수함을 보여준다.

이 논문은 2절에서는 사전 소개를 통해 앞선 TBL에 대하여 설명하고 3장에서 개선 방법에 대하여 기술하고 4장에서는 개선 방법의 구현 및 실험을 보여주며 4장에서는 개선점 및 실험적 결론을 언급하고 미래 연구와 방향에 대하여 제시한다.

II. Related Work

1. Context-free grammar parsing

문맥자유 문법(context-free grammar: CFG)는 4개 항목 G=(Σ,V,R,S) 으로 이뤄진다. 여기서 Σ는 단말(terminal) 심볼들의 유한 집합, V는 Σ∩R=∅인 비단말(nonterminal) 심볼들의 유한 집합, R은 A→α 형태의 생성규칙들의 집합인데 A∈V, α∈(V∪Σ)+ 이고 S∈V는 특별한 비단말 심볼로 시작 심볼이라고 한다. 모든 유도는 시작 심볼 S로 시작하여 나타내고 이 유도는 CFG의 생성규칙을 적용하여 (V∪Σ)+ 에 포함된 심볼들의 순열들 즉, 스트링을 나타낸다. 유도과정에 나타난 스트링중 비단말 심볼은 생성규칙의 오른편의 표현으로 대체된다. CFG G에 의해 생성되는 언어는 $L(G)=\{x\in\Sigma^* | S\overset{*}{\underset{G}{\Rightarrow}}x\}$라고 표현하고 x는 CFG G의 생성규칙을 사용하여 유도된 유한개의 단말 심볼들의 스트링으로 단어라고 한다.

CFG G=(Σ,V,R,S)의 촘스키 정상 형태(Chomsky Normal Form: CNF)는 R의 생성규칙이 A→AB 또는 A→a의 형태이고 A,B,C∈V, a∈Σ이다.

CFG G는 특정한 L(G)를 나타내고 L(G)에 속하는 단어들을 인식하는데 사용할 수 있다. 임의의 단어들이 L(G)에 속하는지 여부를 판별하기 위하여 기존의 방식으로 여러 분류의 파싱(parsing algorithm)을 필요로 한다.

모든 CFG G는 대응되는 CNF 형태의 CFG G′로 변환될 수 있으며 같은 스트링의 집합을 정확하게 인식한다[7].

2. Coke-Younger-Kasami(CYK) algorithm

CFG G를 CNF로 표현하고 인식하고자 할 때 파싱 알고리즘으로 CYK 알고리즘을 사용하여 해결할 수 있다. 이 알고리즘은 멤버 문제(membership problem)를 상향식(bottom-up) 동적 프로그래밍 기법을 사용하여 다항식 시간(polynomial time)에 해결한다. 이 CYK 알고리즘은 CNF 형태의 CFG G를 나타내며 인식할 수 있다. CNF는 각 생성규칙의 오른편(right-hand side : RHS)이 2개의 비단말들이나 1개의 단말로 확장된다. 입력 문자열

w=abaab에 대하여 동작하는 CYK 파싱 알고리즘은 Table 1과 같다.

Table 1. CYK parsing algorithm

```
• Input:
    w = a1, ..., an : input string of CFG G
    P[1,n] : array of set of nonterminals. Initally 0.
• Output:
    Successful if P[1,n] includes a start symbol.

for i = 1... n
    for every A -> ai, add A to P[i,1].
for j = 2 ... n
    for i = 1 ... n-j+1
        for k = 1 ... j-1
            for all A -> T1T2 where T1 in P[i,k] and
                                    T2 in P[i+k, j-k]
                add A to P[i,j].
```

이 CYK 알고리즘에 의해 생성된 CYK 파싱 테이블은 Fig. 1와 같고 결과적으로 p[1,5]에 시작 심볼이 포함되므로 파싱이 성공되어 CFG G의 단어로 인식됨을 보여준다.

j=5	*A*				
j=4	*A*	*B*			
j=3	*A*	*B*	*A*		
j=2	*A*	*B*	*A*	*A*	
j=1	*A*	*B*	*A*	*A*	*B*
	i=1	*i=2*	*i=3*	*i=4*	*i=5*
	a	*b*	*a*	*a*	*b*

Fig. 1. CYK parse table of input string 'abaab'

3. Tabular representation for CFG induction

CFG을 유도하려는 목표는 학습 데이터 집합을 사용하여 이 집합의 모든 원소들에 일치하는 최소의 생성규칙과 비단말들의 문법적 구조의 집합을 찾는 것이다. 이 과정에서 학습 예제들의 단어 길이에 따라 기하급수적으로 문법적 구조들의 개수가 많아지게 되기 때문에 문법적 구조를 찾는 것은 대단히 어렵고 복잡한 문제가 된다. 이러한 복잡한 문제를 회피하기 위하여 유전자 알고리즘을 적용하고자 하는 것이다.

각 입력 문자열에 대응하는 CYK 파싱 테이블을 각 개체로 정하고 유전자 알고리즘을 적용하여 수렴하는 해를 구하는 방법이 사카키바라(Sakakibara)에 의해 테이블 표현(tabluar representation: TBL)과 알고리즘이 소개되었다[6]. 이 방법으로 한 학습 문자열에 대한 기본적인 테이블 표현이 구성된다. 예를 들면 aabb에 대한 문법적 구조 표현과 테이블 표현이 Fig. 2와 같이 나타내는데 (a)는 문자열에 대한 한 CFG G의 5가지의 유도과정을 보여주고 이 유도과정에서 필요한 총 비단말들의 개수는 (b)에서 같이 14개이고 X_{141}, X_{142}, X_{143}, X_{131}, X_{132}, X_{231}, X_{232}, X_{121}, X_{221}, X_{321}, X_{111}, X_{211}, X_{311}, X_{411}로 구성되는 TBL을 보여준다.

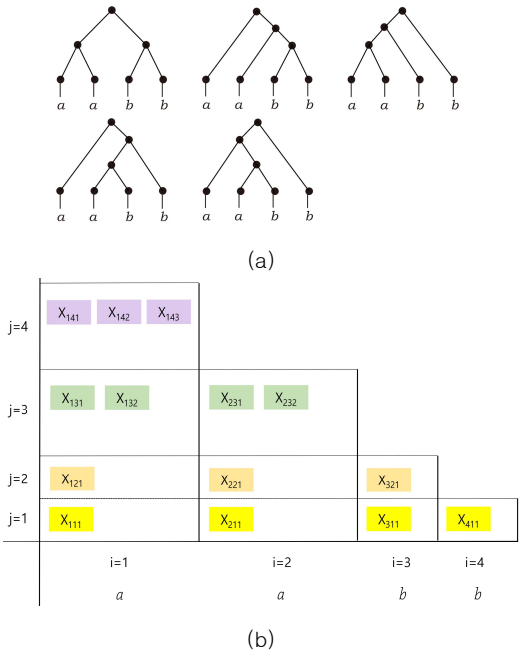


Fig. 2. (a) all possible representations of grammatical structures (b) nonterminal symbols of generated TBL for input string 'aabb'

이 TBL의 비단말들을 블록 크기 1이고 블록의 개수가 비단말들의 개수가 되는 초기 파티션을 구성하여 시작한다. 비단말들의 전체 개수들을 줄이기 위하여 파티션들을 서로 다른 부분들은 합치는 등 결합하여 생성된 다양한 파티션들의 TBL들에 학습 예제를 인식하여 성공하는 TBL은 생존하고 실패하는 것은 버린다. 여기서 파티션이 최소가 되는 문법을 찾는 문제는 어려운 NP-hard의 문제이므로 이 부분을 유전자 알고리즘으로 해결한다. 유전자 알고리즘의 절차로 선택, 교배, 돌연변이 등을 적용하며 적절한 적합도의 값에 따라 최소의 비단말들의 개수와 대다수의 학습 예제들과 일치하는 우수한 TBL들을 다음 세대에 생존하도록 하여 진화하며 진행한다.

3.1 Partitioning the set of nonterminals

TBL을 구성하는 알고리즘은 다항식(polynomial) 크기의 CYK 테이블과 유사하며 기하급수적 개수의 모든 가능한 문법적 구조들을 저장하는데 동적 프로그래밍 방법을 사용한다. TBL의 구조를 보면, 입력 문자열 $w=a_1...a_n$에

대하여 TBL은 삼각형 테이블 T(w)이다. 이 테이블의 원소 $t_{i,j}(1 \le i \le n$이고 $2 \le j \le n-i+1)$는 j-1개의 비단말들 $\{X_{i,j,k_1}, \cdots, X_{i,j,k_{j-1}}\}$을 갖는다. T(w)은 입력 문자열에 대한 CNF형태의 원시적인 CFG G=(Σ,V,R,S) 생성하는데 사용된다. 여기서,

$$V = \{X_{i,j,k} | 1 \le i \le n, 1 \le j \le n-i+1, 1 \le k \le j\} \cup \{X_{i,1,1} | 1 \le i \le n\}$$

$$R = \{X_{i,j,k} \rightarrow X_{i,k,l_1} X_{i+k,j-k,l_2} | 1 \le i \le n, \; 1 \le j \le n-i+1, \; 1 \le k \le j, 1 \le l_1 \le k, 1 \le l_2 \le j-k\} \cup \{X_{i,1,1} \rightarrow a_i | 1 \le i \le n\} \cup \{S \rightarrow X_{1,n,k} | 1 \le k \le n-1\}$$

이다.

3.2 TBL algorithm

Table 2의 TBL 알고리즘에서 CFG를 학습하는 것은 1) 긍정적(positive) 예제들 중 한 샘플에 대하여 TBL T(w)을 구성하고 2) 각 w∈U에 대하여 G(T(w))와 이 CFG의 비단말들을 개명(rename)하여 $G^w(T(w))$를 유도한다. 3) 모든 샘플들에 대하여 유도된 CFG들을 모두 합하여 $G(T(U^+))$를 생성한다. 4) $G(T(U^+))/\pi$가 U의 모두 원소들에 일치하는 최소의 파티션 π를 찾는다.

Table 2. TBL algorithm

- Input:
  - U is a set of all learning examples,
    $U = U^+ \cup U^-$,
    where U^+ is a set of positive examples and U^- is a set of negative examples.
- Output:
  - CFG $G(T(U^+))/\pi_*$

1. For each w∈U^+, construct T(w).
2. For each w∈U^+,
   induce primitive G(T(w))와 renamed $G^w(T(w))$.
3. Generate union of all primitive CFGs,
   $G(T(U^+)) = \cup_{w \in U^+} G^w(T(w))$.
4. Find minimal partition π_*,
   where $G(T(U^+))/\pi_*$ is identical to U.

3.3 Genetic algorithm

위 TBL알고리즘은 유전자 연산자들의 교배, 돌연변이 기능을 가지고 있는 그룹-번호 부호화(group-number encoding)[8] 방법을 채용한다. 파티션은 정수의 문자열 p = (i_1, i_2, ... ,i_n)으로 표현하는데 여기서 $i_j \in \{1,2, \ldots, k\}$는 원소 j에 할당된 블록의 번호이다. 예를 들면 문자열 p = (1 3 4 3 6 3 3 4 5 2 6 4)은 파티션 π_p: {1}, {10}, {2,4,6,7}, {3,8,12}, {9},{ 5,1} 을 나타낸다. 이것에 적용하는 유전자 연산자들다음과 다음과 같다.

• 구조적 교배

무작위로 선택된 부모 파티션들의 임의의 블록들을 합한다.

• 구조적 돌연변이

문자열의 한 정수 값을 다른 문자열의 값으로 대체한다.

• 특수 제거 연산

특별한 값 -1으로 문자열에 한 개 정수값으로 치환한다. 이것은 이 비단말이 더 이상 사용되지 않는 것을 의미한다.

• 적합도 함수

파티션들 중에서 유전자 알고리즘의 선택과정으로 최소의 파티션을 찾기 위해 적용하는 적합도 함수는 다음과 같다.

$$f_1(p) = \frac{|\{w \in U^+ \; G(T(U^+))/\pi_p\}|}{|U^+|} \quad (1)$$

$$f_2(p) = \frac{1}{|\pi_p|} \quad (2)$$

$$f_3(p) = \frac{|\{w \in U^- \; G(T(U^-))/\pi_p\}|}{|U^-|} \quad (3)$$

$$f(p) = \begin{cases} -(C_3 \cdot f_3(p) + C_4 \cdot (1 - f_1(p))) & \text{if any negative example is generated} \\ \dfrac{C_1 \cdot f_1(p) + C_2 \cdot f_2(p)}{C_1 + C_2} & \text{otherwise} \end{cases} \quad (4)$$

파티션 p에 대하여 (1) $f_1(p)$는 긍정적(positive) 예제들에 대한 인식률, (2) $f_2(p)$는 파티션들의 개수의 역수, 즉 파티션들의 개수가 클수록 0에 근접한다. (3) $f_3(p)$는 부정적(negative) 예제들에 의해 발생하는 오류율, 이 값은 작을수록 유리하다. (4) 마지막 함수 $f(p)$는 위의 모든 함수를 고려하여 부정적 예들이 적용될 때 계산되는 식과 긍정적 예들이 적용될 때의 계산식이 분리되어 있다. 부정의 예들이 인식이 되는 오류가 발생하면 위 식이 적용되어 음수의 값을 가진다. 이 함수의 값은 [-1,1] 범위의 값을 가지게 되고 [0,1]로 정규화가 가능하다.

III. The Proposed Scheme

기존의 TBL과 그 알고리즘의 기본 틀은 그대로 이용하고 유전자 알고리즘의 적합도 함수(fitness function)의 수식을 수정하고 유전자 학습과정의 운영방식에서 입력 문자열의 크기별 부류에 따른 진화과정의 개선 방법을 설명한다.

1. Relation of nonterminals in TBL

앞서 설명한 바와 같이 입력 문자열 w=$a_1 \ldots a_n$에 대하여 TBL은 삼각형 테이블 T(w)이다. 예를 들면 입력 문자열이 w=aabb이라 할 때 4개의 입력 문자의 길이 맞춰 4 × 4 삼각행렬로 구성된 TBL의 모습은 Fig. 3과 같다. 일반적으로 원소 $t_{i,j} \in$T(w)는 j-1개의 비단말들 $\{X_{i,j,k_1}, \ldots, X_{i,j,k_{j-1}}\}$을 가진다. 이들 비단말들의 관계는 CNF의 생성규칙 형태로서 생성규칙의 오른편 표현(right-hand side: RHS)과 왼편 표현(left-hand side: LHS)의 관계를 Fig. 3에서 같은 종류의 화살표시 2개로 연결 관계를 나타내고 $X_{141} \rightarrow X_{111}X_{231}$ 또는 $X_{141} \rightarrow X_{111}X_{232}$ 등의 생성 규칙들의 연결 관계가 화살표시로 나타낸다.

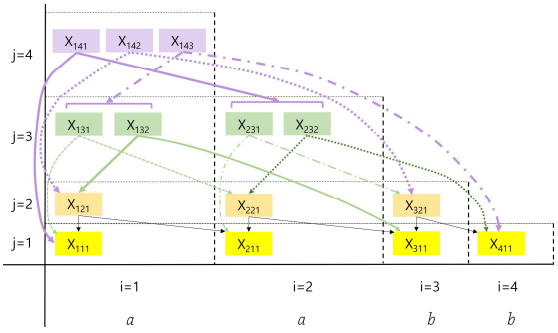


Fig. 3. Relations of nonterminal symbols and its production rules

T(w)에 의해 나타내는 CNF형태의 원시적인 CFG $G = (\Sigma, V, R, S)$의 생성규칙들의 수는 입력 문자열의 길이에 따라 기하급수적으로 증가함을 알 수 있다. 이 생성규칙들의 비단말들로 원시 파티션을 구성한다. 위 예의 TBL을 근거로 하여 파티션 p를 형성하면 초기에 블록 크기가 1이고 블록 개수는 14인 π_p는 다음과 같다.

π_p: {X_{141}}, {X_{142}}, {X_{143}}, {X_{131}}, ... , {X_{321}}, {X_{111}}, {X_{211}}, {X_{311}}, {X_{411}}.

2. Partitioning of TBLs

입력 문자열 w에 대응하는 삼각형 배열 TBL T(w)의 요소 t(i,j)들의 총 개수는 입력 문자열의 수가 |w|=n 이라고 하면 최소 n(n+1)/2가 된다. 각 t(i,j)에는 적어도 1개 이상의 비단말들이 있고 생성규칙에서 RHS가 상이함에 따라 같은 LHS를 다르게 분류하여 차례대로 인덱스(index)번호를 부가하여 비단말 명칭을 지정한다. 따라서 원시적 T(w)의 초기 비단말들의 개수는 TBL의 요소들의 수 이상이 되며 길이의 증가에 따라 기하급수적으로 증가한다. 각 입력 문자열의 TBL 파티션의 초기 블록 크기는 1이고 블록의 수는 최소 n(n+1)/2 이상으로 시작한다.

같은 w에 대하여 원시 파티션를 근거로 하여 가능한 많은 새로운 파티션들을 생산하는데 이 과정에서 무작위 확률로 새로운 파티션의 블록 수와 크기를 임의로 변형하여 한 세대의 필요한 개체수를 생성한다. s개의 학습 문자열들을 w_1, w_2, ..., w_s 이라고 하면, 각 문자열 w_i에 의해 초기에 생성되는 파티션들의 개수는 총 개체수 중에 $|w_i|/(|w_1|^2+|w_2|^2+\ldots+|w_s|^2)$의 비율로 파티션들을 생성하고 s개의 문자열 별로 관리한다. 즉, s개의 각 TBL에 관련된 파티션들을 분류하여 이 부류 내의 파티션들에 유전자 알고리즘의 동작을 적용한다. 선택(selection)과정에서 s개의 부류에 속한 모든 파티션들의 적합도 값에 따라 파티션들을 선정하여 각 부류에 되돌리고 각 부류에서 유전자 연산 교배(crossover), 돌연변이(mutation), 특별 삭제(special deletion)을 실행한다. 이 과정은 기존의 방법과 동일하다.

3. Improved TBL algorithm

이 개선된 TBL알고리즘에는 유전자 연산들 중에서 개선된 내용들과 적합도 함수의 내용을 표현하고 이 알고리즘의 전체 순서도를 표현한다.

3.1 Genetic operators

각 입력 문자열로 생성된 TBL별(여기부터 부류별로 칭함)로 원시 파티션이 생성되고 이것으로부터 파생적인 파티션들을 생성되어서 각 부류별로 파티션들에 유전자 진화과정이 적용이 된다.

1) 선택(selection)과정 : 모든 파티션들에 적합도 함수를 적용하여 적합도 값을 계산하고 부류의 구분 없이 부모 파티션을 경쟁적으로 선택하여 각 부류마다 다음세대를 위해 생존하도록 한다.

2) 일반화(generalization) 과정: 선택과정에서 예비 부모들을 형성하고 각 부류마다 적정한 개체수가 각 부류의 성공률에 비례하여 부족하지 않도록 보충하여 유지한다.

3) 교배(crossover) 과정 : 선택과정에서 선정된 부모 partition들에 대하여 기존 방법과 같은 방식으로 교배를 실시한다.

4) 돌연변이(mutation) 과정 : 임의의 작은 확률로 partition의 무작위한 위치에 값이 변화되도록 한다. 기존 방법과 같다.

5) 특별 삭제 연산(special delete operator) : 기존 방법과 같다.

3.2 Modified fitness function

기존의 방법에서 사용하던 적합도 함수에 포함되는 내부 함수들 중에서 $f_2(p)$의 값을 파티션의 블록의 수를 상대적인 크기로 평가하도록 보정하고 최종 적합도 함수 $f(p)$는 긍정 예들로 기인한 값과 부정 예들로 기인한 값들을 합성하여 수치를 얻도록 지정하였다. $f(p)$의 값의 범위는 [-1: 1] 이다. 여기서 기본(default)으로 $C_1 = C_2 = C_3 = C_4 = 1$로 정한다.

$$f_1(p)=\frac{\left|\{w\in U^+ \; G(\mathcal{I}(U^+))/\pi_p\}\right|}{|U^+|} \quad (1)$$

$$f_2(p)=\frac{1}{(|\pi_p|)} \quad (2)$$

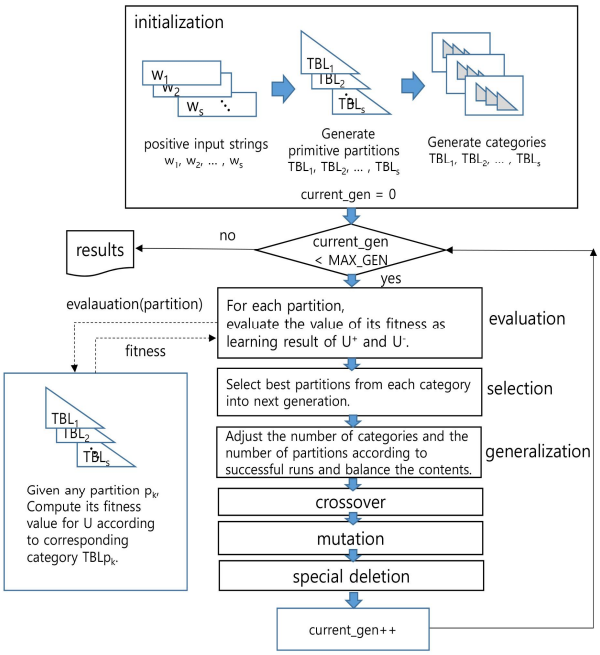
$$f_3(p)=\frac{\left|\{w\in U^- \; G(\mathcal{I}(U^-))/\pi_p\}\right|}{|U^-|} \quad (3)$$

$$f(p)=\frac{C_1\cdot f_1(p)+C_2\cdot f_2(p)}{C_1+C_2} - \frac{C_3\cdot f_3(p)+C_4\cdot (1-f_1(p))}{C_3+C_4} \quad (4)$$

3.3 Improved TBL algorithm

위의 유전자 연산자들과 수정된 적합도 함수들을 포함하여 긍정적 예들과 부정적 예들을 문자열 크기별로 분류하여 모두 학습한 결과로 다음세대의 개체를 경쟁적으로 선택하여 진화하는 개선된 TBL알고리즘을 구성한 순서도로 나타내면 Table 3. 과 같다. 초기화에서는 입력 문자열로부터 TBL이 구성이 되고 이에 대응되는 원시 파티션으로 표현되고 파티셔닝(partitioning)을 통하여 파생된 많은 파티션들의 개체들이 생성되어 같은 입력 문자열로부터 같은 카테고리(category)로 분류되어 유전자 알고리즘에 의해 진화되는 과정을 준비하게 된다. 이러한 준비가 모든 입력 문자열들에 의해 동일하게 적용된다. 이어서 모든 문자열에 대한 개체들이 진화과정을 순환하며 진행하게 된다. 진화 과정의 순번은 current_gen 변수로 관리되고 있고 단순하게 MAX_GEN으로 지정된 만큼만 진화를 진행한다. 유전자 과정에서는 평가에서는 적합도 함수를 호출하여 각 개체, 즉 파티션의 경쟁력을 측정하여 선택에서는 경쟁력이 우수한 개체를 선정하여 후보로 놓고 일반화에서 각 카테고리 마다의 성공률을 반영하여 조정해서 구성한다. 이렇게 구성된 세대들의 개체들은 각 카테고리별로 교배, 돌연변이, 특별 제거 등을 실행하는데 이는 앞서 설명한 바와 같이 실행한다. 이와 같은 매세대의 진화과정이 진행되는데 지정된 MAX_GEN까지 수행하여 그 성공률을 측정한다. 이 때 최대 세대수에 이르기 전에 허용된 기간 동안 변화가 없을 경우에는 진화를 중단하게 되고 완전 학습에 실패한 것으로 인정하도록 수정가능하다.

Table 3. Improved TBL algorithm



IV. Implementation and experimental results

1. Implementation

이 실험은 Window 11 환경에서 C++ 프로그램으로 구현되어 실험되었다. 기존의 방법과 이 논문에서 제시한 개선된 TBL알고리즘을 실험 결과로 비교하기 위해서 사용된 CFG 문법은 기존의 방법에서 기술할 때 사용한 두 개의 언어 $L_1=\{a^n b^n c^m \mid n,m\geq 1\}$ 과 $L_2=\{ac^n \cup bc^n \mid n\geq 1\}$를 채택하여 실험한다. 두 개의 언어를 학습하는 동안에 학습의 인식률을 더 높이기 위한 부분적 부분적인 정보의 제공이 없이 단순한 문자열들을 입력으로 실험한다.

2. Experimental results

매 실험에서 10개의 성공적인 실행들(runs)를 받을 때까지의 실험횟수를 계산하고 한 개의 성공적인 실행은 긍정적인 예들과 부정적인 예들이 승인되며 1100세대 동안 실행되어 인식이 성공이 되는 것을 의미한다. 아래의 Table에서 나타나는 성공률은 (성공한 실행의 수)/(전체 실행의 수)를 나타낸다. 이 실험한 내용은 두 가지 언어에

대한 실험내용으로 첫 번째는 $L_1=\{a^nb^nc^m \mid n,m\geq 1\}$ 이고 두 번째는 $L_2=\{ac^n \cup bc^n \mid n\geq 1\}$에 대한 실험이다. 이 들 실험에 적용한 알고리즘의 몇 가지 변화 가능한 요소들의 내용을 정리하면 전체 집단(population)은 100개의 개체 수가 사용하고 알고리즘의 적합도 함수에서 계수들은 $C_1 = C_2 = C_3 = C_4 = 1$로 정하고 입력 문자열의 길이는 긍정의 예일 경우는 최대 6자까지로 하고 부정의 예일 경우는 최대 15자까지로 정하여 실험하였다.

2.1 Experiment for $L_1=\{a^nb^nc^m \mid n,m\geq 1\}$

첫 번째 언어에 대한 실험의 결과는 다음과 같은 결과의 문법을 얻게 되었다.

CFG L_1 = (N, {a,b,c}, P, S), where

N = { <111>, <211>, <311>, <212>, <216>, <340> }

P = { S→<340>,

<340>→<340><311>, <340>→<216><311>

<216>→<111><211>, <216>→<212><211>,

<212>→<212><211>

<111>→a, <211>→b, <311>→c }

이 실험의 결과에 대한 통계는 Table 1에서 나타난 바와 같이 긍정의 예들과 부정의 예들을 합한 입력된 문자열들에 대하여 두 가지 방법들이 결과적으로 100% 성공률을 보였고 100% 성공률에 도달할 때까지의 평균 세대수를 보면 기존의 방법은 770세대이나 개선된 방법은 620세대에 달성되는 것을 볼 수 있다. 이로써 개선된 방법인 100% 성공률에 도달할 때까지 소모되는 세대수가 짧음을 다음 Table 4.에서 확인할 수 있다. 이 Table은 기존의 방법과 개선된 방법을 비교하고 최종 성공률(Successful runs)을 Succ. runs으로 평균 실험세대수(Average generations)는 Avg. gen.으로 표기한다.

이 Table 4의 통계적 자료를 제공하는 이 언어에 대한 실험에서 두 방법의 각각의 세대별 적합도 함수 값과 수렴되는 최대값의 모습을 그래프로 나타내면 Fig. 4와 같다.

Table 4. Learning result of examples for $L_1=\{a^nb^nc^m \mid n,m\geq 1\}$

existing method		improved method	
Succ. runs	Avg. gen.	Succ. runs	Avg. gen.
100	770	100	620

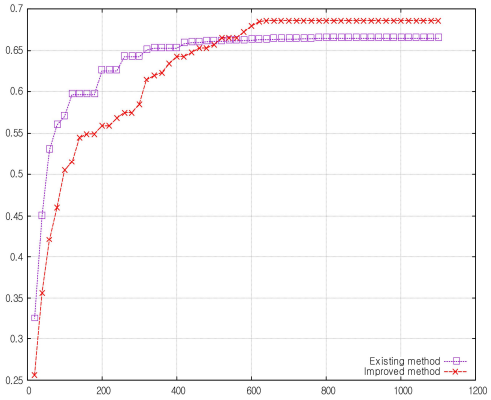


Fig. 4. Comparison of two methods for $L_1=\{a^nb^nc^m \mid n,m\geq 1\}$

2.2 Experiment for $L_2=\{ac^n \cup bc^n \mid n\geq 1\}$

두 번째 언어에 대한 실험의 결과는 다음과 같은 결과의 문법을 얻게 되었다.

CFG L_2 = (N, {a,b,c}, P, S), where

N = { <111>, <211>, <311>, <212>, <313> }

P = { S→<313>,

<313>→<111><212>, <313>→<211><212>

<313>→<111><311>, <313>→<211><311>,

<212>→<212><311>, <212>→<312><311>

<111>→a, <211>→b, <311>→c }

이 실험의 결과는 Table 5에서 나타난 바와 같이 긍정의 예들이나 부정의 예들에 대하여 모두 결과적으로 100% 성공률을 보이고 두 방법의 100% 성공률에 도달할 때까지의 평균 세대수를 보면 기존의 방법은 950세대이나 개선된 방법은 760세대에 달성되는 것을 Table 2에서 볼 수 있다. 이 것은 개선된 방법인 100% 성공률에 도달할 때까지 소모되는 세대수가 190세대 정도 짧음을 보여준다.

이 Table 5의 통계적 자료의 근거는 이 언어에 대해 두 방법으로 각각 실험한 세대별 적합도 함수 값과 수렴되는 최대값을 그래프로 나타낸 모습은 Fig. 5와 같다.

Table 5. Learning result of examples for $L_2=\{ac^n \cup bc^n \mid n\geq 1\}$

existing method		improved method	
Succ. runs	Avg. gen.	Succ. runs	Avg. gen.
100	950	100	760

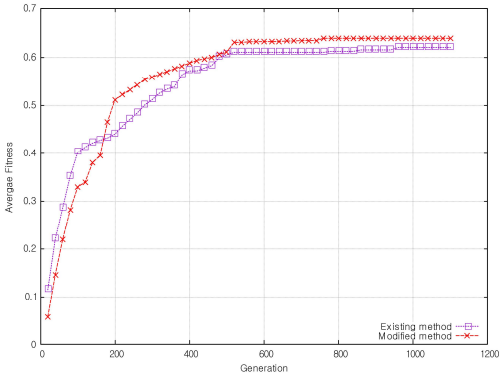


Fig. 5. Comparison of two methods for $L_2=\{ac^n \cup bc^n \mid n \geq 1\}$

위의 두 언어에 대한 기존의 방법과 개선된 방법의 실험 결과를 비교한 결과 개선된 방법이 같은 성공률 100%을 달성하는데 비교적 적은 세대수가 소모된다는 것을 확인할 수 있다.

V. Conclusions

이 논문에서 기존의 테이블 표현(Tabular representation: TBL)을 이용하여 문맥자유 문법(context-free grammar)을 학습하는 방법을 첫째 적합도 함수를 긍정과 부정의 예들의 학습 결과를 동시에 반영하도록 변경하여 적용하고 둘째 문자열 크기별 부류 진화 학습을 통하여 기존의 방법보다 100% 성공률을 달성하면서 학습에 소요되는 세대수를 줄이는 실험결과를 보였다.

이 연구는 문법적 추론 및 학습에 대한 연구로서 많이 고려되는 유전자 알고리즘의 진화 대상의 개체로서 컴파일러 제작과정의 파싱처리를 단순화한 CYK방법을 입력 문자열에 일반화하여 형식화한 TBL을 사용하고 유전자 알고리즘의 교배, 돌연변이, 특별 연산, 적합도 함수 등의 적용하는 기존의 방법론에서 학습 능력을 제고하는 요소들을 제안하고 실험한 것이다. 앞으로 이 방법론을 좀 더 확장적인 확장된(extended) CYK에 시도할 수 있으며 또 다른 형태의 좀 더 복잡한 파싱 테이블에도 적용할 가능성을 제시한다.

ACKNOWLEDGEMENT

This work was supported by a Research Grant of PuKyong National University(2020).

REFERENCES

[1] C. de la Higuera, “Current Trends in Grammatical Inference” in: F. Ferri, J. Inestam A. Admin, p.Puil (Eds.), Advances in Pattern Recognition. Joint IAPR International Workshops SSPR+SPR’2000, LNCS 1876, Springer Verlag, Berlin, 2000.

[2] E. Tanaka, “Theoretical aspects of synrtactic pattern recognition” Pattern Recognition Vol. 28, No. 7, pp. 1053-1061, 1995.

[3] O. Unold, M. Jaworski, “Learning context-free grammar using improved tabular representation” Applied Soft Computing, Vol. 10, No. 1, pp. 44-52, 2010.

[4] L. Lee, Learning of context-free languages: “A Survey of the Literature” Tech. Re TR-12-96, Havard University, Cambridge, Massachusetts, 1996.

[5] O. Unold, “Context-free grammar induction with grammar-based classifier system” Archives of Control Science Vol. 15, No. 4, pp. 681-690, 2005.

[6] Y. Sakakibara, “Learning context-free grammars using tabular representations” Pattern Recognition Vol. 38, pp. 1372-1383, 2005.

[7] R. C. Dharmik, R.S. Bhanuse, A.D. Gaikwad, “Ambiguity of Context Free Grammar using the CYK algorithm” 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare, Feb. 2016.

[8] Z. Michalewicz, “Genetic Algorithms + Data Structures = Evolution Programs” Springer, 1999.

[9] F. M. Massimo, G. Satta, G. Cristini, “Parsing with CYK over Distributed Representations” Algorithms Vol. 13, No. 10, pp. 262, 2020.

[10] C.G. Nevill-Manning, I.H. Witten, “Identifying hierachical structure in sequences: a linear-time algorithm” Journal of Artificial Intelligence Ressearch, pp. 67-82, Jul. 1997.

[11] D. Hrnčič, M. Mernik, B. R. Bryant, F. Javed, “A memetic grammar inference algorithm for language learning” Applied Soft Computing, pp. 1006-1020, Dec. 2012.

[12] M. Alomari, F. Li, D. C. Hogg, A. G. Cohn, “Online perceptual learning and natural language acquisition for autonomous robots” Artificial Intelligence26, Vol. 303 November 2021.

[13] A.v. Aho, J.D. Ullman, The Theory of Parsing and Compiling, vol. I: Parsing, Pentice-Hall, Englewood Cliffs, NJ, 1972.

[14] F. Denis, U. Tarmo, “Certified CYK parsing of context-free languages” Journal of Logical and Algebraic Methods in Programming September-November, Vol. 83, No. 5-6, pp. 459-468, 2014.

[15] P. Adriaans, H. Ferrnau, M. Van. Zaannen, “Grammatical Inference: Algorithms and Applications” Proceedings of ICGI 00, LNAI 2484, SPringer-Verlag, Belrin, Heidelberg, 2002.

[16] A. Kádár, S. Prince, “Tutorial #15: Parsing I: context-free grammars and the CYK algorithm” https://www.borealisai.com /blog/tutorial-15-parsing, Jun. 14. 2021.

[17] J. Heinz, Colin de la Higuera, M. van. Zaanen, “Grammatical Inference for Computational Linguistics” Grammatical Inference for Computational Linguistics, 2015.

[18] Ž. Kovačević, M. Mernik, M. Ravber, M. Črepinšek, “From Grammar Inference to Semantic Inference—An Evolutionary Approach” Mathematics, Vol. 8, Iss. 816, pp. 816, 2020.

[19] J.A. Laxminarayana, T. Nagaraja, “Inference of a subclass of context-free grammars using positive samples” ECML/PKDD 2003 Workshop on Learning Context-Free Grammars, pp. 29-40, 2003.

[20] A. Dubey, P. Jalotes, S. Aggarwal, “Learning context-free grammar rules from a set of program” IET Software, pp. 223-240, Feb. 2008.

Authors

Soon-Ho Jung received the B.S. degree in Mathematics Education from Seoul National University in 1982 and M.S. and Ph.D. degrees in Computer Science from KAIST. Dr. Jung joined the faculty of the Department of Engineering at Pukyong National University, Busan, Korea, in 1987. He is currently a Professor in the Department of Computer Engineering, Pukyong National University. He is interested in Embedded Intelligent System, Computer Security and Machine Learning.