高速検索を可能とする日本語テキストの二段階圧縮法

大塚真吾[†] 宮崎収兄[††]

本論文ではテキストファイルの圧縮と検索について述べる．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ともな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索引などを用いる方法がある．しかし，索引を用いることにより必要な記憶容量が増加する．我々は索引を用いてテキストファイルを符号化し，そのファイルをさらに他の符号化法で符号化を行うことにより，圧縮率が高く高速な検索が可能となる二段階圧縮法を提案した．本論文では日本語テキストの二段階圧縮法とその有効性について論じる．

Two Stage Compression for Efficient Japanese Text Search

Shingo Otsuka[†] and Nobuyoshi Miyazaki[††]

In this paper, we discuss compression and search of text files. 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 applies the two stage compression method to Japanese text files and discusses its effectiveness.

1. はじめに

インターネットの普及やハードディスクの大容量化により個人でも大量なテキストファイルを保持することが容易になっている．テキストファイルの保存と利用には大きく分けて２つの技術が重要である．ファイルの保存には「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[1),11)]．

この２つの技術の両立を考えた場合，何らかの圧縮が施されたテキストファイルに対して検索を行うことを考えなければならない．圧縮されたテキストファイルに対して検索を行う場合，通常はその圧縮ファイルを復号してから検索を行うため検索時間がかかる．この解決策としては，直接検索できる符号化法を用いる方法[2),3)]がある．これは復号を行ってから検索を行うより検索時間は速くなるが，圧縮率は高性能な符号化法に劣ってしまう．

一方，索引ファイルをあらかじめテキストファイルから抽出しておき，元のファイルを圧縮しておく方法もある．これはインターネットなどの検索エンジンなどで使われており高速な検索が可能である．しかし，これらの方法では索引ファイルの分だけ圧縮率が悪化する．また，検索結果が該当数，URL，概要などで表示されるためファイルそのものの復号時間についてもあまり考慮されてない．一般的に検索効率を上げると圧縮率が犠牲になる傾向がある[9)]．

我々はテキストファイルの検索時間，復号時間，圧縮率の向上を目的とした二段階圧縮法[5),6)]を提案した．本論文では二段階圧縮法の原理と日本語テキストへの適用について述べる．そして，大量なテキストデータに対する二段階圧縮法の適用方法を検討し，その性能の実験と考察を行う．以下，本方式とは二段階圧縮法を指すものとする．

2. 圧縮ファイルに対する検索

一般によく使われているアーカイバは適応符号化法の一種であるユニバーサル符号化法が用いられる場

† 千葉工業大学大学院工学研究科情報工学専攻
Department of Computer Science, Chiba Institute of Technology
†† 千葉工業大学情報科学部情報工学科
Department of Computer Science, Chiba Institute of Technology

合が多い．また，英文テキストの圧縮に関してはファイルから単語を抽出し，それを用いて符号化を行う方法[4),13)]があり高い圧縮率をあげている．

テキスト検索にはシーケンシャルサーチを前提にしたものと，索引ファイルや転置ファイルを用いるものが考えられる．シーケンシャルサーチは検索する文字列が言語的に意味をなさなくても検索可能である．また，熟語などの検索にも適している．索引ファイルを用いる検索はシーケンシャルサーチより高速に検索が行える．しかし，基本的に単語単位でしか検索が行えない．

圧縮されたファイルに対して検索を行う場合，圧縮されたファイルの符号化法によって検索の仕方が異なる．静的符号化法で圧縮されたファイルは圧縮ファイルのヘッダ部分に文字と符号語との対応表があるため検索は簡単なパターンマッチングで行え復号を要さない．したがって，検索時間はヘッダ部分に書かれた符号語の復号時間が影響する．適応符号化法はある部分の符号化にそれ以前の情報を利用しているためファイルの一部を復号することが困難である．したがって，復号しながら検索を行うことになり，検索時間には圧縮ファイルの復号時間が影響する．単語を用いて符号化を行う方法は単語も符号化されているため検索には単語の復号時間がかかる．また，基本的には単語単位の検索しか行えない．このように，圧縮されたファイルに対しての検索時間の評価は復号時間を考慮しなければならない．

3. 二段階圧縮法

この章では，我々が提案した英文テキストの二段階圧縮法について述べる．

3.1 基本原理

本方式は索引を用いてキーワード検索を行うことを基本としている．索引ファイルは元のテキストファイルから単語を抽出して作成しているため，元のテキストファイルの部分集合と考えることができる．索引ファイル中の情報はすべて元のテキスト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．そこで，索引ファイル中の単語の位置情報を用いて元のテキストファイルを圧縮する．このファイルと索引ファイルをさらに他の符号化法で圧縮を行うことにより，圧縮率を高めることができる．

また，索引ファイルの符号化法は検索時間と圧縮率のどちらに重点を置くかで，適応符号化法，静的符号化法，または索引の圧縮を行わないなど任意の方式を用いることができる．

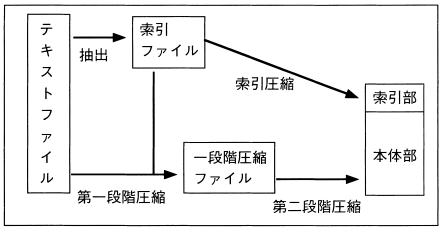

図 1 圧縮の概要
Fig. 1 The two stage compression.

本方式で索引のハッシュなどを行わない場合は索引ファイルの検索はシーケンシャルサーチになる．このとき，本方式とシーケンシャルサーチ，索引を用いた検索の検索速度を比較した場合以下のようになると考えられる．

シーケンシャルサーチ ＜ 本方式 ＜ 索引

また，これらとは別に転置ファイルに圧縮技術を取り入れた方式[8),9)]が提案されている．転置ファイルとは文章中に現れる各単語に対してその出現位置のリストを格納したものである．これらの方式では転置ファイルを用いることにより，検索時に該当テキストファイル中でのキーワード位置を知ることができる．本方式の索引を転置ファイル化することも可能であるが，本論文は圧縮率と検索時間の両立を主題としているため，ここでは転置ファイルについては論じない．

3.2 処理方式

本方式は**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 3 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．この節では索引ファイルの生成法とそれぞれの符号化処理について述べる．本方式の処理ステップは以下のようになる．

(1) 索引ファイルの作成
(2) 第一段階圧縮
(3) 索引圧縮・第二段階圧縮

3.3 索引ファイルの作成

テキストファイルから単語を抽出し索引ファイルを作成する．英文は語間にスペースやピリオドなどの文字があるため，単語の抽出が容易に行える．索引ファイル中の単語の順番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．

3.4 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．本方式の符号語長は 2 バイトまたは 3 バイトとし，符号語長以下の単語は符号化を行わない．単語の位置情報とは索引ファイル中の単語に通し番号を付けたものである．

この番号を用いてテキストファイルの符号化を行う．通し番号は 2 進数で表現する．ASCII コードでは 1 バイトコードの一番左のビットはつねに ‘0’ である．そこで，符号語は一番左のビットを ‘1’ にすることでその文字が符号語かそうでないか識別することができる．また，英文は語と語の間にスペースがあることが非常に多い．そこでその次のビットは ‘符号化される単語の次がスペースかどうか’ を判別するのに用いる．‘10...’ ならば符号語の次の文字がスペースでないことを表し，‘11.....’ ならば符号語の次の文字がスペースであることを表している．これにより，テキストファイルから大部分のスペースを取り除くことができる．

符号語の識別とスペースの有無を考慮すると，符号語長が 2 バイトなら約 16,000 単語を表現することができる．また，索引ファイル中の単語数がこの数を超えた場合，索引ファイル中の単語の通し番号がこれよりも大きい単語については符号化を行わない．一方，符号語長が 3 バイトならば約 800 万単語を表現でき，事実上すべての単語を表現できる．

ファイルサイズが大きくなれば重複する単語数が多くなり圧縮率が高くなることが予想できる．

3.5 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符号化法で符号化を行う．これにより，さらに圧縮率を高めることができる．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，検索時間に重点を置く場合は索引部の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，既存の符号化法を取り入れているので，新たに高性能な符号化法が提案されても，本方式は柔軟に対応することができる．

3.6 復　　号

復号は以下の処理手順で行う．

(1) 第二段階圧縮と索引圧縮でそれぞれ用いた符号化法の復号を行い，一段階圧縮ファイルと索引ファイルを生成する．

(2) 一段階圧縮ファイル中の記号が符号語か文字か判別し，符号語なら単語の位置情報とスペース情報を得る．

表 1 英文テキストの結果
Table 1 The result of English text file.

圧縮方式	圧縮率	復号時間	検索時間	検索方式
gzip	37.6%	0.10	0.12	zgrep
LHA	37.6%	0.20	0.20	復号+grep
二段階圧縮法（復号検索）	35.0%	0.39	0.04	zgrep
二段階圧縮法（直接検索）	35.9%	0.42	0.09	直接検索
二段階圧縮法（索引部無圧縮）	40.3%	0.32	0.01	grep

ファイルサイズ 1,379,628 byte

(3) 位置情報とスペース情報を基に文字列を出力する．

3.7 検　　索

本方式で圧縮されたファイルに対して検索を行う場合，ファイル全体を検索するのではなく索引部のみ検索を行う．索引部の検索方式としては索引圧縮の符号化法により以下のようになる．

- 復号を行いながら検索を行う．
- 索引部を直接検索する．

どちらもシーケンシャルサーチを基本としているが，ファイル全体をシーケンシャルサーチするより検索時間が短かくなる．

本方式は元のテキストファイルから単語を抽出して索引ファイルを作成している．したがって，検索できる検索語は単語単位である．熟語検索については熟語で使われている語句がすべて索引ファイル中にあるかどうかを検索することは可能だが，それらの語句が元のテキストファイル中で熟語として使われているかを判別することは難しい．

3.8 実 験 結 果

二段階圧縮法と gzip，LHA との英文テキストを対象とした実験結果について**表 1** に示す．圧縮率とは圧縮ファイルの容量と元のテキストファイルの容量との比である．実験はすべて Linux 上で行い，時間の測定には time コマンドを使用した．

gzip はレンペル・ジブ動的辞書法に基づいた符号化法であり，LHA は LZH に基づいた二段階符号化法である．zgrep とは gzip で圧縮したファイルを復号しながら文字照合を行う Linux のコマンドである．

符号語長を 2 バイトとし，第二段階圧縮は gzip を用いた．索引圧縮は復号検索を行う圧縮法に gzip，直接検索を行う圧縮法に静的ハフマンを用いた．

この結果から本方式で索引の圧縮を行うと既存の圧縮法と比べ検索時間，圧縮率とも良いことが分かる．復号時間については gzip や LHA より遅いという結果が得られた．実効検索時間（ファイルを可読形式で得られるまでの時間）は復号時間+検索時間となるので gzip などより時間がかかる．しかし，検索対象のファイル数が多く，検索結果に該当し復号するファイ

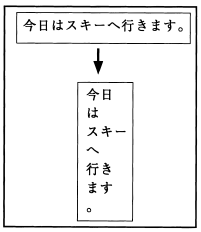

図 2 索引ファイルの例
Fig. 2 Index file.

ルが少ない場合は本方式の実効検索時間は gzip より速くなる．実効検索時間についての考察は 5 章で詳しく述べる．

また，本方式の索引圧縮の違いについて考察すると，圧縮率は復号検索が良く，復号時間と検索時間は索引部無圧縮が良い．圧縮率を重視する場合は復号検索が一番良い方法だといえる．

4. 日本語テキストの二段階圧縮法

日本語テキストの二段階圧縮法は英文テキストとの違いを考慮しなければならない．日本語テキストの特徴として，

- 単語の抽出に複雑な処理が必要である．
- 2 バイトコードである．
- 符号語と文字の識別に必要なビット数が多くなり，符号語が表現できる単語数が少なくなる．

があげられる．以下，漢字コードは “EUC” を前提として述べる．

4.1 索引ファイルの作成

日本語は「分かち書き」ではないため単語の抽出が困難である．そこで，自然言語分野で研究されている形態素解析ツールを用いて単語の抽出を行う．単語抽出の例を**図 2** に示す．

4.2 第一段階圧縮

索引ファイルを用いて第一段階圧縮を行う．索引ファイル中の単語に通し番号を付けると以下のようになる．

今日 ··· 0
は ··· 1
スキー ··· 2
.
.
. ··· 6

この番号を用いてファイルの符号化を行う．英文テキストのように符号語でスペースの「ある」「なし」の表現は行わない．

英文テキストの場合，ASCII コードのため符号語の判定には先頭のビットが ‘1’ か ‘0’ かを識別すればよい．しかし，EUC コードの場合，第 1 バイトコードのうち 16 進数で ‘80’ から ‘8D’ 以外はほとんど使用されている．符号語長が 2 バイトの場合，16 進数で ‘8000’ から ‘8DFF’ までが符号語として使用できるコードであり，約 3,500 個である．符号語長を 3 バイトにすると，表現できる単語数は約 90 万個となり，ほとんどの単語を符号化することができる．また，16 進数の ‘8’ は一番左のビットが ‘1’ であるため ASCII コードが混在する文章でも符号化を行うことができる．

表 2 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
Table 2 The order of words on two byte coding.

	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

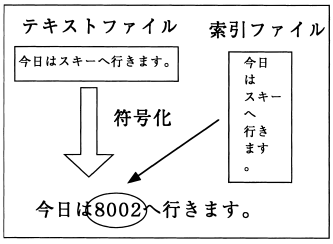

図 3 符号化の例
Fig. 3 An example of coding.

符号語長が 2 バイトの場合，単語の番号が小さいものだけを符号化することにすれば，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えることができ，圧縮率を向上することができる．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．単語の並び順についてまとめたものを**表 2** 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高くなる．

4.3 第二段階圧縮，索引圧縮，復号

第二段階圧縮，索引圧縮ともに日本語化にともなう変更点はなく，英文の二段階圧縮法と同様に圧縮を行う．復号はスペースの処理を除いては英文の二段階圧縮法と同様である．

表 3 100 キロファイルの結果
Table 3 Result of 100 KB file.

圧縮方式	圧縮率	復号時間（s）	検索時間（s）
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 byte

表 4 2 メガファイルの結果
Table 4 Result of 2 MB file.

圧縮方式	圧縮率	復号時間（s）	検索時間（s）
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 byte

4.4 二段階圧縮法の性能

日本語テキストに適応した場合について以下の実験を行った．

- 符号語長による圧縮率の変化
- 既存のアーカイバとの比較
- 索引ファイルの大きさ

実験は英文テキストと同様に Linux 上で行った．実験で第二段階圧縮と索引圧縮に gzip を用いたのは，時間測定を含めた統合的検討のできる Linux/Unix 環境で一般的によく利用され圧縮率も高いからである．なお，Windows 環境などで，より圧縮率の高いアーカイバを用いて圧縮した場合でも，二段階圧縮法の圧縮率は単にそのアーカイバで圧縮した圧縮率と同等またはそれ以上となる[7]．

実験は圧縮率，検索時間，復号時間の 3 項目で行った．また，形態素解析のツールとして NTT の「すもも」を用いた．

ファイルサイズが 100 K バイト，2 M バイトおよび 130 M バイトの新聞記事データを用いた結果を**表 3**，**表 4**，**表 5** に示す．130 M バイトは「CD–毎日新聞 99 版」の新聞記事 1 年分である．なお，130 M バイトに対する 2 バイト方式は圧縮率が低く，また圧縮時間が大きいため，表 5 では 3 バイト方式のみを示した．

2 バイト方式相互間では大きな差はなく，ファイルサイズが小さい場合は 3 バイト方式の圧縮率は他の方式よりも劣るが，ファイルサイズが大きくなると 3 バイト方式が優れている．3 バイト方式でファイルサイズが大きくなると圧縮率が向上するのは新しい単語の出現が減少し索引ファイルの割合が低下するためである．

表 5 130 メガファイルの結果
Table 5 Result of 130 MB file.

圧縮方式	圧縮率	復号時間（s）	検索時間（s）
3 バイト方式	41.4%	41.14	0.19
gzip	45.7%	10.25	10.25

ファイルサイズ 133,030,045 byte

次に，本方式と gzip との比較を行うと，表 4 は検索時間が gzip の 1/6 で復号時間が 4 倍なのに対し，表 5 では検索時間は 1/50 で復号時間は同様という結果が得られた．復号時間は gzip だけで圧縮したものより劣っているのは，本方式が第二段階圧縮である gzip のほかに第一段階圧縮の復号を行っているためである．

このように，本方式はファイルサイズが小さい場合は符号語長を 2 バイトに，大きい場合は 3 バイトにすることにより圧縮率，検索時間ともに良くなる．また，英文の場合と同様に単一ファイル数の実効検索時間は gzip に劣るが，多数のファイルを対象に検索を行う場合は実効検索時間を短くできる可能性がある．

5. 二段階圧縮法を用いたテキスト検索

テキスト検索の特徴として比較的小さなテキストファイルが大量にあることがあげられる．Web ページであればその容量は 3〜5 K バイト程度であり，単行本でも大きくて 2 M バイト程度である．

本方式をこのような環境で利用するには，

- ある程度まとめて圧縮する（圧縮率の向上）．
- 1 ファイルごとに圧縮する（復号時間の短縮化）．

の 2 つが考えられる．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**，**図 5** に示す．一括方式はファイルサイズが大きいため，圧縮率と索引の検索時間が良くなると考えられるが，復号時間が増加し，結果として実効検索時間が増加すると考えられる．一方，個別方式は 1 ファイルだけ復号すればよいと仮定すれば復号時間が速いが，ファイルが小さいため圧縮率が高くない．このように，本方式をテキスト検索に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な場合がある．

5.1 一括索引方式

ファイルサイズと索引ファイルの関係を検討するため，1 年分の新聞データから作成した索引ファイルについて**表 6** に示す．ここでファイルの個数 '1' とは 1 年分のファイルから索引ファイルを作成したことを意味し，同様に一番下の '12' とは 1 カ月ごと索引を作成したことを意味する．索引ファイルを 1 カ月ごとに作

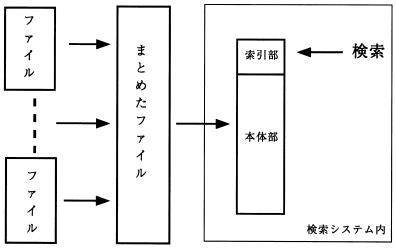

図 4 一括方式
Fig. 4 The combined method.

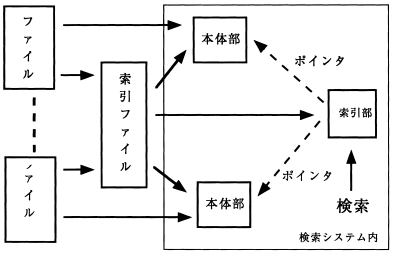

図 6 一括索引方式
Fig. 6 The combined index method.

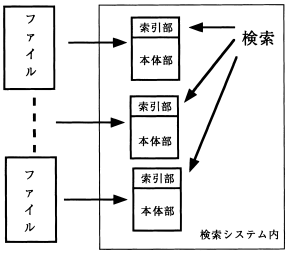

図 5 個別方式
Fig. 5 The individual method.

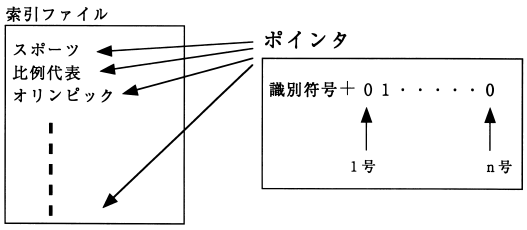

図 7 ポインタ
Fig. 7 Pointers in index.

表 6 索引ファイルの大きさ
Table 6 Size of index file.

ファイルの個数	索引ファイルの合計サイズ（byte）	検索時間（s）
1	1,921,278	0.19
2	2,978,622	0.31
3	3,639,566	0.33
4	4,200,622	0.45
6	5,144,554	0.57
12	6,437,941	0.85

ファイルサイズ 133,030,045 byte

成したものは 1 年まとめたものより 3.4 倍も大きく，検索時間も 4 倍かかる．なお，1 年をまとめた索引の単語数は約 20 万単語である．

そこで，索引ファイルはある程度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ファイル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ファイル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一括索引方式を**図 6** に示す．ポインタは識別符号とその単語がどのファイル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ないことを示し，'1' の場合にはその単語があることを示している．検索時にはこのポインタを使って，その単語がどのファイルに使われているか判別し，そのファイルを復号することができる．また，検索語が複数ある場合でもビット列どうしを論理演算することにより検索が行える．

5.2 テキスト検索の構成法の評価

新聞記事を用いて以下の評価を行った．

- 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，元のテキストファイルを圧縮した索引ファイル検索方式との比較

また，一括索引方式についてはさらに

- まとめるファイルサイズによる変化
- 索引ファイルサイズによる変化

についても評価を行った．

実験ではインターネット上からダウンロードした新聞記事 90 日分で約 5 M バイトの新聞記事 A と，前述の新聞記事 1 年分で約 130 M バイトの新聞記事 B を用いた．実験では第一段階圧縮の符号語長を前節でファイルサイズが大きいときの実験結果が良かった「3 バイト方式」とした．一括方式ではファイルをまとめて 1 つのファイルとし，新聞記事 A の個別方式と一括索引方式は 1 日ごとに分割したファイルとした．また，新聞記事 B の個別方式と一括索引方式は 10 日ごとにファイルに分割した．

実験は実効検索時間，圧縮率，全体復号時間，平均復号時間，検索時間の 5 項目で行った．ここで，実効検索時間は前述のように検索結果に該当するファイルの復号時間を検索時間に加えたもので，該当ファイル数に比例して増加する．実験結果では検索結果に該当するファイルが 1 つの場合と，すべてのファイルが該

表 7 一括方式，個別方式，一括索引方式の比較（新聞記事 A）
Table 7 Comparison of methods (newspaper A).

圧縮方式	圧縮率	全体復号時間(s)	平均復号時間(s)	検索時間(s)	実際の検索時間(s)	
					1 つ該当	すべて該当
一括方式	44.51%	1.59	—	0.04	1.63	1.63
個別方式	55.48%	2.89	0.03	2.41	2.44	5.30
一括索引方式	46.83%	3.17	0.04	0.06	0.16	3.29

ファイルサイズ 5,329,650 byte

表 8 一括方式，個別方式，一括索引方式の比較（新聞記事 B）
Table 8 Comparison of methods (newspaper B).

圧縮方式	圧縮率	全体復号時間(s)	平均復号時間(s)	検索時間(s)	実際の検索時間(s)	
					1 つ該当	すべて該当
一括方式	41.37%	41.14	—	0.19	41.33	41.33
個別方式	44.80%	42.23	1.17	1.76	2.93	43.99
一括索引方式	41.51%	48.98	1.36	0.25	1.83	49.45

ファイルサイズ 133,030,045 byte

表 9 gzip の結果（新聞記事 A）
Table 9 Result of gzip (newspaper A).

圧縮方式	圧縮率	全体復号時間(s)	平均復号時間(s)	検索時間(s)	実際の検索時間(s)	
					1 つ該当	すべて該当
gzip 一括方式	47.93%	0.45	—	0.49	0.94	0.94
gzip 個別方式	50.09%	0.71	0.01	2.45	2.46	3.16
gzip 索引方式	53.73%	0.71	0.01	0.06	0.07	0.83

ファイルサイズ 5,329,650 byte

表 10 gzip の結果（新聞記事 B）
Table 10 Result of gzip (newspaper B).

圧縮方式	圧縮率	全体復号時間(s)	平均復号時間(s)	検索時間(s)	実際の検索時間(s)	
					1 つ該当	すべて該当
gzip 一括方式	45.70%	10.25	—	10.87	21.12	21.12
gzip 個別方式	45.76%	10.45	0.29	11.65	11.94	22.10
gzip 索引方式	47.02%	10.45	0.29	0.25	0.54	10.70

ファイルサイズ 133,030,045 byte

当する場合の時間を示している．平均復号時間とは 1 ファイルあたりの平均復号時間である．

5.2.1 一括方式，個別方式，一括索引方式の比較

本方式の結果について**表 7**，**表 8** に示す．圧縮率は一括方式，一括索引方式，個別方式の順になり，個別方式は圧縮率が他の 2 つより低い．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り，実効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．また，ファイルサイズが大きくなると圧縮率の差が小さくなる．

5.2.2 他方式との比較

比較実験として gzip で以下のように圧縮を行った．

(1) 記事を一括して圧縮し，検索は復号しながら行う（gzip 一括方式）．
(2) 記事を 1 日ごとに圧縮し，検索は復号しながら行う（gzip 個別方式）．
(3) (2) に二段階圧縮法の一括索引方式で用いた索引ファイルを付け，検索は索引に対して行い該当したファイルのみ復号を行う（gzip 索引方式）．

実験結果を**表 9**，**表 10** に示す．ファイルの個数は表 7，表 8 と同様である．

表 9 より圧縮率は gzip 一括方式が良く，実効検索時間は gzip 索引方式が良い．また，表 10 では gzip 個別方式と gzip 一括方式の圧縮率の差が小さくなっている．

gzip と二段階圧縮法を比較すると，圧縮率は二段階圧縮法が高いことが分かる．また，検索時間は gzip 索引方式が良い．しかし，gzip 索引方式は圧縮率が最も悪い．多数のファイルを対象とした検索では結果ファイルの個数が多い場合は利用者がすべてを見ることは困難である．このため，さらに条件を追加して該当ファイルを少なくしたり，一部のファイルだけを見て次の処理を考えることが多い．したがって，結果ファイルが多い場合にはすべてを復号する必要がないことも多い．このような環境で圧縮率と検索時間の両立を図るには二段階圧縮法で一括索引方式を行うのが最良であるといえる．

5.2.3 一括索引方式の利用方法

新聞などファイルが日々増加する環境では，すべてのファイルをまとめて索引ファイルを抽出することは困難である．そこで，ファイルや索引ファイルをどの程度の大きさにすればよいか検討しなければならない．ファイルサイズに対する索引ファイルの変化は表 6 で示したが，一括索引方式ではポインタ部を考慮する必要がある．まとめる索引ファイルの日数による変化についての実験結果を**図 8**，**図 9** に示す．図 8 の「本体部の割合」とは索引部を除いたときの圧縮率である．したがって，2 つの曲線間が索引部の割合となる．図 9 の「該当なし」とは検索結果に該当するテキストがない場合で，検索時間にはテキストの復号時間を含まない．「すべて該当」では検索時間にすべてのファイルの復号時間が含まれている．

実験は新聞記事 A を 10 日ごとに分けて 1 つのファイルとした．図中の x 軸の数字は索引を作成するのに用いた新聞記事の日数である．数値が '10' ならば新聞記事データを 10 日まとめて 1 つのファイルとしているので個別方式となり，'90' ならば前節の実験と同様にすべての記事から索引を抽出していることを意味している．

実験結果より，多くのファイルから索引ファイルを

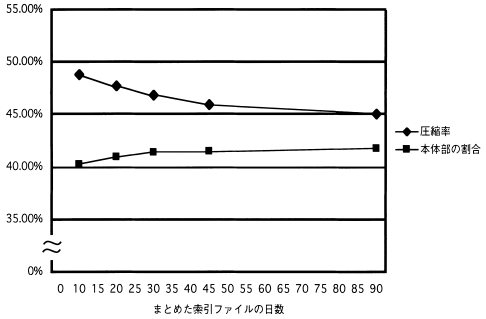

図 8 索引ファイルのまとめた日数と圧縮率
Fig. 8 The length of period and compression ratio.

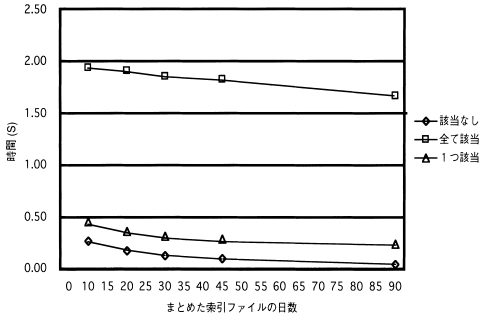

図 9 索引ファイルのまとめた日数と検索時間
Fig. 9 The length of period and search time.

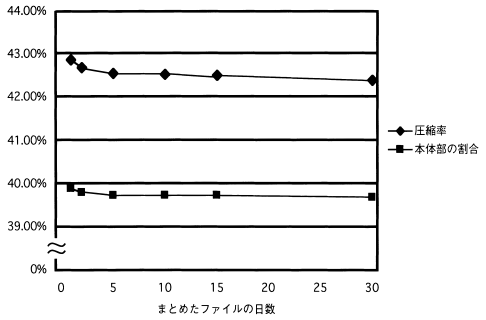

図 10 個々のファイルの大きさと圧縮率（1 カ月）
Fig. 10 The size of each file and compression ratio (month).

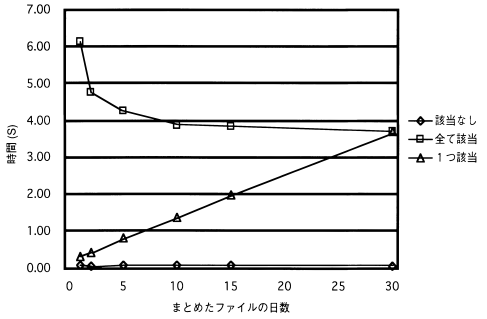

図 11 個々のファイルの大きさと検索時間（1 カ月）
Fig. 11 The size of each file and search time (month).

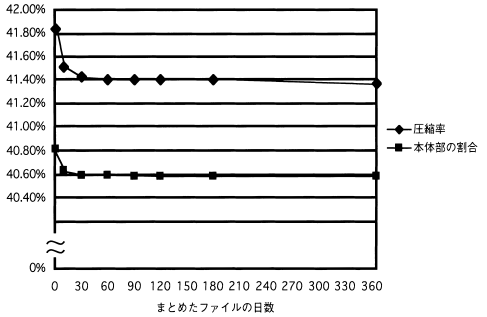

図 12 個々のファイルの大きさと圧縮率（1 年）
Fig. 12 The size of each file and compression ratio (year).

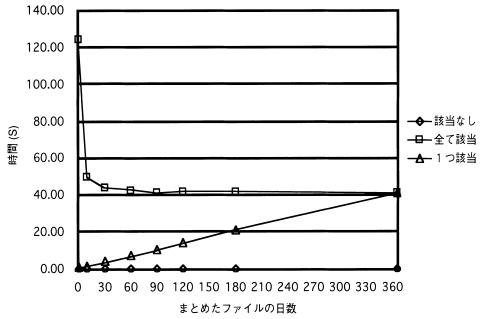

図 13 個々のファイルの大きさと検索時間（1 年）
Fig. 13 The size of each file and search time (year).

作成した方が索引ファイルの割合が減少し，圧縮率が高くなることが分かる．また，検索時間に関しては多くのファイルから索引ファイルを作成した方が検索時間が速くなる．

次に，ファイルのサイズによる圧縮率と検索時間の変化について実験を行った．索引ファイルは全体に対して作成し，ファイルを何日分かずつまとめることによりファイルサイズを変化させた．実験は新聞記事 B を用いた．

1 カ月のファイルで行った実験結果を**図 10**，**図 11** に示す．図中の x 軸はまとめた新聞記事の日数である．数値が ‘1’ ならば 1 日分のファイルが 30 個あることを意味する．また，1 年のファイルで行った実験結果を**図 12**，**図 13** に示す．

実験結果より小さいファイルが多くある場合は圧縮率が悪くなる．これは，索引内のファイルへのポインタサイズが大きくなるためである．また，検索時間に関しては小さいファイルが多くある場合，検索した結果に該当するファイルが少数であれば高速な検索が行えるが，すべて該当する場合は遅くなる．索引ファイ

ルの割合には大きな変化が見られなかった．

これらの結果より，ファイルサイズが小さすぎると圧縮率が悪化するが，ある程度大きくなれば，あまり変化しないことが分かる．また，すべて該当の場合の実効検索時間についても同様である．

6. ま と め

本論文では圧縮ファイルに対し高速検索が可能な二段階圧縮法を日本語テキストに適用した．本方式では索引ファイルの冗長性を利用した圧縮を行ってから通常の圧縮を行うことにより圧縮率の向上を図っている．本方式で生成される圧縮ファイルは本体部と索引部とに分かれているため検索は索引部のみ行えばよい．したがって，復号化して検索する方式や圧縮ファイルに対して直接検索を行う方式に比べ検索時間が短縮できる．また，この方式は第二段階圧縮と索引圧縮に既存の符号化法を取り入れているため，新たな符号化法が提案されても柔軟に対応することができる．

また，本論文では二段階圧縮法の適用方法として比較的小さなファイルが大量にある場合の有効性についても論じた．このような環境に二段階圧縮法をそのまま適用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ことによって検索時間と圧縮率の向上を図り，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．

符号化は必要に応じて 1 回行えばよいので重大な問題になることはないが，符号化時間については 130 M バイトの場合，555 秒と gzip の約 5 倍の時間がかかっており改善の余地がある．今後はより大量のテキストに対して有効であるか検討したい．そのためには，ハッシュテーブルや B+木などを用いて検索の高速化を図ることが重要だと考えられる．

参 考 文 献

1) Ingwersen, P.: 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) Moffat, A., Neal, R. and Witten, I.H.: Arithmetic coding revisted, *ACM Trans. Information Systems*, Vol.16, No.3, pp.256–294 (1998).
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 59 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，宮崎収兄：二段階圧縮法の圧縮率の検討，情報処理学会第 61 回全国大会，Vol.3 (2000).
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，情報処理学会論文誌：データベース，Vol.40-No.SIG 8(TOD 4), pp.85–94 (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 56 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) 横尾英俊：記号列の長さと位置との関係をポインタ符号化に利用した ziv-lempel 符号，電子情報通信学会論文誌 A，Vol.J78-A, pp.1175–1173 (1995).
13) Zobel, J. and Moffat, A.: Adding compression to a full-text retrieval system, *Software-Practice and Experience*, Vol.25, No.8, pp.891–903 (1995).

(平成 12 年 9 月 20 日受付)
(平成 13 年 2 月 2 日採録)

（担当編集委員 河野 浩之）

大塚 真吾（学生会員）

1973 年生．1998 年千葉工業大学大学院工学研究科情報工学専攻博士前期課程修了．現在，千葉工業大学大学院工学研究科情報工学専攻博士後期課程在学中．

宮崎 収兄（正会員）

1948 年生．1973 年京都大学理学部卒業．1977 年イリノイ大学大学院計算機科学専攻修士課程修了．沖電気工業（株），(財) 新世代コンピュータ技術開発機構を経て，現在，千葉工業大学教授（情報工学科）．工学博士．データベースシステム，知識情報処理等の研究に従事．共著書「新データベース論」（共立出版）等．電子情報通信学会，人工知能学会等会員．