Vol.2012-IFAT-106 No.7
Vol.2012-DD-85 No.7
2012/3/27

適合的ビットデータ化による ニュースストリームデータの圧縮方法の提案

Minami Matsumoto[†1] and Takao Miura[†1]

静的ハフマン符号では同じ記号の集合に出現頻度に応じた符号を割り当てることでデータを圧縮する．しかしニューステキストのような連続データでは、次々に流れてくる大量のデータをリアルタイムに処理することが求められている．本研究では、ニューステキストの特性にあわせて適合的に辞書の変更を行い、圧縮するための圧縮符号化方法を提案する．この技法では KL(Kullback-Leibler) 情報量によりニューステキストの分布の変化を調べることにより、チャンクの区切りを作成する．本報告では、静的圧縮及び動的圧縮方法と比較し、その特徴および実験による性能を論じる．

1. 前書き

近年, 電子マネーや電子商取引の普及やシステムのクラウド化によって, インターネット上には多種多様で膨大な量のデータが発生しており, 次々に流れてくる大量のデータをリアルタイムに処理することが求められている. 特に, ネットワークを介し無限に到来する時刻順のテキストデータはテキストストリームと呼ばれ, ニュースメディアや Twitter のようなミニブログ,Facebook に代表される SNS(ソーシャルネットワーキングサービス) により大量のテキストストリームデータが生み出されている．しかし, インターネット上の空間は増加する一方であり, 限られた中で効率的に利用するために, ストリームデータを圧縮することが求められる．

データ圧縮は「可逆圧縮」と「非可逆圧縮」の 2 つに大きく分けることができる．可逆圧縮は元のデータを完全に復号可能であり, 完全に再生可能であるべきテキストデータやプログラムファイルに用いられる. 非可逆圧縮は元のデータを近似的に再現する．データが少し変化しても比較的問題のない画像データや音声データを圧縮する場合に用いられる．可逆圧縮には２通りの方法がある. 全てのデータを俯瞰し符号化していく「静的符号化」があり, ここでは最初に全データをスキャンしてモデルを作成し, 次に符号化するという２パスで圧縮する. それに対し, データを符号化していく過程で動的にモデルを作成・更新していく「動的符号化」がある．

Table 1 データ圧縮の分類

可逆圧縮	静的符号化	シャノン・ファノ符号化	
		ハフマン符号化	
		算術符号化	
	動的符号化	連長符号化	
		動的ハフマン符号化	
		ユニバーサル符号化	LZ 符号化
			BSTW 符号化
非可逆圧縮	画像符号化	JPEG 圧縮法	
	音声符号化		

それぞれの代表的な符号化を表 1 に示す.[1)] ハフマン法 (Huffman coding) は代表的な静的符号化の１つである. 文字に対する符号語は可変長であり, 出現頻度の高い文字には, より短い符号語を割り当てている．そのため, ハフマン木を用いて語頭条件 (prefix property) を満足する可変長符号を生成する必要がある．特に, 事前にすべての文字の出現頻度情報を必要とする静的ハフマン法 (static Huffman coding) と動的に頻度を見積もりながら符号化する動的ハフマン法 (dynamic Huffman coding) がある．静的ハフマン法では事前に全ての文字の出現頻度情報を必要とするため, ストリームデータの圧縮に用いることはできない．一方, 動的ハフマン方は動的にモデルを作成・更新していくため, 柔軟である．しかし, 頻度情報しか利用しておらず, 冗長性や局所性に全く対処できない.

本研究ではニュースストリームデータを適合的に圧縮する方法を提案する. この手法では,KL(Kullback - Leibler) 情報量を用いてニューステキストの分布の変化を調べ, 適合的にストリームをチャンクに区切る. また, 符号語は 3 分木を構成することで語頭条件を満足する可変長符号を生成し, 出現頻度の高い文字にはより短い符号語を割り当てる．

2. 静的ハフマン法と KL 情報量

本章では，静的ハフマン法と KL 情報量を要約する．

2.1 静的ハフマン法

ハフマン符号は 1952 年に David Huffman によって開発された符号であり, コンパクト符号やエントロピー符号の１つである．JPEG や LZH などの圧縮フォーマットで使用されている．擬似的に実数の符号語長を割り振る算術符号と比較すれば

†1 法政大学
Hosei University

Vol.2012-IFAT-106 No.7
Vol.2012-DD-85 No.7
2012/3/27

データ圧縮効率は劣るが, 整数の符号語長という制約のもとでは常に最適な符号を構成できる. 以下では, 2 元符号でのハフマン符号の構成法を示す. 各シンボルは葉ノードに対応しており, 葉から根に枝状のラベルを辿ることで一意に復号化できる.

まず, 各シンボルの出現回数を求める. 次に葉を含むすべての節点のうち親を持たないものから, 最小の値のものと 2 番目に小さい値のものを選択する. それらを子供にもつ新しい節点を作成し, 両方の子供の値の和をこの節点の値とする. これを全ての節点及び葉が一つの木に束ねられるまで繰り返す. 次に根からに枝の左右に 0,1 を割り振る. 作成した木の葉から根に向かって枝を辿ってできたものが, この葉 (シンボル) における符号語となる.

12 バイトからなる文字列 Ex.1 *abcaabbcacde* がある. 文字列 Ex.1 は { a,b,c,d,e } の 5 つの文字からなる文字列である Ex.1 の各文字を 1 つのシンボルとした際の出現回数を表 2 に示す. この中で 1 番と 2 番めに小さいものは d,e である. 節点 { d,e } を作成し, 節点の値を 2 とする. 次に, 作成した節点と葉の中で 1 番小さいものはと節点 { d,e } である. 2 番めに小さいものは値が 3 となる葉 b と c である. 値が同じ場合はどちらをとっても構わないので, 今回は c を選択する. よって,c と節点 { d,e } から節点 { c,de } を作成し, その値を 5 とする. 残っている葉 a,b と子を持たない節点 { c,de } の中から小さいもの 2 つは葉 a,b である. これより, 節点 { a,b } を作成し, その値を 5 とする. そして節点 { a,b } と節点 { c,de } から根節点を作成し, 値は 12 となる. 枝の左右に 0,1 を割り振り生成したハフマン木を図 1 に示す. このハフマン木の葉から根に向かって枝に付いたラベルを辿り, 表 2 の符号語が生成される.

Table 2 Ex.1 の出現回数と符号語

辞書項目	出現回数	符号語
a	4	00
b	3	01
c	3	10
d	1	110
e	1	111

2.2 KL(Kullback - Leibler) 情報量

KL(Kullback - Leibler) 情報量とは, 確率論と情報理論における 2 つの確率分布の差異を計る尺度である. 相対エントロピー（Relative entropy）とも呼ばれる. 真の離散確率分布 $P = \{p_1, \Gamma, \{p_n\}$ とモデルとする離散確率分布 $Q = \{q_1, \Gamma, \{q_n\}$

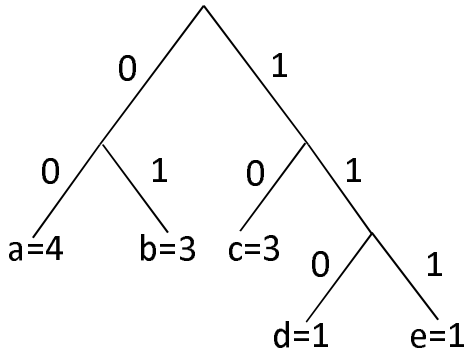

Fig. 1 Ex.1 のハフマン木

に対するカルバック・ライブラー情報量は以下のように定義される

$$D_{KL} = \sum_{i=0}^{n} P(i) \log(\frac{P(i)}{Q(i)})$$

P(i),Q(i) はそれぞれ確率分布 P, Q に従って選ばれた値が i になる確率である. また,KL 情報量は次の性質を持つ.

(1)$D_{KL} \geq 0$
(2)$D_{KL} = 0, P = Q$

D_{KL} の値が小さく 0 に近いほどモデル P は真の分布 Q に近いとみなす. 実際の応用では真の分布 P はデータや観測値、正確に計算で求められた確率分布などを表し, Q は理論値、モデル値、P の予測値などを表す. これを文書間で用いると,KL 情報量は文書における語の発生確率の差を表す. これにより文書の類似度を求めることができる. 3 つの文書 A,B,C における語の発生確率を $P_A = \{0.5, 0.3, 0.2\}, P_B = \{0.8, 0.1, 0.1\}, P_C = \{0.3, 0.3, 0.3\}$ とする. このとき, 文書 A に対する文書 B の類似度は

$$D_{KL}(A \parallel B) = 0.5 \log(\frac{0.5}{0.8}) + 0.3 \log(\frac{0.3}{0.1}) + 0.2 \log(\frac{0.2}{0.1})$$
$$\Gamma\ 0.233$$

文書 A に対する文書 C の類似度は

Vol.2012-IFAT-106 No.7
Vol.2012-DD-85 No.7
2012/3/27

$$D_{KL}(A \parallel C) = 0.5\log(\frac{0.5}{0.3}) + 0.3\log(\frac{0.3}{0.3}) + 0.2\log(\frac{0.2}{0.3})$$
$$\Gamma\ 0.174$$

となる. よって, $KL_{AC} \leq KL_{AB}$ となるため, 文書 C の方が文書 A に似ている.

3. 適合的ニュースストリーム圧縮方法の提案

3.1 適合的ニュースストリーム圧縮の狙い

本稿で提案する適合的ニュースストリーム圧縮の狙いは, 連続するテキストの類似度に応じた辞書の切り替えによってその区間において最適な符号割り当てを行うことである. ストリームデータでは一度にすべて読み込むことができないため, メモリに読み込むことができる一定サイズ (ウィンドウサイズ) に分けて読み込み圧縮する必要がある. ニュースストリームでは時系列のニュースであるという特性を利用して, 連続するニュースの類似度の変化に応じてストリームを区切り, 圧縮を行う.

図 2 に示すように, 芸能面の記事が連続した後に株価データが連続していたとする. 人名や芸能関係の語が多い前の集合と, 株の銘柄や数値データからなる後の集合では出現する語や文字の頻度には大きな差がある. ストリームを読み込み圧縮する際にウィンドウサイズよって分けると, このような変化に対応せずに区切ることになる. その結果, 前の集合と後ろの集合に跨ったチャンクが作成されてしまう.2 でウィンドウサイズが 4 だとすると, { 政治 2, 芸能 2 } と { 芸能 2, 株価 2 } , { 株価 2,⋯ } と分けるよりも, { 政治 1, 芸能 3 } と { 株価 4 } , { ⋯ } と分けた方が効率的に圧縮できる.

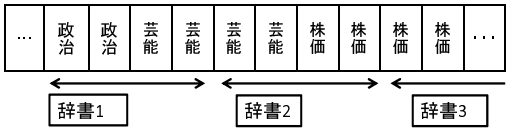

Fig. 2 適合的ニュースストリーム圧縮の狙い

3.2 KL 情報量におけるニュースストリームの区切り

図 3 のようにしてニュースストリームデータの KL 情報量 式 1 を求める. ニュースストリームの先頭から X までを元になる集合 P とする. 比較対象である集合 Q は集合 P と新たに読み込んだニュースストリーム x の和集合である. それぞれの集合におけるシンボルの発生回数を Pcount, Qcount とし, 式.2, 式/3 で表す. 式.2, 式 3 における α はシンボルの出現確率が 0 となるのを回避するための補正定数である. M,N はそれぞれ P,Q におけるシンボルの出現回数の総和であり n はシンボルの数である. $D_{KL}(P \parallel Q) \leq y(\Gamma\Gamma)$ となる間は, Q は集合 P と類似性があるして圧縮を続ける. 図 3 に示したようにこの値が閾値 y を超えると, Q に別の出現頻度分布を持ったニュースストリームが追加されたとみなして直前に読み込んだニュースストリーム x の前でストリームを区切る. そして, 新たに x を先頭としたストリームから集合 P を作成して, 上記を繰り返す.

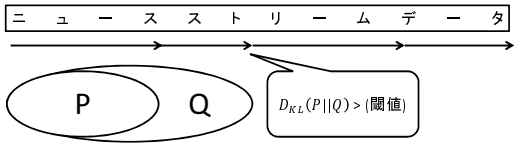

Fig. 3 ストリームの区切り方

$$D_{KL}(P \parallel Q) = \sum_i P(i)\log(\frac{P(i)}{Q(i)}) \tag{1}$$

$$P(i) = \frac{Pcount(i) + \alpha}{N + \alpha * n} \tag{2}$$

$$Q(i) = \frac{Qcount(i) + \alpha}{M + \alpha * n} \tag{3}$$

$$N = \sum_i Pcount(i) \tag{4}$$

$$M = \sum_i Pcount(i) \tag{5}$$

3.3 符号語の割り当て

図 4 に示す 3 分木を用いて符号化を行う. 辞書における各項目は葉ノードと節点及び根ノードに相当する. 最も頻度が高いシンボルは根ノードに位置し, 符号語 00 を割り当てる. これはスットプコードの役割を兼ねる. 各枝には 01,10,11 の 2 ビット符号 3 つを左から順にに割り振る. シンボルは根に向かって枝のラベルを辿り,

Vol.2012-IFAT-106 No.7
Vol.2012-DD-85 No.7
2012/3/27

根ノードに割り当てられた符号 00 で終わることで, 一意に復号化することができる. また, この符号語は瞬時復号可能である. 木の深さ d は符号長 l に対応しており,$d = \frac{l}{2} - 1$ である. 深さ d における符号語の数は 3^d となる. よって深さ d までに表現できる符号語の総数は $\sum 3^d$ となる.

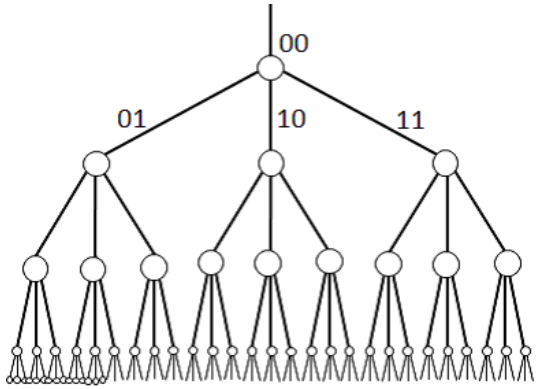

Fig. 4 辞書木 (3 分木)

符号語の割り当ては以下の 2 段階で行う.

STEP1. P の読み込み及び符号化
ストリームの先頭から X までを読み込みシンボルの発生回数を得る. シンボルを発生頻度順にソートする. シンボルの出現確率を得る. 最も頻度が高いシンボルを根ノードとし,2〜4 番目に頻度の高いシンボルを子とする. 深さ 1 の葉が埋まったら, 深さ 2 の葉の左から順に頻度の高い文字を 3 つずつ子ノードとして追加する. これをすべてのシンボルの追加が終わるまで繰り返す. 辞書木によって構成された符号に基づき, 元データを可変長符号に変換する.

STEP2. Q の読み込み及び符号化と KL 情報量によるストリームの区切り
Q のシンボルの発生回数 Qcount=Pcount, その総和 M=N とする. ストリームの先端から x まで読み込む. シンボル i がすでに辞書にあれば Qcount(i) に 1 を加える. なかった場合は空いている葉ノードに追加し,Qcount=1 とする. シンボルの出現確率を求める. P の確率分布と Q の確率分布から,$D_{KL}(P \parallel Q)$ を求め, 閾値 (y) と比べる. $D_{KL}(P \parallel Q) > y$ となるかメモリへの読み込みがウィンドウサイズを超えるまでステップ 2 を繰り返す. $D_{KL}(P \parallel Q) > y$ もしくはウィンドウサイズを超えたら, 直前に読み込んだニュースストリーム x の前でストリームを区切る. 新たに x を先頭としたストリームから集合 P を作成して, 上記を繰り返す.

3.4 圧縮ファイルの構成

圧縮ファイルの構成を図 5 に示す. 文字列単位はシンボルの文字数を 1 バイトで示す. ウィンドウサイズは一度にメモリに読み込み可能なサイズ (M バイト) を 1 バイトで示す. ウィンドウサイズを 5M バイトとする場合,5 とする. Q の圧縮データと P の辞書の区切りには管理コード 1 を入れる. これは常に木の 2 番目に空いている葉ノードの位置から生成される符号語を指す. 圧縮データと辞書との区切りには管理コード 2 を入れる. これは常に木の初めの空いている葉ノードの位置から生成される符号語を指す.

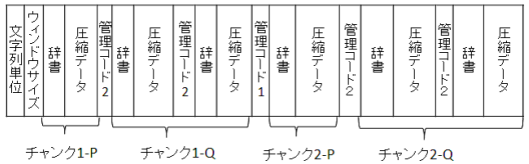

Fig. 5 圧縮ファイルの構成

3.5 辞書の構成

チャンク 1-P の辞書の構成を図 6 に示す. 辞書の頭には辞書の項目数 (2 バイト) がくる. その後にソートした項目が出現頻度の高い順に並ぶ. 図 **??**は文字列単位が 3 のときの例である. すべての文字は 1 バイトで出力され, 先頭からの 3 バイトずつが辞書の 1 項目と対応する. 図 6 にあるように, 改行やタブも辞書の項目の中の 1 文字として扱う. ファイルの終端などで, 文字列単位に満たないシンボルの場合は後ろに null を入れて文字列単位にする.

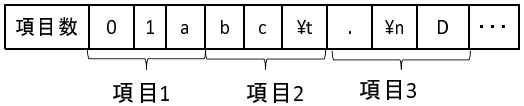

Fig. 6 辞書の構成 (チャンク 1-P)

Vol.2012-IFAT-106 No.7
Vol.2012-DD-85 No.7
2012/3/27

チャンク 1 以外の P の辞書の構成を図 7 に示す. 辞書の頭には辞書の項目数 (2 バイト) がくる. その後ろに前のチャンクの辞書との差分情報を入れる. 置換個数は前のチャンクから辞書の位置に変更があった項目の個数を 2 バイトで示す. 置換情報は新しい辞書の位置と前の辞書の位置によって示す. 前の辞書のインデックス 14 から新しい辞書のインデックス 8 への移動の場合, (8,14) の 2 バイトで示す. 置換情報の後には追加情報を入れる. 追加個数を 2 バイトで示す. これは新しいチャンクで新たに出現したシンボルの個数である. 追加情報は新しい辞書に挿入する位置 (2 バイト) と追加するシンボル (3 バイト) からなる. インデックス 20 にシンボル abc を追加したい場合, (20, abc) の 5 バイトとなる.

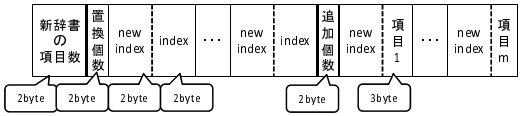

Fig. 7 辞書の構成 (チャンク 1 以外の P)

Q で追加する辞書の構成を図 8 に示す. 辞書の頭には辞書の項目数 (2 バイト) がくる. P で辞書を更新した際に使われなかったシンボルは辞書木からは削除されるが, 次のチャンクの間は辞書のサブリストとして保持している. 置換個数はサブリストから辞書木に置きなおす項目の数を指す. そのあとに続くインデックスは辞書木に置きなおしたいシンボルが入っているサブリストの位置を 2 バイトで示す. シンボルを辞書木に挿入する場所は初めの空葉ノードなので, 新しい辞書のインデックス情報は持たない. 追加個数は辞書木にもサブリストにも存在しないシンボルの数である. そのあとには追加シンボルが 3 バイトずつ続く.

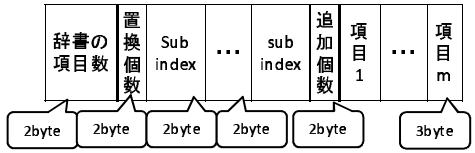

Fig. 8 辞書の構成 (Q)

3.6 辞書の圧縮

先に示したように, 辞書の構成はチャンクと集合によって変わる. チャンク 1-P では辞書の項目をそのまま羅列していくのに対して, チャンク 1 以外の P では差分情報の更新とすることで, 辞書のサイズを小さくしている. 辞書の更新における置換とは前のチャンクと新しいチャンクで最適となる符号長が違うものを指す.P の出現頻度において最適となる符号の位置が, 木の同じ深さの別の位置を指していた場合, そのシンボルは位置の変更は行わない.

図 4 に示すように, インデックス 2,3,4 は同じ深さにあるのでこの中での移動は行わない. 同様に 5〜13 においてもこの中での移動は行わない. $4 \rightarrow 5, 5 \rightarrow 4$ のように深さが変わる場合のみ変更を行う.

4. 実験

ここでは, ニュースストリームに対する適合的圧縮法の評価実験について述べる. まず実験方法について述べ, 次に実験結果を示し, 最後に考察および提案アルゴリズムの評価を行う.

4.1 実験方法

今回, 実験には Reuters corpus の 1996.08.20-1996.09.19 のデータ 62935 件,4557183 バイト (約 90.1 MB) をニュースストリームデータとして用いる. 実験に使用したパラメータは文字列単位=3, windowsize=5, X=1000, x=100 である. 比較対象として, 静的圧縮法及び動的圧縮法との比較検証を行う. 本来であれば, ニュースストリームデータに対して静的圧縮は行えないが, 性能を比較検証するためにすべてのニュースストリームデータでのシンボルの出現確率を求め, 静的圧縮を行う. 静的圧縮での符号の割り当ては章 2.3.3 の図 4 を用いて行う. また, 動的圧縮ではウィンドウサイズのパラメーターごとに章 2.3.3 の STEP1 のみを行い, 符号化する.

4.2 実験結果

実験結果を以下に示す.

表 3 は閾値とストリームの区切りを示したものである. これはチャンクの始まりがどの位置から始まっているかを示している. チャンク 2 では閾値 0.09 が 3400 から始まり 6899 までであるのに対し, 閾値 0.1 では 3586 から 6886 までと閾値 0.09 の方がチャンクを広く取っている. 一方，チャンク 4 では閾値 0.08 が 10100 から 13471, 閾値 0.1 が 10286 から 13710 であるのに対し, 閾値 0.09 では 10300 から 13400 と他の閾値よりも狭くなっている. また, 今回の閾値の範囲ではどれもスト

Vol.2012-IFAT-106 No.7
Vol.2012-DD-85 No.7
2012/3/27

リームデータを 19 のチャンクに分けている.

表 4 では動的及び適合的 (閾値=0.08,0.09,0.1, ∞) での辞書のサイズ及び圧縮率を示した. 静的圧縮では辞書は 1 つしか生成されないので圧縮は行わない. 圧縮前というのは, すべての辞書を章 2.3.5 で示したチャンク 1-P と同様に書き出した場合の辞書の総サイズである. 最も辞書の圧縮率がよかったのは閾値 0.08 のときで, 約 52.9 % 削減できた.

表 5 はニュースストリームを静的, 動的及び適合的 (閾値=0.08,0.09,0.1, ∞) に圧縮した際の圧縮後のサイズと圧縮率である. 閾値= 0.09 としたときに辞書を覗いた圧縮率が約 54.07 % , 辞書を含めた圧縮率が約 55.015 % と最もよい値を示した. これは辞書を含めても静的圧縮の圧縮率約 55.44 % よりも圧縮率がよい.6 に示したように, 約 400k バイト閾値 0.09 の方が小さい.

Table 3 閾値とストリームの区切り

チャンク	0.08	0.09	0.1	∞
1	0	0	0	0
2	3300	3400	3586	3586
3	6600	6900	6886	7418
4	10100	10300	10286	10787
5	13471	13400	13710	14469
6	16871	16900	17310	18081
7	20371	20500	20935	21645
8	23471	23600	24138	24805
9	26724	26895	27477	28223
10	30108	30306	30877	31642
11	33608	33849	34394	35357
12	36808	36949	37641	38536
13	39808	40361	41186	42035
14	43208	43461	44665	45471
15	46508	46840	48190	49074
16	49808	50440	51524	52353
17	53096	53663	54942	55755
18	56524	57073	58443	59360
19	59824	60673	61843	62820

4.3 考察・評価

実験の結果, 表 5 から閾値 0.09 とした時に約 55.015 % となり, 最もよい圧縮率を得られる. これは表 3 に示すように, 閾値によってチャンクの範囲が変わりシンボルの発生確率が変わったためである. このことは前後のチャンクの辞書の変化からもいえる. 表 7 は閾値 0.09 のチャンク 4 から 5 に切り替わる際に, 前のチャンクと後のチャンクでの符号長が増減した文字列の一部である. 上 3 つの文字列はチャンク 4 には 1997.08.27(火) の記事が多かったが, チャンク 5 の途中から翌日の水曜日の記事になったため, ”Wednesday”に含まれる文字列の発生頻度が増加し, より短い符号語が割り当てられる. その下”Co”までは株価や為替市場の記事で多く見られる文字列である. 実際にチャンク 5 では株価や為替市場の記事が増加しており, そのため符号長が短くなる. 一方, 増加したものとしては USA,MFS,AEL,ISR がある. コーパスを見ると,MFS という文字列が出現する記事には必ず USA という文字も出現している.MFS は”MFS Communication Co”や”MFS-WorldCom”という文字列の一部である. MFS はアメリカの企業であり, チャンク 4 に多く含まれている 1997.08.27 にはワールドコム (WorldCom) が、MFS Communications 社を US $ 124 億の株式で買収したという報告がある. そのため, 買収が報告された日の

Table 4 辞書の圧縮率

	圧縮前 [byte]	圧縮後 [byte]	圧縮率 [%]
動的	1641.28125	887.4229	54.06891
0.08	1639.45313	867.5703	52.91827
0.09	1634.75977	869.9404	53.21518
0.1	1622.0332	866.4063	53.41483
∞	1587.92871	857.1113	53.97669

Table 5 圧縮率の比較

	圧縮後 (辞書除く) [byte]	圧縮後 [byte]	圧縮率 (辞書除く) [%]	圧縮率 [%]
静的	50979.84	51201.33	55.20824	55.4481
動的	50830.84	51718.27	55.04689	56.00791
0.08	49936.98	50804.55	54.07888	55.01841
0.09	49931.62	50801.56	54.07308	55.01518
0.1	50610.69	51477.1	54.80847	55.74674
∞	50627.87	51484.98	54.82707	55.75528

Table 6 KL=0.09 との差

	圧縮後 (辞書除く) [byte]	圧縮後 [byte]	圧縮率 (辞書除く) [%]	圧縮率 [%]
静的	1048.217	399.7607	1.135159	0.432918
動的	899.2207	916.7031	0.973804	0.992737
∞	5.355469	2.985352	0.0058	0.003233

Vol.2012-IFAT-106 No.7
Vol.2012-DD-85 No.7
2012/3/27

ニュースでは多く取り上げられていたが, 次の日に報道は一段落したためチャンクの切り替えによる文字列長の増加につながったといえる.

その下の文字列は”ISRAEL”という文字列として現れており, イスラエルに関するニュースが減少したため, 文字列長も増加してる.

Table 7 チャンク 5 と 6 での辞書の変化 (閾値 0.09)

文字列	符号長 (前)	符号長 (後)	増減
sda	16	12	-4
esd	16	12	-4
Wed	18	14	-4
—	10	8	-2
cen	12	10	-2
N	16	14	-2
N/	16	14	-2
N/A	16	14	-2
Co	18	16	-2
USA	12	14	2
MFS	16	18	2
AEL	16	18	2
ISR	16	18	2
MOR	18	20	2

5. 結論

本研究では, 静的圧縮法及び動的圧縮法と比較して,KL 情報量を用いてストリームを区切ることで, よりチャンクごとに適した符号を割り当てる手法を提案した. 結果として, 閾値 0.09 とした時には動的圧縮だけでなく静的圧縮よりもよい圧縮率を得られる. このことは表 7 で示すように曜日の変化が文字列長の変化に現れたことや, チャンク 4 からチャンク 5 で文字列長の増加が見られた文字列”MFS”から, 実際にチャンク 4 に含まれる日に買収されていたことが確認出来たという点から, ストリームの区切りと文字の分布は対応していたといえる.

References

1) D.A.Lelewer and D.S.Hirschberg: ”Data Com-pression,” ACM Computing Surveys, Vol.19, No3, pp.261-296, 1987.