Perl によるコンコーダンス作成ソフトの開発

重 見 晋 也

はじめに：コンコーダンス作成の現状

文学・言語学の研究に限らず，テクストを一次資料として用いるような研究において，そのテクストのコンコーダンスを作成して研究の資料として用いることは，研究に多大な成果をもたらしている．自らテクストを電子化し，テクスト・データとして保存し活用しようとしている研究者の数は年々増加しており，internet を媒介としたメーリングリストの中にも，こうしたテクストデータの活用を目的としたものが数多く存在する．とはいえ，コンコーダンスを実際に作成するとなると，ソフトウェアの選択や操作に問題があり，誰でも簡単に作成できるわけではない，というのが現状であるように思われる．

あるテクストのコンコーダンスを実際に作成しようとする場合で，そのテクストの電子化がすでになされているような場合には，internet などを通じて入手可能な既存のコンコーダンス作成ソフトを活用することができる．よく知られているものとしては Conc や Le Concordeur/Concorder/Free Text Browser/HyperBase などがある．

これらのソフトは，それ自体がコンコーダンスを作成する目的で開発されたものであるが，こうしたコンコーダンス作成ソフト以外にも，NisusWriter という Macintosh 用のワープロ・ソフトを用いて簡単に単語のインデックスを作成する方法や[1]，やはりこれも Macintosh 用のエディタである Edit7 と統合ソフトであるクラリス・ワークスを用いる方法[2]なども報告されている．

しかしこれらのコンコーダンス作成ソフトの場合には，多くの機能を持ってはいるものの，動作が遅い，元のテクストが大きすぎると動作が不安定になる，ある特定の OS しかサポートしていないなど，使用する際に問題点も指摘されている．

そこで，本論においては，平成九年度の『フランス文学におけるデータベースの構築と有効的活用に関する研究』プロジェクトの一環として，独自に開発したコンコーダンス作成プログラム contexpro を紹介するとともに，そのソース・コードを公開する．ただし，開発者自身は Macintosh ユーザーであり，Windows を使用する環境をもたないため Windows 上での動作を確認することができなかった．それ故，本論においては，contexpro の開発環境である UNIX と Macintosh に限定して論を進めたい．

I. contexpro の開発・動作環境及び Perl について

開発環境としては，昨今注目されている PC-UNIX 環境の一つで，Macintosh

上で動作するMkLinuxDR2.1update5をOSに用い，これをPowerMacintosh8500上にインストールして開発を行った．

プログラミング言語としては，インタプリタ型言語の一つであるPerl5.003を用いている[3]．Perlはinternetの Webサイトにおいて，動的ホームページを作成する際に，CGI(Common Gateway Interface)スクリプトとして用いられることで知られている言語である．Perl4からPerl5へとバージョン・アップされた際に，オブジェクト指向プログラミングの要素を取り入れ，より現代的なプログラミング言語に進化した．

今回はこのPerlをプログラミング言語して用いたわけであるが，というのは，Perlがインタプリタ型の言語であるため，その実行に際してコンパイルする必要がなく，開発時間が短縮できるという理由からである．また，Perl自体がMacintoshやWindows3.1/95/NT（以下全てWindowsと略す）といった複数のOSに移植されているため，一つのコードをそのままマルチプラットフォームで実行可能であることも，理由の一つとして挙げられる．

実際，今回開発したcontexproも，前述したように，その開発の大部分をMkLinux上で行いはしたものの，バグ・フィクスの作業は，Macintosh上にMacPerlをインストールして行った．そして，そのようにして書いたプログラムをMacintosh及びUNIX上で動かしても，全く問題なく動作するのである．この点，C/C++などと大きく異なる点であろう．

しかしPerlはC/C++などのようにコンパイル型の言語ではないため，Perlで書いたプログラムを動かすためには，使用するパソコンあるいはワークステーションにPerlインタプリタがインストールされていることが必要になる．この点は，Perlを初めて使う場合には障害となるかもしれない．また，Windowsに移植されたPerlはAPIに重大なバグが見つかっており，正常に動作しない可能性があることを述べておく．

PerlがWebサイトにおいてCGIスクリプトとして用いられていることはすでに述べたが，このことにより，現在の大学のワークステーションにはPerlがインストールされている場合が多い．また，MacintoshやWindowsといったパソコン上で動作するPerlもinternetを通じて無料で入手可能である．パソコン上で動くPerlについては，`http://language.perl.com/info/software.html`で最新の情報を得ることができる．

II. contexpro

II.1. インターフェース

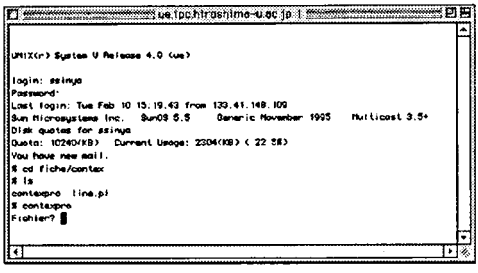
インターフェースはコンピュータとユーザーとの間を取り持つ重要な要素であり，如何にすばらしいプログラムを作ったとしても，インターフェースがわ

かりやすくなければ，実際にはなかなか使いにくいものとなってしまう．その好例が，GUI とCUI の差ということになるだろう．如何にUNIX が強力で安定したOSとCPUを備えていようとも，個人レベルではMacintoshやWindowsなどのGUI 環境のOS が圧倒的に利用されていることの所以である．

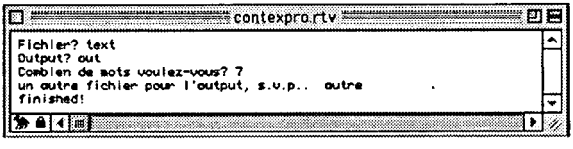
しかし，残念なことに，GUI ベースのMacintosh においても，Perl を動かすためには，UNIX のようなCUI 環境に逆戻りしなければならない．

それではPerl あるいはcontexpro はどのようなインターフェースをもっているのだろうか．

次に示すのは，MacPerl でcontexpro を実行した際の画面と，telnet で経由でMacintoshからUNIXシステムにログインしてcontexpro を実行した際の画面である．



<図1：UNIX>



<図2：Macintosh>

ここに示した二つのウィンドウの中には，全くグラフィカルな要素はなく，一見するとどちらがUNIXでどちらがMacintoshなのか判別しづらい．その上，全ての操作はキーボードからの入力で行う．

しかし，contexpro はUNIX を操作するためのコマンドのような類をいっさい用いなくてすむようになっているため，初めてこのプログラムを実行する場合にも，それほど困惑することはないであろう．実際にプログラムを実行する際の問題については後述する．

II.2. ソースコード

まず最初にcontexpro のソース・コードを以下に示す．contexpro 自体は，Windows3.1などのconfig.sysと同じくテクストファイルであって，それを各環

境にインストールされているPerlインタプリタが実行していくことになる.なお，ソースコード左の数字は，説明の便宜のために付した行番号である.

```
1:  #!/usr/bin/perl
2:  ################ [sub1] ###################
3:  #  不要な記号類の削除
4:  ##########################################
5:  sub conditions {
6:            if ($a[0] =~ /[A-Z]/) {
7:              $a[0] =~ tr/A-Z/a-z/;
8:            }
9:
10:           if ($a[0] =~ /[\~\\\.\,\:\+\*\#\»\;!?\(\)\>\<-]/
) {
11:             $a[0] =~ s/[\~\\\.\,\:\+\*\#\»\;!?\(\)\>\<-]//
g;
12:                 }
13:       }
14:
15:
16: ################[sub2]###################
17: #  メインのところてん式処理
18: ##########################################
19:
20: sub core {
21:                   if ($mot =~ /\[\w+.+\]/)      {
22:                             s/($1)//g;
23:                         }     else {
24:           $total++;
25:           $line[$l] = $mot;
26:           @a = (@a, $mot, $no, $total, $pageno);
27:           conditions;
28:           shift(@line);
29:           print OUT "$a[0] $a[1] $a[2]| $a[3]@ @line\n";
30:           shift (@a);
31:           shift (@a);
32:           shift (@a);
33:           shift (@a);
34:                   }
35:       }
```

```
36:
37: #################[sub3]####################
38: #  ソートのサブルーチン
39: ########################################
40:
41: sub now {
42:     @indexa = split (/ /, $a);
43:     @indexb = split (/ /, $b);
44:       ($indexa[0] cmp $indexb[0]) || ($indexa[2] <=>
$indexb[2])||($indexa[1] <=> $indexb[1]);
45: }
46:
47: ##################[body]##################
48:
49: print "Fichier? ";
50: $file = <STDIN>;          #入力用ファイル・ハンドルのオープン
51:
52: print "Output? ";
53: $output = <STDIN>;        #出力用ファイル・ハンドルのオープン
54:
55: print "Combien de mots voulez-vous? ";
56: $l = <STDIN>;             #コンテクストの単語数指定
57:
58: while ($l % 2 < 1) {
59:     print "Ooops! Pair is Unfair!\nPrint Inpair number.\n";
60:     print "How many words d'you want? ";
61:     $l = <STDIN>;
62: }
63:
64: open (IN, $file) || die ": couldn't open $file.\n";
65: open (OUT, ">$output") || die "couldn't open $output.\n";
66:
67:
68: $ctr = ($l - 1) / 2;
69: $no = 1;                        #行番号をセット
70: $total = 0;                     #先頭からの単語数をセット
71: $a[$ctr*4 - 1] = 0;             #指定された語数の空配列を準備
72:
73: foreach $array (<IN>) {
74:     @str = split(/ /, $array);
```

```
75:     foreach $mot (@str) {
76:       if ($mot =~ /\[\w+.+\]/)      {  #章あるいは頁番号の処理
77:             chomp ($mot);
78:             $pageno = $mot;
79:             $no = 1;
80:             }
81:       if ($mot =~ /\n/) {
82:           chomp($mot);
83:           core;
84:           $no++;
85:                 } else {
86:           core;
87:                 }
88:  }
89: }
90:
91:
92: while ($line[$ctr]){              #最後の $ctr 個分の単語を処理
93:           conditions;
94:           shift(@line);
95:           print OUT "$a[0] $a[1] $a[2]| $a[3]@ @line\n";
    #出力ファイルへの書き出し
96:           shift (@a);
97:           shift (@a);
98:           shift (@a);
99:           shift (@a);
100:}
101:
102:
103:close (IN);                  #ファイル・ハンドルのクローズ
104:close (OUT);
105:
106:print "un autre fichier pour l'output, s.v.p..  ";
107:$fin = <STDIN>;
108:
109:open (IN, $output);
110:open (OUT, ">$fin");
111:
112:foreach (<IN>) {
113:    ($fst, $snd) = split (/\|/);
```

```
114:       chomp ($fst);
115:       chomp ($snd);
116:     $flag{$fst} = $snd;
117:}
118:
119:@srtd = sort now keys (%flag);
120:
121:foreach (@srtd) {
122:       @last = split(/ /, $_);
123:       @ref = split (/\@/, $flag{$_});
124:       @conref = split (/ /, $ref[1]);
125:print  OUT  "\t$last[0],  $ref[0],  $last[1]
\t@conref[0..$ctr]\t$conref[$ctr+1]\t@conref[$ctr+2..$1]\n";
    #最終結果の出力
126:}
127:
128:print "finished!\n";          #終了メッセージ
129:
130:print OUT "\n$total mots sont contenus, en totale, dans le
texte.\nMERCI!\n";
131:print OUT "\n This programme is brought to you by SHIGEMI
Shinya. All rights reserved.\n";
132:print OUT "Ce logiciel est realise par SHIGEMI Shinya. Tous
les droits sont reserves.\n";
133:
134:close (IN);
135:close (OUT);
```

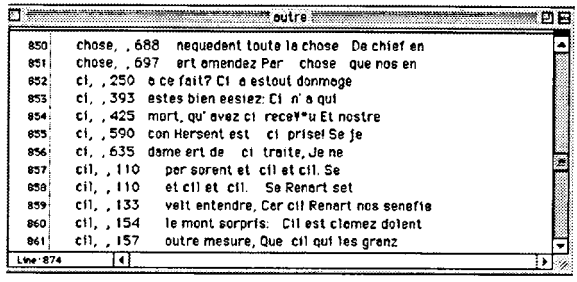
II.3. アルゴリズム

contexpro のアルゴリズムは二つの段階に分けることができる．まず，contexpro は処理すべきファイルが指定されると，ファイルの先頭から最後まで指定された数（奇数でなければならない）の分だけ，テクストの先頭から，単語を順次抜き出し，その単語の列の中央にある単語をキーの単語として別に取り出す．それと同時に，キーとして抜き出した単語がそのテクストの何行目にありそれが先頭から何単語目に当たるかを出力する．この際に，単語の区切れはスペースで挟まれた文字列として認識されている．出力先は第53行目で指定されるファイルであり，ディレクトリはcontexpro と同じディレクトリである．図3に最終的な出力結果の例を挙げておく．

このようにして，抜き出した単語とその単語の参照値，及びその単語を中心

として，指定された語数で構成されたコンテクストが一行に出力される．この作業は元のテクストが終わるまで続けられる．

第二段階では，第一段階で指定されたファイルに保存されているデータを元にして，キーとなる語によってソートし，その結果を第107行目でユーザーが指定したファイルに保存する．以上がcontexproのアルゴリズムの概観である．



<図3：出力例>

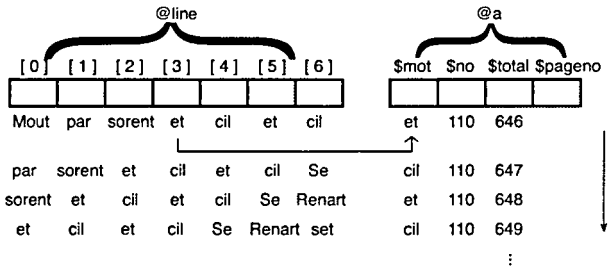
それではもう少し細かくプログラムを見ていくことにしよう．

このプログラミングの第1行目は，UNIX上でcontexproを使用する際に必要なものであり，使用するUNIXシステム上にインストールされているPerlへの絶対パスを示している．これにより，UNIX上のどのディレクトリにcontexproが置かれていても，シェル上で`contexpro`と入力するだけで，contexproが実行できるようになっている．但し，Perlへのパスはシステムによって異なるため，利用するUNIX環境によってはこの第1行目のパスを変更する必要が出てくる．Macintosh環境では，この第1行目は不要であるが，実行に何ら支障を来すものではない．

第2行目から第45行目まではサブルーチンであり，プログラム本体（「`[body]`」として示している部分以下）で繰り返される処理を別に記述したものである．サブルーチンは全部で3つあり，最初のサブルーチンはキーとして取り出した単語についている不要な記号を取り除くためのものである．`sub conditions`となっているが，`conditions`がこのサブルーチンの名前であり，第27行目と第93行目に用いている．

第二のものは，「`[sub2]`」であるが，これが処理の第一段階において一番重要な部分である．contexproでは角括弧（[ ]）で囲まれた文字列があれば，それをページ番号として認識するようになっている．そのため，文字列が角括弧に囲まれてる場合には，その文字列を消去することによってコンコーダンスのキーになる単語とならないよう処理するのである．そして，文字列が角括弧に囲まれていなければ，contexproは，テクストの先頭から一単語づつ取り出し，ところてん式に，あらかじめ用意した配列`@a`に代入していく．このサブルーチンには`core`という名前を付けており，第83行目と第86行目に使用し

ている．図式化すればおおよそ図4のようになる．



<図4：概念図>

三番目のサブルーチンは，第一段階で作られたリストを，キー単語などを基準としてソートするためのものである．第119行目で用いているのが，このサブルーチンである．ソートする場合に，キー単語だけを基準としてソートしただけでは，同一行に同一語形が出てきた場合に意図したとおりに並び替えが実行されない．そのため，キーとなる単語の位置する行番号と，その単語の先頭からの単語番号をも，併せて参照してソートするようにしている．第44行目に見られるcmp及び<=>は，スカラー演算子の一つで，左右のオペランドを，前者は文字列として比較を行い，後者は数値として比較を行う．

第47行目以降がcontexproのメインのプログラムである．第49行目から第65行目の部分が，ユーザーに入力を促す部分である．第49・50行目では処理するファイルをユーザーに入力してもらい，第64行目で入力用のファイル・ハンドルをオープンしている．それに対して，次の第52・53行目では，出力用のファイルをユーザーに指定してもらっており，第65行目で出力用のファイル・ハンドルをオープンしている．

第55・56行目では，ユーザーにコンテクストを表示するのに必要な単語数を入力してもらっており，それをスカラー値$1に代入する．第58行目以降は，このときに奇数が入力されたかどうかをチェックし，奇数でなければ第59・60行目で再入力を促すようになっている．

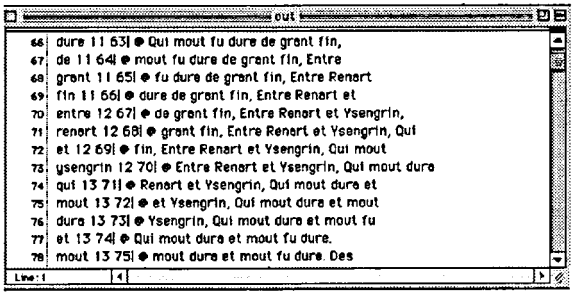
contexproでは，第56行目（入力を失敗した場合には第61行目）の入力の際に指定された数から，コンテクストの出力に必要な配列の要素数を計算する（第71行目）．その後に，図4で示したように，電子テクストの先頭から指定された単語数分を一単語ずつずらしながら出力ファイルに切り出していく．

第95行目に見られるように，出力にあたってはCなどで一般的に用いられるprintf関数を用いていないが，それは処理にかかる時間を短縮するためであ

る．ここまでが前述した第一段階に相当している．図5に示すのが，第一段階の出力例である．

| 及び@ は第二段階で処理をするためにつけるもので，ソートで比較を行う際のオペランドになる．

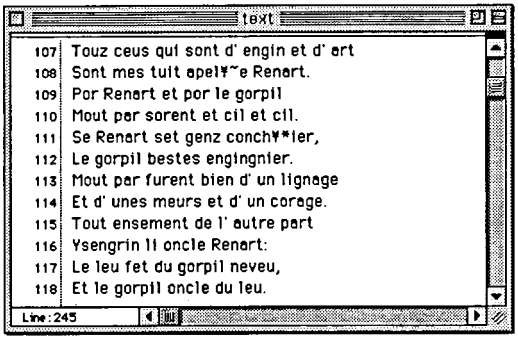
第103・104行目で第一段階でのファイル・ハンドルをクローズしている．



<図5：出力例1>

第二段階は第109-127行目の間で，ソートとソート結果の書き出しを処理している．ソートした結果を出力するために，第109・110行目で，第一段階で出力されたファイルを自動的に読み込み，ユーザーによって指定される最終的な出力先のファイル・ハンドルをオープンしている．ソートの際に問題となるのは，前述したように，元のテクストの一行の中に同じ単語がでてくる場合である．というのも，Perlのsort関数を用いてプログラムにキーとなる単語だけを頼りにソートさせただけでは，上述のような場合には単語が正しくソートされず，ソートの結果が不正確なものになるからである．

こうした問題を解決するために，contexproでは，第一段階でキーとなる単語にテクストの先頭から何番目の単語になるかということも，併せて参照させておいた．これにより，一行中に同じ単語がでてきたとしても，プログラムはそれらを別々の単語として認識し，正しくソートすることができるようになる．次のような例を示しておく．



<図6：同一行にある同じ単語>

これは『狐物語』の107行目から118行目であるが，このように110行目には«cil»という単語が二度

でてきている．これらの語を«cil»だけを頼りにソートさせたのでは，正しくソートされない．しかし，先頭からの単語番号をあらかじめ付けておくことにより，同じ«cil»でも，図3に示したように先頭からの出現順でソートされて出力されるのである．こうして同一行にある同一単語も，別々の単語としてプログラムには認識されるようになり，結果として，単語の出現順にコンコーダンスに組み込まれるようになるのである．

contexproの第125行目は非常に煩雑な`print`命令になっているが，これは，不要な記号類を取り除く処理を第122-124行目で行っているためと，キーワードの前後や参照番号の前後にタブ記号`\t`を挿入するためである．マルチプラットフォームでの使用を考えると，printf関数よりも直接タブ記号を挿入しておく方が使い勝手がよいと考えたからである．

III. contexproの利用

III.1. 事前に必要な処理

まず最初に述べておかなければならないのは，contexproが処理するテクストの形式についてである．

あるテクストのコンコーダンスを作成する場合には，そのテクストの基となるエディションが存在する．そして，コンコーダンスはそのエディションのページ番号や行番号などを参照するのが一般的であろう．もちろん韻文においては，ページ数への参照は不要となることもあろうし，あるいはそのテクストの全体の行数などが決まっているために，特定のエディションへの依拠が不要な場合もあるかもしれない．しかし，作品によっては，エディションによって，テクスト自体が異なっている場合もあろう．また特に，テクストが散文である場合には，基となるエディションへの参照が不可欠なものとなってくる．そうでなければ，ページ番号や行番号への参照は不可能になり，結果的には出来上がったコンコーダンスの有用性は著しく低減することになる．

そこで，contexproを実行してコンコーダンスを作成する際に重要になってくるのが，そうした元のテクストにあわせて，改行を入れるという作業である．OCRなどによってテクストをデジタル化する際に，元のテクストを忠実にデジタル化することは，言わずもがなであるが，それだけでなく，あるエディションの一行が電子テクストにおいても同じ一行でなければならない．

また，ページ番号を参照させたい場合には(ページ番号でなくとも章番号などでも良い)，頁の先頭の単語の前に「[67]」のように角括弧をあらかじめ挿入しておくことも必要である．対象とするテクストにおいて角括弧が用いられている場合は，これを別の記号に置き換えることが必要になってくるが，現段階では，contexproは角括弧に囲まれた数字をページ番号として認識すること

になっている.

III.2. 使用法

ここでは，contexpro の操作法を具体的に見ていくことにする.

まずcontexpro を起動しなければならない．これは，Macintosh と UNIX とで方法が異なっている. Macintoshにおいては, MacPerl を起動した後メニューから「Run script」メニューを選び，その際に現れるオープン・ダイアログで，「contexpro」を選択する．UNIX では，カレント・ディレクトリを contexpro のおいてあるディレクトリに移動させた後，プロンプトに続いて「`contexpro`」とタイプすれば，プログラムが起動する.

一度プログラムが起動してしまえば，後はUNIX でも Macintosh でも，操作は同じである．ただ，先程も述べたように，入力にはマウスをいっさい用いず，全てキーボードから入力しなければならない.

contexpro を起動すると，「`Fichier?`」と表示される．ここでは，処理するテクストのファイル名を入力する. 例えば，«text»という名前で処理すべき電子テクストが保存されているのであれば，「`Fichier?`」の後に，「`text`」と入力し，return キーを押す．ここで注意しなければならないのは，contexpro と処理すべきテクストとが同じディレクトリ（Macintosh ではフォルダ）に置かれていなければならないということである. 確かに同じディレクトリになくてもパスを入力すれば作業は可能であるが, Perlは常に現在実行しているファイルから，対象となるファイルへの相対パスを要求するため，特に UNIX 上では, 処理したいファイルのパスを探すのに大変苦労することも考えられる. また，Macintosh ユーザーの場合には，ファイルのパスを書くことに慣れていないことが多いであろう．それ故，contexpro と処理される電子テクストとが同じディレクトリにあった方が，入力の手間も省ける上，混乱も少ない.

処理の対象となるファイル名を入力し `return` キーを押すと，次に「`Output?`」と表示される．ここには任意の文字列を入力すれば，それがファイル名となる. そのファイルには第一段階の作業の結果が保存される. 例えば「`out`」などと入力し return キーを押す.

出力用のファイルを指定すると，次に「`Combien de mots voulez-vous?`」と表示される.「何単語必要ですか？」ということだが，ここには奇数で任意の数字を入力し return キーを押す．もし偶数を入力した場合は，奇数を入力するように促すダイアログが表示されるので，奇数を入力する.

これで第一段階の操作が終了である.

次に処理が終了すると，contexpro は第二段階に入り，「`un autre ficher pour l'output, s.v.p..`」と表示する．これは新たに出力用のファイル

を指定することを促すものである．それ故，ここでは，先ほど「Output?」と表示された際に指定したのとは別の名前を入力する．例えば「autre」などである．この第二段階は，前述したように，最初の「out」に出力されたデータを最終的に並べ替えるためのものであって，この«autre»というファイルに，作成されたコンコーダンスが保存されることになる．

以上のように，途中で入力を間違えて再入力したりすることがなければ，全部で四回キーボードから入力することにより，後は自動的にcontexproがコンコーダンスを作成してくれるのである．

作成されたファイルは，一般的なエディタやワープロ・ソフトで開くことができるので，結果は通常のワープロで確認することが可能である．

III.3. 利用の可能性

contexproの使用法を見てわかるように，CUI環境による親しみにくさという不利な点はある．とはいえ，若干のチェック・ポイントを確認し，キーボードから入力することによって，コンコーダンスが完成し，その結果は普段使っているワープロで確認することができる．その点からすれば，プログラムを使用することは，非常に簡便な部類に属するいえよう．それ故，複雑なコンピュータ操作が苦手であっても，容易に操作ができるだろう．また，最終的に出力されるコンコーダンスファイルは，すでに見たようにタブで区切られているため，既存のデータベースソフトや，場合によっては表計算ソフトなどを用いて，簡単にデータベース化することができる．例えば，次の例はMacintosh及びWindowsプラットフォームで代表的な表計算ソフトであるMicrosoft社のExcelに取り込んだ例である．

conc

	A	B	C	D
824	chose, , 567	Ainz que la	chose	fust fenie, Li
825	chose, , 615	en et boute	Chose	que j' en
826	chose, , 619	Si vaut a	chose	mainbornir Qu' en
827	chose, , 687	qui si la	chose.	Mes nequedent toute
828	chose, , 688	nequedent toute la	chose	De chief en
829	chose, , 697	ert amendez Par	chose	que nos en
830	ci, , 250	a ce fait?	Ci	a estout donmage
831	ci, , 393	estes bien eesiez	Ci	n' a qui
832	ci, , 425	mort, qu' avez	ci	rece*u Et nostre
833	ci, , 590	con Hersent est	ci	prisei Se je
834	ci, , 635	dame ert de	ci	traite, Je ne
835	cil, , 110	par sorent el	cil	et cil. Se
836	cil, , 110	et cil el	cil.	Se Renert set
837	cil, , 133	velt entendre, Ce	cil	Renart nos senefie
838	cil, , 154	le mont sorpris	Cil	est clemez dolent
839	cil, , 157	outre mesure, Que	cil	qui les grenz

conc

<図7：Excelへの取り込み>

実際に，コンコーダンスを資料として用いることを考えると，データベース

化することで検索を容易にしておいた方が，研究には有効であろう．いずれにせよ，コンコーダンス自体は，contexpro によって，以上のように簡単に作成することができるため，研究の便宜にあわせて，参照しやすい形態にすることも考えるべきである．

結論：contexpro の課題と今後のコンコーダンス作成ソフト

マルチ・プラットフォームに対応しているというメリットはあるものの，contexpro 自体には問題点があることも事実である．それは何よりもまず，Perl というプログラミング言語の選択に起因するものである．インタプリタ型の言語である Perl は，その性質上，実効速度が C 言語などと比較して遅いという欠点がある．また，コンパイルしないためインタプリタをあらかじめインストールしていなければ，プログラムを実行することができない．この問題を解決するための一つの選択肢として，Java の存在に注目したい．但し，残念ながら Java は，マルチプラットフォームに対応してはいるものの，現在はインタプリタ（Perl におけるそれとは若干役割が違う）をもたない環境では動作しない（Web ブラウザの多くには Java をインプリメントすることが可能であるが，ここではむしろスタンド・アローン型のアプレットを考えている）．しかし，将来的には Macintosh や Windows などのパソコンの OS にインプリメントされる計画がある．それ故，マルチプラットフォームで動作するアプレットの開発も，十分可能である．

次に，Conc などの既存のコンコーダンス作成ソフトと比較した場合，機能面での物足りなさが見られることも事実であろう．例えば，contexpro では Conc のように，ある語彙がそのテクスト中に何％含まれているかなどを解析することはできないし，語彙の Index を作成するような機能も備わっていない．

このようにして contexpro の問題点を考えると，理想的なコンコーダンス作成ソフトの在り方とはどのようなものなのかという疑問に行き着く．contexpro に実現されている機能としては，1)マルチプラットフォーム対応，2)キーワードを中心とした任意の語数によるコンテクストが出力されること，3) 散文韻文を問わず使用できること，などが挙げられるだろう．それに加えて，実行速度が速いということも重要だろう．さらに，具体的に技術的な解決策を用意しているわけではないが，次の三つの条件を挙げておきたい．まず第一に，いくつかのテクストのコンコーダンスをリンクすることができる．これは全集などのテクストを扱う場合に考えられることだが，ある作品のコンコーダンスを独立して使うのではなく，既存のコンコーダンスと簡単に比較できることが望ましいだろう．例えば，サルトルの『嘔吐』に出てくる «crabe» という単語が，

『自由への道』の何頁の何行目に出てくるか，という風にである．

次に，文法解析の機能が備わっていることも望まれる．というのも，同じ«le»という単語にしても現代フランス語においては代名詞と定冠詞の二つに分類することができるのであり，現在のコンコーダンス作成ソフトではこの二つは区別されず，同じ語形«le»という一つの項目に分類されてしまうのである．さらに，文法解析機能を備えていれば，動詞の場合などでは，常に活用形を不定形に参照させてやることなども可能になるであろう．例えば«venais»も«vint»も動詞«venir»の活用形であるということを，プログラムが解析してくれることが望ましいのである．

そして，最後に，これまでのコンコーダンス作成ソフトは単語を中心として処理を行っているが，そうした機能に加え，任意の語群を検索語に指定して，その結果をコンコーダンスとして表示させるような機能も必要だろう．例えば，«se souvenir de»という表現は«se»，«souvenir»，«de»と別々にコンコーダンスに入れられるよりも，«se souvenir de»を一つとして処理される方が良いだろう．

こうした文法解析の機能を備えることが，今後のコンコーダンス作成ソフトに何よりも求められる機能だといえよう．

このようにコンコーダンスの作成などに代表されるコンピュータを用いた文学研究は今後も数多く行われ，成果を上げ続けるだろうが，あるテクストを処理する際に避けて通れないのが著作権の問題である．写本研究や草稿研究などでない限り，電子テクストを利用する研究には常に著作権の問題が生ずることを忘れてはならない．言い換えれば，電子テクストを用いた研究は，常に出版社や図書館などと研究者との間に，問題を引き起こす可能性をはらんでいるのである．今後は，こうした著作権問題などにも踏み込みつつ，文学研究におけるコンピュータ利用の在り方を確立していくことも必要であろう．

注

[1] 原野昇・中川正弘・太古隆治・前田弘隆，「中世文学研究におけるコンピュータ利用」，『広島大学フランス文学研究 15』，広島大学フランス文学研究会，1996年，p.67-76.

[2] 久後貴行，「マッキントッシュ上での単語リストとコンコーダンスの作成」，*Travaux de Linguistique et Littérature Médiévale Français* [TLLMF] 第7号，大阪市立大学大学院文学研究科 TLLMF 研究会，1996年，p.1-10.

[3] Perl によるプログラミングについては次を参照のこと：
Randal L. Schwartz 著，『初めての Perl』，近藤嘉雪訳，オライリー・ジャ

バン, 1995.

Larry Wall, Tom Christiansen, Randal L. Schwartz 共著,『プログラミング Perl 改訂版』, 近藤嘉雪訳, オライリー・ジャパン, 1997.

Contexpro : établir la concordance de text avec le langage Perl

SHIGEMI Shinya

Il semble qu'il y ait la forte demande d'utilisation d'ordinateur chez les chercheurs du domaine littéraire. Cet article présente la source du logiciel, contexpro, qui permet aux utilisateurs d'établir automatiquement la concordance de texte, soit en prose, soit en vers.

Le contexpro se caractérise par sa facilité d'usage aussi bien que l'adaptation aux plusieurs systèmes informatiques existants (UNIX, Macintosh, Windows) grâce au langage Perl.

Il est, d'ailleurs, disponible librement sur l'internet à l'URL suivant : http://www.styx.lit.nagoya-u.ac.jp/~shinya/etudes/contexpro.html.