括弧付きコーパスからの日本語確率文脈自由文法の自動抽出

白井　清昭[†]　徳永　健伸[†]　田中　穂積[†]

本論文では，括弧付きコーパスから確率文脈自由文法を効率良く自動的に抽出する方法を提案する．文法規則の抽出は，日本語の主辞が句の一番最後の要素であるという特徴を利用して，括弧付けによる構文構造の内部ノードに適切な非終端記号を与えることによって行う．また，文法規則の確率は規則のコーパスにおける出現頻度から推定する．さらに，文法サイズの縮小と解析木数の抑制という2つの観点から，抽出した文法を改良するいくつかの方法を提案する．文法サイズの縮小は，文法に含まれる冗長な規則を自動的に削除することによって行う．解析木数の抑制は，(1) 同一品詞列に対して右下がりの二分木のみを生成し，(2)“記号” と “助詞” の2つの品詞を細分化し，(3) 法や様態を表わす助動詞に対する構造を統一することにより行う．最後に，本手法の評価実験を行った．約180,000の日本語文から確率文脈自由文法の抽出およびその改良を行ったところ，2,219の文法規則を抽出することができた．抽出された文法を用いて20,000文のテスト例文を統語解析したところ，受理率が約92%となり，適用範囲の広い文法が得られたことを確認した．また，生成確率の上位30位の解析木の評価を行ったところ，括弧付けの再現率が約62%，括弧付けの適合率が約74%，文の正解率が約29%という結果が得られた．

キーワード:　コーパス，日本語，確率文脈自由文法

Automatic Extraction of Japanese Probabilistic Context Free Grammar From a Bracketed Corpus

Kiyoaki shirai[†] , Takenobu Tokunaga[†] and Hozumi Tanaka

In this paper, we describe a method to extract a probabilistic context free grammar of Japanese from a bracketed corpus. To extract grammar rules, we assign appropriate non-terminal symbols to the intermediate nodes of the bracketed trees by taking account of the heads of phrases. We estimate the probabilities of the rules based on their frequency of occurrence. We also propose several improvements to the extracted grammar. The size of the grammar is reduced by removing any redundant rules. The number of the parse tree is reduced (1) by allowing only a right linear binary branching tree for a constituent that consists of items of the same POS, (2) by subcategorizing the POSs “symbol”(“KIGOU”) and “postposition”(“JOSI”), and (3) by assigning a consistent structure to constructs representing clausal modality. Finally, we conducted an experiment that evaluated the proposed methods. 2,219 grammar rules were extracted from about 180,000 sentences. When we analyzed 20,000 test sentences with the extracted grammar, a 92 % acceptance rate was calculated, showing that the grammar has a broad coverage. For the most probable 30 parse trees, we obtained a 62 % brackets recall, 74 % brackets precision and 29% sentence accuracy.

KeyWords:　*Corpus, Japanese, Probabilistic Context Free Grammar*

1　序論

近年，機械可読な言語データの整備が進んだことや，計算機能力の向上により大規模な言語データの取り扱いが可能になったことから，自然言語処理に用いる様々な知識を言語データから自動的に獲得する研究が盛んに行われている (宇津呂・松本 1995)．大量の言語データから自動的に獲得した知識は，人手によって得られる知識と比べて，獲得した知識が人間の主観に影響されにくい，知識作成のためのコストが低い，知識の適用範囲が広い，知識に何らかの統計情報を容易に組み込むことができる，といった優れた特徴を持っている．

言語データから自動獲得される自然言語処理用知識には様々なものがあるが，その中の1つとして文法がある．文法には様々なクラスがあるが，統語解析の際に最もよく用いられるのは文脈自由文法 (Context Free Grammar，以下CFGと呼ぶ) であり，一般化LR法，チャート法などのCFGを用いた効率の良い解析手法がいくつも提案されている．ところが，人手によってCFGを作成する場合，作成の際に考慮されなかった言語現象については，それに対応する規則がCFGに含まれていないために解析することができない．これに対して，コーパスから自動的にCFGを抽出することができれば，コーパス内に現れる多様な言語現象を網羅できるだけでなく，人的負担も極めて軽くなる．また，CFGの拡張の1つとして，文法規則に確率を付与した確率文脈自由文法 (Probabilistic Context Free Grammar，以下PCFGと呼ぶ) がある (Wetherell 1980)．PCFGは，生成する複数の解析結果の候補 (解析木) に対して，生成確率による順序付けを行うことができるという点でCFGよりも優れている．そこで本論文では，CFGをコーパスから自動抽出し，その後各規則の確率をコーパスから学習することにより最終的にPCFGを獲得する手法を提案する．

CFGまたはPCFGをコーパスから自動獲得する研究は過去にもいくつか行われている．文法獲得に利用されるコーパスとしては，例文に対して何の情報も付加されていない平文コーパス，各形態素に品詞が割り当てられたタグ付きコーパス，内部ノードにラベルのない構文木が与えられた括弧付きコーパス，内部ノードのラベルまで与えられた構文木付きコーパスなど，様々なものがある．以下ではまず，文法獲得に関する過去の研究が，どのような種類のコーパスからどのような手法を用いて行われているのかについて簡単に概観する．

平文コーパスからの文法規則獲得に関する研究としては清野と辻井によるものがある (Kiyono and Tsujii 1993, 1994, 1994)．彼らの方法は，まずコーパスの文を初期のCFGを用いて統語解析し，解析に失敗した際に生成された部分木から，解析に失敗した文の統語解析を成功させるために必要な規則 (彼らは仮説と呼んでいる) を見つけ出す．次に，その仮説がコーパスの文の解析を成功させるのにどの程度必要なのかを表わす尤度 (Plausibility) を計算し，高い尤度を持つ仮説を新たな規則として文法に加える．彼らは全ての文法規則を獲得することを目的として

† 東京工業大学大学院 情報理工学研究科 計算工学専攻, Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology

いるわけではなく，最初からある程度正しいCFGを用意し，それを新たな領域に適用する際にその領域に固有の言語現象を取り扱うために必要な規則を自動的に獲得することを目的としている．

タグ付きコーパスからCFGを獲得する研究としては森と長尾によるものがある (森・長尾 1995)．彼らは，前後に現われる品詞に無関係に出現する品詞列を独立度の高い品詞列と定義し，コーパスに現われる品詞列の独立度をn-gram統計により評価する．次に，ある一定の閾値以上の独立度を持つ品詞列を規則の右辺として取り出す．また，取り出された品詞列の集合に対して，その前後に現われる品詞の分布傾向を利用してクラスタリングを行い，同一クラスタと判断された品詞列を右辺とする規則の左辺に同一の非終端記号を与える．そして，得られた規則のクラスタの中からコーパス中に最もよく現れるものを選び，それらをCFG規則として採用すると同時に，コーパス中に現われる規則の右辺の品詞列を左辺の非終端記号に置き換える．このような操作を繰り返すことにより，最終的なCFGを獲得すると同時に，コーパスの各例文に構文木を付加することができる．

括弧付きコーパスからCFGを獲得する研究としては，まずInside-Outsideアルゴリズムを利用したものが挙げられる．LariとYoungは，与えられた終端記号と非終端記号の集合からそれらを組み合わせてできる全てのチョムスキー標準形のCFG規則を作り，それらの確率をInside-Outsideアルゴリズムによって学習し，確率の低い規則を削除することにより新たなPCFGを獲得する方法を提案した (Lari and Young 1990)．この方法では収束性の悪さや計算量の多さが問題となっていたが，この問題を解決するために，PereiraらやSchabesらはInside-Outsideアルゴリズムを部分的に括弧付けされたコーパスに対して適用する方法を提案している (Pereira and Schabes 1992; Schabes, Roth, and Osborne 1993)．しかしながら，局所解は得られるが最適解が得られる保証はない，得られる文法がチョムスキー標準形に限られるなどの問題点も残されている．一方，括弧付きコーパスから日本語のCFGを獲得する研究としては横田らのものがある (横田, 安部, 青島, 藤崎 1996)．彼らは，Shift-Reduceパーザによる訓練コーパスの例文の統語解析が最も効率良くなるように，コーパスの内部ノードに人工的な非終端記号を割り当てることによりCFGを獲得する方法を提案している．これは組み合わせ最適化問題となり，Simulated Annealing法を用いることにより解決を求めている．1000〜7500例文からCFGを獲得し，それを用いた統語解析では15〜47%の正解率が得られたと報告している．この方法では，CFG獲得の際に統計情報のみを利用し，言語的な知識は用いていない．しかしながら，利用できる言語学的な知識はむしろ積極的に利用した方が，文法を効率良く獲得できると考えられる．

構文木付きコーパスから文法を獲得する研究としてはSekineとGrishmanによるものがある (Sekine and Grishman 1995)．彼らは，Penn Tree Bank (Marcus, Santorini, and Marcinkiewicz 1993) の中からSまたはNPを根ノードとする部分木を自動的に抽出する．

解析の際には，得られた部分木をSまたはNPを左辺とし部分木の葉の列を右辺としたCFG規則に変換し，通常のチャート法により統語解析してから，解析の際に使用した規則を元の部分木に復元する．得られた解析木にはPCFGと同様の生成確率が与えられるが，この際部分木を構成要素としているため若干の文脈依存性を取り扱うことができる．しかしながら，SまたはNPがある記号列に展開されるときの構造としては1種類の部分木しか記述できず，ここでの曖昧性を取り扱うことができないといった問題点がある．また，構文木付きコーパスにおいては，例文に付加された構文木の内部ノードにラベル(非終端記号)が割り当てられているため，通常のCFGならば構文木の枝分れをCFG規則とみなすことにより容易に獲得することができる．

大量のコーパスからPCFGを獲得するには，それに要する計算量が少ないことが望ましい．ところが，統語構造情報が明示されていない平文コーパスやタグ付きコーパスを用いる研究においては，それらの推測に要する計算コストが大きいといった問題がある．近年では，日本においてもEDRコーパス (日本電子化辞書研究所 1995) といった大規模な括弧付きコーパスの整備が進んでおり，効率良くCFGを獲得するためにはそのような括弧付きコーパスの統語構造情報を利用することが考えられる．一方，括弧付きコーパスを用いる研究(Pereira and Schabes 1992; Schabes et al. 1993; 横田他 1996) においては，平文コーパスやタグ付きコーパスと比べて統語構造の情報が利用できるとはいえ，反復アルゴリズムを用いているために文法獲得に要する計算量は多い．本論文では，括弧付きコーパスとしてEDRコーパスを利用し，日本語の言語的特徴を考慮した効率の良いPCFG抽出方法を提案する (白井 1995; 白井, 徳永, 田中 1995)．

本論文の構成は以下の通りである．2節では，括弧付きコーパスからPCFGを抽出する具体的な手法について説明する．3節では，抽出した文法を改良する方法について説明する．文法の改良とは，具体的には文法サイズを縮小することと，文法が生成する解析木の数を抑制することを指す．4節では，実際に括弧付きコーパスからPCFGを抽出し，それを用いて統語解析を行う実験について述べる．最後に5節では，この論文のまとめと今後の課題について述べる．

2　括弧付きコーパスからの文法抽出

2.1　EDRコーパスの概要

本論文では，言語データとしてEDR日本語コーパスを使用する．EDRコーパスに収録されている例文数は207,802である．それぞれの文には補助情報として形態素情報，構文情報，意味情報が付加されている．本論文では形態素情報(特に品詞情報)と括弧付けによる構文構造を利用する．EDRコーパスの例文，及びそれに付加された形態素情報・括弧付けによる構文構造の例を図1に示す．

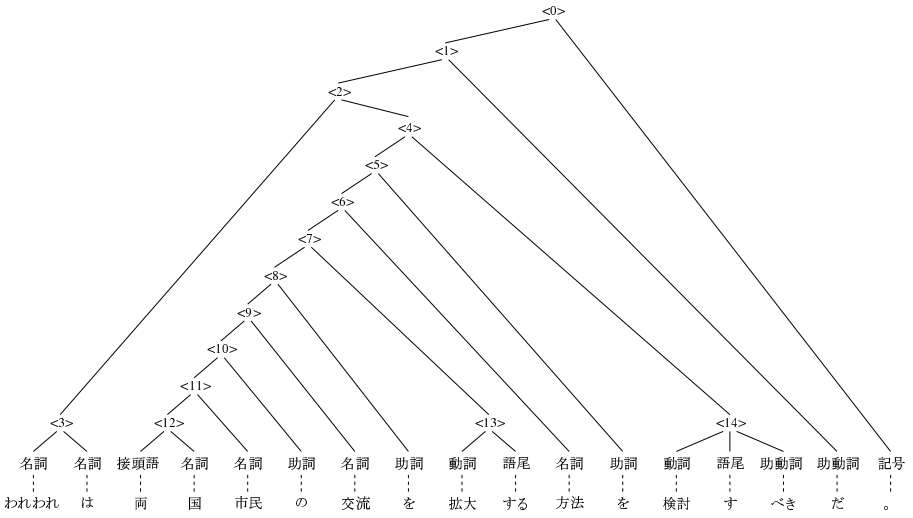

図 1： EDR コーパスの構文構造

EDR コーパスで使われている品詞は以下に挙げる 15 種類であり，比較的粗い品詞体系になっている．

名詞，動詞，形容詞，形容動詞，連体詞，副詞，接続詞，数字，感動詞，助詞，
助動詞，語尾，接頭語，接尾語，記号

ここで注意しなければならないのは，“動詞” という品詞は動詞語幹に対して割り当てられ，語尾には “語尾” という品詞が割り当てられている点である．同様に，“形容詞”，“形容動詞”，“助動詞” という品詞は，それぞれ形容詞語幹，形容動詞語幹，助動詞語幹に割り当てられている．

2.2　ノードへの非終端記号の付与

図 1 は図 2 のような書き換え規則の集合とみなすことができる．図 1 のような構文構造の各ノードに対して適切なラベル (非終端記号) を割り当てることができれば，図 2 の規則は CFG 規則となる．このように，括弧付けによる構文構造の内部ノードに適切なラベルを与えることは括弧付きコーパスから CFG を抽出することと等価である．そこで，2.3 節では構文構造の内部ノードに与えるラベルを決定する方法について考える．

⟨0⟩	→	⟨1⟩	記号		⟨8⟩	→	⟨9⟩	助詞
⟨1⟩	→	⟨2⟩	助動詞		⟨9⟩	→	⟨10⟩	名詞
⟨2⟩	→	⟨3⟩	⟨4⟩		⟨10⟩	→	⟨11⟩	助詞
⟨3⟩	→	名詞	助詞		⟨11⟩	→	⟨12⟩	名詞
⟨4⟩	→	⟨5⟩	⟨14⟩		⟨12⟩	→	接頭語	名詞
⟨5⟩	→	⟨6⟩	助詞		⟨13⟩	→	動詞	語尾
⟨6⟩	→	⟨7⟩	名詞		⟨14⟩	→	動詞	語尾　助動詞
⟨7⟩	→	⟨8⟩	⟨13⟩					

図 2： 構文構造から得られる書き換え規則

2.3　ラベルの決定方法

日本語の特徴として，前の要素が後ろの要素を修飾する，すなわち句の主辞はその句における一番最後の要素であるということが知られている (三原 1994)．例えば，図 2 の中の

⟨12⟩　→　接頭語　名詞

という規則について考えよう．[接頭語 名詞] という句の主辞は句の一番最後にある “名詞” であると考えられる．そこで，この主辞 “名詞” に “句” をつけたラベル “名詞句” を左辺のノード ⟨12⟩ に与えることにする．同様に，

X　→　形容詞句　名詞句

という規則が存在すると仮定し，ラベルの決定されていないノード X に非終端記号を与える場合を考える．この時，[形容詞句 名詞句] という句全体の主辞もまた句の最後にある “名詞句” であると考えられる．先ほどと異なるのは主辞となる記号が非終端記号であるという点である．このような場合には，右再帰を用いて左辺ノード X にも主辞と同じ “名詞句” というラベルを与える．

しかしながら，このようなラベルの与え方が常に適切であるわけではない．

- 主辞にならない品詞

  例えば，

  X　→　接続詞　記号

  という規則について考える[1]．[接続詞 記号] という句の 1 番最後にある品詞は “記号” であるが，この句の主辞は “記号” ではなく “接続詞” である．したがって，左辺のノード X に与えるラベルも “記号句” ではなく “接続詞句” とすべきである．このように，“記号” は主辞にはならない品詞であるとみなし，句の一番最後にある要素が “記号” である場合には，その左隣にある要素を主辞とみなす．

- “語尾” と “助動詞” の取り扱い

  図 2 の中の

  ⟨13⟩　→　動詞　語尾

  という規則について考える．今までのやり方では，[動詞 語尾] という句の 1 番最後にある品詞は “語尾” であるので，左辺のノード ⟨13⟩ に与えるラベルは “語尾句” となる．ところが，2.1 節で述べたように，EDR コーパスにおいては，“語尾” という品詞は動詞の語尾にだけではなく形容詞・形容動詞・助動詞の語尾にも割り当てられている．したがって，このようなラベルの付け方では，

  X　→　形容詞　語尾

  X　→　形容動詞　語尾

  X　→　助動詞　語尾

1 この規則の右辺は「しかし,」などに対応している．

といった規則の左辺にも “語尾句” というラベルを与えることになる．この場合，“語尾句” というラベルを割り当てられたノードが “動詞”，“形容詞”，“形容動詞”，“助動詞” のどれを含んでいるのかを識別することができない．同様に，規則の右辺の一番最後にある要素が “助動詞” のときも，左辺に “助動詞句” というラベルを与えるのは好ましいことではない．このような理由から，句の一番最後にある要素が品詞 “語尾” または “助動詞” である場合には，その左隣にある要素から左辺に与える非終端記号を導出する．

- 主辞が “助詞” の場合

  左辺に “助詞句” というラベルを与えることも考えられるが，わかりやすさのため “後置詞句” というラベルを与える．
- 主辞が “接尾語” の場合

  EDR コーパスにおいては，品詞が “接尾語” となる形態素は「月」,「日」,「メートル」など単位を表しているものが多く，他にも「区」,「氏」など全体として名詞句を形成するものがほとんどである．そこで，主辞が “接尾語” のときには左辺ノードに “名詞句” というラベルを与える．

以上のようないくつかの例外処理が必要ではあるが，基本的には句の一番最後にある要素を主辞とみなして，それから左辺ノードに与えるラベルを決定することにする．

本節で提案した括弧付きコーパスから文法を抽出するアルゴリズムを以下にまとめる．

【文法抽出アルゴリズム】

(1)　構文構造の中で，まだラベルが割り当てられていなくて，かつその子ノードには全てラベルが割り当てられているノードを見つける．そのようなノードがなければ (3) へ．

(2)　(1) で見つけたノードが構文構造のルートである場合には，そのノードのラベルを開始記号 S とする．それ以外は【ラベル決定アルゴリズム】(後述) を用いてノードに与えるラベルを決定する．(1) へ戻る．

(3)　構文構造の全ての内部ノードにはラベルが与えられているはずなので，それを

ノード　→　子ノードの列

という形に分解し CFG 規則とする．

【ラベル決定アルゴリズム】

“記号”,“語尾”,“助動詞” 以外の要素で子ノードの列の最も右側にあるものを選び，それを X とする．

- X が “助詞” の場合，左辺ノードに “後置詞句” というラベルを与える．
- X が “接尾語” の場合，左辺ノードに “名詞句” というラベルを与える．
- X が “助詞”，“接尾語” 以外の品詞の場合，左辺ノードに “X 句” というラベルを与える．例えば主辞が “名詞” の場合，“名詞句” というラベルを与える．

- Xが非終端記号の場合，左辺ノードにも同じXというラベルを与える．例えば主辞が “名詞句” の場合，左辺ノードにも同じ “名詞句” というラベルを与える．

上記の方法によって図1の内部ノードにラベルを与えて抽出された文法規則を図3に示す．この操作をコーパスの全ての構文構造に対して行うことによりCFGを抽出することができる．

S	→	助動詞句	記号	
助動詞句	→	動詞句	助動詞	
動詞句	→	後置詞句	動詞句	
動詞句	→	動詞	語尾	
動詞句	→	動詞	語尾	助動詞
後置詞句	→	名詞	助詞	
後置詞句	→	名詞句	助詞	
名詞句	→	後置詞句	名詞	
名詞句	→	接頭語	名詞	
名詞句	→	動詞句	名詞	
名詞句	→	名詞句	名詞	

図 3： 抽出された文法規則

次に，本手法の文法抽出に要する計算量について考察する．【文法抽出アルゴリズム】は，「句の主辞はその句における一番最後の要素である」という日本語の言語学的特徴を利用して括弧付けによる構文構造の内部ノードに非終端記号を与えているため，文法抽出に必要な計算量はコーパスの構文構造の内部ノード数に比例する．また，長さ n の文があったとき，それに対する最も内部ノード数の多い構文構造は完全な二分木であり，そのときの内部ノード数は $n-1$ である．したがって，文法抽出に必要な計算量は入力文の長さ n にも比例する．このことは大規模なコーパスからの文法抽出を可能にしている．これに対し，本研究と同じく括弧付きコーパスを用いてCFGを獲得するPereiraらの方法 (Pereira and Schabes 1992; Schabes et al. 1993) では，Inside-Outsideアルゴリズムによる規則の推定に必要な計算量は $O(n)$ であり[2]，しかもこの作業を反復しなければならない．また，同じく括弧付きコーパスを利用した横田らの方法 (横田他 1996) では，内部ノードに与える非終端記号をランダムに変化させることを繰り返すSimulated Annealing法を用いてCFG規則を獲得しているため，内部ノードに決定的に非終端記号を与える本手法よりも多くの計算量を必要とするのは明らかである．

2.4　規則の確率の推定

前節で提案した方法により括弧付きコーパスから抽出したCFGに対して，各規則の確率を次のように推定した (Wetherell 1980)．

2 厳密には，コーパスに付加された構文木が完全な二分木のときのみ $O(n)$ となり，それ以外の場合の計算量は $O(n)$ よりも多い．

【規則の確率の推定】

(1) コーパスからCFG規則を抽出する際に，同じ規則を抽出した回数，すなわちその規則のコーパスにおける出現頻度を数える．規則 r_i の出現頻度を $C(r_i)$ とする．

(2) 規則 $r_i : A \rightarrow \zeta_i$ の確率 $P(r_i)$ を次式により求める．

$$P(r_i) = \frac{C(r_i)}{\sum_{\forall r_j : A \rightarrow \zeta_j} C(r_j)} \quad (1)$$

すなわち $P(r_i)$ は，r_i の出現頻度を，A を左辺とする全ての規則の出現頻度の和で割った値とする．

以上のように規則の確率を推定することにより，括弧付きコーパスからPCFGを抽出することができる．

3 文法の改良

サイズの小さなコーパスを用いて，前節で説明した方法によりPCFGを抽出する予備実験を行ったところ，以下のような問題点が明らかになった．

- 文法のサイズが大きい

  EDRコーパスからランダムに選び出した3,000例文からPCFGを抽出したところ，文法規則の数は1,009となり，コーパスサイズに比べて非常に多くの文法規則が抽出されることがわかった．統語解析に要するコストを考えると，文法サイズが不必要に大きいことは望ましいことではない．

- 生成される解析木の数が多い

  抽出したPCFGを用いてEDRコーパスからランダムに選び出した100例文[3]を統語解析したところ，解析結果の候補として生成された解析木の数は平均 1.5×10^6 となり，非常に多くの解析木を生成することがわかった．また，メモリ不足によって解析に失敗した文は69文あった．統語解析を意味解析や文脈解析などの前処理と考えるなら，統語解析結果の候補の数はできるだけ少ないことが望まれる．

本節ではこれらの問題への対応策について述べる．

3.1 文法サイズの縮小

ここでは，コーパスから抽出した文法のサイズを縮小する方法を提案する．文法サイズを縮小する方法としてまず考えられるのは，出現頻度の低い規則を削除することである．しかし，単純に出現頻度の低い規則を削除した場合，その規則がコーパスの構文構造作成時の誤りによって生じた不適切な規則であればよいが，稀にしか現われない言語現象に対応した規則である場

3 PCFGを抽出した3,000例文とは別の例文である．

合には，そのような規則を削除することにより文法の適用範囲(coverage)が狭くなる．両者を出現頻度のみで区別することは難しく，出現頻度が低いからといってその規則を削除することは必ずしも適切ではない．

予備実験で抽出した文法を調べたところ，右辺長の長い規則が多く含まれていることがわかった．予備実験で抽出した文法規則の右辺長の分布を表1に示す．

表 1： 文法規則の右辺長の分布

右辺長	2	3	4	5	6	7	8	9	10	11	12	13	14	16
規則数	235	205	155	161	111	69	31	25	7	6	1	1	1	1

右辺長の長い規則が多く含まれていることがわかる．そのような規則の一例を次に挙げる．

動詞句　→　動詞　語尾　名詞　助詞　形容動詞　語尾　動詞　語尾

これは，コーパスのある例文において，

[　動詞　語尾　名詞　助詞　形容動詞　語尾　動詞　語尾　]

といった括弧付けがなされているためである．本来，その例文の構文構造を反映させるためにはもう少し細かい括弧付けが必要である．しかし，EDRコーパスの中には多くの要素を1つの括弧で括ってしまう例文も存在する．このような右辺の長い規則の存在が文法サイズを大きくしている原因の1つと考えられる．

右辺の長い規則の場合，その規則を除去しても文法中の他の規則によって右辺の記号列を生成できる場合がある．例えば，文法中に次のような規則があったとする．

r_b：　動詞句　→　動詞句　後置詞句　動詞句
r_{c1}：　動詞句　→　動詞　語尾
r_{c2}：　後置詞句　→　名詞　助詞
r_{c3}：　動詞句　→　形容動詞　語尾　動詞　語尾

これら4つの規則を用いれば，非終端記号“動詞句”から“動詞 語尾 名詞 助詞 形容動詞 語尾 動詞 語尾”という記号列を生成することが可能である．このことを図式化したものを図4に示す．このように，ある規則を文法から除去しても，他の規則によって右辺の記号列を生成できるような場合は文法の生成能力は変わらない．

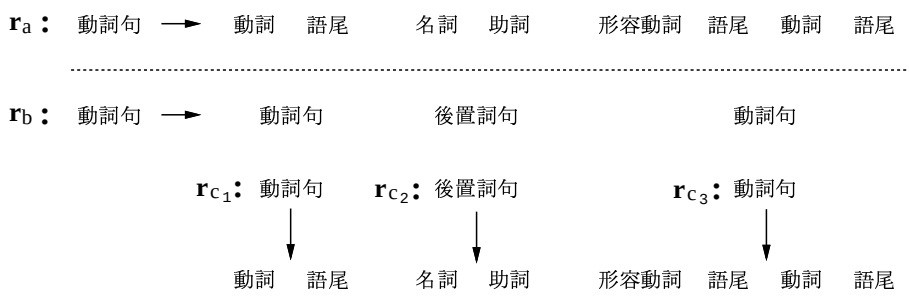

図 4: 複数の規則を用いた記号列の展開

そこで,「冗長な規則」を次のように定義する.

> ある規則 $r_i : A_i \rightarrow \zeta_i$ があるとき，文法内の r_i 以外の規則を用いて
> 非終端記号 A_i を記号列 ζ_i に展開できるならば，すなわち $A_i \stackrel{*}{\rightarrow} \zeta_i$
> であるならば，r_i は冗長な規則である.

$\stackrel{*}{\rightarrow}$ は規則を 1 回以上適用することを示す．冗長な規則を削除する前の文法によって受理される文は，冗長な規則を削除した後の文法でも必ず受理される．したがって，冗長な規則を自動的に検出しそれを削除すれば，文法の適用範囲を狭めることなく文法サイズを縮小することができる.

ここで問題となるのは，冗長な規則のコーパスにおける出現頻度をどのように取り扱うかということである．本論文では，式 (1) に示した通り，規則の出現頻度を規則の確率の推定に用いている．そのため，冗長な規則を文法から削除する際に，その出現頻度をも破棄してしまうのは望ましいことではない．冗長な規則を削除するのは，その規則の右辺の記号列が他の規則によって生成できることが保証されているからである．したがって，削除された冗長な規則の出現頻度は，その規則の右辺の記号列を生成するのに必要な規則の出現頻度に加えるべきである．例えば図 4 において，r_a の右辺の記号列は r_b, r_{c1}, r_{c2}, r_{c3} をそれぞれ 1 回ずつ適用することによって生成されるので，r_a を文法から除去する場合には，r_b, r_{c1}, r_{c2}, r_{c3} の出現頻度に r_a の出現頻度をそれぞれ加えるべきである.

さらに，冗長な規則の右辺を生成する規則の組 $\{r_b, r_{ci}\}$ (図 4 においては $\{r_b, r_{c1}, r_{c2}, r_{c3}\}$) が複数ある場合には，冗長な規則 r_a の出現頻度を，各組の r_b に該当する規則の出現頻度で比例配分してから各規則に足し合わせる．また，ある規則 r_a が冗長であるかどうかを調べる際には右辺長の長い規則から順番に行い，$\{r_b, r_{ci}\}$ が冗長であるかどうかについては考慮しない．そして，r_a が冗長であるとわかった際には，$\{r_b, r_{ci}\}$ の規則の出現回数を更新してから次の規則が冗長であるかどうかを調べる．したがって，例えば図 4 の r_{c3} が冗長な規則である場合でも，r_a の出現頻度は r_{c3} の出現頻度に一旦加えられた後，r_{c3} の右辺の記号列を生成する規則の出現頻度にも足し合わされる.

本節で提案した冗長な規則を検出しそれを削除するアルゴリズムを以下にまとめる.

【冗長規則削除アルゴリズム】

R, R_{new}, $C(r)$, $C_{new}(r)$ を次のように定義する.

R	$\cdots$	抽出した文法規則の集合
$C(r)$	$\cdots$	R 中の規則 r の出現頻度
R_{new}	$\cdots$	冗長な規則を削除して作られる新しい文法規則の集合 (R の中から冗長でない規則を取り出した集合)
$C_{new}(r)$	$\cdots$	R_{new} の各規則の出現頻度

(1)　R_{new} を空集合とする.

(2)　R の中から右辺長の一番長い規則 r_a を1つ抜き出す.

(3)　以下の条件を満たす規則の組 $\{r_b^j, r_{c1}^j, \cdots, r_{cn}^j\}$ を可能な限り見つける.

規則 r_b^j の右辺に含まれる非終端記号 B_i^j を，B_i^j を左辺とする規則 r_{ci}^j の右辺の記号列 β_i^j に置き換えた記号列が r_a の右辺の記号列と一致する.

この条件を図示すると図5のようになる．但し，図5において，$A, B_i \in N$ ， $\alpha_i, \beta_i \in (N+T)*$ である．(N は非終端記号の集合，T は終端記号の集合)

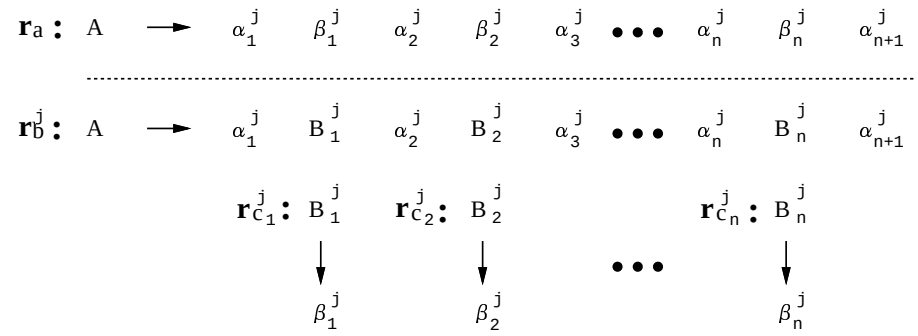

図 5： 冗長な規則のチェック

※ このような規則の組が1つも見つからなかった場合 ($j = 0$ の場合)

r_a は冗長な規則ではない．この規則を R_{new} に加え，$C_{new}(r) = C(r)$ とする.

※ このような規則の組が1つ以上見つかった場合 ($j >= 1$ の場合)

r_a は冗長な規則である．このときは r_a を R_{new} には加えず，出現頻度 $C(r)$ の更新のみを図6のように行う．すなわち，見つけた規則の組の r_b^j に該当する規則の出現頻度で $C(r_a)$ を比例配分し，それを $C(r_b^j)$, $C(r_{ci}^j)$ に加える.

$$C(r_b^j) \quad \leftarrow \quad C(r_b^j) \quad + \quad C(r_a) \times \frac{C(r_b^j)}{\sum_j C(r_b^j)} \qquad for\ all \quad j$$

$$C(r_{ci}^j) \quad \leftarrow \quad C(r_{ci}^j) \quad + \quad C(r_a) \times \frac{C(r_b^j)}{\sum_j C(r_b^j)} \qquad for\ all \quad i,\ j$$

図 6: 出現頻度の更新

(4)　R が空なら終了．それ以外は (2) へ戻る．

以上のように冗長な規則を削除することにより，文法の適用範囲を狭めることなく文法サイズを縮小することができる．この方法により文法サイズをどの程度縮小することができるのかについては第 4 節の実験で評価する．

3.2 解析木数の抑制

ここでは，抽出した文法が生成する解析木の数を抑制するための 3 つの方法を提案する．

3.2.1 同一品詞列の取り扱い

統語解析を行う文の中に同じ品詞が複数並んだ句が存在する場合には，生成される解析木数が増大すると予想される．例えば，“名詞” が 3 つ並んで構成される句の構造としては，名詞間の修飾関係に応じて図 7 に示す 3 つの構造が考えられる．

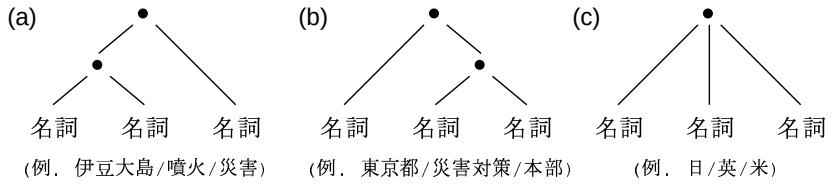

図 7: “名詞” が 3 つ並んだ句の構造

ところが，これらの構造の中から正しいものを選択するためには何らかの意味的な情報が必要である (小林, 徳永, 田中 1996)．したがって，意味的な情報を用いない統語解析の段階では，これらの構造全てを解析結果の候補として生成する．一般に，生成される解析木の数は組合せ的に増大するため，同一品詞列に対して不必要な構造を無意味に生成することが解析木数を増大させる原因の 1 つとなっている．そこで統語解析の段階では，図 7 のような構造を全て生成する代わりに，図 8 のような右下がりの構造のみを出力することにし，この部分の係り受け解析については統語解析の後で行われる意味解析に任せることにした．また，他の非終端記号と区別するために，図 8 の構造の内部ノードには “X 列” (例えば X が “名詞” の場合は “名詞列”) というラベルを与えることにした．

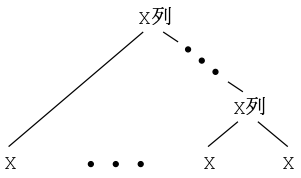

図 8： 右下がりの構造

このように同一品詞列に対する構造を一意に決めれば解析結果として得られる解析木の数を減少させることができる.

同一品詞列に対して図8 のような右下がりの構造のみを生成するために，2.3 節に述べた【文法抽出アルゴリズム】に，次の手続きを最初のステップとして追加する.

【文法抽出アルゴリズム】

0. 構文構造において一種類の品詞のみを支配するノードがあれば，そのノードの下の構造を図 8 のような右下がりの構造に修正する．
1. ～ 3.　　変更なし．

また,【ラベル決定アルゴリズム】に次の手続きを追加する．

【ラベル決定アルゴリズム】

- 子ノードが品詞 “X” または非終端記号 “X 列” のみによって構成されている場合には，“X 列” というラベルを与える．

3.2.2　品詞の細分化

2.1 節で述べたようにEDRコーパスで使われている品詞は 15 種類である．したがって，コーパスから抽出した文法に含まれる終端記号 (品詞セット) の数も 15 であるが，これは統語解析を行うのに十分であるとは言えない。例えば，コーパスの中に

［　名詞　助詞　名詞　］　　(e.g.　記者席 / と / 傍聴席　)

という括弧付けが存在し，名詞並列を表わすCFG規則が抽出されたとする．ところが，この規則は “名詞 助詞 名詞” という品詞列に常に適用され,「地上 / に / 茅 (を出す)」といった名詞並列でない入力に対しても，それが名詞並列であるといった解析結果を出力してしまう．これは全ての助詞に対して “助詞” という品詞を与えているためであり，並列助詞と他の助詞に異なる品詞を与えれば，このような誤った解析を回避することができる．

そこで，EDRコーパスに用いられている品詞を細分化して生成される解析木の数を抑制することを試みた．ここでは “記号” と “助詞” の 2 つの品詞に着目する．

- 品詞 “記号” の細分化

  EDRコーパスにおいては，記号には全て “記号” という品詞が割り当てられている．しかし，読点は文の切れ目を，句点は文の終りを表す特別な記号であり，他の記号とは区別するべきである．そこで，形態素「、」と「,」には “読点” という品詞を与えることにした．また，EDRコーパス中の例文の文末に現れる形態素のほとんどは「。」,「.」,「？」,「！」のいずれかであり，しかもこれらは文末以外に現れることはほとんどなかった．そこで，形態素「。」,「.」,「？」,「！」には “文末記号” という品詞を与えることにした．また，これらの以外の形態素が文末に現れる文，及びこれらの形態素が文末以外の場所に現れる文，合計 102 文を例外としてコーパスから除去した．
- 品詞 “助詞” の細分化

  EDRコーパスにおいては，助詞には全て “助詞” という品詞が割り当てられているが，その助詞の持っている機能により “格助詞”, “係助詞” などの品詞を割り当てるべきであ

る．しかしながら，助詞の中には2つ以上の機能を持っているものもあり，助詞の機能をその表層だけから判断することは一般に困難である．そこで，EDRコーパスにおいて“助詞”という品詞を割り当てられた形態素「M」については，その形態素毎に独自の品詞“助詞M”を割り当てることにした．例えば，形態素「は」が“助詞”という品詞を割り当てられていたならば，その品詞を“助詞は”に変更する．

PCFGの抽出は，まずコーパスの品詞を上記のように細分化し，その後で2.3節で提案した【文法抽出アルゴリズム】に従って行う．また，品詞の細分化に伴い2.3節の【ラベル決定アルゴリズム】を以下のように変更する．下線を引いた部分が変更箇所である．

【ラベル決定アルゴリズム】

“記号”，“語尾”，“助動詞”，“読点”，“文末記号”以外の要素で子ノードの列の最も右側にあるものを選び，それをXとする．

- Xが“助詞M”の場合，左辺ノードに“後置詞句”というラベルを与える．
  (以下同じ)

3.2.3　法・様相を表わす助動詞に対する構造の統一

文末に現われる助動詞は文全体の法や様態を表していることが多い．例えば，EDRコーパス中の2つの例文

(a)　１０月中旬には、袋から顔を出しそうだ。
(b)　そのうえソ連は対越援助を削減しそうだ。

には「そう」と「だ」という2つの助動詞が含まれている．これらは文全体にそれぞれ伝聞，断定の意味合いを持たせる働きをしている．ところがEDRコーパスにおいては，このような助動詞は，文全体に付加している構造(図9の(a))と，文末の最後の要素に付加している構造(図9の(b))の2通りの構造で表されている．このような2種類の構文構造を含むコーパスから抽出された文法は，文末に助動詞を含む文に対して少なくとも図9のような2つの構造を生成し，このことが解析木の数を増加させる一因となっている．

そこで，助動詞が文全体に付加された図9の(a)のような構造を，図9の(b)のような構造に修正してから文法を抽出することにした．助動詞に対する構造を統一することにより，生成される解析木数の減少が期待できる．統一後の構造として図9の(a)ではなく(b)を選択したのは，(a)のような構造からは解析木数を著しく増加させる文法規則が抽出されるからである．例えば，図9の(a)のノード⟨1⟩,⟨2⟩,⟨3⟩,⟨8⟩,⟨10⟩,⟨12⟩には，2.3節の【文法抽出アルゴリズム】に従って“動詞句”という非終端記号が割り当てられ，その結果次のような規則が抽出される．

動詞句　→　動詞句　助動詞　　(“⟨1⟩ → ⟨2⟩ 助動詞”という枝分かれに対応)

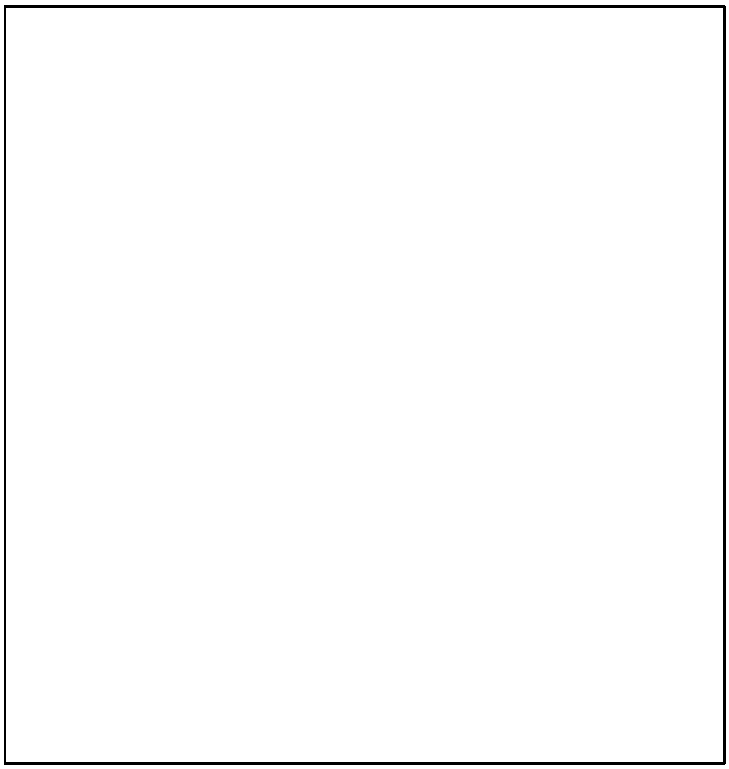

図 9： 助動詞に対する 2 つの構造

ところが，この規則により “助動詞” がノード ⟨8⟩,⟨10⟩,⟨12⟩ に付加される構造も生成されることになり，生成される解析木の数を増加させる要因の 1 つとなっている．これに対して，図 9 の (b) のような構造からは上述のような文法規則は抽出されないため，無駄な解析木を生成することはない．

4　評価実験

本論文で提案した手法の評価実験を行った．まず，EDR コーパスの 207,802 例文のうち，約 10 分の 1 に相当する 20,000 例文をランダムに選んでテストデータとし，残りを訓練データとした．そして，訓練データから PCFG を抽出し，抽出した PCFG を用いてテストデータの例文を統語解析することにより，抽出した PCFG の品質を評価した．

4.1　文法抽出実験

文法抽出を以下の手順で行った．

(1)　訓練データの例文の品詞を細分化した．(3.2.2 節)
　　また，文末の助動詞に対する構造を統一した．(3.2.3 節)

(2)　【文法抽出アルゴリズム】に従って訓練データからPCFGを抽出した．(2.3 節, 3.2.1 節)

(3)　【冗長規則削除アルゴリズム】に従って冗長な規則を削除した．(3.1 節)

(4)　式(1)より各規則の確率を推定した．(2.4 節)

コーパスから抽出したPCFGの概要を表2 に示す.

表 2: 抽出したPCFG

	非終端記号数	終端記号数	規則数
G_0	41	149	15206
G_1	41	149	2219

G_1 は上述の手続きによって訓練データから抽出されたPCFG，G_0 は冗長規則を削除する前のPCFGである．冗長規則を削除したことにより文法サイズを約85%縮小することができた．

4.2　統語解析実験

得られたPCFGを用いてテストデータの例文の統語解析を行った．統語解析は一般化LR法 (Tomita 1986) により行った．LRパーザをSun Sparc Station 10/51(主記憶64Mbyte) 上に実装した．結果を表3 に示す.

表 3: 統語解析結果

受理	不受理	メモリ不足	合計
12,658 文	23 文	7,319 文	20,000 文

「受理」はパーザが解析に成功して1個以上の解析木を出力したことを,「不受理」は解析に失敗したことを,「メモリ不足」はメモリ不足のためにパーザが解析を中断したことを示す．全体の約36%に当たる7,319 文がメモリ不足のために解析できなかった．そこで，これらの文については，生成確率の低い部分木を解析途中で破棄する枝刈りを行いながら再度統語解析を行った．その結果を表4 に示す.

表 4: 枝刈りを行う統語解析結果

受理	不受理	メモリ不足	合計
5,822 文	562 文	935 文	7,319 文

これにより，$12,658 + 5,822 = 18,480$ 文を受理することができた．受理した文の平均単語数は24.45 単語であった．また，生成した解析木数の1 文当たりの平均は 3.24×10^9 であった．非常に多くの解析木が生成されているが，PCFGにより解析木の生成確率を計算し，その上位何位

かを出力することによって解析結果の候補数を絞り込むことが可能である.

まず，文法の適用範囲の広さを示す尺度として，受理率を次のように定義する.

$$\text{受理率} = \frac{\text{受理した文の数}}{\text{統語解析した文の数}}$$

受理率は 18480/20000 $\simeq$ 0.924 となり，適用範囲の広い文法が得られたことがわかる．また，受理しなかった 1520 文のうち 935 文 (約 61.5%) がメモリ不足によるものである．したがって，パーザの使用メモリを増やすことができれば受理率はさらに向上することが予想される.

次に，パーザが出力した解析木の評価を行った．出力された解析木がどれだけ正しいかを評価するための尺度として，括弧付けの再現率，括弧付けの適合率，文の正解率をそれぞれ以下のように定義した.

$$\text{括弧付けの再現率} = \frac{\text{正しい括弧付けの数}}{\text{コーパスの構文構造に含まれる括弧付けの数}}$$

$$\text{括弧付けの適合率} = \frac{\text{矛盾しない括弧付けの数}}{\text{解析木に含まれる全ての括弧付けの数}}$$

$$\text{文の正解率} = \frac{\text{出力した解析木の中に正しい解析木が含まれる文の数}}{\text{受理した文の数}}$$

ここで「正しい括弧付け」とは，コーパスに付加された構文構造の括弧付けと完全に一致している解析木中の括弧付けを表し,「矛盾しない括弧付け」とは，コーパスに付加された構文構造の全ての括弧付けと交差していない括弧付けを表す (Pereira and Schabes 1992)．また「正しい解析木」とは，解析木中の全ての括弧付けが矛盾していない解析木を表す．解析木の評価方法としては，コーパスの各例文に付加された構文構造を正解とみなし，これと同じ構造を持つ解析木を正しい解析結果とする方法も考えられる．しかしながら，3.1 節で述べたように，EDR コーパスの括弧付けの中には多くの要素を 1 つの括弧で括ってしまうものも含まれている．これに対し，冗長な規則すなわち右辺長の比較的長い規則を削除した PCFG は，EDR コーパスに付加された括弧付けよりも細かく括弧付けする傾向を持っている．したがって，コーパスの構文構造と単純に比較して正しい解析結果か否かを判断するのは適切であるとは言えない.「括弧付けの適合率」及び「文の正解率」を計算する際に，コーパスに付加された構文構造と完全に一致していなくても,「矛盾しない括弧付け」及び「矛盾する括弧付けを含まない解析木」を正解としたのはこのためである.

まず，生成確率が 1 位の解析木について，括弧付けの再現率，括弧付けの適合率，文の正解率の値を計算した．結果を表 5 に示す.

表 5: 解析結果の評価 (1 位のみ)

括弧付けの再現率	括弧付けの適合率	文の正解率
54.30%	65.74%	8.47%

Schabesらは，英語の括弧付きコーパス(Wall Street Journal Corpus)からInside-Outsideアルゴリズムにより獲得した文法を用いた統語解析実験を行い，20〜30単語のテスト文に対して括弧付けの適合率が71.5%，文の正解率が6.8%であったと報告している[4](Schabes et al. 1993)．我々の実験の結果は，括弧付けの適合率ではSchabesらの結果に劣るが文の正解率では優っている．しかしながら，本研究とは使用しているコーパスや対象言語が異なるため，単純な比較はできない．

表5では，生成確率が1位の解析木についてのみ評価を行ったが，生成確率は統語的にみた解析木の尤もらしさを示しており，係り受け関係などの意味的な関係を考えた場合，生成確率の最も高い解析木が必ずしも正しい解析結果を表わしているわけではない．正しい解析結果を選択するのには何らかの意味解析が必要であるが，統語解析の結果出力される全ての解析結果の候補に対して意味解析を行うのは現実的ではない．統語解析を，意味解析を行う解析結果の候補の数を絞り込み，意味解析にかかる負担を軽減するための前処理と考えるなら，正解となる解析木の生成確率が1位とならなくても，生成確率の上位何位かに含まれていれば十分であろう．そこで，生成確率の上位k位の解析木を出力し，その中から矛盾する括弧付けの最も少ない解析木を選んで評価した．結果を表6に示す．

表 6： 統語解析結果の評価(上位k位)

k	括弧付けの再現率	括弧付けの適合率	文の正解率
1	54.30%	65.74%	8.47%
5	57.60%	69.04%	16.23%
10	59.53%	71.10%	21.11%
20	61.46%	73.18%	26.51%
30	62.48%	74.13%	29.06%

上位30位までの解析木を出力した場合，その中に正解となる解析木が含まれている文の割合は8.47%から29.06%に向上することがわかった．

最後に，統語解析を行う文法のサイズを変化させ，受理率と正解率および生成される解析木数との相関を調べる実験を行った．まず，G_1の中からある一定の閾値P以下の確率を持つ文法を除去し，サイズの小さい文法G_2〜G_5を抽出した[5]．次に，テストデータの中から枝刈りなしで受理した12,658文(表3参照)をG_2〜G_5を用いて統語解析し，結果を比較した．テスト文をこのように限定したのは，パーザがメモリ不足によって統語解析を中断した場合には文法が生成する解析木の数を測定することができないからである．解析した文の平均単語数は19.01単語であった．

4 彼らは括弧付けの再現率は示していない．
5 実際には，閾値以下の規則を削除した後，残された規則の出現回数をもとに各規則の確率の推定をやり直した．

実験結果を表 7 に示す．メモリ不足によって解析に失敗した文はなかった．括弧付けの再現率，括弧付けの適合率，文の正解率は，生成確率の 1 位の解析木のみについて評価した．表 7 により，文法サイズが小さくなるにつれて受理率が低下していることがわかる．また，受理率の低下に伴い平均解析木数も減少する傾向が見られる．これに対し，受理率が変化しても括弧付けの再現率，適合率，文の正解率はほとんど変化していない．このことから，受理率を向上させるために文法サイズを大きくして，その結果得られる解析木の数が増大しても，生成確率の上位の解析木のみを出力すれば正解率はほとんど変わらないということがいえる．

表 7: 文法サイズと解析結果の変化

文法 (閾値 P)	G_1 (—)	G_2 (10^{-5})	G_3 (10^{-4})	G_4 (10^{-3})	G_5 (10^{-2})
非終端記号数	41	37	34	23	15
終端記号数	111	80	57	33	23
規則数	2,219	1,289	871	390	115
受理率	100%	99.39%	95.57%	76.74%	34.22%
平均解析木数	2.730×10^7	1.810×10^7	1.071×10^7	9.841×10^5	4.485×10^4
括弧付けの再現率	62.71%	62.73%	62.66%	62.91%	60.11%
括弧付けの適合率	75.59%	75.62%	75.68%	76.58%	73.64%
文の正解率	12.07%	12.00%	12.25%	13.38%	11.45%

5　結論

本論文では，括弧付きコーパスから確率文脈自由文法(PCFG)を自動的に抽出する方法を提案した．PCFG の抽出は，日本語の主辞が句の一番最後の要素であるという特徴に着目し，括弧付けによる構文構造の内部ノードに適切な非終端記号を与えることによって行った．また，抽出した規則の確率はその規則のコーパスにおける出現回数から推定した．さらに，抽出した PCFG に対して 2 つの面から改良を加えた．1 つは文法サイズの縮小，もう 1 つは生成される解析木数の抑制である．前者は冗長な規則を削除することにより行った．後者は同一品詞列に対する構造を右下がりの二分木のみに限定したこと，品詞を細分化したこと，文末の助動詞に対する構造を統一したことにより行った．最後に，提案した方法により抽出・改良された PCFG を用いた統語解析実験を行ったところ受理率が約 92% となった．また，生成確率の上位 30 個の解析木を出力した場合，文の正解率が約 29%，括弧付けの再現率が約 62%，括弧付けの適合率が約 74% という結果が得られた．

最後に本論文の今後の課題について述べる．コーパスから抽出した PCFG の問題点の 1 つは，3.2 節で生成される解析木の数を抑制したにも関わらず，依然として多くの解析木を生成す

ることである．実験では，PCFGが出力する解析木数の1文当たりの平均は3.24×10^9 であった．生成確率の高い解析木のみを出力することにより解析結果の候補数を絞り込むことができるものの，文法が多くの解析木を生成するのは効率の面から見ても望ましいことではない．また，本論文では対象言語を日本語とし，句の主辞を特定する際に日本語の特性を考慮に入れているが，他の言語についても句の主辞を特定することができれば本手法をそのまま適用することができる．今後は，日本語以外の言語の文法を獲得することについても検討していきたい．

参考文献

Kiyono, M. and Tsujii, J. (1994). "Hypothesis Selection in Grammar Acquisition." In *Proceedings of the 14th COLING*, Vol. 2, pp. 837–841.

Kiyono, M. and Tsujii, J. (1993). "Linguistic Knowledge Acquisition from Parsing Failures." In *Proceedings of the 6th EACL*, pp. 222–231.

Kiyono, M. and Tsujii, J. (1994). "Combination of Symbolic and Statistical Approaches for Grammatical Knowledge Acquisition." In *Proceedings of the ANLP*, pp. 72–77.

小林義行, 徳永健伸, 田中穂積 (1996). "名詞間の意味的共起情報を用いた複合名詞の解析." 自然言語処理, **3** (1), 29–43.

Lari, K. and Young, S. (1990). "The Estimation of Stochastic Context-free Grammars Using the Inside-Outside Algorithm." *Computer speech and languages*, **4**.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). "Building a Large Annotated Corpus of English:The Penn Treebank." *Computational Linguistics*, **19** (2), 313–330.

三原健一 (1994). 日本語の統語構造. 松柏社.

森信介・長尾真 (1995). "統計によるタグ付きコーパスからの統語規則の獲得." 情報処理学会自然言語処理研究会, **110**, 79–86.

Pereira, F. and Schabes, Y. (1992). "Inside-Outside Reestimation from Partially Bracketed Corpora." In *Proceedings of the 30th ACL*, pp. 128–135.

Schabes, Y., Roth, M., and Osborne, R. (1993). "Parsing the Wall Street Journal with the Inside-Outside Algorithm." In *Proceedings of the 6th EACL*, pp. 341–347.

Sekine, S. and Grishman, R. (1995). "A Corpus-based Probabilistic Grammar with Only Two Non-terminals." In *Proceedings of the IWPT*.

白井清昭 (1995). "言語データからの自然言語処理用知識の自動獲得に関する研究." Master's thesis, 東京工業大学.

白井清昭, 徳永健伸, 田中穂積 (1995). "構文構造付きコーパスからの確率文脈自由文法の自動抽出." 情報処理学会第50回全国大会講演論文集, 3巻, pp. 61–62.

Tomita, M. (1986). *An Efficient Parsing for Natural Languages*. Kluwer, Boston, Mass.

宇津呂武仁・松本裕治 (1995). “コーパスを用いた言語知識の獲得.” 人工知能学会誌, **10** (2), 197–204.

Wetherell, C. (1980). “Probabilistic Languages: A Review and Some Open Questions.” *Computing surveys*, **12** (4), 361–379.

横田和章, 安部賢司, 青島直哉, 藤崎博也 (1996). “コーパスに基づく日本語文法の自動獲得.” 言語処理学会第2回年次大会発表論文集, pp. 169–172.

日本電子化辞書研究所 (1995). EDR電子化辞書仕様説明書 (第2 版).

略歴

白井 清昭: 1970年生．1993年東京工業大学工学部情報工学科卒業．1995年同大学院理工学研究科修士課程修了．1995年同大学院情報理工学研究科博士課程入学，現在在学中．コーパスからの自然言語処理用知識の自動獲得に関する研究に従事．情報処理学会会員．

徳永 健伸: 1961年生．1983年東京工業大学工学部情報工学科卒業．1985年同大学院理工学研究科修士課程修了．同年(株)三菱総合研究所入社．1986年東京工業大学大学院博士課程入学．現在, 同大学大学院情報理工学研究科計算工学専攻助教授．博士 (工学)．自然言語処理, 計算言語学に関する研究に従事．情報処理学会, 認知科学会, 人工知能学会, 計量国語学会, Association for Computational Linguistics, 各会員．

田中 穂積: 1941年生．1964年東京工業大学工学部情報工学科卒業．1966年同大学院理工学研究科修士課程修了．同年電気試験所(現電子技術総合研究所)入所．1980年東京工業大学助教授．1983年東京工業大学教授．現在, 同大学大学院情報理工学研究科計算工学専攻教授．博士(工学)．人工知能，自然言語処理に関する研究に従事．情報処理学会, 電子情報通信学会，認知科学会, 人工知能学会, 計量国語学会, Association for Computational Linguistics, 各会員．

(1996 年 4 月 11 日 受付)
(1996 年 6 月 11 日 再受付)
(1996 年 7 月 18 日 採録)