Mem. Fac. Educ. Ehime Univ.,Nat. Sci., Vol. 13, No. 2 57~63 (1993)

# 短い符号語長の漢字符号を使用した日本語文書のデータ圧縮

稲 井 義 正
（愛媛大学教育学部技術科教室）
（平成4年10月12日受理）

# Japanese Text Compression using shortened length Kanji code

Yoshimasa INAI
*Department of Technology, Faculty of Education,*
*Ehime University, Bunkyo-cho, Matsuyama 790, JAPAN*
(Received October 12, 1992)

This paper describes a simple and useful coding method of Kanji characters. This kanji code is suitable for doing data compression of Japanese text. The length of this Kanji code is 12 bits, and 4 bits shorter than the 16-bit shifted JIS-Kanji code.

The performance of compression schemes using this Kanji code are compared with original schemes using 8-bit code, and the results shows that the schemes using shortened-length Kanji code are more efficient than original schemes.

## 1. まえがき

多くの実用的なデータ圧縮法においては[2)~4)]，その圧縮（符号化）法をどの様な文書データにも適用可能とするため，通常は入力データの1バイトを圧縮の単位とする．すなわち1バイトを1記号（1文字）としている．そして，入力アルファベットの大きさは，128あるいは256としている．これは，たとえば文書が日本語で書かれていても，1文字が2バイトの日本語であることは利用されていないことを意味する．

また，十分大きな長さのデータ列に対してその最良性が証明されている方法を，実際には余り大きくない長さのデータ列に適用する必要があり圧縮率を下げている．このような場合でも，

圧縮対象の基本とする単位を大きくできれば，より高い圧縮率が達成できる可能性がある．

本論文では，圧縮対象を日本語で書かれた文書に限った場合の圧縮法を考える．日本語で書かれた文書に一般的に使用されている2バイト文字は約6000種である．すなわち2バイト文字の全てを符号化するには，16ビットは必要なく，最小13ビットあればよい．

提案する日本語文字の符号化法では，文書での使用頻度が高いと思われる2バイト文字の一部分だけを，より少ないビット長である12ビットで符号化し，その符号語を入力記号と考えて圧縮を行う．これより，入力アルファベットは大きくなるが，圧縮単位の一部をブロック化して扱うことになり，圧縮率の改善が期待される．

さて，一般的に文書データ等の圧縮では，圧縮されたデータ列から，圧縮前のデータ列が完全に復元されなければならない．そのような無雑音符号化法には[1]，例えば，文字列の出現頻度を抽出しその結果で符号語を構成するような，情報源の確率構造を前提としたものと（ハフマン符号，算術符号など），確率構造は前提としないで符号化するユニバーサル符号化法（Ziv-Lempel 符号， BSTW アルゴリズムなど）がある．

本論文では，この日本語文字の符号化法の有効性を実証するため，最も基本的なデータ圧縮法であるハフマン法と，ユニバーサル符号化法から Lempel-Ziv-Welch の符号化法（LZW 法）のプログラムを作成し，データ圧縮の実験を行った結果を報告する．

## 2．入力文字の符号

日本語で書かれた文書には，漢字やひらがな等の1文字に2バイトの長さを使用する漢字符号と，1バイトで表されるアルファベット，数字，記号などが混在している．

ここで，データとして日本語を表す符号には複数種類（シフト JIS 符号[5]，拡張 UNIX 符号（EUC）等）あるが，いずれも， JIS による情報交換用漢字符号（JIS C6226）を基にしている．この漢字体系には，数字，記号，ひらがな，カタカナなどの非漢字（524種類）と第一水準の漢字（2965種類）および第二水準の漢字（3388種類）が文字として含まれる．

上に挙げた全ての文字を区別して表すには最小13ビット（8192文字まで区別できる）必要とするが，非漢字と第一水準の漢字だけならば12ビット（4096まで区別可能）あればよい．ここで，第二水準の漢字は普通の文書ではあまり使うことのない文字であることを考慮して，実際のデータ圧縮においては，非漢字と第一水準の漢字のみを日本語とみなし（以下誤るおそれのない場合は，この部分の文字をひらがな等を含むが漢字，その符号を漢字符号と書く），第二水準の漢字は1文字を2バイトの符号語と見なすことにする（同様に，第二水準の漢字と1バイトの文字を合わせて非漢字と書く）

以下では，シフト JIS, EUC などどの符号体系でも，ほとんど同じ方法で圧縮プログラムは作成できるので，圧縮するデータはシフト JIS 符号で表されているものとする．

次の算法でシフト JIS 符号の符号語から漢字符号語へ変換できる．但し，シフト JIS 符号語は上で漢字とした文字であるとする．

［算法］

high ：シフト JIS 符号語の上位1バイトの値

low ：シフト JIS 符号語の下位1バイトの値

m ：漢字符号語の最大値（＝4067）

code ：求める漢字符号語

(1) もし high≧136 ならば
　　high ← high−132
　そうでなければ
　　high ← high−129
(2) もし low＞127 ならば
　　low ← low−65
　そうでなければ
　　low ← low−64
(3) code ← high*188+low+256
(4) もし 0≦code≦m ならば
　　code を漢字符号語とする.
　そうでなければ
　　エラーである.

プログラムでは，非漢字符号語の値は0から255まで，漢字符号語の値は，256から4095までの値をとるものとした.

ハフマン符号および LZW 法の詳細についてはここでは省略する.

## 3. LZW 法の辞書と出力符号

以下，LZW 法について必要な部分のみ記述する.

LZW 法には，入力データを増分分解することによって切り出した文字列を登録する辞書が必要である．そこで，まずこの辞書に登録される文字列の参照番号を以下のように定める.

(1) 漢字および非漢字のそれぞれ1文字を登録済みとし，参照番号には，0～4095を与える.

(2) 従って，入力文字列から増分分解によって切り出され新しく辞書に登録される文字列の参照番号は4096以上となる.

次に，圧縮データ列として出力される符号語について説明する．これは本来の LZW 法の符号語とは異なっている．もちろん以下の符号化法の方が圧縮率がよい．また，その符号化法として2種類の方法を検討した．これを K−LZW1 法および K−LZW2 法と呼ぶことにする.

(3) 参照番号4096以上の文字列を符号語として出力する場合は，新規登録の文字列から見つかったことを示すフラグとして〝1〟を1ビット出力し，それに続けて見つかった参照番号から4095（4096としないのは，値0で符号語長の拡大を示す符号語とするため）を引いた値を，その時点で出力される参照番号の最大値を表すのに必要な最小のビット長で出力する．即ち，可変長の符号語で表す.

(4) 参照番号4095以下の漢字および非漢字を符号語として出力する場合は，登録済みのフラグとして〝0〟を1ビット出力し，それに続けて，

(a) K−LZW1 法では，

漢字ならば，漢字を示すフラグ〝1〟を1ビット，それに続けて12ビットの漢字符号語を，

非漢字ならば，非漢字であるフラグ〝0〟を1ビット，それに続けて8ビットの非漢字符号語を出力する．

(b) K−LZW2 では，

漢字，非漢字を共に，12ビットの符号語として出力する．

この符号化法では，同じ入力データ列に対して，新規登録文字列の符号語が出力される回数は同数である．従って，ふたつの方法の圧縮性能の優劣には無関係である．しかし，漢字および非漢字の符号語が出力される回数には依存している．いま漢字符号語の出力にしめる割合をkで表すとすれば，

$$k=\frac{\text{漢字の出力回数}}{\text{漢字の出力回数}+\text{非漢字の出力回数}}$$

出力された漢字と非漢字が合わせてn文字あったとすれば，必要なビット数は，

K−LZW1 法では，

$n\cdot k\cdot 14+n\cdot(1-k)\cdot 10$ [bit]

K−LZW2 法では，

$n\cdot k\cdot 13+n\cdot(1-k)\cdot 13$ [bit]

である．従って，両式が等しくなるkを求めることより，

$k<0.75$ ならば，K−LZW1 法

$k>0.75$ ならば，K−LZW2 法

が圧縮性能がよい．

また，比較のために実験する漢字符号を使用しないLZW法では，辞書の登録済み参照番号は0～255であり，出力符号における漢字と非漢字の区別をするフラグは必要なく，符号語長は8ビットである．

## 4．実 験 結 果

漢字符号を使用することにより圧縮率が改善されることを示す．漢字符号を使用しないデータ圧縮法と比較する．

ここで，圧縮率を次式で定義する．

$$\text{圧縮率}=\frac{\text{符号化後のデータ長}}{\text{符号化前のデータ長}}\cdot 100\ [\%]$$

但し，データ長はバイト数で表すものとする．

実験では32個の文書ファイルを圧縮対象としたが，その文書例に対するデータの大きさなどの値を表1に示す．表の文字数比とは，漢字符号を使用しない場合を1としたとき，漢字符号を使用する場合の実際に処理される文字数の比である．

表1 実験に使用した文書データの例

| 文書 | 大きさ [byte] | 漢字の出現回数 (注1) | 文字数比 (注2) |
|---|---|---|---|
| A | 2094 | 698 (66.7 [%]) | 0.667 |
| B | 7290 | 2069 (56.8 [%]) | 0.716 |
| C | 10129 | 3533 (69.8 [%]) | 0.651 |
| D | 19802 | 4948 (50.0 [%]) | 0.750 |
| E | 27772 | 12357 (89.0 [%]) | 0.555 |
| F | 31041 | 9730 (62.7 [%]) | 0.687 |
| G | 40999 | 16532 (80.6 [%]) | 0.597 |
| H | 62444 | 6610 (21.2 [%]) | 0.894 |
| I | 64308 | 12223 (38.0 [%]) | 0.810 |
| J | 102855 | 36115 (70.2 [%]) | 0.649 |
| K | 152850 | 44924 (56.8 [%]) | 0.716 |
| L | 174543 | 44083 (50.5 [%]) | 0.747 |

注1．( )内はデータに占める漢字のバイト数での割合

注2．処理する文字数の比
（漢字符号を使用しない場合を1）

### 4．1 ハフマン符号による圧縮

静的なハフマン符号による圧縮実験の結果を表

表2　ハフマン符号による圧縮率

| 文書 | K−HUF [%] | HUF [%] | 改善率 [%] |
|---|---|---|---|
| A | 70.0 | 79.7 | 9.7 |
| B | 73.3 | 83.5 | 10.2 |
| C | 64.3 | 78.0 | 13.7 |
| D | 67.8 | 79.5 | 11.7 |
| E | 52.8 | 71.1 | 18.3 |
| F | 65.1 | 78.2 | 13.1 |
| G | 57.3 | 76.0 | 18.7 |
| H | 56.4 | 62.8 | 6.4 |
| I | 54.6 | 63.7 | 9.1 |
| J | 61.3 | 79.2 | 17.9 |
| K | 52.6 | 68.1 | 15.5 |
| L | 50.7 | 64.2 | 13.5 |

表3　ハフマン符号化に要した時間

| 文書 | K−HUF [sec] | HUF [sec] | 時間の比 |
|---|---|---|---|
| A | 1.68 | 1.31 | 1.282 |
| B | 4.80 | 4.30 | 1.116 |
| C | 5.95 | 5.64 | 1.055 |
| D | 11.61 | 11.09 | 1.047 |
| E | 13.66 | 14.43 | 0.947 |
| F | 17.56 | 17.14 | 1.024 |
| G | 20.85 | 22.17 | 0.940 |
| H | 32.66 | 29.89 | 1.093 |
| I | 32.63 | 31.06 | 1.050 |
| J | 53.86 | 56.89 | 0.947 |
| K | 76.71 | 79.42 | 0.966 |
| L | 83.08 | 84.44 | 0.984 |

時間の比はHUFを1とした場合

2に示す．漢字符号を使用しない方法をHUF法，漢字符号を使用する方法をK−HUF法と表した．表からわかるように，圧縮率は6から19%（全データに対する実験の結果でも6から19%，平均14%）程度改善された．

また，表3は圧縮に要した時間を比較した結果である．表よりわかるように，時間の差はほとんどないといえる．

表4　LZW法の圧縮率

(a)　辞書の大きさ：8192

| 文書 | K−LZW1 [%] | K−LZW2 [%] | LZW [%] | K−LZW1 改善率 |
|---|---|---|---|---|
| A | 55.8 | 55.4 | 60.2 | 4.4 |
| B | 59.7 | 62.0 | 65.1 | 5.4 |
| C | 50.5 | 51.1 | 57.6 | 7.1 |
| D | 52.3 | 54.6 | 55.2 | 2.9 |
| E | 41.4 | *41.0 | 47.6 | 6.2 |
| F | 52.2 | 54.5 | 55.1 | 2.9 |
| G | 45.5 | 47.5 | 49.8 | 4.3 |
| H | 34.2 | 36.6 | 34.9 | 0.7 |
| I | 38.5 | 40.8 | 37.3 | −1.2 |
| J | 61.5 | 64.5 | 61.7 | 0.2 |
| K | 44.0 | 47.1 | 46.5 | 2.5 |
| L | 40.3 | 43.3 | 41.4 | 1.1 |

(b)　辞書の大きさ：16384

| 文書 | K−LZW1 [%] | K−LZW2 [%] | LZW [%] | K−LZW1 改善率 |
|---|---|---|---|---|
| D | 49.6 | 50.4 | 55.2 | 5.6 |
| E | 38.7 | *37.8 | 47.6 | 8.9 |
| F | 48.1 | 48.7 | 54.7 | 6.6 |
| G | 40.7 | 41.0 | 48.7 | 8.0 |
| H | 31.0 | 31.7 | 34.2 | 3.2 |
| I | 32.6 | 32.8 | 36.4 | 3.8 |
| J | 46.0 | 46.2 | 51.9 | 5.9 |
| K | 36.6 | 36.7 | 40.8 | 4.2 |
| L | 33.8 | 34.9 | 38.4 | 4.6 |

A〜Cは(a)と同じ値

(c)　辞書の大きさ：32750

| 文書 | K−LZW1 [%] | K−LZW2 [%] | LZW [%] | K−LZW1 改善率 |
|---|---|---|---|---|
| I | 32.6 | 32.8 | 36.4 | 3.8 |
| J | 43.2 | 43.3 | 50.3 | 7.1 |
| K | 33.2 | 33.3 | 38.8 | 5.6 |
| L | 28.1 | *28.0 | 33.5 | 5.4 |

A〜Hは(a)と同じ値
*K−LZW2法の方が圧縮率がよい．

## 4.2　LZW法

LZW法には辞書の大きさ（参照番号の最大値）がパラメタとして存在する．比較実験を行う場合に等しくする量として，圧縮開始時点での辞書の空き領域の大きさと辞書全体に確保する領域の大きさが考えられる．

以下の実験では，プログラムの大きさがほぼ等しくなると考え，辞書全体に使用する記憶領域の大きさを等しくした．登録済み参照番号の最大値が約4000であることから辞書の大きさの最小値は8192（符号語を有効に利用するには2のべき乗が都合がよい）となる．但し，この最小値の場合，空き領域はLZW法で約8000，K−LZW法で約4000となり，約2倍の差があるためあまり比較しても意味はないかも知れない．

表4に結果を示す．表の(b)及び(c)の項目で値が省略されているのは，小さなデータに対しては全ての入力データ列を処理しても，辞書に空き領域が残り小さな辞書の場合と同じ結果となるためである．

表5　LZW法の実行に要した時間
（辞書の大きさ：8192）

| 文書 | K−LZW1 [sec] | LZW [sec] | 時間の比 |
|---|---|---|---|
| A | 0.69 | 0.80 | 0.862 |
| B | 2.26 | 2.53 | 0.893 |
| C | 2.82 | 3.30 | 0.855 |
| D | 5.65 | 6.28 | 0.900 |
| E | 6.77 | 8.26 | 0.820 |
| F | 8.73 | 9.81 | 0.890 |
| G | 10.56 | 12.10 | 0.873 |
| H | 14.84 | 14.72 | 1.008 |
| I | 15.64 | 16.16 | 0.968 |
| J | 30.67 | 34.79 | 0.882 |
| K | 40.70 | 44.54 | 0.914 |
| L | 42.78 | 48.81 | 0.876 |

時間の比はLZW法を1とした場合

表からわかるように，ほとんどの場合で（例では，(a)の文書Iを除いて）K−LZW1法の圧縮率は改善されている．

全ての文書に対する実験結果は次の通りである．まず，辞書の大きさ8192の場合−1.2から7.5%，平均で4.0%，辞書の大きさが16384の場合2.8から8.9%，平均で6.0%，辞書の大きさが32750の場合3.3から8.9%，平均で6.3%の圧縮率の改善が行われた．

また，表5は圧縮に要した時間を比較した結果である．表よりわかるように，圧縮に要する時間はK−LZW法がLZW法に対して平均値で約10%少なくてよいことがわかる．

## 5．考　　察

### 5.1　圧縮率がよくなる理由

実験結果で見たように，本方法により圧縮率が改善できることがわかった．その理由を挙げてみる．

**(1)　入力データの大きさの減少**

ひらがなや漢字の一部を2文字ではなく1文字として扱った．即ち，入力データの大きさを実効的に減少させたことになる（表1の文字数比の項を参照）．

**(2)　符号化する文字列長の拡大**

ハフマン法では一度に符号化する文字列の長さを長くしたほうがより平均符号長が短くできる．漢字符号を使用すれば符号化する文字列長は長くなるので，実験結果のように圧縮率が改善された．

**(3)　日本語文字列として早く辞書に登録**

従来の方法では，例えばひらがな1文字を2バイト2文字として処理するので，日本語を含む文字列が増分分解によって辞書に登録されるには，その文字列の出現回数が本方法よりも多くなければならない．従って，本方法がより早く日本語文字列を辞書に登録できる．これにより，辞書で見つかる可能性が高くなり，LZW法の圧縮率がよくなる．

### 5．2　圧縮処理に要する時間

本方法では，入力文字列からひらがなや漢字などを識別して漢字符号に変換する漢字の符号化過程と，実際の圧縮のための符号化過程の2つの過程を経て圧縮が行われる．従って，この漢字符号化過程がオーバーヘッドとなるが，次の圧縮処理過程においては，漢字符号を使用しない方法より符号化する文字数が減少するので，その部分に必要な時間は減少する．

ハフマン符号による圧縮法では，漢字符号化過程のオーバーヘッドと，圧縮処理過程の時間の減少がちょうど相殺しあい，全体の処理に要する時間がほとんど等しくなったものと思われる．

LZW 法では，圧縮処理過程に必要な時間の減少の効果が少しだけ大きく影響し，全体の処理に要する時間を減少させる結果となったと思われる．

### 5．3　K－LZW1 法と K－LZW2 法の比較

実験結果からわかるように，K－LZW1 法に比べて K－LZW2 法の圧縮率がよくなる場合は少ない，またその場合でもそれほど圧縮率に差があるわけではない．更に，K－LZW2 法の圧縮率は LZW 法に比べて悪くなる場合も多い．従って，基本的には K－LZW1 法で十分である．

## 6．む　す　び

本論文では，日本語で書かれた文書を効率よく圧縮するための日本語の符号化法を説明し，その符号語を使用したハフマン符号による圧縮法および LZW 法の圧縮実験の結果を示した．その結果から，圧縮単位として，ひらがなや漢字等の文字を1文字として扱うことによって，従来の方法を適用するよりも圧縮率がよくなることを明らかにした．また，LZW 法の場合は，圧縮処理に要する時間も減少することを示した．

今後の課題は，他の圧縮法との組合せについての実験，圧縮率を更により良くする方法の検討である．

## 文　献


1）D. A. Lelewer and D. S. Hirshberg : "Data compression", ACM computing Surveys, 19, 3, pp. 261－295 (Sept. 1987).

2）荻原剛志：〝仮名文字の短縮表現を用いた日本語文書の圧縮について〟，信学論（A），J74－A，9，pp. 1431－1438 (1991－09).

3）朴，今井，山森他：〝パターンマッチングと算術符号を用いた実用的なデータ圧縮法〟，信学論（A），J71－A，8，pp. 1615－1628 (1988－08).

4）T. A. Welch : "A Technique for High-Performance Data Compression", IEEE computer, 17, 6, pp. 8－19 (1984－06).

5）上柿力編：〝パソコン／ワープロ漢字辞典〟，ナツメ社 (1991－03).