1 例文を使って文の解析をしよう

乾 健太郎 inui@ai.kyutech.ac.jp
九州工業大学情報工学部知能情報工学科

白井 清昭 kshirai@cl.cs.titech.ac.jp
東京工業大学大学院情報理工学研究科

「文を解析する」とは

言語理解と構文解析 ——

周知のように，近年のIT（Information Technology）産業の急速な成長は社会を大きく変革しつつある．これに伴い，しばしば「洪水」に喩えられる情報過多の問題も加速的に拡大し，ビジネスシーンだけでなく日常生活にも顕在化するようになった．このため，大量の文書から目的の文書を検索したり，検索した文書を要約したり，翻訳したりする高度で大規模な自然言語処理技術の必要性がますます高まっている．自然言語の文を解析する技術，中でも本稿で取り上げる構文解析技術は，こうした自然言語処理技術の基盤となる要素技術である．

自然言語の文が与えられたとき，その文が表す意味やその背後にある書き手（あるいは話し手）の意図を汲みとることを「文を理解する」あるいは「文を解析する」という．文を解析する処理は，利用する知識や情報の観点から大きく次の4つのレベルに分けられる．

形態素解析：辞書や語形変化規則などの語彙的な知識と語の並びのパターンに関する知識を参照して行える範囲の処理．入力文の単語列を認識し，個々の単語の品詞の候補を絞り込む．

構文解析：文法的知識や語の用法に関する知識を参照して行える範囲の処理．入力文の構文構造の候補を探索し，構文的整合性に照らして候補を絞り込む．

意味解析：語の意味や語と語の意味的な関係に関する知識を参照して行える範囲の処理．文の意味的な解釈（意味構造）の候補を生成し，意味的整合性に照らして候補を絞り込む．

文脈解析：文の前後の文脈情報を利用して解釈をさらに絞り込み，最終的に一意に決定する．

本稿では，このうち構文解析に焦点を絞り，研究の現状を紹介する．

構文解析の目標は，入力文の構文構造を特定する，あるいは構文構造の候補を絞り込むことである（これを「構文的曖昧性の解消」という）．上の分類では文法的知識や語彙知識を使ってできる処理を構文解析としたが，実際の構文的曖昧性解消では意味的知識や文脈情報を必要とする場合が少なくない．このことは**図-1**の例からも想像されよう．そこで広義には，意味的知識や時には文脈的情報を利用する処理を含めて，構文構造を特定する作業全体を「構文解析」と呼ぶ．広義の構文解析では，構文構造の候補を探索する処理に比べ，候補を絞り込む処理（曖昧性解消）の方が問題の重要性が高い．本稿でも後者の処理に重点を置き，古典的な手法と近年活発に研

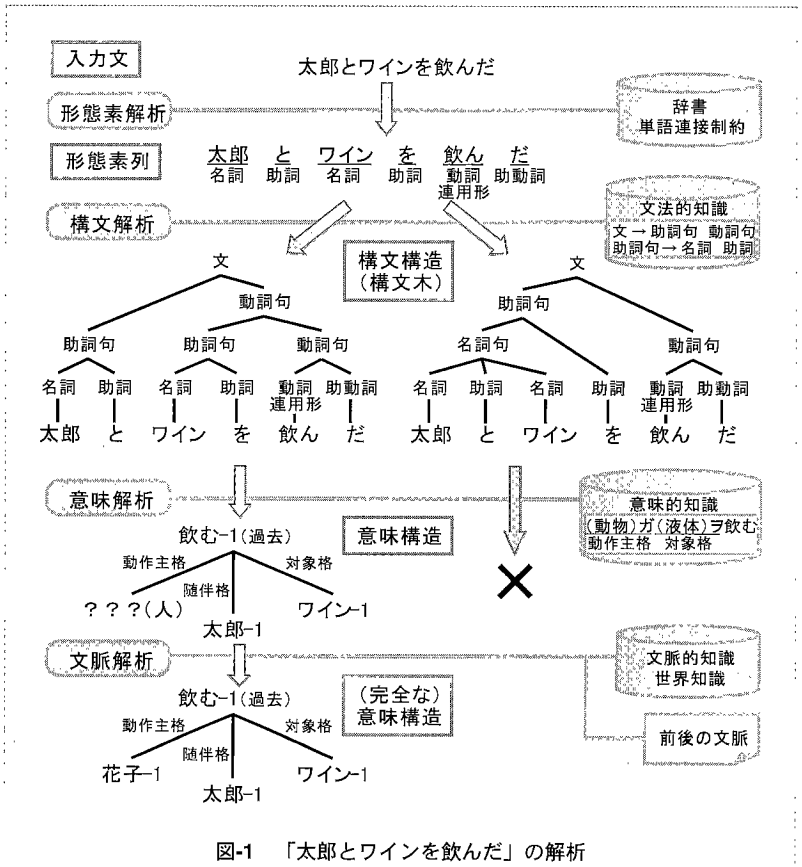

図-1 「太郎とワインを飲んだ」の解析

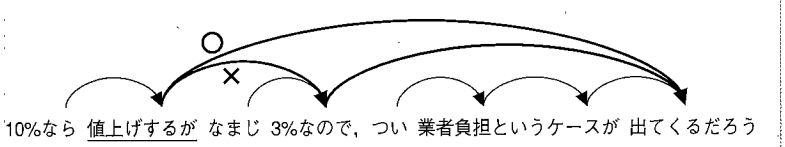

図-2　文節間の係り受け構造

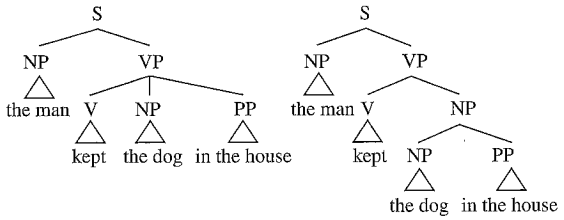

図-3　構文的な曖昧性

究されているコーパス（例文集）を利用する手法を紹介する．

構文解析の用途

構文解析は自然言語処理の基盤となる要素技術である．その用途は多岐にわたる．

第1に，構文解析は言語理解に不可欠な部分タスクである．構文解析技術なくしては，翻訳や要約，情報抽出といった，何らかの言語理解を必要とする高度な言語情報処理は実現できない．たとえば，「太郎とワインを飲んだ」を“I drank Taro and wine.”と誤訳せずに“I drank wine with Taro.”と正しく翻訳するためには，正確な構文解析が必要である．また，要約生成では，テキストから不要な修飾語句を削除する処理が必要になるが，文の構文構造を無視して削除すると，文法的におかしな文を生成するかもしれない．

第2に，構文解析の結果は，文と文，あるいはテキストとテキストの意味的な類似性を評価するための重要な手がかりになる．言語表現間の意味的類似性を評価する技術にはさまざまな応用がある．たとえば，文書検索では，ユーザが自然言語で入力する検索要求と文書データベース中の各文書との意味的な類似性／関連性を計算し，関連性の高い文書を選択する．出現単語の頻度分布のような表層的な情報を使って評価するのが今のところ一般的だが，構文構造も類似性評価の効果的な手がかりと考えられる．また，類似翻訳例検索に基づく翻訳支援においても，翻訳対象文と翻訳例の類似性計算が必要になるので，事情は文書検索の場合と同じである．

第3に，知識獲得の道具としての構文解析の有用性も見逃すべきでない．電子化文書の増大とともに，そこから言語処理に必要な大規模な知識を自動的あるいは半自動的に獲得する方法が活発に研究されている．言語データからの知識獲得は，言語処理技術を実用レベルに引き上げるための急務の課題である．ここでも構文解析は重要な役割を担う．これについては本特集の「コーパスが先か，パーサーが先か」，「言語コーパスをより有効に使うために」を参照されたい．

☆1　文節は日本語に特有の文法カテゴリで，一般には（接頭辞)*（自立語)+（接尾辞または付属語)*で定義される語の並びを表す（ただし，*，+はそれぞれ0回，1回以上の繰り返し）．

古典的な手法とその限界

はじめに，人手で作成した規則を用いて曖昧性を解消する古典的な手法を紹介しよう．

構文構造の候補を規定する制約

入力文を受理する構文構造のうち，構文的制約に無矛盾なものが構文構造の候補である．代表的な構文的制約は文法である．文法の記述形式には，文脈自由文法（Context Free Grammar; CFG），文脈依存文法（Context Sensitive Grammar; CSG），木接合文法（Tree Adjoining Grammar; TAG）などがある．このうちCFGがこれまで最もよく使われてきた．CFGには高速な構文解析アルゴリズムが存在するためである．CFGの具体例については「確率文脈自由文法に基づく統計モデル」(p.766)の項を参照されたい．

日本語の構文解析では，その部分問題として文節係り受け解析がしばしば取り上げられる．文節係り受け解析は，**図-2**のような文節間の係り受け構造を特定する問題である☆1．係り受け構造の候補を規定する制約は，文節クラス間の係り受け可能性として記述できる．文節クラスには，〈名詞＋ガ〉，〈名詞＋ヲ〉，〈動詞連用形〉，〈動詞連体形〉，〈副詞〉などが考えられる．係り受け可能性は，たとえば〈名詞＋ガ〉は〈動詞連用形〉や〈動詞連体形〉に係るが，〈名詞＋ガ〉や〈名詞＋ヲ〉には係らないといった制約である．

規則に基づく曖昧性解消

候補を絞り込むための規則には，個別の単語に依存する規則と，より一般性の高い構文的優先度に関する規則がある．

構文的優先度に関する規則

英語の構文的優先度に関する代表的な規則にminimal attachmentとright associationと呼ばれるヒューリスティクスがある．前者は「より簡単な構文木（中間節点の少ない構文木）を優先する」という原則で，後者は「より近いものに係る解釈を優先する」という原則である．たとえば，**図-3**の曖昧性にminimal attachment規則を適用すると，左側の正し

い解釈が優先される．後者の傾向は日本語にも見られる．新聞記事を用いた文節係り受け解析の実験によると，「より近いものに係る解釈を優先する」という規則を用いるだけで，65%近い精度（文節単位の正解率）が得られることが分かっている[1]．

日本語ではこのほかに，言語学の成果に基づいて作成された，従属節の係り先に関するヒューリスティクスが知られている．それによると，従属節を次の種類に分類し，A類＜B類＜C類のように優先度を仮定した場合，「従属節は自分よりも優先度の低い従属節には係らない」という強い傾向がある[2]：

A類：「～とともに，～ながら」など，〈同時〉を表す従属節
B類：「～ため，～て，～ので」など，〈原因〉，〈中止〉を表す従属節
C類：「～が」で終わる従属節

この原則を図-2の例の「値上げするが（C類）」に適用すると，正解でない係り先の候補「3%なので（B類）」をうまく落とすことができる．

選択制約と意味素性

構文的曖昧性の中には，構文的優先度に関する規則だけでは正しく解消できないものも多い．次の例を考えよう．

(a) Mary bought a plant with John.
(b) Mary bought a plant with yellow leaves.

(a) と (b) の違いは文末の前置詞句の中の名詞句（“John” と “yellow leaves”）だけである．にもかかわらず，(a) の前置詞句の係り先は “bought” であり，(b) の係り先は “a plant” が正しい．この種の問題は前置詞句係り先決定（PP attachment）問題と呼ばれる．この例からも明らかなように，PP attachment問題は構文的優先度の規則だけではうまく扱えない．

日本語にもこれと同様の問題がある．たとえば次の例では，「その女性に」が「せまる」に係る可能性と「出合った」に係る可能性があるが，やはり構文的な手がかりだけではどちらが正しいか判断できない．

私はその女性に夕闇がせまる街角で再び出合った

これらの問題の難しさは，語の意味的な情報を利用しないと解消できない点にある．

語の意味的な情報の代表的な例としては選択制限（selectional restriction）が挙げられる．選択制限とは，単語の共起（どのような単語がどのような単語と文中に一緒に現れるか）に対する意味的制約である．選択制限を使えば，構文構造の各候補についてある種の意味的整合性をチェックすることができる．

1. N1[hum/org] ガ N2[ani/phe/act] ニ 出合う
（不意に望ましくない状況に出合う）
2. N1[hum] ガ N2[hum] ニ 出合う
（不意に誰かに会う）

図-4 選択制限の記述例（IPAL動詞辞書）

選択制限の例として，IPAL動詞辞書[3]，☆2の「出合う」の記述例を**図-4**に示す．これによると，動詞「出合う」は2つの語義を持つ．語義1の場合は，「N1ガN2ニ出合う」という構文パターンをとり，N1にはhum（人間）かorg（組織）を表す名詞，N2にはani（動物）かphe（現象）かact（動作）を表す名詞が入る．語義2も同様である．humやorgは名詞の意味的性質を表すラベルで，意味素性あるいは意味クラスと呼ばれる．IPAL動詞辞書では，動詞と名詞の選択制限の記述に18種類の意味素性を使っている．

このような選択制限の知識と個々の名詞が持つ意味素性の知識を使えば，前述の「その女性に」の曖昧性もうまく解消できる．「女性ニ街角デ出合う」の解釈は「出合う」の選択制限を満たすが，「女性ニ夕闇ガせまる」の解釈は「せまる」の選択制限を満たさない．

人手による規則開発の限界 ——

規則の開発者はさまざまな事例を横並びで比較し，そこに見られる傾向を一般化することによって規則を作成する．しかし，人手で規則を開発する方法には深刻な問題がある．

まず，他の知識工学の分野と同様に，観察された現象をどの程度，どのように一般化すべきかの判断がきわめて難しい．たとえば，「出合う」のニ格になり得る名詞には「友達，群，災難，反撃」などがあるが，これらをどのような意味素性に一般化すべきかを決めるのは容易でない．IPAL辞書では図-4に示したような意味素性に一般化しているが，実際のテキストを探すと「グラフィックに出合う」のような例外が簡単に見つかってしまう．質の高い規則を網羅的に作成するには，それに見合う十分な数の事例を調査し，適当な粒度の意味素性を用いて一般化する必要がある．しかし，人間が横並びで比較できる事例の数には限りがあるので，ことは簡単でない．

工学的には，規則の開発・保守のコストの問題も重要である．人手による規則作成は，次章で述べるような機械的な知識獲得に比べると，明らかにコス

☆2 情報処理振興事業協会技術センター（IPA）が計算機での利用を前提に開発した機械可読辞書．

(1) 文 → 助詞句 動詞句 ; 1
(2) 助詞句 → 名詞 助詞 ; 0.7
(3) 助詞句 → 名詞句 助詞 ; 0.3
(4) 動詞句 → 動詞連用形 助動詞 ; 0.6
(5) 動詞句 → 助詞句 動詞句 ; 0.4
(6) 名詞句 → 名詞 助詞 名詞 ; 1

図-5 確率文脈自由文法の例

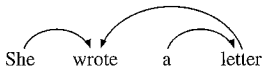

図-6 依存構造の例

トが高い．それでも，一度完全なものを作ってしまえばそれで済むというのであれば，高いコストを払う意味があるかもしれない．ところが，実際はそうでない．構文的あるいは意味的な制約や優先度は，対象となるテキストのジャンルによって大きく異なる．新聞記事によく現れる構文や格パターンは電子メールのそれとはかなり違う．対象テキストのジャンルにうまく合った規則を用意しないと，解析システムの性能は上がらない．規則開発・保守のコストが高いと，ジャンルごとに規則をきめ細かくチューニングする作業が難しいので，結果的に高い解析性能が得られないことになる．

では，人手による規則開発の限界を克服するにはどうすればよいか？ 糸口の1つは，大量の例文から知識を自動的に学習することである．以下，これについて述べる．

コーパスに基づく構文解析

社会の情報化の進展とともに，大量の電子化文書が安価に入手できるようになった．これに伴い，大規模な例文集を用いて構文的曖昧性を解消する方法が活発に研究されている．例文の集まりを一般に言語コーパス，あるいは単にコーパスと呼ぶ．「コーパスに基づく構文解析」は，例文集を利用して構文的曖昧性を解消する方法の総称である．コーパスに基づく構文解析は大きく2つに分けられる．1つは統計モデルを用いる構文解析（統計的構文解析），もう1つは類似用例を用いる構文解析（類推的構文解析）である．

統計的構文解析 ——

構文解析にコーパスを用いる最も単純な方法は，入力文の解析結果（構文構造）の候補に対して，それと同じ文が同じ構文構造を持ってコーパス中に現れる頻度を調べ，その頻度の高いものから優先順位をつけることであろう．しかしながら，現実の文の多様性を考えれば，入力文とまったく同じ文をコーパスから見つけ出すことすら不可能に近い．

そこで，コーパスにおける構文構造の出現頻度を近似的に求める統計的なモデルを考える．統計モデルがうまく設計できれば，モデルのパラメタはコーパスから学習できる．統計モデルを用いる手法は，「コーパスによく現れる構文構造が正しい構文構造である」という，いわば多数決の原理に基づいて構文的曖昧性を解消しているとみなせる．

統計モデルには，大きく分けて，確率文脈自由文法に基づく統計モデルと依存構造に基づく統計モデルの2種類がある．なお，これらの統計モデルについては文献4）が詳しい．

確率文脈自由文法に基づく統計モデル

文脈自由文法（CFG）は構文的制約の記述形式である．CFGの例を**図-5**に示す．

$A \rightarrow \zeta$ という形式の規則（1）～（6）は，それぞれ書き換え規則と呼ばれ，左辺Aの構成要素が右辺の構成要素の列ζによって構成されることを表す．たとえば，規則（1）は，「文」が「助詞句」と「動詞句」から構成されることを表している．

確率文脈自由文法（Probabilistic Context Free Grammar; PCFG）は，CFGの書き換え規則$A \rightarrow \zeta$に対して，左辺Aが与えられたとき，それが記号列ζに書き換えられる条件付確率$P(\zeta|A)$を付与したものである．図-5では，";" の右の数値がその規則の適用確率を表す．規則の適用確率は，構文木が例文に付与された構文構造付コーパスを用いれば容易に学習することができる．すなわち，コーパス中の構文木の集合から，各規則が使われている回数$C(A \rightarrow \zeta)$を求め，規則の適用確率を以下のように最尤推定すればよい．

$$P(\zeta|A) = \frac{C(A \rightarrow \zeta)}{\Sigma_i C(A \rightarrow \zeta_i)} \quad (1)$$

構文木の生成確率は，構文木の生成に用いられた規則の適用確率の積として計算される．たとえば，図-1の左の構文木の生成に使われた規則は（1），（2），（5），（2），（4）であり，その適用確率の積は0.1176である．一方，図-1の右の構文木の生成に使われた規則は（1），（3），（6），（4）であり，その適用確率の積は0.18である．このようにして得られる構文木の生成確率は，コーパスにおける構文木の出現頻度を反映したスコアとみなすことができ，解の優先順位付けに利用できる．

確率文脈自由文法の語彙化

PCFGが与える優先順位はいつも正しいとは限らない．たとえば，図-5のPCFGを図-1の例に適用すると，右側の正しくない方の構文木に正解（左側）より高い生成確率を与えてしまう．そこで，PCFGを改良し，近似の精度を上げることによって，曖昧性解消の正解率を向上させる試みが数多く行われている．ここでは，その具体例として，PCFGの語彙化を紹介する．

図-5の規則（6）は，「AとB」（A，Bは名詞）のような名詞の並列構造に対応した規則である．ところが，この規則が適用される確率は，AとBにどのような名詞が現れるかによって大きく異なる．「日本酒とワイン」のように，A，Bが似た意味を持つ単語の場合，規則（6）の適用確率は高い．一方，図-1の「太郎とワイン」のように，A，Bの意味があまり似ていなければ，この規則の適用確率は低くなる．ところが，図-5のPCFGでは，名詞はすべて「名詞」という記号に抽象化されているので，単語に依存した規則の適用確率の違いを構文木の生成確率に反映させることはできない．

このような問題は，PCFGを語彙化することによってある程度解決できる．PCFGの語彙化とは，たとえば以下のように，文法中の記号を単語によって細分化することを指す．

(6-1) 名詞句→名詞:太郎 助詞:と 名詞:ワイン; 0.001
(6-2) 名詞句→名詞:日本酒 助詞:と 名詞:ワイン; 0.05

語彙化したPCFGを用いて図-1の構文木の生成確率を計算すると，規則（6-1）の適用確率が十分低いため，左の正しい構文木の生成確率が右の構文木よりも高くなる．このように，文法中の記号を単語で細分化すれば，単語に応じて変化する規則の適用確率の違いが構文木の生成確率に反映されるため，より精密なモデルを構築することができる．

ただし，PCFGを語彙化すると，書き換え規則の数が組合せ的に増大する．そのため，推定すべきパラメタ（規則の適用確率）の数も増大し，学習用データが不足するという問題，すなわちデータの過疎性の問題が生じる．データの過疎性の問題を解決するためのさまざまな試みについては，本特集の「言語コーパスをより有効に使うために」の解説を参照していただきたい．

依存構造に基づく統計モデル

文の構文構造を表現するものとして，図-1のような構文木のほかに，依存構造と呼ばれる構造がある．依存構造は，文中の単語がどの単語を修飾するかを弧によって表現したものである．たとえば，“She wrote a letter.”という英文の依存構造は**図-6**のようになる．図-6の各弧$w_i \rightarrow w_j$は，単語w_iが単語w_jを修飾することを示す．ここで，依存構造Dのコーパスにおける出現のしやすさを表す尺度として，個々の依存関係が成立する確率$P(w_i \rightarrow w_j)$の積を考える（式（2））．

$$Score(D) = \prod_{w_i \rightarrow w_j \in D} P(w_i \rightarrow w_j) \qquad (2)$$

$P(w_i \rightarrow w_j)$は式（3）のように推定できる．$C(w_i, w_j,$ 依存），$C(w_i, w_j,$ 非依存）は，それぞれ，同一文中にある単語w_iとw_jの間に依存関係が存在するイベント，存在しないイベントのコーパスにおける出現頻度である．

$$P(w_i \rightarrow w_j) = \frac{C(w_i, w_j, \text{依存})}{C(w_i, w_j, \text{依存}) + C(w_i, w_j, \text{非依存})} \qquad (3)$$

こうして得られる式（2）の値は，PCFGによる構文木の生成確率と同様に，コーパスにおける依存構造の出現頻度を反映したスコアとみなすことができ，解の候補の絞り込みに利用できる．

日本語を対象とした構文解析では，図-2のような文節間依存構造（係り受け構造）によって文の構文的な構造を表現することも多い．文節間依存構造に対するスコアは，単語の依存構造と同様に，個々の文節b_iが文節b_jを修飾する確率の積によって計算できる（式（4））．

$$\prod_{b_i \rightarrow b_j \in D} P(b_i \rightarrow b_j) \qquad (4)$$

実際には，文節の組合せは非常に多様であるため，確率$P(b_i \rightarrow b_j)$を直接推定することは難しい．そのため，各文節b_iを，その文節を特徴付けるいくつかの素性の集合F_iに抽象化し，$P(b_i \rightarrow b_j)$を$P(F_i \rightarrow F_j)$として推定する方法が一般的である．F_iの要素として用いられる素性としては，文節の意味的な主辞（文節を構成する単語のうち，その文節の意味を最もよく表す単語），主辞の品詞，活用語の活用形，文節末にある助詞，読点の有無，文節間の距離（b_iとb_jの間に存在する文節の数）などがある．

類推的構文解析 ——

コーパスに基づく構文解析では，統計モデルのパラメタを学習する代わりに，コーパスに付与された構文構造を直接的に利用する類推的構文解析と呼ばれる手法がある．類推的構文解析とは，入力文と似た文，すなわち類似用例をコーパスから検索し，それに付与されている構文構造を加工して入力文の構文構造を出力する手法である．ここでは，類推的構文解

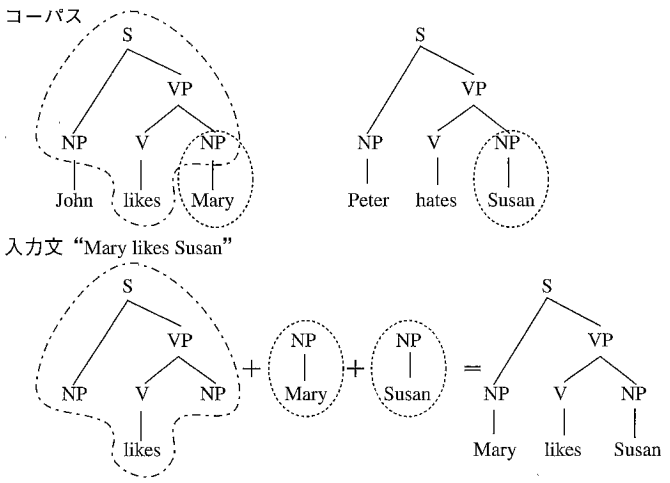

図-7　Data Oriented Parsingの例

析の一例として，Data Oriented Parsing（DOP）[5]を紹介する．

DOPでは，文法などの構文的な制約はまったく用いずに，コーパス中の構文木に含まれる部分木を組み合わせて入力文の構文木を決定する．たとえば，“Mary likes Susan”を構文解析する際には，コーパス中に存在する例文“John likes Mary”，“Peter hates Susan”の構文木に含まれる3つの部分木を組み合わせて，入力文の構文木とする（**図-7**）．すなわち，入力文と同じ単語を含む部分木（用例）をコーパスから検索し，それらを組み合わせて入力文の構文木を決定する．構文木を生成する部分木の組合せが複数存在する場合には，コーパスにおける部分木の出現頻度に基づいて最適な組合せを1つ選択する．コーパスの規模が大きくなると部分木の組合せの数は膨大になるので，DOPではモンテカルロ技法と呼ばれる手法を用いて効率的に準最適解を求める工夫をしている．

コーパスに基づく構文解析の現状と課題

コーパスに基づく構文解析は，どれくらい正確に文を解析できるのだろうか．英語文を対象とした構文解析については，ペンシルバニア大学が作成したPenn Treebankを用いた実験が数多く報告されている．Penn Treebankは，雑誌Wall Street Journalなどから収集した構文構造付コーパスである．報告されている実験結果によると，正しく解析できる文の割合は68%程度である[6],☆3．一方，日本語を対象とした実験では，新聞記事の文を対象とし，図-2のような文節間依存構造を求めるという問題設定が多い．これまでの報告によると，文節単位の正解率で88%程度，文単位の正解率で40%程度である[1]．いずれにせよ，新聞記事や雑誌などを対象とした場合，コーパスに基づく構文解析の正解率はそれほど高くはないというのが現状である．

では，正解率を向上させるためには，どのような方法が考えられるだろうか．1つは，コーパスの規模を拡大することである．ここで紹介した構文解析手法は，各例文に構文木や依存構造が付与された構文構造付コーパスを必要とする☆4．現在利用できる構文構造付コーパスとしては，英語ではPenn Treebank（約43,000文），日本語ではEDRコーパス（約200,000文）や京大コーパス（約20,000文）などがある．しかしながら，高い正解率を得るためには，これら既存のコーパスだけでは必ずしも十分ではなく，より大規模なものを作成する必要がある．また，作成した構文構造付コーパスを公開し，研究者の誰もが利用できる体制を整えることも重要であろう．

2つ目は，利用可能なコーパスの量が十分でないことを仮定し，少ないデータをうまく利用して構文的な曖昧性を解消する技術を開発することである．構文構造付コーパスの作成は，人手による構文構造のチェックが必要なので作成コストが高い．そのため，利用可能な構文構造付コーパスの量がすぐに100倍または1000倍になるとは考えにくい．限られたコーパスを最大限に効率よく利用することが重要になる．

3つ目は，人手によって作成された規則とコーパスから得られた知識を融合することである．人手によって作成された規則は信頼性が高いが，構文的曖昧性解消に有効なあらゆる規則を網羅的に作成することは困難である．一方，大規模なコーパスから得られる統計モデルや類似用例は，広範な言語現象をカバーしているという点で網羅的ではあるが，人手で作成する場合ほど正確な知識が得られるわけではない．この2つは相反するものではなく，むしろお互いの欠点を補う関係にあると考えるべきである．両者をいかに組み合わせるべきかについては，今後積極的に取り組むべき課題であろう．

☆3　ここでいう正しい解析結果とは，コーパスに付与された構文木と矛盾しない構文木を指す．したがって，コーパスに付与された構文木と完全に一致したときを正解とする場合よりも緩い評価基準となっている．この評価基準の詳細については文献6）を参照していただきたい．

☆4　統計モデルのパラメタを学習する場合には，構文構造が付与されていないコーパスを用いる方法も提案されているが，これはいわゆる教師なし学習に相当するため，得られる統計モデルの信頼性も低い．

参考文献

1) 内元清貴, 関根　聡, 井佐原均: 最大エントロピー法に基づくモデルを用いた日本語係り受け解析, 情報処理学会論文誌, Vol.40, No.9, pp.3397-3407 (Sep. 1999).
2) 白井　諭, 池原　悟, 横尾昭男, 木村淳子: 階層的認識構造に着目した日本語従属節間の係り受け解析の方法とその精度, 情報処理学会論文誌, Vol.36, No.10, pp.2353-2361 (Oct. 1995).
3) 情報処理振興事業協会: 計算機用日本語基本動詞辞書IPAL (Basic Verbs) (1987).
4) 北　研二: 確率的言語モデル, 東京大学出版会, 東京 (1999).
5) Bod, R.: Using an Annotated Language Corpus as a Virtual Stochastic Grammar, Proceedings of the 11th National Conference on Artificial Intelligence, pp.778-783 (1993).
6) Charniak, E.: A Maximum-Entropy-Inspired Parser, Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics, pp.132-139 (2000).

(平成12年5月23日受付)