TR-I-0027

テキスト･データベースからの慣用表現の自動抽出

Automatic Idiom Extraction from Text Database

北　研二　　森元　逞
Kenji KITA　　Tsuyoshi MORIMOTO

1988.5

概要

テキスト･データベースから慣用表現を自動的に抽出する方法を提案するとともに、提案した方法を実際のテキスト･データベースに適用した結果について述べる。

ATR Interpreting Telephony Research Laboratories
ATR 自動翻訳電話研究所

－目次－

1．はじめに

自然言語処理の分野では、従来から慣用表現の重要性が指摘されており、慣用表現を網羅的に収集する試みもいくつかの研究機関で行われてきた。しかし、慣用表現の抽出は、これまではすべて人手で行われており、膨大な手間と時間を要した。また、慣用表現抽出の基準も曖昧であった。

本レポートでは、慣用表現抽出の基準を定式化するとともに、それを計算機上にインプリメントする方法について論じる。最後に、本レポートで提案した方法を実際のテキスト・データベースに適用した結果を示す。

2．研究の背景

自動翻訳電話実現のためには、音声処理と言語処理のインタフェースの研究が必要不可欠である。現在までに、言語処理の研究は種々のところで行われ、機械翻訳システムにおいては既に商用化の段階にまで至っている。しかし、従来の言語処理の研究は、テキスト中心であり、入力として言語的に正しいもののみを扱ってきた。しかし、言語処理の対象を実際の発話にまで広げようとすると、さまざまな問題が生じてくる。そのような問題点の一つに、言語処理への入力を一意に決められないというものがある。音声処理側では、類音語のうちのいずれであるかを確定することはきわめて困難なのである。特に、発声の短い単語ほど確定することが難しく、短い単語（助詞、助動詞）の連鎖である日本語の述語部分では、この傾向が顕著となる。

我々は、音声／言語インタフェースに関する広範な研究を行ってきたが、頻繁に使用される慣用的な表現を一つの単語とみなして処理することにより、かなりの割合で曖昧性を除去できることに気づいた。この方法は、単純ではあるが非常に効果的なものであると確信している。

次に、我々は慣用表現を効率的に収集する手法を研究した。本レポートは、この研究の成果をまとめたものである。

3．基本的な考え方

基本的な考えは、テキスト中に多く出現する単語列を見つけ出すということである。しかし、単に単語列といっても、２つの単語から成る列もあるし、３つの単語から成る列もあり、単純に単語列の出現回数で比較するわけにはいかない。

例えば、あるテキスト中に、"in spite" という単語列が１１０回、"in spite of" という単語列が１００回、"in spite of XXX" という単語列が２０回出現したとする。この場合、"in spite" が最も出現回数が多いからといって、単純に "in spite" を慣用表現とするのには問題がある。なぜならば、"in spite" という用いられ方において、ほとんどの場合はその後に "of" を伴って用いられているからである。いまの場合は、"in spite" は無視して "in spite of" を慣用表現としなければならない。

従って、テキスト中から慣用表現を抽出する際には、単語列の出現頻度と、単語列が生成される系列（上の例の場合、"in spite" -> "in spite of" -> "in spite of XXX" ... という系列）間の出現頻度の差を同時に考慮する必要がある。

４．慣用表現抽出の基準

最初に、いくつかの表記法を導入する。αを単語列とするとき、次のような表記法を用いる。

| α |　・・・ 単語列の長さ（いくつの単語から成る単語列であるかを示す）.
n（α）・・・ 単語列の出現回数.

単語列の比較に用いる尺度に、我々は以下の式の値を用いる。すなわち、下の式の値の大きい単語列を慣用表現とみなし、テキスト・データベースから抽出する。

$$K(\alpha) = (|\alpha| - 1) \times n(\alpha)$$

まず、K（α）の意味付けを行う。

いま、テキスト中にｎ個の単語から成る慣用表現αがあったとする。もし、αをｎ個の単語から成るものとして処理すれば、ｎに比例するだけの仕事量が必要である。しかし、αを１つの表現として処理すれば、仕事量は１でよい。すなわち、（ｎ－１）分の仕事量が節約されたことになる。αが、テキスト中にｍ回出現するならば、

$$(n-1) \times m$$

分の仕事量が節約される。これがK（α）の意味付けである。

次に、同一の単語列生成系列中の２つの単語列の扱いについて説明する。

αとβが同一の単語列生成系列に属し、しかも | α | < | β | であったとする（すなわち、αはβの部分単語列である）。このとき、当然のことながら、

$$n(\alpha) \geq n(\beta)$$

である。αとβの両方を慣用表現としてテキストを処理する場合を考える。βはテキスト処理の際にn（β）回参照されるが、αについて考えると、αの出現回数n（α）のうちn（β）回についてはβが参照されているので、純粋にαが参照されるのは、

$$n(\alpha) - n(\beta)$$

回だけである。従って、αとβの両方を慣用表現として採用する場合には、αの持つKの値を

$$(|\alpha| - 1) \times (n(\alpha) - n(\beta))$$

に変更する必要がある。

一般に、共通の部分単語列を持つ単語列の集合

α

$\alpha, \beta i \quad (i = 1, \ldots, m)$

$\gamma j, \alpha \quad (j = 1, \ldots, n)$

を同時に慣用表現と考える場合、共通部分単語列αに対しては、

$$K(\alpha) = (|\alpha| - 1) \times \left(n(\alpha) - \sum_i n(\alpha, \beta i) - \sum_j n(\gamma j, \alpha) + \sum_i \sum_j n(\gamma j, \alpha, \beta i) \right)$$

となる。

5．インプリメンテーション

慣用表現抽出のためには、次の２つの操作を実現する必要がある。

（１）テキストを走査し、各単語列の出現回数を調べる操作.
（２）同一の単語列生成系列に属する単語列のKの値を動的に変更しながら、Kの値の大きい方から単語列を順次取ってくる操作.

（２）の操作を、厳密な意味で実現しようとすると、計算量が爆発的に増大する。なぜならば、単語列を１つ取ってくる度に、これまでに取られた単語列のすべてに対して同一の生成系列に属するか否かを調べて、Kの値を再計算しなおさなければならないからである。

例で説明しよう。いま、単語A，B，C，D，E・・・に対して、単語列の出現回数が以下のようになったとする。

単語列	出現回数	最初	再計算１	再計算２	再計算３	・・・
ＡＢ	１００	１００	７０	６０	４０	
ＡＢＣＤ	３０	９０	９０	９０	９０	
ＡＢＣ	４０	８０	２０	２０	２０	
ＡＢＥ	２０	４０	４０	４０	４０	
・・・	・・	・・	・・	・・	・・	

このとき、まずＫの値の最も大きいＡＢ（Ｋ＝１００）が取られる。次に、２番目にＫの値の大きいＡＢＣＤ（Ｋ＝９０）が取られるが、既に取られているＡＢは、ＡＢＣＤの部分単語列となっているので、ＡＢの持つＫの値は７０に修正される（再計算１）。このあと、ＡＢＣが取られ、再度ＡＢの持つＫの値が再計算される（再計算２）。同様に、ＡＢＥが取られた時点でも、ＡＢに対して再計算が行われる（再計算３）。

上の例は、比較的単純な例であるが、実際には単語列は１方向だけではなく、両方向に成長する。この場合、４節の最後に示した複雑な計算を、単語列が１つ取られる度にしなおさなければならない。

我々は、計算量の爆発を防ぐために、（２）の操作を近似的に行う方法をとった。この方法は、以下のようなものである。

- Ｋの値の再計算は生成系列中で隣合った単語列どうしに限る。
  上の例の場合、ＡＢとＡＢＣＤは生成系列中で隣合っていないので、再計算１は行われない。
- 再計算は１度だけに限る。
  上の例の場合、ＡＢＣが取ってこられた時点で、ＡＢの再計算が行われているので、ＡＢＥが取ってこられてもＡＢの再計算は行わない。つまり、再計算３は行われない。

次に、操作（１）のテキスト走査について説明する。

操作（１）は、比較的単純であるが、操作（２）の再計算の際に、２つの単語列が生成系列中で隣合っているか否かを高速に判断できるようにしなければならない。このため、テキスト走査の際には、各単語列の出現回数をカウントするとともに、単語列生成系列をあとから参照できるように、同時に作成する。

次の２つのデータ型を用意する。

(1) FIFO(First-In First-Out) 型のスタック WordStack[N_GRAM].

WordStack の大きさ N_GRAM は、慣用表現としていくつの単語から成るものまでを許すかを意味する。プログラムは、テキスト内の単語を先頭から走査し、WordStack に積む。WordStack[0] には最も新しく参照された単語が、WordStack[1] には直前に参照された単語が、WordStack[i] には i 個前に参照された単語が格納される。

(2) レコード構造 WordSequence.

WordSequence は、次のようなレコードである。

```
var
  WordSequence : record
    String   : array of char;
    Count    : integer;
    UpperSeq : array of ↑WordSequence;
    Next     : ↑WordSequence;
  end;
```

String は、単語列を格納するためのものであり、Count は格納された単語列がテキスト中に何回出現したかを記憶する。UpperSeq は、単語列生成系列中で現在の String よりも１つ単語数の多い WordSequence を指すポインタの集合である。単語列生成系列は、一般に多数の分岐を持つために、ポインタの集合となっている。

なお、レコード WordSequence は、１つの単語列に対し１つ存在するが、これらはすべてリスト状に結ばれており、新たなレコード WordSequence が作られる度に、リストの最後につながれる。Next は、リストの次の要素を指すポインタである。

以下に、テキスト走査のプログラムを示す。text[i] は、テキスト中の i 番目の単語を表している。

```
procedure TextScan
var
  i, j, k, l : integer;
  CurrentSeq : WordSequence;
begin
  i := 0;
  while text[i] <> END-OF-TEXT do begin
    push text[i] to WordStack;
    j := 1;
    while j < N_GRAM do begin
      if already exists WordSequence such that
      WordSequence.String = WordStack[j]...WordStack[0], begin
        CurrentSeq := WordSequence;
        CurrentSeq.Count := CurrentSeq.Count + 1;
      end;
      else begin
        CurrentSeq := allocate new WordSequence;
        CurrentSeq.String := WordStack[j]...WordStack[0];
        CurrentSeq.Count := 1;
      end;
      if j <> 1 begin
        search WordSequence such that
          WordSequence.String = WordStack[j]...WordStack[1];
        search empty UpperSeq[k] in WordSequence;
        WordSequence.UpperSeq[k] := CurrentSeq;
        search WordSequence such that
          WordSequence.String = WordStack[j-1]...WordStack[0];
        search empty UpperSeq[l] in WordSequence;
        WordSequence.UpperSeq[l] := CurrentSeq;
      end;
      j := j + 1;
    end;
    i := i + 1;
  end;
end;
```

６．実験結果

上で提案した手法の有効性を確認する実験を行った。

実験には、３つの異なる分野のテキストを用い、単語列の生成系列は１０単語から成るものまでを扱った。また、単語列のうちＫの値の大きいもの０．５％についてのみＫの再計算を行い、再計算を行ったものの中から上位１／２を慣用表現として抽出した。

実験に用いたテキストを次に示す。

（実験１）対象　　：キーボード会話
　　ディレクトリ：atr-ln:/data3/MORPH/DATA/key
　　総文数　　：１１９７
　　総単語数　　：１２６６９
　　異り単語列数：５７４４５

（実験２）対象　　：電話会話
　　ディレクトリ：atr-ln:/data3/MORPH/DATA/tel
　　総文数　　：２７５８
　　総単語数　　：３２３９９
　　異り単語列数：１５６５９８

（実験３）対象　　：新聞記事
　　ディレクトリ：atr-ln:/data3/MORPH/DATA/new
　　総文数　　：６２６
　　総単語数　　：１６０９５
　　異り単語列数：１０９４３４

次ページ以降に、慣用表現として抽出されたものの一覧を示す。これを見ると、比較的妥当と思えるものが抽出されており、本レポートで提案した手法の有効性をうかがうことができる。

キーボード会話からの慣用表現抽出結果

K: <N-gram> Occurrences: Word-sequence

130: <3> 93: でしょ,う,か
109: <2>150: です,か
 88: <5> 28: し,たい,の,です,が
 84: <3> 85: の,です,が
 72: <3> 60: 分り,まし,た
 72: <4> 28: の,でしょ,う,か
 72: <4> 24: はい,分り,まし,た
 70: <3> 41: の,です,か
 70: <2> 79: ます,か
 66: <4> 22: どうも,ありがとう,御座いまし,た
 64: <2> 76: し,て
 60: <2>145: の,です
 57: <4> 27: お,願い,し,ます
 56: <3> 39: て,おり,ます
 55: <2> 67: ませ,ん
 46: <3> 23: そう,です,か
 45: <4> 43: たい,の,です,が
 44: <2> 79: し,ます
 42: <3> 31: と,思い,ます
 40: <2>100: まし,た
 38: <2>123: です,が
 38: <2> 46: の,方
 36: <3> 25: て,い,ます
 34: <2> 44: ます,ので
 34: <3> 17: に,なり,ます
 33: <2> 56: そう,です
 33: <2> 44: 致し,ます
 32: <3> 16: はい,そう,です
 32: <5> 8: を,お,願い,し,ます
 30: <4> 10: ます,でしょ,う,か
 28: <3> 36: ありがとう,御座いまし,た
 28: <2> 37: ます,が
 28: <2> 28: に,は
 27: <2> 39: を,お
 27: <4> 9: て,戴け,ます,か
 26: <2> 26: 会議,の

25: <6> 5: し,たい,と,思い,ます,が
25: <6> 5: を,し,たい,の,です,が
24: <3> 12: ませ,ん,が
24: <4> 8: 御,願い,し,ます
24: <5> 6: さ,せ,て,戴き,ます
24: <7> 4: お,聞き,し,たい,の,です,が
24: <7> 4: を,お,待ち,し,て,おり,ます
23: <2> 31: です,ね
23: <2> 23: を,御
22: <3> 11: を,し,て
22: <3> 11: かしこまり,まし,た
21: <4> 7: お,送り,致し,ます
21: <4> 7: た,の,です,が
21: <4> 7: て,い,ます,か
21: <4> 7: を,教え,て,下さい
21: <4> 7: 御,願い,致し,ます
21: <4> 7: ない,の,です,が
21: <8> 3: 会議,に,参加,し,たい,の,です,が
20: <2> 20: と,いう
20: <2> 20: 結構,です
20: <3> 10: 会議,に,参加
20: <3> 10: 登録,料,は
20: <6> 4: お,待ち,し,て,い,ます
20: <6> 4: 事,に,なっ,て,おり,ます
19: <2> 19: ホテル,の
19: <2> 19: 会議,場
18: <2> 33: て,下さい
18: <4> 11: し,て,おり,ます
18: <2> 18: 私,は
18: <3> 9: が,有り,ます
18: <3> 9: さ,れる,の
18: <3> 9: と,申し,ます
18: <3> 9: ん,です,が
18: <3> 9: 通訳電話国際会議事務局,です,か
18: <3> 9: ます,ので,それ
18: <3> 9: 失礼,し,ます
18: <3> 9: 京都,駅,から
18: <4> 6: て,いる,の,です
18: <4> 6: て,おり,ませ,ん
18: <4> 6: は,いつ,です,か

18: <4> 6: 良い,の,です,か
18: <4> 6: れる,の,です,か
18: <7> 3: 御,尋ね,し,たい,の,です,が
18: <7> 3: 用紙,を,お,送り,し,ます,ので
18: <10> 2: を,お,出し,に,なり,たい,の,でしょ,う,か
18: <10> 2: ペーパー,を,お,出し,に,なり,たい,の,でしょ,う
18: <10> 2: 電話,番号,クレジットカード,の,名前,と,番号,を,教え,て
18: <10> 2: 番号,クレジットカード,の,名前,と,番号,を,教え,て,下さい
17: <2> 17: まし,て
16: <3> 35: 願い,し,ます
16: <2> 45: し,たい
16: <3> 15: 教え,て,下さい
16: <2> 28: お,送り
16: <2> 16: あり,ます
16: <2> 16: 会議,は
16: <3> 8: こちら,の,方
16: <3> 8: し,まし,た
16: <3> 8: て,居り,ます
16: <3> 8: の,です,ね
16: <3> 8: の,方,で
16: <3> 8: そう,です,ね
16: <3> 8: どう,すれ,ば
16: <3> 8: ませ,ん,か
16: <3> 8: 出来,ます,か
16: <3> 8: 大阪,城,の
16: <5> 4: 宜しく,お,願い,し,ます
16: <5> 4: と,思う,の,です,が
16: <5> 4: に,なる,と,思い,ます
16: <5> 4: 学生,な,の,です,が
16: <5> 4: 有る,の,でしょ,う,か
16: <9> 2: が,そちら,で,ホテル,の,手配,は,し,て
16: <9> 2: て,おり,ます,ので,それ,を,御,利用,下さい
16: <9> 2: 研究,を,し,て,いる,者,です,が,会議
16: <9> 2: 住所,電話,番号,クレジットカード,の,名前,と,番号,を
16: <9> 2: クレジットカード,を,持っ,て,い,ない,の,です,が

電話会話からの慣用表現抽出結果

K: <N-gram> Occurrences: Word-sequence

460: <3>370: でしょ,う,か
348: <4>140: ん,でしょ,う,か
326: <3>221: ん,です,けれども
322: <2>411: の,方
255: <4> 85: あ,そう,です,か
254: <3>212: そう,です,か
222: <4> 74: はい,わかり,まし,た
192: <3>123: と,いう,こと
160: <2>381: ん,です
156: <2>221: です,ね
150: <4> 65: ます,でしょ,う,か
144: <4> 48: ああ,そう,です,か
143: <2>176: し,て
142: <3> 92: と,思い,ます
141: <4> 58: な,ん,です,けれども
140: <3> 89: の,方,に
132: <3>121: て,おり,ます
128: <3> 64: の,方,は
126: <3>137: わかり,まし,た
120: <5> 45: に,なっ,て,おり,ます
117: <4> 48: そう,し,まし,たら
116: <2>328: そう,です
110: <3> 80: こちら,の,方
108: <4> 36: あっ,そう,です,か
100: <3>108: な,ん,です
98: <3> 70: ん,です,が
93: <2>216: と,いう
93: <4> 39: よろしい,でしょ,う,か
88: <3> 44: の,方,が
86: <3> 65: そう,です,ね
86: <3> 43: はい,そう,です
76: <5> 31: と,思う,ん,です,けれども
76: <3> 38: そう,いっ,た
75: <4> 25: こちら,の,方,から
71: <2>283: です,か
66: <2> 84: まし,て

64: <3> 53: に,なり,ます
64: <3> 47: の,方,で
63: <4> 21: な,ん,です,が
63: <4> 21: わけ,でしょ,う,か
60: <2> 71: に,は
60: <4> 20: どうも,ありがとう,ございまし,た
60: <4> 20: と,いう,こと,で
60: <5> 15: て,いる,ん,です,けれども
60: <5> 15: よろしく,お,願い,いたし,ます
59: <2> 89: ます,ので
57: <2> 90: お,願い
57: <4> 30: て,おり,ます,ので
57: <4> 27: と,いう,こと,です
57: <4> 19: た,ん,です,けれども
57: <4> 19: と,いう,の,は
57: <4> 19: ん,です,けれども,あの
54: <3> 44: ない,ん,です
54: <10> 6: そちら,第,１,回,通訳電話国際会議,の,事務局,でしょ,う,か
52: <5> 24: し,たい,ん,です,けれども
52: <3> 39: そちら,の,方
52: <2> 66: 結構,です
51: <4> 26: と,いう,こと,に
51: <4> 17: と,いう,こと,は
51: <4> 17: ない,ん,です,けれども
50: <6> 15: こと,に,なっ,て,おり,ます
50: <3> 25: です,ね,あのー
50: <6> 10: と,いう,こと,でしょ,う,か
48: <2>269: です,けれども
48: <4> 47: と,思う,ん,です
48: <4> 22: し,て,おり,ます
48: <5> 12: ん,です,けれども,あ,はい
48: <7> 8: すれ,ば,よろしい,ん,でしょ,う,か
45: <4> 23: ん,です,けれども,はい
45: <4> 15: こちら,の,方,で
45: <4> 15: お,願い,し,ます
44: <2> 82: いっ,た
44: <2> 55: 会議,の
44: <2> 44: １００,ドル
44: <3> 22: を,お,送り
44: <3> 22: ん,です,か

44: <5> 11: な,ん,です,けれども,あのー
42: <3> 67: に,なっ,て
42: <3> 41: ありがとう,ございまし,た
42: <2> 69: こと,に
42: <4> 22: そう,です,ね,あの
42: <2> 64: お,送り
42: <4> 21: こと,に,なり,ます
42: <3> 21: わけ,です,ね
42: <3> 21: 京都,駅,から
42: <4> 14: はい,その,通り,です
42: <7> 7: 第,1,回,通訳電話国際会議事務局,でしょ,う,か
41: <2> 69: いたし,ます
41: <2> 55: ませ,ん
41: <2> 52: ます,が
41: <2> 41: あ,もしもし
41: <2> 41: は,あの
40: <5> 24: よろしい,ん,でしょ,う,か
40: <3> 20: 会議,場,の
40: <5> 10: さ,せ,て,いただき,ます
40: <5> 10: はい,そう,で,ございます,が
39: <4> 37: し,たい,ん,です
39: <4> 28: お,願い,いたし,ます
39: <4> 19: こちら,の,方,に
39: <2> 39: ます,と
39: <2> 39: まで,に
39: <4> 13: そちら,の,方,に
39: <4> 13: と,いう,風,に
39: <4> 13: ない,と,いう,こと
39: <4> 13: あの,こちら,の,方
38: <3> 44: の,方,から
38: <3> 38: て,いただき,ます
38: <2> 75: し,たい
38: <2> 38: それ,は
37: <2> 59: を,お
37: <2> 37: で,は
36: <3> 33: お,願い,し
36: <7> 11: お,伺い,し,たい,ん,です,けれども
36: <2> 54: に,お
36: <3> 18: に,お,送り
36: <3> 18: ます,ので,はい

36: <3> 18: 失礼,いたし,ます
36: <4> 12: だ,と,思い,ます
36: <4> 12: ん,です,けれども,えーと
36: <4> 12: 登録,用紙,の,方
36: <5> 9: できる,ん,でしょ,う,か
36: <5> 9: な,ん,でしょ,う,か
36: <5> 9: えー,そう,し,まし,たら
36: <7> 6: お,教え,願え,ます,でしょ,う,か
36: <7> 6: し,たい,と,思う,ん,です,けれども
36: <7> 6: て,いただき,たい,と,思う,ん,です
36: <7> 6: こと,は,でき,ます,でしょ,う,か
35: <2> 63: て,いる
35: <8> 5: の,で,は,ない,か,と,思い,ます
35: <8> 5: ちょっと,お,伺い,し,たい,ん,です,けれども
34: <2> 72: て,いただき
34: <3> 25: そ,し,たら
34: <2> 34: あり,ます
34: <2> 34: これ,は
34: <3> 17: の,方,も
34: <3> 17: の,方,の
34: <3> 17: の,方,を
34: <3> 17: スピーカー,の,方
33: <4> 35: たい,ん,です,けれども
33: <2> 57: たい,と
33: <4> 11: お,願い,致し,ます
33: <4> 11: し,て,いただい,て
33: <4> 11: どう,いたし,まし,て
33: <4> 11: 恐れ,入り,ます,が
32: <3> 53: たい,ん,です
32: <2>102: です,が
32: <5> 15: でき,ます,でしょ,う,か
32: <5> 13: ある,ん,でしょ,う,か
32: <2> 46: あ,はい
32: <3> 16: し,て,いただく
32: <3> 16: で,お,支払い
32: <3> 16: の,中,に
32: <3> 16: ん,です,ね
32: <3> 16: どう,いっ,た
32: <3> 16: 参加,さ,れる
32: <5> 8: し,て,おり,ませ,ん

32: <5> 8: て,る,ん,です,けれども
32: <5> 8: と,いう,こと,です,ね
32: <5> 8: な,ん,です,けれども,はい
32: <5> 8: ええ,そう,です,ね,あの
32: <5> 8: 送ら,せ,て,いただき,ます
32: <9> 4: て,いただき,ます,ので,よろしく,お,願い,いたし,ます
31: <2> 48: 登録,用紙
30: <4> 55: なっ,て,おり,ます
30: <6> 14: ば,よろしい,ん,でしょ,う,か
30: <6> 12: たい,と,思う,ん,です,けれども
30: <3> 28: 第,１,回
30: <6> 11: に,なっ,て,おり,ます,ので
30: <4> 18: ございます,でしょ,う,か
30: <4> 16: ある,ん,です,けれども
30: <2> 30: って,いう
30: <2> 30: ます,けれども
30: <3> 15: て,いただけれ,ば
30: <3> 15: と,いい,ます
30: <3> 15: こう,いっ,た
30: <4> 10: の,方,に,は
30: <6> 6: て,いただき,たい,と,思い,ます
30: <6> 6: と,いう,こと,に,なり,ます
30: <6> 6: と,いう,こと,に,なる,と
30: <6> 6: に,なる,ん,でしょ,う,か
30: <6> 6: の,こと,な,ん,です,けれども
30: <7> 5: 含まれ,て,いる,ん,でしょ,う,か
30: <7> 5: お,願い,し,たい,ん,です,が
30: <7> 5: に,お,送り,すれ,ば,よろしい,ん
30: <7> 5: ん,です,が,よろしい,でしょ,う,か
30: <7> 5: こと,に,なっ,て,おり,ます,ので
29: <2>166: まし,た
29: <2> 48: いただい,て
29: <2> 29: こと,が
28: <5> 13: と,思う,ん,です,が
28: <5> 13: いただき,たい,と,思い,ます
28: <2> 49: わけ,です
28: <8> 7: その,ほか,に,何か,ございます,でしょ,う,か
28: <3> 24: し,たい,と
28: <5> 12: し,たい,ん,です,が
28: <2> 43: し,ます

28: <2> 40: ます,か
28: <2> 28: 書い,て
28: <3> 14: し,て,いる
28: <3> 14: わけ,です,か
28: <3> 14: でも,結構,です
28: <3> 14: 失礼,し,ます
28: <3> 14: 国際,会議,場
28: <5> 7: ように,なっ,て,おり,ます
28: <5> 7: え,そう,し,まし,たら
28: <5> 7: さ,し,て,いただき,ます
28: <5> 7: し,たい,と,思い,ます
28: <5> 7: せ,て,いただき,ます,ので
28: <8> 4: し,たい,こと,が,ある,ん,です,が
28: <8> 4: と,いう,こと,に,なっ,て,おり,ます
28: <8> 4: 会議,に,参加,し,たい,ん,です,けれども
27: <4> 19: せ,て,いただき,ます
27: <2> 47: こと,は
27: <2> 27: です,ので
27: <2> 27: それ,と
27: <2> 27: 送っ,て
27: <4> 9: でしょ,う,か,それとも
27: <4> 9: そちら,の,方,で
27: <4> 9: お,送り,いたし,ます
27: <4> 9: て,ない,ん,です
27: <4> 9: あー,そう,です,か
27: <4> 9: ない,ん,です,が
27: <10> 3: あの,そちら,第,1,回,通訳電話国際会議,の,事務局,でしょ,う
26: <2> 42: で,お
26: <2> 41: に,なる
26: <2> 41: でき,ます
26: <2> 38: でし,たら
26: <2> 26: ホテル,の
26: <2> 26: スピーカー,として
26: <3> 13: で,ある,とか
26: <3> 13: ませ,ん,ので
26: <3> 13: 参加,者,の
26: <3> 13: 参加,し,たい
25: <2> 55: の,は
25: <2> 44: 登録,料
25: <2> 25: お,電話

25: <6> 5: が,ある,ん,でしょ,う,か
25: <6> 5: さ,れる,ん,でしょ,う,か
25: <6> 5: て,おり,ます,でしょ,う,か
25: <6> 5: て,いただき,たい,ん,です,けれども
25: <6> 5: で,よろしい,ん,でしょ,う,か
25: <6> 5: いけ,ない,ん,でしょ,う,か
25: <6> 5: こと,に,なる,と,思い,ます
25: <6> 5: 書い,て,ある,ん,です,けれども
25: <6> 5: 予定,に,なっ,て,おり,ます

新聞記事からの慣用表現抽出結果

K: <N-gram> Occurrences: Word-sequence

110: <2>169: て,いる
104: <3> 59: し,て,いる
 90: <2>100: し,た
 60: <3> 42: て,い,た
 59: <2> 71: と,いう
 59: <2> 70: で,ある
 55: <2> 79: で,は
 49: <2>108: し,て
 48: <2> 48: に,は
 45: <4> 21: か,も,しれ,ない
 34: <3> 23: さ,れ,た
 33: <4> 11: さ,れ,て,いる
 32: <9> 4: し,て,いる,の,で,は,ない,か,と
 30: <3> 24: で,は,ない
 30: <2> 44: た,の
 30: <3> 15: に,よっ,て
 28: <2> 40: だろ,う
 27: <4> 12: し,て,い,た
 26: <2> 33: だっ,た
 26: <2> 26: に,も
 24: <3> 17: て,き,た
 24: <2> 29: だ,と
 24: <2> 24: と,の
 22: <3> 15: と,し,て
 22: <3> 14: た,の,だ
 22: <3> 11: た,の,は
 21: <2> 33: た,と
 20: <3> 14: て,いる,と
 20: <2> 27: ない,と
 20: <3> 10: こう,し,た
 19: <2> 24: 的,な
 18: <2> 32: の,だ
 18: <3> 16: て,いる,の
 18: <3> 12: た,と,いう
 18: <3> 12: だろ,う,と
 18: <2> 18: なかっ,た

18: <3> 9: に,とっ,て
18: <3> 9: 彼,ら,は
18: <4> 6: さ,れ,た,と
18: <4> 6: て,いる,の,だ
17: <2> 40: さ,れ
17: <2> 17: こと,を
16: <2> 16: さ,れる
16: <2> 16: に,対する
16: <2> 16: 指導,者
16: <3> 8: と,し,た
15: <2> 38: れ,た
15: <6> 5: た,の,か,も,しれ,ない
15: <4> 8: た,の,で,ある
15: <4> 5: し,て,き,た
15: <4> 5: と,いう,の,だ
15: <4> 5: の,だろ,う,か
15: <4> 5: れ,て,い,た
15: <4> 5: わけ,で,は,ない
15: <6> 3: し,なけれ,ば,なら,ない,と
15: <6> 3: に,なる,か,も,しれ,ない
14: <2> 29: と,し
14: <2> 25: の,は
14: <2> 19: に,なる
14: <2> 14: た,が
14: <2> 14: て,は
14: <2> 14: こと,は
14: <2> 14: 左翼,の
14: <3> 7: で,ある,と
14: <3> 7: これ,ら,の
14: <3> 7: もの,だっ,た
14: <3> 7: られ,て,いる
13: <2> 19: なっ,た
13: <2> 18: こと,が
13: <2> 13: さ,せる
13: <2> 13: と,する
13: <2> 13: も,ある
13: <2> 13: ヨーロッパ,の
12: <2> 27: れ,て
12: <2> 12: が,ある
12: <2> 12: だ,が

12: <2> 12: 年,に
12: <2> 12: 人,の
12: <2> 12: あっ,た
12: <2> 12: から,の
12: <2> 12: する,と
12: <2> 12: すれ,ば
12: <2> 12: それ,は
12: <2> 12: 保守,派
12: <3> 6: に,なっ,た
12: <3> 6: に,ある,と
12: <3> 6: の,ため,に
12: <3> 6: なっ,て,いる
12: <3> 6: 語っ,て,いる
12: <4> 4: し,た,こと,が
12: <4> 4: し,て,いる,と
12: <4> 4: て,いる,と,いう
12: <4> 4: と,し,て,いる
12: <4> 4: と,考え,て,いる
12: <4> 4: に,も,かかわら,ず
12: <4> 4: なっ,て,い,た
12: <5> 3: と,し,て,い,た
12: <5> 3: の,ある,西側,外交官,は
12: <5> 3: 南欧,の,社会主義,政党,は
12: <7> 2: に,なっ,た,か,も,しれ,ない
12: <7> 2: の,で,は,ない,か,と,いう
12: <7> 2: の,で,は,ない,か,と,の
12: <7> 2: だっ,た,の,か,も,しれ,ない
11: <2> 23: う,と
11: <2> 16: 党,の
11: <2> 16: こと,に
11: <2> 11: は,この
11: <2> 11: 者,は
11: <2> 11: 彼,は
11: <2> 11: する,こと
10: <2> 52: て,い
10: <6> 6: の,で,は,ない,か,と
10: <3> 9: し,た,こと
10: <3> 8: し,た,の
10: <3> 8: し,よう,と
10: <3> 8: て,い,ない

10: <2> 10: と,は
10: <2> 10: は,言う
10: <2> 10: 者,に
10: <2> 10: 派,の
10: <2> 10: 一,人
10: <2> 10: トウ小平,の
10: <3> 5: か,と,いう
10: <3> 5: と,思わ,れる
10: <3> 5: と,なっ,た
10: <3> 5: と,いう,もの
10: <3> 5: に,なっ,て
10: <3> 5: に,よれ,ば
10: <3> 5: は,党,の
10: <3> 5: なく,なっ,た
10: <3> 5: こと,が,できる
10: <3> 5: こと,に,なる
10: <3> 5: だっ,た,の
10: <3> 5: べき,だ,と
10: <6> 2: さ,れ,た,と,発表,し
10: <6> 2: し,よう,と,し,て,いる
10: <6> 2: て,いる,の,に,対し,ＳＰＤ
10: <6> 2: と,いう,わけ,で,は,ない
10: <6> 2: と,し,た,もの,だっ,た
10: <6> 2: は,年初,来,の,株価,急騰
10: <6> 2: を,阻止,し,た,こと,を
10: <6> 2: ない,と,警告,し,て,いる
10: <6> 2: ない,と,トウ小平,は,語っ,た
10: <6> 2: 被害,を,受け,た,の,は
10: <6> 2: 宣言,し,た,の,で,ある

付録. プログラム

```
/*-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 *
 *     Automatic Idiom Extraction from Text Database
 *     ---------------------------------------------
 *
 *
 *     Copyright (C) 1988,  Kenji Kita.
 *
 *     EDIT HISTORY
 *        First version created at 3-Mar-1988.
 *        Last modified at 24-May-1988.
 *
 *-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-*/

#include <stdio.h>
#include <ctype.h>

#define EOS             (-2)    /* End of sentence */
#define FALSE           0
#define TRUE            1
#define N_GRAM          10
#define WORDSIZE        256
#define STRSIZE         1024
#define STRING          'a'
#define NUMBER          '0'
#define DELIMIT_CHAR    '¥001'

int     verbose = 0,
        sentences = 0,          /* Total sentences */
        totals[N_GRAM],         /* Total N-grams, totals[0]...total words */
        d_totals[N_GRAM];       /* Different total N-grams */

float           percent = 2.0;
```

```
/*
 * 'WordSeq' structure definition.
 */

#define B_PRIME         1001
#define HASHSIZE        B_PRIME*N_GRAM

struct WordSeq
{
        char    *Sequence;
        int     Count;
        struct ConStruct        *UpperLinks;
        struct WordSeq          *Next;
};
/*
 *  Sequence   -> Word sequence.
 *  Count      -> Occurrences.
 *  UpperLinks -> Set of pointers to upper N-gram.
 *  Next       -> Pointer to next N-gram (same level).
 */

struct WordSeq  *hashtable[HASHSIZE];
/*
 *  hashtable[0]          ... hashtable[B_PRIME-1]        -> for Bigrams.
 *  hashtable[B_PRIME]    ... hashtable[2*B_PRIME-1]      -> for Trigrams.
 *     .....
 *  hashtable[n*B_PRIME] ... hashtable[(n+1)*B_PRIME-1]  -> for (n+2)-grams.
 */

struct ConStruct
{
        struct WordSeq          *UpperLink;
        struct ConStruct        *Next;
};
```

```
Usage(progname)
char    *progname;
{
        fprintf(stderr, "Usage: %s file or %s @meta-file¥n", progname);
        exit();
}

main(argc, argv)
int     argc;
char    *argv[];
{
        register int    i;
        double  atof();

        for(i = 0; i < N_GRAM; i++)
        {
                totals[i] = 0;
                d_totals[i] = 0;
        }
        if(argc < 2)
                Usage(argv[0]);
        while(--argc)
        {
            if(***++argv == '-')
            {
                switch((*argv)[1])
                {
                case 'p':
                    if(argc-1 <= 0 || !isdigit(**(argv+1)))
                        fatal_error("! Missing value after 'p' option¥n");
                    percent = (float)atof(*(argv+1));
                    --argc;
                    ++argv;
                    break;
                case 'v':
                    ++verbose;
                    break;
                default:
                    fatal_error("! Unknown option : %c¥n", (*argv)[1]);
                }
```

```
            }
            else if(**argv == '@')
            {
                FILE    *fp;
                char    filename[256];

                if((fp = fopen(&((*argv)[1]), "r")) == NULL)
                    fatal_error("Cannot open %s", *argv);
                while(fgets(filename, sizeof(filename), fp) != NULL)
                {
                    if(strlen(filename) == 1)
                        continue;
                    filename[strlen(filename) - 1] = NULL;
                    textscan(filename);
                }
            }
            else
                textscan(*argv);
        }
        dump_ngram();
}

textscan(filename)
char    *filename;
{
        register int    i, s;
        char    wordstack[N_GRAM][WORDSIZE],
                seq[STRSIZE],       /* wordstack[i]   ... wordstack[0] */
                subseq0[STRSIZE],   /* wordstack[i-1] ... wordstack[0] */
                subseq1[STRSIZE];   /* wordstack[i]   ... wordstack[1] */
        struct WordSeq  *p, *q, *lookup(), *count_up();
        FILE    *infp;

        /*
         *  wordstack[0] ... Current word.
         *  wordstack[1] ... Previous word.
         *  ......................
         */
```

```
fprintf(stderr, "[ Processing file %s ]¥n", filename);
if((infp = fopen(filename, "r")) == NULL)
{
        error("Cannot open %s¥n",  filename);
        return;
}
for(i = 0; i < N_GRAM; i++)
        wordstack[i][0] = NULL;
while(1)
{
        if((s = getword(infp, wordstack[0], WORDSIZE)) == EOF)
                break;
        else if(s == EOS)  /* End of sentence */
        {
                if(verbose)
                        printf("[ End of sentence ]¥n");
                sentences += 1;
                for(i = 0; i < N_GRAM; i++)
                        wordstack[i][0] = NULL;
                continue;
        }
        if(verbose)
                printf("%s¥n", wordstack[0]);
        totals[0] += 1;
        strcpy(seq, wordstack[0]);
        strcpy(subseq0, wordstack[0]);
        subseq1[0] = NULL;
        for(i = 1; i < N_GRAM; i++)
        {
                if(wordstack[i][0] == NULL)
                        break;
                r_strcat(seq, wordstack[i]);
                r_strcat(subseq1, wordstack[i]);
                p = count_up(seq);
                if(verbose)
                {
                        printf("Count up: ");
                        printseq(seq);
                        printf("¥n");
                }
```

```
                        if(i != 1)
                        {
                                q = lookup(subseq0);
                                    makelink(q, p);
                                if(verbose)
                                {
                                        printf("Link: ");
                                        printseq(subseq0);
                                        printf(" -> ");
                                        printseq(seq);
                                        printf("¥n");
                                }
                                q = lookup(subseq1);
                                makelink(q, p);
                                if(verbose)
                                {
                                        printf("Link: ");
                                        printseq(subseq1);
                                        printf(" -> ");
                                        printseq(seq);
                                        printf("¥n");
                                }
                        }
                            r_strcat(subseq0, wordstack[i]);
                }
                for(i = N_GRAM - 2; i >= 0; i--)
                        strcpy(wordstack[i+1], wordstack[i]);
        }
        fclose(infp);
}

r_strcat(s, t)
char    *s, *t;
{
/*
 * Ex. s->"abc", t->"xyz" ==> s->"xyz¥001abc"
 */
        register int    i, x, y;
```

```
        if(*s == NULL)
                strcpy(s, t);
        else
        {
                x = strlen(s);
                y = strlen(t);
                for(i = x+y+1; i >= y+1; i--)
                        s[i] = s[i-y-1];
                for(i = 0; i < y; i++)
                        s[i] = t[i];
                s[i] = DELIMIT_CHAR;
        }
}

/*
 * 'WordSeq' handling functions.
 */

unsigned hash(sequence)
unsigned char   *sequence;
{
        unsigned char   *p;
        int     val, n = 0;

        for(p = sequence; *p != NULL; p++)
        {
                if(*p == DELIMIT_CHAR)
                        n++;
        }
        for(val = 0; *sequence != NULL; )
        {
                if(*sequence == DELIMIT_CHAR)
                        break;
                val += *sequence++;
        }
        return((n - 1)*B_PRIME + val%B_PRIME);
}
```

```
struct WordSeq  *count_up(sequence)
char    *sequence;
{
        register char   *q;
        int     n = 0;
        struct WordSeq  *p, *lookup(), *install();

        for(q = sequence; *q != NULL; q++)
        {
                if(*q == DELIMIT_CHAR)
                        ++n;
        }
        totals[n] += 1;
        if((p = lookup(sequence)) != NULL)
        {
                p->Count += 1;
                return(p);
        }
        else  /* First N-gram */
        {
                d_totals[n] += 1;
                return(install(sequence));
        }
}

struct WordSeq  *lookup(sequence)
char    *sequence;
{
        struct WordSeq  *p;
        unsigned        hash();

           for(p = hashtable[hash(sequence)]; p != NULL; p = p->Next)
        {
                if(strcmp(sequence, p->Sequence) == 0)
                        return(p);
        }
        return(NULL);
}
```

```
struct WordSeq  *install(sequence)
char    *sequence;
{
        struct WordSeq  *p;
        char    *strsave(), *malloc();
        unsigned        val, hash();

        if((p = (struct WordSeq *)malloc(sizeof(struct WordSeq))) == NULL)
                error("No memory");
        if((p->Sequence = strsave(sequence)) == NULL)
                error("No memory");
        p->Count = 1;
        p->UpperLinks = NULL;
        val = hash(p->Sequence);
        p->Next = hashtable[val];
        hashtable[val] = p;
        return(p);
}

makelink(p, q)
struct WordSeq  *p, *q;
{
        register struct ConStruct       *cp1, *cp2;

        for(cp1 = p->UpperLinks; cp1 != NULL; cp1 = cp1->Next)
        {
                if(cp1->UpperLink == q)
                        return;
        }
        if((cp1 =
          (struct ConStruct *)malloc(sizeof(struct ConStruct))) == NULL)
                error("No memory");
        cp1->UpperLink = q;
        cp1->Next = NULL;
        if(p->UpperLinks == NULL)
                p->UpperLinks = cp1;
```

```
        else
        {
                for(cp2 = p->UpperLinks; cp2->Next != NULL; cp2 = cp2->Next)
                        ;
                cp2->Next = cp1;
        }
}

printseq(sequence)
char    *sequence;
{
        register char   *p;

        for(p = sequence; *p != NULL; p++)
        {
                if(*p == DELIMIT_CHAR)
                        putchar(',');
                else
                        putchar(*p);
        }
}

char    *strsave(s)
char    *s;
{
        char    *p, *malloc();

        if((p = malloc(strlen(s) + 1)) != NULL)
                strcpy(p, s);
        return(p);
}

/*
 * Write idioms.
 */

#define N_SEQS          2048
```

```
struct SelList
{
        struct WordSeq  *Wseq;
        int     K;
        int     Ngram;
        int     Modified;
}       SeqList[N_SEQS];

dump_ngram()
{
        register int    i, j, k, l;
        register struct WordSeq *p;
        int     max = 0, total = 0, count = 0;
        struct SelList  *q;

        fprintf(stderr, "[ Dumping ... ]¥n");
        printf(" Total Sentences = %5d¥n", sentences);
        printf(" Total Words     = %5d¥n", totals[0]);
        printf(" Bigrams         = %5d / %5d¥n", d_totals[1], totals[1]);
        printf(" Trigrams        = %5d / %5d¥n", d_totals[2], totals[2]);
        printf(" 4-grams         = %5d / %5d¥n", d_totals[3], totals[3]);
        printf(" 5-grams         = %5d / %5d¥n", d_totals[4], totals[4]);
        printf(" 6-grams         = %5d / %5d¥n", d_totals[5], totals[5]);
        printf(" 7-grams         = %5d / %5d¥n", d_totals[6], totals[6]);
        printf(" 8-grams         = %5d / %5d¥n", d_totals[7], totals[7]);
        printf(" 9-grams         = %5d / %5d¥n", d_totals[8], totals[8]);
        printf("10-grams         = %5d / %5d¥n", d_totals[9], totals[9]);

        for(i = 1; i < N_GRAM; i++)
            total += d_totals[i];
        printf("Total N-grams = %d¥n", total);

        for(i = 0; i < N_GRAM - 1; i++)
        {
            for(j = i*B_PRIME; j < (i+1)*B_PRIME; j++)
            {
                for(p = hashtable[j]; p != NULL; p = p->Next)
                {
                    if((p->Count)*(i+1) > max)
                        max = (p->Count)*(i+1);
```

```
                }
            }
        }
        fprintf(stderr, "Max K = %d¥n", max);

        for(k = max; k >= 1; k--)
        {
            for(i = 0; i < N_GRAM - 1; i++)
            {
                for(j = i*B_PRIME; j < (i+1)*B_PRIME; j++)
                {
                    for(p = hashtable[j]; p != NULL; p = p->Next)
                    {
                        if((p->Count)*(i+1) == k)
                        {
                            SeqList[count].Wseq = p;
                            SeqList[count].Ngram = i+2;
                            SeqList[count].K = k;
                            SeqList[count].Modified = 0;
                            for(l = 0; l < count; l++)
                                Kmodify(l, count);
                            ++count;
                            if(count > N_SEQS - 1
                            || count > total*percent/100)
                                goto EndSelect;
                        }
                    }
                }
            }
        }
  EndSelect:
#ifdef DEBUG
        for(i = 0; i < count; i++)
        {
            q = &SeqList[i];
            printf("%3d: <%d>%3d: ", q->K, q->Ngram, q->Wseq->Count);
            printseq(q->Wseq->Sequence);
            printf("¥n");
        }
#endif
```

```
        /*
         * Now start to write N-gram...
         */
        max = 0;
        for(i = 0; i < count; i++)
        {
            if(SeqList[i].K > max)
                max = SeqList[i].K;
        }
        fprintf(stderr, "Modified Max K = %d¥n", max);

        for(k = max; k >= 1; k--)
        {
            for(i = 0; i < count; i++)
            {
                if(SeqList[i].K == k)
                {
                    q = &SeqList[i];
                    printf("%3d: <%d>%3d: ", q->K, q->Ngram, q->Wseq->Count);
                    printseq(q->Wseq->Sequence);
                    printf("¥n");
                }
            }
        }
}

Kmodify(m, n)
int     m, n;
{
        register struct ConStruct       *p;
        int     upper, lower;

        upper = (SeqList[m].Ngram > SeqList[n].Ngram ? m : n);
        lower = (upper == m ? n : m);
        if(SeqList[upper].Ngram - SeqList[lower].Ngram == 1)
        {
                for(p = SeqList[lower].Wseq->UpperLinks;
                  p != NULL; p = p->Next)
                {
                        if(p->UpperLink == SeqList[upper].Wseq)
```

```
                        break;
                }
                if(p == NULL)
                        return;
                if(SeqList[lower].Modified)
                        return;
                SeqList[lower].K =
                  (SeqList[lower].Wseq->Count - SeqList[upper].Wseq->Count)
                  * (SeqList[lower].Ngram - 1);
                ++SeqList[lower].Modified;
        }
}

/*
 * Lexical analyzer.
 * Get one word from text database.
 *
 * Sample text structure.
 * ----------------------
 * (CHANGED
 *  #S(OUTPUT
 *        :O_DATA (196204 204245 197197 207195 185241 186221 178241 181196
 *                 187246 204179 182201)
 *        :O_REGULAR_HIRAGANA (164196 164166 164228 164175 164199 164243
 *                             164239 164179 164175 164181 164164 164171
 *                             164164 164174 164184 164224 164173 164231
 *                             164175)
 *        :O_REGULAR_KANJI ((196204 204245 197197 207195 185241 186221 178241
 *                           181196 187246 204179 182201))
 *        :O_HINSHI 30 :O_KATSUYOU NIL :O_KATSUYOU_TYPE NIL :O_BEFORE NIL
 *        :O_AFTER NIL) ...
 */

getword(fp, buf, bufsize)
FILE    *fp;
char    *buf;
int     bufsize;
{
        char    regform[WORDSIZE];
```

```
GetData:
        if(skip_to(fp, "O_DATA") == EOF)
                return(EOF);
        getcodelist(fp, buf, bufsize);
        skip_to(fp, "O_REGULAR_KANJI");
        getcodelist(fp, regform, WORDSIZE);
        if(strcmp(regform, "NIL") == 0)
        {
                if(buf[0] == '¥241'
                  && (buf[1] == '¥243'    /* Zenkaku Maru */
                   || buf[1] == '¥251'    /* Zenkaku ? */
                   || buf[1] == '¥252'))  /* Zenkaku ! */
                        return(EOS);
                else
                           goto GetData;
        }
        return(TRUE);
}

skip_to(fp, keyword)
FILE    *fp;
char    *keyword;
{
        char    buf[WORDSIZE];

        while(1)
        {
                switch(gettoken(fp, buf, sizeof(buf)))
                {
                case EOF:
                        return(EOF);
                case ':':
                        if(gettoken(fp, buf, sizeof(buf)) == EOF)
                                return(EOF);
                        if(strcmp(buf, keyword) != 0)
                                continue;
                        return(TRUE);
                }
        }
}
```

```
getcodelist(fp, buf, bufsize)
FILE    *fp;
char    *buf;
int     bufsize;
{
        int     level = 0, type;
        char    tmp[4], str[256], *strp;

        while((type = gettoken(fp, buf, bufsize)) == '(')
                ++level;
        if(type != NUMBER)
                return(FALSE);
        strp = str;
        while(type == NUMBER)
        {
                strncpy(tmp, &buf[0], 3);
                *strp++ = atoi(tmp);
                strncpy(tmp, &buf[3], 3);
                *strp++ = atoi(tmp);
                type = gettoken(fp, buf, bufsize);
        }
        *strp = NULL;
        if(type != ')')
        {
                error("Illegal code list¥n");
                return(FALSE);
        }
        level--;
        while(level--)
                gettoken(fp, buf, bufsize);
        strncpy(buf, str, bufsize);
        return(TRUE);
}

gettoken(fp, buf, bufsize)
FILE    *fp;
char    *buf;
int     bufsize;
{
        int     c;
```

```
        char    *bufp;

        bufp = buf;
        while(isspace(c = getc(fp)))
                ;
        *bufp++ = c;
        if(!isalpha(c) && !isdigit(c))
        {
                *bufp = NULL;
                return(c);
        }
        if(isalpha(c))
        {
                while(--bufsize)
                {
                        c = *bufp++ = getc(fp);
                        if(!isalnum(c) && c != '_' && c != '-')
                        {
                                ungetc(c, fp);
                                break;
                        }
                }
                *(bufp - 1) = NULL;
                return(STRING);
        }
        if(isdigit(c))
        {
                while(--bufsize)
                {
                        c = *bufp++ = getc(fp);
                        if(!isdigit(c))
                        {
                                ungetc(c, fp);
                                break;
                        }
                }
                *(bufp - 1) = NULL;
                return(NUMBER);
        }
}
```

```
/*
 * Error printing functions.
 */

error(fmt, args)
char    *fmt;
int     args;
{
        _doprnt(fmt, &args, stderr);
}

fatal_error(fmt, args)
char    *fmt;
int     args;
{
        _doprnt(fmt, &args, stderr);
        exit();
}
```