データベースシステム　122－56
(2000. 7. 28)

# 二段階圧縮法を用いた大規模テキストファイルの検索と圧縮

大塚真吾　　宮崎収兄

千葉工業大学大学院　工学研究科　情報工学専攻
千葉県習志野市津田沼 2-17-1
Tel 047-478-0541
{otsuka, miyazaki}@mz.cs.it-chiba.ac.jp

あらまし　本稿では二段階圧縮法を用いた大規模テキストファイルの検索と圧縮について述べる。通常、テキストはハードディスクなどの二次記憶装置に格納されるため、その容量の節約のため何らかの圧縮が施されていることが多い。それら圧縮テキストへの検索は復号処理を伴うため高速化が困難である。一方、検索の高速化にはあらかじめインデックスなどを用いる方法がある。しかし、インデックスを用いることにより必要な記憶容量が増加する。我々はインデックスを用いてテキストファイルを符号化し、そのファイルを更に他の符号化法で符号化を行うことにより、圧縮率が良く高速な検索が可能となる二段階圧縮法を提案した。本稿ではこの方式が、新聞や雑誌データのような、大規模テキストファイルに対し有効であるか検討し、日本語で書かれた新聞データでの評価を示す。

キーワード　テキスト検索、テキスト圧縮、高速検索

# Search and compression of large amount of text files with two stage compression method

Shingo Otsuka and Nobuyoshi Miyazaki

Department of Computer Science,
Chiba Institute of Technology

2-17-1 Tsudanuma, Narashino, Chiba, Japan
Tel +81-47-478-0541
{otsuka, miyazaki}@mz.cs.it-chiba.ac.jp

Abstract

In this paper, we discuss search and compression of large amount of text files with two stage compression method. The texts are usually stored in secondary storage, and they are frequently compressed for file size saving. When we search compressed text files, it is usually necessary to decode them before search. Therefore, search is time consuming. On the other hand, we can use indexing for fast search. But indices consume extra amount of secondary storage. We proposed a two-stage compression method to improve the performance. It compresses text files using index files and compresses the result again with another algorithm. This paper discusses application of the two-stage compression method for large amount of text files such as newspaper and magazine data, and proposes an improved method.

key words　text search, text compression, efficient search

# 1 はじめに

インターネットの普及により新聞や雑誌などのバックナンバーや電子図書館などのテキスト文書を容易に検索や閲覧できるようになった。このように電子化されたファイルの保存と利用には大きく分けて二つの技術がある。データの保存には「圧縮技術」が必要であり、ユーザが要求する文章を検索するためには「検索技術」が必要である [11]。

現在使われている検索システムの多くは索引を使って高速な検索が可能となっている。このような検索システムの評価は一般的に検索速度で決められることが多い。しかし、記憶容量が同じ場合、保存できるテキストの数が多い方が良いと言え、検索の高速化のための索引などで多くの記憶容量を消費したのでは優れたシステムとは言えない。

検索の高速化には索引ファイルなどの二次情報ファイルが必要であり、保存できるテキストの数を増やす為にはファイルを圧縮する事が必要である。また、その場合には検索時間にファイルの復号時間も考慮に入れなければならない。このように、ファイルの容量と検索時間にはトレードオフが成り立っている [9]。

そのような問題を解消するために、我々は索引ファイルを用いた二段階圧縮法を用いて検索効率と良好な圧縮率を両立させることを提案した [5, 6]。この提案では対象ファイルと索引ファイルが一対一に対応しており、大量の文書を圧縮するのに適しているかどうか議論の余地がある。そこで、本稿では新聞や雑誌データなど比較的小さいファイルが大量にある場合の二段階圧縮法の適応について述べる。

本稿ではまず、二段階圧縮法について述べ、次に大規模データへの適用について述べる。最後に日本語で書かれた新聞データを用いた実験結果について述べる。

# 2 二段階圧縮法

この章では以前提案した二段階圧縮法の概要とその実験結果について述べる。また、対象となるファイルは日本語ファイルとする。漢字コードは2バイトコードで表現し、一般的には「JIS」「s-jis」「EUC」「Unicode」の4種類のコードが使われているが、ここでは「EUC コード」を用いる。

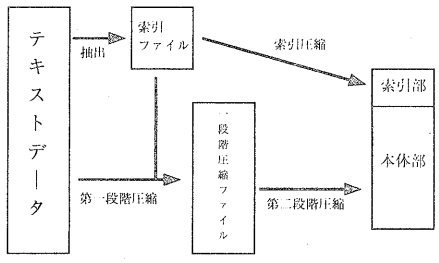


図 1: 圧縮の概要

## 2.1 基本原理

二段階圧縮法は大量の文書に対しキーワード検索を行う事を目的としている。検索対象となるファイルの全てに対しシーケンシャルサーチを行うと検索時間がかかってしまう。そこで、索引を用いた検索を行う方式が用いられる。この方式の欠点は検索に用いる索引ファイルがあるため圧縮率が良くないという点である。

索引ファイルは元のファイルから単語を抽出して作成しているため、元のファイルの部分集合と考える事ができる。したがって、索引ファイル中の情報は全て元のファイル中にあるため、この冗長性が圧縮率の低下につながっていると考えられる。そこで、この冗長性を少なくするために、索引ファイル中の単語の位置情報を用いて元のファイルを圧縮する。このファイルと索引ファイルを更に他の符号化法で圧縮を行う事により、圧縮率を高める事ができる。両ファイルの冗長性が少なくなっているので、両ファイル圧縮後のサイズの合計は原理的には元のテキストを単独で圧縮したものと同程度になることが期待できる。

## 2.2 処理方式

二段階圧縮法は図1のように索引ファイルを生成し、第一段階圧縮、第二段階圧縮、索引圧縮の計三回の圧縮を行う。また、符号化されたファイルは索引部と本体部とに分かれ、索引部は検索に用いることができる。

- 索引ファイル

  テキストデータから単語を抽出し索引ファイルを作成する。単語の抽出には形態素解析ツールを用いる。索引ファイル中の単語の順序は次の処理である第一段階圧縮を行う前であれば、どのような順番でもか

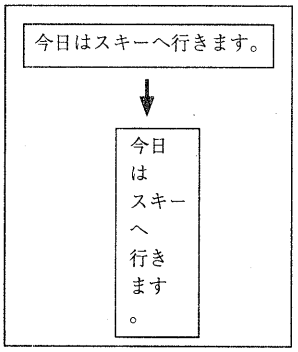


図 2: 索引ファイルの例

まわない。また、単語が重複する場合、索引ファイルには一度だけ登録する。索引ファイルの例を図 2 に示す。

・第一段階圧縮

索引ファイル中の単語の位置情報を用いてその単語の符号化を行う。単語の位置情報とは索引ファイル中の単語に 2 進数の通し番号を付けたものである。符号語の識別を考慮すると、符号語長が 2 バイトの場合、約 3,500 語、符号語長が 3 バイトの場合、約 90 万語の通し番号を表現することができる。また、符号語長以下の単語や符号語が表現できる通し番号より大きな単語については符号化を行わない。

符号語長が 2 バイトの場合、表現できる通し番号の数が約 3,500 と少ない。そこで、索引ファイル中の単語の順序を変えることにより符号化の優先順位を変更すれば、圧縮率を向上する事が出来る。一般的には単語の文字数が大きいものや出現頻度の高いものを優先して符号化することにより圧縮率が向上すると考えられる。符号化する単語の優先度ついてまとめたものを表 1 に示す。

符号語長が 2 バイトの場合の符号の例を図 3 に示す。索引ファイル中で符号語長より長い単語は"スキー"で通し番号"2"である。符号語の識別を考慮すると符号語は 16 進数で"8002"となる。この例では、索引ファイルの容量を考えると実際には圧縮されていない。しかし、文章中には同じ単語が何度も頻繁に使われているので、文章がある程度大きくなれば圧縮率は良くなる。

表 1: 2 バイト符号語方式における索引ファイル中の単語の順序

| | 符号化する単語の優先度 |
|---|---|
| 大優先方式 | 単語の文字数（大きい順） |
| 小優先方式 | 単語の文字数（小さい順） |
| 頻度方式 | 頻度順 |
| 出現方式 | 出現順 |

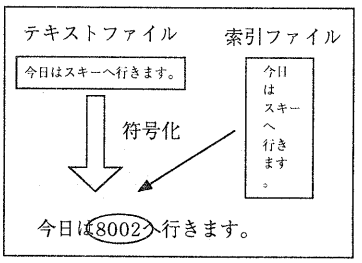


図 3: 符号化の例

・第二段階圧縮、索引圧縮

第二段階圧縮は第一段階圧縮によって生成された一段階圧縮ファイルを他の高性能な符号化法で符号化を行う。これにより、更に圧縮率を高める事ができる。索引圧縮は検索時間を重視する符号化法で圧縮を行う。そのため、ここでは復号時間が短い符号化法または、直接検索が行える符号化法で符号化を行う。また、索引部の圧縮は検索時間に重点を置く場合は圧縮を行わないこともできる。このように、第二段階圧縮、索引圧縮とも、既存の符号化方式を取り入れているので、新たに高性能な符号化方式が提案されても柔軟に対応することができる。

## 2.3 二段階圧縮法の実験結果

二段階圧縮法の評価は文献 [6] で述べた。ここではファイルサイズが 100 キロバイトと 2 メガバイトの圧縮結果の例について表 2,3 に示す。第二段階圧縮と索引圧縮には gzip を用いた。gzip を用いた理由については、

・復号時間が速く索引部の復号検索方式に適している。
・gzip は LZ77 符号化法の一種で第二段階圧縮に適し

表 2: 100 キロファイルの結果

| 圧縮方式 | 圧縮率 | 復号時間 | 検索時間 |
|---|---|---|---|
| 大優先方式 | 44.5% | 0.04 | 0.02 |
| 小優先方式 | 44.8% | 0.04 | 0.02 |
| 出現方式 | 44.5% | 0.04 | 0.02 |
| 頻度方式 | 44.8% | 0.04 | 0.02 |
| 3バイト方式 | 53.2% | 0.04 | 0.02 |
| gzip | 49.7% | 0.01 | 0.03 |

ファイルサイズ　102,434 バイト

表 3: 2 メガファイルの結果

| 圧縮方式 | 圧縮率 | 復号時間 | 検索時間 |
|---|---|---|---|
| 大優先方式 | 49.9% | 0.56 | 0.03 |
| 小優先方式 | 50.1% | 0.55 | 0.03 |
| 頻度方式 | 48.0% | 0.52 | 0.03 |
| 出現方式 | 50.2% | 0.58 | 0.03 |
| 3バイト方式 | 43.7% | 0.59 | 0.03 |
| gzip | 48.7% | 0.15 | 0.17 |

ファイルサイズ　2,081,612 バイト

ている。
・圧縮率が良い。

があげられる。また、比較対象として単に gzip で圧縮を行ったものを用いた。gzip はレンペル・ジブ動的辞書法に基づいた符号化法である。

符号化する単語の優先度について比較を行うと、ファイルサイズが小さい時は 3 バイト方式が他の方式よりも劣るが、ファイルサイズが大きくなると圧縮率が良くなっている。一方、gzip だけで圧縮したものと比較すると、二段階圧縮法は圧縮率と検索時間とも良好な結果を示している。

二段階圧縮法はファイルのサイズがある程度大きくなると、検索時間や圧縮率が良くなる。また、復号時間は gzip だけで圧縮したものより劣っている。これは、本方式が第二段階圧縮である gzip の他に第一段階圧縮の復号を行っているためである。

# 3　二段階圧縮法の大規模データへの適応

この章では二段階圧縮法を新聞や雑誌のような、テキストデータが大量にあるシステムを考えた時のシステム構成について検討する。

## 3.1　新聞や雑誌の保存

新聞や雑誌のデータは発行された日付、号数など時間的な分類と、記事の内容などジャンル別の分類でまとめられている。ここでは、時間的な分類を前提にして述べる。このようなデータへの検索は索引を使って高速に行う事ができる。保存スペースの削減を考えた場合、元のテキストには何らかの圧縮が施される。元のテキストを圧縮した場合、検索時間には検索結果に該当するファイルの復号時間が含まれる。

## 3.2　二段階圧縮法の問題

二段階圧縮法の特徴は前章で述べたようにファイルのサイズがある程度大きくなると、

・検索時間や圧縮率が良くなる。
・復号時間は gzip だけで圧縮したものより劣る。

という点が挙げられる。一方、新聞や雑誌のデータは一号単位ではファイルサイズは大きくても 1 メガ以下になる。二段階圧縮法を新聞データに適応した場合、

・数号分をまとめて圧縮するする。(圧縮率の向上)
・号ごとに圧縮する。(復号時間の短縮化)

の方式が考えられる。前者を一括方式、後者を個別方式とし、それぞれの方式について図 4,5 に示す。一括方式はファイルが大きいため、圧縮率、検索時間とも良くなると考えられるが、復号時間が増加し、結果として復号時間を含む検索時間が増加すると考えれらる。一方、個別方式は一部の号だけ復号すれば良いと仮定すれば復号時間が速いが、号ごとに全ての索引を検索しなければならず、検索時間がかかる。このように、新聞や雑誌のデータに二段階圧縮法をそのまま応用したのでは、検索時間と圧縮率の両方を良くすることは困難だと言える。

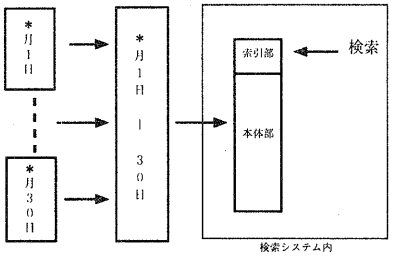


図 4: 一括方式

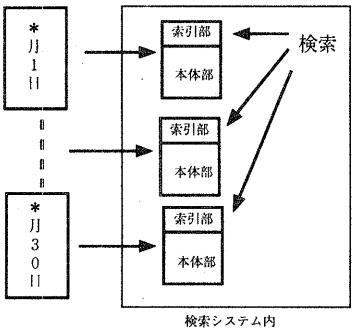


図 5: 個別方式

### 3.3 索引ファイルの改良

新聞や雑誌には「経済」、「社会」など、日付に関係なく毎日使われる単語や、「選挙」、「梅雨」などある期間に頻繁に使われる単語などがある。一日ごとに二段階圧縮を行うと、このような単語がどの索引ファイルにもあることになり、効率の良い索引であるとは言えない。30 日分の新聞データを一括して二段階圧縮した場合と一日ごと二段階圧縮を行った 30 日分の索引ファイルの大きさを表 4 に示す。一日ごとに圧縮したものが一括で圧縮したものに比べ索引ファイルの大きさが約 2.5 倍になっている。このように、日ごとに圧縮を行った索引ファイルの中には重複単語が大量にあることが分かる。

そこで、索引ファイルは何号かまとめたものを使い、その索引ファイル中の単語の位置情報を使って、一号ごとに圧縮を行えば、圧縮率と検索時間の向上が期待できる。索引ファイル中の単語にはどの号に使われているのかの情報が必要となる。そこで、索引ファイル中の単語にそ

表 4: 索引ファイルの大きさ

| 圧縮するファイル | 索引ファイルサイズ |
|---|---|
| 一日ごと | 603,804 byte |
| 一括 | 242,806 byte |

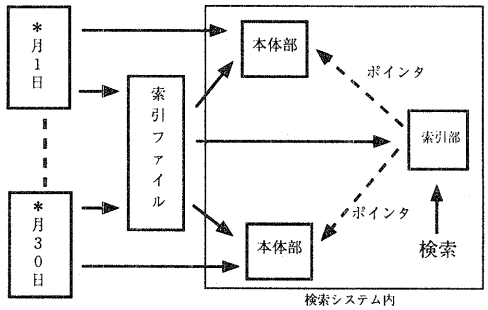


図 6: 一括索引方式

の単語が使われているファイルへのポインタを持たせるように改良した方式を図 6 に示す。この方式を一括索引方式と呼ぶ。ポインタとはポインタの識別子とその単語がどの号で使われているかを示すビット列とで構成される。図 7 の例では'0' の場合その号に単語が無いことを示し、'1' の場合にはその単語が有ることを示している。検索時にはこのポインタを使って、その単語がどの日付に使われているか判別し、そのファイルを復号する事ができる。一括索引付き二段階圧縮で検索システムを作成する手順は以下のようになる。

1. 数日分のファイルをまとめた索引ファイルを作成する。
2. 索引ファイルの単語の位置情報を用いて一日ごとに第一段階圧縮を行う。その時使用した単語のポインタにはフラグを立てる。
3. 第二段階圧縮と索引圧縮を行う。

また、検索の手順は以下のようになる。

1. 索引部の検索を行う。
2. 検索語に該当したファイルのポインタを調べ、フラグが立っている日付けのファイルを復号する。

従来の圧縮方式でも一括索引を使った検索の効率化が

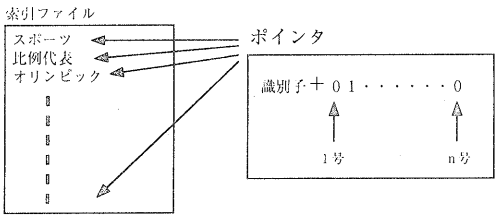


図 7: ポインタ

行われている [1]。本方式では従来の一括索引＋圧縮方式と比較し、検索時間はほぼ同等だが圧縮率が良い。

## 4 実験結果

この章では、日本語の新聞データに対して行った実験結果について、

- 一括方式、個別方式、一括索引方式の比較
- 通常行われている、元ファイルを圧縮した索引ファイル検索方式との比較

の点から述べる。

実験に用いたファイルはインターネット上からダウンロードした新聞記事 30 日分で約 3 メガバイトの容量である。本方式の第一段階圧縮の符号語長は 3 バイトとし、第二段階圧縮と索引圧縮にはに gzip を用いた。gzip はレンペル・ジブ動的辞書法に基づいた符号化法である。gzip を選んだ理由として、

- 復号時間が速く索引部の復号検索方式に適している。
- gzip は LZ77 符号化法の一種で二段階圧縮に適している。
- 圧縮率が良い。

があげられる。

実験は圧縮率、全体復号時間、平均復号時間、実際の検索時間の 3 項目で行った。圧縮率とは圧縮ファイルの容量と元ファイルの容量との比である。全体復号時間とは全てのファイルを復号した場合の時間であり、平均復号時間とは 1 ファイルあたりの復号時間である。実際の検索時間とは検索結果に該当するファイルを復号する時間を考慮に入れた検索時間である。復号時間の測定はファイルを復号しながら文字照合を行う"zgrep"コマンドを用いた。

表 5: 一括方式、個別方式、一括索引方式の比較

| 圧縮方式 | 圧縮率 | 全体復号時間 | 平均復号時間 | 実際の検索時間 |
|---|---|---|---|---|
| 一括方式 | 42.8% | 0.81 | ——— | 0.85 |
| 個別方式 | 52.4% | 1.36 | 0.05 | 0.59-1.90 |
| 一括索引方式 | 44.0% | 1.52 | 0.05 | 0.10-1.57 |

時間の単位は秒

ファイルサイズ 2,924,220 バイト

表 6: gzip のみの結果

| 圧縮方式 | 圧縮率 | 全体復号時間 | 平均復号時間 | 実際の検索時間 |
|---|---|---|---|---|
| 一括方式 | 48.6% | 0.23 | ——— | 0.47 |
| 個別方式 | 49.8% | 0.33 | 0.01 | 0.80-1.12 |
| 一括索引方式 | 53.4% | 0.33 | 0.01 | 0.04-0.36 |

時間の単位は秒

ファイルサイズ 2,924,220 バイト

実験は *Linux* 上で行い、時間の測定には *time* コマンドを使用した。また、形態素解析のツールとして NTT が無償で提供している「すもも」を用いた。

### 4.1 一括方式、個別方式、一括索引方式の比較

それぞれの方式の結果について表 5 に示す。実際の検索時間は検索結果に該当するファイルが 1 つの場合と、全てのファイルが該当する場合の結果について示している。圧縮率の面からみると、一括方式、一括索引方式、個別方式の順になる。個別方式は圧縮率が他の 2 つよりかなり悪いことが分かる。全体復号時間は一括方式、個別方式、一括索引方式の順に遅くなっている。実際の検索時間は検索結果に該当するファイルが少ない場合は一括索引方式が一番速くなる。

### 4.2 他方式との比較

比較対象として gzip だけで、

1. 30 日分の記事を一括して圧縮
2. 一日ごとに圧縮
3. 2 に一括索引ファイルを付けたもの

表 7: 総合評価

| 圧縮方式 | | 圧縮率 | 実際の検索時間 |
|---|---|---|---|
| 二段階 | 一括方式 | 1 | 6 |
| 二段階 | 個別方式 | 5 | 3 |
| 二段階 | 一括索引方式 | 2 | 2 |
| gzip | 一括方式 | 3 | 5 |
| gzip | 個別方式 | 4 | 4 |
| gzip | 一括索引方式 | 6 | 1 |

該当ファイルが少ない場合　　単位：　位

の 3 つで実験を行った。その実験結果を表 6 示す。圧縮率の面からみると 1 が良く、実際の検索時間の面からみると 3 が良い。

この結果と二段階圧縮法の結果 6 つを比較した結果を表 7 に示す。圧縮率を重視するには二段階一括方式が良いと言える。一方、実際の検索時間の高速化を重視するには gzip 一括索引方式が良いと言える。しかし、二段階一括方式は実際の検索時間が一番悪く、gzip 一括索引方式は圧縮率が一番悪い。従って、本稿の目的である圧縮率と検索時間の両立ができていない。一方、二段階一括索引方式は圧縮率、検索時間それぞれ 2 位であり圧縮率と検索時間の両立という面から見るとこの方式がもっとも優れた方式だと言える。

## 5　おわりに

本稿では新聞や雑誌データのような比較的小さなファイルが大量にあるシステムでの検索時間と圧縮率の向上について述べた。このようなシステムに二段階圧縮法をそのまま適応すると圧縮率と検索時間の両立が難しい。そこで、複数のファイルを一括した索引を用いる事によって検索時間と圧縮率の向上を図り、30 日分の新聞データを用いた実験を行い良い結果を得ることができた。

本稿では符号化時間について述べなかったが、符号化時間は他の方式と比べてまだ時間がかかっており改良中である。しかし、符号化は必要に応じて行えば良いので重大な問題になることはない。今後はより大量の新聞データ対して有効であるか検討していきたい。

## 参考文献


[1] P. Ingwersen. 情報検索研究. トッパン, 1995.

[2] 松本光崇, 角田達彦, 松本裕治. 圧縮ファイルへの直接照合を可能にする符号化法の提案. 電子情報通信学会論文誌 A, Vol. J80-A, pp. 969–976, 1997.

[3] 宮崎正路, 深町修一, 竹田正幸, 篠原武. 圧縮テキストに対するパターン照合機械の高速化. 情報処理学会論文誌, pp. 2638–2648, 1998.

[4] A. Moffat. Word-based text compression. *Software-Practice and Experience*, Vol. 19-2, pp. 185–198, 1989.

[5] 大塚真吾, 宮崎収兄. 検索効率を考慮したテキストファイル圧縮の検討. 情報処理学会第 5 9 回全国大会, Vol. 3, pp. 77–78, 1999.

[6] 大塚真吾, 宮崎収兄. 高速検索を可能とする日本語テキストの二段階圧縮法. *DEWS2000*, 2000.

[7] 大塚真吾, 高谷健文, 宮崎収兄. テキストファイルにおける圧縮率と検索効率の向上. 情報処理学会第 5 2 回全国大会, Vol. 4, pp. 221–222, 1996.

[8] 定兼邦彦, 今井浩. 転置ファイルおよび接尾辞配列の効率的圧縮法. 電子情報通信学会データ工学研究会, Vol. 40, pp. 57–62, 1999.

[9] 須藤真理. 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. 情報処理学会情報学基礎研究会, Vol. 40, pp. 33–40, 1995.

[10] 多々納勉, 大塚真吾, 宮崎収兄. 圧縮ファイルに直接検索を行う一手法. 情報処理学会第 5 6 回全国大会, Vol. 1, pp. 416–417, 1998.

[11] 植松友彦. 文書データ圧縮アルゴリズム入門. ＣＱ出版, 1994.

[12] J. Zobel and A. Moffat. Adding compression to a full-text retrieval system. *Software-Practice and Experience*, Vol. 25-8, pp. 891–903, 1995.