タグ付きコーパスを使用した言語研究

―XML コーパスの利用の実例と技術―

今田　水穂

キーワード：タグ付きコーパス、XML、形態素情報、構文情報、韻律情報

1. はじめに

近年、京都テキストコーパス、日本語話し言葉コーパスなど、形態素情報、構文情報の付与された大規模テキストコーパスが、研究用に容易に利用可能となってきている。しかし、これらのタグ付きコーパスは単純なプレーンテキストとは違って扱いが簡単でない部分もあり、また利用方法も十分に確立されていない。プレーンテキストが対象であれば、テキストエディタなどの検索機能で容易に用例検索が可能であり、またスクリプト言語などを用いたやや高度な処理についても田野村・服部・杉本・石井(2007b)のようなガイドブックが利用可能である。一方、タグ付きコーパスの利用にはより高度な知識・技術が要求され、またそれを利用することによって（あるいは、利用方法を習得するためのコストを払うことによって)、プレーンテキストを用いた用例検索以上のどのようなことが実現可能かということについても、あまり知られていないように思われる。本稿は、日本語話し言葉コーパス（以下 CSJ）を例にとって、その利用方法と有用性、XML 文書の操作に必要な技術、起こり得る問題などについて検討する。

2. タグ付きコーパスで何ができるか

2.1 形態素情報の利用

タグ付きコーパスの利点は、形態素情報、構文情報、音声・音韻情報などの多様な付加情報を利用することができるという点である。例えば形態素情報を使えば、文字列のどこからどこまでが単一の語であるかとか、その語の読み、品詞、活用形などの情報を検索や分析のために利用することができる。

これらの情報が役に立つ状況のひとつは、格助詞の「が」のような語形が短く形態的特徴が少ない要素を検索する場合である。単純なプレーンテキストの検索では、格助詞の「が」を接続助詞の「が」や「上がる」などの語の途中に現れる「が」と区別することは難しい。形態素情報付きコーパスを使えば、品詞が「格助詞」である「が」だけを選んで抽出することが可能になる。図 1 は、格助詞の「が」という条件で検索処理を行った結果である。検索結果は、[講演 ID, 節 ID, 前文脈, ガ, 後文脈]という形式のコンマ区切りテキストで出力してある。

A01F0055,1, えー 私 共 は 乳児,が,音楽 を どの よう に 聞い て いる か また 聴取 に 発達 齢 差 が 見 られる か を 検討 し て おり ます
A01F0055,1, えー 私 共 は 乳児 が 音楽 を どの よう に 聞い て いる か また 聴取 に 発達 齢 差,が,見 られる か を 検討 し て おり ます
A01F0055,4,また 海外 の えー 研究 で は 四五 歳 児 幼児 は え 両者 の 弁別,が,可能 で ある と いう 報告 も あり ます
A01F0055,7,え ゼロ 歳 児 を 対象 と し た 研究 に は コーエン ら の えー 報告,が,えー 挙げ られ ます
A01F0055,8,彼 ら は 七 から 十 一 か月 児 を 対象 と し て 長 三 和音 例えば ド ミ ソ ミ ド の えー 分散 和音 型 を ずっと 聞か せ 続け て で その 後 に 短 三 和音 例えば レ ファ ラ ファ レ と いう よう な え 分散 和音 を 聞か せる と え その 変化 に 気付い て 音源 の 方向 を 振り向く 率,が,え 変化 し なかっ た 場合 に 比べ て 有意 に 高かっ た こと から 七 から 十 一 か月 児 は 長 三 和音 と 短 三 和音 を 弁別 し て いる と いう 風 に 報告 し て おり ます

【図１】検索結果

形態素情報は、検索条件として利用するばかりでなく、検索結果を分析するために利用することもできる。例えば、格助詞の「が」がどのような環境に生起するかを調べるのに利用することができる。表１は、格助詞の「が」に前接する語の品詞を調べて、その頻度を集計したものである。

【表１】格助詞「が」に前接する要素の品詞

名詞	助詞	代名詞	(なし)	記号	助動詞	感動詞	言いよどみ
6385	726	585	142	55	23	16	15
接頭辞	動詞	接尾辞	形容詞				
7	5	2	1	合計 7962			

結果を見ると、名詞、代名詞や助詞が非常に多く、動詞や形容詞はあまりないということが分かる。さらに詳しく、どのような助詞が多く生起するのかとか、「なし」（品詞情報が検出されなかった）とはどういう場合なのかといったことを調べることができるが、ここではこれ以上は触れない。

2.2 構文情報の利用

構文情報を利用すると、ある要素が他のどのような要素と係り受け関係を結んでいるかを調べたり、要素間の係り受け関係を検索の条件に利用したりすることができる。例えば、「ＡがＢだ」というパターンのコピュラ文を調べたい場合には、単に格助詞の「が」を検索するのではなく、「Ｂだ」に係る「が」のみを検索することによって、より効率的に目的の表現を検索することができる。図２は、格助詞の「が」が助動詞の「だ」「です」「である」のいずれかに係る、という条件で検索処理を行った結果である。検索結果は、[講演

ID, 節 ID, 前文脈, ガ, 中文脈, ダ, 後文脈]という形式のコンマ区切りテキストで出力してある。

```
A01F0055,4,また 海外 の えー 研究 で は 四五 歳 児 幼児 は え 両者 の 弁別,が,
  可能,で ある,と いう 報告 も あり ます
A01F0055,9,え ただし えー 私 達,が,日常 耳 に する よう,な,音楽 の 旋律 の 形
  態 を 素材 と し た 聴取 反応 に つい て は えー まだ ゼロ 歳 児 で え そう い
  う こと が 可能 で ある と か あの一 そう いう こと に 関し て は まだ 報告 さ
  れ て おり ませ ん
A01F0055,9,え ただし えー 私 達 が 日常 耳 に する よう な 音楽 の 旋律 の 形
  態 を 素材 と し た 聴取 反応 に つい て は えー まだ ゼロ 歳 児 で え そう い
  う こと,が,可能,で ある,と か あの一 そう いう こと に 関し て は まだ 報告 さ
  れ て おり ませ ん
A01F0055,23,で えーっと 実験 時 に は 乳児 の 耳元 で それぞれ の 音圧,が,六十
  五 から 七十 デシベル 程度 に なる よう,に,調整 し て あり ます
A01F0055,26,え 上 に あり ます の,が,えーと 元 の えー メロディー,です,ね
```

【図２】検索結果

構文情報も、検索条件として利用するばかりでなく、検索結果を分析するために利用することができる。表２は、格助詞「が」の係り先文節の先頭の単語について、その品詞の頻度を集計したものである。動詞、形容詞、名詞、形状詞（形容動詞語幹）などが多いのが分かる。

【表２】格助詞「が」の係り先文節の先頭要素の品詞

動詞	形容詞	名詞	形状詞	(なし)	代名詞	副詞	助詞
5836	821	816	299	92	29	21	18
記号	助動詞	連体詞	言いよどみ	接続詞	感動詞	接尾辞	
8	7	7	3	2	2	1	合計 7932

これは係り先を調べた例だが、逆に修飾語を調べることもできる。例えば、ある述語がどのような格助詞を取るかとか、それらの語順はどうか、といったことを調べることができる。もっとも、このような方法がいつも上手くいくわけではない。そのような上手くいかない事例のひとつについては、第４節で述べたいと思う。

2.3 音声・音韻情報の利用

音声や音韻に関わる情報も、用例の検索条件や検索結果の分析のために利用可能な情報のひとつである。CSJ には、X-JToBI[1]という規格によって記述された、文節音や韻律に関する情報が付与されている。これを利用して、格助詞「が」の部分に現れる音調の変化を

[1] 国立国語研究所(2006)の第７章を参照。

調べたり、上昇調で発話される「が」のみを検索したりすることができる。図３は、図２の結果の末尾に「が」の部分のトーンラベル（句末音調など）を追加したものである。

A01F0055,4,また 海外 の えー 研究 で は 四五 歳 児 幼児 は え 両者 の 弁別,が,
可能,で ある,と いう 報告 も あり ます,H%
A01F0055,9,え ただし えー 私 達,が, 日常 耳 に する よう,な,音楽 の 旋律 の 形
態 を 素材 と し た 聴取 反応 に つい て は えー まだ ゼロ 歳 児 で え そう い
う こと が 可能 で ある と か あの一 そう いう こと に 関し て は まだ 報告 さ
れ て おり ませ ん,L%
A01F0055,9,え ただし えー 私 達 が 日常 耳 に する よう な 音楽 の 旋律 の 形
態 を 素材 と し た 聴取 反応 に つい て は えー まだ ゼロ 歳 児 で え そう い
う こと,が,可能,で ある,と か あの一 そう いう こと に 関し て は まだ 報告 さ
れ て おり ませ ん,L%
A01F0055,23,で えーっと 実験 時 に は 乳児 の 耳元 で それぞれ の 音圧,が,六十
五 から 七十 デシベル 程度 に なる よう,に,調整 し て あり ます,L%
A01F0055,26,え 上 に あり ます の,が,えーと 元 の えー メロディー,です,ね,pH
HL%

【図３】検索結果（韻律ラベル付き）

頻繁に現れるラベルとしては、下降調(L%)、上昇調(H%)、上昇下降調(pH HL%)などがある。韻律情報は、形態や構文情報などでは判別できないような、言語表現の文法的特徴を判別するために利用できるかも知れない。例えば、久野(1973)の言う中立叙述の「が」と総記の「が」の違いが、韻律的特徴に現れている可能性が考えられる。しかし、網羅的に調査したわけではないが、結果を概観する限り「が」の用法と句末音調の間には、それほど明確な対応関係は観察されないように見える。図４は「～が特徴だ」というタイプの文で、「が」は総記の用法であると考えられるが、「が」周辺の句末音調は下降調(L%)の場合も上昇調(H%)の場合もあり得る。

A03M0004,49,え つまり 最 年少 の と いう 意味 的 重要 部分 は 削除 さ れ て し
まい ます が え 主語 述語 の よう な 日本 語 らし さ は 維持 さ れ て いる の,
が,とっ この 日本 語 らし さ のみ を 判断 基準 に し た 場合 の 要約 えー の 特
徴,で あり,ます,H%
A06F0075,39,で あの ピアレビュー を 行なう と いう こと が もう 既に エフティー
エー で ある と いう そう いう 前提 を 立て て えー 三 本 共 研究 が 進ん で い
る と いう こと,が,あの一 特徴,だ,と 言え ます,A L%
S03M0046,21,で まー これ,が,大体 文京 区 の 全体 的 な 特徴,な,ん です けど,
S03M0317,60,まー その 他 に も え ケヤキ 通り と か です ね あの一 緑 の 非常
に 多い の,が,えー 常盤平 の 特徴,だ,と 思い ます,L%

【図４】「～が特徴だ」の「が」部分の句末音調

郡(1997)はフォーカスとイントネーションの関係について、「フォーカスがある語のアク

セントの高低変化が強調され、同時にそれより後にある語群のアクセントの高低変化が抑えられる」としている。また、谷口・丸山(2001)の実験では、文の焦点は、焦点要素のアクセントが卓立されるパターンと、それに後続する助詞も卓立されるパターンの二種類が観察されたという。従って「が」の部分の句末音調ばかりではなく、「Aが」の部分のピッチ変化と、「Bだ」の部分のピッチ変化について、その起伏の度合いを調べることができれば、何らかの特徴を観察できるかも知れない。これについては本稿が扱うことのできる範囲を超えるので、コーパスの利用の可能性として示唆するだけに留める。しかしもし必要であれば、そのためのデータ（5ms 刻みで記録された基本周波数データ）も CSJ には収録されている。

3. 構文タグ付きコーパスの構造と検索方法

3.1 日本語話し言葉コーパス

構文情報が利用可能な大規模日本語コーパスの代表的なものに、京都テキストコーパス[2]、日本語話し言葉コーパス[3](CSJ)などがある[4]。本稿はここまで、CSJ を用いたタグ付きコーパスの利用事例を紹介してきた。本稿が CSJ を選んだのは、このコーパスが比較的入手、導入が容易であるためである。例えば京都コーパスの場合、本文にあたる毎日新聞データを購入するために 12 万円程度の費用がかかり、またインストール作業を行うために UNIX に関する知識などが要求される（Appendix I 参照）。

一方、CSJ は学術利用の場合５万円（学生は２万５千円）と比較的安価で、入手が容易である。また、データ自体も、京都コーパスのように本文とタグ情報が分離した形式で提供されるのではなく、あらかじめタグ付けされたデータが DVD-ROM に収録されているので、特にインストール作業などをしなくても、ただちに利用することができる。

CSJ には、音声ファイル、転記テキスト、単位データベース、XML 文書など、いくつかの形式でデータが収録されている。形態素情報を利用するだけであれば、単位データベースが便利かも知れない。単位データベースは KWIC に似た形式のタブ区切りテキストファイルで、MS-Excel のような表計算ソフトを利用して簡単に閲覧、検索等の操作をすることができる。一方、構文情報や音声・音韻情報までを利用したい場合には、XML 形式のデータを使用する必要がある。CSJ には Disk2 他に収録されている（IPU[5]単位）XML 文書と、Disk18 に収録されている節単位 XML 文書があり、後者の方が構文分析に適した構造を有している。本稿では特に断りのない限り、節単位 XML（Disk18 の/CU/core フォルダ収録分）を操作対象とするものとする。

---

[2] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/corpus.html
[3] http://www.kokken.go.jp/katsudo/seika/corpus/
[4] 日本語研究に利用可能な大規模コーパス、電子化辞書等の言語資料については、Asahara, Den, and Matsumoto(2006)に詳しくまとめられている。
[5] Inter-Pausal Unit。0.2 秒以上のポーズを切れ目とした音声的単位。

3.2 XML コーパスの構造

XML とは、文書の意味や構造を記述するための言語である。CSJ の場合は、節、文節、語などの言語単位や、文節間の係り受け情報、語の形態素情報（基本形、活用形、読み、品詞など）を記述するために用いられている。ごく単純化すると、図５のような構造を有している。

```
<Talk>
    <ClauseUnit>
        <Bunsetsu Dep_BunsetsuUnitID="0" Dep_ModifieeBunsetsuUnitID="1">
            <LUW LUWLemma="鯨" LUWPOS="名詞" />
            <LUW LUWLemma="は" LUWPOS="助詞" LUWMiscPOSInfo1="係助詞"/>
        </Bunsetsu>
        <Bunsetsu Dep_BunsetsuUnitID="1">
            <LUW LUWLemma="哺乳類" LUWPOS="名詞" />
            <LUW LUWLemma="だ" LUWPOS="助動詞" />
        </Bunsetsu>
    </ClauseUnit>
</Talk>
```

【図５】節単位 XML ファイルの文書構造

CSJ の節単位 XML データは、Talk（談話全体）、ClauseUnit（節）、Bunsetsu（文節）、LUW（長単位単語）、SUW（短単位単語）、Mora（モーラ）という階層構造になっている（図５では SUW と Mora は省略している）。係り受け関係は Bunsetsu 要素に記述されている。Dep_BunsetsuUnitID がその文節の ID で、Dep_ModifieeBunsetsuUnitID が係り先文節の ID である。文節の下には語があるが、語には長単位単語(LUW)、短単位単語(SUW)という二種類の単位が用意されている。例えば「哺乳類」という単語は、LUW では「哺乳類」だが、SUW では「哺乳」と「類」の二語に分けられる。LUW および SUW にはそれぞれ代表形、品詞、活用形などの形態素情報や、他の様々な情報が付与されている（図５では省略している）。代表的なものを以下に挙げる。

【表３】単語要素に付与された形態素情報属性

代表形	LUWDictionaryForm	SUWDictionaryForm	例：イウ（辞書見出し）
代表表記	LUWLemma	SUWLemma	例：言う（辞書見出し漢字表記）
品詞	LUWPOS	SUWPOS	例：動詞
活用の種類	LUWConjugateType	SUWConjugateType	例：ワア行五段
活用形	LUWConjugateForm	SUWConjugateForm	例：未然形
タグ無し出現形		PlainOrthographicTranscription	例：言わ
発音形		PhoneticTranscription	例：イワ

音声、音韻に関する情報は、Mora 要素よりさらに下位の、Phone 要素や Phoneme 要素その他に記述されている。しかしこれらの要素は、IPU 単位 XML データには収録されているが、節単位 XML データでは省略されている。そのため、2.3 節で示したような音声・音韻情報を利用した検索処理を行うためには IPU 単位 XML を直接利用するか、あるいは独自に音声・音韻情報付きの節単位 XML データを作成する必要がある。CSJ の Disk 18 には IPU 単位 XML データを節単位 XML データに変換する XSLT スクリプトが付属されており、これを修正することによって音声・音韻情報付き節単位 XML を作成することができる（ただし XSLT の知識が必要である）。

3.3 XML コーパスの検索

3.3.1 XPath と XSLT

タグなどの付加されていないテキストファイル（プレーンテキスト）を検索する際には、正規表現という言語がよく用いられる。XML 文書もテキストファイルの一種であるが、通常のプレーンテキストとは違って本文以外の付加情報を多数含んでいるので、プレーンテキストと同じように正規表現で検索処理をするのは（不可能ではないが）難しいかも知れない。XML 文書を検索するときには、XPath という言語が有用である。XPath は XML 文書の特定の部分を指定するための言語である。例えば、品詞が「名詞」である LUW を指定するときには、次のように指定する。

（1） `/Talk/ClauseUnit/Bunsetsu/LUW[@LUWPOS="名詞"]`

XPath 言語を用いて実際の検索処理をするには、いくつかの方法がある。ひとつは XSLT と呼ばれるものである。XSLT は XPath を拡張した言語であり、本来は XML 文書を別の形式の文書（HTML など）に変換するためのものであるが、元の XML 文書の特定の部位だけを指定して変換することもできるので、検索処理のために用いることもできる。XSLT スクリプトを実行するための処理系（プログラム）にはいくつかのものがあるが、国立国語研究所(2006: 537)では xsltproc を推奨している。国立国語研究所(2006)の第 8 章、第 9 章では、実際に XSLT を用いた検索、整形処理について説明している。

CSJ に付属の検索ツール CSJ XML Browser も、XPath や XSLT による検索、整形処理に対応したツールである。このツールは、GUI[6]環境で XML 文書の検索処理を行うためのツールであるが、XPath や XSLT の直接入力による検索、整形処理にも対応している。国立国語研究所(2006: 517)によると、このツールは最低限の XML 関連技術(XPath1.0、XSL1.0)の習得も狙いの一つであるということなので、手早くこれらの技術を習得するために

[6] Graphical User Interface の略。画像を多用して情報を表示し、マウス操作などによって入力を受け付けるインターフェイスのこと。これに対して、コマンドプロンプトなど、情報の表示と入力を文字だけで行うインターフェイスを CUI(Character-based User Interface)と言う。

は、このツールを利用するのもひとつの手段であろう[7]。

3.3.2 XQuery

XPath を利用した別の検索方法として、XQuery が挙げられる。XSLT が本来変換処理用の言語であるのに対して、XQuery は XML データ問い合わせ（XML データベースへの照会）用の言語である。データの検索という目的で言えば、XSLT よりも XQuery の方がより適切な処理方法と言えるかも知れない。以下は国立国語研究所(2006: 13)に示されている XSLT スクリプト[8]と同等の処理を、XQuery で記述したものである（この節と次の節では、検索対象は節単位 XML ではなく、IPU 単位 XML である)。

```
xquery version "1.0";

for $suw in document("A01F0132.xml")//SUW
let $cbl := xs:string($suw/@ClauseBoundaryLabel)
where $cbl != ""
return
  string-join((
    xs:string($suw/ancestor::Talk/@TalkID),
    xs:string($suw/ancestor-or-self::IPU/@IPUID),
    xs:string($suw/ancestor-or-self::LUW/@LUWID),
    xs:string($suw/ancestor-or-self::SUW/@SUWID),
    xs:string($suw/preceding::SUW[4]/@SUWLemma),
    xs:string($suw/preceding::SUW[3]/@SUWLemma),
    xs:string($suw/preceding::SUW[2]/@SUWLemma),
    xs:string($suw/preceding::SUW[1]/@SUWLemma),
    xs:string($suw/@SUWLemma),
    xs:string($suw/following::SUW[1]/@SUWLemma),
    xs:string($suw/ancestor-or-self::SUW/@ClauseBoundaryLabel),
    $suw/descendant::XJToBILabelTone/text(),
    $suw/descendant::XJToBILabelBreak/text()
  ),",")
```

【図６】XQuery スクリプト

XSLT の実行に xsltproc などのプログラムが必要であるのと同様、XQuery の実行にも対応したプログラムが必要である。今回は eXist[9]というソフトウェアを使用して元の XSLT スクリプトと同等の結果が得られることを確認した。結果は以下の通りである。

---

[7] ただし、筆者の環境では CSJ XML Browser が正常に動作しない問題が発生した。Appendix II 参照。

[8] 「節境界ラベルを保有するすべての短単位を検索し、講演 ID、転記基本単位 ID、先行するよっつの短単位代表形、当該短単位の代表形、後続するひとつの短単位代表形とともに、当該短単位に付与された節境界ラベル、当該短単位の時間区間内に存在する韻律情報中のトーンラベルと BI ラベルを出力する」（国立国語研究所 2006: 12）という検索処理を行うスクリプト。

[9] http://exist.sourceforge.net/

```
A01F0132,0237,25,1,と,考える,て,居る,ます,えー,[文末],A,L%,3
A01F0132,0240,6,1,今後,の,課題,です,が,あの,/並列節ガ/,L%,3
A01F0132,0240,16,2,実験,段階,です,の,だ,えー,<理由節ノデ>,pH,HL%,3
A01F0132,0241,14,1,を,,あー,使う,て,更に,<テ節>,H%,2+b
A01F0132,0245,7,1,進める,て,行く,予定,です,以上,[文末],L%,H%,3
A01F0132,0246,2,1,行く,予定,です,以上,です,[文末],L%,1
```

【図7】図6に示したXQueryスクリプトの実行結果

単純な比較はできないが、図6のXQueryスクリプトは元のXSLTスクリプトと比べて20行程度短く、筆者の印象で言えば、書き易さ、可読性などの面でもXQueryの方が扱いやすく感じられる。XQueryにはFLWOR表現式(for let where order by return)という構文が用意されており、forが要素の繰り返し、letが変数の代入、whereが条件の絞込み、order byが並べ替え、returnが出力データの整形を担当している。whereやorder byはデータベース問い合わせ言語として広く用いられているSQL言語にもある要素であり、SQLに馴染みのある利用者にとっては理解し易いかも知れない。国立国語研究所(2006: 539)に指摘のある通り、各手法の特徴を理解した上で目的に応じて使い分けることが重要であろう。

3.3.3 スクリプト言語とXMLパーサー

XML文書を扱う別の方法は、PerlやRubyのようなスクリプト言語を用いるというものである。筆者はXML文書を扱うとき、主にRubyを使用している。これらの言語は必ずしもXML文書の処理に向いた機構を持っているというわけではなく、目的によってはXSLTやXQueryのようなXML文書の処理に特化された言語を用いた方が効率よく目的を達成できるかも知れない。

にもかかわらず筆者がRubyを使うのは、スクリプト言語にも相応の利点があるためである。スクリプト言語の利点のひとつは、比較的扱いやすく、生産性が高く、汎用性が高いという点である。これらの言語は特にテキストファイルの処理に向いているが、使い方次第ではXML文書の処理、データベース操作、統計処理その他様々な用途で利用することができる。awk、Perl、Python、Rubyといったスクリプト言語のいずれかひとつにでも習熟しておくことは、言語研究のためにコンピュータを利用する上で非常に有益であると言える。スクリプト言語の一般的な使用方法、言語研究のための利用については、田野村・服部・杉本・石井(2007b)などを参照されたい。

すぐ上で述べたように、Rubyは必ずしもXML文書の処理に適した機構を持った言語ではない。このような汎用的なスクリプト言語でXML文書を処理する場合には、XMLパーサーと呼ばれるXML文書を解析して扱い易い形式のデータに変換するプログラムを援用

するのが一般的である。Ruby の場合、REXML というパーサーが標準で利用可能であるが、このパーサーは処理速度が遅く、大規模コーパスのような大量の XML 文書を処理する用途には向かない。このような場合には、C 言語などで書かれたより高速な XML パーサーを利用することが好ましい。

Ruby から利用可能な XML パーサーにはいくつかの種類のものがあるが、Windows 環境の場合は msxml を利用するのが簡単であろう。msxml は Internet Explorer のコンポネントであり、Windows 環境であれば通常は利用可能である。また、Ruby から msxml を利用するために必要な win32ole ライブラリも、Windows 版の Ruby には標準で添付されている。そのため、XML パーサーやライブラリを新たにインストールする必要なく、直ちに利用することが可能である。win32ole については Thomas et al.(2005)を、msxml については Microsoft のドキュメント[10]等を参照されたい。参考までに、図６と同等の処理を行う Ruby スクリプトを以下に示しておく。

```
require 'win32ole'
class WIN32OLE; def to_s; self.value; end; end  # WIN32OLE のカスタマイズ
doc = WIN32OLE.new('MSXML.DOMDocument')          # MSXML パーサーの初期化
doc.setProperty 'SelectionLanguage', 'XPath'     # XPath の使用宣言
ARGV.each do |file|
  doc.load file
  doc.selectNodes('//SUW[@ClauseBoundaryLabel]').each do |suw|
    puts [
      suw.selectSingleNode('ancestor::Talk/@TalkID'),
      suw.selectSingleNode('ancestor-or-self::IPU/@IPUID'),
      suw.selectSingleNode('ancestor-or-self::LUW/@LUWID'),
      suw.selectSingleNode('ancestor-or-self::SUW/@SUWID'),
      suw.selectSingleNode('preceding::SUW[4]/@SUWLemma'),
      suw.selectSingleNode('preceding::SUW[3]/@SUWLemma'),
      suw.selectSingleNode('preceding::SUW[2]/@SUWLemma'),
      suw.selectSingleNode('preceding::SUW[1]/@SUWLemma'),
      suw.selectSingleNode('@SUWLemma'),
      suw.selectSingleNode('following::SUW[1]/@SUWLemma'),
      suw.selectSingleNode('ancestor-or-self::SUW/@ClauseBoundaryLabel'),
      suw.selectSingleNode('descendant::XJToBILabelTone/text()'),
      suw.selectSingleNode('descendant::XJToBILabelBreak/text()')
    ].join(',')
  end
end
```

【図８】Ruby スクリプト

[10] http://msdn2.microsoft.com/en-us/library/ms763742.aspx

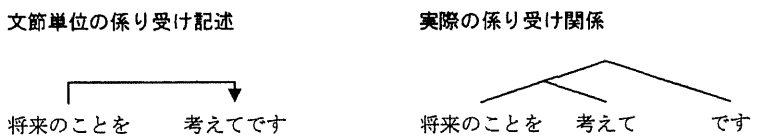
4. 係り受け関係の記述単位とその問題点

ここまで、タグ付きコーパスの有用性と、利用のための技術について概観してきた。ここでは、タグ付きコーパスの利用の際に生じる問題のひとつについて言及する。その問題とは、係り受け関係記述のための言語単位の設定に関わる問題である。

よく知られているように、言語の音韻構造、統語構造、意味構造は、必ずしも平行的な構造を有しているとは限らない。例えば、音韻的な単位が、統語的な節や句と一致しないということがしばしばあり得る。従って、これらの複数の構造に関わる情報を含むタグ付きコーパスにおいては、どのように言語単位を設定するかが問題となる。CSJ が IPU 単位 XML と節単位 XML という異なる構造の XML データを収録するのも、そうした単位認定の問題と無関係ではないものと考えられる。

係り受け情報の記述単位も、その妥当性を検討すべき要素のひとつであるように思われる。CSJ に限らず、京都コーパスなども同様であるが、これらの構文タグ付きコーパスは、係り受け情報を文節単位で記述している。すなわち、ある文節がある文節に係る、という形式で係り受け情報を記述している。ところが、文節単位の係り受け記述は、実際の係り受け関係を正確に記述できない場合がある。そのひとつは、「だ」「である」のようなコピュラ要素を含む文節と、その修飾要素との係り受け関係である。

「～だ」「～です」「～である」のような述語文節に係る名詞句を調べてみると、「～は」「～が」などの他に、「～の」「～という」のような連体修飾成分や、「～を」のような普通は他動詞文にしか生起しないような連用修飾成分が数多く検出される。実際には、これらの要素の多くは「～だ」というコピュラ述語に係る要素ではない。「～の」や「～という」の多くは、「～だ」に係るのではなく「だ」に前接する名詞の部分にのみ係っている。同様に、「～を」の多くは、「～だ」に係るのではなく「だ」に前接する従属節の述語に係っている（図９）。



【図９】文節単位の係り受け記述

これはひとつには、コピュラという要素の形態的な特異性に原因がある。原理的には、コピュラが助詞「で」と動詞「ある」に分かれていて、これらが別々の文節を形成するなら、このような修飾成分の混在は起こらないはずである。図９の例で言えば、「考えてで」と「ある」が別々の文節に分かれていれば、従属節の修飾成分は「考えてで」に、主

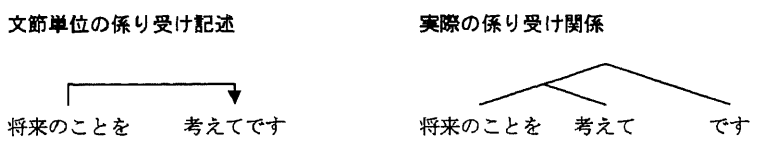
4. 係り受け関係の記述単位とその問題点

ここまで、タグ付きコーパスの有用性と、利用のための技術について概観してきた。ここでは、タグ付きコーパスの利用の際に生じる問題のひとつについて言及する。その問題とは、係り受け関係記述のための言語単位の設定に関わる問題である。

よく知られているように、言語の音韻構造、統語構造、意味構造は、必ずしも平行的な構造を有しているとは限らない。例えば、音韻的な単位が、統語的な節や句と一致しないということがしばしばあり得る。従って、これらの複数の構造に関わる情報を含むタグ付きコーパスにおいては、どのように言語単位を設定するかが問題となる。CSJ が IPU 単位 XML と節単位 XML という異なる構造の XML データを収録するのも、そうした単位認定の問題と無関係ではないものと考えられる。

係り受け情報の記述単位も、その妥当性を検討すべき要素のひとつであるように思われる。CSJ に限らず、京都コーパスなども同様であるが、これらの構文タグ付きコーパスは、係り受け情報を文節単位で記述している。すなわち、ある文節がある文節に係る、という形式で係り受け情報を記述している。ところが、文節単位の係り受け記述は、実際の係り受け関係を正確に記述できない場合がある。そのひとつは、「だ」「である」のようなコピュラ要素を含む文節と、その修飾要素との係り受け関係である。

「～だ」「～です」「～である」のような述語文節に係る名詞句を調べてみると、「～は」「～が」などの他に、「～の」「～という」のような連体修飾成分や、「～を」のような普通は他動詞文にしか生起しないような連用修飾成分が数多く検出される。実際には、これらの要素の多くは「～だ」というコピュラ述語に係る要素ではない。「～の」や「～という」の多くは、「～だ」に係るのではなく「だ」に前接する名詞の部分にのみ係っている。同様に、「～を」の多くは、「～だ」に係るのではなく「だ」に前接する従属節の述語に係っている（図９）。



【図９】文節単位の係り受け記述

これはひとつには、コピュラという要素の形態的な特異性に原因がある。原理的には、コピュラが助詞「で」と動詞「ある」に分かれていて、これらが別々の文節を形成するなら、このような修飾成分の混在は起こらないはずである。図９の例で言えば、「考えてで」と「ある」が別々の文節に分かれていれば、従属節の修飾成分は「考えてで」に、主

節の修飾成分は「ある」に係り、互いに区別されるのである。しかしこのような分析は現代日本語の記述としてはあまり意味のあるものではない。現代語としては、コピュラは文法化した単一の辞として（従ってコピュラ述語は単一の文節として）扱うのが妥当であろう。特に、「だ」や「です」の場合は形態的にも分割することが難しく、「である」についても、これをいちいち「で」と「ある」に分けるのはかえって検索の手間を増やすだけである。

このような言語要素が生起する環境では、文節単位の係り受け記述は、必ずしも十分に機能しない。要素間の係り受け関係を精密に記述するために、文節よりも小さな言語単位の設定が要求される。「だ」「です」「である」の直前に文節区切りを入れるというのも一つの方法かもしれないが、この方法が CSJ の言語単位設定のポリシーに適合するものであるかどうかは不明である。別の方法としては、文節要素の属性としてこのような特殊な係り受け関係を記述するという方法が考えられる。CSJ では、通常の依存構造にない関係（並列関係、同格関係など）を Bunsetsu 要素の Dep_Label 属性によって記述している。このような属性要素によってコピュラ述語と修飾要素の部分依存関係を記述することが可能であるかも知れない。

いずれにしろ現時点においては、コピュラ述語に係る要素が主節の連用成分なのか、従属節の連用成分なのか、連体成分なのかは、利用者が何らかの方法で判定、判断しなくてはならない。「の」や「こと」が名詞に係る成分であることは明らかであろうし、「を」はほとんどの場合（「～を開催中だ」のような例が問題になるかも知れないが）従属節中の要素と考えることができる。一方で、連用的にも連体的にも用いられる「と」のような要素や、「は」「が」などが主節の要素か従属節の要素かといった判定には、より高度な文法的知識が要求されるかも知れない。タグ付きコーパスを使いこなすためには、情報処理に関わる技術の習得ばかりでなく、文法やコーパスの仕様に関する知識も要求される。

5. まとめ

本稿は XML コーパスの利用方法と有用性、利用のための技術、文法上の問題点などについて検討してきた。タグ付きコーパスを使うと、プレーンテキストでは利用できない諸情報（形態素情報、構文情報、音声・音韻情報など）を、検索のための条件や、検索結果を分析するための手掛かりとして利用することが可能となる。

一方で、タグ付きコーパスの利用には、通常のプレーンテキストの場合と比べて、やや高度な技術が必要となる。XML コーパスを利用する場合には、XPath という言語の習得が不可欠であり、またそれを実際に利用するためには XSLT、XQuery、スクリプト言語などの知識・技術も要求される。こうした技術の習得は、コンピュータの使用に不慣れな利用者にとってはかなりの負担になるものと思われるが、今後、CSJ XML Browser のようなコーパス管理・検索用のソフトウェアの開発、整備が進めば、こうした負担は軽減され、

より手軽にタグ付きコーパスを利用することが可能になるかも知れない。

ただし、タグ付きコーパスを十分に使いこなすには、情報処理の技術ばかりではなく、文法やコーパスの仕様に関する知識も必要である。例えば当該コーパスの XML データがどのような構造になっているのかとか、どのような属性要素が付与されているのかといったことを理解しておく必要がある。また、本稿の第4節で触れたようなコーパスの設計上の特性や、それによって生じる問題についても考慮しなくてはならない。また、CSJ のような大規模コーパスは、多かれ少なかれ機械処理によって言語単位の切り出しや情報の付与を行っているのが普通なので、与えられている情報が必ずしも正確であるとは言い切れないという点にも注意しなくてはならない。こうした特性を十分に理解して、適切な用途と用法を選択することが重要である。

Appendix

I. 京都テキストコーパスの利用

京都テキストコーパス（以下 KC）を利用するには、次のような手続きが必要である。

1．本文データ（CD-毎日新聞'95 データ集[11]）を購入する。約 12 万円。
2．タグデータ（KyotoCorpus4.0.tar.gz）をダウンロードする。無償。
3．インストール作業を行う。

インストール作業を行うには、UNIX 環境を用意する必要がある（少なくとも、bash と perl というソフトウェアが必要になる）。Windows 環境でインストールを行う場合には、Cygwin[12]という擬似 UNIX 環境をインストールするのが簡単である。環境によっては（Windows や一部の Linux など）、KyotoCorpus4.0.tar.gz に同梱のインストールスクリプトをそのまま実行すると、文字化けが発生するようである。問題を回避するには、format.pl および num2KNP.pl というファイルを探し、先頭付近の `use open IO => ':endocing(euc-jp)';` の次の行に `use open ':std';` という行を挿入するとよい。

KC は XML 形式ではない。利用するには、スクリプト言語を利用する、XML データに変換する、データベースに格納するなどの、いずれかの方法を取らなくてはならない。比較的簡単な方法のひとつは、茶器(ChaKi)[13]というタグ付きコーパス管理・検索ソフトウェアを利用することである。茶器を用いると、KC をデータベースに格納し、GUI 環境で管理や検索を行うことができるようになる（MySQL[14]という無償のデータベースソフトが別途必要である）。ただし、茶器のインストールや KC のデータベースへの格納には、多少の

[11] http://www.nichigai.co.jp/sales/mainichi/mainichi-data.html
[12] http://www.cygwin.com/
[13] http://chasen.naist.jp/hiki/ChaKi/
[14] http://www.mysql.com/

手間が掛かるかも知れない。筆者が利用したバージョン(chaki-2006-8-21-preview)では、MySQL4 では動作したが、MySQL5 ではコーパスの格納に失敗するという問題が生じた。

II. CSJ XML Browser の利用

CSJ XML Browser は Java[15]で作成されたソフトウェアである。そのため、実行には Java ランタイム(J2RE1.4.0 以降)がインストールされている必要があるとされる。筆者の環境(WindowsXP+JRE1.6.0)では、このプログラム(ver.0.2)はうまく動作しなかった。問題を回避する方法のひとつは、Java ランタイムを 1.4 にダウングレードする（1.5 以降の Java ランタイムを全てアンインストールし、J2RE 1.4.2 をインストールする）というものである。最新の Java ランタイムを使い続けたい場合には、Xalan-Java をダウンロードしてきて/lib/ext フォルダにコピーするという方法がある[16]。最新のパッケージ(xalan-j_2_7_0-bin.zip)をダウンロードしてきて、同梱の jar ファイル(serializer.jar, xalan.jar, xercesImpl.jar, xml-apis.jar)をすべて C:¥Program Files¥Java¥jre1.6.0_03¥lib¥ext にコピーする。ファイル名、フォルダ名等はいずれも 2007.10.22 現在のものであり、バージョン番号等は変わる可能性がある。適宜読み替えられたい。

【参考文献】

久野暲(1973)『日本文法研究』大修館書店.

郡史郎(1997)「日本語のイントネーション：型と機能」国広哲弥・廣瀬肇・河野守夫[編]『アクセント・イントネーション・リズムとポーズ』三省堂, 169-202.

国立国語研究所(2006)『日本語話し言葉コーパスの構築法』国立国語研究所報告 124.

谷口未希・丸山岳彦(2001)「焦点構造と助詞の卓立」*KLS* 21, 56-66.

田野村忠温(2006)「コピュラ再考」藤田保幸・山崎誠[編]『複合辞研究の現在』和泉書院.

田野村忠温・服部匡・杉本武・石井正彦(2007a)『コーパスを用いた日本語研究の精密化と新しい研究領域・手法の開発 I』特定領域研究「日本語コーパス」平成 18 年度研究成果報告書.

田野村忠温・服部匡・杉本武・石井正彦(2007b)『コーパス日本語ガイドブック』特定領域研究「日本語コーパス」平成 19 年度研究成果報告書.

Asahara, Masayuki, Yasuharu Den, and Yuji Matsumoto (2006) Computational linguistics, In Mineharu Nakayama, Reiko Mazuka, and Yasuhiro Shirai (eds.) *The Handbook of East Asian Psycholinguistics: Japanese.* Cambridge University Press. 323-332.

Thomas, Dave, Chad Fowler, and Andy Hunt (2004) *Programming Ruby: The Pragmatic Programmar's Guide*, 2nd ed. Pragmatic Bookshelf.（まつもとゆきひろ[監訳]・田和勝[訳](2006)『プログラミング Ruby 第 2 版 言語編／ライブラリ編』オーム社）

[15] http://www.java.com/

[16] http://xalan.apache.org/xalan-j/

訂正とお詫び

「筑波応用言語学研究」１４号におきまして２５頁と２６頁に全く同じものが入ってしまいました。

ここに訂正して深くお詫び申し上げます。

株式会社いなもと印刷