第6章 スペイン語教科書の基本語例文の抽出法

第6章
スペイン語教科書の基本語例文の抽出法

Capítulo 6
Método para la extracción de ejemplos del vocabulario básico en los textos de español

宮　本　正　美
Masami Miyamoto

1. はじめに

2004年から本格的な西和辞典、和西辞典が電子辞書に搭載され、インターネットの普及も相まって、学習者がスペイン語のデジタルテキストに日常的に触れる機会が増えてきました。今のところ、教室で利用されるスペイン語教科書（の大半）は紙の教科書ですが、将来的にはデジタルテキストも増えてくるものと思われます。テキストがデジタル化されれば、紙のテキストにはないいくつものメリットをユーザである学習者は享受することができます。その一つは、テキストを加工して、各自の目的に沿った語彙集や例文集が簡単に作成できるということです。

入門・初級の学習者にとって、基本語彙をその例文とともに学ぶことは極めて重要です。学習する教科書から必要な基本語彙の例文を集めて、自分なりの学習用例文集を作ることはその目的にかなうでしょう。自分の目的にぴったりかなったものは、やはり、自分で工夫しながら作るのが理想的です。市販の電子辞書に搭載された語学辞書と同じ質のものを自作することは、初中級の学習者には不可能かも知れませんが、自分なりの語彙集、例文集なら、一人でゼロ

から始めても十分に可能です。

デジタルテキストを処理するには、本格的なプログラミング言語は必ずしも必要ではありません。AWK や Perl、あるいはシェルスクリプトといったスクリプト言語のどれか一つを少し学べば、語彙集、例文集といった作品を作るテキスト処理は十分に可能です。

ここでは、スペイン語教科書を対象テキストとして、AWK, Perl, シェルスクリプトの簡単な機能を使って、その「基本語例文を抽出する手法」について、スクリプト例をいくつか挙げながら、述べてみます。これを参考に、興味ある学習者が自家製の語彙集、例文集、さらには辞書を作ってもらえればと思うからです。

サンプルのデジタルテキストとしては、3点のスペイン語教科書から数十頁ずつ入力して作成したファイルを利用しました。[1]

2. 基本語例文の抽出法

2.1. 動詞以外の基本語例文を集める

変化形の多い動詞と動詞以外の単語は分けて、それぞれの基本語例文を集めることにします。まず、動詞以外の単語の基本例文を集めてみます。

2.1.1. 単語の頻度を数える

まず、基本語を選定するために、テキストの各単語の頻度数を数えます。ここでは、高頻度の単語を基本語とみなすことにします。

対象テキストに現れる単語の頻度数の数え方は、1行目の先頭から各単語を1つずつ記憶し、同じ単語が現れたら、その数を増やしていくという単純なものです。[2] 但し、文中の単語には、, . ; : ? ! ( ) - などさまざまな記号の付くこ

---

[1] 入力作業は、ゼミの人たちと協力して行いました。協力してくださった次のみなさんに、心から感謝します：佐々木笑梨, 阿南裕子, 岩吹由記, 桑本千幸, 小山圭, 山縣絵莉子。

[2] 複合語やコロケーションに代表される連語を抽出する場合には、宮本(1998)のような、それなりの工夫が必要になります。

とがありますから、それらの記号は削除して単語を取り出し、大文字(で始まる)単語は、小文字化して単語の頻度を数えるようにします。対象の教科書テキストがファイル int_pap_sue.txt だとすると、AWK スクリプト u80620b (**Appendix 1** 参照) を利用して、まず、単語の頻度数を数えます。その結果を sort コマンドで降順に並べて、単語の頻度順リスト (xx25zf) を作ります[3]。

```
$awk u80620b int_pap_sue.txt | sort +0nr > xx25zf
```

こうして、作成された単語の頻度数リスト(xx25zf) は次のようなものです:

```
1845 de
1442 que
...中略...
561 es
...中略...
91 porque
89 bueno
88 esta
88 son
87 nos
84 an~os
84 usted
81 mucho
78 eso
78 tiene
...以下略...
```

2.1.2. 動詞以外の単語のリストを作る

動詞以外の単語は、頻度数リスト(xx25zf)から、頻度順に目視で、必要な語を選んでいきます。例えば:

```
91 porque
```

[3] 以下の作業はすべて、 Mac OSX のターミナル、あるいは Linux の Kterm 上で行っています。Windows のコマンド・プロンプト (あるいは MS-DOS プロンプト) でも、Cygwin などのソフトを利用すれば、同様に行えます。

```
89 bueno
84 an~os
84 usted
81  mucho
...以下略...
```

その際に、複数形は、単数形に戻し、形容詞に女性形がある場合、女性形を書き加えておくことにします。このようにして例文を集めたい動詞以外の単語リスト(xx25nv)を作ります[4]。

```
porque
bueno # buena
an~o
usted
mucho #   mucha
...以下略...
```

2.1.3. 動詞以外の基本語例文を出力する

対象の教科書テキスト（int_pap_sue.txt）から、動詞以外の単語リスト(xx25nv)を利用して、例文を集めるには、シェルスクリプト y40717.sh を使用します。シェルに Bash を使うとすると、コマンド行は次のようになります：

```
$bash y40717.sh  -a10  -b30    -c20 -d80  -sy  -hn xx25nv
int_pap_sue.txt
```

オプション -a10 は集める例文数が10であることを、-b30 は頻度数 30 以上の単語の例文を集めることを意味します。また、-c と -d は集めた例文を、その単語を中心に縦揃えして表示する際に、-c20 は 単語の左右幅20語、-d80 は80文字の枠内に表示することを意味します。-sy はxx25nv に # 記号が用い

[4] この作業がめんどうで、高頻度の各語形(de, que, ..., es, ..., porque, bueno, esta, son, an~os, ...)の例文を集めるのでよければ、2.1.1. の 頻度数リスト(xx25zf) を、頻度数の部分を除いて、(動詞の活用形も含みますが) 動詞以外の単語リスト(xx25nv)として利用しても構いません。

られていることを表し、-hn は品詞データが含まれていないことを意味しています。各単語の例文はその単語名のファイルとして、ディレクトリ temp25 内に作成されます。

シェルスクリプト y40717.sh[5] が実質的に行っている処理は次のようなものです：

a. 動詞以外の単語リスト(xx25nv) の各行を「小文字男性単数形　大文字男性単数形　小文字男性複数形　大文字男性複数形　(# 小文字女性単数形　大文字女性単数形　小文字女性複数形　大文字女性複数形)」、例えば、

bueno Bueno buenos Buenos # buena Buena buenas Buenas
an~o An~o an~os An~os

に変換したリストxx67_lusp を作る。

b. xx67_lusp を利用して、対象テキスト中の各変化形の頻度を数え、「合計頻度数　各語形とその頻度数」のリスト xx67_freq を作る。

c. xx67_freq を利用して、xx67_lusp から指定頻度数以下の単語行を削除した単語変化形リスト xx67_luspM を作る。

d. 対象テキストから、xx67_luspM の各行の変化形を検索スクリプト y40717 によって、指定された単語数の左右幅で、指定された例文数を打ち出して例文ファイル xx67f を作る。

e. xx67f から単語名と例文部分を取り出し、単語名のファイルに、y20315

[5] y40717.sh は後述のシェルスクリプト y50926.sh (**Appendix 6 参照**) とほぼ同じスタイルですが、本文の a. の過程で使用したCV・綴り字分節スクリプト、形容詞・名詞の複数形化スクリプト、また、d. の検索スクリプト y40717 などかなり長いスクリプトを組み込んでいるので、ここでは、紙幅の都合上 Appendix に掲載することを断念しました。スペイン語単語の CV 分節法は、Miyamoto (1997 : 337-339) が参考になります。また、y40717 は、単語列検索スクリプト y30104(Miyamoto (2005:60-63) 参照) がベースになっています。検索語をファイルのリスト (xx25nv) から与えるように変更し、また、打ち出す例文数をコマンド行から設定する機能を加えたスクリプトです。

(**Appendix 5** 参照)によって、指定された文字数の左右幅の枠内で出力する。

参考に、出力された bueno とaño の例文を10ずつ挙げます。ここでは、紙面の都合で、単語の左右幅を短くして 4 語、30文字の枠内で表示しますが、当然、ユーザの望む任意の幅で表示することができます[6]。

```
in:            4. El turista en [Buenos] Aires
in:               Decide irse a [Buenos] Aires, donde es fama
in:    superficie del pai's hay [buenas] condiciones para el ganado
in:       pasean en la noche en [Buenos] Aires hari'an las delicias
in:    atenta a sus gustos, tan [buena] ama de casa, tan
in:                     !Pero [bueno,] si falta una maleta...
in:     espan~ol es muy fa'cil. [Bueno,] para los italianos es
in:                   Ah,!que' [bueno!] Me alegro.
in:    encontre' un trabajo muy [bueno.]
in:                        4. [Bueno,] adio's. Hasta pronto.

in:        que ha pasado muchos [an~os] en Estados Unidos tratando
in:                   A los 52 [an~os] sigo pensando lo mismo
in:                   A los 52 [an~os] sigo pensando lo mismo
in:                   A los 52 [an~os,] me planto en medio
in:       me engan~aron a los 7 [an~os,] a los 17 y
in:                   A los 52 [an~os,] escribo y no escarmiento
in:                   A los 52 [an~os,] Ma'laga. Y escribo como
in:                   A los 52 [an~os,] ni tengo bicicleta, ni
in:                   A los 52 [an~os,] los caramelos son de
in:                   A los 52 [an~os,] escucho el agua de
```

2.2. 動詞の基本語例文を集める

テキスト中で、動詞はさまざまな活用形となって現れます。そこで、高頻度動詞の例文を集めるにあたっては、まず、動詞の活用形の頻度数をそれぞれの不定詞形に集約して、動詞の頻度数としました。対象テキスト では、es が 559 回、son 86、...、ser 51、...、ha sido 11、... 合計 ser 950 回です。これで初めて、テキストでは ser の方が estar よりも頻度が高いかどうか分かります。そして、この作業を行うためには、すべての（あるいは、多くの）

[6] 左端の in: は例文を含む元のテキスト名（*Intercambio, 2*）の略号です。

動詞のすべての活用形リストを用意しておく必要があります。

2.2.1. 不定詞形に集約して動詞の頻度を数える

ここでは、3,108個の再帰動詞と（かなり重複しますが、）7,785個の非再帰動詞のすべての活用形、すなわち、約36万語形の再帰動詞のリスト(40730s)、と約89万語形の非再帰動詞のリスト(40401k)を利用して[7]、対象テキスト の活用形を不定詞に戻しながら動詞の頻度を数えることにします。この手続きを、y50924（**Appendix 2** 参照）で次のように実行します：

```
$awk y50924 50919func 40730s 40401k int_pap_sue.txt | sort
+0nr > xx25zvf
```

50919func は活用形と同形の他品詞語から成るリストです。同形異義語のチェックに使用します。40730s は再帰動詞の活用形リストで、以下の［表１］のような形式をしています。40401kは非再帰動詞の活用形リストで、int_pap_sue.txt は今回使用している対象テキストです。この処理で得られる対象テキストの動詞頻度リスト（xx25zvf）は次のようなものです：

```
950 ser
313 estar
287 ir
286 tener
214 poder
201 decir
...中略...
1 venderse
1 venerar
1 verificar
1 vivificar
1 vomitar
1 votar
```

[7] それぞれの作成法については、宮本(2005b)を参照。

再帰動詞の頻度も、あわせて、見ておきます。対象テキスト の動詞頻度リスト(xx25zvf)から se で終わるものを取り出します：

```
41 irse
33 decirse
23 encontrarse
22 llamarse
20 verse
15 volverse
...中略...
1 ten~irse
1 terminarse
1 tirarse
1 traerse
1 usarse
1 venderse
```

私たちが(再帰)動詞を数え上げた方法は、多くの動詞のすべての活用形リストとテキストを照合するというものです。例えば、comerse を数え上げるには、comerse の以下のような、すべての活用形を利用しています：

[表１]

```
comerse(comerse.0.inf.0)
comie'ndose(comerse.0.ger.0)
comido(comerse.0.pp.0)
me como(comerse.i.p.1s)
te comes(comerse.i.p.2s)
se come(comerse.i.p.3s)
nos comemos(comerse.i.p.1p)
os come'is(comerse.i.p.2p)
se comen(comerse.i.p.3p)
me comi'(comerse.i.ind.1s)
...中略...
nos habri'amos comido(comerse.i.cp.1p)
os habri'ais comido(comerse.i.cp.2p)
se habri'an comido(comerse.i.cp.3p)
haberse comido(comerse.0.infp.0)
habie'ndose comido(comerse.0.gerp.0)
co'mete(comerse.m.p.2s)
co'mase(comerse.s.p.3s)
```

coma'monos(comerse.s.p.1p)
comeos(comerse.m.p.2p)
co'manse(comerse.s.p.3p)

対象テキスト の中に、me comí（あるいは --<Me comí など）が出てくると、参照する「再帰動詞の活用形リスト」に me comí（comerse.i.ind.1s）が挙っているので、comerse が１つというように数えていきます。「再帰動詞の活用形リスト」には、3,108 個の再帰動詞のすべての活用形が挙っていますから、初・中級のスペイン語テキストを対象とする限り、テキストの再帰動詞を見逃すことはないはずです。但し、このリストには、me lo comí, me la comí, me los comí,... のような目的人称代名詞との組み合わせは含まれていませんから、当然、このような場合を見逃してしまいます[8]。数え上げがまだ完璧でないとはいえ、目視で数えてもいくらかミスをするでしょうし、まして大量のテキストを目視で数えることが不可能である以上、コンピュータで頻度を数えることは十分に意味のあることです。

2.2.2. 動詞の基本語例文を出力する

対象テキスト の動詞頻度リスト（xx25zvf）を作ったので、これを利用して、高頻度動詞の例文を集めてみます。 まず、その前処理として、例文を集めたい頻度順位Ｎ番までの動詞のすべての活用形（xx26v_c）を集めておきます。ここでは、ser から頻度順位50番の preguntar までを、次の y50925 (**Appendix 3** 参照)によって集めることにします：

$awk y50925 50 xx25zvf 40401k 40730s > xx25v_c

[8] 目的人称代名詞との組み合わせも数え上げるにはどうしたらよいでしょうか。活用語形と目的人称代名詞の組み合わせを、すべてデータ化することはもちろん可能ですが、さすがに、データが大きくなり過ぎます。テキストを走査して動詞を探す際に、前に現れる目的人称代名詞（群）と後接する目的人称代名詞（群）をチェックしながら、活用語形リストと照合することになるでしょうか。解決しなければならない課題の一つです。

この50番目までの動詞には、 irse, decirse, encontrarse の 3 つの再帰動詞も含まれます。そして、この50個の動詞のすべての活用形リスト(xx25v_c)を利用して、対象テキスト中のそれぞれの動詞の例文を、その動詞別のファイル名で集めるには、次のシェルスクリプト y50926.sh（**Appendix 6** 参照）を利用します：

```
$bash y50926.sh -a10 -b3 -c27 50919func xx25v_c 40730s int_pap_
sue.txt
```

-a10 は取り出す例文数10 を指定するオプションです。以下、-b3 は動詞の前後幅語数を 3 語に指定するオプション、 -c27 は動詞の前後幅文字数を27文字に指定するオプションです。50919func は同形異義語をチェックする同形異品詞語リスト、xx25v_c は例文を抽出したい動詞の活用形リスト、40730s は再帰動詞の全活用形リスト、int_pap_sue.txt は対象テキストです。各動詞の例文はその不定詞名のファイルとして、ディレクトリ temp17 内に作成されます。

このシェルスクリプトが行っている主な処理は、既に見たy40717.sh の d. と e. に相当する以下の二つです：

i. 対象テキストから、y50926（**Appendix 4** 参照)が、xx25v_c を利用して、指定された単語数の左右幅で、指定された例文数を打ち出して例文ファイル xx67f を作ります。但し、同形異品詞リストで同形異義語の簡単なチェックと、再帰動詞の全活用形リストで、再帰と非再帰のチェックを行いながら、例文を抽出します。

ii. xx67f から不定詞名と例文部分を取り出し、不定詞名のファイルに、y20315（**Appendix 5** 参照)によって、指定された文字数の左右幅の枠内で求める動詞の基本語例文を出力します。

y50924 や y50926 では、動詞の活用形をすべて記憶してから、記憶した活用形と照合しながらテキストを走査して、求める動詞の頻度を数えたり、その動詞を含む例文を抽出するという方法を採っています。しかし、宮本・高垣(2004:95) で行ったように、まずテキストに動詞の活用形データをすべてタグ付けし、タグの付いたテキストから、頻度を数えたり、例文を抽出する方法もあります。

参考に、出力されたtener と decirse のファイルの中身、つまり、例文10ずつを挙げます[9]。実際には、動詞の左右はもっと多くの語数、文字数の幅で表示する方が実用的です：

```
in:                ?Con que' [has tenido] ma's problemas?
in:             el mundo que [tiene] esa lengua y
in: maravilloso pero tambie'n [tiene] sus problemas. A
in:               a la calle [tiene] que decidir si
in:            muy linda que [teni'a] mesas en la
in:           actividad va a [tener] un papel económico
in:             bonita y que [teni'a] ma's en la
in:            El pobre mozo [teni'a] una chaqueta que
in:            muy lindo que [teni'a] mesas en la
in:            saco pareci'a [tener] una cobija: sudaba

in:  la ganaderi'a, puede [decirse] que en el
in:         del 5%. Puede [decirse,] aun asi', que
in: cifras exactas, puede [decirse] que la capital,
in: no entiendo... ?Co'mo [se dice...?] No entiendo la
in:                   No [me diga.] !Que' pena!
in:                ?Que' [se dice] en Chile el
in:          En Colombia [se dice:] "Feliz cumplean~os", simplemente.
in:         En Argentina [se dice:] "Feliz cumplean~os", tambie'n.
in:           En Me'xico [se dice:] "Muchas felicidades".
in:              ?Y que' [se dice] el di'a de
```

[9] decirse は紙面の都合で、単語の左幅を23文字にして表示しています。

3. むすび

頻度数を手がかりに、基本語の例文を集める手法について述べてきました。変化形の多い動詞と動詞以外の単語に分けて、動詞の場合は活用形を不定詞形に集約することで頻度数の確定精度を高め、動詞以外の場合は手作業の過程を含めることで精度を高めました。今回はスペイン語の教科書を対象テキストとしましたが、この手法で、さらに大きなテキスト、例えば、1ギガバイト・レベルのテキストを対象として、辞書編集用の基本語例文集などを作成することもできます。[10] 末尾のAppendix に挙げたように、AWK, Perl, シェルなどのスクリプト[11]を組み合わせることで、比較的簡単に、基本語例文を抽出できることを示しました。

対象テキスト：

González Hermoso, Alfredo, y Romero Dueñas, Carlos: *Español lengua extranjera, Curso de puesta a punto en español*, Edelsa, 1998.

Miquel, Lourdes, y Sans, Neus: *Intercambio, 2,* Difusión, 1990.

Torres Álvarez, María Jesús, et al.: *Sueña 4, libro de alumno, nivel superior,* Anaya, 2001.

参考文献：

エイホ, A.V., B.W. カーニハン, P.J. ワインバーガー(1989): 『プログラミング言語 AWK』, トッパン.

ブリン, ブルース(2003): 『入門 UNIX シェルプログラミング』, (改訂第2版), ソフトバンク.

クリスチャンセン, トム, ネイザン トーキン(2004): 『Perl クックブック』, 第2版, 2vols., オライリー・ジャパン.

Hammond, Michael(2003): *Programming for Linguists, Perl for Language Researchers,* Blackwell.

Kawaguchi, Yuji, Susumu Zaima, Toshihiro Takagaki, Kohji Shibano &

---

[10] 辞書編集用の「基本語彙3000語の例文抽出」については、宮本(2005a) を参照。

[11] AWK の参考書としては、そのバイブルとも言えるエイホ, カーニハン & ワインバーガー(1989) とRobbins (2001)が秀れています。語学研究の実例集としては上田(1998)があります。Perl の優れた書籍は、クリスチャンセン & トーキン(2004)、ウオール, クリスチャンセン & シュワルツ (1997) など多数挙げられますが、語学用のものとしては、Hammond (2003)や、日本語処理の佐野(2003)が参考になります。シェルスクリプトについても、ブリン(2003) や Powers, Peek, O'Reilly & Loukides (2003) などいくつも参考書が挙げられます。

Mayumi Usami(2005): *Linguistic Informatics —State of the Art and the Future: The First International Conference on Linguistic Informatics,* John Benjamins.
Miyamoto, Masami(1997): "Sobre la estructura del léxico en *Cien años de soledad",* Torre y García Barrientos(1997), 329-340.
———— (2005): "A Formal Analysis of Spanish Adjective Position", Kawaguchi et al. (2005), pp.46-63.
宮本正美 (1998):「El Mundo 紙における連語の自動抽出」,『神戸外大論叢』, 49:2, pp.3-27.
———— (2004) :「電子辞書のための動詞活用形の展開とスペイン語不規則動詞の分類」、日本イスパニヤ学会第50回　学会創立50周年記念大会、南山大学.
———— (2005a) :「スペイン語コーパス言語学入門」, 東京外国語大学2005年度COE集中講義。
———— (2005b):「電子辞書のためのスペイン語動詞活用形の展開」,『神戸外大論叢』, 56:5.
宮本正美・高垣敏博(2004):「スペイン語コーパスによる受動文の検索」,『語学研究所論集』, 東京外国語大学, pp.87-109.
Powers, Shelley, Jerry Peek, Tim O'Reilly, & Mike Loukides(2003):『UNIXパワーツール』(第3版), オライリー・ジャパン.
Robbins, Arnold(2001): *Effective awk Programming*, 3rd Edition, O'Reilly.
佐野　洋 (2003):『Windows PC による日本語研究法』, 共立出版.
Torre, Estéban, y José Luis García Barrientos(1997): *Comentarios de textos literarios hispánicos,* Editorial Síntesis, Madrid.
上田博人(1998):『パソコンによる外国語研究(II) 文字データの処理』, くろしお出版
ウオール, ラリー, トム クリスチャンセン, ランダル L. シュワルツ(1997):『プログラミング Perl』(改訂版), オライリー・ジャパン.

Appendix 1:　u80620b :

```
#!/bin/gawk -f
#       名称 : u80620b
#       機能 : 対象テキスト中の単語の頻度を数える.
#       書式 :  u80620b 対象テキスト
#       書式例 : u80620b int_pap_sue.txt
################################################
BEGIN{FS="[^0-9a-zA-Z¥'¥"~^]"  # フィールドセパレータの設定
}
{for(i=1; i<=NF; i++)
        x[tolower($i)]++          # 各単語を小文字化して頻度数を記憶する
}
END {for(w in x) {
            print x[w], w         # 各単語を頻度数と共に打ち出す
            }
 }
```

Appendix 2: y50924 :

```
#!/bin/gawk -f
#       名称：y50924
#       機能：テキスト中の動詞の頻度を、不定詞形に戻して数える
#       書式：y50924 同形異品詞リスト 再帰動詞活用形リスト
#             非再帰動詞活用形リスト  対象テキスト
#       書式例：y50924 50919func 40703s 40401k int_pap_sue.txt
#       解説：40730s と 40401k の活用形リストを利用して、テキスト中の動詞の頻度を
#             数える．まず、再帰動詞がどうかチェック、そうでなければ、非再帰動詞
#             かどうかチェックする．
#       参考：同形異品詞語があるので，簡単にチェックする．
###############################################################################
BEGIN{
file1=ARGV[1];ARGV[1]=""
file2=ARGV[2];ARGV[2]=""
file3=ARGV[3];ARGV[3]=""
while(getline<file1>0) {                    # 活用形と同形の主な他品詞語を記憶する．
       th[$1]="@"
       }
while(getline<file2>0){                     # 再帰動詞の活用形を記憶する．
       if((NF==1) && ($1 !~ "¥.pp¥.")){     # 過去分詞は除く．
              split($1, y, "[()]")
              split(y[2], z, ".")
              x[y[1]]=z[1]
              }
       if(NF==2){
              split($2, y, "[()]")
              split(y[2], z, ".")
              x[$1 " " y[1]]=z[1]
              }
       if(NF==3){
              split($3, y, "[()]")
              split(y[2], z, ".")
              x[$1 " " $2 " " y[1]]=z[1]
              }
       }
while(getline<file3>0){                     # 非再帰動詞の活用形を記憶する．
       if((NF==1) && ($1 !~ "¥.pp¥.")){     # 過去分詞は除く．
              split($1, b, "[()]")
              split(b[2], c, ".")
              a[b[1]]=c[1]
              }
       if(NF==2){
              split($2, b, "[()]")
              split(b[2], c, ".")
              a[$1 " " b[1]]=c[1]
              }
       }
}
{for(j=1; j<=NF; j++){
       Sj1=tolower($j)
       gsub("[^a-zA-Z'¥"~]", "", Sj1)
       Sj2=tolower($(j+1))
```

```
        gsub("[^a-zA-Z'¥"~]", "", Sj2)
        Sj3=tolower($(j+2))
        gsub("[^a-zA-Z'¥"~]", "", Sj3)
        if(x[Sj1] !=""){
                p[x[Sj1]]++
                }                       # Sj1 という活用形，例えば、atre'vete が
                                        # 再帰動詞にあれば,それを不定詞形で数える.
        else if(x[Sj1 " " Sj2 " " Sj3] !=""){
                p[x[Sj1 " " Sj2 " " Sj3]]++
                j=j+2                   # あれば、2つ余分に進める.
                }
        else if(x[Sj1 " " Sj2] !=""){
                                        #  これを先にすると,Se ha;  te has;
                                        #  os habe'is;...だけで取り出してしまう.

                p[x[Sj1 " " Sj2]]++
                j=j+1                   # あれば、１つ余分に進める.
                }
                                        # ここまで、再帰動詞かどうかチェックした.
                                        # 以降は、非再帰動詞かどうかチェックする：
        else if(a[Sj1 " " Sj2] !=""){
                p[a[Sj1 " " Sj2]]++
                j=j+1                   # あれば、１つ余分に進める.
                }
        else if((a[Sj1] !="") && (th[Sj1] =="")){
                                        # これを先にすると，has; He; など単独で数えてしまう.
                                        # 非再帰動詞の単純活用形には，una，tema，casa,...など
                                        # 多数の同形異品詞語があるので，簡単にチェックする.
                p[a[Sj1]]++             # Sj という活用形が非再帰動詞に
                }                       # あれば，それを不定詞形で数える.
        }
}
END{
        for(w in p){print p[w], w
                }
        }
```

Appendix 3:y50925：

```
#!/bin/gawk -f
#       名称： y50925
#       機能： Inf-List の N番目までの動詞のすべての活用形を
#              活用形(文法データ) の形式で取り出す.
#       解説： y50924 の出力ファイルInf-List(対象テキスト中の
#              動詞を不定詞に集約して作った頻度数リスト)：
#              950  ser
#              313  estar
#              ...中略...
#               41  irse
#              の形式の指定頻度順位 N までの(再帰)動詞の活用形を、
#              (再帰)動詞活用形リストConj-Listから取り出して
#              abacoro(abacorar.i.p.1s)
#              he abacorado(abacorar.i.per.1s)
#              あるいは、
```

```
#             me como(comerse.1.p.1s)
#             me he comido(comerse.1.per.1s)
#             co'mete(comerse.m.p.2s)        の形式で打ち出す.
#      書式 : y50925        N  Inf-List Conj-List  >出力file
#      書式例 : y50925      50  25zvf  40401k  40730s > xx25v_c
#################################################################
BEGIN{
num=ARGV[1]; ARGV[1]=""
file1=ARGV[2]; ARGV[2]=""
n=0
while(getline<file1>0){
            n++
            if(n<=num){                          # 頻度順位 num 番まで
                  x[$2]=$1
            }
      }
}
{if(NF==1){
      split($1, y, "[()]")
      split(y[2], z, ".")
      if(z[1] !~ "se$"){                         # 非再帰動詞の場合 :
            if(x[z[1]] != 0){                    # !~ ""  は不可.
                  print  y[1] "(" y[2] ")"
                  }
            }
      else{                                      # 再帰動詞の場合 :
            if(x[z[1]] != 0){
                  print  y[1] "(" y[2] ")"
                  }
            }
      }
      if(NF==2){
            split($2, y, "[()]")
            split(y[2], z, ".")
      if(z[1] !~ "se$"){
            if(x[z[1]] != 0){
                  print  $1, y[1] "(" y[2] ")"
                  }
            }
      else{
            if(x[z[1]] != 0){
                  print  $1, y[1] "(" y[2] ")"
                  }
            }
      }
if(NF==3){                                       # 再帰動詞の完了形の場合 :
      split($3, y, "[()]")
      split(y[2], z, ".")
      if(x[z[1]] != 0){
            print $1, $2, y[1] "(" y[2] ")"
            }
      }
}
```

Appendix 4: y50926 :

```
#!/bin/gawk -f
#       名称 : y50926
#       機能 :  y50925 の出力ファイル(xx26v_c)にある(再帰)動詞の
#               例文を指定数、対象テキストから出力する.
#       書式 :  y50926  出力例文数 前幅語数 後幅語数
#               同形異品詞語リスト  検索(再帰)動詞活用形リスト
#               再帰動詞活用形リスト 対象テキスト >出力ファイル
#       書式例 : y50926       5  20  20  50919func  xx25v_c 40730s
#               int_pap_sue.txt > xx26ve
#       解説 :  例文を集めたい(再帰)動詞の活用形リスト(xx26v_c) を
#               利用して、 対象テキスト中の(再帰)動詞の例文を打ち出す.
#               同形異品詞リスト(50919func)と再帰動詞活用形リスト
#               (40703s) は不適切な語(列)をチェックするのに利用する.
##########################################################################
BEGIN{
Num=ARGV[1]; ARGV[1]=""
Wide1=ARGV[2]; ARGV[2]=""
Wide2=ARGV[3]; ARGV[3]=""
file1=ARGV[4];ARGV[4]=""
file2=ARGV[5];ARGV[5]=""
file3=ARGV[6];ARGV[6]=""
while(getline<file1>0){             #  動詞と同形の主な他品詞語
        th[$1]="@"                  #  を記憶する.
        }
while(getline<file2>0){             # 例文を集める動詞の活用形を記憶する.
        if((NF==1) && ($1 !~ "¥.pp¥.")){    # 非再帰動詞の単純形か、
                                    # 再帰動詞の一部(命令形など). 過去分詞は除く.
                split($1, y, "[()]")
                split(y[2], z, ".")
                x[y[1]]=z[1]
                }
        if(NF==2){                  # 非再帰動詞の完了形か、再帰動詞の単純形.
                split($2, y, "[()]")
                split(y[2], z, ".")
                x[$1 " " y[1]]=z[1]
                }
        if(NF==3){                  # 再帰動詞の完了形のみ
                split($3, y, "[()]")
                split(y[2], z, ".")
                x[$1 " " $2 " " y[1]]=z[1]
                }
        }
while(getline<file3>0){             #  再帰動詞の活用形を記憶する.
                                    #  チェックに利用する.
        if((NF==1) && ($1 !~ "¥.pp¥.")){
                split($1, b, "[()]")
                split(b[2], c, ".")
                a[b[1]]=c[1]
                }
        if(NF==2) {
                split($2, b, "[()]")
                split(b[2], c, ".")
```

```
                a[$1 " " b[1]]=c[1]
                }
        if(NF==3){
                split($3, y, "[()]")
                split(y[2], z, ".")
                a[$1 " " $2 " " y[1]]=z[1]
                }
        }
}
{for(j=1; j<=NF; j++){
        Sj0=tolower($(j-1))
        gsub("[^a-zA-Z'¥"~]", "", Sj0)
        Sj1=tolower($j)
        gsub("[^a-zA-Z'¥"~]", "", Sj1)
        Sj2=tolower($(j+1))
        gsub("[^a-zA-Z'¥"~]", "", Sj2)
        Sj3=tolower($(j+2))
        gsub("[^a-zA-Z'¥"~]", "", Sj3)
        if(x[Sj1 " " Sj2 " " Sj3] !=""){ # 再帰動詞の完了形の場合
                seq=Sj1 " " Sj2 " " Sj3
                c[x[seq]]++             # x[seq] の語数をカウントする
                if(c[x[seq]] <= Num) {
                        if((j-Wide1) <= 1) {
                                printf("%s@%s ", x[seq], $1)
                                        # 後のシェルでの処理のために、先頭に
                                        # x[seq]@  を打ち、出典略号の $1 を打つ.
                                for(p=2; p<=j-1; p++) printf("%s ", $p)
                                        # シェル中の y20315 により、$1: となる.
                                printf("[%s %s %s]", $j, $(j+1), $(j+2))
                                }
                        if((j-Wide1) > 1){
                                printf("%s@%s ", x[seq], $1)
                                for(p=j-Wide1; p<=j-1; p++)
                                        printf("%s ", $p)
                                printf("[%s %s %s] ", $j, $(j+1), $(j+2))
                                }
                        for(r=j+3; r<=j+Wide2+1; r++) printf("%s ", $r)
                        printf("¥n")
                        j=j+2           #  再帰動詞の完了形だったので、
                                        #  j を 2 つ余分に進める.
                        }
                else   continue
                }
        else if((x[Sj1 " " Sj2] !="") && (a[Sj0 " " Sj1 " " Sj2] =="")) {
                                        #  再帰動詞の単純形か、非再帰動詞の完了形の場合.
                                        #  再帰動詞の完了形でもないという条件を付ける.
                                        #  これを先にすると, Se ha;  te has;
                                        #  os habe'is;...だけで取り出してしまう.
                seq=Sj1 " " Sj2
                c[x[seq]]++             #   x[seq] の語数をカウントする.
                if(c[x[seq]] <= Num) {
                        if((j-Wide1) <= 1) {
                                printf("%s@%s ", x[seq], $1)
```

```
                        for(p=2; p<=j-1; p++) printf("%s ", $p)
                    printf("[%s %s] ", $j, $(j+1))
                    }
                if((j-Wide1) > 1){
                    printf("%s@%s ", x[seq], $1)
                    for(p=j-Wide1; p<=j-1; p++)
                        printf("%s ", $p)
                    printf("[%s %s] ", $j, $(j+1))
                    }
                for(r=j+2; r<=j+Wide2+1; r++) printf("%s ", $r)
                printf("¥n")
                j=j+1              #  j を１つ余分に進める.
                }
            else   continue
            }
        else if((x[Sj1] !="") && (th[Sj1] =="") && (a[Sj0 " " Sj1] =="")){
                # 非再帰動詞の単純形か、一部の再帰動詞(命令形など)の場合.
                # これを先にすると, has;  He; など単独で数えてしまう.
                # 非再帰動詞の単純活用形には, una, tema, casa, ... など
                # 多数の同形異品詞語があるので, 簡単にチェックする.
            c[x[Sj1]]++              #   x[Sj1] の語数をカウントする.
            if(c[x[Sj1]] <= Num) {
                if((j-Wide1) <= 1) {
                    printf("%s@%s ", x[Sj1], $1)
                    for(p=2; p<=j-1; p++) printf("%s ", $p)
                    printf("[%s] ", $j)
                    }
                if((j-Wide1) > 1) {
                    printf("%s@%s ", x[Sj1], $1)
                    for(p=j-Wide1; p<=j-1; p++)
                        printf("%s ", $p)
                    printf("[%s] ", $j)
                    }
                 for(r=j+1; r<=j+Wide2; r++) printf("%s ", $r)
                printf("¥n")
                }
            else   continue
            }
        }
}
```

Appendix 5: y20315 :

```
#!/bin/gawk -f
#       名称 :  y20315
#       機能 :   入力行を、Keyword を中心にして、指定した左右幅
#                (文字数) で打ち出す.
#       書式 :  y20315 第１フィールド(テキスト略号)の表示指定
#               Keywordの指定   文字数による左右幅の指定    対象テキスト
#       書式例 : y20315    f1     1    160     int_pap_sue.txt
#       解説 :  第１フィールド(多くの場合、テキスト略号)の指定は :
#                       n0      表示しない.
#                       n?      ?文字数分(n3 なら、3文字)表示する.
#                       f1      第１フィールドを表示する.
```

```
#               Keyword の指定は、[... ] で囲まれた何番名の単語を
#               Keyword とするかの指定.
#########################################################################
BEGIN  {Str1__key=ARGV[1]; ARGV[1]=""
       parenfieldnum=ARGV[2]; ARGV[2]=""
       wid = ARGV[3]; ARGV[3] = ""
       key="¥[" "[a-zA-Z0-9'¥"`()<>{};:,.!?~=_¥xb2¥%¥/-]+" "¥]"
                                              # Keyword  候補.
       system( "rm fss" )
       }
{n=0
for(i=1; i<=NF; i++)
       {if($i ~ key)
              {n++}
       if(n==parenfieldnum)                   # n 番目がKeyword なら :
              {printf("%s¥t", $1)         >> "fss"
              for(j=2; j<=i-2; j++)
                     printf("%s ", $j)    >> "fss"
              printf("%s¥t%s ", $(i-1), $i)      >> "fss"
              if(NF==i)
                     {printf("¥n")        >> "fss"
                     next
                     }
              if(NF-i==1)
                     {printf("%s¥n", $NF)>> "fss"
                     next
                     }
              if(NF-i>=2)
                     {for(k=i+1; k<=NF-1; k++)
                            printf("%s ", $k)    >> "fss"
                     printf("%s¥n", $NF)         >> "fss"
                     next
                     }
              }
       }
}
END    {
close( "fss")
while(getline < "fss" >0 )  {
       split($0, x, "¥t")
       if(Str1_key ~ "n0")                    # 第1フィールドの指定 : n0
              printf("%s", "")
       else if(Str1__key ~ "n")    {
              field1=substr(x[1], 1, substr(Str1_key, 2))
                                              #   : がいくつも付かないようにする.
              gsub(":", "", field1)
              printf("%s: ", field1)
              }
       else if(Str1__key ~ "f1"){
              gsub(":", "", x[1])
              printf("%s: ", x[1])
              }
       printf("%" wid "s %s¥n", substr(x[2], length(x[2])-wid+1),
```

```
        substr(x[3], 1, wid))
        }
```

Appendix 6: y50926.sh :

```
#!/bin/bash
#       名称 :    y50926.sh
#       機能 :    (y50925で作った)抽出したい動詞の活用形リスト(xx26v_c)
#               を利用して、対象テキストから、指定した条件で、その
#               不定詞名のファイルで例文を、ディレクトリtemp17 中に作る.
#       書式 : y50926.sh -a抽出例文数 -b検索幅語数 -c表示幅指定
#               同形異品詞リスト  抽出対象動詞の活用形リスト
#               再帰動詞活用形リスト  対象テキスト
#       書式例 : y50926.sh   -a5   -b20   -c150   50919func   xx25v_c
#               40730s    int_pap_sue.txt
#       解説 : 同形異品詞リストと再帰動詞活用形リストは、処理の精度を
#               上げるために使用している.
#                 抽出対象動詞の活用形リストは、
#                       abandonar(abandonar.0.inf.0)
#                       abandonando(abandonar.0.ger.0)
#                       ...中略...
#                       he abandonado(abandonar.i.per.1s)
#                       ...以下略...
#                       のように、活用形(文法データ)  の形式リスト.
#########################################################################
Aflag=False                                     # オプション処理 :
Bflag=False
Cflag=False
Avalue=
Bvalue=
Cvalue=
OPT=
rm   -f  temp17/*
rm   -f  temp/xx67*
while    getopts   a:b:c:  OPT                 # オプション処理 :
do
        case  $OPT  in
        a)   Aflag=True
             Avalue=$OPTARG    ;;
        b)   Bflag=True
             Bvalue=$OPTARG    ;;
        c)   Cflag=True
             Cvalue=$OPTARG    ;;
        ¥?)  echo "Usage: $0   [-a out-put Wide words number]
                        [-b Keyword]"  1>&2
             exit  1
             ;;
        esac
done
shift  `expr  $OPTIND - 1`
if [ "$Aflag" = "True" ]; then
    Num=$Avalue                                 # 集める最大例文数(Num)
fi
if [ "$Bflag" = "True" ]; then
```

```
    Wide=$Bvalue                          #  例文抽出の Keyword の左右幅(語数)
fi
if [ "$Cflag" = "True" ]; then
    Wide_m=$Cvalue                        #  整形用の左右幅(文字数で)
fi
awk -f prog/y50926  $Num  $Wide  $Wide  $1  $2  $3  $4  >temp/xx67f
IFS_old=$IFS                              # スクリプトy50926は
IFS='                                     # Appendix 4.に掲載
'
for LINE in `cat temp/xx67f`
do
       s1=`echo "$LINE"  | perl  -F@  -lane  '{print $F[0]}'`
                                          # -F@  で @ をセパレータとする.
                                          # 第1フィールドをファイル名として取り出す.
       s2=`echo "$LINE"  | perl  -F@  -lane  '{print $F[1]}'`
                                          # 第2フィールドを例文として取り出す.
                                          # -l 入力行から改行を削除;  -a AWK 互換モード;
                                          # -n  1行ずつ読む;  -e スクリプトを読む.
       echo "$s2"  | awk -f  prog/y20315  f1  1  $Wide_m  >>temp17/$s1
                                          # 例文を、s1 名のファイルに、指定した左右幅
                                          # (文字数)で出力する.
                                          # スクリプトy20315は
                                          # Appendix 5.に掲載
Done
```