Perl・sortf・秀丸エディタを用いた用法分析の一例

服部　匡

Perl スクリプトによって簡単な KWIC 索引を生成する。それを利用し、Perl・sortf・秀丸エディタを組み合わせて語の用例の仮分類や計数を行っていく方法の一例を紹介する。

内容

はじめに

コーパスを用いて語の用例などを研究する際の参考に、文学作品から「全然」という語の用例のKWIC索引を作成し、用法(共起傾向)を分析する一つの方法を紹介する。

事前の準備

エクスプローラなどで添付 CD のディレクトリ（＝フォルダ）hattori を開くと、JPERL installer.exe という名前のファイルが入っている。これをダブルクリックして実行するとインストール先のドライブ・ディレクトリを聞いてくる。入力欄にあらかじめ C:¥JPERL（＝ドライブ C のすぐ下の JPERL というディレクトリ）と表示されているので、それでよければそのまま[OK]をクリックする。他のドライブ・ディレクトリにしたければ書き換えて[OK]をクリックする。これにより、必要なファイル一式が指定した場所にコピーされる。

インストール先はハードディスクとするのが便利であろうが、USB メモリなどのリムーバブルメディアでも差し支えない。インストールと言ってもファイルを数個コピーするだけで、アンインストール時は当該ディレクトリを消すだけでよい。

以下の説明では C:¥JPERL にインストールしたものとして話を進める。他の場所にインストールした場合は適宜読み替えていただきたい。

次のフリーウェアが利用可能になる。

- sortf ---- 豊島正之氏による、高機能なテキストファイル用ソートプログラムである。
- jperl ---- Larry Wall 氏により開発された言語であり特にテキストファイルの処理に非常に威力を発揮する。ここで用いるのは Perl 5.0035_03 と、Yasushi Saito, Hirofmi Watanabe 両氏による日本語パッチにより作成される jperl(日本語対応 Perl)である。

ファイル名の拡張子を表示する設定にしておくとよい(第2部「ファイル名拡張子」参照)。

§ 1　データの準備

添付 CD の corpora フォルダを丸ごとドライブ C のトップにコピーするのが最も簡単である。以下、その場合を例として説明する。

もちろん、他のデータを検索対象とすることも可能でありその場合は§3以下で"c:¥corpora"のかわりに当該のドライブ:フォルダを指定する。

§ 2　分析開始時の準備

分析を始めるときは、まずコマンドプロンプトを開き、次の操作によってカレントドライブ・カレントディレクトリを設定する（第2部「コマンドプロンプト」を参照)。コマンドプロンプトのデフォルト設定ではカレントドライブは C になっているので、(1)は必要ない。

(1) c: とタイプして[Enter]キーを押す

(2) cd ¥jperl とタイプして[Enter]キーを押す

いったんコマンドプロンプトを閉じた後に分析を行う場合は、以上の操作を行うこと。

§ 3　KWIC 索引の生成

簡単な Perl スクリプト(プログラム)で KWIC 索引を作成してみよう。コマンドプロンプトで次のようにタイプし[Enter]キーを押す。スペースは「半角」のものを使用すること。

jperl　kw.pl　全然　c:¥corpora　10　30

すると、「全然」という語を中心にした KWIC 索引がコマンドプロンプトの画面上に出力される(頭の方は流れてしまう)。

kw.pl というスクリプトは、一般に、次のようにして実行すると指定されたディレクトリ(=フォルダ)以下のすべてのテキストファイル(txt という拡張子を持つもの)に対して検索を実行し、検索文字列の現れを挟んで前後の文脈のついた KWIC 形式ファイルを生成する。[]の部分は省略可能である。生成されたファイルは 1 ファイル名・行番号、2 前文脈、3 検索文字列、4 後文脈の 4 つのフィールドをタブで区切り、秀丸エディタ・EmEditor 等でのタグジャンプ(元のファイルの当該箇所を参照できる機能)が可能な形式をしている(タグジャンプについては第 2 部「エディタを用いた文字列検索」を参照されたい)。

jperl kw.pl 検索文字列　ディレクトリ名　[前文脈の長さ　[後文脈の長さ]]

先ほどのような画面上への出力では検索結果をうまく利用できないので、今度はその結果をファイルに保存する。コマンドプロンプトで次のようにタイプし[Enter]キーを押す。

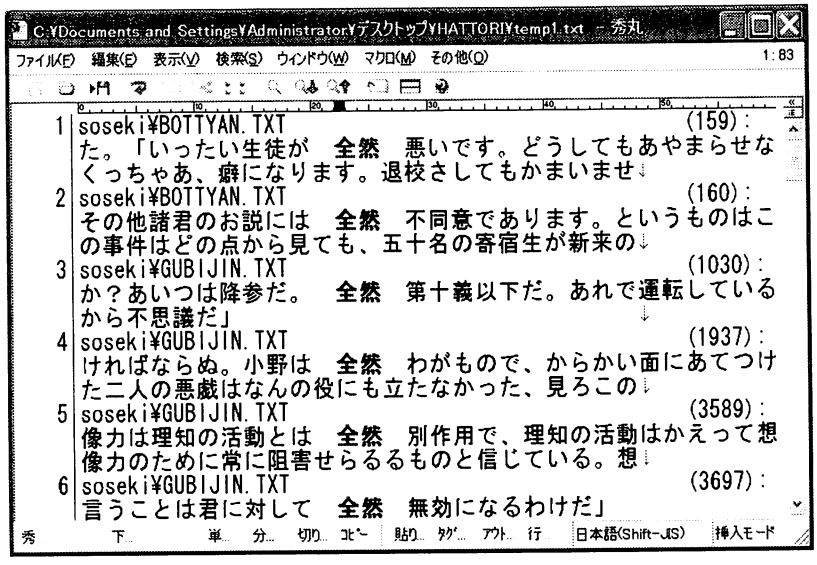
jperl　kw.pl　全然　c:¥corpora　10　30　>　temp1.txt

>の後に適当なファイル名を書けば、その名前で出力結果が保存されるのである。同名のファイルが既にある場合、エラーメッセージや確認は一切ないので注意のこと。

秀丸エディタで temp1.txt を開いてみると、次頁のようなものになっている。[検索]-[検索]で「全然」を検索し反転させた状態である。

スクリプトの詳しい説明は省略するが、本質的な部分は次の数行である。$ptn には検索パターンがが代入されている。①でファイルから 1 行を読み込み、②で検索文字列を検索し、一致した場合、③と④で前後の文脈を取得し、⑤で一致した部分を取得する。その後②に戻り、今一致した部分の次の文字からまた検索を始める。

```
while(<F>){                ←① 1 行の読み込み
        while (/$ptn/g) {←② 検索
                【略:ファイル名、行番号の出力】
                $mae=$`; ←③ 前文脈
                $ato=$'; ←④ 後文脈
                $str=$&; ←⑤ 検索文字列
                【略:所定の長さへの切りそろえ】
                【略:出力】
        }
}
```



なお、検索文字列に正規表現を用いる場合や空白文字・特殊文字を含む場合には、検索文字列全体を""で囲む必要がある。

本稿のKWICは、元のテキストの行の区切りをそのままにして検索するものである。行を連結して検索するものは田野村忠温氏により、行を文単位に分割した上で検索するものは杉本武氏により紹介されている。どの方式がよいかはテキストの性質や研究目的によるので一概に言えないが、元のファイルの構造を一切損なわないという意味ではこの方式が最も基本的と言える。

§4　KWIC索引のソート

こうして得られたリストに対してソート(並べ替え)を行うには、sortfを使用すると便利である。

コマンドプロンプトから次のように実行すると、検索語である「全然」の後続部分によってファイルがソートされ保存される。

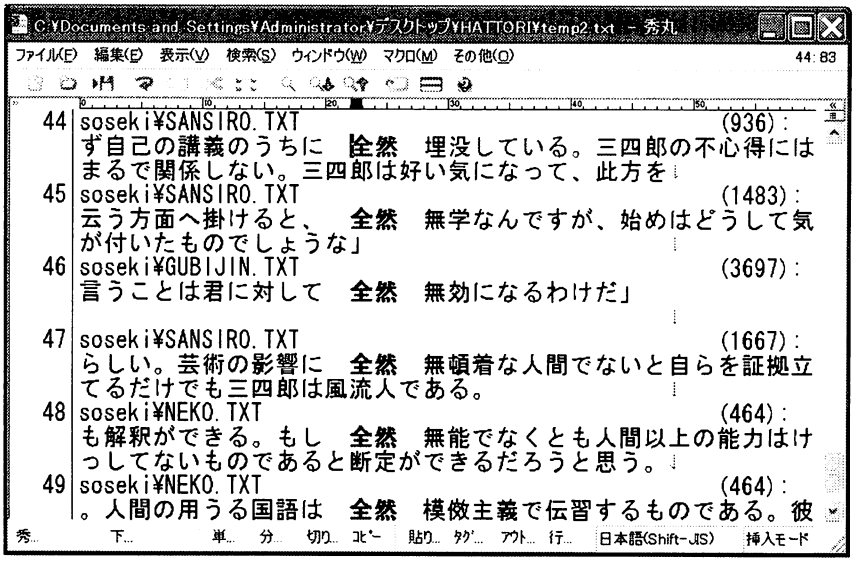
```
sortf -t0x09 +3  temp1.txt > temp2.txt
```

-t0x09は、タブをフィールドの区切りにすることを表わす。+3は、先頭フィールド（＝ファイル名等）を0として3番目のフィールド（＝後文脈）を比較してソートするすることを表わす。

ちなみにsortfでは、ここで示すような用い方以外に「第何フィールドの何文字目から第何フィールドの何文字目」のような範囲の比較によるソートも可能である。また、指定部分が重複する行をまとめたり、その重複数を求めたりすることもでき、検索結果の処理に非常に便利である。

作成されたtemp2.txtを秀丸で読み込むと次頁のようになっている。「全然」の後文脈によって各行が並べ替えられているのが分かる。

なお、ソートの順は文字コードの順になるので、平仮名が片仮名より先になるなど、普通の辞書の配列とは異なる（sortfではより自然な順序でのソートの方法も準備されている）。

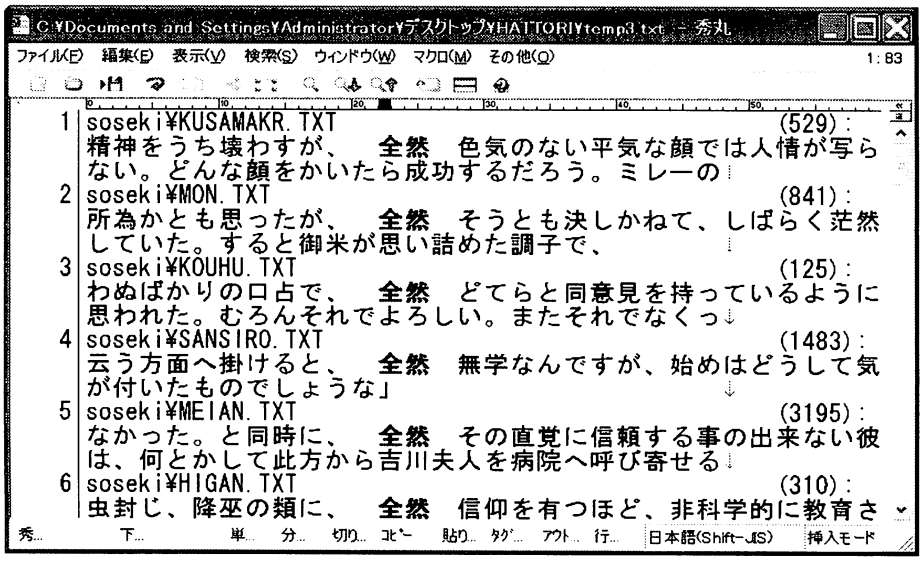


ところで、日本語用例では、特に検索語の先行部分を逆引き辞典のように綴り字と逆順の排列でソートしたいことがある。そのためにはスクリプト rsort.pl が利用できる。

コマンドプロンプトから次のように実行すると、先行部分の逆引きソートが行われる。1 は先頭フィールドを 0 として 1 番目のフィールド(=前文脈)を比較することを表している。

```
jperl  rsort.pl 1 temp1.txt >temp3.txt
```

temp3.txt は、次のような内容になっている。



なお、コマンドプロンプト上で何度も同様のことをタイプするのが面倒なら、バッチファイルを作成するとよい（便宜のため、作成したものを添付する)。例えば、秀丸で新規ファイルに次の 3 行

を書き、kw.batという名前でJPERLフォルダに保存する。%1と%2とは、このバッチを実行するときに与える「引数」を表す。

```
jperl kw.pl %1 %2  10 30 > temp1.txt
sortf -t0x09 +3  temp1.txt > temp2.txt
jperl rsort.pl 1 temp1.txt > temp3.txt
```

そうして、コマンドプロンプトから次のように実行すれば、temp1.txt,temp2,txt,temp3.txtの3つが一度に得られる。

kw 全然 c:¥corpora

§5　KWIC索引からの用例仮分類

KWIC索引をさまざまな仕方でソートしてみるとなかなか興味深い事実が浮かび上がることが少なくない。一種の発見の手段として利用できる。

参考までに、KWIC索引を用いて語の用例の仮分類を進めていく方法の一例を示す。

「全然」の後にどのような表現が続くかによって、用例を仮に分類したい。

前節で temp1.txt,temp2.txt,temp3.txt という3つのファイルを作成したが、上記の目的にはtemp2.txtのみが役立つのでそれを残し、他の2つはとりあえず消去する。また、JPERLフォルダに、他のテキストファイル(.txtの拡張子を持つもの）があるなら、消去するかまたは他のフォルダに移動するかしておく。それらのファイルがフォルダに残っていると、以下の処理が正常に行われないことや、ファイルが破壊されることがある。

§5．1　用例の仕分け

まず、temp2.txtファイルの356行目から374行目にかけて「全然感覚~」「全然関心~」「全然無關係~」を含む一連の行があるので、これらを仮に一類型と見てMUというラベルをつける。356行目の前に「+MU」（プラスの後に続けてラベル名）とだけ書いた一行を挿入し、新しく375行目になった行の後に「－」の1文字だけからなる行を挿入する。同様に例えば「全然不~」の一連の例をFUという名の類型にするなら、307行目から314行目までを「+FU」と「－」で挟む。

注）「＋」は全角でも半角でもよく、「－」はハイフン/マイナス/長音記号/ダッシュのどれでもよい。ラベル名は、ファイル名として使える文字からなり全角計算で16文字以内のものであれば何でもよいが、半角英字に大文字/小文字の区別はできない。

例えば「+MU」と「－」とで挟まれるブロックが同一ファイルの中に何箇所あってもよい。またラベルは何種類であってもよい。

ラベルのマークをつけたならば(必要箇所のすべてにつけなくてもとりあえず思ったところにつければよい)、次のようにスクリプトを実行する。

jperl bunrui.pl

そうすると、例えば「+MU」と「－」ではさまれた各行はMU.txtという名のファイルに移動さ

れるのでファイルの数が増える。安全のために元のファイルは.sav の名前で保存される。

以下、元のファイルと新しいファイルのそれぞれを開いて、必要な箇所にマークをつけては上のスクリプトを実行する(必要ならファイルをソートしなおす)ことを繰り返す。いったん MU として分類した例の中に元のファイルに戻したいものがあれば、その前後を「+temp2」「－」で囲めばよい。もちろん例えば MU からＦＵへの移動もできる。

作業を進める途中で、前後文脈をより長く参照したいならば秀丸エディタのタグジャンプ([F10])機能が利用できる。また、用例のタイプ分けをする過程で秀丸エディタの検索機能が役に立つこともある(第 2 部参照)。用例として不適切な行を見つけたら適宜削除する。

なお、「＋ラベル名 1」で始まるブロックを「－」で閉じないうちにに次の「＋ラベル名 2」を書いた場合は、その行の前で「ラベル名 1」のブロックが終わったものと解釈する。

また、行頭に「＋」だけを書いた場合は、その行は処理上は無意味なものとして無視される。これは次のような場合に一種のメモとして利用できる。「＋ラベル名」から「－」までのブロックが長く 1 画面に収まらないような場合、スクロールしたときに画面がブロック内なのかどうかが分からなくなる。これを防ぐため適当な位置に「＋」だけの行を挿入しておくことができる。

§５．２　ソートのやり直し

作業途中ですべてのファイル(*.txt)をそれぞれ後文脈でソートし直すにはバッチファイルを用いると便利である。次の内容を新規ファイルに書き、sortall.bat という名前で保存する（便宜のため、作成したものを添付する)。[注)]

```
@echo off
if exist *.old goto error
rename　*.txt　*.old
for　%%I　in (*.old)　do　sortf　-t0x09　+3　%%~nI.old > %%~nI.txt
goto end
:error
echo　.old ファイルが既にあるため処理できません。まず消去してください。
:end
echo on
```

コマンドプロンプトから次のように実行すれば、フォルダ内のすべてのテキストファイルに対して一挙にソートが行われる。安全のために元のファイルは.old の名前で保存される。

```
sortall
```

注)　このバッチファイルは Windows の古い版には対応していない。

§5．3　再検索による仕分け

既に得たKWICファイルに対して、新たな検索条件を加えて用例を絞り込みたいことがある。この際、新たな検索条件に合致する用例と合致しない用例とをファイルを分けて出力するのがsibori.plである。例えば先に生成したtemp2.txtを基に、後文脈に関しての再検索を行う場合、次のようにする(コマンドラインで複雑なパターンをタイプするのが難しければ、下記の1行を例えばsibori.batという名のバッチファイルに書いて実行することもできる)。

```
jperl　sibori.pl　"^[^。]*ない"　temp2.txt　3
```

この例では、正規表現を用いて「句点以外の文字が0個以上続いたあとに"ない"が続く」という条件で検索している。その結果、!01temp2.txtというファイルに上記の検索条件に合う後文脈を持つ例のすべてが、また、元のtemp2.txtには上記の検索条件に合わない後文脈を持つ例のみのすべてが出力される。再検索は前回の検索で出力された後文脈の長さの範囲内で行ったのであり、元のコーパスに対して行ったのではない点に注意されたい。

一般的な用法は、次の通りである。

```
jperl　sibori.pl　パターン　ファイル名　フィールド番号　[-c]
```

フィールド番号の1は前文脈、2は元の検索語句、3は後文脈を表す。結果は、"!n ファイル名"、"ファイル名"という二つのファイルに出力される(前者のファイル名が長くなる場合は調整される。nは2桁の数字で既存ファイルと重複しない最も若いもの)。元のファイルはその名前に".sav"を付加した名前で保存される。

フィールド番号を0と指定した場合は、「前文脈ー元の検索語句ー後文脈」を連結したものに対して検索を行う。

注)　例えば、"全然"を検索した結果のファイルに対してフィールド番号を0と指定して"は全然ない"を再検索する場合、一致部分は前回の検索文字列の"全然"を含むものとは限らない—前回の前文脈か後文脈にたまたま現れた"全然"を含む一致もありうる(この例ではまれなことであるが)—点に注意されたい。

フィールド番号の後に-c と指定したときは、条件に合致する方("!n ファイル名")の出力は元の区切りによるのではなく新たな検索語句を中心にした区切りに整形し直される。

§5．4　用例の計数

このようにして、仮分類の妥当性を検討しながら、少しずつ分析を進めていく。ある程度進んだならば、次のようにタイプすると、類別の用例数と、その比率とが表示される(結果を保存するには「＞kekka.csv」などを後につける)。

```
jperl count.pl
```

なお、bunrui.plとcount.plという二つのスクリプトはKWICファイルに限らずいかなるテキストファイルにも適用可能であり、行ごとに情報を分類する様々な作業に利用できる。

おわりに

コーパスを用いて語の用例などを研究する方法の一例を示したが、ご参考になれば幸いである。紹介したスクリプトは、筆者が実際に研究に利用しているものの一部である。

KWIC 索引に関しては、より高機能なプログラムが第 1 部で田野村忠温氏により提供されている。また、第 2 部で紹介するようなソフトウェアの利用も可能である。

しかしながら、既存ソフトを利用するにしても、それによって得られたデータを自由自在に活用するには、AWK や Perl のような言語の知識が有効である(AWK については第 1 部に分かり易い入門編がある)。コーパスを用いた研究を本格的に行おうとする場合は、できればそれらを習得されることをお勧めする。