情報処理 II 第 8 回

情報の電子化 (3) 日本語文書における検索

桂田（かつらだ） 祐史（まさし）

2001 年 6 月 14 日

日本語文書処理の問題、特に日本語文書における検索について説明する。

1 日本語文書における検索

複数の文書の中から、指定した文字列を検索するというのは、最もコンピューターらしい仕事の一つと言えるだろう。

1.1 まずは grep を使ってみよう

例 1. 「電子メイルはもっとも重要な個人向けデータベースである」と言う人がいる。利用している本人にその意識がなくとも、色々な人との連絡を電子メイルを用いて行なっていると、知らないうちに重要なデータが集まって来て、大変便利なデータベースになる[1]。例えば手帳への書き込みと異なり、何時、誰からの or 誰への、という基本的なことが必ずついている。またバックアップを取ることも簡単である。そして、最も大事なことは、検索が容易に出来ることである。 — 例えば、筆者はメイラーとして mh-e を使っていて、ある知人からのメイルを `Mail/terada` というディレクトリィに保存してあるが、その中から「水戸」という文字列を含むファイルを検索するには

これは桂田にしか意味がない

```
oyabun% grep 水戸 Mail/terada/*
```

(総容量 3MB, 個数にして 1000 個程度のファイルの検索が 10 秒未満)

とすればよい (実はこれは最終的な解決法ではないのだが、それについては、おいおい分かる)。

このように UNIX の場合は、ファイルの中から文字列を検索するために、普通 **grep** (とその兄弟である egrep) が利用される。これは電子メイルに限らず、テキスト・ファイルならば何でも有効である。例えばソース・プログラム、TeX のソース (`*.tex` ファイル) など何でも検索できる。

[1]野口悠紀男氏は、著作の中で文書作成のソフトウェアとして 5 つのタイプのものをあげているが、その 5 番目は「メーラ」となっている。「メーラをあえて文書作成ソフトとして評価対象としたのは、文書管理機能が優れているためである。(中略) 管理機能の点でいえば、他の四つのタイプのソフトウェアに比べて、遜色のないものである。」

これも桂田にしか意味がない

```
oyabun% grep 関数解析 math/*.tex math/*/*.tex
oyabun% grep trid Sotsuken/*/*.c Sotsuken/*/*/*.c
oyabun% grep malloc /usr/include/*.h
```

1.2 grep の問題点

しかし、 grep (正確には日本語対応 grep) は日本語の文書に用いるには、今一つ不十分なところがある。大きな問題点を二つほど説明しよう。

(1) grep は日本語の「形態素」を認識しない オンライン・マニュアルによると grep は「パターンにマッチする行を表示する」とある。要するに行単位で検索をしているわけである。英語の場合はこれで特に問題は生じないのだが、日本語の場合は、「単語」の途中で行が変わることがあるため、 grep では検索されないことがある。つまり

```
弁慶がな
ぎなたを
```

というテキスト・ファイルがあるとき、「なぎなた」という単語を grep で探し出すことはできない。英語の場合は、行の中にびっしり文字を詰めなくても構わないので、適当な単語の終りで改行するようにのが普通である。つまり (極端な例ではあるが)

```
I am
Katsurada.
```

のようにしてしまうわけである[2]。このため、日本語でテキスト・ファイルを作成する場合に、画面の右端近くに来ても改行をせずに、段落の変わり目でだけ、改行 0x0a を入れるという流儀の人がいる (後の青空文書の実例を見よ)。そのため、日本語テキスト・エディターには、画面上での行と、「改行 0x0a」で区切られた行、という二つの「行概念」を持ち込み、後者の行 (行と言うよりは段落) については、非常に長いものを作成可能にしたものがある (不自然な苦し紛れをやっているのであって、ここで説明していることが分からなくてもよい)。

以上のことと少し関係するが[3]、英語では単語の境界が空白というもので明らか (機械的に判明する) であるが、日本語ではそのようになっていない。日本語の文章を「形態素[4]」に分解することは、英文ほど単純にはできない。

例えば (厳密には単語への分解とは違うが)、前回紹介した

[2]きちんとした印刷文書の場合には、単語の途中でハイフンを入れて次の行に続ける — この作業を hyphenation と呼ぶ — とか、単語間の空白を調整してちょうど行の終りに単語の終りが来るようにする、などの工夫をするが、タイプライターなどで作成する文書では、行の終りが近づいたら、適当な単語の終りで改行してしまう。英文のテキスト・ファイルはこの「作法」に則って作成されるのが普通である。例えば前回の実験に使った Gutenberg テキストを見よ。

[3]英語の文章では行の終りは、単語間の空白のように単語の切目を意味することが多いが (例外は hyphenation してある場合)、日本語の文章では行の終りは、単語の切目を意味しない。

[4]意味を有する最小の言語形態のことを形態素という。英語で言う morpheme.

```
cat alice29.txt | /usr/ucb/tr -cs A-Za-z '\012'
```

のようなことを、日本語のテキスト・ファイルについて行なうにはどうすれば良いか考えてみて欲しい。機械的にやることは不可能で、少なくとも日本語にどういう単語があるか知ることが必要で、またそれだけでは十分でないことが分かるであろう。

(2) 日本語には複数の文字コードがある 一つのコンピューター・システムで、普通どの文字コードを利用するかは決められていることが多いが、実際の運用では完全に統一されているわけではない。例えば、情報科学センターのワークステーションでは、日本語文書はファイルに格納する際には日本語 EUC を使うのが普通であり、システムに標準で備わっている grep 等のコマンドは日本語 EUC に対応しているが、電子メイルは (既に述べたように) ISO-2022JP (いわゆる「JIS 漢字」) という文字コードで送受信されていて、システムのメール・スプールにあるファイル (`/var/mail/` ユーザー名 というファイル — これは受信したままの状態でコードの変換等は一切行われていない) に対して grep をかけてもまともな検索はできない。さらに、これは結構深刻なことだが、情報科学センターにインストールされている日本語 MH の設定では、メイルの文字コードを ISO-2022JP のままで保存する。だから、情報科学センターのシステムをデフォールトの設定のままで使っていると

多分うまく検索できない

```
waltz21% grep 日本語文字列 Mail/inbox/*
```

のような検索 (日本語文字列としては自分の名字などを試してみるとよい) は、実はうまく行かない。上の例 (oyabun で桂田が「水戸」を検索した) が一見うまく行ったのは、実はメールを日本語 EUC に変換して保存するようなしかけを (一時期) していたことがあったからである。

1.3 どうすればいいか？

このうち文字コードの問題だけは比較的簡単に解決できる[5]。日本語 EUC でも ISO-2022JP でも、日本語データとしては同じと見なすような grep を作れば良い。例えば成田多良氏の作成した **lgrep** (`http://www.ff.iij4u.or.jp/~nrt/lv/` から入手した) は、この条件 (実はそれ以上の条件を満たしている素晴らしいソフトウェアである) を満たす。

[5] nkf, less, mule などのコマンドも、日本語テキスト・ファイルの文字コードを自動判別して適切に処理するようになっている。

grep は EUC しか検索できない, lgrep は何でも OK

```
waltz21% source ~re00018/syori2rc     ← 桂田の用意したコマンドを利用できるようにする[a]。
waltz21% cd ~re00018/nihongo-text     ← サンプル・データのあるディレクトリィに移動する。
waltz21% ls
euc.txt   jis.txt   sjis.txt          → 3 つテキスト・ファイルがある。
waltz21% nkc *                        ← 各ファイルの文字コードを調べる (結果は省略する)
waltz21% cat euc.txt
桂田祐史
弁慶がな
ぎなたを
waltz21% grep 桂田 *
euc.txt:桂田祐史                      → EUC のテキストしか検索できていない。
waltz21% lgrep 桂田 *
euc.txt:桂田祐史
jis.txt:桂田祐史
sjis.txt:桂田祐史                     → lgrep なら三つとも検索に成功する。
waltz21% grep なぎなた *              ← これはうまく行かない。
```

[a] lgrep を使いたい人は、.cshrc にこのコマンドを書いておくと良い。

しかし、たとえ、そういう (文字コードの問題を解決した) grep を作ったとしても、(1) の問題は残ったままである。これは日本語の文書を処理するには、grep という made in USA の (行単位で検索する) ソフトはもうあきらめて、日本語文書のためのソフトを作るべきだ、ということだろう (少し飛躍のある主張？)。

1.4 Namazu の紹介

実は、当初 WWW の検索エンジンとして開発された **Namazu**[6] という日本語全文検索システムが、今ではかなり汎用目的に使えるように改良されていて、多くの検索用途にかなり手軽かつ便利に使える。それを紹介しておこう。

例を二つほどあげる。

(i) Namazu 本来 (元来) の使い方の例として、数学科の WWW ページの検索用ページ

`http://www.math.meiji.ac.jp/cgi-bin/namazu.cgi`

をあげておく (あまり手入れはしていません...)。

(ii) 桂田個人の使用例だが、MH のメイル・ボックス `~/Mail` 内の保存メッセージを Namazu を使って検索できるようにしてある。

oyabun 上のユーザー mk の環境で — 桂田以外は試せません

```
oyabun% namazu 千葉 .Mail
```

Mail の下にあるファイル (ほとんど全ては MH によるメイル・メッセージ) の容量は約 100 MB 程度で (結構多い)、日本語 EUC と ISO-2022JP の二つのコードのファイルが混在しているが、瞬時にほぼ完全な検索ができる。

[6] Namazu について知りたければ、ホームページ `http://www.namazu.org/` を見よ。なお、情報処理 II のページにもいくつか説明を用意してある。

Namazu を利用するには、事前にインデックス (index, 索引) を作る作業がいるので、一度だけちょっと調べたくなったような用途には向かないが、その分、高速に検索ができるし、何よりも (1), (2) の問題をクリアしていて、かなり満足の行く (漏れのない) 検索ができる。

(1), (2) の問題をクリアしていることの確認 — これは誰でも試せます

```
waltz21% cd ~re00018
waltz21% namazu 桂田 index         ← この結果は見てのお楽しみ。
waltz21% namazu なぎなた index     ← 同上。
```

この index は索引ファイル (インデックス・ファイル) を納めてあるディレクトリィで、mkdir index; mknmz -O index nihongo-text として作成した。

1.5 脱線: KAKASI と ChaSen

Namazu が上の二つの問題をクリアしている仕組みを簡単に説明する。

- Namazu が (1)『形態素の区切りの問題』をクリアしているのは、内部で KAKASI または ChaSen (いずれも、すぐ後で説明) を呼び出すことにより、文書を形態素に分解しているためである。
- Namazu が (2)『複数の文字コードの問題』をクリアしているのは、内部で nkf (これは既に何度か登場した) を呼び出すことにより、文字コードを自動判別＆変換しているためである。

KAKASI (http://kakasi.namazu.org/) は元々、普通の日本語の文書 (仮名と漢字が混在している) を入力として受取り、それを仮名、またはローマ字の文書に変換するためのソフトウェアであるが、内部で文章を形態素に分解する (ことに相当する) 操作をしていることに注目され、「分かち書き」をするように拡張され、それが Namazu でも利用されるようになった。

これは試せます

```
waltz21% source ~re00018/syori2rc     ← 既に一度してあったら省略可能。
waltz21% cat ファイル名 | hiragana    ← 平仮名への変換 (結果省略)
waltz21% cat ファイル名 | romaji      ← ローマ字への変換 (結果省略)
waltz21% cat ファイル名 | kakasi -w   ← 分かち書き (結果省略)
```

(~re00018/syori2/bin/romaji, ~re00018/syori2/bin/hiragana はシェルスクリプトである。cat 等で中身を見ると、kakasi を呼び出していることが分かる。)

最近では茶筌 (http://cactus.aist-nara.ac.jp/lab/nlt/chasen.html) という「日本語形態素解析器」を使うこともある (例えば情報科学センターにインストールされている Namazu は、デフォルトでは kakasi ではなく ChaSen を用いているらしい)。

蛇足　KAKASI, ChaSen, Namazu, nkf, さらには Apache (明治大学ホームページ、明治大学数学科ホームページ等でも採用されている、定番 WWW サーバー・ソフトウェア) も、ソース・プログラムが無償で公開されているフリー・ソフトウェアである。

A　日本語のテキスト・ファイルの例

青空文庫 http://www.aozora.gr.jp/ から、著作権フリーの文書が入手できる。例えば芥川龍之介「蜘蛛の糸」を読んでみよう。

```
waltz21% mkdir kumonoito
waltz21% cd kumonoito
waltz21% cp ~re00018/kumonoito.zip .          ← コピーする。
waltz21% unzip kumonoito.zip                  ← 圧縮されているので復元する。
waltz21% ls
waltz21% ~re00018/bin/nkc kumonoito.txt       ← 文字コードをチェックする。
waltz21% qkc -eu kumonoito.txt                ← UNIX 形式 (日本語 EUC, 0x0a で改行) に変換する。
waltz21% cat kumonoito.txt
waltz21% mule kumonoito.txt &
```

二つのことに気が付くと思う。

- 改行を滅多に入れずに「長い行」(実質的には段落？) を作ってある。
- 漢字によっては (実は JIS の第一水準、第二水準に含まれていないので電子化しようがなくて) 偏(へん)と旁(つくり)を指定することで表わしてある。

B　日本語文書を電子化する上での問題点について

上の話を聴いた人は、なぜ日本語には複数の文字コードがあるのか、また何故一つに統一できないのか、不思議に思うかも知れない。これについては長い話がある。

日本語の文字コードの問題について論じた文書は書籍、雑誌記事、 WWW ページなどで色々あるが、なぜ複数の文字コードが存在するかについては、

加藤 弘一、電脳社会の日本語、文春新書 (2000).

をあげておこう[7]。

そのほか、

- 小形克宏の「文字の海、ビットの舟」
  http://www.watch.impress.co.jp/internet/www/column/ogata/
- 「日本語フォントや文字コードについて」
  http://www.zukeran.org/shin/jdoc/

C　検索あれこれ

WWW の検索エンジンはお馴染であろうが (と言うよりも説明済みなので略する)、以下、利用しているコンピューター・システム内での「検索」について。

[7]なお、この著者のホームページ http://www.horagai.com/ にも色々載っている。

編集中の文書内での文字列検索 テキスト・エディターには編集中の文書の中の文字列を検索する機能がある。 mule の場合、

C-s (incremental search forward)	文字列の検索 (ファイルの末尾に向かって) 次の候補へ行くには C-s を打つ。 検索をやめるには ESC
C-r (incremental search backward)	文字列の検索 (ファイルの先頭に向かって) C-s とは検索の方向が反対
C-s C-k	日本語の検索
C-r C-k	日本語の検索
M-x search-forward-regexp	正規表現を用いた検索
M-x search-backward-regexp	正規表現を用いた検索

大抵はそれと同時に「文字列の置換」という機能もある。

```
M-%
M-x replace-string
M-x replace-regexp
```

☆脱線になるが、時々「指定した行番号の行に移動するには？」と尋ねられる。これには M-x goto-line とすれば良い。ちなみに私は .emacs に

```
(global-set-key "\C-cg" 'goto-line)
```

と書いてあるので、 C-c g とするだけでこの機能が呼び出せる。

(複数個の) テキスト・ファイルの中の文字列検索 grep が使える。本文を参照のこと。基本的な使い方は

grep 正規表現 ファイル名リスト

このコマンドは非常に強力であり、是非ともマスターして利用すべきである。しかし、日本語文書に関しては、今一なところがある。

コマンドのありかを探せ 普段使っているコマンド、名前は分かっていても本体は一体どこにある？ which いうコマンドを使うと良い。

```
waltz21% which コマンド名
```

ただし大きなコマンドの代理人のようなコマンドであることも多く、ご本尊を見つけるには今一段のおっかけが必要になることもある。

普段使っている netscape、その本体のサイズは？

```
waltz21% which netscape.v47j
/usr/meiji/X11/bin/netscape.v47j
waltz21% ls -l /usr/meiji/pub/bin/netscape.v47j
waltz21% file /usr/meiji/pub/bin/netscape.v47j
/usr/meiji/X11/bin/netscape.v47j:  実行可能 shell スクリプト
waltz21% less /usr/meiji/X11/bin/netscape.v47j
  → 中身を読んで本体が /usr/meiji/X11/netscape.v47j/netscape と分かる
waltz21% file /usr/meiji/X11/netscape.v47j/netscape
/usr/meiji/X11/netscape.v47j/netscape:  ELF 32- ビット MSB 実行可能形式..
  → 確かに機械語実行ファイル
waltz21% ls -l /usr/meiji/X11/netscape.v47j/netscape
  さて、サイズはどの程度？
```

使えそうなコマンドを探せ man -k キーワード でオンライン・マニュアルの検索ができる。

ファイルのありかを探せ **find** というコマンドがある。

`find パス名 -name 'コマンド名のパターン' -print`

名前だけでなく、最近一週間に変更したものとか、色々な検索ができる。

`find パス名 -mtime -3 -print`

この 3 日の間に変更したファイルを表示する。

find はシステム管理者にとっては必修コマンドであるが、普通の人にはあまり縁がないかもしれない。