情　報　学　基　礎　40－5
（１９９５．　１１．　９）

# 情報検索とデータ圧縮とを統合したシステムmgの日本語化

須藤 真理　　横尾 英俊

群馬大学 工学部 情報工学科
〒376 桐生市天神町 1-5-1
E-mail: {mari, yokoo}@ail.cs.gunma-u.ac.jp

Mgシステムは，英文を対象とする全文検索システムである．ファイルの効率的な管理のために，データ圧縮技術を最大限に利用している点に特徴がある．本論文では，このシステムを日本語対応に拡張する．拡張したシステムでは，日本語文書に対する字面処理によって転置ファイルを自動生成し，これを利用したブール検索や重み付き検索を可能にしている．このシステムによって，特に，日本語文書と重み付き検索との相性のよさが明確になる．

# A Japanese-Language Oriented Extension of the mg Full-Text Retrieval System

Mari Suto and Hidetoshi Yokoo

Department of Computer Science
Gunma University
Kiryu, Gunma 376 JAPAN
E-mail: {mari, yokoo}@ail.cs.gunma-u.ac.jp

The mg system is a full-text retrieval system for English documents, in which data compression techniques are fully utilized in order to realize maximal storage efficiency. This paper extends the system so that it can deal with Japanese texts. The extended system generates an inverted file from a set of documents by automatic extraction of terms with no use of dictionary or grammatical knowledge of the Japanese language. The system can accept several types of queries including Boolean and ranked ones. This paper shows that the retrieval by ranked queries and the Japanese language match well with each other.

## 1. はじめに

マルチメディア時代の今日，電子図書館などに象徴される大規模データの格納と検索の自動化システムでは，データを効率的に保存すること，及び必要な情報を必要なときに参照できることが当然のこととして要求される．しかし，そのための基本演算である，**データ圧縮と情報検索**とは，実は相容れない要求である．大規模データの格納では圧縮が不可避であり，同時に，情報検索を直接的に行うには，本来のテキストを圧縮等を施すことなく記憶しておくことが重要である．本論文では，このような矛盾する要求である**データ圧縮と情報検索とを両立する**ための試験システム mg[11] に日本語機能を導入した結果を報告する．

Mg システム[1]は，欧文から成る大規模テキストを対象として，索引の自動生成，テキストおよび索引の圧縮，さらに圧縮後のデータ上での全文検索等を効率的に行うことを主たる機能としている.[2] Mg システムでは，これらのいずれの機能においても，テキストの構成単位としての**単語**が重要な働きをしている．たとえばテキスト圧縮の手段としては，単語単位の (word-based) 正準ハフマン符号 (canonical Huffman code)[4] が利用されている．また，索引語としての単位も単語である．このような「"単語" を基本単位にする」ということが可能なのは，欧文テキストにおける単語抽出の容易さに由来している．一方，テキストから単語の境界を自動的に判定することの困難な和文においては，同様の機能を類似の手法によって実現することは簡単ではない．実際，国内で行われている研究であっても，たとえば転置ファイルにおける hapax legomenon の処理の効率化 [7] のように，索引生成の対象を欧文に限定することも少なくない．そのため，和文を対象とした検索では，二次情報をあらかじめ準備しないフルテキストサーチ方式がより一般的であり，そのための種々のテキスト照合方式がハード／ソフトの両面から検討されている [9].

そのような狭義のフルテキストサーチ方式，すなわち，シーケンシャルな文字列照合方式の最大の問題点はその効率にある．ここで問題となる効率とは，照合に必要な時間的効率，および，照合のためにテキストをそのまま保管しなければならないという領域上の効率である．もちろん，データ圧縮を併用することによって，両効率を同時に高め得るという可能性も指摘できる [3]．しかし，いずれにしても，シーケンシャルサーチ方式は結局は文字列照合であり，検索方式の高度化 (たとえば，後述するような「重み付き検索」など) への対応という点では柔軟さに欠ける．また，利用度が高まりつつあるシグネチャファイルも一種のデータ圧縮の利用と見ることができるが，データ圧縮という観点からは**転置ファイル** (inverted file) を作ってそれを圧縮する方が効率的なことが多い [11]．このような考察を反映させた mg システムでは，転置ファイルの圧縮と格納に細部に至る注意を払い，圧縮したテキスト上での**ブール検索**と**重み付き検索**とを可能にしている．

単語を基本単位とするシステムが和文を対象とする場合にまず考えることのできる対処法は辞書の利用である (不要語辞書や切断用辞書 [2]．また [6] なども参照)．テキスト等とは独立に辞書を用意すれば，それを使ってテキストから単語を切り出すことができる．しかし，欧文を対象とする本来の mg システムでは，辞書利用の負荷が大きいことから，さらに，英文等に対しては十分精度のよい単語抽出のアルゴリズムがあることから [1]，辞書を利用するアプローチは採っていない．一方，和文に対して辞書を利用する場合に問題になるのは，漢字を使うか仮名を使うかといった表記のゆれである．また，本論文で具体例として取り上げる計算機ネットワーク上のメールの文書の場合には，そもそも，和文という制約が実際的ではない．計算機上のメールは，いわば多言語データであり，これに対処可能な辞書をあらかじめ用意することは現実的ではなく，仮に用意できたとしても，それを使う計算機の負荷は相当なものになる．このようなことから，mg システムの日本語化に当っては，辞書の導入を最小限にとどめ，本来の mg システムが行っているようなアルゴリズムによる単語抽出の可能性を探ることにした．その際に重要なことは，日本語としての単語抽出の精度が問題なのではなく，mg システムの本来の機能としてのデータ圧縮や情報検索の性能を十分なものにするということである．たとえば，あるフレーズを含む文書を検索しようとしたとき，検索の精度が十分であれば，このフレーズがどのような単語に分解されているかは副次的なことにすぎない．

本論文で対象としている日本語文書とは，漢字，平仮名，片仮名，そして ASCII 符号からなるテキストファイルである．本論文では，文書中のこのような字種の変化に着目して字面のみから単語抽出を

---

[1] mg という名前は，「ギガバイトを管理する (Managing Gigabytes)」という参考文献 [11] のタイトルに由来している．本論文で日本語化を試みた mg システムは，この文献で試験用に作成されたものである．UNIX 上で動作可能である．

[2] このほかに画像や文書画像のためのデータ圧縮機能なども有しているが，本稿では，本文中で述べた機能だけを考察の対象とする．

行う方法を採用する．一般に，漢字，片仮名は内容語 (content word) になる可能性が高く，平仮名は低い．このようなヒューリスティクスを利用することによって，日本語文書の圧縮と検索のための単語抽出を実現している．実現した mg システムは，結果的に，フルテキストサーチとキーワードサーチとの中間的な目的を実現するものとなっている．ただし，本来の mg システムの機能をすべて保っているため，中間的な目的の広範な用途に対して，現実の要求に十分応え得る強力なシステムとなっている．本論文の次節以降，単なる日本語化についてだけでなく，mg システムの基本的考え方の紹介も交えて述べることにする．

## 2. データ圧縮と情報検索

本節では，mg システムの基本機能であるデータ圧縮と情報検索について，両者をある程度独立に考察してみる．

ここ数年のテキスト圧縮技術の進歩には非常に著しいものがあり，平均的な英文であれば 1 記号当り 2 ビット程度までの圧縮が可能なようになってきている．これは，圧縮を全く行わない場合に対し，記憶領域を相対的に 3 ～ 4 倍に利用できるということであり，ハードウェアの低廉化がいかに進んでいるとはいえ，決して無視できない効果がある．一方，このようなデータ圧縮技術の進歩を直接リードしている手法の多くは，適応的符号化と呼ばれる技法を利用するのが普通である [10]．これは，データを系列として 1 走査で圧縮するもので，系列のある部分の符号化のために系列のそれ以前の過去の部分の情報を利用するものである．したがって，圧縮結果からデータの途中のある部分だけを解凍して (復号して)，その部分だけを知るという目的に利用するには原理的な困難を伴う．すなわち，情報検索と併用するには，データの先頭から復号するという自明な手法以外，簡単な利用は困難である．したがって，情報検索を念頭においたデータ圧縮法としては，静的あるいは準静的 (semi-static[11]) な手法を使わざるをえない．そのような静的な手法によってデータ圧縮を行う場合には，データのアルファベットの選択が圧縮性能を決める要因になる．たとえば，テキストの文字には関係なく 1 バイト (8 ビット) 単位で圧縮する場合と日本語の文字を 1 記号として圧縮する場合とを比較すると，通常は，後者の方がより効率的な圧縮を得ることができる．さらに，単語を圧縮の単位とすることができれば，より効率的な圧縮が期待できる．しかし，前節で述べたように，日本語では単語の自動分割が困難であり，さらに活用語尾の変化などを考慮すると，高度な辞書と文法なしに単語の正確な認識を行うことは不可能に近い．しかし，目的が圧縮に限定されているなら，正確な単語の認識よりもデータ圧縮に利用可能な意味のあるストリング (部分列) への分割がより重要となる．そこで，テキストを部分列に分解し，各部分列をデータ圧縮の単位とする手法として，次のような方法を考察することにする．

まず，単語を機械的に決定する手続きを用意し，テキストをこの手続きによって単語と認定される部分列とそうでない部分とに区別する．その結果，単語は単語を単位として，それ以外の部分は各文字を基本として圧縮を行う．たとえば，単語を機械的に決定する手続きとしては，連続する漢字，片仮名，英字を単語とみなす方法などが考えられる．この手続き例を採用した場合のこの文のデータ圧縮の単位は次のようになる．

単語: 「手続」「例」「採用」「場合」「文」「データ」「圧縮」「単位」「次」.

非単語: 「こ」「の」「\n」「き」「を」「し」「た」「は」「よ」「う」「に」...「. 」.

なお，「\n」は改行を意味する記号とする.[3] 以上のような手続きの性能は，組み合わせるデータ圧縮法と総合してデータ圧縮力の観点から評価を加えなければならない．そこで次に，組み合わせるべきデータ圧縮法の検討を行う．

上述したように，データ圧縮法のタイプとしては適応的な手法よりも静的あるいは準静的な手法が候補として考えやすい．ここで「静的」とは，上のようにして認定された単語，非単語に対する符号語が文書によらずに一定の符号に固定されている場合を言い，「準静的」とは，符号を文書などのある単位ごとに決め直す方法を言う．後者の場合，使用した符号についての情報も圧縮結果に含めなければならない点に注意する必要がある．しかし，単語分割の手続きがある程度機械的なものである場合，それによって認定される単語の集合をあらかじめ特定することは難しい．また，静的なデータ圧縮法を固定してしまうと，それに適合しないデータに対しては圧縮の劣化を引き起こす．あるいは，そのような劣化

[3] 改行はデータ圧縮と情報検索とで，また，和文と英文とで扱いが異なる．英文では，通常，単語の途中に改行が入ることがないのに対し，和文では比較的自由に改行が挿入される．単語の途中に改行があった場合，索引語の単位としての単語の途中には改行を含んではならないのに対し，文書を忠実に再現する場合には改行の位置も正しく再現されなければならない．

が生じないように静的符号を特定することは簡単ではない．このような点から，準静的な圧縮法を利用することにする．準静的なデータ圧縮モデルに組み合わせる符号としてよく調べられているのはハフマン符号であり，実際，mg システムでも準静的なハフマン符号を採用している．

一方，上述の単語分割の手続き例による分類結果を情報検索の視点から見ると，“単語” として分割された部分列[4]は，意味内容の点からも一つの意味のある単位を成していることが多く，検索のためのキーワードとなる可能性が高い．したがって，上の手続きと類似の (しかし，論理的には全く独立の) キーワード抽出手続きを用意することにより，検索用の索引を作ることができる．ただし，テキスト圧縮がこれと本質的に異なる点は，圧縮結果からもとのテキストを誤りなく忠実に再現しなければならず，単語，非単語ともこの点での重要度に差がないのに対し，情報検索は単語のみをキーとするのが普通である点である．転置ファイルとは，各単語を含む文書の番号をリストにし (これを「エントリ」という)，すべての単語に対するエントリを格納したファイルを使って情報検索を行う手法である．通常，転置ファイルの大きさは，本来の文書ファイルとオーダーに差がない程度まで大きくなる．そのため，転置ファイル自身の圧縮 (inverted file compression あるいは index compression [11]) も重要な検討課題となる．

ところで，キーワードをブール演算子で結合した質問 (Boolean query) によって検索を行うブール検索を使う場合，機械的に抽出したキーワードから成る索引を利用したのでは十分な検索精度が得られない可能性が高い．たとえば，前述の例のように「手続」が “単語” として認定された場合，「手続き」をキーとする検索には失敗することになる．このようなことから，日本語文書を対象とする場合には特に，重み付き検索の重要性が増すことになる．たとえば，

$Q$：「単語を単位とするデータ圧縮の手続き」

に関する文章を検索したいとする．この場合，まず，索引生成のためのキーワード抽出手続きと同じ単語抽出手続きをこの質問文 $Q$ に適用する．仮に，$Q$ に含まれる単語として

「単語」「単位」「データ」「圧縮」「手続」

[4]ここでいう「単語」とは，文法上の概念ではなく，それを抽出する特定の手法がある場合に，それによって “単語” として認識されたもののことである．

表 1. Mg の検索機能．

| query | 質問 (問合せ) の形式 |
|---|---|
| boolean | 単語を 3 種の論理演算子 ($\vee, \wedge, \neg$) で結合したもの． |
| ranked | 単語のリスト (文章)． |
| approx-ranked | ranked に同じ．重みの近似値． |
| docnums | 文書番号 (のリスト)． |

が認定されたとする．次に，この単語の集合と検索対象との距離 (類似度，similarity) が測定され，検索結果として距離の近いものから順に返す (適合度順出力 [5])．この過程では，「手続き」という語が索引でも質問文でも共に「手続」として扱われている限り，「手続き」という単語が日本語の単語として正しく抽出されたかどうかは余り重要でなくなる．重み付き検索のこのような簡単な例からも分るように，日本語文書を対象として「フルテキストサーチ」「ブール検索」「重み付き検索」を比較した場合，その柔軟さにおいて「重み付き検索」は極めて注目に値する．重み付き検索は mg システムの重要な機能の一つであり，この点でも，mg の日本語化の意義を強調することができる．

ここで，mg における文書間の類似度の算出法に簡単にふれておく．tf·idf 規則を採用した余弦尺度 (cosine measure [1], [5]) を利用している．まず，次の記法を導入する．括弧内は，次節で述べる計算機ネットワーク上のメールの管理を例にとった場合の対応する概念である．

$D_d$: $d$ 番目の文書 (ある特定のメール)．
$N$: 文書の総数 (メールの総数)．
$f_t$: 単語 $t$ を含む文書 (メール) の総数．
$f_{d,t}$: 文書 (メール)$d$ での単語 $t$ の出現回数．
$Q$ or $q$: 質問．
$f_{q,t}$: 質問 $q$ に単語 $t$ があるなら 1，なければ 0．

上記の諸量を使って，文書 $D_d$ および質問 $Q$ の重み $W_d$, $W_q$ を次のようにして求める．

$$w_{d,t} = f_{d,t} \log \frac{N}{f_t}, \qquad w_{q,t} = f_{q,t} \log \frac{N}{f_t},$$

$$W_d = \left( \sum_{t=1}^{n} w_{d,t}^2 \right)^{1/2}, \quad W_q = \left( \sum_{t=1}^{n} w_{q,t}^2 \right)^{1/2}$$

ここで，$n$ はそれぞれの単語数である．質問 $Q$ と文書 $D_d$ との類似度は

$$cosine(Q, D_d) = \frac{1}{W_d W_q} \sum_{t \in Q} f_{d,t} \left( \log \frac{N}{f_t} \right)^2 \quad (1)$$

で測られる．特定の質問 $Q$ に対し，式 (1) の値の大きいものから順に文書を返すのが適合度順出力である．表 1 に示すように，mg では，このような重み付き検索 (query = ranked) を含む 4 種の検索が可能である．

## 3. Mg の日本語対応化

### 3.1 メールの検索と mg

Mg システムの機能の一般的な議論の代わりに，計算機ネットワーク上での受信メールの管理を具体的な例として述べる．

電子メールの普及に伴って，受信メール数の激増と受信後のメールの管理が問題になってきている．これらの問題のうち，本論文の議論の対象になるのは受信したメールの管理である．具体的には，膨大な数のメールが受信済みであるときに，必要なメールに必要なときにアクセスするための効率的な管理手法である．基本的には，不要なメールを残さないようにし，残したメールを適切に分類しておくことで対処できるはずの問題である．しかし，不要なものの「廃棄」と必要なものの「分類」に要する手間および心理的負担は決して無視できないものがある [8]．メール文書の最終的な廃棄の判断はユーザによらなければならないが，それまでのメールの効率的な格納と検索とを，ユーザによる「分類」なしに実現するのが mg システムである．

さて，mg システムの最初の機能は，対象となっているデータを mg 用のデータに加工する操作である．受信メールが対象となっていて，個々のメール文書が検索の基本単位である場合には，まず，個々のメールごとに特別な区切り記号 (具体的には，“Control-B”) を挿入し，全メールを一つの文書ファイルに併合する．次に，この文書ファイルから転置ファイルを作り，さらに，文書ファイル，転置ファイルの両者をそれぞれ独立した圧縮法によって圧縮する．以上の操作を実現するのが mgbuild というコマンドである[5]．コマンド mgbuild を実行した後のファイルの状況を知るには，コマンド mgstat を使う．図 1 に mgstat の使用例を示す．図 1 では，最初の文書ファイルの大きさが “Input bytes” として示され，それに続いて，mgbuild によって新たに生成されたファイル名がそれぞれの大きさと共に出力される．ファイルの大きさを示す数字のあとのパーセンテージは，最初の文書ファイルに対するそのファイルの相対的な大きさである．新たにできたファイルのうち，“.text” を拡張子とするファイルは最初の文書ファイルを圧縮した結果である．圧縮に必要となる辞書は，拡張子 “.text.dict” を持つファイルに作られる．2 番目の “.invf” を拡張子とするファイルは転置ファイルを圧縮したものである．圧縮したファイルにおける各文書の開始位置へのポインタは，“.text.idx.wgt” を拡張子とするファイルに格納される．このほかにも検索で必要となるファイルができている．文書ファイル以外のこのような補助ファイルの存在にもかかわらず，全ファイルの最初のファイルに占める割合は約 86% であり，圧縮の効果が読み取れる．このような mg 用のデータが準備できると検索が可能になる．検索の例については 4 節で述べる．

```
> mgstat allfiles ↵
Input bytes                    932057,      0.89 Mbyte

(中略)

                Files required by mgquery
allfiles.text                :   309.97 Kb    34.055%
allfiles.invf                :    74.16 Kb     8.148%
allfiles.text.idx.wgt        :     3.92 Kb     0.430%
allfiles.weight.approx       :     0.39 Kb     0.043%
allfiles.invf.dict.blocked:   268.65 Kb    29.515%
allfiles.text.dict           :   126.18 Kb    13.863%
SUB TOTAL                    :   783.28 Kb    86.054%
(後略)
```

図 1. コマンド mgstat の使用例．mgbuild によって作られたファイルを見る．

### 3.2 日本語からの単語抽出

2 節で述べたように，圧縮および検索の両者において，日本語文書から単語を抽出することが必要になる．一般的には，日本語として正確な単語に分割するることが望ましいとはいえ，ここでの目的が「圧縮」と「検索」とに限定されているので，単語への分割結果はそのような具体的な尺度によって評価されなければならない．

検索と単語との関係は，いわゆる “キー” になる単語によるキーワードサーチから，“キー” か否かを前もって固定しないコンコーダンス流のものまで広いスペクトラムを想定することができる．本来の mg システムは，単語の重要度に検索システム製作者の予断が入るべきではないという考えから，後者の立場を採用している．したがって，4 節で例を示すように，不要語 (stop word) と見なされることの多い英語の “the” や “in” によっても検索を行うことができる．Mg の日本語化に際してもこの立場の

[5] この処理には比較的時間を要するので，新しいメールの到着のたびに行っていたのでは効率が悪い．現在は，ログアウトに伴って自動的に行われるように設定してある．

踏襲を基本方針としたが，辞書や構文解析を利用しない単語抽出との両立は必ずしも容易ではない．そこで，基本的には，助詞および活用のある語の活用語尾を除くすべての単語を索引に登録することを目標にした．

単語への分割は，文書中の文字種の変化に着目して機械的に行う．まず，字種として次の6種を考える．

① 平仮名，② 片仮名，③ 漢字，④ 英字，
⑤ 数字，⑥ これら以外の記号．

単語分割の第一近似は，単語を“連続する一つの字種の並び”と定義することである．これ以降，「連続する同一字種の最長の並び」を字種に応じて，“平仮名列”，“片仮名列”，“漢字列”などと呼び，それらをまとめて“単一字種列”と呼ぶことにする．たとえば，「大学院高度化推進特別予算を使って超LSIを研究する」という文からは，

| | |
|---|---|
| 漢字列: | 「大学院高度化推進特別予算」「使」「超」「研究」 |
| 平仮名列: | 「を」「って」「を」「する」 |
| 英字列: | 「LSI」 |

が得られる．こららのうち，単一字種列をそのまま単語とみなし得るものは助詞の「を」だけで，単語を得るためには更に

**分割:** 「大学院高度化推進特別予算」→「大学院」＋「高度化」＋…，
**合成:** 「超」＋「LSI」→「超LSI」，

および，分割と合成との組合せの操作が必要になる．辞書を利用しない場合，**分割**を合理的な規則によって行うことは難しく，何らかのヒューリスティクスに依らざるを得ない．一方，**合成**には上の「超LSI」や「CTスキャン」のように，英字列，片仮名列，漢字列の異なる組合せが考えられる．さらに，「使っ(て)」のように，動詞の語幹を漢字で表したものに活用語尾を平仮名で送った場合も，単語を抽出するという目的のためには，合成の必要な場合と考えられる．しかし，上で基本的目標としたように，同じ動詞による検索でも同じ活用形が使われるとは限らないので，検索のためには，動詞の語幹に相当する漢字列だけを単語として抽出する方が合理的である．すなわち，欧米語に対するstemmingの一種の考え方である．また一方，このような活用語尾の合成にも利点がないわけではない．たとえば，合成のための仮の規則として

**『漢字列に「っ」「ん」が後続している場合，動詞の音便形とみなすことができ，さらにこれに続く平仮名1文字を含めて分割する.』**[6]

を使って，「転んだけが」を「転んだ」＋「けが」に分割したとする．すると，平仮名列「んだけが」からだけでは抽出の困難な「けが(怪我)」の自動抽出の可能性を与えることになる．次節で述べる検索例では，このようにして残った平仮名列の先頭の高々3文字を平仮名の単語として索引に登録するようにしている．

一般に，**分割**は単語抽出の粒度(granularity)を上げ(精細にし)，**合成**は疎にする．上の議論からも分かるように，単語の機械的分割という点では，$n$字単位というような純粋に機械的な分割を別にすれば，合成に比べて分割の方が困難である．一方，検索という点からは，たとえば，粒度を最も極端な1記号単位とした場合，単語としてはほとんど共通性のない「ヘンデル作曲ラルゴのリズム構成」と「曲面生成アルゴリズムのデータ構造」との類似性が高まることになる．すなわち，検索の再現率(recall)は粒度の精細化に伴って向上するが，適合率(precision)は低下することになる．逆に，粒度を疎にした場合，「データを圧縮する」という文が「デー

[6]この規則は，実は，もっと拡張することができる．促音「っ」や撥音「ん」は，通常は語頭に来ることがないので，その直前が漢字列であれば，その漢字列を主成分とする単語の一部に含めるというのがこの規則のポイントである．「っ」や「ん」のこのような性質，すなわち，直前が漢字列である場合には単語の一部として漢字列に吸い寄せられる，という性質を持つ平仮名をほかに探せばよい．たとえば，平仮名の「さ」がこのような性質を持つかどうかを調べるために，次の2例を考えてみる．

「愛さえあれば何もいらない.」
「愛さず，愛されず，ただ世を怨む.」

文頭の「愛さえ」と「愛さず」に着目すると，前者が「名詞＋副助詞」であるのに対し，後者は「動詞の未然形＋助動詞」であることが分かる．つまり，漢字列の直後の「さ」は助詞の一部であったり(したがって，漢字列には吸い寄せられない)，動詞の一部であったりし，上述のような性質を持っていない．このようなチェックをすべての平仮名に対して行うと，次の平仮名(列)が直前の漢字に吸い寄せられる確率が高いことが分かる．

「い」「う」「え」「き」「く」「ち」「つ」「ぬ」「み」「む」「め」「り」「る」「れ」「ける」「らん」「ろう」「わん」「ん」「っ」

たとえば，「き」は「驚き」などとして使われる．この場合，「驚き」が名詞であるか動詞の連用形であるかは問題ではなく，「驚」という漢字に「き」が吸い寄せられるという点では同等である．2節で例に使った「手続き」も，この規則によって抽出することができる．

タ圧縮」では検索できないことにもなる．したがって，単語分割の方法は，機械的に可能な範囲で，実用的に十分な検索能力が得られる粒度に設定することが重要である．本研究では，本節で述べたようないくつかのヒューリスティクスを種々組み合わせ，単語抽出の粒度を様々に設定して評価を行っている．その際に問題となるのは，データ圧縮力という観点からは定量的な評価が加えやすいのに対し，検索能力の方は一般的な評価が難しいという点である．現在のところ，次節で示すようなメールの検索に実際に利用することで経験的な改良を続けている．アルゴリズムの詳細は，別途発表の予定である．

## 4. メールの検索例

Mg システムによって検索を行うには，3.1 節で述べた方法によって対象をあらかじめ処理しておかなければならない．その後，検索を開始するには，新たに mgquery というコマンドを入力する．プロンプト ">" が表示されるので，それに引続きコマンドまたは質問を入力する．コマンドとは，ピリオドから始まる文字列で，主として，mg システムの種々のパラメータの設定に使う．パラメータには，表 1 にあげた検索の形式などがある．先頭にピリオドを持たない文字列は，すべて検索のための質問と見なされる．たとえば，「情報検索」と「研究」とを含むメールは次のように検索される．

```
Enter a command or query (.quit to terminate, .help f
> 情報検索 & 研究
**************************************************
* 「情報検索」と「研究」とを含むメールが実際に表示される．*
**************************************************
5 documents retrieved.
Enter a command or query (.quit to terminate, .help f

>
```

上の例の記号 "&" は論理積 $\wedge$ を表し，検索の標準モードはこのようにブール検索になっている．これを重み付き検索に変えて検索を行うには次のようにすればよい．

```
> .set query ranked
> 情報検索とデータ圧縮の研究
------------------------------ 488        2.910381
To: mari@ail.cs.gunma-u.ac.jp
Subject: ronbun
Date:  Wed, 25 Jan 1995 14:47:44 +0900
From: YOKOO Hidetoshi <yokoo@ail.cs.gunma-u.ac.jp>
Content-Type: text
Content-Length: 3908
.............. 省略.............................
データ圧縮と情報検索とを両立する手法についての研究は，北米やを中心として既に活発に行われている．しかし，英語圏での研究でテキストの主たる部分が英語であるという性質に強く依存した方法えば，テキストを圧縮する場合，それと同時に検索のためのインデ動生成して，テキスト，インデックスの両者を圧縮するという手法
しかし，インデックスの自動生成は言語に強く依存し，このようした既存のソフトウェアを日本語のテキストに適用しても，十分
.............. 省略.............................

46 documents retrieved.
```

上の出力結果において，メール本体のすぐ上の数字は，488 がメール番号を表し，2.910381 がこのメールと質問文との類似度 (式 (1) から $W_q$ の寄与を除いた値) である．

また，以下のように比較的長い質問文を使うことで，ほとんど文字列照合に等しい使い方をすることもできる．

```
> ソースには不十分な箇所がありました。本日それを新しいものに入れ替えました
------------------------------ 482        1.968589

.............. 省略.............................
本日（10 月 18 日）は、都合により休講にします。　来週（25 日）より平常に行う予定です。

ところで、以下のソースには不十分な箇所がありました。本日それを新しいものに入れ替えましたので、必要な学生は新しいものを再コピーするようにして下さい。
.............. 省略.............................

1 documents retrieved.
```

メール本文は出力せずに，検索したメールの番号だけを出力するには，mode とよぶ変数を docnums にセットすればよい．"set mode docnums" とすると，次の例のようにメール番号とそれぞれの大きさだけが表示される．

```
> .set mode docnums
> data compression

       483        2.2613        3401
       482        1.2514         881
       426        0.3315        1196
       388        0.2553         860
       475        0.1770        1104
       329        0.1633         997
        88        0.1423         940
        48        0.1309        1146
       499        0.0861        2068
9 documents retrieved.
```

上の例のように英文による検索も全く同様に行うことができる．次の例は，"in" と "the" のような英単語によっても検索が可能であることを示すものである．なお，この例では，多数のメールが質問にマッチするので，"set mode count" によって条件に合うメールの数だけを出力させている．

```
> .set mode count
> in the
112 documents match.
```

一方，次の例は要求に合うメールが検索できなかった失敗の例である．「まずい」という平仮名列で検索を試みている．番号 13 のメールにそのような平仮名列があるにもかかわらず，検索されて

いない．これは，メール本文中にある「書かないとまずい」からは，「かない」が単語として抽出されて「まずい」の方は単語と認識されなかったためである．なお，下の例において，“set query docnums”は，検索をメール番号によって行うことの宣言であり(表1参照)，“set mode text”は，メールの本数だけの出力からメール本文も表示するように再切替えを行ったものである．

```
> .set query ranked
> まずい
No entries correspond to that query.
> .set query docnums
> .set mode text
> 13
------------------------------ 13
..........省略.....................................
.forwardファイルに関して質問します。2階システムの.forward
ルの中身を、mizuki@ari.sa.gunma-u.ac.jpと書いているので
"mizuki@ari.sa.gunma-u.ac.jp"と書かないとまずいのでしょ
 .........省略.....................................

1 documents retrieved.
```

上の例のように，平仮名列が長く続く場合，そこから単語を精度よく抽出することは困難で，機械的な単語抽出の限界を示すものになっている．しかし，上のような例は恣意的に与えない限り現実にはほとんど遭遇することはなく，経験的には十分実用に耐えるシステムと言える．

なお，mgシステムには，以上の機能をXウィンドウ上でより使いやすくしたxmgシステムが付随している．図2に示すように重み順に提示されたメールの頭書き部分をマウスでクリックすると，ウィンドウの下半分でメール本文を読むことができる．これに対する日本語入力機能の導入は未完成で，この点での拡張は今後の課題である．

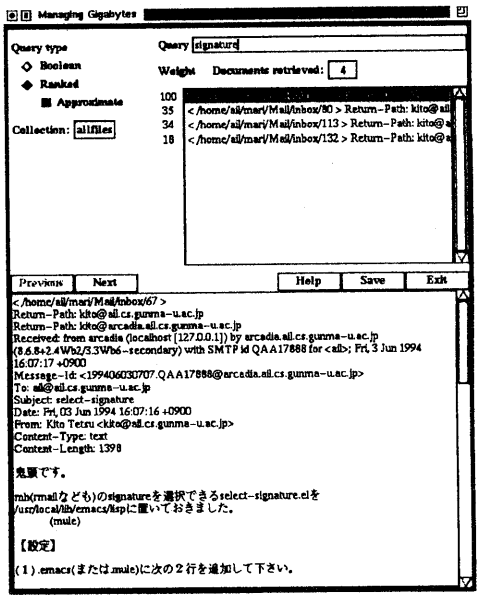


図2. Xmgのウィンドウ例.

## 5. むすび

本論文では，全文検索システムmgに日本語機能の導入を行った．検索の基本技法として文字列照合とキーワードサーチとを比較した場合，特に日本語の場合は両者とも柔軟性に欠け，これらの中間的な検索方式が実用的には望ましい．本論文の単語抽出法は単純ではあるが，重み付き検索と組み合わせることで，この要求に現実的な解答を与えるものになっている．今後は単語抽出の精度をより改善し，さらに未拡張のxmgの日本語化などによってユーザインターフェースの向上を図ることが必要である．

## 参考文献


[1] W. B. Frakes and R. Baeza-Yates (eds.): *Information Retrieval: Data Structures and Algorithms.* Prentice Hall (1992).

[2] 藤澤浩道，絹川博之: “情報検索における自然言語処理.” 情報処理，Vol. 34, No. 10, pp. 1259–1265 (Oct. 1993).

[3] 深町，下薗，有村，篠原: “文字列パターン照合のための損失のあるデータ圧縮.” 電子情報通信学会技術研究報告, Vol. 95, No. 29, pp. 41–48 (March 1995).

[4] D. S. Hirschberg and D. A. Lelewer: “Efficient decoding of prefix codes.” *Commun. ACM*, Vol. 33, No. 4, pp.449–459 (April 1990).

[5] 細野公男 (監訳): 情報検索論 — 認知的アプローチへの展望．丸善 (1994).

[6] 増市，山浦，小山，舘野: “形態素解析を用いた全文検索システムとその応用.” 情報処理学会研究報告, Vol. 94, No. 63, pp. 17–24 (July 1994).

[7] 松男文碩，佐藤誉夫，高山 悟: “情報検索システムにおける文書参照ファイルの効率的構成.” 情報処理学会論文誌，Vol. 36, No. 6, pp. 1486–1494 (June 1995).

[8] 野口悠紀雄: 「超」整理法．中央公論社 (1993).

[9] 小川隆一，菊池芳秀，高橋恒介: “フルテキスト・データベースの技術動向.” 情報処理，Vol. 33, No. 4, pp. 404–412 (Apr. 1992).

[10] 植松友彦: 文書データ圧縮アルゴリズム入門．CQ出版社 (1994).

[11] I. H. Witten, A. Moffat, and T. C. Bell: *Managing Gigabytes: Compressing and Indexing Documents and Images.* Van Nostrand Reinhold (1994).