データベースシステム　92－1
（１９９３．３．２２）

文字成分表による文字列検索の実現と評価

岩崎 雅二郎 小川泰嗣

（株）リコー研究開発本部　中央研究所

隣接する文字のペアがどのテキストに出現するかを示す隣接文字成分表を利用するテキスト検索方式を提案する。文字成分表は処理の重い日本語処理を必要としない点で有利であるが文字数の多い日本語には適さない。そこで、次のような改良を加えた。1) 隣接する文字コードの一部を連結して隣接文字成分とした。2) 隣接文字成分表を文字種ごとに分けた。3) 文字成分表を圧縮した。このような改良の結果、提案する方式では高い適合率と小型な文字成分表を実現することができた。

Implementation and evaluation for a text retrieval method using character bitmap tables

Masajirou IWASAKI, Yasushi OGAWA

Research And Development Center,RICOH Co.,Ltd.

In this paper we propose a new text retrieval method using adjacent character bitmap tables that shows which pair of adjacent characters exists in the text. For Japanese text retrieval, character bitmap table methods are preferable to natural language processing (NLP) based methods. However, the sheer number of Japanese characters prohibits the use of existing character bitmap table methods for Japanese. Thus, we have modified the adjacent character bitmap table method as follows: 1) Only certain bits are used to form an adjacent code. 2) The adjacent character bitmap table is divided into several parts based on character sets of Japanese. 3) Character bitmap tables are compressed. As a result, this method achieves high precision and compact bitmap tables.

1 はじめに

近年、二次記憶などの周辺機器を含めコンピュータのハードウェアは低価格になり、さらに、CD-ROM のような低価格で、しかも大容量の新しい記憶装置も普及してきた。これに伴い、ユーザは容易に大容量の記憶装置を手に入れるようになり、データベースへの要求も一段と大きくなってきている。しかも日常身の回りにある情報の多くはテキストデータのような非定型データであり、関係モデルのような従来型データベースで扱うには適していない [15, 16]。したがって従来型データベースを補うものとしてテキスト検索システムが重要視されるようになってきた。

テキスト検索の一つとして自然言語処理により単語（キーワード）を抽出し、検索する方式がある。しかし自然言語処理は膨大な辞書を必要とするだけでなく、単語（キーワード）を抽出する精度も必ずしも高くない。

自然言語処理を必要としないテキスト検索の方法として、全テキストデータと検索語を照合してテキストに存在するか否かを調べるパターンマッチングの方式 [1, 2] による全文検索がある。しかし、この方式では大量のデータを処理しなければならず、高速な検索を行なうにはパターンマッチング用の LSI や高速な二次記憶装置が不可欠である。[12, 13]

一方、自然言語処理を必要とせず、しかもソフトウェアのみで全文検索を実現する方式として、どの文字がどのテキストに出現するかを示す文字列成分表を利用するインデックス方式がある。我々はこの文字成分表方式の中でも、隣接する文字のペアがどのテキストに出現するかを示す隣接文字成分表 [5] による方式に着目した。この方式は検索ノイズは少ないが、日本語の全文字コードに適用すると隣接文字成分表が膨大な大きさになってしまう。そこで、われわれは次のように隣接文字成分表を改良した。1) 隣接する文字コードの一部を連結して隣接文字成分とした。2) 隣接文字成分表を文字種ごとに分けた。3) 文字成分表を圧縮した。

本論文の 2 章では文字成分表とその問題点について述べる。3 章では文字成分表を改良した新しい方式を提案する。そして提案する方式の評価を 4 章で述べる。

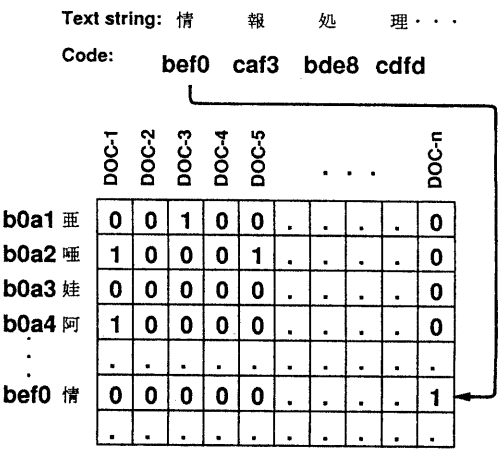

図 1: 単一文字成分表

2 文字成分表方式

2.1 単一文字成分表

図 1で示すように単一文字成分表はどの文字がどのテキストに出現するかを示す。"1" は対応する文字が対応するテキストに出現することを意味し、"0" は出現しないことを意味する。

l個の文字 t からなるテキスト Tを $t_1 \cdots t_l$で表す。テキスト Tの文字 c に対応するビット $S_{sig}(c,T)$ は次のように表される

$$S_{sig}(c,T) = \begin{cases} 1 & : \exists t_j, c = t_j \\ 0 & : otherwise \end{cases} \quad (1)$$

検索時には、検索語の各文字に対応するテキスト集合を文字成分表から得る。単一文字成分表で文字 c に対応するテキスト集合を $L_{sig}(c) = \{T_i | S_{sig}(c,T_i) = 1\}$ とし、検索語 Q を $q_1 \cdots q_m$ とすると、検索テキストは集合は次のように表される。

$$\text{検索テキスト集合} = \bigcap_i L_{sig}(q_i) \quad (2)$$

この方式では全文検索と比較し二次記憶へのアクセスが格段に減り検索時間を短くすることができる。しかし、検索語を含む文書は必ず検索されるので洩れはないが、検索語が複数の文字からなる場合には、検索結果にはたくさんのノイズが含まれる。すなわち、文字成分表が文字の隣接情報

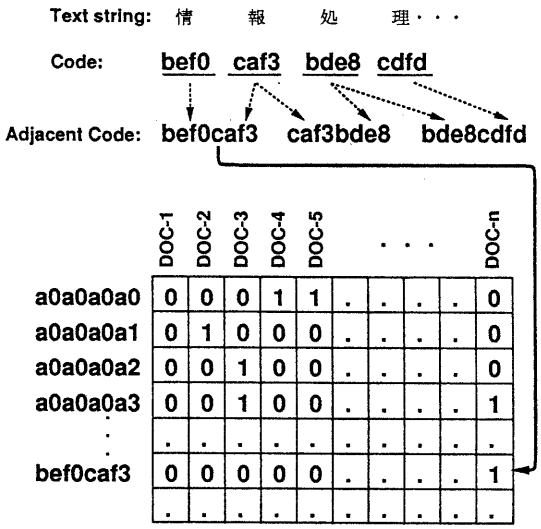

図 2: 隣接文字成分表

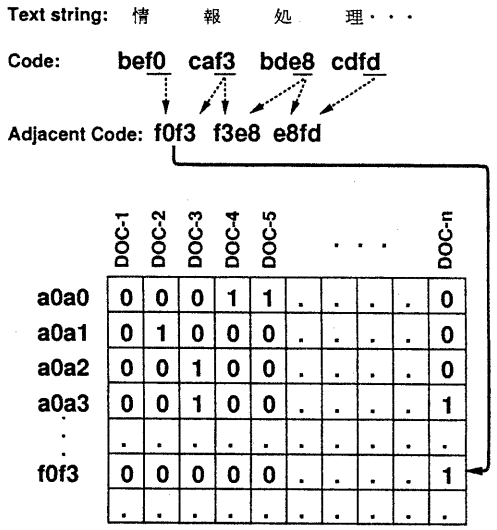

図 3: 隣接文字成分表

を持っていないので、検索語の各文字がばらばらにテキスト中に出現するようなテキストも検索してしまう。そこで、日立のシステムでは、単一文字成分表で検索したテキストをさらにスキャンしてノイズを除去している [18]。

2.2 隣接文字成分表

前述の問題を解決するために、隣接する文字のペアがどのテキストに出現するかを示す隣接文字成分表を利用する方式 [5] がある。図 2は隣接文字成分表を示す。

テキスト Tの隣接文字成分 d に対応するビット $S_{adj}(d,T)$ は次のように表される。

$$S_{adj}(d,T) = \begin{cases} 1 & : \exists t_j, d = t_j \oplus t_{j+1} \\ 0 & : otherwise \end{cases} \quad (3)$$

$\oplus$ はビットの結合演算子である。

隣接文字成分表で隣接文字成分 d に対応するテキスト集合が $L_{adj}(d) = \{T_i | S_{adj}(d,T_i) = 1\}$ とすると検索テキスト集合は次のように表される。

$$\text{検索テキスト集合} = \bigcap_j L_{adj}(q_j \oplus q_{j+1}) \quad (4)$$

2文字からなる検索語の場合には検索集合はノイズを含まない。しかし3文字以上の場合には、隣接する文字の各ペアがばらばらにテキスト中に存在するテキストも検索するのでノイズを含む。しかしノイズは単一文字成分表の場合に比べてはるかに少ない。

単一文字成分表のサイズは全文字数に比例するが、隣接文字成分表は2文字の組合せなので全文字数の二乗に比例する。したがって、日本語は文字数が多い (7,270 文字) ので隣接文字成分表は膨大なものとなってしまう。すなわち、$7{,}270^2 = 52{,}852{,}900$ ビットが各文書ごとに必要となり、実用的ではない。

隣接文字成分表のサイズを抑える方法としては、隣接文字のペアに二段階のハッシュをかけることにより文字成分表のエントリを減らす方式 [18] がある。

3 提案方式

文字成分表はテキスト検索のアクセスメソッドとして有望であるが前述のような問題があるので、以下のような改良を提案する。

1. 隣接する文字コードの一部を連結して隣接文字成分とした。
2. 隣接文字成分表を文字種ごとに分けた。

3. 文字成分表を圧縮した。

これらの改良点について以下に詳細に述べる。

3.1 文字成分表の構成

前述のように、隣接文字成分表は極めて巨大になる。隣接文字成分表のサイズは隣接する文字コードのビット数の二乗に比例するので、隣接する各文字コードの一部を連結したものを隣接文字成分としている。隣接する文字の下位1バイトを連結した隣接文字成分とした場合の様子を図3に示す。この場合には各文書ごとに必要なビット数が $FFFF$(16進) = 65535 ビットで済む。

3.2 隣接文字成分の分割

文字種は以下のように分類し、新聞記事 20,000 件について各文字種の出現頻度を調べた。

- 記号: ☆★○●◎♂♀ ↓...
- 英数字: ０１２３... , ＡＢＣＤ...
- ひらがな: あいうえお...
- カタカナ: アイウエオ...
- 第一水準: 亜唖娃阿哀...
- 第二水準: 弌丐丕个丱...
- その他[1]: Ａ Ｂ Γ Δ Ｅ..., А Б В Г Д...

各文字種ごとの文字の出現率を表 1に示す。表中の出現率は以下の式で表される値の平均である。

$$\text{出現率} = \frac{\text{文字が出現したテキスト数}}{\text{全テキスト数}}$$

提案する方式では隣接文字成分表を、各文字種ごとの文字成分表及び隣接する文字が異なる文字種の文字成分表とに分けた。以後前者を同種隣接文字成分表、後者を異種隣接文字成分表と呼ぶことにする。このように分けることによって次のような効果がある。

- ひらがな、カタカナ、英数字の出現率は漢字よりはるかに高いのでこれらの文字種の適合率を上げるために漢字よりも隣接文字成分のビット数を多くする必要がある。分離した結果、文字種の出現率に応じて隣接文字成分のビット数を設定することができる。

[1]アラビア文字、ロシア文字、グラフィック文字

表 1: 各文字種の出現率

文字種	出現率 (%)	文字数
記号	6.41	150
英数字	7.76	62
ひらがな	36.70	83
カタカナ	25.16	86
第一水準	3.72	3150
第二水準	0.07	3593
その他	0.10	146
全体	2.56	7270

- 隣接する文字コードの一部を隣接文字成分としているので、異なる隣接する文字のペアが同一の隣接文字成分となることがある。したがって、隣接文字成分表を分離していない場合には、隣接する出現率の低い漢字の隣接文字成分が出現率の高いひらがなの隣接文字成分と一致し漢字の適合率を下げることになる。一方、隣接文字成分表を分離した場合にはこのようなノイズを排除できる。

本方式では、隣接文字成分として隣接する文字の一部しか利用していないので、検索語の文字を含まないテキストを検索する可能性がある。このようなノイズを避けるために、本方式では単一文字成分表も利用している。最後に本方式の文字成分表の構成を以下にまとめる。

- 単一文字成分表：各文字がどのテキストに出現するかを示す表。
- 隣接文字成分表：各隣接成分がどのテキストに出現するかを示す表。隣接文字成分は隣接する各文字の一部を連結したビット列である。t_i や t_j から隣接文字成分を抽出するフィルタを $F(t_i, t_j)$ とすると、テキスト Tの隣接文字成分 d に対応するビット $S_{adj}(d,T)$ は次のように表される。

$$S_{adj}(d,T) = \begin{cases} 1 & : \exists t_j, d = F(t_j, t_{j+1}) \\ 0 & : otherwise \end{cases} \tag{5}$$

  - 同種隣接文字成分表：隣接する同種の文字から抽出した隣接文字成分がどのテキ

ストに出現するかを示す表。記号、英数字、ひらがな、カタカナ、第一水準、第二水準、その他の文字種についてそれぞれ同種隣接文字成分表をもつ。

- 異種隣接文字成分表：隣接する異種の文字から抽出した隣接文字成分がどのテキストに出現するかを示す表。

ただし、検索テキスト集合は各文字や隣接文字成分がばらばらにテキスト中に出現するテキスト及び検索語とは異なる隣接文字であるにも関わらず偶然隣接文字成分が一致する隣接文字が出現するテキストを含む。

3.3 文字成分表の圧縮

提案する隣接文字成分の構成は、隣接する文字すべてを対象とした隣接文字成分表よりサイズは極めて小さくなっているが、それでもかなり大きい。例えば、単一文字成分表のみで1テキスト当たり909バイト(7,270ビット)が必要となる。しかし、表1からもわかるように、各文字種の出現率はかなり低い。したがって、文字成分表での"1"の出現はかなり疎らであり、容易に高い圧縮をすることができる。また文字成分表の圧縮は、二次記憶のアクセス量を減らすことができ、結果的に検索時間の短縮にも効果がある。

いろいろなデータ圧縮方式[8, 10, 11]があるが以下の理由によりランレングス符合化の一種であるExp-Golomb符合化[9]を採用した。

- 疎らなデータの圧縮に適している。
- アルゴリズムが比較的簡単で高速化が可能である。

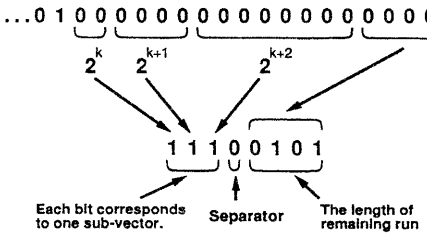

図4: Exp-Golomb符合化

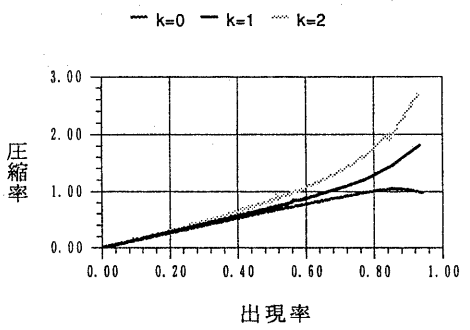

図5: 圧縮率

- 出現率の変動が圧縮率にあまり影響しない。
- 伸長せずにデータの追加が可能である。

ランレングス符合化は連続する"0"ビットの長さ(ランレングス)によってビットストリングを表現する。図4に19ビットのランレングスのExp-Golomb符合化を示す。まず、ランをブロックに分ける。ただしブロックのサイズは2の乗数で大きくし、一番目のブロックのサイズはkの値で与えられる$(2^k, 2^{k+1}, 2^{k+2}, \cdots)$。図は$k=1$の場合を示し、ブロックのサイズは2,4,8ビットとなる。セパレータの前の"1"ビットの個数がブロックの個数を示す。セパレータの後のビットストリングはブロック化されなかった残りのランの長さである。

このアルゴリズムでは、kの値が圧縮率に影響する。図5は$k=0,1,2$の場合の出現率に対する圧縮率を示す。$p \approx 0$の時にはkが大きいほど高い圧縮率となるが、その差はほとんどない。$p > 0$の時にはkの値が大きいほど低い圧縮率となり、しかもその差が極めて大きい。したがって、本方式では$k=0$を採用した。

3.4 データ構造

図6に文字成分表のデータ構造を示す。文字成分表はインデックスファイルとデータファイルからなる。文字成分表本体はデータファイルに保管される。データ処理が容易なように、文字成分表本体の各文字及び各隣接文字成分に対応する1行は固定長のブロックに分割されている。インデッ

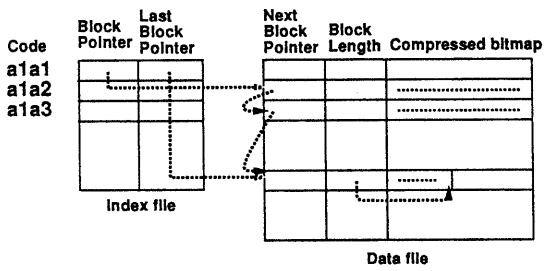

図 6: 文字成分表のデータ構造

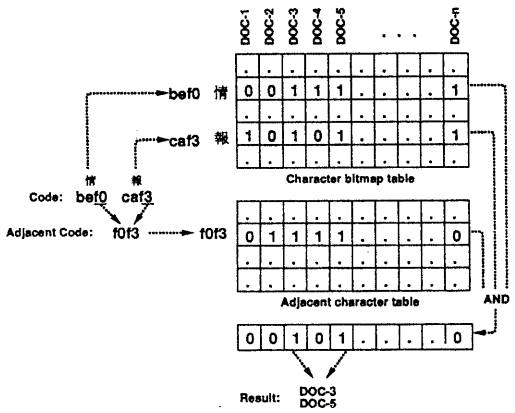

図 7: 検索

クスファイルは最初のブロックと最後のブロックへのポインタを持つ。したがって最後のブロックを得る時に次ブロックポインタをたどる必要がないので、テキストを追加する時の処理を高速にできる。

3.5 テキストの登録検索

テキストは式 (1) 及び (2) に従って文字成分表に登録する。

以下の式は検索テキスト集合を表す。

$$\text{検索テキスト集合} = \{\bigcap_i L_{sig}(q_i)\} \cap \{\bigcap_j L_{adj}(F(q_j, q_{j+1}))\} \tag{6}$$

実際の検索処理の手順（図 7）を以下に示す。

1. 検索語から各文字及び隣接文字成分を抽出する。
2. 各文字や隣接文字成分に対応するビットマップをデータファイルから読み出し伸長する。
3. 伸長したビットマップから得られたテキスト集合の検索集合として AND 集合を求める。

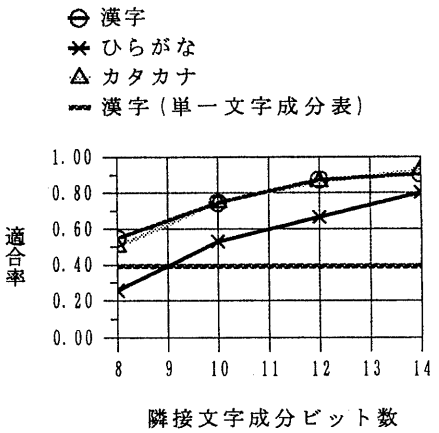

図 8: 隣接文字成分ビット数に対する適合率

4 評価

提案した方式について 20,280 件の新聞記事（平均テキストサイズ 1043 バイト）を登録し評価した。

4.1 適合率

各文字種（第一水準漢字、ひらがな、カタカナ）について 2〜5 文字の検索語で検索した。それぞれの文字数について 20 個の検索語を用い適合率[2]を算出した。適合率は以下の式で表される。

$$\text{適合率} = \frac{\text{正解検索件数}}{\text{全検索件数}}$$

隣接文字成分のビット数（各文字から抽出するビット数はこのビット数の 1/2 となる。）の変動に対する適合率を図 8に示す。なお本システムでは、文字コードの下位の連続する 4〜7 ビットを抽出し隣接文字成分（8〜14 ビット）とした。比較のため単一文字成分表を利用した場合の適合率も図 8

[2]検索方式の評価のめやすとして再現率があるが、本方式では洩れがないので再現率は必ず 1.0 となる

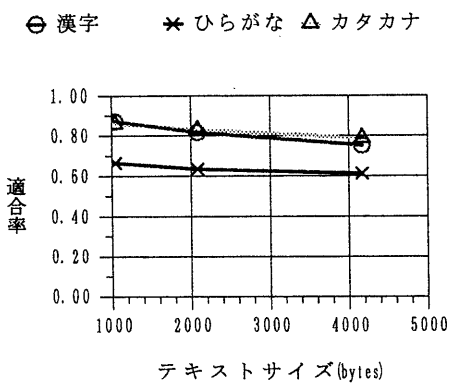

図 9: テキストサイズに対する適合率

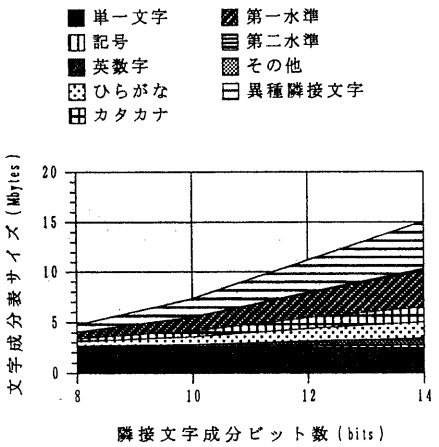

図 10: データサイズ

に示す。図 8から隣接文字成分のビット数が大きくなるほど適合率が高くなることがわかる。特に、漢字やカタカナの適合率は単一文字成分表のみの場合に比べ劇的に高くなっている。ひらがなの出現率は漢字やカタカナよりかなり高いので、ひらがなの適合率は漢字やカタカナより低くなっている。しかしひらがなは一般に検索語となることが少ないので実用上問題がない。

図 9は隣接文字成分のビット数が 12 ビットの時のテキストサイズに対する適合率を示す。テキストサイズが大きいほど適合率は下がる。これはテキストサイズが大きいほど異なる隣接文字のペアであるにも関わらず偶然隣接文字成分が一致する可能性が高くなるためである。しかし、適合率の低下は比較的少ないので、実用上は問題ないであろう。

4.2 文字成分表のサイズ

図 10に隣接文字成分のビット数に対する各文字成分表のサイズを示す。ただし、このデータサイズはインデックスファイルとデータファイルの合計である。隣接文字成分として抽出するビット数が増えるほど、各文字成分表のサイズは増えている。特に第一水準漢字の同種隣接文字成分表や異種隣接文字成分表の割合は高い。ひらがなや異種隣接文字はあまり検索語として使われないので、このような文字種の隣接文字成分のビットを減らすことによって全体のデータサイズを減らすことができる。なお単一文字成分表は常に文字コードの全ビットを使用するのでサイズは一定である。

隣接文字成分のビット数に対する各ファイルのデータ構成を図 11に示す。データファイルは固定長のブロックによって構成されているので、ブロック中に空き領域を含んでいる。隣接文字成分のビット数が増えると、データファイルで占める空き領域の割合が増えている。隣接文字成分が 14 ビットの時には、データファイルのほぼ 1/3 が使われていない。しかし、ブロックサイズを小さくすることによってこの空き領域は容易に少なくすることができる。なお評価に使ったシステムでは、ブロックサイズは 256 バイトである。

4.3 処理速度

処理速度は Sun SPARCstation 2(SPARC CPU 40MHz、メインメモリ 16 Mbytes)、内蔵 SCSI ディスク (アクセスタイム 16 msec, 転送速度 4 Mbytes/sec)、Sun OS 4.1.2、シングルユーザモードという条件下で測定した。また、文字成分表のインデックス及びデータファイルは内蔵ディスク上にある。

図 12にテキストサイズに対する登録時間を示す。登録時間は完全にテキストサイズに比例して増加

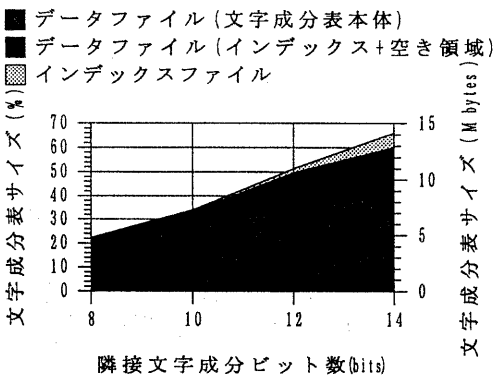

図 11: データサイズ

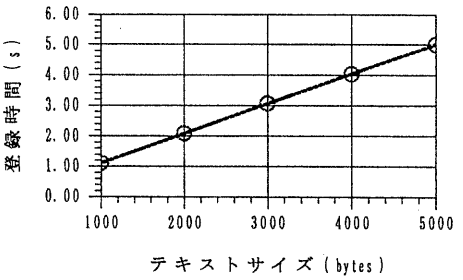

図 12: 登録時間

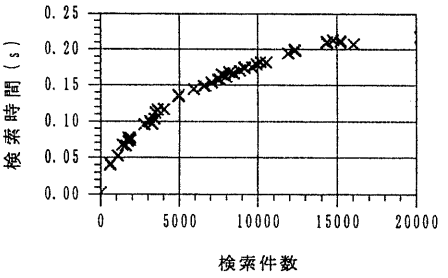

図 13: 検索件数に対する検索時間

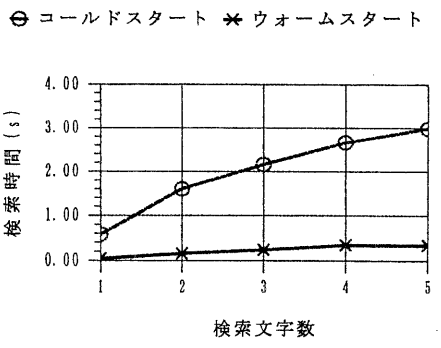

図 14: 検索文字数に対する検索時間

する。

図13に1文字の検索語について検索件数に対するウォームスタート時の検索時間を示す。図は検索件数が増えるほど検索時間がかかることを示している。これは検索テキスト数が多いほどビットマップの伸長処理に時間がかかるためである。

コールドスタート及びウォームスタート時の検索語の文字数と検索時間の関係を図14に示す。検索語の文字数を1〜5文字とし、検索件数の変動による影響を受けないように検索語は検索件数が100件以下のものとした。検索語が1文字から2文字での増加が大きいが、これは1文字の場合には単一文字成分表しか参照しないが、2文字以上の場合には隣接文字成分表をも参照しなければならないためである。

また、コールドスタートの検索速度にはディスクへのアクセスを含むので、ウォームスタートより時間がかかる。実用上では、ある程度キャッシュの効果を期待できるので、十分な検索速度を得られる。

本システムでは、文字成分表をディスク上においたがメモリ上に置くことによって、登録検索処理共に容易に高速化が可能である。

5 おわりに

我々は隣接する文字のペアがどのテキストに出現するかを示す隣接文字成分表を使ったテキスト検索方式を提案した。

日本語のテキストでは形式上の単語の区別がなく単語を分離するために自然言語処理を必要とする。しかし、自然言語処理は膨大な辞書を必要とし、しかも、高い精度を期待できない。そこで、自然言語処理を用いない隣接文字成分表による方式を採用した。

しかし、日本語の文字数は極めて多くすべての隣接する文字について隣接文字成分表を作ることは現実的ではない。そこで隣接文字成分表に次のような改良を加えた。

1. 隣接する文字コードの一部を連結して隣接文字成分とした。
2. 隣接文字成分表を文字種ごとに分けた。
3. 文字成分表を圧縮した。

こうして高い適合率と小型な文字成分表を可能とした。隣接文字成分のビット数と適合率はトレードオフの関係にあるので、小型な文字成分表を重要視するなら隣接文字成分のビット数を減らし、また、高い適合率を重要視するならビット数を増やせば良い。また、検索文字種ごとの隣接文字成分のビットについても同様なことがいえ、どの文字種の検索を重要視するかによってビット数を設定することができる。このように適切な隣接文字成分のビット数を設定することにより最小限のデータ量で十分な検索が可能となる。

また、この方式は中国語、韓国語などの２バイトコードの言語にも、抽出ビット数などを変えることにより適用できる。

参考文献

[1] Aho,A.V. and Corasick, M.J., Efficient string matching: An aid to bibliographic search,Communications of ACM, 118(6), 333-340, 1975

[2] Boyer,R.S. and Moore,J.S., A fast string searching algorithm, Communications of ACM, 20(10), 762-772, 1977

[3] Christodoulakis,S. and Faloutsos, C.,Design consideration for a message file server.IEEE Transaction on Software Engineering, Se-10(2), 201-210, 1984

[4] Roberts,C.S., Partial-match retrieval via the method of superimposed codes, Proceeding of IEEE,67(12), 1624-1642, 1979

[5] Tavakoli,N. and Ray,A., A new signature approach for retrieval of documents from free-text database, Information Processing & Management , 28(2), 153-163, 1992

[6] Faloutsos,C. and Christodoulakis,S., Signature Files:An access method for documents and its analytical performance evaluation

[7] Moffat,A. and Zobel,J., Parameterised compression for sparse bitmaps, SIGIR '92, 274-285, 1992

[8] Jakobson,M., Huffman coding in bit-vecter compression,Information processing letters, 7(6), 304-307, 1978

[9] Teuhola, J.,A compression method for clusterd bit-vecters,Information processing letters, 7(6), 1978

[10] Choueka,Y., Fraenkel,A.S., Klein,S.T., Segal,E., Improved hierarchical bit-vector compression in document retrieval systems, In Proc. 9th ACM-SIGIR Conf, 88-96, 1986

[11] Fraenkel,A.S., Klein,S.T., Novel Compression of sparse bit-strings-Preliminary report, NATO ASI Series,Combinatorial Algorithms on Words Edited by A.Apostolico and Z.Galil, F12, 169-183, 1985

[12] Hollaar,L.A., Text retrieval computers, IEEE Computer, 12(3), 40-50, 1979

[13] Ozakarahan,E., Database machines and database management, Englewood clifs, NJ:Prentice-Hall, 1986

[14] JIS Handbook (Information processing), Japanese Standards Association, 1991

[15] Shapiro,E., Text databases, BYTE, 147-150, Oct. 1984

[16] Clifton,C. and Garcia-Molina,H., The design of a document database, ACM Conf. on document processing systems, 125-134, Dec. 1988

[17] 増井、シグネチャ法と曖昧検索を用いた文書検索システム、第 18 回 jus UNIX シンポジウム論文集、1991

[18] 加藤 藤澤 大山 川口 他、大規模文書データベース用テキストサーチマシンの開発、情報学シンポジウム 講演論文集、1991

[19] 浅川 川下 坂田 畠山、フルテキストサーチシステム Bibliotheca/TS の開発 (2)、情報処理学会第 45 回全国大会（文冊 3）、241-242、1992