

題名	文法学習支援システム - <i>LESSON</i>
目的	このシステムの目的は、帰納的推論メカニズムに基づいた構文的知識(文法)の獲得を支援することにある
概要及び特徴	<p>この文法学習支援システムは次の特徴を持つ</p> <ol style="list-style-type: none"> <li>1. 構造的データからの学習</li> <li>2. 正の例だけからの学習</li> <li>3. 多項式時間学習</li> </ol> <p>さらにこのシステムは、学習された文法からパーザを自動合成する機能を持つ</p>
構成	<pre> graph TD     LD[学習データ (文+木構造)] --&gt; WT[文法推論部]     WT --&gt; G[文法]     G --&gt; T[教師]     G --&gt; PG[パーザ生成部]     PG --&gt; P[パーザ]     P --&gt; CT[構文解析木]     TD[テストデータ] --&gt; P   </pre> <p style="text-align: center;"><i>LESSON</i></p>

このシステムの目的は、帰納的推論メカニズムに基づいた構文的知識(文法)の獲得を支援することにある。我々の目標は、ボトムアップパーザに適した文法を構築するための文法推論のメカニズムを用いたシステムを提供することである。我々は構造的な正の例だけから文脈自由文法を学習するシステムを提案する。文脈自由文法の構造的例とは、その文法のラベルなしの構文解析木、すなわち構文解析木の形、である。このように学習システムへの入力、有限個の構文解析木の形となる。我々のシステムは、出力される文法は意図した構造を持ち、そして正の例だけから効率良く文法を学習するという好ましい特徴を持つ。

(1) 学習された文法は教師であるユーザが意図したとおりの構造を持つ。未知の文法と構造的に等価な文法を学習するためには、その文法の構造に関する情報が学習システムに利用可能でなければならない。我々のシステムは、構造的データから未知の文法と構造的に等価な文法を学習する。

(2) 文法は正の例だけから学習される。文法に関する正のデータだけを与える教師を仮定することは、実際的な使用において妥当であるが、完全な(正と負の)データを与える教師を仮定することは、教師であるユーザにとって大きな負担となる。我々は、構造的な正の例だけから学習可能な、リバーシブル文脈自由文法と呼ばれる文脈自由文法の部分クラスを定義する。

文脈自由文法  $G = (N, \Sigma, P, S)$  がインバーティブルであるとは、 $P$  中に  $A \rightarrow \alpha$  と  $B \rightarrow \alpha$  なる形の2つの生成規則があるならば、 $A = B$  であるときを言う。文脈自由文法  $G$  がリセットフリーであるとは、任意の2つの非終端記号  $B$  と  $C$ 、それから  $\alpha, \beta \in (N \cup \Sigma)^*$  に対して、 $P$  中に  $A \rightarrow \alpha B \beta$  と  $A \rightarrow \alpha C \beta$  なる形の2つの生成規則があるならば、 $B = C$  であるときを言う。文脈自由文法  $G$  がリバーシブルであるとは、 $G$  がインバーティブルかつリセットフリーであるときを言う。

(3) 文法は効率的に学習される。文法推論の実際的使用において最も重要な問題点は、学習システムの実行時間の効率性である。学習システムの効率性を評価する1つの基準は、多項式時間学習である。我々の学習システムは、多項式時間学習を達成している。

## 1. 例文の入力

Input Examples	Output Grammar
Example : The girl likes a cat	
inference check exit	
Check with Parser	

## 2. 木構造の作成

Input Examples	Output Grammar
Example Make Tree	
inference	
Check with Parser	

```
graph TD; Root(( )) --- Node1(( )); Root --- Node2(( )); Node1 --- the[the]; Node1 --- girl[girl]; Node2 --- likes[likes]; Node2 --- Node3(( )); Node3 --- a[a]; Node3 --- cat[cat];
```

### 3. 学習された文法の出力

<p><b>Input Examples</b></p> <pre>((a) (dog)) ((chases) ((the) ((girl)))) Example : a dog chases a cat ((a) (dog)) ((chases) (a) ((cat)))) Example : a dog chases a girl ((a) (dog)) ((chases) (a) ((girl)))) Example : a dog chases a young girl ((a) (dog)) ((chases) (a) ((young) ((girl)))) Example : &lt;&lt; Inference &gt;&gt; Example : &lt;&lt; Check &gt;&gt;</pre>	<p><b>Output Grammar</b></p> <pre>G = (N,A,P,S) N = (S, NT1, NT2, NT3, NT4,       NT5, NT6, NT7) A = (the, a, girl, cat, dog,       likes, chases, young) P = {   S -&gt; NT1 NT5   NT1 -&gt; NT2 NT3   NT2 -&gt; the   NT2 -&gt; a   NT3 -&gt; NT4   NT3 -&gt; NT7 NT3   NT4 -&gt; girl   NT4 -&gt; cat   NT4 -&gt; dog   NT5 -&gt; NT6 NT1   NT6 -&gt; likes   NT6 -&gt; chases   NT7 -&gt; young }</pre>
<p>inference check exit</p>	
<p><b>Check with Parser</b></p> <p>Example : the young girl chases a dog</p>	

### 4. 学習された文法による文の構文解析

<p><b>Parse Tree</b></p> <pre> graph TD     S --&gt; NT1     S --&gt; NT5     NT1 --&gt; NT2     NT1 --&gt; NT3     NT2 --&gt; the     NT7 --&gt; young     NT3 --&gt; NT3     NT3 --&gt; NT4     NT3 --&gt; girl     NT4 --&gt; chases     NT5 --&gt; NT6     NT5 --&gt; NT1     NT6 --&gt; likes     NT6 --&gt; chases     NT1 --&gt; NT2     NT1 --&gt; NT3     NT2 --&gt; a     NT3 --&gt; dog     </pre>	<p><b>Output Grammar</b></p> <pre>G = (N,A,P,S) N = (S, NT1, NT2, NT3, NT4,       NT5, NT6, NT7) A = (the, a, girl, cat, dog,       likes, chases, young) P = {   S -&gt; NT1 NT5   NT1 -&gt; NT2 NT3   NT2 -&gt; the   NT2 -&gt; a   NT3 -&gt; NT4   NT3 -&gt; NT7 NT3   NT4 -&gt; girl   NT4 -&gt; cat   NT4 -&gt; dog   NT5 -&gt; NT6 NT1   NT6 -&gt; likes   NT6 -&gt; chases   NT7 -&gt; young }</pre>
---	---