適応パーサによる文脈自由文法の自動修正

4M－7

中澤 聡　東京大学
浜田 喬　学術情報センター

1 はじめに

ある一定の形式にしたがって作成されたテキストなどを文脈自由文法パーサを用いて構文解析することはよく見られる状況である。しかし、入力され得る全てのテキストの形式を正しく記述する文法規則を予め用意しておくことは難しい場合が多い。また、仕様の変更や、内容的にはほとんど同じでありながらこれまでとは異なる形式の入力を構文解析する際に、手動で文法規則を修正するのは大きな労力を必要とする。

そこで本稿では、予め基礎となる文法規則を元文法として与えておき、以後、元文法では正しく構文解析できない修正用の例文が入力されると、それを受理できるよう新たな生成規則を自動的に追加していく構文解析システムを提案する。

2 全体の構成

システム全体の大まかな構成は図 1 のようになっている。ここでは大きな処理の流れを説明した後、スペースの都合上特に構文解析手法について取り上げる。

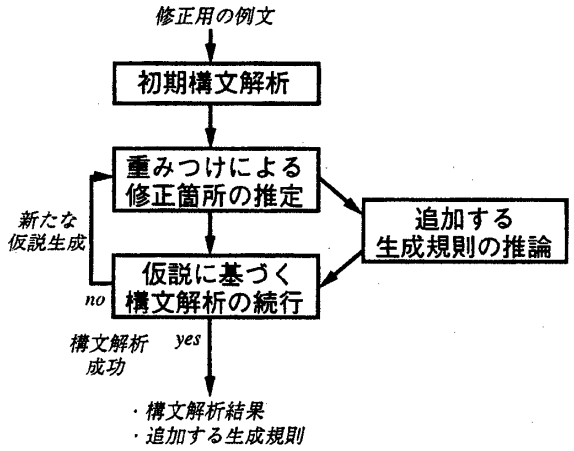

図 1: システムの構成

修正用の例文が入力されると、まず元文法だけを用いて構文解析を行なう。ただし、当然のことながら元文法だけでは構文解析は成功せず、途中で停止する。

停止後、それまでの解析結果から例文中の修正箇所、すなわちその部分の構文解析に新たな生成規則が必要となる場所を、大まかに推定する。この手法については節 3 で説明する。

ついで、推定された修正箇所から追加する生成規則の仮説をたてる。この生成規則の仮説は、既存の生成規則に対して単語の削除、位置交換、挿入、置換の操作を何度か施すことにより作成する。ここで、単語とは元文法の生成規則に頻出する記号列のパターンを指しており、元文法が与えられた時点で予め計算しておく。また、推定された修正箇所に対して何通りもの生成規則が考えられるので、上記の削除、位置交換などの各操作ごと、単語ごとにコスト関数を設定しておき、修正箇所に適する生成規則の仮説のうち、このコスト関数が小さいものから順にパーサに渡していく。

パーサは渡された仮説ごとに ID をつけて、構文解析を続行する。新たな生成規則の仮説を用いてもあまり構文解析が進まなければ (節 3 参照)、処理を中断して、また別の仮説を試みる。

その仮説を用いて構文解析がうまく終了すれば、解析結果と追加する生成規則を出力する。

3 構文解析手法

3.1 初期構文解析

最初に行なう構文解析は、元文法だけで可能な限り解析を進めることにより、例文中の修正箇所をおおまかに推定することを目的としている。

その解析手法としては、並列ボトムアップチャート法 [1] を基本とする。ただし、以下のような変更を施している。

- 左隅分岐法[3]の拡張
  生成規則の右辺のどの位置の記号が修正されても、可能な限り構文解析が進むよう、任意の順で記号を探索する。例えば、$A \rightarrow a\ b\ c$ のような生成規則を $A \rightarrow a'\ b\ c$ のように修正するための例文が入力されたとする。例文中に "$a'\ b\ c$" のような記号列が存在しても左隅分岐法では最左未決定記号 a に対応

Automatic Modification of Context Free Grammars by use of an Adaptive Parser
Satoshi Nakazawa[1], Takashi Hamada[2]
[1]University of Tokyo
[2]National Center for Science Information Systems

[3]最も左端にある不確定の記号を探索していく手法

する記号がないため、この記号列の活性弧[4]はまるで作られないが、本稿の手法では b からも探索するため、少なくとも $[A\ [a\ ?]\ [b]\ [c]]$ のような活性弧が作られる。これはその後の修正箇所の推定の手掛かりとなる。

- 記号の重み付け
  終端記号の重みを一律 1 とし、以後、ある記号の還元の仕方が n 通り存在すれば、各生成規則に対してその記号の重みを $1/n$ にして伝えることにより、再帰的に全ての記号の重み付けを行なう。
  例えば、$A \rightarrow a\ b\ c$ という生成規則と $B \rightarrow b\ c$ という生成規則があるとき、"$a\ b\ c$" という入力記号列に対して、図 2 のように重み付けする。

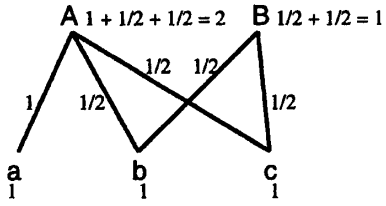

図 2: 重み付けの例

3.2 修正箇所の推定

一般にある記号を生成規則に従って還元する仕方には複数の解釈が考えられ、どの解釈が正しいかは、最終的に開始記号を根とする解析木の一部となっているかどうかによって判断される。

しかし、修正用の例文の構文解析は途中で停止してしまうため、そのような判断はできない。そこで、解釈の仕方が少ない、すなわち重みの大きい記号を基準とする。例えば、$D \rightarrow E\ F\ G$ のような生成規則に対して、$[D\ [E\ ?]\ [F]\ [G]]$ のような活性弧が作られ、この弧の記号 F と G の重みが大きければ修正箇所は $[E\ ?]$ に当たる部分ではないかと推定できる。このような箇所に対しては、まず $[E\ ?]$ の部分が仮に確定したと仮定して構文解析を進める。その結果、開始記号を根とする解析木ができるならば、仮定した部分を修正箇所の候補とする。

また同時に、追加する生成規則の左辺記号の候補としてはこの例の場合 D または E であると考えられる。

以上のような条件が成立する弧を全て求めて、その結果を追加する生成規則の推論部に送る。

[4]並列ボトムアップチャート法の用語。ある節点 (文中における記号の位置) からある節点までに並んで存在している記号列が、対応する生成規則の左辺記号に還元され得ることを示すリスト表現。生成規則の右辺の記号列が全て確定している弧を不活性弧、まだ確定していない記号がある弧を活性弧と呼ぶ。

3.3 仮説に基づく構文解析の続行

追加する生成規則の仮説に対して ID をつけ、その生成規則と元文法の生成規則を用いて構文解析を続行する。追加する生成規則の仮説によって作られた弧、及び、その弧からさらに作られた弧にはそれと分かるように仮説の ID を振っておく。以後、仮説の ID ごとに、その仮説によって作られた記号の重みの総和を記録していく。仮説を導入した結果、構文解析が進んだかどうかはこの重みの総和が増加していくかどうかによって判断される。導入しても構文解析が進まない仮説に対しては処理を中断して、また別の仮説を試みる。

4 おわりに

現在のところ、簡単な文法規則、すなわち記号数や生成規則数が少ない文法や、良質の手掛かりとなり得る重みの大きい記号が多い文法に対しては、本手法は有効に働く。しかし文法規模が大きくなるに従って計算時間の増大、複数解釈の存在などの問題が生じてくる。

また、詳しく触れることはできなかったが、追加する生成規則の推論に用いるコスト関数、ある仮説による処理を中断して別の仮説を試みる時の判断のしきい値などにいくつかのパラメータを用いているが、これらは全て実験的に適当に定めたものであり、処理中は固定である。

今後の改良点として、これらのパラメータを与えられた文法規則、例文に従って動的に変化させることが考えられる。また、入力される修正用の例文も同時に 1 文ずつ処理するのではなく、複数の例文が与えられた時にはそれらを相互に利用する手法が有効であろう。

さらに、今回提案した手法ではあまり考慮されていないが、再帰関係とくに括弧関係を取り扱う生成規則を修正するためには、特別の処理が必要になる。

今後は、以上のような点に関して改良を考えていくとともに、本手法が有効に働く文法規則の理論的条件などに関しても調べていきたい。

参考文献

[1] M. Kay. "Algorithm Schemata and Data Structures in Syntactic Processing", *TR CSL-80-12*, Xerox PARC, Oct. 1980.