EDRコーパスからの確率文脈自由文法の自動抽出に関する研究

白井清昭　　徳永健伸　　田中穂積

東京工業大学 大学院情報理工学研究科 計算工学専攻

1　はじめに

最近では、自然言語処理に用いる様々な知識を人手で作成する代わりに、コーパスなどの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。このような研究が行われるようになった背景としては、計算機能力が飛躍的に向上したことや、EDR辞書やEDRコーパスなどの言語データの整備が進んだことがあげられる。大量の言語データから自動的に獲得した知識は、人手によって作ったものと比べて、適用範囲が広い、構築のための人的負担が軽い、人の主観が入りにくいなどの利点を持っている。

これまでに様々な自然言語処理用知識を自動獲得する手法が提案されているが、その中の1つに確率文脈自由文法を獲得するものがある。文法の自動獲得に関する研究としては、Inside-Outsideアルゴリズムによるものや[2, 3]、統語解析に失敗した事例から文法規則を獲得する研究[1] などがあるが、多くの計算量を必要とするなどの問題点も多い。本論文では、EDRコーパスから効率良く実用的な確率文脈自由文法を自動的に抽出する方法を提案する[5, 6]。

2　確率文脈自由文法の抽出

本研究では次の2つのEDRコーパスの補助情報を利用した。

- 品詞情報(各形態素毎に付加された品詞)
- 構文情報(各例文に付加された構文構造)

図1は、EDRコーパスの例文、およびそれに付加された括弧付けによる構文構造の例である。この

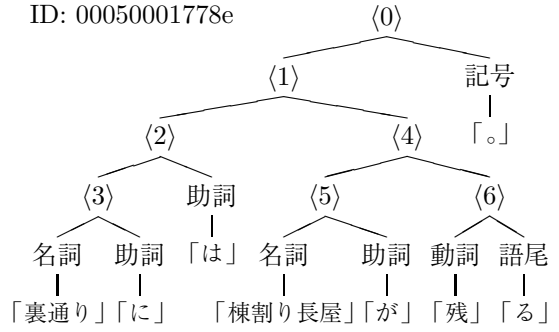

図 1: EDRコーパスの例文とその構文構造

構文構造は表1のような生成規則の集合に変換することができる。表1における数字は図1の内部ノードを表している。この内部ノードに対して適切な非終端記号を割り当てれば文脈自由文法の規則が抽出できる。ここでは、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日本語の特徴に着目し、次のような方法で内部ノードに非終端記号を与える。

- 右辺の記号列の一番最後の要素をその句の主辞とみなし、その主辞の末尾に “句” をつけた非終端記号を左辺に与える。また、主辞の末尾に既に “句” がついている場合には、右再帰を用いてそれをそのまま左辺に与える。

  [例]　名詞句　→　連体詞　名詞

  　　動詞句　→　後置詞句　動詞句

  ※　下線部が新たに与えた非終端記号

ところが、このやり方では常に適切な非終端記号を与えられるわけではない。そこで、次のような例外処理を行うことにした。

表 1: 構文構造から直接得られる生成規則

⟨6⟩	→	動詞　語尾
⟨5⟩	→	名詞　助詞
⟨4⟩	→	⟨5⟩　⟨6⟩
⟨3⟩	→	名詞　助詞
⟨2⟩	→	⟨3⟩　助詞
⟨1⟩	→	⟨2⟩　⟨4⟩
⟨0⟩	→	⟨1⟩　記号

- 右辺の一番最後の記号が “語尾” の場合：
  EDR コーパスでは、動詞、形容詞、形容動詞、助動詞の語尾には全て “語尾” という品詞が与えられている。したがって、左辺に “語尾句” という非終端記号を与えると、動詞句、形容詞句などの区別がつかなくなってしまう。そこで、この場合は左隣の記号を主辞とみなして左辺に非終端記号を与える。
  [例]　動詞句 → 動詞　語尾
- 右辺の一番最後の記号が “記号” の場合：
  “記号” は主辞にならない品詞であるとみなして、“語尾” と同じ処理を行う。
- 主辞が “助詞” の場合：
  左辺に非終端記号 “後置詞句” を与える。
  [例]　後置詞句 → 名詞　助詞
- 主辞が “接尾語” の場合：
  EDR コーパスでは、“接尾語” は「日」「メートル」などの単位が多く、句全体で名詞句を構成していると考えられるので、左辺に非終端記号 “名詞句” を与える。
  [例]　名詞句 → 名詞　接尾語
- 構文構造の根ノードには開始記号 “S” を与える。

左辺のノード番号に新しい非終端記号を与えると同時に他の規則の右辺にあるそれと同じノード番号に対しても同じ記号を与える。また、以上の操作は構文構造の葉から根ノードに向かってボトムアップに行う。表1 の各ノード番号に対して非終端記号を与えると表2 のような文法規則が抽出できる。これをコーパスの全ての例文に対して行うことにより文脈自由文法を抽出することができる。

表 2: 抽出された文法規則

動詞句	→	動詞　語尾
後置詞句	→	名詞　助詞
動詞句	→	後置詞句　動詞句
後置詞句	→	名詞　助詞
後置詞句	→	後置詞句　助詞
動詞句	→	後置詞句　動詞句
S	→	動詞句　記号

次に、抽出した規則の確率を推定する。本研究では、ある規則 $A \to \zeta_i$ の確率を、各規則 r のコーパスにおける出現回数 — $C(r)$ と表す — をもとに式 (1) のように推定する。

$$P(A \to \zeta_i) = \frac{C(A \to \zeta_i)}{\sum_j C(A \to \zeta_j)} \qquad (1)$$

3　文法の洗練

前節では、EDR コーパスから確率文脈自由文法を自動的に抽出する基本的な方法について説明した。しかしながら、このように抽出した文法には問題点がいくつか含まれている。本節では、これらの問題点を克服するために抽出した文法を洗練する方法について説明する。

3.1　文法サイズの縮小

2 節で述べた方法の問題点の 1 つとして、あまりに多くの文法規則が抽出されてしまうことが挙げられる。文法サイズを縮小するには出現頻度の低い規則を文法から除去する方法も考えられるが、この方法では文法の生成能力が低下してしまう可能性がある。ここでは文法の生成能力を低下させることなく文法サイズを縮小する方法を提案する。

まず、冗長な規則を次のように定義する。

冗長な規則

ある規則 $A \to \zeta$ について、文法中のそれ以外の規則を用いて非終端記号 A を記号列 ζ に展開できる場合、規則 $A \to \zeta$ は冗長である。

例えば、次の r_b, r_{c1}, r_{c2}, r_{c3} の規則が文法中に含まれていれば、“動詞句” という非終端記号を “動詞 語尾 名詞 助詞 形容動詞 語尾 動詞 語尾” と

いった記号列に展開することができるので、r_a は冗長な規則である。

r_a：	動詞句	→	動詞 語尾 名詞 助詞 形容動詞 語尾 動詞 語尾
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

冗長な規則を検出しこれを文法から削除すれば、文法の生成能力を損なう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、冗長な規則の出現回数をどのように取り扱うのかということである。(1)式に示すように、抽出した規則の確率は各規則のコーパスにおける出現回数をもとに推定している。したがって、冗長な規則を文法から削除したとしても、その冗長な規則のコーパスにおける出現回数は破棄するべきではない。先ほどの例で考えると、冗長な規則 r_a の出現回数は、r_a の右辺の記号列を生成する際に用いられる r_b, r_{c1}, r_{c2}, r_{c3} の出現回数に足し合わせるべきである。また、次の r'_b, r'_{c1}, r'_{c2} といった規則もまた文法中に存在する場合を考えてみよう。

r'_b：	動詞句	→	後置詞句 動詞句
r'_{c1}：	後置詞句	→	動詞 語尾 名詞 助詞
r'_{c3}：	動詞句	→	形容動詞 語尾 動詞 語尾

r_a の右辺の記号列は規則の集合 $\{r_b, r_{c1}, r_{c2}, r_{c3}\}$, $\{r'_b, r'_{c1}, r'_{c2}\}$ のいずれを用いても生成可能である。したがって、r_a の出現回数を r_b と r'_b の出現回数の比で分配し、それぞれ $\{r_b, r_{c1}, r_{c2}, r_{c3}\}$, $\{r'_b, r'_{c1}, r'_{c2}\}$ の出現回数に加えるべきである。冗長な規則の右辺の記号列を生成する規則の組が3つ以上存在する場合においても同様に、その規則の出現回数を r_b に該当する規則の出現回数の比で比例配分しなければならない。

冗長な規則を削除するアルゴリズムを以下にまとめる。ここで、もとの文法規則の集合を R、冗長な規則を削除した後の文法規則の集合を R_{new} とする。R_{new} の初期値は空集合である。

1. R の中から右辺長の最も長い規則を取り出し r_a とする。R が空集合なら終了。

2. r_a に対して、次の条件を見たす規則の組 $\{r_b^j, r_{c1}^j, \cdots, r_{cn}^j\}$ を R の中から可能な限り見つける。

$$
\begin{array}{ll}
r_a: & A \rightarrow \alpha_1^j\ \beta_1^j\ \alpha_2^j\ \cdots\ \alpha_n^j\ \beta_n^j\ \alpha_{n+1}^j \\
\hline
r_b^j: & A \rightarrow \alpha_1^j\ B_1^j\ \alpha_2^j\ \cdots\ \alpha_n^j\ B_n^j\ \alpha_{n+1}^j \\
r_{c1}^j: & B_1^j \rightarrow \beta_1^j \\
 & \vdots \\
r_{cn}^j: & B_n^j \rightarrow \beta_n^j
\end{array}
$$

3. このような規則の組が見つからなかったとき ($j = 0$)、r_a は冗長な規則ではないので R_{new} に加える。それ以外の場合は ($j \geq 1$)、r_a は冗長な規則であるので、r_a を R_{new} には加えず、$\{r_b^j, r_{c1}^j, \cdots, r_{cn}^j\}$ の出現回数を次のように更新する。 1. へ戻る。

$\forall\, i, j$:

$$C(r_b^j) \leftarrow C(r_b^j) + C(r_a) \times \frac{C(r_b^j)}{\sum_k C(r_b^k)}$$

$$C(r_{ci}^j) \leftarrow C(r_{ci}^j) + C(r_a) \times \frac{C(r_b^j)}{\sum_k C(r_b^k)}$$

3.2 統語的曖昧性の抑制

抽出した文法のもう一つの問題点は、非常に多くの統語的曖昧性を生じるということである。本節では、文法の統語的曖昧性を抑制する2つの方法について述べる。

3.2.1 同一品詞列の取り扱い

入力文の中に同じ品詞が連続して並んだ句が存在する場合には統語的曖昧性が増大すると考えられる。例えば、“名詞” が3つ連続して並んだ構造には図2の (a),(b) のようなものが考えられる。これらの構造はそれぞれの名詞間の関係を適切に表している。図2の (a) においては、「伊豆大島」が「噴火」を修飾し、「伊豆大島 噴火」全体が「災害」を修飾している。これに対して (b) の構造は、「災害」が「対策本部」を修飾し、「東京都」が「災害対策本部」全体を修飾している。しかしながら、“名詞 名詞 名詞” という品詞列が図2の (a),(b) のどちらの構造を取るのかを決定するには意味的な情報が必要となる。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、解析結果の候補と

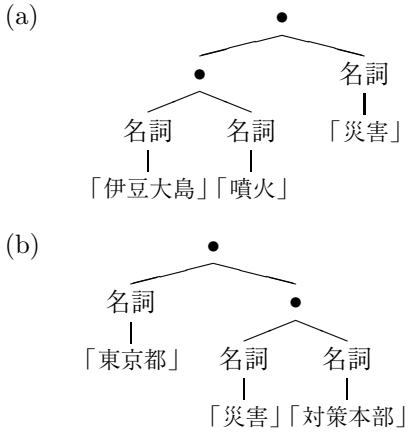

図 2: 名詞が 3 つ並んだときの構造

して両方の構造を出力してしまう。同じ品詞が 4 つ以上並んだ場合にはさらに多くの構造を出力し、また統語的曖昧性が組合せ的に増大することを考えると、このことが多くの解析木が出力される一因となっている。

そこで、同一品詞列の構造の解析は意味解析に任せることにし、統語解析の段階では図 3 のような右下がりの構造のみを生成することにより、統語的曖昧性を抑制することを試みた。このため、2

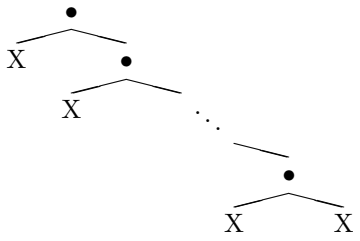

図 3: 右下がりの解析木

節の文法抽出アルゴリズムを以下のように変更し、同一品詞列に対して右下がりの構造だけを生成する文法を抽出した。

1. 訓練コーパスの構文構造の中から、葉が全て同じ品詞 X である部分木を見つけ、この部分木を非終端記号 “X 列” (e.g. 名詞列) に置き換える。“X 列” は図 3 の構造の根ノード及び内部ノードに割り当てる非終端記号である。
2. 2 節の文法抽出アルゴリズムを実行する。
3. 1. で見つけた部分木が n 個の葉を持つとき、次の 2 つの規則を文法に加える。

$$\text{X列} \quad \rightarrow \quad \text{X} \quad \text{X} \qquad (1)$$

$$\text{X列} \quad \rightarrow \quad \text{X} \quad \text{X列} \qquad (n-2)$$

また、それぞれの規則の出現回数に ( ) 内の数字を足し合わせる。

10,000 文の例文を用いた予備実験の結果、上記の操作により統語的曖昧性を約 20%減少させることができた。

3.2.2 品詞の細分化

EDR コーパスで用いられている品詞の数は 15 である。しかしながら、抽出した文法を統語解析に用いる場合には品詞の数は十分であるとはいえず、このことも文法の統語的曖昧性を増やす一因であると考えられる。ここでは “記号” と “助詞” の 2 つの品詞に着目し、これらを次のように細分化した。

記号の細分化

EDR コーパスにおいては、句点、読点、括弧類などに “記号” という品詞が割り当てられている。しかし統語解析を行う際には、句点や読点などの文の終末や区切りを表すものとそれ以外の記号を同じ品詞で表すのは望ましいことではない。そこで、次のようにして品詞 “記号” の細分化を行った。

- 形態素「、」には “読点” という品詞を与える。
- 形態素「。」及び「？」には “文末記号” という品詞を与える。EDR コーパスにおいては、これらの形態素は文末にしか現れない[1]。
- これら以外の “記号” という品詞を持つ形態素については品詞を変更しない。

助詞の細分化

EDR コーパスにおいては、「が」、「に」などの助詞に対しては全て “助詞” という品詞が割り当てられている。ところが、助詞は主格、対格、与格など様々な格を表しているので、これらに同じ品

[1]実際には、「－少しは慣れたか？と、彼は聞かない。」(ID: 00050008482e) の文中においてのみ「？」が文末以外の場所に現れるが、この文は例外としてコーパスから除去してある。

詞を割り当てるのは好ましくない。そこで、次のような方法で品詞 “助詞” の細分化を行った。

- 助詞のそれぞれの形態素に対してユニークな品詞を与える。例えば、形態素「が」に対しては “助詞が” という品詞を与える。

上記の操作は自動的に行うことができる。10,000例文を用いた予備実験において、訓練コーパスの品詞を細分化してから文法を抽出したところ、統語的曖昧性を約70%減少させることができた。

4 実験

本節では、本論文で提案する方法を用いてEDRコーパスから確率文脈自由文法を抽出し、それを用いて文の統語解析を行う実験について述べる。まず、EDRコーパスの75,000例文を分割し、1割をテストデータ、9割を訓練データとした。

訓練データから確率文脈自由文法を抽出したところ、表3に示すような文法が得られた。3.1節で提案した方法で冗長な規則を削除することにより、文法サイズを約23% に縮小することができた。

表 3: 抽出された確率文脈自由文法

非終端記号数	22
終端記号数／品詞数	161
規則数 (冗長規則を削除する前)	1781 (8011)

次に、この確率文脈自由文法を用いてテストデータの例文を統語解析する実験を行った。統語解析には一般化LR法[7] を用いた。また、確率文脈自由文法による確率モデルを用いて、生成確率の低い解析木を除去することにより統語的曖昧性を抑制した。具体的には、次の2つの方法を用いた。

1. 統語解析の際にビームサーチによる枝刈りを行った。すなわち、ある一定の閾値よりも低い生成確率を持つ部分木を削除することにより、生成される解析木の数を減らした。しかしながら、この方法では最大確率を持つ解析木の生成が妨げられる可能性がある。そこで、まず枝刈りを行わないで統語解析を行い、メモリ不足によって解析が中断した文に対してのみ枝刈りを行いながら統語解析をやり直すことにした。
2. 生成した全ての解析木を出力する代わりに、生成確率の高い上位15位の解析木のみを出力した。

表4に統語解析の結果を示す。acceptとはパーザが1つ以上解析木を出力した場合を、rejectはパーザが統語解析に失敗した場合を、overflowはメモリ不足によりパーザが統語解析を中断した場合を表す。accept率は約94%となり、適用範囲の広い文法が得られたことがわかった。

表 4: 統語解析結果

	枝刈りなし	枝刈りあり	(合計)
accept	5897	1034	6931
reject	12	3	15
overflow	—	428	428
(合計)	5909	1465	7374

$$\text{accept率} = \frac{6,931}{7,374} = 93.99\%$$

次に、パーザが出力した解析木の評価方法について説明する。まず、次の用語を定義する。

正しい括弧付け
解析木に含まれる括弧付けのうち、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ないもの。

正しい解析木
解析木中に含まれる全ての括弧付けが正しい括弧付けである解析木。

パーザが正しい解析木を出力した場合、その文の統語解析に成功したとみなす。また、統語解析の精度を調べるために次の尺度を導入する。

- 解析成功率
  $= \dfrac{\text{正しい解析木が得られた文の数}}{\text{acceptした文の数}}$
- 括弧付けの再現率
  $= \dfrac{\text{正しい括弧付けの総数}}{\text{コーパスの構文構造の括弧付けの総数}}$

- 括弧付けの適合率
$$= \frac{\text{正しい括弧付けの総数}}{\text{解析結果に含まれる括弧付けの総数}}$$

我々は、まず生成確率の高い上位15位の解析木の中から、矛盾した(正しくない)括弧付けが最も少ないものを1つ選択し、これらの選択された解析木のみについて解析成功率、括弧付けの再現率、適合率を調べた。

acceptした文について上記の値を調べたところ、表5のようになった。acceptした文の平均長は約22単語であった。Schabesら[4]は、Inside-Outsideアルゴリズムによって獲得した文法を用いて長さが20〜30単語の英語文を統語解析したところ、6.8%の解析成功率と71.5%の括弧付けの適合率が得られたと報告している[2]。我々の実験結果はこの結果よりも優れている。しかしながら、対象言語や実験に用いたコーパスが異なることから、両者を単純に比較することはできない。

表 5: 統語解析の精度

	枝刈りなし	枝刈りあり	(合計)
成功	1731	0	1731
失敗	4166	1034	5200
(合計)	5897	1034	6931

$$\text{解析成功率} = \frac{1,731}{6,931} = 24.97\ \%$$

$$\text{括弧付けの再現率} = \frac{105,485}{124,377} = 84.81\ \%$$

$$\text{括弧付けの適合率} = \frac{105,485}{141,067} = 74.78\ \%$$

5 まとめ

EDRコーパスの構文構造の内部ノードに自動的に非終端記号を与えて確率文脈自由文法を抽出する方法を提案した。また、抽出した確率文脈自由文法を改善する方法をいくつか提案した。1つは冗長な規則を削除することにより、文法の適用範囲を損なうことなく文法サイズを縮小する方法、もう一つは文法の統語的曖昧性を抑制する方法である。後者の具体的な方法としては、同一品詞の並びに対して右下がりの部分木のみを与えることと、“記号” と “助詞” の品詞を細分化することを提案した。さらに、EDRコーパスの例文をテストデータと訓練データに分け、訓練データから文法を抽出してテストデータの文を統語解析する実験を行った。その結果、適用範囲の広い文法が得られたことを確認した。

今後の課題としては、3.2節で統語的曖昧性を抑制したのにも関わらず、依然としてかなりの統語的曖昧性が生じてしまうことがあげられる。4節の実験において、1つの文に対して平均10^9個の解析木が得られることがわかっている。今後は、適用範囲や解析精度を維持しながら統語的曖昧性を抑制する方法を探求していく予定である。

謝辞 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に感謝いたします。

[2]括弧付けの再現率については述べられていない。

参考文献

[1] M. Kiyono and J. Tsujii. Hypothesis selection in grammar acquisition. In *Proceedings of the 14th International Conference on Computational Linguistics*, Vol. 2, pp. 837–841. COLING '94, 1994.

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

[3] Fernando Pereira and Yves Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pp. 128–135. ACL '92, 7 1992.

[4] Yves Schabes, Michael Roth, and Randy Osborne. Parsing the wall street journal with the inside-outside algorithm. In *Proceedings of the 6th Conference of European Chapter of the Association for Computational Linguistics*, pp. 341–347. EACL '93, 1993.

[5] 白井清昭, 徳永健伸, 田中穂積. コーパスからの文法の自動抽出. 情報処理学会自然言語処理研究会, Vol. 101, No. 11, 5 1994.

[6] 白井清昭, 徳永健伸, 田中穂積. 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, 第3巻, pp. 61–62. 情報処理学会, 3 1995.

[7] M. Tomita. *An Efficient Parsing for Natural Languages*. Kluwer, Boston, Mass, 1986.