自 然 言 語 処 理 101−11
（1994. 5. 28）

コーパスからの文法の自動抽出

白井清昭　徳永健伸　田中穂積
東京工業大学工学部情報工学科

本論文は構文構造が付加されたコーパスから自動的に文脈自由文法を抽出する方法について述べる。まず構文構造内のノードに自動的に非終端記号を割り当てて、規則数210600の曖昧性のない文法を抽出した。次に自動的に割り当てられた非終端記号に対して、抽出された規則の右辺の記号列を見て人間が適切な名前を与えることにより文法の規則の数を削減した。さらに非終端記号に対する名前付けを自動的に行う方法も提案した。また、文法中の右辺長の長い規則を右辺長の短い規則を用いて分解することにより、さらに文法サイズの縮小を試みた。最後に抽出した文法を用いてコーパスの例文の統語解析を行い、統語的な曖昧性の数を調べる実験を行った。

Automatic grammar extraction from bracketed corpus

SHIRAI Kiyoaki　TOKUNAGA Takenobu　TANAKA Hozumi
Department of Computer Science
Tokyo Institute of Technology

This paper discribes a method of automatic extraction of context-free grammar from bracketed corpus. First, unambiguous grammar with 210600 rules is extracted by automatically replacing nodes in trees with non terminal symbols. In order to reduce the number of the rules, we give proper names to non terminal symbols taking account of the right hand side of the rules. For further reduction of grammar size, we decompose rules which have many symbols in the right hand side. Finally, we conducted an experiment to analize sentences with the extracted grammar.

1 研究の背景

近年コーパスなどの事例をもとに、自然言語解析を統計的に取り扱う研究が盛んに行われている。そのような研究の一つとして、統計文脈自由文法 (Stochastic Context Free Grammar) に関する研究がある。Lari と Young [1] は、コーパスから統計文脈自由文法の確率の学習や文法規則の抽出を行う Inside-Outside アルゴリズムを提案している。また Pereira と Schabes [2] は、部分的に括弧付けされたコーパスに対して Inside-Outside アルゴリズムを適用する方法を提案している。

本論文では、括弧付けによる構文構造が付加されたコーパスから、Inside-Outside アルゴリズムとは異なる方法で、効率良くかつ実用的な文脈自由文法を抽出する方法を提案する。

2 文法の自動抽出

本研究では、文法を抽出するコーパスとして EDR コーパスを使用した。我々が使用したコーパスは、新聞・雑誌・辞書などから取りだした 34192 の例文を持ち、補助情報として括弧付けによる構文構造、形態素情報 (品詞、読み、形態素の意味を表す概念識別子)、文全体の意味を表す概念関係表現が含まれている。図 1 は、EDR コーパスの「裏通りには棟割り長屋が残る。」という例文に対する構文構造である。

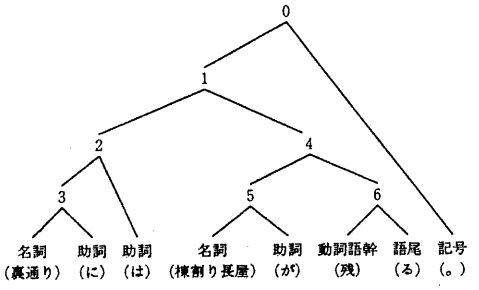

図 1: EDR コーパスに含まれる構文構造

この構文構造の枝分かれは文法規則に対応している。すなわち、図 1 の構文構造は次のような規則の集合であると考えられる。

0 → 1 記号
1 → 2 4
2 → 3 助詞
3 → 名詞 助詞
4 → 5 6
5 → 名詞 助詞
6 → 動詞語幹 語尾

この規則に含まれている数字は構文構造内の 1 つのノードを表している。これらのノード番号に適切な非終端記号を割り当てれば文法規則が得られる。ここで、全ての非終端記号がただ一通りの記号列に展開されるように、ノード番号に対して非終端記号の割り当てを行えば全く曖昧性のない文法が抽出できると考え、次のような方法により文法を抽出することにした。

【 抽出アルゴリズム I 】

(1) コーパスに含まれる全ての例文の構文構造を、上述のような規則に分解する。

(2) 全ての規則の中から、右辺に含まれる記号が全て非終端記号または品詞であり、かつ左辺にまだ非終端記号が割り当てられていない規則を見つける。そのような規則がなければ終了。

(3) (2) で見つけた規則の左辺ノードがもとの構文構造のルートに相当する場合は、左辺ノードに開始記号 S を与える。それ以外の場合は、右辺の記号列に対してユニークな非終端記号をその規則の左辺のノードに与える。また、この左辺ノードと同じノードが別の規則の右辺にあるはずなので、このノードも同じ非終端記号を割り当てる。

(4) (2) へ戻る。

ここで、品詞とは図 1 のような構文構造の葉に割り当てられた記号を指し、非終端記号とはそれ以外のノードに割り当てられるべき記号を指している。この抽出アルゴリズム I を用いて各ノードにボトムアップに非終端記号を割り当てることにより、自動的に文法を抽出することができる。例えば、図 1 の構文木からは次のような文法規則が抽出される。

S → a628 記号
a628 → a21 a43
a21 → x1 助詞
x1 → 名詞 助詞
a43 → x1 x0
x1 → 名詞 助詞
x0 → 動詞語幹 語尾

a または x で始まる記号が新たに割り当てられた非終端記号である。非終端記号の割り当ては規則の右辺に対してユニークに行うので、例えば“動詞語幹 語尾”という記号列を右辺に持つ規則が他の例文の構文構造に含まれているならば、その規則の左辺にも同じ x0 という非終端記号を割り当てる。

コーパスに含まれる全ての構文木に対して同様の操作を行ったところ、新たに 176846 個の非終端記号を割り当て、210600 個の規則を抽出することができた。こうして抽出された文法を文法 I と呼ぶことにする。文法 I は、コーパスに含まれる全ての文に対して、コーパスと同一の構文木を生成することができる。しかもこの文法には曖昧性が全くない。すなわち、開始記号 S を除く全ての非終端記号は必ず 1 通りの記号列にしか展開されない。一方、文法 I の問題点としては、規則の数が多過ぎることがあげられる。実際に解析に使用するには、文法のサイズをもっと小さくしなければならない。

3 文法サイズの縮小

文法 I は、コーパスから自動的に抽出することができ、曖昧性も全くないなどの優れた特性を持ちながら、規則数があまりにも多いため実用には適さないものであった。この章では、文法 I の文法サイズを縮小する方法を提案する。

次の 3 つの文法規則は、文法 I に含まれている規則の例である。

x6 → 連体詞 名詞
x9 → 名詞 名詞
x16 → 数字 名詞

これらの規則の非終端記号 x6, x9, x16 は全て“名詞句”であると考えられるが、機械的に非終端記号の割り当てを行ったため、実際には異なる非終端記号が割り当てられている。したがって、これらの非終端記号に全て“名詞句”と名前を付けて同一の非終端記号とすれば、規則中に含まれる非終端記号、ひいては文法規則の数を減らすことができる。そこで、次のような方法を用いて文法 I に含まれる非終端記号に適切な名前を与えることにする。

【 抽出アルゴリズム II 】

(1) 抽出アルゴリズム I で文法を抽出する。

(2) 右辺の全ての非終端記号に対して名前付けが終了していて、かつ左辺の非終端記号に対しては名前付けが終了していない規則を全て見つける。そのような規則がなければ終了。ただし、品詞および開始記号 S は既に名前付けが終了しているものとみなす。

(3) (2) で見つけた規則について、右辺の非終端記号列を見て適切と思われる非終端記号名を左辺に与える。

(4) (3) で新たな非終端記号に置き換える前に左辺に割り当てられていた非終端記号が他の規則の右辺に現れていたら、その非終端記号も新たな非終端記号名に置き換える。

(5) (2) に戻る。

しかしながら、文法 I に含まれる 210600 個の規則全てに対してこの作業を行うのはあまりにも大変である。そこでコーパスから 50 文をランダムに取り出し、これらの例文のみから文法を抽出することにした。まず抽出アルゴリズム II の (1) の操作により、非終端記号数 434, 規則数 469 の文法が得られた。次に抽出アルゴリズム II の (2)〜(5) の操作により、非終端記号数 14, 規則数 137 の文法が得られた。この文法を文法 II-A とする。非終端記号に対して名前付けを行うことにより、規則の数を 30%程度に減らすことができた。ところが、文法 II-A の解析率 (EDR コーパスの全ての例文のうち、コーパスと同一の構文木を作れる文の割合) は 25.51%であり、全ての例文についてコーパスと同一の構文木を生成することはできなくなってしまった。

文法の解析率を向上させるためには、より多くの例文から文法を抽出しても良いのだが、文法規

則の数も増大するうえ、コーパスにおける出現回数の少ない規則を取り出してしまう可能性もある。そこで、コーパスにおける出現回数の少ない規則を文法から除去し、出現回数の多い規則を文法に加えれば、文法サイズを維持しながら解析率を改善することができる。以下、この方法について説明する。

【 文法の洗練アルゴリズム 】

(1) 構文構造の枝分かれの葉の記号列を右辺とする規則が文法に含まれていれば、その枝分かれの根にあたるノードをその規則の左辺に置き換える。このときその規則の使用回数、すなわちその規則のコーパスにおける出現回数を数えておく。

例. 図1の構文構造の場合、"後置詞句 → 名詞 助詞" という規則を使って3のノードを後置詞句に置き換える。

(2) (1) の操作を構文構造のルートが開始記号 S に置き換えられるまで繰り返す。S に置き換える前に、枝分かれの葉の記号列を右辺とする規則が文法になかった場合、その記号列の出現回数を数える。

(3) 全ての例文の構文構造について、以上の操作を繰り返す。

(1) で数えた出現回数の少ない規則は文法から削除する。また、(2) で数えた出現回数の多い記号列に対しては、その記号列を右辺とする規則を文法に加えれば、文法の解析率の向上が期待できる。このとき、右辺となる記号列を見て適切な非終端記号名を左辺につける。(1) でどれくらいの数の文法規則を削除し、(2) でどれくらいの数の文法規則を新たに追加するかについては色々なアルゴリズムが考えられる。また、(1) で削除する文法規則の数と (2) で新たに追加する文法規則の数を調整すれば、文法のサイズを任意に調整することができる。

文法 II-A をもとにして、規則の入れ替えを次のように行った。

- 文法 II-B の作成
  文法 II-A の中から、(1) における出現回数が 100 以下である 35 個の規則を除去し、(2) における出現回数が 500 以上である 6 個の記号列を右辺とする規則を追加した。
- 文法 II-C の作成
  文法 II-B の中から、(1) における出現回数が 200 以下である 6 個の規則を除去し、(2) における出現回数が 450 以上である 3 個の記号列を右辺とする規則を追加した。

これらの文法の非終端記号数 (NT 数と表記)、規則数、解析率をまとめると、表1のようになった。規則の入れ換えを行うにつれて、規則数が減少しているにも関わらず、解析率が向上していることがわかる。

	文法 II-A	文法 II-B	文法 II-C
NT 数	14	14	14
規則数	137	108	105
解析率	25.51%	30.67%	31.55%

表 1: 洗練された文法

4 名前付けの自動化

抽出アルゴリズム II によって文法を抽出する場合、非終端記号に名前付けを行う際に適切と思われる非終端記号名を人間が考えなければならなかったので効率が悪かった。ここでは、非終端記号に対する名前付けを自動的に行うことを考える。

日本語の特徴として一般的に知られているのは、前の単語が後ろの単語を修飾する、すなわち句の主辞はその句における一番最後の単語であるということである。例えば、先ほど

x6 → 連体詞 名詞
x9 → 名詞 名詞
x16 → 数字 名詞

という規則の左辺には全て "名詞句" と名前を付けたが、これは右辺の一番右にある品詞が "名詞" であることに起因していると考えられる。そこで、右辺の一番右にある品詞を見て、その品詞に "句" をつけた記号を左辺の非終端記号名とすることにより、自動的に非終端記号の名前付けを行うことができる。

しかしながら、EDR コーパスに含まれる全ての品詞について、このような名前付けが適切であ

るわけではない。例えば、“x0 → 動詞語幹 語尾”という規則については、右辺の一番右にある品詞は“語尾”であるが、x0 を“語尾句”とするよりも“動詞句”とする方が適切である。このような場合、左辺に名前を与える際に例外的な処理が必要となる。そこで、右辺の記号列から左辺の非終端記号名を決定する方法として、次のようなものを考えた。

【 左辺の非終端記号名の決定方法 】
右辺の一番右にある記号をXとする。

- Xが“接尾語”のとき
  “接尾語”という品詞を与えられているのは「月」「メートル」などの単位を表す単語が多い。したがって、“接尾語”で終わる句は名詞句であるとみなし、左辺に“名詞句”という非終端記号名を与える。
- Xが“助詞”のとき
  左辺ノードに“後置詞句”という非終端記号名を与える。
- Xが“語尾”または“記号”のとき
  “記号”という品詞を与えられているのは、主に読点と句点である。このときはXの左隣にある記号を改めてXとし、再び**左辺の非終端記号名の決定方法**を適用する。ただし、右辺が“語尾”または“記号”のみから構成されている場合、左辺には“語尾句”もしくは“記号句”という非終端記号名を与える。
- Xがそれ以外の品詞のとき
  左辺ノードに“X句”という非終端記号名を与える。
- Xが非終端記号のとき
  Xの末尾には“句”が付けられているはずである。このとき、この規則は右再帰を用いて、左辺ノードにもXという非終端記号名を与える。ただし、Xが“語尾句”または“記号句”のときは、Xの左隣にある記号を改めてXとして、再び**左辺の非終端記号名の決定方法**を適用する。

抽出アルゴリズム III を次のように定義する。

【 抽出アルゴリズム III 】

抽出アルゴリズム I と同様の方法で文法の抽出を行う。ただし、(3) で左辺に非終端記号を割り当てる際には先ほどの**左辺の非終端記号名の決定方法**を用いる。

抽出アルゴリズム III により EDR コーパスから文法を抽出したところ、非終端記号数 29、規則数 3074 の文法が得られた。この文法を文法 III-A とする。文法 III-A は、コーパスの全ての例文についてコーパスと同一の構文木を生成することができるが、規則数は文法 II に比べるとかなり多い。そこで規則数を減らすために、文法 III-A の各規則のコーパスにおける出現回数を調べ、出現回数が 1 の規則を文法から取り除いたものを文法 III-B、10 以下の規則を取り除いたものを文法 III-C とした。各文法の非終端記号数、規則数、解析率をまとめたのが表 2 である。

	文法 III-A	文法 III-B	文法 III-C
NT 数	14	14	13
規則数	3074	1479	505
解析率	100%	95.50%	86.65%

表 2: 抽出アルゴリズム III による文法

5 右辺の長い規則の分解

前章で提案した方法により、規則数を文法 I の約 1.46%に抑えた文法を自動抽出することができたが、それでも規則数は 3074 と依然として多い。そこで抽出された文法に含まれている具体的な規則を調べてみたところ、次のような右辺の長い規則が多く含まれていることがわかった。

名詞句 → 動詞語幹 語尾 名詞 助詞
　　動詞語幹 助動詞 名詞

ところが、文法に“動詞句 → 動詞語幹 語尾”や“後置詞句 → 名詞 助詞”というような規則が含まれていれば、この規則を次のように書き換えても、同じ記号列を生成できる。

名詞句 → 動詞句 後置詞句 動詞語幹 助動詞 名詞

このように、右辺の部分列に対して他の規則を適用することを、規則の分解と呼ぶことにする。右辺の長い規則を分解して右辺長を短くすれば、最終的に得られる文法規則の数も減少することが期待できる。規則の右辺を分解した場合、分解後の文法がコーパスと同一の構文木を生成することは保証されなくなるが、少なくともコーパスの構文構造に近い構文木が得られるはずである。本研究では、コーパスと同一の構文木を生成することができなくても、それに近い構文木が得られれば問題はないとし、右辺の分解による文法サイズの縮小を試みた。本章では、右辺の長い規則を適切に分解するために2つの方法を提案する。

5.1 規則の両端に現れる品詞の利用

右辺の長い規則を分解しようとしたときに、右辺の記号列のどこに区切りを入れるかが問題となる。区切りを入れる目安として、常に右辺の両端(先頭もしくは末尾)に現われる品詞を探し出す方法が考えられる。例えば、"語尾"という品詞は右辺の末尾に現われることが多いので、右辺の記号列の中に"語尾"が含まれていれば、"語尾"の直後に切れ目を入れるのが妥当であろう。

まず、文法Iの規則の中で右辺が全て品詞からなるもののうち、右辺長が2であるものを取り出し、各品詞に対して次の値を計算した。

$$R_f = \frac{\text{その品詞が右辺の先頭に現われる回数}}{\text{その品詞が右辺に現われる回数}}$$

$$R_l = \frac{\text{その記号が右辺の末尾に現われる回数}}{\text{その記号が右辺に現われる回数}}$$

R_fの値が大きい品詞は右辺の先頭に現われる傾向があり、R_lの値が大きい品詞は右辺の末尾に現われる傾向があるといえる。上式の計算を右辺長が2である規則に限定して行ったのは、右辺長が3以上の規則を含めると先頭あるいは末尾以外の場所に現れる品詞が多くなり、必然的にR_f,R_lの値が低く抑えられてしまうので、どの品詞が右辺の先頭または末尾に現れるのかという傾向が読み取りにくくなると考えたからである。各品詞についてR_f,R_lの値をまとめると、表3のようになる。

この結果から、コーパスに含まれている品詞をR_fとR_lの値を比較して、右辺の先頭に現れやすい品詞(以下、Fタイプと呼ぶ)と末尾に現れやすい品詞(以下、Lタイプと呼ぶ)に分類することができる。各品詞がどちらのタイプに分類されるかは、表3の一番右の欄に示してある。

品詞	R_f	R_l	タイプ
形容詞	0.998239	0.001761	F
連体詞	0.996192	0.003808	F
接続詞	0.994789	0.005211	F
接頭語	0.988794	0.011206	F
動詞	0.985512	0.014488	F
形容動詞	0.976696	0.023304	F
副詞	0.956467	0.043533	F
数字	0.943640	0.056360	F
感動詞	0.937500	0.062500	F
名詞	0.729209	0.270791	F
助動詞	0.058070	0.941930	L
接尾語	0.001721	0.998279	L
記号	0.000692	0.999308	L
助詞	0.000331	0.999669	L
語尾	0.000207	0.999793	L

表3: 各品詞のR_f,R_l式の値

以上の結果を利用して、規則の分解を次のように行った。

【分解アルゴリズムI】

(1) 規則の右辺に含まれる品詞または非終端記号を、表3に従ってF,Lのいずれかに置き換える。"名詞句"などの品詞以外の非終端記号は、"句"を取り除いて得られる品詞でFかLかを判断する。

(2) LにFが後続する場合、これらの間に区切りを入れて右辺を部分列に分解する。区切りを入れることができなければ、その規則の分解は行わない。

例. 先ほどの規則は次のような部分列に分解される。

名詞句 → (動詞語幹 語尾) (名詞 助詞)
(動詞語幹 助動詞) (名詞)

(3) 分解した部分列を右辺とする規則を作る。左辺の非終端記号は抽出アルゴリズムIIIと同じ方法で自動的に名前をつける。作られた規

則が今までの文法に含まれていなければ、この規則を文法に追加する。ただし、右辺の最初の部分列がLのみからなる場合と、右辺の最後の部分列がFからなる場合は、新たな規則は作らない。

例. 次の3つの規則を文法に追加する。

- 動詞句 → 動詞語幹 語尾
- 後置詞句 → 名詞 助詞
- 動詞句 → 動詞語幹 助動詞

一番最後の(名詞)という部分列は、Fのみからなるので何も行わない。

(4) 右辺の部分列を、(2)で作った規則の左辺に置き換える。この規則が今までの文法に含まれていなければ、この規則を文法に追加する。

例. 次の規則を文法に追加する。

名詞句 → 動詞句 後置詞句 動詞句 名詞

文法III-Aに含まれる文法規則を分解アルゴリズムIにより分解して得られる新たな文法を文法IV-Aとする。文法IV-Aの規則数は1498となり、これは文法III-Aの規則数の約49%に相当する。

5.2 既存の文法規則の右辺を用いる方法

前節の分解アルゴリズムIではLにFが後続する場合にこれらの間に機械的に区切りを入れたため、操作(3)で新たな文法規則を追加してしまう可能性があった。このとき、新たに追加された規則が文法規則として適切であるとは限らない。それに加えて、ここでの目的は規則の数を減らすことにあるので、新たな規則を追加するのは好ましいことではない。このような問題は、右辺を分解して得られる部分列が既存の文法規則の右辺と必ず一致するようにすれば解決できる。そこで、次のような分解アルゴリズムを考えた。

【分解アルゴリズムII】

(1) 既存の文法規則の右辺を組み合わせて、分解の対象となる右辺の記号列を作る。可能な組合せの候補を全て列挙する。

例. 先ほどの規則の場合、次の2つの候補が考えられる。

a 名詞句 → (動詞語幹 語尾) (名詞 助詞)
(動詞語幹 助動詞 名詞)
b 名詞句 → (動詞語幹 語尾 名詞 助詞)
(動詞語幹 助動詞 名詞)

(2) (1)で見つけた全ての候補について、分解した部分列を右辺に持つ規則のコーパスにおける出現回数の和を計算する。そして、計算した和の一番大きい候補を一つ選ぶ。

例. 候補aの場合、次の3つの規則

- 動詞句 → 動詞語幹 語尾
- 後置詞句 → 名詞 助詞
- 名詞句 → 動詞語幹 助動詞 名詞句

のコーパスにおける出現回数の和を計算すると77348となる。候補bについて同様に和を計算すると25となる。したがって、分解の候補としてaを選択する。

(3) (2)で選択した分解の候補について、部分列を既存の規則の左辺に置き換える。この規則が今までの文法に含まれていなければ、この規則を文法規則に追加する。

例. 次の規則を付け加える。

名詞句 → 動詞句 後置詞句 名詞句

文法III-Aに含まれる文法規則を分解アルゴリズムIIにより分解して得られる新たな文法を文法IV-Bとする。文法IV-Bの規則数は1132となり、これは文法III-Aの規則数の約37%に相当する。

6 実験

この章では、前章までで提案した方法により抽出した文法を用いて実際に構文解析を行い、統語的な曖昧性の数を調べる実験を行った。実験は、文長が比較的短いEDRコーパスの例文を100個選択し、この文をLR法を用いて構文解析した。入力文の平均長は15.47であった。実験に用いた文法は、文法II-A, 文法III-B, 文法IV-Bである。各文法の構文木の数の平均は表4のようになった。

人間が非終端記号に名前を付けて抽出した文法II-Aでは、妥当な数の構文木が得られてはいるものの、EDRコーパスの約25%の例文しか正しい構文木を生成することはできない。自動的に名前付けを行った文法III-Bでは、コーパスの約95%の例

	文法 II-A	文法 III-B	文法 IV-B
規則	137	1479	1132
構文木	82.88	28258.53	57208.10

表 4: 文法の曖昧性

文に対して正しい構文木を生成することができるが、比較的短い文を入力文としているにも関わらず、かなりの数の構文木が得られてしまう。また、規則を分解して文法サイズを縮小した文法 IV-B でも、規則数を減らした代わりに統語的な曖昧性が増加していることがわかる。

統語的な曖昧性を減らすために、我々は次のようなヒューリスティックを考えた。

1. EDR コーパスでは、文末の「。」や「?」の品詞は全て記号となっていて、他の読点などの記号とは区別されていなかった。そこで、文末の「。」と「?」の品詞を全て "文末記号" と修正した。

2. コーパスから読点「、」を除去した。

3. 右辺の先頭が L タイプである規則と、右辺の末尾が F タイプである規則をチェックし、誤りと思われる規則を含む例文 91 個をコーパスから削除した。

4. 右辺が "名詞 助詞 名詞" となっている規則を含む文を見つけ、助詞が「や」、「と」、「か」、「とか」以外である例文 88 個をコーパスから削除した。また、規則の右辺を "名詞 名詞接続助詞 名詞" に置き換えた。

以上の操作を行い、再び抽出アルゴリズム III を用いて文法を抽出したところ、規則数 2766 の文法文法 V-A が得られた。また、この文法から出現回数が 1 である規則を除去して、規則数 1346 の文法 V-B を作成した。この文法 V-B を用いて先ほどと同じ例文を統語解析したところ、1 文当たりの構文木数は 18950.6 であった。文法 III-B に比べて約 70%程度に抑えることができたものの、依然として多くの統語的曖昧性が存在している。

7 まとめ

本論文では、構文木付きコーパスから文法を取り出すことを目的とし、いくつかの方法を提案した。まず、EDR コーパスの各構文木のノードに対して自動的に非終端記号を割り当てることにより文法 I を抽出した。次に、文法 I よりも規則数の少ない文法を獲得するために、自動的に割り当てられた文法 I の非終端記号に人間が適切な名前を与えて文法 II を、また名前付けを自動的に行って文法 III を抽出した。また、文法 III の文法規則の右辺を分解することにより、さらに文法サイズを縮小した文法 IV を作成した。最後に抽出した文法により実際にパージングを行い、どれだけ構文的な曖昧性が生じるかを実験した。

今後の課題としては、統語的な曖昧性を解消することが挙げられる。コーパスから抽出した文法規則についてはコーパスにおける出現回数がわかっているので、規則に確率を割り当てて確率文脈自由文法を作り出すのは非常に簡単である。この確率文脈自由文法とビームサーチなどの手法を用いて探索空間を狭めることにより、統語的曖昧性を解消することが考えられる。また、EDR コーパスの品詞は全部で 15 種類しかないので、品詞をもう少し細かく分類してきめの細かい文法を抽出することも考えられる。

謝辞 本研究で使用した EDR コーパスの利用を許可くださいました日本電子化辞書研究所の横井所長、コーパスに関する技術的な支援をしてくださいました日本電子化辞書研究所の仲尾氏に感謝いたします。

参考文献

[1] K.Lari and S.J.Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, Vol. 4, pp. 35–56, 1990.

[2] Fernando Pereira and Yves Schabes. Inside-outside reestimation from partially bracketed corpora. *ACL-92*, pp. 128–135, 1992.