文章中の文字列を検索する

白井 英俊

2009 年 10 月 27 日

1 KWIC とコーパス

コーパス (corpus、複数形は corpora)—体系的に集められた文章のサンプルで、特に電子化されたもの。

- 生 (raw) コーパス: 手が加えられていないコーパス
- タグつき (tagged) コーパス: 単語に分割され、 品詞や意味情報などが付け加えられたコーパス
- 現代の辞書の作成法 — コーパスにおける語の用法に基づく
  例: Longman Dictionary of American English, Cambridge Grammar of English
- KWIC (Key Word In Context) コンコーダンス — 調査対象の単語がどのような語や句といっしょに現れているか調べるために、それをコーパスから抽出して表にしたもの

```
Ghostly little book, to raise the Ghost of an Idea, which shall
selves, with each other, with the season, or with me.  May it h
 I: Marley's Ghost Stave  II: The First of the Three Spirits St
                     First of the Three Spirits Stave III: The
    Three Spirits Stave III: The Second of the Three Spirits S
                    Second of the Three Spirits Stave  IV: The
    Three Spirits Stave  IV: The Last of the Spirits Stave   V
                      Last of the Spirits Stave   V: The End of
          Spirits Stave   V: The End of It
```

このようなコンコーダンスをつくることを考えよう。そのために次節で正規表現について学ぶ。

2 正規表現 (1) — 猫 (cat) を探せ

- 講義用 Web ページ (`www.cyber.sist.chukyo-u.ac.jp/sirai/classes/semi2/index.html`) に Edgar Alan Poe の "The Black Cat"(黒猫) のテキストが置いてある。これを対象に cat が現れている文を探してみよう。
  そのために使うパタンマッチング (パタン照合、指定された文字列を見つけること) の一番の方法が「正規表現」(regular expression, 略して regex)
  Ruby において、正規表現の一つの書き方は、文字列をスラッシュ (/) でくくること。だから、

  ```
  /cat/
  ```

  という正規表現は、テキスト中の cat にマッチしてくれるはず...
  実際に、The Black Cat で試してみよう。以下は、講義用のウェブにおいたプログラム findingCat1.rb のリスト：

```
#! /usr/bin/ruby
# USAGE: ruby findingCat1.rb FileName.txt (range)
if (ARGV[0] != nil)
  file = ARGV[0]      # ファイルの名前を記録
else
  puts "USAGE: findingCat1.rb FileName.txt"
  exit
end

if (ARGV[1] != nil)
     num = ARGV[1].to_i    # 表示する行数
else
     num = 20              # 指定がなければ２０に
end

inf = open(file,"r")  # ファイルを開く
while (line = inf.gets)  # 一行ずつ読み込み
    break if (num == 0)    # num = 0 なら終了
    if (line =~ /cat/)   # cat と照合する
      puts line          # cat があれば、その行を表示する
      num -= 1     # 表示行の countdown
    end
end
inf.close      # ファイルを閉じる
```

簡単に説明する。USAGE は「使用法」ということ。**プログラムの先頭にプログラムの使い方の簡単な説明を書く**ことをお勧めする (古いプログラムを発見したようなときに便利)。

ARGV は、プログラムをコマンド行として実行したとき (つまり、いわゆる DOS 窓から行を打ち込み、プログラムを実行した場合) に、プログラムの名前以降の引数がその要素として与えられる配列。「用法」から察せられるように、探索対象のファイル名と、オプションで (つまり、無くとも良い) 表示すべき行数を表す数が与えられた場合、それぞれが ARGV[0] と ARGV[1] に与えられる。

真ん中ほどの inf = open(file,"r") が、ファイルを読み込む準備をしているところ。while 文で一行ずつ読み込み、line に代入。その二行下の if 文の条件、

```
line =~ /cat/
```

で、cat という文字列が line の値になっている文字列中にあるかどうかを調べている。それがある場合は、nil 以外のものが値となる。

問題 1: line 中にその文字列があった場合、どんな値が返されるか?

問題 2: 実際にこのプログラムを The Black Cat で試して、その結果を調べてみよう。どこに cat がいるだろうか？また、この結果はあなたが期待したものと同じだろうか？

3　正規表現 (2) — 猫 (cat) だけを探せ

プログラム findingCat1.rb は、次のような結果を表示したはず:

```
intensity of the gratification thus derivable. There is something in the
had birds, gold-fish, a fine dog, rabbits, a small monkey, and _a cat_.
black cats as witches in disguise. Not that she was ever _serious_ upon
Pluto--this was the cat's name--was my favorite pet and playmate. I
(以下略)
```

ここには猫もいるが、そうでないものもある。これはどうしたことか？ Ruby が間違ったのだろうか？

いや、Ruby は間違っていない。それぞれの行にはちゃんと、**cat** が隠れている。しかし、それは予想していなかったものもあった、ということ (たとえば、gratifi**cat**ion)。

どのような結果が欲しいかを正確に正規表現として表す。それには cat が隠れていそうなパタンを (できるだけ多く) 考えて、それらを表す正規表現を選ぶ (作り出す) ことが必要となる。

最初の試みとして、特殊パタン (\b) を使うことを考えよう。\b は、「単語の境界」にマッチする「アンカー」の一つです。これ以外のアンカーとして、^と$があります。

プログラム findingCat2.rb は、プログラム findingCat1.rb の cat を \bcat\b で置き換えたもの。これを走らせて、前の結果と比べてみよう。

改善されただろうか？新たな問題が起っていないだろうか？

注意: 正規表現で用いられる文字列のほとんどは、その文字「自体」を表すが、以下のものは例外で、特殊な意味があったり、他の文字と組み合わされて特別なパタンを表す。どのような特殊なパタンがあるかを知ることが、正規表現の習得の道。

```
.|()[]{}+\^$*?
```

問題 3: ^と$は何にマッチするか、調べよ。また、実際に試してみよう。

問題 4: 多くの特殊パタンは \ から始まる。たとえば、\w, \W, \s, \S, \d, \D がそうである。これらは何とマッチするか、調べよ。

4　正規表現 (3) — 猫たち (cats) も探せ

プログラム findingCat2.rb はかなりの改善になった、少なくとも

```
intensity of the gratification thus derivable. There is something in the
```

という表示はなくなった。しかし、注意深くみると、新たな問題が生じていることもわかる。それは、二つのプログラムの結果を注意深く比べてみればわかる。つまり、次のようなものが「見つけられなくなった」のだ。

```
had birds, gold-fish, a fine dog, rabbits, a small monkey, and _a cat_.
black cats as witches in disguise. Not that she was ever _serious_ upon
```

これはもちろんパタンとして、\bcat\b を指定したためである。最初の例は、cat の後のアンダスコア (下線、_) が\b とマッチしないため (つまり、正規表現では、アンダスコアも単語の一部とみなされる) である。二番の例は、cats という cat の複数形だから (これも最後の s は単語の一部) だからである。

これらを表示させるために、新たな道具を紹介する。それは、「パタンの選択」を表す | と、文字の選択を表す [ ] である。

- 「パタンの選択」を利用した解:

  ```
  \b(cat|cats|cat_|cats_)\b
  ```

  これは、括弧内の四つのパタンのどれでも、単語境界 (\b) をはさんで現れる場合にマッチするパタンを表わしている。このように、| はパタンの「選択」を表わす。
  しかし、よくみると cat が何回も現れており、整理すればもうちょっと簡単なパタンになりそうである。つまり、

  \bcat に続けて、 (s か _ か s_ の後に\b) もしくは、直接\b が続く

  というパタンが書ければ簡単になりそう。そのアイデアを取り入れたものが以下。

  ```
  \bcat(|s|_|s_)\b
  ```

  さらに、もうちょっと工夫して、次のようにしてもよさそう:

  ```
  \bcat(|s)(|_)\b
  ```

  ここで、(|s) や (|_) というのは、s や_ が一個現れるか、もしくはまったく現れない、というパタンを表わしている。これを「0 回か 1 回の繰り返し」のパタンと呼び、?を用いて表わすことができる。つまり、s?や_?とも書ける。この表記を用いた解は以下:

  ```
  \bcats?_?\b
  ```

  問題 5: それぞれのパタンを使ったプログラムを書いて、どのような出力をするか、確かめよ。

- 「文字の選択」を利用した解:
  [と] で文字列を囲むと、その文字列に現れた一字にマッチする正規表現となる。たとえば、[abcdef] は、a から f までの一文字にマッチする。ここでこのような連続する文字の場合、- を使うことができる。たとえば先の例は、[a-f] と書ける。したがって、[a-zA-Z] は大文字でも小文字でもアルファベット一文字にマッチするパタンを表す。
  これを使えば、大文字小文字問わず書かれた cat にマッチするパタンを次のように表すことができる:

  ```
  [cC][aA][tT]
  ```

  このアイデアと先のパタンの書き方を合わせると次のようなパタンが書ける:

  ```
  \b[cC][aA][tT][sS]?_?\b
  ```

  問題 6: このパタンを使ったプログラムを書いて、それがどのような出力をするか、確かめよ。
  参考: 正規表現が大文字小文字問わずマッチするようにするには、次のように書いてもよい。

  ```
  if (line =~ /\bcats?_?\b/i)
  ```

5　正規表現 (4)—何でもかんでも探せ

次のプログラムは、findingCat1.rb を修正し、第一引数にファイル名、第二引数に正規表現をとり、その正規表現にマッチする文字列があれば、その行を表示するプログラムである。

```
#! /usr/bin/ruby
# USAGE: ruby findingPat.rb FileName.txt regex
if (ARGV[0] != nil)
  file = ARGV[0]      # ファイルの名前を記録
else
  puts "USAGE: findingPat.rb FileName.txt regex"
  exit
end
if (ARGV[1] != nil)
     pat = Regexp.new(ARGV[1])      # 正規表現
else
  puts "USAGE: findingPat.rb FileName.txt regex"
  exit
end
inf = open(file,"r")  # ファイルを開く
while (line = inf.gets)  # 一行ずつ読み込み
    if (line =~ pat)   # pat（正規表現）と照合する
      puts line          # cat があれば、その行を表示する
    end
end
inf.close      # ファイルを閉じる
```

問題 7: 正規表現では「繰り返し」を表わす特殊パタンがある。その中で、`+` と `*` と `{n}` (ここで、n は正の整数) がどのようなパタンであるかを調べよ。また、上のプログラムを用いて、The Black Cat において、母音が 3 個以上連続する単語が含まれる行を表示せよ。

問題 8: `[ ]` は、角括弧の中の文字列の任意の一文字とマッチすると述べたが、「角括弧の中の文字列**以外**の任意の文字とマッチする」パタンも書ける。それは、`[^aeiou]` のように、`[`の直後に`^`を書くことで表わされる。これを用いて、The Black Cat において、子音が 3 個以上連続する単語が含まれる行を表示せよ。

問題 8: `.` は、空白も含む任意の一文字にマッチするパタンである。(ただし、改行にはマッチしない—マッチさせることもできるが、それには「オプション」が必要)。これを用いて、The Black Cat において、q の後一文字おいて a か e か o が来る単語を見つけよ。

発展問題 (Ruby の正規表現の説明書が必要): いままでは「これこれという単語を含む行」を表示するだけだった。これを、「これこれという条件を満たす単語**だけ**」表示するように、findingPat.rb を作り直せ。