ファイル検索システムのための二段階圧縮法

学生員 大塚 真 吾 (千葉工業大学大学院)

非会員 宮崎 収 兄 (千葉工業大学)

Two-stage compression for file search system

Shingo Otsuka, Student Member, Nobuyoshi Miyazaki, Non-member (Chiba Institute of Technology)

Texts are frequently compressed for file size saving when they are stored in secondary storage. 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 this method does not consider compression ratio. We consider compression algorithms for file search systems with large amount of text files. Important factors of these methods are search time, expansion time and compression ratio. This paper proposes a two-stage compression method, which compresses text files using index files and compresses the result again with another algorithm. A preliminary experiment shows good performance in terms of compression ratio and search time when the same words appear frequently. We also discuss application of the two-stage compression method for large amount of text files such as newspaper and magazine data.

キーワード : テキスト検索, テキスト圧縮, 高速検索, text search, text compression, efficient search

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事が可能となった。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⁾⁽²⁾。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事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⁽³⁾⁽⁴⁾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る。

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等で使われており高速な検索が可能である。しかし, これらの方法は検索速度に重点を置いているため圧縮率についてあまり考慮されていない。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。

本論文では新聞や雑誌, 個人で集めたテキストファイルが大量にある環境でのファイル検索システムを想定する。この場合, 検索時間, 復号時間, 圧縮率の3つを考慮しなければならない。このような研究として,

- (1) テキストファイルと転置ファイルの両者を独立した符号化法で圧縮を行う mg システム⁽⁵⁾⁽⁶⁾
- (2) テキストファイル中の単語を用いて符号化を行う方法⁽⁷⁾

などがある。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。(1)は圧縮した転置ファイルに対して検索を行うため高速な検索が可能だが高性能な符号化法と比べ圧縮率が劣る。一方, (2)は圧縮率が良く検索も高速に行えるが, 復号に時間がかかる。

本論文で提案する二段階圧縮法はテキストから索引ファイルを作成し, この索引ファイルを用いて元のテキストの符号化を行う。更に符号化された元のファイルと索引ファイルは既存の符号化法で圧縮を行う。この方法は索引ファイルを使った符号化と既存の符号化法を組み合わせる事により圧縮率を高め, 更に索引を用いて高速な検索が可能になる。このように二段階圧縮を行う事により従来の方法では困難であった高速検索と高い圧縮率の両立が可能となる。

本論文ではまず, 英語テキストファイルを対象とした二段階圧縮法を提案する。次に大量のテキストファイルを保持するファイル検索システムに対しての二段階圧縮法の適用について述べる。また, 二段階圧縮法と他方式を比較し

た実験結果について述べる。

2. 圧縮ファイルに対する検索

テキストファイルに対する圧縮は、そのファイルの性質上から可逆圧縮で行われる。可逆圧縮は大きく分けて静的符号化法と適応符号化法の2つに分類できる。

静的符号化法で圧縮されたファイルはモルス符号のように文字と符号語との対応が常に一定になっており符号語の対応が途中で変化しない。検索は圧縮ファイルのヘッダ部分に書かれた符号語を基に簡単なパターンマッチングで行え圧縮ファイルに対して復号を行わずに検索する事ができる。従って、ヘッダ部分に書かれた符号語の復号時間が検索時間に影響する。

適応符号化法で圧縮されたファイルはファイルのデータを系列として圧縮を行っている。ある部分の符号化にはそれ以前の情報を利用しているので、ファイルの一部を復号する事が困難なため圧縮ファイルに対して直接検索する事が困難である。従って、復号しながら検索を行う事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。

また、テキストファイルの圧縮に関してはファイルから単語を抽出する符号化法⁽⁶⁾⁽⁷⁾があり高い圧縮率をあげている。この符号化法で圧縮されたファイルの場合、索引も符号化されているため検索時間には索引の復号時間が影響する。

圧縮率の面から見ると適応符号化法と単語を抽出する符号化法が良く、実際のアーカイバなどに利用されている符号化法は適応符号化法一種であるユニバーサル符号化法が用いられる場合が多い。静的符号化法はアーカイバなどには単体で用いられることは少なく適応符号化法と一緒に使われる。

一方、ファイル検索にはフルテキストサーチのような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ファイルの先頭から文字照合を行うため、そのファイル中に一致する文字列があれば、その文字列が言語的に意味を成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。一方、索引ファイルを用いれば高速に検索が行える。しかし、元の文章から単語など予め決められた法則で索引語を抽出しているため、検索は基本的に単語単位でしか行えない。

このように、圧縮と検索には様々な技法があり、圧縮されたファイルに対しての検索の評価は検索時間、復号時間、圧縮率の3つから判断しなければならない。

3. 二段階圧縮法

〈3・1〉 基本原理 本論文では大量の文書に対しキーワード検索を行うファイル検索システムを対象としている。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る。そこで二段階圧縮法では索引を用いた検索を行う。この方式の欠点は2章で述べたよ

図1 圧縮の概要

Fig. 1. The two-stage compression

うに検索に用いる索引ファイルがあるためシステムの容量を増大させる点である。

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。従って、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。両ファイル間の冗長性が少なくなるので、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ファイル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。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かに重点を置くかで、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。

複数のテキストファイルに対して検索を行う場合、シーケンシャルサーチは全てのファイルを検索しなければならない。また、予め索引ファイルを作成しておく場合は索引のみ検索を行えば良い。一方、二段階圧縮法はテキスト各々に付加してある索引ファイルを検索するため、全てのファイルをシーケンシャルサーチするより速く検索を行えるが、予め索引ファイルを作成する方法より検索時間が劣る。

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⁽⁵⁾⁽⁸⁾が提案されている。これらの方式では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事ができる。二段階圧縮法の索引を転置ファイル化する事も可能であるが、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 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。二段階圧縮法の処理ステップは以下になる。

- (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

次節より、これらの処理について詳しく述べる。

〈3・2・1〉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。本論文では英文テキストファイルを対象とするので単語は「アルファベットの連続した文字列」と定義する。このようにして単語を抽出した索引ファイルの例を図2に示す。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。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。また、'And' と 'and' のように大文字と小文字は区別して登録する。

〈3・2・2〉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。二段階圧縮法の符号語長は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。図2の例では以下になる。

I 0

went 1

to 2

aunt 9

この番号を用いてテキストファイルの符号化を行う。また、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常に'0'である。そこで、符号語は一番左のビットを'1'にする事でその文字が符号語かそうでないか識別する事ができる。また、英文は語と語の間にスペースがある事が非常に多いため、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無いことを表し、'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越えた場合、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全ての単語を表現できる。2バイトの符号語長を用いた符号化の例を図3に示す。索引ファイル中の'went'は通し番号に'1'、'Tokyo'は'3'に対応する。'went'の次の文字がスペースなので符号語は'1100000000000001'となり16進数で表現すると'c001'となる。また、'Tokyo'の次の文字はスペースではないので符号語は'1000000000000011'となり16進数表示は'8003'となる。この例では、文章自体は約60%に圧縮されるが、索引ファイルの容量を考えると実際にはその容量は減少していない。しかし、文章中には同じ単語が何度も頻繁に使

図2 索引ファイルの例

Fig. 2. Index file

図3 第一段階圧縮

Fig. 3. The first stage compression

われるので、文章が大きくなれば圧縮率は良くなる。

〈3・2・3〉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、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し、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事もできる。

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、二段階圧縮法は柔軟に対応することができる。

〈3・3〉検索 二段階圧縮法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速くなる。

二段階圧縮法は元の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。従って、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全て索引ファイル中にあるかどうかを検索する事は可能だ

図 4 ファイル検索システム
Fig. 4. File retrieval system

が、それらの語句が元のファイル中で熟語として使われているかを判別する事は難しい。また、英語には動詞の活用形、3 単現や複数形の 's' があるため検索語と一致せずに意味が同じでも検索できない場合がある。これに対してはワイルドカードを用いる事である程度カバーできる。

4. 二段階圧縮法を用いたファイル検索システム

〈4・1〉 ファイル検索システム ファイル検索システムの構成には様々な形が考えられるが、現在は検索エンジンや電子図書館のように索引を用いたものが主流になっている。これらのシステムは索引を用いることによって高速な検索を行う事が可能である。しかし、検索できる文字列は索引ファイルの構成の仕方にもよるが、一般的には単語単位の検索しか行えず、全文検索のように言語的に意味を成さない文字列の検索は行えない。

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、保存できる記憶容量が同じ場合、テキストの数が多いほど良く、索引ファイルの大きさやテキストの圧縮率も検索システムの重要な評価と言える。

索引ファイルなど二次情報ファイルの容量を考えると、テキストファイル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。

本論文ではテキストファイルを大量に保持し、全体として大きなサイズになる場合のファイル検索システムを想定している。

元ファイルを圧縮してあるファイル検索は図 4 のように索引と圧縮された元ファイルで構成される。システムの大きさは索引ファイルと圧縮された元ファイルとの合計となる。また、検索時間は索引を検索する時間と検索結果に該当するファイルの復号時間の合計となる。

〈4・2〉 二段階圧縮法の適用 二段階圧縮法は索引を用いて元のファイルを圧縮しているので同じ単語が頻繁に使われれば圧縮率が良くなる。従って、ファイルサイズが小さいと圧縮率は良くない。前節で述べたようにテキストファイルのサイズはあまり大きくない。従って、二段階圧縮法をそのままファイル検索システムに適用したのでは後

図 5 一括方式
Fig. 5. The combined method

図 6 個別方式
Fig. 6. The individual method

述の実験結果が示すように圧縮率があまり良くならない。そこで、二段階圧縮法をファイル検索システムに適用した場合、

- 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 5,6 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られる。一方、個別方式は 1 ファイルだけ復号すれば良いと仮定すれば復号時間が速いが、ファイル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、二段階圧縮法をファイル検索システム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。

〈4・3〉 索引ファイルの改良 テキストファイルには 'today', 'tomorrow' など、どのファイルでも頻繁に使われる単語や、'cyclone', 'election' などある期間に頻繁に使われる単語などがある。ファイル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。例として 1 年分の雑誌ファイルを一括して二段階圧縮した場合と、1 号ごとに二

表 1 索引ファイルの大きさ

Table 1. Size of index file

圧縮するファイル	索引ファイルサイズ
一日ごと	3,079,992 byte
一括	575,788 byte

図 7 一括索引方式

Fig. 7. The combined index method

段階圧縮を行った 1 年分の索引ファイルの大きさを表 1 に示す。1 号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 5 倍になっている。このように、ファイル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。

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方式を図 7 に示す。この方式を一括索引方式と呼ぶ。ポインタは識別子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 8 の例ではポインタの識別子を「*」としている。ポインタのビット列はまとめたファイルの数だけあり、図中の n ビットとはまとめたファイルが n 個ある事を意味する。ビットが「0」の場合、そのファイルに単語が無いことを意味する。また、「1」の場合はその単語が使われている事を意味する。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事ができる。一括索引付き二段階圧縮でファイル検索システムを作成する手順は以下のようになる。

- (1) ある程度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 1 ファイル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。

検索の手順は索引部の検索を行い、検索語に該当したファ

図 8 ポインタ

Fig. 8. Pointer in index

表 2 実験で用いる二段階圧縮法の圧縮方式

Table 2. Alternatives of our method for experiment

	第二段階圧縮方式	索引圧縮方式	特徴
gzip併用方式	gzip	gzip	圧縮率に重点
ハフマン併用方式	gzip	静的ハフマン	中間
索引無圧縮方式	gzip	圧縮なし	検索時間に重点

イルのポインタを調べ、フラグが立っているファイルを復号する。また、検索語が複数ある場合でもビット列同士を論理演算する事により速く検索が行える。

5. 実験と評価

この章ではまず、二段階圧縮の効果を評価するための実験を行った。また、二段階圧縮法の符号語長による圧縮率の比較を行った。次に二段階圧縮法をファイル検索システムに用いた実用例について検討した。実験は CPU が Celeron466MHz、メモリが 256M バイトの Linux 上で行った。

〈5・1〉二段階圧縮法の評価 実験に用いた英文ファイルは文献⁽⁹⁾の性能評価ファイルのうちのテキストファイルを用いた。テキストファイルは全部で 14 あり全てのファイルで実験を行ったが、その中で 750K バイト、350k バイト、11k バイトの 3 つの結果について示す。

実験は検索時間、圧縮率、復号時間の 3 項目で行った。ここで検索時間とは圧縮ファイルが直接検索できる符号化法の場合はその時間となる。直接検索できない場合は圧縮ファイルの復号時間とフルテキスト検索の時間を足したものである。また、圧縮率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。

提案する 3 種類の二段階圧縮法の圧縮方式として表 2 のように索引圧縮の符号化の違いから、gzip 併用方式、ハフマン併用方式、索引無圧縮方式の三つの方式を用いた。実験を行うファイルサイズがあまり大きくない事から第一段階圧縮の符号語長は 2 バイトとした。また、第二段階圧縮には gzip を用いた。第二段階圧縮にフリーソフトの gzip を選んだ理由は、

- 復号時間が速い。
- 圧縮率が良い。

図 9 検索時間 (その 1)
Fig. 9. Search time(1)

図 11 圧縮率 (その 1)
Fig. 11. Compression ratio(1)

図 10 検索時間 (その 2)
Fig. 10. Search time (2)

図 12 圧縮率 (その 2)
Fig. 12. Compression ratio(2)

があげられる。

検索時間は索引部のみを検索した時間になり、gzip 併用方式は索引部を復号しながら検索を行い、ハフマン併用方式は索引部を直接検索した。また、索引無圧縮方式は索引部のフルテキストサーチを行った。

比較対象として LZW,LZB,gzip,LHA、静的ハフマン符号化法を用いた。また、テキストファイルに対して高い圧縮率を得られる arith_coder⁽⁶⁾ のオプションを 'word' にしたものを用いた。これは単語を用いて算術符号化法を行う圧縮である。gzip はレンペル ジブ動的辞書法に基づいており、LHA は LZH に基づいた二段階符号化法である。

〈5・1・1〉 検索時間 検索時間の測定方法は静的ハフマン符号とハフマン方式の直接検索にはビット単位の文字列照合を行う自作の検索プログラムを用い、gzip と gzip 併用方式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。また、LZW,LZB,LHA,arith_coder は復号時間に 'grep' コマンドの時間を足したものである。

検索時間の実験結果を図 9,10 に示す。図 10 は高速な結果を示しているため、時間軸の尺度は図 9 の 1/10 になっ

ている。gzip 併用方式、索引無圧縮方式ともに良好な結果が得られている。

〈5・1・2〉 圧 縮 率 圧縮率を図 11,12 に示す。gzip と LHA の圧縮率はほぼ同じで、ファイルサイズに関係なく 40%前後の圧縮率を示した。静的ハフマン、LZW、LZB は 50%-65%の圧縮率を示した。arith_coder はファイルサイズが大きくなると急激に圧縮率が良くなっている。gzip 併用方式とハフマン併用方式はファイルサイズが小さい場合、圧縮率は 50%とあまり良くない。ファイルサイズが大きいと gzip より圧縮率が良くなる。また、索引無圧縮方式はファイルサイズが小さい場合圧縮率は悪いが、ファイルサイズが大きくなると gzip や LHA より良くなる。

次に二段階圧縮法で圧縮したファイルの本体部と索引部の元ファイルに対する割合を図 13 に示す。ここでは gzip 併用方式のみを示しているが、他の方式も同様な割合を示しているので省略した。これより、ファイルサイズの増加と共に索引部の占める割合が少なくなっていることが分かる。また、図では索引圧縮と第二段階圧縮を行う前の圧縮率についても示したがファイルサイズがある程度大きな

図 13 索引部と本体部の割合 (gzip 併用方式)

Fig. 13. Ratio of index and main part

表 3 符号語長と圧縮率

Table 3. Code length and compression ratio

符号語長	11 K	350 K	750 K	10 M
2 byte	48.5%	38.5%	38.8%	38.0%
3 byte	51.3%	41.2%	41.9%	36.5%

ると、圧縮率は 60%前後になることが分かる。これはファイルサイズがある程度大きくなると、索引ファイルの単語数はあまり増加しなくなるためである。

提案する 3 種類の二段階圧縮法や arith_coder はテキストファイルから単語を抽出しているためファイルサイズが小さい場合、圧縮ファイル全体に対しての索引部の割合が大きくなり圧縮率が悪くなってしまふ。これに対し、ファイルサイズが十分大きくなれば圧縮ファイル全体に対しての索引部の割合が少なくなり、高い圧縮率を得ることが出来る。

〈5・1・3〉復号時間 実験結果を図 14,15 に示す。図 15 は高速な結果を示しているため、時間軸の尺度は図 14 の 1/4 になっている。提案する 3 種類の二段階圧縮法は gzip,LHA の復号時間には劣るもののその他の符号化法より優れている事が分かる。

〈5・1・4〉符号語長による違い 二段階圧縮法の第一段階圧縮の符号語長は 3.2.2 節で述べたように 2 バイトまたは 3 バイトである。符号語長の違いによる圧縮率の違いについて表 3 に示す。提案する 3 種類の二段階圧縮法による違いはあまり無いため、ここでは gzip 併用方式の結果のみ示す。10M のファイルは 5.2 節で使用した英文週間誌 1 年分のファイルである。

実験結果よりファイルサイズが小さい場合は符号語長が 2 バイトの方が良いが、ファイルが大きくなると 3 バイトが良い事が分かる。また、検索時間と復号時間については実験を行ったが違いはなかった。

図 14 復号時間 (その 1)

Fig. 14. Expansion time(1)

図 15 復号時間 (その 2)

Fig. 15. Expansion time (2)

〈5・1・5〉結果のまとめ このように、提案する 3 種類の二段階圧縮法の検索時間は他の符号化法や圧縮ツールと比べて同等かそれ以上の結果が得られた。圧縮率はファイルサイズがある程度大きくなると arith_coder には劣るものの gzip や LHA より良くなっている。また、復号時間は gzip や LHA の圧縮ツールには劣っているが他の符号化法より復号が速く行える。なお、直接検索可能な静的ハフマン法と比較すると、ファイルサイズが小さい場合を除き検索時間、圧縮率ともに優れている。提案する 3 種類の二段階圧縮法の中では gzip 併用方式が検索時間、圧縮率、復号時間ともに比較的良好な結果を示した。

〈5・2〉ファイル検索システムの評価 この節では、英文雑誌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英文週間誌を 1 年分用いた。ファイルの全体容量は

表 4 一括方式、個別方式、一括索引方式の比較

Table 4. Comparison of combined, individual and combined index method

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	36.5%	2.48	——	2.55
個別方式	47.4%	3.63	0.07	1.18-4.74
一括索引方式	37.3%	6.22	0.12	0.25-6.35

size: 10,631,729 byte time [s]

表 5 gzip の結果

Table 5. Result of gzip

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
gzip一括方式	40.6%	0.75	——	1.49
gzip個別方式	41.2%	0.86	0.02	1.65-2.49
gzip索引方式	44.4%	0.86	0.02	0.11-0.88

size: 10,631,729 byte time [s]

約 10M バイトである。実験では第一段階圧縮の符号語長は 3 バイトとした。一括方式ではファイルをまとめて 1 つのファイルとした。個別方式と一括索引方式は週ごとに 52 個のファイルに分割した。また、実験は提案する 3 種類の二段階圧縮法の全てで行ったが、ここでは実験結果が良好だった gzip 併用方式を用いた。

実験は実際の検索時間、圧縮率、全体復号時間、平均復号時間の 4 項目で行った。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。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。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。

〈5・2・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 4 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り、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。

〈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) 1 年分の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) 1 号ごとに圧縮し、検索は 1 号ずつ復号しながら行う。(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 5 に示す。圧縮率の面からみると gzip 一括方式が良く、実際の検索時間の面からみると gzip 索引方式が良い。

この結果と二段階圧縮法の結果を比較する。なお、ここでは前節での一括方式、個別方式、一括索引方式はそれぞれ二段階一括方式、二段階個別方式、二段階一括索引方式と呼ぶ。圧縮率を重視するには二段階一括方式が良い。一方、実際の検索時間の高速化を重視すると gzip 索引方式が良い。しかし、二段階一括方式は実際の検索時間が一番悪く、gzip 索引方式は圧縮率が一番悪い。

本論文の目的である圧縮率と検索時間の両立について考察すると、二段階一括索引方式は圧縮率が良好で、該当するファイルの数が少ない場合は実際の検索時間も良好である。

多数のファイルを対象とした検索では検索結果に該当するファイルの個数が多い場合は利用者が全てを見ることは困難である。このため、更に条件を追加して該当するファイルを減少させるなどの処理を行う場合が多い。従って、検索結果に該当するファイルを全て復号する事はあまり起こらない可能性がある。このような環境では二段階一括索引方式が有効である。

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提案した。二段階圧縮法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。二段階圧縮法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を新聞や雑誌データのようなファイル検索システムに適用した場合の効果についても論じた。二段階圧縮法を比較的小さなファイルが大量にあるシステムに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事によって検索時間と圧縮率の向上を図り、一年分の英文雑誌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。

本論文では符号化時間について述べなかったが、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。今後は二段階圧縮法の日本語化について検討していきたい。

(平成 12 年 8 月 24 日受付, 同 13 年 3 月 2 日再受付)

文 献

- (1) 植松友彦. 文書データ圧縮アルゴリズム入門. CQ出版, 1994.
- (2) P. Ingwersen. 情報検索研究. トッパン, 1995.
- (3) 松本光崇, 角田達彦, and 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, J80-A:969-976, 1997.
- (4) 宮崎正路, 深町修一, 竹田正幸, and 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, pages 2638-2648, 1998.
- (5) 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, 40:33-40, 1995.
- (6) A. Moffat, R. Neal, and I.H. Witten. Arithmetic coding revisited. *ACM Transactions on Information Systems*, 16(3):256-294, 1998.
- (7) J. Zobel and A. Moffat. Adding compression to a full-text retrieval system. *Software-Practice and Experience*, 25-8:891-903, 1995.
- (8) 定兼邦彦 and 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, 40-No. SIG 8(TOD 4):85-94, 1999.
- (9) T.C. Bell, J.G. Cleary, and I.H. Witten. Text compression. *Prentice-Hall*, 1990.
- (10) 多々納勉, 大塚真吾, and 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, 1:416-417, 1998.
- (11) 大塚真吾 and 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, 3:77-78, 1999.
- (12) 横尾英俊. 記号列の長さや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, J78-A:1175-1173, 1995.

大 塚 真 吾 (学生員) 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 (非会員) 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。情報処理学会, 電子情報通信学会, 人工知能学会など各会員。