高速全文検索の一手法

4E－2

堀川 恵美　新谷 義弘　長坂 篤

沖電気工業（株）マルチメディア研究所

1　はじめに

コンピュータやワープロの普及によって大量の文書が電子化されてきており，またCD-ROMや電子メール，ニュースの普及によって，これらの情報が容易に利用可能になってきた．これに伴い，このような情報を有効に利用するための検索技術へのニーズが高まっている．このような動きの１つとして従来の情報検索システムに対して，使いやすく柔軟な検索機能をもった情報検索システムとしてのフルテキストサーチシステム(全文検索システム)の有効性が認識されてきている．

本論文では，我々が新たに開発した，高速かつメモリ効率の高いフルテキストサーチ方式FTSについて報告する．

2　従来のフルテキストサーチの問題

代表的なものとして，String search，Signature Index，Inverted Index，Pat Treeなどがある．しかし検索効率，速度などに以下のような問題がある．各語のインデックスを作成し，それらを元に本文へのポインタで参照するInverted Indexや，各語をビットパターンを表現し，問い合わせにマッチングとり検索するSignature Indexでは元データをインデックスにどの様に反映させるかで，検索効率が大きく変わってくる．また，任意の文字列を検索できるString searchは，逐次検索のため膨大なテキストデータベースに対しては，実用的な速度が出ない．Pat Treeはメモリ効率は高いが，実装が困難という問題がある．これらの諸々の問題から，我々は高速かつメモリ効率の高いフルテキストサーチの方式を新たに開発した．

3　フルテキストサーチ方式 FTS

3.1　概要

我々の開発したシステムFTSは，文字列パターンを利用し，その文字列の該当場所を確定する．一通りの文字列を検出するのに２文字の組み合わせからなる文字列の論理積をとっていく．モデルを図１～３に示す．

3.2　構造

あらかじめ，元データから下記のものを用意する．

- 文字コードテーブル１
  32K個の要素からなる１次元配列(TABLE)．TABLE[n]には文字コード“n”の内部番号が入っている．文書中で未使用の文字コードのところには０が入り，使用されたものについては使用頻度の高い順から振られた一意の番号が入っている．
- 文字コードテーブル２
  文書中で使用されている文字に対して生成される２次元配列．N種類の文字が使用されている時，N×(N+1)のテーブルで用意され，要素は文字コードテーブル１で変換される内部番号の組合せで参照される．列ごとに頻度順を振った番号が書いてある．要素が０ならば，そのような文字の組合せは検索対象文書中に存在しないことを示している．実現上メモリ効率の点から完全な２次元配列にはなっていない．
- 検索ベクタ
  使用されている各文字について存在する．検索ベクタの大きさは，文字コードテーブル２中でその文字に対応する列の要素の最大値サイズ＋１で，文字列に対する以下の情報を示す．

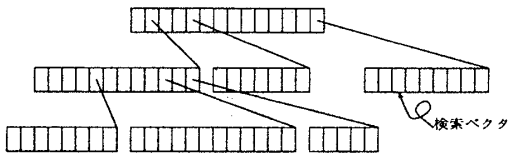

図 1: 木構造

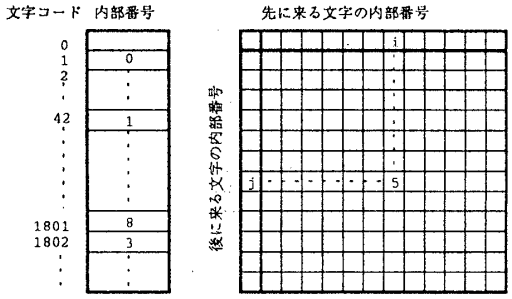

図 2: 文字コードテーブル１，文字コードテーブル２

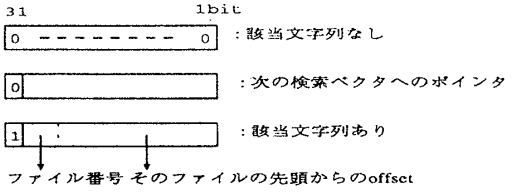

図 3: 検索ベクタの要素

A method of fast Full-Text-Search
Emi Horikawa,Yoshihiro Shintani, Atsushi Nagasaka
Oki Electric Industry Co, Ltd.,Media Laboratories

最上位 bit が0: 次の検索ベクタを示す.
最上位 bit が1: 該当文字列あり. それより下位にファイル番号, そのファイル上での位置を示す.
全ての bit が0: この組合せの文字列はない.

3.3 アルゴリズム

(1) 長さLのある検索文字列Sに対し文字コードテーブル1より, 内部番号 $S'(n)(0 < n \leqq L)$に変換. $S'(n)$に0があればその文字列はないので検索終了.
(2) n=1 とする. 図 1 の木のルートに当たる検索ベクタをとりだし, それより $S'(n)$に対する値を取り出す. 最上位ビットが1ならば検索終了. それより下位にファイル番号, 文字位置が示される. $S'(n)$に対する値が0でなく, かつ上位1ビットが0でなければその値は次の検索ベクタをさし, これをカレント検索ベクタとする.
(3) $S'(n)$と $S'(n+1)$について, 文字コードテーブル2より, $S'(n)S'(n+1)$ と続く文字列の $S'(n+1)$に対する検索ベクタのインデックスを取り出す.
(4) カレント検索ベクタの(3)で得たインデックスの値を取り出す.
(5) この検索ベクタの値が0ならば, 該当文字列なし. 検索終了. 上位1ビットが1ならば該当文字列あり. 検索終了. 正ならば, その値は次の検索ベクタを指し, これをカレント検索ベクタとする. n=n+1 として(3)へ. 但し, n ≧ N ならばカレント検索ベクタ以降の全てについて(3)〜(5)を繰り返す.

4 方式評価

本方式の評価のために, 2分木法, 256分木法, System1との比較を行なった. ここで, 2分木法は, 文字コードの大小で作成した木のインデックスによる検索方式, 256分木法は, 半角ずつ取り出したコードの大小で作成した木のインデックスによる検索方式をあらわす. またSystem1は2文字のペアでグループ化して, インデックスを作成したもので, 著者が [4] を元に独自に作成したものである.

性能評価は 同一環境(SPARCstation10/30 上で同一文書データ)で行ない, 検索速度およびメモリ効率について測定した.

(1) 検索時間
検索時間の比較を図 4に示す. 3 つの方式の中ではFTS が全体的に最も高速である. 文字列が 12 文字以上では256 分木法が高速であるが, 一般的な使用では, より短い文字列での検索が多いと考えられる.
(2) メモリ効率
各方式のメモリサイズを表 1 に示す. ここでFTSの1次テーブルは固定部分であり, 2次テーブル, インデックス木は文書データ量に応じてサイズが変化することに注意する. FTS は最もメモリ効率が高い.

以上から, FTS は実行速度及びメモリ効率の点から他の方式に比較して優れていることがいえる.

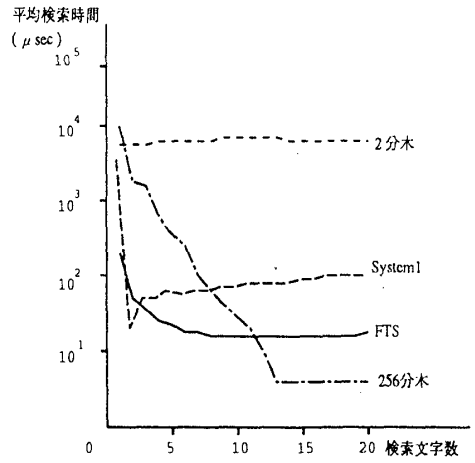

図 4: 検索時間の比較

表 1: メモリ効率

	合計 (Byte)	内訳	
元データ	54,512		
FTS	611,344	一次テーブル	131,072
		二次テーブル	164,128
		木	316,088
2 分木	965,064		
256 分木	23,696,428		
System1	123,040,448	一次個数テーブル	61,864,000
		一次アドレステーブル	61,864,000
		二次テーブル	112,448

5 まとめ

検索速度及びメモリ効率共に優れたフルテキストサーチ方式を開発した. この方式は現在 Unix WS 上に Interleaf5 をフロントエンドとして実現されている. 今後の課題として, 実用化に向けて次のような機能拡張, 性能向上を行なっていく.

- 検索文字数が多い場合の高速化
- メモリサイズの一層の圧縮
- 検索論理式の機能拡張
- シソーラス機能

参考文献

[1] Williams B.Frakes, Richard Baexa-Yates ed.: "Information Retrieval - Data Structures & Algorithmhs -", Prentice Hall,1992
[2] Robert Sedgewick: "ALGORITHMS, 2nd ed.", Addison-Wesley Publishing Company,Inc.,1988
[3] 伊藤哲郎: "情報検索 ",ソフトウェア講座 19,昭晃堂,1985
[4] 菊池忠一: 電子情報通信学会論文誌 D-I Vol.J75-D-I No.9, "日本語文書用高速全文検索の一手法,電子情報通信学会,1992.9