インターフェイスの街角 (30)– 辞書遊び

増井俊之

世の中には、クロスワードやスクラブル、しりとりなど、文字や単語を使うパズルやゲームがたくさんあります。最近は計算機上で引ける各種の辞書が入手しやすくなりましたが、これらを利用すればこのようなパズルを簡単に解けることがあります。たとえば、grep と辞書を使うとクロスワード・パズルがいとも簡単に解けてしまいます。

もともと人間の頭で考えるために作られたパズルを計算機で解くのは無粋の極みというべきかもしれませんが、問題を作成するときには計算機が重宝します。今回は、辞書を使ったお遊びプログラムをいくつか紹介しましょう。

四文字熟語クイズ

〝「○肉○食」の○を埋めよ〟という問題に〝焼肉定食〟と答えたという笑い話がありますが、このような穴埋め問題は四文字熟語の辞書があれば簡単に作れます。**four** は、このような四文字熟語クイズを自動的に生成する Perl プログラムです。

four を起動すると、自動的に生成された問題が表示されます。

```
% four
世○記○
```

答が分かった場合は、リターンを入力すると解答と次の問題が表示されます。

```
% four
世○記○
世界記録
大○生○
```

four は、四文字熟語の辞書からランダムに単語を選び、その単語のなかの 2 文字を○で置き換えているだけで、プログラム自体はごく単純なものです（図 1）。ただし、任意の 2 文字を○にするだけでは、問題があまりにやさしくなってしまうことがあります。たとえば、〝弱肉○○〟や〝○○強食〟といった問題では、解くのが簡単すぎておもしろくありません。一般に、前半と後半の連続する 2 文字は選ばないほうが難しくなるようです（図 2）。さらに、いわゆる四文字熟語以外の、通常は熟語と考えられていない単語（人名や地名など）を含めておくと、意外性が高まり、より難しくなります。

図 1　四文字熟語クイズプログラム four

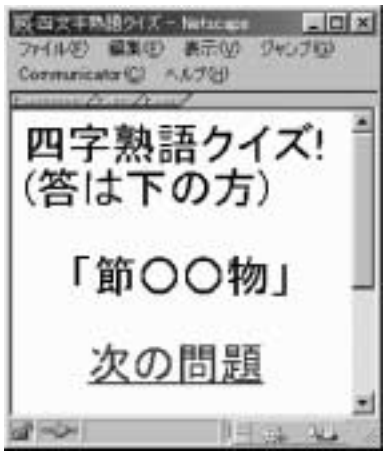
```
#!/usr/local/bin/perl
@words = (
    "合縁奇縁",
    "愛別離苦",
    "青息吐息",
    "青色申告",
    "秋雨前線",
    "悪戦苦闘",
    "悪口雑言",
      ……
);

srand(time);
$| = 1;
open(tty,"/dev/tty");
for(;;){
    $n = int(rand($#words+1));
    $a = $q = $words[$n];
    # 先頭2文字の片方を「○」に変更
    $m = int(rand(2));
    substr($q,$m*2,2) = '○';
    # 最後の2文字の片方を「○」に変更
    $m = int(rand(2));
    substr($q,$m*2+4,2) = '○';
    print "$q ";
    $_ = <tty>;
    exit if $_ eq '';
    print "$a\n";
}
```

図２　四文字熟語クイズの例（何問解けますか？）

人○○学	金○湯○
予○知○	軽○○動
○天順○	○換○件
○即○離	心○一○
○杯機○	○下○品
言○生○	○機応○
○共料○	事○無○
民○自○	罵○雑○
不○所○	○言○色

図３　i モード用四文字熟語クイズ画面



fourはこのようにきわめて単純なプログラムですが、あるとき宴席で NTT ドコモの i モードで動くようにしたもの[1]を披露したところ、何人かが i モード対応の携帯電話を持っていたためたいへんに盛り上がりました。問題を解くことや解答の速さを競うことに興じたわけではなく、難問に対して〝焼肉定食〟のような楽しい解答をする人が続出したためです。

四文字熟語クイズを解こうとすると、心の奥底にある考えが思わず解答に反映されてしまうことがあるようです。以前、〝○情騒○〟という問題を出したところ、〝痴情騒動〟という答が返ってきたことがありました（できすぎた話のように思うかもしれませんが、実話です）。この手の事情が好きだと公言する人はあまりいないと思いますが、クイズのおかげでその人の深層心理 (?) が図らずも明らかになってしまったようです（このクイズの正解は〝物情騒然〟です）。

たしかに、お腹が空いているときに〝○肉○食〟という問題に対して〝焼肉定食〟と答えても不思議ではありません。その意味では〝四文字熟語は心の鏡〟ともいえますから、うまくすれば心理テストに応用できるのではないでしょうか。

1 http://tracer.csl.sony.co.jp/four/four.cgi

地口・駄洒落作成支援システム

- 財テク勤務
- エコノミ焼きで倹約しよう
- サミットが沖縄れます

といった地口・駄洒落を自動的に作るにはどうすればよいでしょうか。以前、雑誌「AERA」の中吊り広告にこうした腰のくだけるような駄洒落が載っていましたが、これらを総称して〝AERA 惹句〟[2]と呼ぶのだそうです。さきほどの四文字熟語クイズと同様、地口や駄洒落などの言葉遊びも自分の頭で考えるからこそおもしろいのかもしれません。しかし、ここではあえて計算機で自動的に生成させることを考えてみましょう。

自然言語処理研究の一環として、駄洒落の解析や自動生成の研究がおこなわれています。郵政省の通信総合研究所に在籍していた滝澤 修さんは、駄洒落に関する研究で博士号を取得された[3]そうで、その研究成果を応用してアニモから「だじゃれどりアララ」[4]という製品が販売されていました。

意味まで考慮して思わず笑ってしまうような駄洒落を自動生成するのは難しいでしょうが、日本語の場合は単語の読みと文字を調べればある程度は受けそうなものが作れそうです。たとえば、〝財テク〟と〝在宅〟を比較すると、もちろん文字はまったく違いますが、読みはよく似ているので候補になるかもしれないというぐあいです。

英語の場合はそう簡単にはいきません。たとえば、anecdote（逸話）と antidote（解毒剤）は、読みは似ているのに意味はまったく違うので駄洒落のネタになります。しかし、stalagmite（鍾乳洞などの石筍）と stalactite（鍾乳石）は、読みだけではなく意味も似ているので候補にはできません。字面は似ているのに意味の違う単語をみつけ

2 http://www.t3.rim.or.jp/~s-muraka/aera/aeracon.html
3 http://www.crl.go.jp/ks/ks521/taki/index-J.html
4 http://www.animo.co.jp/arara_news.htm
残念ながら、すでに販売は終了しているようです。この製品名は、やはり「AERA」に触発されたものなのでしょうか。

図 4　ローマ字辞書 (romadic)

```
  ……
zasshi 雑誌
zaitaku 在宅
zaiteku 財テク
zyouhou 情報
  ……
```

るには、どうしても意味まで含む辞書が必要になるでしょう。

以下に紹介するのは、「AERA」に負けないお洒落な駄洒落を半自動で作成するための Perl プログラム **aera** です。

まず、図 4 のようなローマ字辞書 romadic を用意します。ご覧のように、これはローマ字による読みと単語を空白文字で区切っただけの単純な形式です。この情報をもとに、駄洒落の素材になる〝文字は違うが読みは近い単語〟を探します。

そして、この辞書をもとに図 5 の駄洒落ネタ検索プログラム aera を使って駄洒落の候補単語を検索します。単語を順にスキャンしていき、読みは似ていても字面はまったく異なる単語を探して出力します。読みが似ているかどうかを判定するために、以前にも何回か紹介した曖昧検索プログラム **agrep** を用いて読みのマッチングをおこなっています。

それでは、aera の出力（図 6）をもとに駄洒落を作ってみましょう。

〝適応の帝王〟〝恐竜の供給〟〝教養の共有〟……。おもしろい駄洒落がいくらでも作れることが分かります。〝焼鳥のやりとり〟なんて、ほとんど笑い死にそうでです。

え、全然おもしろくない? それは、きっとあなたの駄洒落のセンスか、左脳の歓喜中枢の刺激が足りないのでしょう。aera でセンスを磨いてから徹夜で仕事をすれば、きっとおもしろさが分かっていただけると思います。

Nitamogy

日本語には似た文字や同音異義語がたくさんあるので、文章を書いているときに思わぬ間違いをすることがよくあります。これを逆用し、わざと似た文字を効果的に使う技術を総称してニタモジー (Nitamogy) といいます[5]。たとえば、〝四十四〟を〝四十匹〟と書くようなものです（この書換えが効果的かどうかは、とりあえず深く考えないことにします）。

Nitamogy にはまじめな応用も考えられます。情報の存在を他者に知られることなく別の情報のなかに隠す技術をステガノグラフィ (steganography) といいます（こちらは本当にあります）。ステガノグラフィに関しては、

5 もちろん私が言っているだけです。

図 5　駄洒落ネタ検索プログラム aera

```
#!/usr/local/bin/perl
$dic = "romadic";
open(dic,$dic) || die "Can't open $dic";
while(<dic>){
  chop;
  ($roma,$word) = split;
  # 短い単語は除く
  $len = length($word);
  next if $len < 4;
  # 母音を4個以上含まないものを除く
  $v = "[aiueo]";
  next unless /$v.*$v.*$v.*$v/;
  @cands = ();
  # 曖昧grepで読みの近い単語を検索
  open(agrep,"agrep -1 '^$roma ' $dic |");
  while(<agrep>){
    chop;
    ($roma,$cand) = split;
    next if $cand =~ /^(\xa4.)+$/;
    # 同じ文字を含むものは除く
    $match = 0;
    for($s=0;$s<$len && !$match;$s+=2){
      $pat = substr($word,$s,2);
      $match = 1 if $cand =~ /$pat/;
    }
    push(@cands,$cand) unless $match;
  }
  close(agrep);
  if($#cands >= 1 && $#cands < 6){
    print "$word -- ",join(", ",@cands),"\n";
  }
}
```

図 6　aera の出力

```
% ./aera
作成 -- 学生, 惑星, 策定, 策士
適応 -- 鉄鋼, 帝王
開発 -- 改札, 海抜
方式 -- 常識, 標識
複数 -- 副手, 復習, 服装, 復する, 腹痛, 復讐
登録 -- 東北, 当落, 灯篭
共有 -- 供給, 教養, 恐竜, 旧友, 強風, 強要
やりとり -- 焼鳥, 焼き鳥
  ……
```

図 7　似た文字辞書 nitadic

へ	へ	千	チ
ベ	ベ	工	エ
ペ	ペ	二	ニ
ソ	ン	人	入
ぬ	め	土	士
わ	れ	四	匹
ろ	る	従	生
え	之	踊	桶
よ	お	縞	稿
力	カ	綱	網
タ	夕	……	

StegoArchive.Com[6]に数多くの情報が公開されています。最近は、StegFS (A Steganographic File System for Linux) というものまで考案されています[7]。これは、ディスクを普通にマウントするとたんなる ext2 ファイルシステムにみえ、特殊なマウント方法を使うと隠してあるデータもみえるようになる、という手法だそうです。

情報を他者に知られたくない場合、一般には暗号化をおこないます。しかし、そもそも情報があるかどうかさえ定かでなければ、さらに安全性を高めることができます。そこで、表面的には画像や文章にみえる情報のなかに、こっそり秘密の情報を隠すためのステガノグラフィ技術がいろいろと考えられています。いわゆる〝電子透かし技術 (watermarking)〟もこの技術の応用の 1 つです。隠したい情報をエンコードし、Nitamogy によって文字を置き換えておけば、一見普通の文章に秘密の情報を混ぜておくことができます[8]。

Nitamogy を使うには、まず〝似た文字辞書 (nitadic)〟を用意し、形の似た 2 文字の組を一覧表にしておきます (図 7)。

この辞書を作成するとき、フォントのビットマップを比較する方法も試してみました。しかし、うまくいく場合もあるとはいえ、〝王〟と〝田〟、〝十〟と〝未〟などが似ていると判定されることがあります。やはり、目で見ながら地道に作るほうが質のよいものができるようです。

Perl プログラム **nitaconv** (図 8) では、nitadic を用いて標準入力に Nitamogy を適用することができます。

6 http://steganography.tripod.com/stego.html
7 http://ban.joh.cam.ac.uk/~adm36/StegFS/
8 あまりに置換え文字が多いと不審に思われるので、多量の情報のエンコードには向いていません。

図 8　似た文字変換プログラム nitaconv

```
#!/usr/local/bin/perl

# nitadicを使って似た文字データベースを作成
open(nitadic,"nitadic");
while(<nitadic>){
  chop;
  ($w1,$w2) = split;
  $list{$w1} .= $w2;
  $list{$w2} .= $w1;
}
close(nitadic);

srand(time);
while(<>){
  chop;
  $out = '';
  while($_ ne ''){
    if(s/^[\x80-\xff]//){
      $w = $&;
      $l = $list{$w};
      $len = length($l)/2;
      if($len){
        $r = int(rand($len));
        $w = substr($l,$r*2,2);
      }
      $out .= $w;
    }
    else {
      s/^.//;
      $out .= $&;
    }
  }
  print "$out\n";
}
```

簡単な実行例を以下に示します。

```
% echo 人口問題 | ./nitaconv
入口問題
%
```

おわりに

今回は、辞書を使ったお遊びシステムをいくつか紹介しました。辞書はデータベースそのものですから、さまざまな用途に使えるはずです。しかし、現在のところ、意外でありながら効果的な活用例はあまり多くないようです。発想を転換すれば、まだまだおもしろい応用が考えられるのではないでしょうか。

（ますい・としゆき　ソニー CSL）