Memoirs of the Osaka Institute
of Technology, Series B
Vol.49,No.1(2004) pp.13～32

専門分野別英語教育のためのジャンルコーパス作成の試み*
— XMLによるタグ付与と検索プログラム —

井村　誠

(知的財産学部　知的財産学科)

〈2004年5月31日受理〉

Building a Genre-Based Corpus for ESP
— XML Tagging and Search Program —

by

Makoto IMURA

Faculty of Intellectual Property

(Manuscript received May,31 2004)

*英語コーパス学会第22回大会にて口頭発表 (2003年10月25日、明海大学)
平成14年度　松下視聴覚教育研究財団助成研究
平成15年度　科学研究費補助金基盤研究(C)(2)(No.15520378)

Abstract

This paper introduces a way to build a genre-based corpus that can be used as a teaching tool for ESP (English for Specific Purposes). Texts with specific communicative purposes have unique features not only in vocabulary and grammar but also in their rhetorical structure; therefore, it is expected that methods be devised to facilitate learning these genre-specific features. A sample of a genre-based corpus was created from the CEO's messages of the Fortune's top eight companies by tagging their rhetorical structure with XML (eXtensible Markup Language). Also developed was a text search system that would find and display parts of the texts including keywords and rhetorical information. The system allows us to see how certain words or expressions are used in the rhetorical structure of a genre text.It is also expected that the system be developed into an e-Learning tool.

This study was made possible by a grant provided by Matsushita Audio-Visual Education Foundation in 2002 and Grant-in-Aid for Scientific Research in 2003(No.15520378)

1. はじめに

専門分野におけるテキストには、語彙・文法レベルの特徴に加えて、文章構造や文体に分野固有の特徴が認められる。したがって、専門分野における英語運用能力を育成するためには、これらのジャンル特性を、学習者に効率的に習得させるための教育方法の開発が望まれる。そのステップとして本稿では、ESP（専門分野別英語）教育への応用に向けた、ジャンルコーパス作成の試みを紹介する。

2. ジャンルテキストにおけるレトリック構造

序論・本論・結論、あるいは起・承・転・結といった文章構造の記述は、主として段落（パラグラフ）の配置に基づく形式的な分類であり、およそあらゆるジャンルのテキストに適用することが可能であろう。しかし、分野によって固有の伝達内容と伝達目的を有するジャンルテキストでは、その伝達内容を効果的に伝え、伝達目的を達成するために、ジャンル固有の、より戦略的・合目的的なレトリック構造が存在すると考えられる。このレトリック構造を Move と Step という記述単位を用いて説明したのが Swales (1990) である。Swales によれば、Move とは Rhetorical Movement、すなわちジャンルテキストの内容を効果的に伝えるための情報配置を構成する単位であり、Step はそれぞれの Move におけるさらに具体的な下位範疇ということになる。たとえば、学術論文のイントロダクション（article introductions）というジャンルテキストに関して、Swales は次のような 3-move モデルを提唱している。（図-1）

Move 1 Establishing a territory（問題領域の確立）
　Step 1 Claiming centrality（重要性の主張）
　Step 2 Making topic generalizations（トピックの一般化）
　Step 3 Reviewing items of previous research（先行研究の批評）

Move 2 Establishing a niche（未解決領域の確立）
　Step 1A Counter-claiming（既存の考えに対する反論）
　Step 1B Indicating a gap（未解決領域の指摘）
　Step 1C Question-raising（問題提起）
　Step 1D Continuing a tradition（継承すべき既存の知見）

Move 3 Occupying the niche（未解決領域の解消）
　Step 1A Outlining purposes（研究目的の要旨）
　Step 1B Announcing present research（実験・調査内容の要旨）
　Step 2 Announcing principal findings（主な発見・実験結果の要旨）
　Step 3 Indicating RA structure（論文の構成）

図-1　Article IntroductionのMoveとStep (Swales 1990 *日本語訳は筆者)

本研究では、この Swales の枠組みを援用してジャンルテキストにレトリック情報を付加し、ジャンルコーパスを作成した。

3. コーパスの作成

3.1 ソースデータ

対象としたジャンルは、2001 年度 Fortune Top 50 企業の年次報告書の冒頭にある、CEO's メッセージ である。CEO's メッセージ は、株主（shareholder）を維持・拡大するという明確な伝達目的を持つジャンルであり、そのために情報の受け手である現在、あるいは将来見込める出資者に対して、信頼のおける企業イメージや企業活動の内容をいかに効果的に伝えるか、ということが、重要な要素となる。図-2 に General Motors 社の年次報告書の例を示す。中ほど、"DEAR STOCK HOLDERS"から始まる部分が、CEO's メッセージである。

ソーステキストは、インターネットの Fortune のサイトからダウンロードした。総語数は約 7 万語である。今回は、この内上位 8 社分のデータ（約 13,600 語）に、レトリックタグを付与して、ジャンル分析用のパイロットコーパスを作成した。[1]

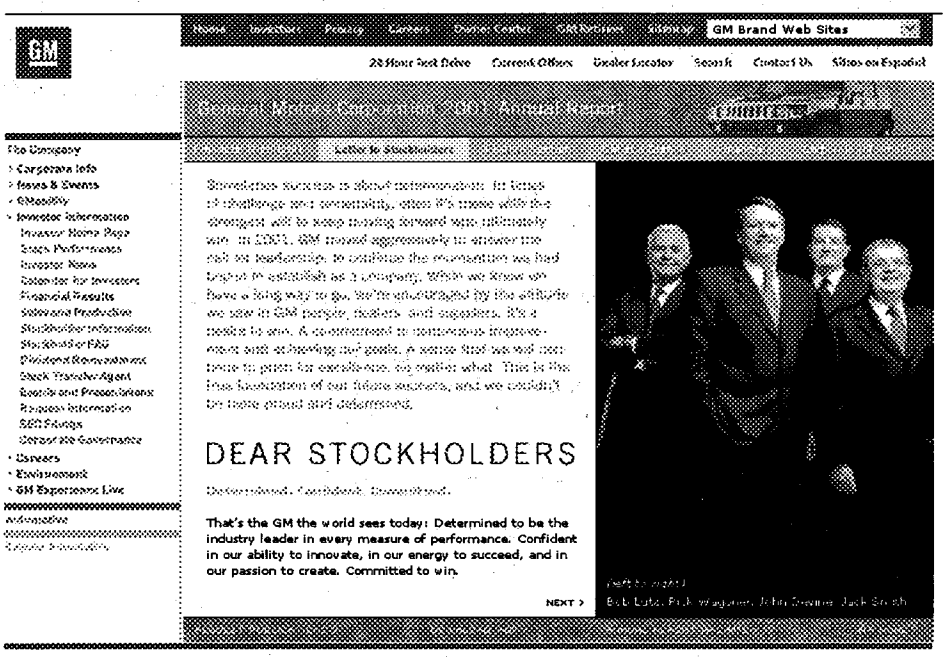


図-2 General Motors Corporation 2001 Annual Report
(http://gm.com/company/investor_information/financial_data/ar.htm)

3.2 CEO's メッセージのレトリック構造

各社のデータを検討した結果、筆者らの研究グループは、CEO's メッセージのレトリック構造として、以下の要素を抽出した。（図-3）

Move 1: Getting Attention（注意の喚起）
- Step 1 Attention line [ATTN]（呼びかけ）
- Step 2 Greeting the audience [GRTNG]（挨拶）
- Step 3 Projecting good corporate images [PRIMG]（良好な企業イメージの投影）

Move 2: Providing Information（企業情報の開示）
- Step 1 Providing background information [BGINFO]（背景状況の説明）
  - overall economic condition [/ECONO]（経済動向）
  - overall industry condition [/INDUS]（産業界の動向）
- Step 2 Providing performance data [PERFO]（業績内容の提示）
  - overall [/OVERALL]（全般）
  - sales and income [/SALES]（売上および収益）
  - cash-flow statement [/CASH]（キャッシュフロー）
  - stock price and/or dividend [/STOCK]（株価および配当）
  - market share [/MARKET]（市場占有率）
  - productivity [/PRDCTV]（生産性）
- Step 3 Presenting topics [TOPIC]（社内外の話題）
  - awards and recognition [/AWARD]（受賞）
  - social contribution [/SOCIAL]（社会貢献）
  - personal credit [/PERSONAL]（社内表彰）
  - world issues [/ISSUE]（世界情勢への関与）
- Step 4 Providing goals [GOAL]（企業目標の提示）
- Step 5 Presenting strategies [STRAT]（企業戦略の提示）
  - M & A [/M&A]（合併と吸収）
  - joint venture [/JOINT]（合弁事業）
  - R & D [/R&D]（研究開発）
  - investment in plant and equipment [/INVEST]（設備投資）
  - customer service [/CSVC]（顧客サービス）
  - restructuring [/RESTR]（経営改革）
- Step 6 Presenting future prospects [FUTURE]（将来展望提示）
- Step 7 Reinforcing good corporate images [REIMG]（企業イメージの強化）

Move 3: Closing（締めくくり）
- Step 1 Appeal for support [APPEAL]（支持へのアピール）
- Step 2 Signature line [SIGN]（署名）
- Step 3 Date line [DATE]（日付）

[共通のステップ]
- Step 0 Header [HEADER]（見出し）

[共通の属性]
その他　[/OTH]（その他）

図-3　CEO's メッセージのレトリック構造

3.3 タグの付与

3.2 で特定したレトリック構造に基づき、XML（eXtensible Markup Language）によるタグ付けを行った。[2] 図-4 に GM の例（一部）を示す。

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<?xml-stylesheet type="text/xsl" href="..¥AnnualReport.xsl" ?>

<AnnualReport title="GM Annual Report 2001">

<Move n="1" position="Getting Attention">
    <P>
        <Step n="1" description="ATTN" item="">
            <S>Dear Stockholders</S>
        </Step>
        <Step n="3" description="PRIMG" item="">
            <S>That's the GM the world sees today: Determined to be the industry
leader in every measure of performance.</S>
            <S>Confident in our ability to innovate, in our energy to succeed, and
in our passion to create.</S>
            <S>Committed to win.</S>
        </Step>
    </P>

    <P>
        <Step n="3" description="PRIMG" item="">
            <S>That determination, confidence, and commitment were clearly
evident to the world in 2001.</S>
            <S>It was a challenging year - one that none of us will ever forget.</S>
            <S>But it also showed GM's ability to build upon the solid foundation
laid over the past decade.</S>
        </Step>
    </P>
</Move>
```

図-4 XML タグ

図中の<Move> </Move>、<Step> </Step> は、それぞれ、Move と Step の開始タグと終了タグを表している。<P> はパラグラフ、<S> はセンテンスを表すタグである。また、Move や Step の番号や、Step のさらに詳しい情報は、それぞれのタグの中に属性情報として記してある。（例：<Step n="2" description="PERFO" item="SALES">

XML は、ホームページで用いられる HTML（Hyper Text Markup Language）と同様に、テキストに特定の情報を付加するために用いられる言語である。しかし、HTML の場合は、タグがあらかじめ定められており、しかもそれがテキストの体裁を整える機能しか持たないのに対して、XML には次のような優位な特徴がある。(1) タグを自由に設定することができ、意味的な情報や、その他様々な情報を付加することができる。(2) XML によってマークアップされたテキストは、タグの情報をもとに、様々な形に加工したり、検索をした

りすることが可能になる。(3) XMLはInternet ExplorerやNetscape Navigatorといったブラウザが標準で対応しているので、XMLでマークアップされたコーパスは、そのままWeb-baseのデータベースとして、情報の共有化を図ることが可能になる。

3.4 スタイルシート（XSL）

XMLでマークアップされたテキストは、そのままでは人の目に見づらい。そこで、表示形式を規定するスタイルシートを適用することによって、図-5のように見やすく表示させることができる。スタイルシートは、XMLのタグに応じてテキストを表示する形式を書き込んだプログラムで、これによって、同じXMLの文書を、目的にあわせていろいろな形で表示させることが可能になる。（ソースコードはAppendix 1を参照）

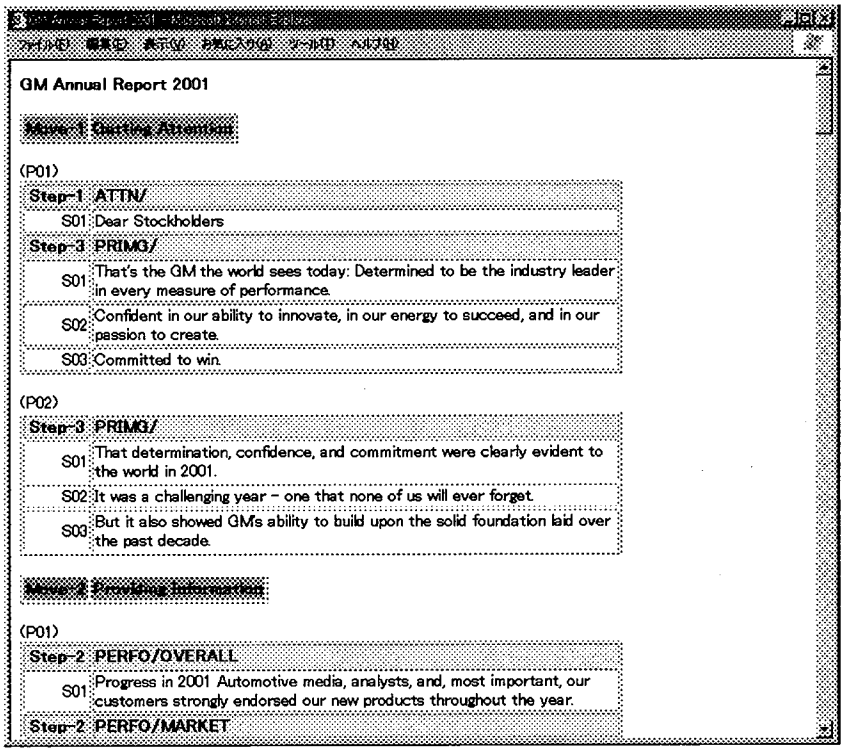


図-5　スタイルシートを適用した表示

4. テキスト検索システム

作成したコーパスから、ある語彙や表現が、ジャンルテキストのレトリック構造のどの部分で、どのように使われているかを容易に調べることができるようにするために、検索用のプログラムを作成した。[3]

4.1 キーワード検索

キーワード検索では、語句を入力すると、その語句を含んだテキストが、Move と Step によるレトリック情報とともに示される。

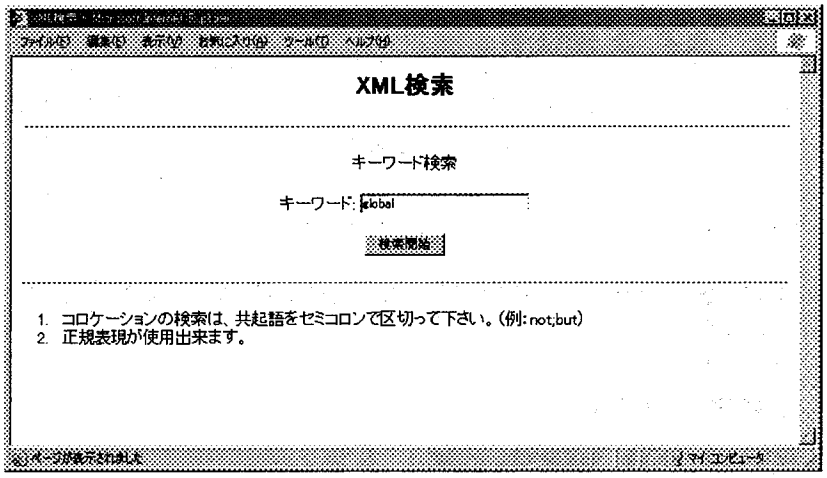


図-6 キーワード検索画面

たとえば、キーワードに"global"と入力すれば、図-7 のような結果が表示される。Ford の例では、Move-2 / Step-7（企業イメージの強化）と Move-2 / Step-4（企業目標の提示）において、それぞれ"global presence"、"global profits"という形で使われていることがわかる。また、複数の語をセミコロンで区切って入力することによって、単語の検索のみならず、スプリットイディオムや、離れた位置にあるコロケーションを検索することもできる。さらに、正規表現[4] を使えば、より複雑で柔軟な検索も可能である。

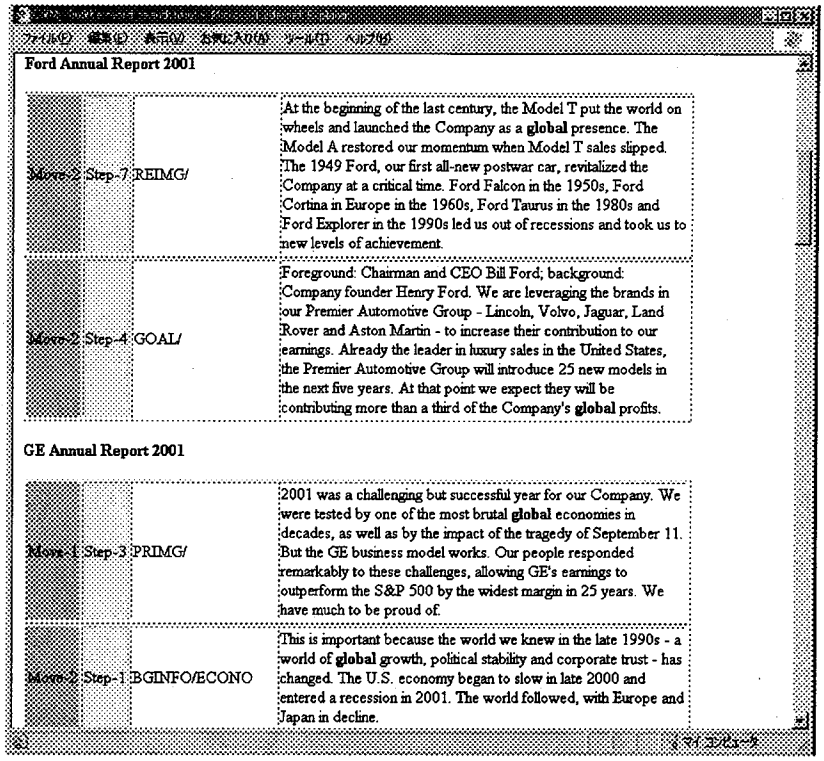


図-7　キーワード検索の結果

4.2　属性検索

次に、属性検索では、Step の属性を指定することによって、該当する箇所のテキストが表示される。

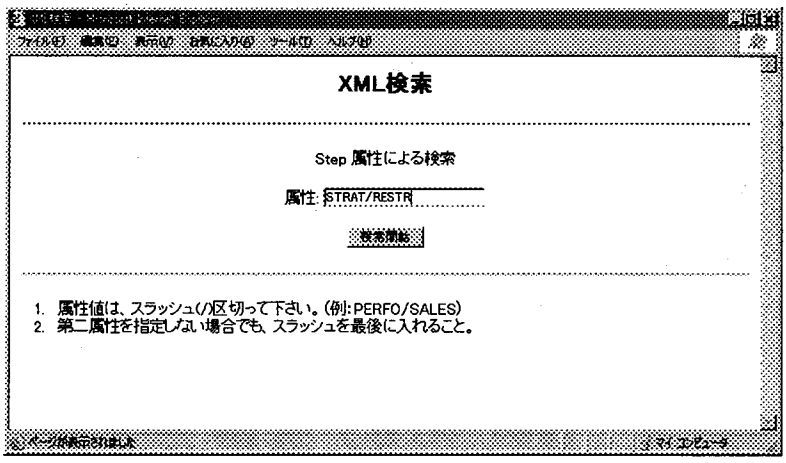
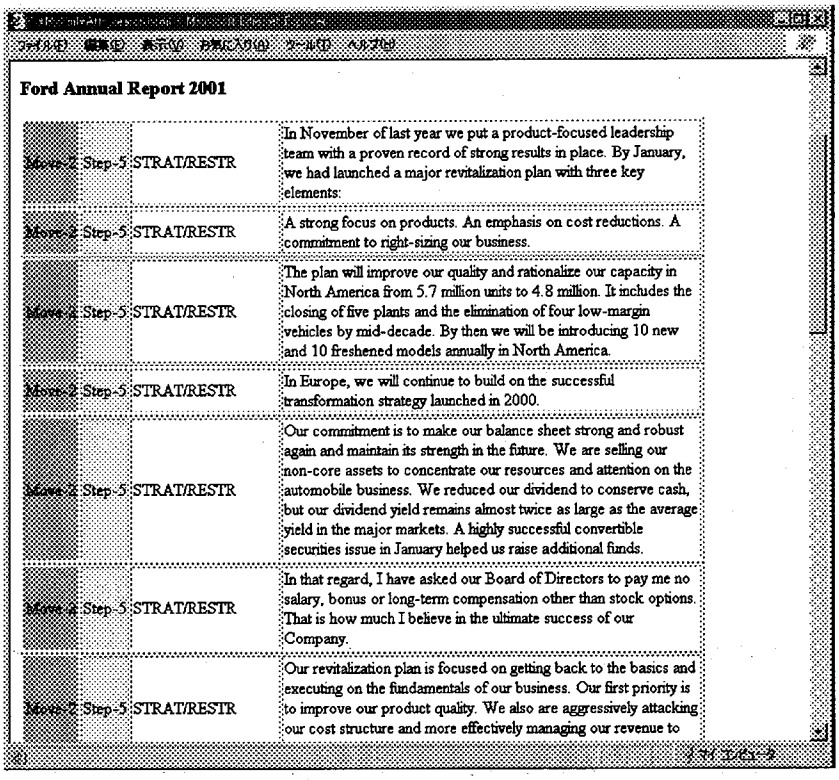


図-8　属性検索画面

たとえば、「企業戦略」として「リストラ」について述べている箇所を調べたい場合、"STRAT/RESTR" と入力すれば、図-9 のような結果が表示される。例から、関連用語として "revitalization", "cost-reduction", "right-sizing"といった語句が使われていることがわかる。先に述べたキーワード検索と、この属性検索を併用することによって、特定のレトリック構造において繰り返し現れる語彙や表現のパターンを容易に見つけることができる。



Ford Annual Report 2001

Move-2	Step-5	STRAT/RESTR	In November of last year we put a product-focused leadership team with a proven record of strong results in place. By January, we had launched a major revitalization plan with three key elements:
Move-2	Step-5	STRAT/RESTR	A strong focus on products. An emphasis on cost reductions. A commitment to right-sizing our business.
Move-4	Step-5	STRAT/RESTR	The plan will improve our quality and rationalize our capacity in North America from 5.7 million units to 4.8 million. It includes the closing of five plants and the elimination of four low-margin vehicles by mid-decade. By then we will be introducing 10 new and 10 freshened models annually in North America.
Move-2	Step-5	STRAT/RESTR	In Europe, we will continue to build on the successful transformation strategy launched in 2000.
Move-2	Step-5	STRAT/RESTR	Our commitment is to make our balance sheet strong and robust again and maintain its strength in the future. We are selling our non-core assets to concentrate our resources and attention on the automobile business. We reduced our dividend to conserve cash, but our dividend yield remains almost twice as large as the average yield in the major markets. A highly successful convertible securities issue in January helped us raise additional funds.
Move-4	Step-5	STRAT/RESTR	In that regard, I have asked our Board of Directors to pay me no salary, bonus or long-term compensation other than stock options. That is how much I believe in the ultimate success of our Company.
Move-7	Step-5	STRAT/RESTR	Our revitalization plan is focused on getting back to the basics and executing on the fundamentals of our business. Our first priority is to improve our product quality. We also are aggressively attacking our cost structure and more effectively managing our revenue to

図-9 属性検索の結果

5. まとめと課題

本稿では、専門分野別英語教育にコーパスを活用する方法として、XML を用いてレトリック情報を付与したジャンルコーパスの作成方法と、キーワードおよび属性によるテキスト検索プログラムの仕様について述べた。本稿で提示した方法論を基に、様々な分野のジャンルコーパスを作成することが可能である。今後は、データ量を拡大し、本格的なコーパスの作成に着手するとともに、これを教材開発へとつなげていくことが課題である。また、今回の研究過程において、最も困難な作業は、レトリック構造の特定であった。複数の研究者によ

る度重なる検討の結果、主観性をある程度排除し、妥当性をもたせることができたと考えられるが、多重性・多層性をもつテキストの切り取り方は千差万別であり、画一的なモデル化は、もとより不可能である。この点については、ジャンルテキストの伝達目的（communicative purpose）に照らして、より説明能力を持つ最適モデルの構築が、理論的研究の課題となる。

参考文献

Bhatia, V. K. (1993). *Analysing Genre: Language Use in Professional Settings*. Longman.

Swales, J. M. (1990). *Genre Analysis: English in academic and research settings*. Cambridge, UK: Cambridge University Press.

Garside R., G. Leech, A. McEnery. (1997). *Corpus Annotation*. Longman.

大津真. (2002). 『JavaScript プログラミング入門』オーム社.

川村博. (2001). 『XML 文書データベース』ソシム.

坂田健二. (2002). 『Windows で学ぶ XML』技術評論社.

升屋正人. (2002). 『はじめての JavaScript』ソフトバンク.

深山晶子・椋平淳・野口ジュディー・井村誠・深山源太・渡海淳子・松岡結. (2003). 「科学技術英語の e-Learning 用教材コンテンツ向上を目的としたコーパス分析の基礎研究」『平成 15 年度　松下視聴覚教育助成成果報告集　第 9 回（平成 14 年度採択分）研究開発助成　報告集』pp.168-177. 財団法人　松下視聴覚教育研究財団.

Appendix 1　スタイルシート（AnnualReport.xsl）

```
1  <?xml version="1.0" encoding="Shift_JIS" ?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3      <xsl:output method="html" encoding="Shift_JIS" />
4      <xsl:template match="/">
5          <html>
6          <head>
7          <title><xsl:value-of select="AnnualReport/@title" /></title>
8          </head>
9          <body>
10         <h3><xsl:value-of select="AnnualReport/title" /></h3>
11         <xsl:apply-templates select="AnnualReport" />
12         </body>
13         </html>
14     </xsl:template>
15     <xsl:template match="AnnualReport">
16         <p><b><xsl:value-of select="@title" /></b></p>
17         <xsl:for-each select="Move">
18            <table border="1">
```

```
19            <tr bgcolor="#00CCFF">
20                 <th>Move-<xsl:value-of select="@n" /></th>
21                 <th><xsl:value-of select="@position" /></th>
22            </tr>
23            </table><br />
24            <xsl:apply-templates select="P" />
25         </xsl:for-each>
26     </xsl:template>
27     <xsl:template match="P">
28         (P<xsl:number format="01" />)
29         <table border="1">
30         <xsl:for-each select="Step">
31            <tr bgcolor="#FFE4E1">
32                 <th width="60">
33                    Step-<xsl:value-of select="@n" />
34                 </th>
35                 <th align="left">
36                    <xsl:value-of select="@description" />/<xsl:value-of
   select="@item" />
37                 </th>
38            </tr>
39            <xsl:for-each select="S">
40            <tr>
41                 <td align="right">
42                    S<xsl:number format="01" />
43                 </td>
44                 <td width="500">
45                    <xsl:apply-templates select =". " />
46                 </td>
47            </tr>
48            </xsl:for-each>
49         </xsl:for-each>
50         </table>
51         <br />
52     </xsl:template>
53     <xsl:template match="hint">
54         <span style="font-weight:bold; color:#4169E1">
55            <xsl:value-of select="." />
56         </span>
57     </xsl:template>
58     <xsl:template match="text()">
59         <xsl:value-of select="." />
60     </xsl:template>
61 </xsl:stylesheet>
```

Appendix 2 キーワード検索プログラム（Keyword_search.html）

```
<html>
<head>
    <title>XML 検索</title>
    <script language="JavaScript">
    <!--
    // ==========================================
    // XML 文書データベース検索プログラム
    // (ver.3) 表示 Step 単位（コロケーションは文単位）
    // (c) Makoto Imura
    // 2003/08/22
    // ==========================================

    function getfiles(){

        //キーワードの取得
        var keyword = document.myForm.txtKey.value;

        var objFs,objFld,objFc,strFn;
        objFs = new ActiveXObject("Scripting.FileSystemObject");
        var strPath = "C:¥¥MyXml¥¥data";
        objFld = objFs.GetFolder(strPath);
        objFc = new Enumerator(objFld.Files);
        for (; !objFc.atEnd(); objFc.moveNext()){
           strFn = "data/"+objFc.item().Name;
           search(keyword,strFn);
        }
    }

        function search(keyword,strFn){

           //xml 文書オブジェクトの作成
           var objDoc = new ActiveXObject("Msxml2.DOMDocument");
           objDoc.async = false;
           objDoc.load(strFn);
           var objRoot=objDoc.documentElement;
           var clnTtl=objRoot.attributes;
           var objTtl=clnTtl.item(0);
           var strTtl=objTtl.text;
           document.write("<p><b>"+strTtl+"</b></p>");
           var objNodeList1 = objDoc.getElementsByTagName("S");
           var nsentence = objNodeList1.length;
           var objNodeList2 = objDoc.getElementsByTagName("Step");
           var nStep = objNodeList2.length;

           if(keyword.indexOf(';') != -1){
```

```
46
47              //コロケーション検索の場合
48
49              //keyword の分割
50              var aryKey = keyword.split(";");
51
52              //ループ内で使う変数の宣言
53              var objStc,stc,stcx,out,objReg,out,hit,clnAtr,stepn,dscr,attr,
54              objP,objMove,clnAtr,moven;
55
56              //パターンマッチ
57              var found = false;
58              var flag = 0;
59              for (i=0; i<nsentence; i++){
60                  objStc = objNodeList1.item(i);
61                  stc = objStc.text;
62                  stcx = stc;
63                  out ="";
64                  for(x in aryKey){
65                      objReg = new RegExp("¥¥b"+aryKey[x]+"¥¥b", "i");
66                      if (objReg.test(stcx)==true){
67                          found = true;
68                          hit = stcx.match(objReg);
69                          out = out+RegExp.leftContext+"<b>"+hit+"</b>";
70                          stcx = RegExp.rightContext;
71                      }else{
72                          found = false;
73                          break;
74                      }
75                  }
76                  if(found == true){
77
78                      //出力データの取得
79                      out = out+RegExp.rightContext;
80
81                      //ノードデータの取得
82                      objStep = objStc.parentNode;
83                      clnAtr = objStep.attributes;
84                      stepn = clnAtr.item(0).text;
85                      dscr = clnAtr.item(1).text;
86                      attr = clnAtr.item(2).text;
87                      objP = objStep.parentNode;
88                      objMove = objP.parentNode;
89                      clnAtr = objMove.attributes;
90                      moven = clnAtr.item(0).text;
91
92                      //作表
```

```
93                      document.write("<table border=1>");
94                      document.write("<tr>");
95                      document.write('<td bgcolor="#00CCFF">');
96                      document.write("Move-"+moven);
97                      document.write("</td>");
98                      document.write('<td bgcolor="#FFE4E1">');
99                      document.write("Step-"+stepn);
100                     document.write("</td>");
101                     document.write("<td width=140>");
102                     document.write(dscr+"/"+attr);
103                     document.write("</td>");
104                     document.write("<td width=400>");
105                     document.write(out);
106                     document.write("</td>");
107                     document.write("</tr>");
108                     document.write("</table>");
109
110                     //発見フラグ
111                     flag = 1;
112                 }
113             }
114             if (flag != 1){
115                 document.write("  (No matching data...)");
116             }
117         }else{
118
119             //通常のキーワード検索の場合
120
121             var objReg = new RegExp("¥¥b"+keyword+"¥¥b", "i");
122             var found = false;
123
124             //ループの中で使う変数の宣言
125             var objStep,step,stepx,found,clnAtr,stepn,dscr,attr,objP,
126             objMove,moven,hit,left,out;
127
128             for (i=0; i<nStep; i++){
129                 objStep = objNodeList2.item(i);
130                 step = objStep.text;
131                 left = "";
132                 if (objReg.test(step)==true){
133                     found = true;
134
135                     //ノードデータの取得
136                     clnAtr = objStep.attributes;
137                     stepn = clnAtr.item(0).text;
138                     dscr = clnAtr.item(1).text;
139                     attr = clnAtr.item(2).text;
```

```
140                         objP = objStep.parentNode;
141                         objMove = objP.parentNode;
142                         clnAtr = objMove.attributes;
143                         moven = clnAtr.item(0).text;
144
145                         stepx = step;
146                         while (stepx.search(objReg) != -1){
147                             hit = stepx.match(objReg);
148                             left = left+RegExp.leftContext+"<b>"+hit+"</b>";
149                             stepx = RegExp.rightContext;
150                         }
151                         out = left+stepx;
152
153                         //作表
154                         document.write("<table border=1>");
155                         document.write("<tr>");
156                         document.write('<td bgcolor="#00CCFF">');
157                         document.write("Move-"+moven);
158                         document.write("</td>");
159                         document.write('<td bgcolor="#FFE4E1">');
160                         document.write("Step-"+stepn);
161                         document.write("</td>");
162                         document.write("<td width=140>");
163                         document.write(dscr+"/"+attr);
164                         document.write("</td>");
165                         document.write("<td width=400>");
166                         document.write(out);
167                         document.write("</td>");
168                         document.write("</tr>");
169                         document.write("</table>");
170                     }
171                 }
172                 if (found== false){
173                     document.write("  (No matching data...)");
174                 }
175             }
176         }
177     //-->
178     </script>
179</head>
180<body>
181     <center>
182         <form name="myForm">
183         <h2>XML 検索</h2><hr>
184         <p>キーワード検索</p>
185         <p>キーワード: <input type="text" size="30" name="txtKey"></p>
186         <p><input type="button" value="検索開始" onclick="getfiles()"></p>
```

```
187          <!--p><input type="button" value="検索開始" onclick="search()"></p-->
188          </form>
189      </center>
190      <hr>
191      <OL>
192      <LI>コロケーションの検索は、共起語をセミコロンで区切って下さい。（例：not;but）
193      <LI>正規表現が使用出来ます。
194      </OL>
195</body>
196</html>
```

Appendix 3 属性検索プログラム（Attr_search.html）

```
1  <html>
2  <head>
3        <title>XML 検索</title>
4        <script language="JavaScript">
5        <!--
6        // ========================================
7        // XML 文書データベース検索プログラム
8        // (ver.1) Step 属性による検索
9        // (c) Makoto Imura
10       // 2003/08/23
11       // ========================================
12
13       function getfiles(){
14
15           //キーワードの取得
16           var strAttr = document.myForm.txtAttr.value;
17           document.write("<h2> <font color='blue'>Step: "+strAttr+"</font></h2>");
18
19           //ファイルアクセス
20           var objFs,objFld,objFc,strFn;
21           objFs = new ActiveXObject("Scripting.FileSystemObject");
22           var strPath = "C:¥¥MyXml¥¥data";
23           objFld = objFs.GetFolder(strPath);
24           objFc = new Enumerator(objFld.Files);
25           for (; !objFc.atEnd(); objFc.moveNext()){
26              strFn = "data/"+objFc.item().Name;
27              search(strAttr,strFn);
28           }
29       }
30
31       function search(strAttr,strFn){
32
33           //strAttr の分割
```

```
34          var aryAttr = strAttr.split("/");
35          var objAttr1 = new RegExp(aryAttr[0]);
36          var objAttr2 = new RegExp(aryAttr[1]);
37
38
39          //xml 文書オブジェクトの作成
40          var objDoc = new ActiveXObject("Msxml2.DOMDocument");
41          objDoc.async = false;
42          objDoc.load(strFn);
43          var objRoot=objDoc.documentElement;
44          var clnTtl=objRoot.attributes;
45          var objTtl=clnTtl.item(0);
46          var strTtl=objTtl.text;
47          document.write("<h3>"+strTtl+"</h3>");
48          var objNodeList = objDoc.getElementsByTagName("Step");
49          var nStep = objNodeList.length;
50
51          //ループ内で使う変数の宣言
52          var objStep,clnAttr,objItem1,strItem1,objItem2,strItem2,strOut,
    stepn,objP,objMove,moven,flag;
53
54          //パターンマッチ
55
56          for (i=0; i<nStep; i++){
57             objStep = objNodeList.item(i);
58             clnAttr = objStep.attributes;
59             objItem1 = clnAttr.item(1);
60             strItem1 = objItem1.text;
61             objItem2 = clnAttr.item(2);
62             strItem2 = objItem2.text;
63             if (objAttr1.test(strItem1) && objAttr2.test(strItem2)){
64
65                //出力データの取得
66                strOut = objStep.text;
67
68                //ノードデータの取得
69                stepn = clnAttr.item(0).text;
70                objP = objStep.parentNode;
71                objMove = objP.parentNode;
72                clnAtr = objMove.attributes;
73                moven = clnAtr.item(0).text;
74
75                //作表
76                document.write("<table border=1>");
77                document.write("<tr>");
78                document.write('<td bgcolor="#00CCFF">');
79                document.write("Move-"+moven);
```

```
80                  document.write("</td>");
81                  document.write('<td bgcolor="#FFE4E1">');
82                  document.write("Step-"+stepn);
83                  document.write("</td>");
84                  document.write('<td width=140>');
85                  document.write(strItem1+"/"+strItem2);
86                  document.write("</td>");
87                  document.write("<td width=400>");
88                  document.write(strOut);
89                  document.write("</td>");
90                  document.write("</tr>");
91                  document.write("</table>");
92
93                  //発見フラグ
94                  flag = 1;
95              }
96          }
97          if (flag != 1){
98              document.write("  (No matching data...)");
99          }
100     }
101     //-->
102     </script>
103</head>
104<body>
105     <center>
106         <form name="myForm">
107         <h2>XML 検索</h2><hr>
108         <p>Step 属性による検索</p>
109         <p>属性: <input type="text" size="30" name="txtAttr"></p>
110         <p><input type="button" value="検索開始" onclick="getfiles()"></p>
111         </form>
112     </center>
113     <hr>
114     <OL>
115     <LI>属性値は、スラッシュ(/)区切って下さい。（例：PERFO/SALES）
116     <LI>第二属性を指定しない場合でも、スラッシュを最後に入れること。
117     </OL>
118</body>
119</html>
```

（注）

[1](1)Walmart (2)Exxon (3)GM (4)Ford (5) Enron (6)GE (7)ChevronTexaco (8)IBM　ただし、Enron のみ 2000 年度のデータ。

[2] タグ付けにあたっては、共同研究者の深山源太氏が独自に開発した XML エディタ、

Itag.exe（© Genta Miyama 2003）を用いた。

[3] 開発言語は Windows に標準装備されている、Jscript（JavaScript）を用いた。ソースコードは、Appendix を参照。プログラムは HTML ファイルの中に書き込まれており、HTML ファイルを開くだけで起動する。コーパスファイルは、"data"という名前のフォルダに格納し、HTML ファイルと同じ場所においておく。

[4] 文字列のパターンを検索するための記号。たとえば、ピリオドは任意の一文字をあらわすので、"s.ng"と入力すれば、（sing, sang, sung, song）などが、一度に検索できる。