2 コーパスが先か，パーサーが先か

黒橋 禎夫 kuro@i.kyoto-u.ac.jp
京都大学大学院情報学研究科知能情報学専攻

コーパスベースがはやるまで

言葉は，表面的には音素や文字の一次元の並びであるが，内部には構造を持っており，それは**図-1**のような木構造で表現することができる．このような言語の構造を明らかにする計算機処理を構文解析（パージング）と呼び，これを行うシステムをパーサーと呼ぶ☆1．言語の構造の把握は言語の理解の第一歩である．これまでの自然言語処理では，よいパーサーを作ることが最大の課題であり，最大の難問であった．

はじめは，人間が言語直感によって規則を書くという方法がとられたが，それでは言語現象を十分にカバーする，使い物になるパーサーはなかなかできないということが分かってきた．

そこで，実際の言語現象をよく見よう，すなわち大量のテキスト（コーパス）をよく見ようということになり，さらに，コーパスの各文に正しい構文情報を人手で与えておいて（これを以下では構文構造付コーパスと呼ぶ），そこからパーサーの規則を自動的に学習するということが行われるようになった．このようなコーパスベースの研究はアメリカのPennsylvania大学のPenn Treebankプロジェクト[1)]で最初に大規模に行われ，その後，自然言語処理における一大ブームとなった．

コーパスが先か，パーサーが先か

コーパスベースの手法の最大の問題は「コーパスに正しい構文情報を人手で与える」という点にある．それは人間があまり得意としない単調な大量の作業であり，注意深く行ったとしてもコーパスに誤った構文情報を与えてしまうことがどうしても起こる．そして，そのような誤りがあるレベル以上含まれると，構文構造付コーパスとしての意味がなくなってしまう．

日本語の場合にはこの問題が特に深刻である．日本語の文は「～して，～して，～した」というように連用節が多く，また主語が頻繁に省略され，構造が明確にとらえにくい．さらにもっと基本的な問題として，そもそも語の区切りが明確でなく，接尾辞，助動詞，補助動詞などの付属的な語が多く，複数の語からなる機能的な表現が非常に多い．「～かどうかにかかわらず」「～だけでないとはいえ」のような表現に対して，どう語を区切るか，どういう構造を与えればよいかということは簡単ではない．

このような，ややこしい日本語文に対して正確で一貫性のある構文情報を与えるためには，まずパーサーによってある程度正しい情報を自動的に付与し，その結果を人間が修正するという方法をとる必要が

☆1 ここでは，語の品詞，活用などを明らかにする形態素解析もパージングの一部であるとして話をすすめる．

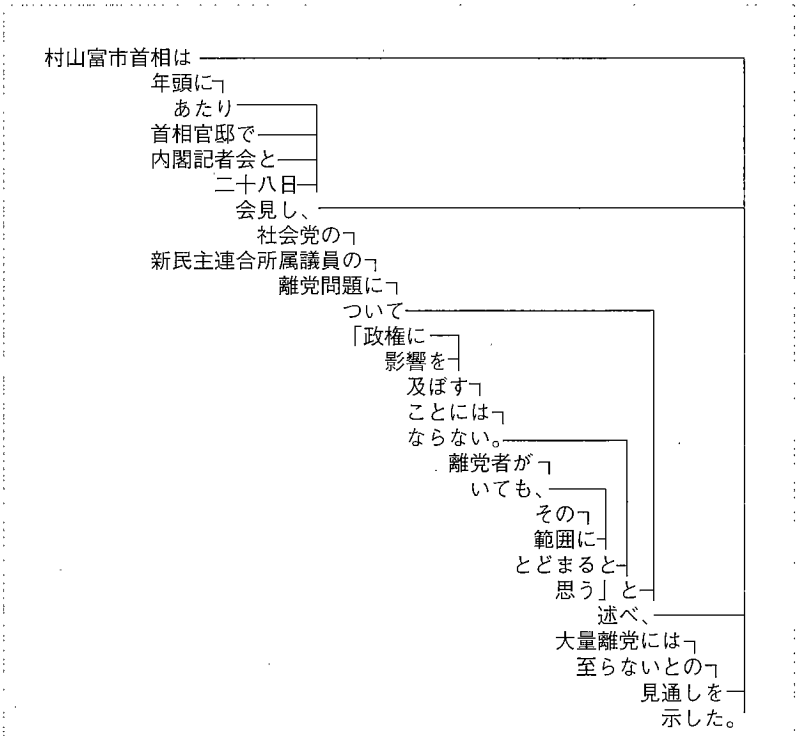

図-1 日本語文の構造（文節間の係り受け表現）

ある．このとき，パーサーの精度がある程度高いということが重要である．精度の低いパーサーが誤った情報ばかり付けるようでは，かえって人間の作業のじゃまになってしまう．できるだけ精度のよいパーサーを使う方が，人間の作業が楽になり，でき上がったコーパスの品質もよくなるはずである．

つまり，「よいパーサーにはよいコーパスがいるが，よいコーパスにはよいパーサーがいる」という，「鶏が先か卵が先か」のような問題が起こるわけである．そこで，もう一度「よいパーサーにはよいコーパスがいる」という点を整理して考えてみることにする．

パーサーはコーパスから何を学ぶべきか

パーサーはコーパスから何を学ぶべきかという問題を考えよう．そのためには，構文解析の難しさがどこにあるかということを考える必要がある．構文解析の難しさは，曖昧性，個別性，手がかりの統合，という3つの点にあると考えられる．

曖昧性 ——

自然言語の問題としてこれまで最もよく議論されてきたのがこの曖昧性の問題である．これには大きく分けて，主従関係，いわゆる係り受けの曖昧性と，並列関係の曖昧性がある．

係り受けの曖昧性は，たとえば次のような曖昧性である．

(1) N_1 を V_1 した N_2 が V_2 した
(2) V_1 した N_1 の N_2

日本語文の係り受けについて「『名詞＋を』は動詞に係る」，「連体形の動詞は名詞に係る」というような基本的な文法規則を考えると，(1) の文では，「N_1を」が「V_1した」に係る構造と「V_2した」に係る構造の2つの構造が得られ，(2) の文では，「V_1した」が「N_1」に係る構造と「N_2」に係る構造の2つの構造が得られる．

このような曖昧性の問題を解決するためには，「N_1をV_1する」と「N_1をV_2する」のどちらが適当であるか，また，「V_1したN_1」と「V_1したN_2」のどちらが適当であるかという情報（語の共起情報）が必要となる．これは，自立語と自立語の組合せ，すなわち自立語の数（数万語）の2乗のオーダの関係についての情報であり，これを学習するためには非常に大規模なコーパスが必要となる．しかし，人手作業のコストを考えると，そのような大規模な構文構造付コーパスを作成することはきわめて難しい．そのため，係り受けの曖昧性の問題を構文構造付コーパスによって解決することもきわめて難しいということになる（ただし，この問題は必ずしも構文構造付コーパスから学習する必要はないということを後で述べる）．

しかし，「ある程度の精度のパーサー」を作るという目的であれば，この問題には簡単な解決策がある．それは，係り受けの曖昧性については「一番近くに係る場合が非常に多い」という経験則があり，この経験則に従うことによってある程度の精度で構文解析を行うことができる．

一方，並列関係の曖昧性とは，たとえば次のような問題である．

(3) N_1 の N_2 と N_3 の N_4 が～

ここでも，「『名詞＋と』は名詞と並列になる」という文法規則を考えると，「N_2と」と並列になるのが「N_3」か「N_4」かという曖昧性が生まれる．

この問題は一見非常に深刻である．なぜなら，まず，この場合にもN_2がN_3と並列しやすいか，N_4と並列しやすいかという情報，すなわち係り受けの問題と同じく自立語数の2乗のオーダの情報が必要となり，構文構造付コーパスからの学習は難しい．

さらに問題なのは，係り受けの曖昧性について「一番近くに係る」という経験則があったのに対して，並列構造の曖昧性にはそのような単純で有効な経験則がない．つまり，並列構造の曖昧性では，小さい並列構造と大きい並列構造のどちらかがもっともらしいということはないのである．

ところが，この問題は類似性という観点を導入することによってうまく扱えることが分かっている[2]．すなわち，並列になりやすい語というのは似たような語であり，語の類似性というのは，同義・類議語などをまとめたシソーラスによって比較的うまく計算することができる．さらに，語の類似性だけでなく，語の並びとしての類似性も大きな手がかりとなる．たとえば「N_1をV_1するN_2と，N_3をV_2するN_4を～」という場合には，並びとしての類似性から，「N_2と，N_3」の並列よりも，「N_1をV_1するN_2と，N_3をV_2するN_4」の並列の方がもっともらしいということがいえる．このように類似性という観点によって並列構造の曖昧性はかなりの程度解決することができる．

個別性 ——

さきほど述べたように，日本語には機能的なさまざまな表現があり，その働きを個別に正しく扱わなければ文の構造が正しくとらえられないという問題がある．これはたとえば次のような問題である．

- 「(Vした) 途端,」は，単純な文法では「『名詞＋読点』は名詞と並列になる」と扱われてしまうが，実際には従属節を導く表現で，「途端,」は用言に係る．「(Vした) 矢先,」も同様の表現である．
- 名詞並列の構造を導くのは「N_1とN_2」「N_1やN_2」「N_1，N_2」などの典型的なものだけでなく，「N_1どころかN_2」「N_1のみならずN_2」「N_1にせよN_2（にせよ)」など多数の表現がある．
- 「(N_1は）もちろん（N_2も)」も並列を導く表現として扱う必要があるが，さらに，直前の文節「N_1は」の係り先を「もちろん」(副詞）にする必要がある．単純な文法では「名詞＋は」は用言に係ると扱われてしまう．

この問題は一見面倒な問題にみえるが，実はコーパスをよく見るということの効果が最も顕著に現れる問題である．このような表現は基本的に機能的な表現に限られ，その数は多くても数百程度である．これは人間が言語直感によって思いつくには多いが，数万文規模のコーパスにはそのほとんどが含まれていると考えられる．

そのような表現を含む文の構文解析は，パーサーがその働きを正しく扱っていなければ誤りとなるので，構文構造付コーパス作成の過程（人間がパーサーの解析結果を直していく過程）で比較的簡単に発見できる．このようにして問題となる表現を発見することができれば，それを正しく扱うための規則を人手で追加することはそれほど難しくない．このような規則は，個々の表現に対して与えるものであるから，その規則の追加によって副作用が出るということはほとんどないからである．

手がかりの統合——

構文解析において，構造を決定するために，いくつかの，別の種類の手がかりを組み合わせて利用するという場合がある．

たとえば，「N_1を，V_1したN_2がV_2した」という文があった場合，「N_1を,」の読点はこの文節が直後の「V_1した」には係りにくいという手がかりになる．また，「N_1をV_1する」と「N_1をV_2する」のどちらが語の共起関係から適当かということも別の手がかりであり，「N_1を,」の係り先の決定にはこの2つの手がかりを統合して判断する必要がある．

また，並列構造の解析の場合にも，さきほど述べたように「N_1をV_1するN_2と，N_3をV_2するN_4を～」では語の並びの類似性からこの全体が並列構造である可能性が高いが，もし「N_2」と「N_3」が非常に似た語である場合には「N_2と，N_3」という小さい並列構造である可能性もある．すなわち，並列構造を決定するためには語の類似性と語の並びとしての類似性を統合する必要がある．このような手がかりの統合をどのような優先付けで行うかということが構文解析の3つ目の難しさとなる．

もし，十分な規模の構文構造付コーパスがあれば，パーサーの出力が構文構造付コーパスに最も近くなるように，手がかりの統合を自動調整することができる．手法としては，適当な解空間の中でパラメタの最適値を求めたり，決定木，Maximum Entropyなどを用いることができる．

この問題では，個々の手がかりの学習ではなく，手がかりのタイプ間の調整が目的である．構文解析での手がかりは数十タイプ程度であるので，学習のためのデータ量はそれほど大量である必要はない．おそらく，数千から数万文の規模で十分であると思われる．

構文構造付コーパスがない場合，すなわちコーパス作成の初期段階では，このような調整は人間が行うしかない．しかし，数十から数百文の解析結果を見ることで経験的に調整を行うことは不可能ではなく，それでも「ある程度の精度」は得られると考えられる．

まとめると——

ここまでの議論から「コーパスが先か，パーサーが先か」について，次のような答えを得ることができる．

曖昧性の問題と，手がかりの統合の問題については「ある程度の精度」のパーサーを用意する手だてがある．そして，個別性の問題は，最初からまとまった量の構文構造付コーパスを必要とする問題ではなく，コーパスを作りながら徐々に文法を整備するということで対処可能である．

すなわち，まず，並列構造の曖昧性をうまく扱い，手がかりの統合を人手で行ったパーサーを構築する．そして，そのパーサーによってある量のテキストの構文解析を行い，その結果を修正しながら個別性の問題について文法整備を行う．新たなテキストの構文解析は，随時，改良されたパーサーによって行い，その結果をまた人手で修正する，というサイクルを繰り返す．

そして，ある程度の構文構造付コーパスができれば，手がかりの統合のための自動学習を行う．これはパーサーの精度を大きく改善するということはないだろうが，人手で微調整を行う必要がなくなり，ある種の安心感が得られるということになる．

このような構文構造付コーパスはどの程度の規模まで作る必要があるだろうか．個別の機能的表現を

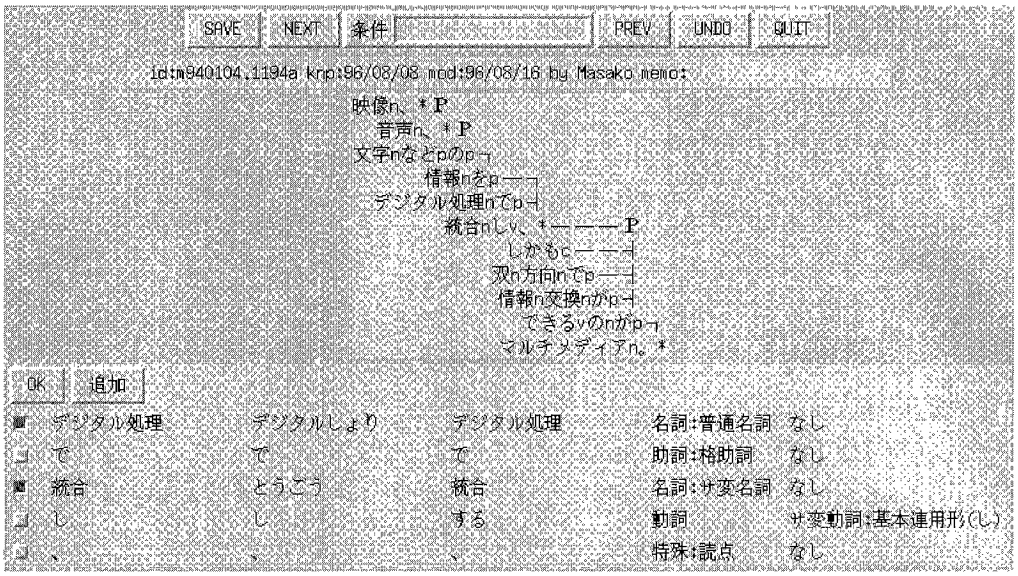

図-2　京大コーパス・プロジェクトでの解析結果修正用インタフェース

ほぼ網羅し，手がかりの統合の学習を行うためには，おそらく数万文あればよいだろう．それ以上増やしても，次に自立語間の関係を学習できるのに十分な規模にするのはほとんど不可能であると考えられる．

しかし，自立語間の関係の学習は必ずしも構文構造付コーパスを必要とするものではない．すなわち，上記のサイクルでかなり精度の高いパーサーができれば，それによって膨大なテキストを自動解析し，その結果から頻度などによってゴミ（パーサーの解析誤りと思われる部分）を取り除くことにより，十分な規模の自立語間の関係を学習できる可能性がある[3]．このような方法で自立語間の関係の学習ができれば，その結果を用いることで，さらにもう一段精度の高いパーサーを得ることができる．

京大コーパス・プロジェクト

我々は，上記のように問題を整理し，構文構造付コーパスを作りながらパーサーを整備するというプロジェクトを1995年にスタートさせた[4]（1995～1996年科学研究費補助金，1996～2000年日本学術振興会未来開拓推進事業）．

解析システムとしては，それまでに我々が開発してきたJUMAN（形態素解析システム）とKNP（パーサー）[5]を用いた．対象は毎日新聞とし，構文解析結果の修正は国語学の修士を持つ2名が行った（プロジェクトの前半1名，後半1名）．

プロジェクトのはじめには，小規模なコーパス（約3,000文）の解析結果を見ながらJUMAN，KNPの修正を行い，同時に，与える構文情報の基準の明確化，作業マニュアルの作成，構文情報修正用のGUI（**図-2**）の整備などを行った．これらが一段落してから，コーパスの修正とパーサーの修正を同時に行っていくという定常状態に入り，1997年9月に1万文，1998年6月に2万文，2000年6月に4万文の正解コーパスを公開した（**図-3**）．また，パーサーについても随時，改訂版を公開してきた．京大コーパスはコーパスベースの自然言語処理研究のための基礎データとして，パーサーは種々の自然言語処理アプリケーション構築のための基礎ツールとして，海外を含め多くの機関で利用されている．

このプロジェクトの成果である，コーパス，パーサー，マニュアル，GUIなどはすべてhttp://www-nagao.kuee.kyoto-u.ac.jp/ から入手可能である．ただし，著作権の問題から，コーパスを得るためには毎日新聞のCD-ROMを別途購入する必要がある．

今後の可能性

コーパスベースの自然言語処理には，今後2つの方向性が考えられる．

1つは，本稿で議論したように，人間が正しい情報を与えることを行いながら，規則などを学習するという方式である．この方式が有効な問題としては省略や代名詞の問題が考えられる．省略や代名詞の処理

```
#S-ID:950101003-001
* 0 26D
村山 むらやま * 名詞 人名 * *
富市 とみいち * 名詞 人名 * *
首相 しゅしょう * 名詞 普通名詞 * *
は は * 助詞 副助詞 * *
* 1 2D
年頭 ねんとう * 名詞 普通名詞 * *
に に * 助詞 格助詞 * *
* 2 6D
あたり あたり あたる 動詞 * 子音動詞ラ行 基本連用形
* 3 6D
首相 しゅしょう * 名詞 普通名詞 * *
官邸 かんてい * 名詞 普通名詞 * *
で で * 助詞 格助詞 * *
* 4 6D
内閣 ないかく * 名詞 普通名詞 * *
記者 きしゃ * 名詞 普通名詞 * *
会 かい * 名詞 普通名詞 * *
と と * 助詞 格助詞 * *
* 5 6D
二十八 にじゅうはち * 名詞 数詞 * *
日 にち * 接尾辞 名詞性名詞助数辞 * *
* 6 26D
会見 かいけん * 名詞 サ変名詞 * *
し し する 動詞 * サ変動詞 基本連用形
、 、 * 特殊 読点 * *
* 7 8D
社会党 しゃかいとう * 名詞 固有名詞 * *
の の * 助詞 格助詞 * *
* 8 9D
新 しん * 接頭辞 名詞接頭辞 * *
民主 みんしゅ * 名詞 普通名詞 * *
連合 れんごう * 名詞 サ変名詞 * *
所属 しょぞく * 名詞 サ変名詞 * *
議員 ぎいん * 名詞 普通名詞 * *
の の * 助詞 接続助詞 * *
* 9 10D
離党 りとう * 名詞 サ変名詞 * *
問題 もんだい * 名詞 普通名詞 * *
に に * 助詞 格助詞 * *
* 10 22D
ついて ついて つく 動詞 * 子音動詞カ行 タ系連用テ形
* 11 13D
「 「 * 特殊 括弧始 * *
政権 せいけん * 名詞 普通名詞 * *
に に * 助詞 格助詞 * *
* 12 13D
影響 えいきょう * 名詞 サ変名詞 * *
を を * 助詞 格助詞 * *
* 13 14D
及ぼす およぼす 及ぼす 動詞 * 子音動詞サ行 基本形
* 14 15D
こと こと * 名詞 形容名詞 * *
に に * 助詞 格助詞 * *
は は * 助詞 副助詞 * *
* 15 21D
なら なら なる 動詞 * 子音動詞ラ行 未然形
ない ない ない 接尾辞 形容詞性述語接尾辞 イ形容詞アウオ段 基本形
。 。 * 特殊 句点 * *
* 16 17D
離党 りとう * 名詞 サ変名詞 * *
者 しゃ * 接尾辞 名詞性名詞接尾辞 * *
が が * 助詞 格助詞 * *
* 17 20D
いて いて いる 動詞 * 母音動詞 タ系連用テ形
も も * 助詞 副助詞 * *
、 、 * 特殊 読点 * *
* 18 19D
その その * 指示詞 連体詞形態指示詞 * *
* 19 20D
範囲 はんい * 名詞 普通名詞 * *
に に * 助詞 格助詞 * *
* 20 21D
とどまる とどまる とどまる 動詞 * 子音動詞ラ行 基本形
と と * 助詞 格助詞 * *
* 21 22D
思う おもう 思う 動詞 * 子音動詞ワ行 基本形
」 」 * 特殊 括弧終 * *
と と * 助詞 格助詞 * *
* 22 26D
述べ のべ 述べる 動詞 * 母音動詞 基本連用形
、 、 * 特殊 読点 * *
* 23 24D
大量 たいりょう * 名詞 普通名詞 * *
離党 りとう * 名詞 サ変名詞 * *
に に * 助詞 格助詞 * *
は は * 助詞 副助詞 * *
* 24 25D
至らない いたらない 至らない 形容詞 * イ形容詞アウオ段 基本形
と と * 助詞 格助詞 * *
の の * 助詞 接続助詞 * *
* 25 26D
見通し みとおし * 名詞 普通名詞 * *
を を * 助詞 格助詞 * *
* 26 -1D
示した しめした 示す 動詞 * 子音動詞サ行 タ形
。 。 * 特殊 句点 * *
EOS
```

注）各行が1つの形態素の情報で，表記，読み，原形，品詞，品詞細分類，活用型，活用形の順．*は指定なし（非活用語の原形，活用型など）．*で始まる行は文節の区切りで，その次の数字が文節（次の*までの形態素列）の番号，次がその文節の係り先の文節番号で，記号D,P,Aによって通常の係り受け関係，並列関係，同格関係を区別する．この係り受け情報を木構造で示したものが図-1.

図-3　京大コーパスの例

には多くの手がかりが関係するので，手がかりの統合のために構文構造付コーパスが有効であると考えられる．

もう1つの方向は，人間が正しい情報を与えるのではなく，普通のコーパスをそのまま使うという方向である．この場合は，非常に大量のコーパスが利用できるという利点があり，さきほども述べたように自立語間の関係の学習などにはこの方向が有望である．ただし，単にコーパスを集めるというのではなく，ジャンル，スタイル，文の難易度などのバランスを考慮する必要があるだろう．

参考文献

1) Marcus, M. P., Santorini, B. and Marcinkiewicz, M.: Building a Large Annotated Corpus of English: the Penn Treebank, Computational Linguistics, 19 (2) (1993).
2) 黒橋禎夫, 長尾　真: 長い日本語文における並列構造の推定, 情報処理学会論文誌, Vol.33, No.8 (Aug. 1992).
3) 河原大輔, 鍛冶伸裕, 黒橋禎夫: 大規模コーパスからの格フレーム辞書構築とそれを用いた格解析, 言語処理学会第6回年次大会, pp.24-27 (2000).
4) 黒橋禎夫, 長尾　真: 京都大学テキストコーパス・プロジェクト, 言語処理学会第3回年次大会, pp.115-118 (1997).
5) 黒橋禎夫, 長尾　真: 並列構造の検出に基づく長い日本語文の構文解析, 自然言語処理 Vol.1, No.1 (1994).

（平成12年5月23日受付）