# 語句切り出しを用いた全文検索の手法

7E－7

池田 恵美　和田 久美子　長坂 篤

沖電気工業（株）研究開発本部

## 1 はじめに

直接本文を検索できるという利点から，全文検索システムへの期待が高まっている．しかし一般には，全文検索を効率良く行うために作成するインデックスファイルのサイズが大きくなる傾向にある．その結果，ファイルアクセスの回数が増え，かえって検索効率を悪くする．いかに，検索漏れを起こさず，インデックスファイルのサイズを小さくするかがメモリ効率の面だけでなく，検索速度の向上にもつながる．

現在，形態素解析を用いて不要語と登録語を抽出する方法が主流であるが，辞書の性能に左右され，辞書の保守も必要である．

我々は，このような問題から，形態素解析を使わずに文章を切り出して不要語を削除し，高速でメモリ効率が高い方式の考案を試みた．本論文ではその結果を報告する．

## 2 全文検索方式 FTS

### 2.1 概要

我々の開発した全文検索システム FTS は，文字種や区切り記号によりある程度意味のあるまとまりの文字列を抽出する．これに2種類の圧縮を施したシグナチャを生成し，インデックスファイルを作る．検索は検索文字列より生成したシグナチャをインデックスとして，検索をすすめる．

### 2.2 構造

一般的な日本語文書から抽出した文字列において，先頭から4文字目までの部分文字列の組合せパターンで，出現位置をかなり絞り込むことが出来る [1]．よって抽出した文字列から生成したシグナチャのうち，先頭から4文字目相当の部分シグナチャを，メモリ上においた参照テーブルのインデックスとして利用する．先頭から2文字目相当の部分シグナチャ( $Sig_{12}$ ) を TABLE1 のインデックスに，3～4文字目相当の部分シグナチャ( $Sig_{34}$) を TABLE2 のインデックスとし，残りの文字列相当の部分シグナチャ( $Sig_{rest}$) と位置情報テーブルのみをファイルに格納しておく (図1参照)．これによりファイルアクセスの回数を減らすことができる．

また，任意の文字列の検索を可能にするため，抽出文字列を先頭から1文字ずつずらした部分文字列について，すべてインデックスを作成している．

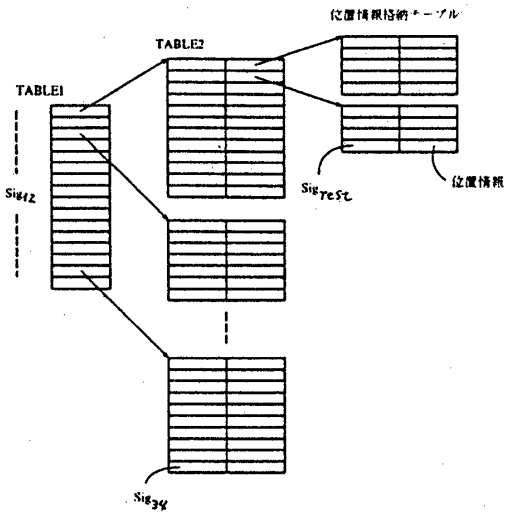


図 1: インデックスファイルの構造

### 2.3 文字列抽出法

形態素解析を使わずに任意の文字列検索を行うのに用いられるデータ構造に PAT Tree [2] がある．これは文章全体を1つの長い文字列とみなし，先頭から1文字ずつずらして得た部分文字列を符号化してインデックスを生成する．このインデックス生成法では，誤検索や検索漏れは全くないが，その文字列に一意性があらわれるまでの情報は，実際の検索には入力されない文字を含んだ部分までである場合が多く，そこまでの情報が必ずしも必要であるとは限らない．ある程度意味のある文字列をインデックス登録文字列とした方が良い．

日本語文書では文字種や記号によって区切った文字列は，ある程度の意味のあるまとまりとなるので，それを利用する．この方法は形態素解析のような複雑な処理を行わないため，抽出に要する時間が短い．

以下のような条件を切り出し条件として，原文より文字列を抜き出す．

(1) 句読点，感嘆符，スペース，括弧などの記号は不要な文字として，削除する．
(2) 平仮名，カタカナの次に出現した漢字，数字．
(3) 数字列の次に “年”“個”“本”“円” など単位を表す語が来た時，それを一まとまりの文字列として抽出．

この条件を用いた区切り結果の例を示す．

次の | アルゴリズムは | 1995 年に | 開発された | .

百科事典より任意の95項目を取り出した文書 (54,512byte，以後 “事典データ”) に対し，上記の条件を用いて切り出しを行なうと，約 15 %の不要な文字を削除できた．

---

A method of fast Full-Text-Search with simple string extraction

Emi Ikeda, Kumiko Wada, Atsushi Nagasaka

Oki Electric Industry Co., Ltd., Research & Development Group

### 2.4 シグナチャの生成

（1）全角文字は2バイトで表現されるので，長さ $n$ の文字列には $2n$ バイトの格納領域が必要である．

これに対し，抽出文字列の構成情報を付加することで，各文字の冗長的な部分を取り除くことができ，それよりシグナチャを作成した．この方法によれば，抽出文字列を約半分から1／3で表すことが出来る．

（2）漢字の種類は非常に多く，また一つの文書で使用される漢字の種類も多い．これは，漢字そのものがその文書での出現位置を示すための情報を多分に含んでいると解釈できる．それにひきかえ，平仮名は51種類と数も少なく，また文書内での出現頻度も高く1文字で出現位置を絞りこむ情報は少ない．

上記の方法で文字列を抽出すると，長い平仮名文字列を含んだ文字列が抽出されることがあるが，この平仮名部分は助詞や動詞の活用語尾がほとんどであるため，位置の特定には情報が少ない．

そこで，この部分平仮名文字列は4文字ごとに論理和を取り，5文字以上続く平仮名文字列を畳み込む格納方式を採用した．畳み込みによる別の平仮名文字列との衝突は，事典データの場合，平仮名を含む文字列3712のうち22例(2.9 %)あったが，原文から抜き出した状態の文字列と比較すると漢字部分が異なるため衝突はなかった．また，平均9バイト必要な情報領域を4バイトに圧縮できる．ゆえに，このシグナチャによる圧縮法は実用に耐えうるものと思われる．

### 2.5 検索方法

(1) あらかじめ TABLE1，TABLE2 はメモリ上にロードしておく．
(2) 検索文字列よりシグナチャを生成する．
(3) 先頭から2文字目相当の部分シグナチャ $Sig_{12}$，3～4文字目相当の部分シグナチャ $Sig_{34}$ を用い，TABLE1，TABLE2 を参照する．参照先には次のテーブルへのポインタが入っているが，該当場所がない場合は，NULL をかえし検索終了．
(4) ファイルより該当位置情報格納テーブルをとりだし，残りの文字列相当の部分シグナチャ $Sig_{rest}$ をインデックスに位置情報を得る．

## 3 方式評価

本方式(FTS Ver.3)の評価のために，FTS Ver.1 [3] との比較を行なった．FTS Ver.1 は改良 PAT Tree のインデックス構造をもつ検索方式である．

性能評価は 同一環境(HP755/99 上で同一文書データ)で行い，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
検索速度の比較を図2に示す．ファイルアクセスの回数の減少が検索速度の向上に如実にあらわれている．

(2) メモリ効率
メモリサイズを表1に示す．Ver.3 のインデックスサイズは元データの約3.8倍で，メモリ効率が高い

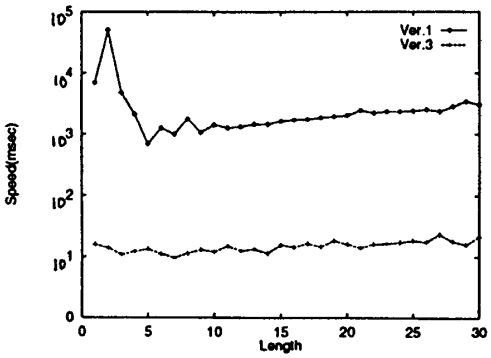


図 2: 検索速度の比較

と思われる．この方式は，インデックス構成上，固定サイズ部分が多く格納されているため，データに対して線形ではなく，大きいデータほどその係数は小さくなる．

表 1: メモリ効率

| | サイズ (Byte) |
|---|---|
| 元データ | 54,512 |
| FTS Ver.3 | 211,674 |
| FTS Ver.1 | 611,344 |

以上から，FTS Ver.3 は実行速度及びメモリ効率の点から優れていることがいえる．

## 4 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．実用化に向けて，今後は次のような機能拡張，性能向上を行なっていく予定である．

- 文字列抽出規則の精緻化
- インデックスファイルの一層の圧縮
- あいまい検索の検討

## 参考文献

[1] 菊池忠一: "日本語文書用高速全文検索の一手法" , 電子情報通信学会, 電子情報通信学会論文誌 D-I Vol.J75-D-I No.9,1992.9
[2] Williams B.Frakes, Richard Baeza-Yates ed.: "Information Retrieval - Data Structures & Algorithmhs -", Prentice Hall,1992
[3] 堀川 (池田) 他: "高速全文検索の一手法", 情報処理学会 第48回全国大会論文集 4E-2,
[4] 細野公男: "情報検索理論・技法の問題点とその解決の方向", 情報処理学会 FI 24-5 ,1991
[5] 長尾 他: "大規模日本語テキストのnグラム統計の作り方と語句の自動抽出" , 情報処理学会 NL 96-1,1993