テキストデータの圧縮率を高めるための前処理について

中村　博文[1]・渕田　孝康[2]

Pre-processing for Enhancing Text Data Compression Ratio

NAKAMURA Hirofumi[1] and FUCHIDA Takayasu[2]

（令和3年10月1日受理）

あらまし　バイトデータを対象とした既存のデータ圧縮プログラムの適用前に、前処理として圧縮対象を加工しておくことで、圧縮率を向上できる場合のあることを実験によって示す。本研究において、前処理は、試みとして、「頻繁に現れる隣接バイト対の中で比較的効果的なものについて、すべての出現箇所について2バイトから1バイトへと未使用のバイトコードに置き換える」ということを繰り返している。この方法を活用できる圧縮対象は、バイトデータの中でも、未使用のバイトコードが存在することが確実な、テキスト形式のファイルである。本方法による圧縮率の向上は、既存のデータ圧縮プログラムの圧縮性能が高いと少ないが、ある程度の長さのテキストファイルなら圧縮性能の向上が見られることが多い。gzip、bzip2及びctwを用いた実験において、gzipについては、圧縮率を更に数%縮められる場合があった。本方法は、比較的効果的な隣接バイト対を探すことが必要なため、符号化の時間は桁違いに増すことがあるが、復号で行う前処理の逆の処理は、バイトデータを置き換えながら一度走査する程度の処理が増すだけである。全体として可逆であっても、前処理による置き換えが情報源の性質をかく乱して、情報源モデルによっては表現に必要な平均情報量を増やしてしまうのではないかという疑問について、単純マルコフ情報源という観点では、前処理による置き換えが表現に必要な平均情報量を増やすことはないことを確認する。

キーワード　［データ圧縮, 前処理, 隣接記号対, テキストデータ, 文字コード］

1　はじめに

多数のデータ圧縮アルゴリズムが提案されてきている（例えば、文献1〜4））。データ圧縮は、圧縮対象に何らかの特徴や構造を見い出すために、圧縮対象を解析したり、統計的性質を計量するなど、圧縮対象から得られる何らかの情報を用いて2値符号化している。なお、念のために明記しておくと、圧縮対象が音声や画像に限られるわけではないため、可逆圧縮を前提としている。

データ圧縮アルゴリズムがアプリケーションソフトウェアに組み込まれている場合もあるが、特にデータを選ばないデータ圧縮プログラムやアーカイバは、バイトデータのファイルを対象として提供されている。

本報告では、バイトデータを対象とした既存のデータ圧縮アルゴリズムの入力の前に、圧縮対象を一度加工することで圧縮率を向上させることを試みている。その加工とは、頻繁に現れる隣接バイト対の中で比較的効果的なものを別の未使用のバイトコードに置き換えておくことである。このため、未使用のバイトコードが確実に存在すると考えられるテキストファイルを対象にして実験を行った。

1 都城工業高等専門学校一般科目　General Education Division, National Institute of Technology(KOSEN), Miyakonojo College
2 鹿児島大学大学院理工学研究科　Gradient School of Science and Engineering, Kagoshima University

2　既存符号の符号化と復号の定式化

本報告では、既存のデータ圧縮の符号化をE、復号をE^{-1}のように表す。また、Eの入力である圧縮対象のデータをI、EがIを入力して得られる圧縮結果をC_Iと表す。

従来の通常のデータ圧縮法では

入力はI，$I \rightarrow E \rightarrow C_I$，出力は$C_I$

とデータ圧縮をして、C_Iを記憶や通信に用い、データを使用するより前に

入力はC_I，$C_I \rightarrow E^{-1} \rightarrow I$，出力はI

と復号して使用していると捉えることができる。ここで、復号して得られるデータは元のデータIと同じ内容であることから、復号結果についても記号は圧縮対象のデータと同じIを用いて表している。

3　提案手法

3.1　提案手法のおおまかな流れ

提案手法はおおまかには、EにIを直接適用するのではなく、Eに対する前処理Pを付加することを考え、IにPを施して得た内容（I^*と表す）をEに適用するというものである。

すなわち、符号化は

入力はI，$I \rightarrow P \rightarrow I^* \rightarrow E \rightarrow C_{I^*}$，出力は$C_{I^*}$

と行う。C_{I^*}を記憶や通信に用い、使用するより前に

入力はC_{I^*}，$C_{I^*} \rightarrow E^{-1} \rightarrow I^* \rightarrow P^{-1} \rightarrow I$，出力はI

と復号して得られたIを使用する。ここで、PはEを行う前の前処理を、P^{-1}はPの逆の処理を表す。

この形式は、符号理論において符号器をうまく構成して総合的に通信路をより効率的に使用するのと同様の構造である。通信路の効率的な使用は、通信路の性質の理解や定式化が進んでいることから可能になっている。

しかし、データ圧縮において、Eにとって圧縮率の面で最適なI^*がどのようなものであるかや、Iを元に最適なI^*を作り出す処理（Pが行う処理）をどのようにすると処理時間や記憶領域の増加を抑えられるかは、まだ分かっていない。

本研究では、いくらかでも効果のあるI^*の作成を試み、前処理による圧縮率改善の可能性を示す。

やや、具体的には、「頻繁に現れる隣接バイト対の中で比較的効果的なものについて、すべての出現箇所について2バイトから1バイトへと未使用のバイトコードに置き換える」ことを必要なだけ繰り返してI^*を得る。

このようにするとき、I^*は、どの隣接バイト対をどの未使用バイトコードに置き換えたかという情報の蓄積（I'と表す）と、（I'に記録されている）すべての置き換えを行った後の（置き換えられなかった部分も含む）内容（I''と表す）とから構成される。

つまり、$I^* = I' + I''$である。

I'の決定においては、まずIについて隣接バイト対の頻度を調べ、頻度の高いものからK種類まで（足りない場合はその数まで）を置き換えの候補と考え、各候補について、圧縮対象中のすべての出現を未使用バイトコードに置き換えてからEを適用して、圧縮率が最良の隣接バイト対を置き換える隣接バイト対として選ぶ（直ぐ後及び3.3でLというパラメータを用いた拡張をする）。

その選んだ隣接バイト対について未使用バイトコードに置き換えた内容を新たに入力と考える。

置き換えられる未使用バイトコードが残っており、圧縮率が改善される間、以上を繰り返す。

I''は置き換えを逐次的に適用してでき上がると考えてもよいが、IにI'にある置き換えをすべて適用した内容と等しい。I'は、二分木による置き換えの辞書と捉えることもできる。

後ほど、I'とI''の具体的なデータ形式について述べる。

本報告の方法では圧縮率の測定が必要になるが、ひとつの圧縮率のデータを測定するのに、いくつの置き換えを行っておくかという数をLと表すことにする（試すときにL回置き換えできない場合はできる回数までとする）。5でLとKをパラメータとして圧縮率改善の実験を行っている。

効果のある最適な置き換えを探す方法はまだ分かっていない。本報告の実験では、未使用のバイトコードを使うごとに、最大K^L種類のEの試用の中から選定した。その制御は、Pのアルゴリズムの一部として、Pから機械的に行った。

I'の情報源としての性質はI''とは異なることから、提案手法でEに直接適用するのはI''のみとする。そのときのEの出力を$C_{I''}$と表す。

ここで、混乱を避けるために断っておくと、導入の分かり易さのために用いてきたC_I、I^*、C_{I^*}という記号を、以下では用いないことにする。

3.2　提案手法の構成

以上をまとめると、符号化の構成は

入力はI, $I \rightarrow P \rightarrow I'+I''$, $I'' \rightarrow E \rightarrow C_{I''}$, 出力は$I'+C_{I''}$

（P ⇅ E）

であり、復号の構成は

入力は $\mathrm{I}'+\mathrm{C}_{\mathrm{I}''}$, $\mathrm{C}_{\mathrm{I}''} \to \mathrm{E}^{-1} \to \mathrm{I}''$,

$\mathrm{I}'+\mathrm{I}'' \to \mathrm{P}^{-1} \to \mathrm{I}$, 出力は I

である。なお、符号化において P は E をいく度も使用するが、復号において P^{-1} からは E や E^{-1} を使用する必要はない。復号では、I″ から、I′ を用いてワンパスで I を得ることができる。

符号化には手間と時間をかけるが、復号はバイトデータを一度走査する程度のワンパスの置き換えが増えるだけで済む。

P と連携したこのような E の使い方を、以下では簡単に P•E と記述する。

P•E を符号化として、$(\mathrm{P}\bullet\mathrm{E})^{-1}$ を復号として、それぞれひとまとまりとして考えると、通常の符号化、復号と同じ構造になる。

つまり、符号化は

入力は I, $\mathrm{I} \to \mathrm{P}\bullet\mathrm{E} \to \mathrm{I}'+\mathrm{C}_{\mathrm{I}''}$, 出力は $\mathrm{I}'+\mathrm{C}_{\mathrm{I}''}$

であり、復号は

入力は $\mathrm{I}'+\mathrm{C}_{\mathrm{I}''}$, $\mathrm{I}'+\mathrm{C}_{\mathrm{I}''} \to (\mathrm{P}\bullet\mathrm{E})^{-1} \to \mathrm{I}$, 出力は I

である。

3.3 前処理の具体的試行内容

実験で用いた前処理の試行内容を、具体例を用いて示す。圧縮対象の文字列の例として

a b a b c d a b c d ···

を用いる。これについて、探し当てた未使用のバイトコードの順序数のひとつを、仮に 129 とする。そのバイトデータのことを ⟨129⟩ と表す。2 進数で表すなら 10000001 というデータである。他の順序数でも、個数の場合（3.4 で出てくる）でも、同様に表記する。更に次に探し当てた未使用のバイトコードの順序数を、仮に 130 とする。

P の処理内容は、「隣接バイト対の中で圧縮率の面で効果的なものを未使用のバイトコードに置き換えておくこと」である。

置き換える隣接バイト対の出現頻度が高いほど、I″ を短くすることになる。I″ を短くすることが全体の圧縮率の改善と同義ではないが、本報告では、第ゼロ近似として、置き換えの候補として出現頻度の高いものから順に試している。

まず、$L=1$ の場合を述べ、後で拡張する。

P が置き換え内容を選ぶ処理の流れは、以下のとおりである。

(1) 圧縮対象について隣接するバイト対すべてについての出現頻度を求め、頻度が高い隣接バイト対から K 種類を順に試しに置き換えをして、それぞれ E で圧縮して圧縮率を求める。

(2) 試行した中で最良の結果を与えた組み合わせ（隣接バイト対）を採用し、圧縮対象の中のその隣接バイト対のすべての出現において未使用のバイトコードに置き換える。

(3) こうしてできたデータについて、新たに圧縮対象と捉えて、以上を繰り返す。未使用のバイトコードを使い切るか、改善しなくなったら繰り返しを終える。

$L=1$, $K=2$ の場合で、(1) と (2) を例示する。

圧縮対象において、頻度の高い方から $K=2$ 種類の隣接バイト対として a b と b c が選ばれたとする。

まず、圧縮対象の中の a b をすべて未使用のバイトコード ⟨129⟩ に置き換えた ⟨129⟩ ⟨129⟩ c d ⟨129⟩ c d ··· について E を使って圧縮率を調べる。

次に、圧縮対象の中の b c をすべて未使用のバイトコード ⟨129⟩ に置き換えた a b a ⟨129⟩ d a ⟨129⟩ d ··· について E を使って圧縮率を調べる。

最大 $K=2$ 通りの試行の中で最も圧縮率が改善できたものを採用する。仮に、a b がそうであったとすると、圧縮対象の中の a b を ⟨129⟩ に置き換えた

⟨129⟩ ⟨129⟩ c d ⟨129⟩ c d ···

を新たに圧縮対象と捉えて以上のような処理を繰り返す。

L が 2 以上の場合については、(1) と (2) において次のように拡張する（ゲームの次の一手を決めるために、L 手先まで手を読むことと類似している。探索は、枝が最大 K 本のゲーム木と同様である）。

隣接バイト対の置き換えを最大 K 種類試す際のそれぞれについて、更に隣接バイト対の置き換えを最大 K 種類試すということを、L 重に行い、圧縮対象に L 重（L 種類）の置き換えをした内容のそれぞれについて E を適用して圧縮率を調べる。新しい圧縮対象を作るために未使用のバイトコードと置き換えるのは、最も圧縮率を改善した L 重の置き換えの一重目の隣接バイト対である。

最大で K 通り試すことを L 重に行うため、E を適用して圧縮率を調べる回数は、ひとつの未使用バイトコードあたり最大で K^L 通りになる。

これは一種の探索問題であり、局所最適解に陥る可能性はあるが、他によい方法を得ていない。

$L=2$, $K=2$ の場合で例示する。

圧縮対象について、頻度の高い方から $K=2$ 種類の隣接バイト対として a b と b c が選ばれたとする。

まず、仮に圧縮対象のすべての a b を ⟨129⟩ に置き換えた

⟨129⟩ ⟨129⟩ c d ⟨129⟩ c d ···

について、頻度の高い方から $K=2$ 種類の隣接バイト対として、仮に ⟨129⟩ c と c d が選ばれたとする。そ

れぞれで⟨130⟩に置き換えをした⟨129⟩ ⟨130⟩ d ⟨130⟩ d ⋯と⟨129⟩ ⟨129⟩ ⟨130⟩ ⟨129⟩ ⟨130⟩ ⋯についてEを使って圧縮率を調べる。

次に、仮に圧縮対象のすべてのb cを⟨129⟩に置き換えた

a b a ⟨129⟩ d a ⟨129⟩ d ⋯

について、頻度の高い方から$K = 2$種類の隣接バイト対として、仮にa ⟨129⟩と⟨129⟩ dが選ばれたとする。それぞれで⟨130⟩に置き換えをしたa b ⟨130⟩ d ⟨130⟩ d ⋯とa b a ⟨130⟩ a ⟨130⟩ ⋯についてEを使って圧縮率を調べる。

以上の調べた4つの中で、仮に3番目の圧縮率が良かったとすると、b cを⟨129⟩に置き換えた

a b a ⟨129⟩ d a ⟨129⟩ d ⋯

を新たに圧縮対象と捉えて以上のような処理を繰り返す。

3.4　置き換え情報I′の表現

ここではI′の表現形式について述べる。

I′として、まず、何種類置き換えをするか（仮にk種類とする）をバイトコード⟨k⟩で表す。

もし置き換えをしないならI′は

⟨0⟩

の1バイトのみである。このようにI′+$C_{I''}$が1バイトは長くなる場合があるが、提案手法に効果がない場合でも1バイトの増加で済む。効果があれば、I′がある程度の長さになっても、I′+$C_{I''}$全体は短くなる。

置き換えるべきバイトコードの空きがない場合もこの表現を用いる。そうすると、復号側で置き換えるべきバイトコードの空きがない場合を扱うための処理は特別必要なくなる。

置き換えをする場合には、$k \geq 1$であり、⟨k⟩の後ろに、k組分の「隣接バイト対と、置き換えをする空きのバイトコードの情報」を置く。それを少しでも短く表現するために、置き換えが何種類あるかによって、表現方法を2種類使い分ける。

置き換えが32種類以下の場合は、隣接バイト対の2つのバイトコードと置き換えをする空きのバイトコードの計3バイトをひと組として必要なだけ並べる。

例えば、a bを順序数129のバイトに、⟨129⟩ c を順序数130のバイトに置き換えたのであれば、I′は

⟨2⟩ $\underbrace{\text{a b ⟨129⟩}}$ $\underbrace{\text{⟨129⟩ c ⟨130⟩}}$

の7バイトになる。ここで、波かっこは、分かりやすさのために付記したものであり、実際の情報には必要なく含まれない。以下でも同様である。

置き換えが33種類以上の場合は、置き換えをする空きのバイトコードを、256ビット（すなわち、32バイト）のビットマップで表す。その後ろに、置き換えをする隣接バイト対すべてについて、バイト対をそのままバイトコードを2つ並べて表す。

例えば、先述の例を含め40種類を置き換えたのであれば、I′は

⟨40⟩ $\underbrace{\langle\cdot\rangle\langle\cdot\rangle\cdots\langle\cdot\rangle}_{32\text{バイト}}$ $\underbrace{\text{a b ⟨129⟩ c}\langle\cdot\rangle\langle\cdot\rangle\cdots\langle\cdot\rangle}_{40\times 2\text{バイト}}$

の113バイトの内容になっている。ここで、⟨·⟩は、具体的な内容の明示をしなかった適当な内容のバイトデータを表している。

I′の長さはkから

$$\begin{cases} 1+3k & (0 \leq k \leq 32) \\ 33+2k & (33 \leq k \leq 255) \end{cases}$$

と容易に知ることができる。

この報告では、使用していないバイトコードを探す必要があるものの、実際に探せばそのときに使用していないバイトコードは容易に分かることから、文字コードの種類が何であるか（例えば、JIS 8単位コードかどうかというようなこと）が分かっている必要はない。

置き換えをするごとにI′が少しずつ長くなるが、使えるバイトコードの数は256を越えないため、I′の長さは圧縮対象のデータ長に対して定数オーダである。

3.5　提案手法に関する性質

提案手法の符号化は、もとにしたデータ圧縮プログラムが適応的であっても、全体としては一括型の符号化になる。

また、比較的効果的な隣接バイト対を探すことが必要なため、符号化の時間は桁違いに増すことがある。

提案手法の復号は元に戻す置き換えが必要になるが、これはバイトデータを置き換えながら一度走査する程度でよく、適応的にワンパスでも動作させられるため、パイプライン処理で既存符号の復号プログラムに渡すこともでき、その場合、既存符号の復号が適応的な処理である場合に、提案手法の全体としての復号の性質は適応的なままである。

符号化の時間は長いが復号は短いため、一度符号化したら幾度もまたは多数の箇所で使用するようなデータに向いている。

置き換えをしたことでI″はかく乱されているのではとの疑問も生じるが、記憶を持つ情報源の中で最も基本的なモデルである単純マルコフ情報源とい

う観点で見たときの平均情報量には影響がないことを確認している。データ圧縮の議論の中で述べると煩雑になるため、切り離して付録１から付録２で述べている。

4　記号対置き換えをする既存の取り組みとの比較

Gage[4]のデータ圧縮プログラムは、隣接バイト対を頻度の高いものから順に空きのバイトコードに置き換えるというものである。これを更に他のバイトコード対応のデータ圧縮プログラムの入力とする、本報告のような利用はされていない。

本報告で試みた方法は、隣接バイト対を空きのバイトコードに置き換える点は同じであるが、目的のデータ圧縮プログラムEを使用して効果の高い隣接バイト対をある程度探索して選択する点が異なる。そして置き換えした内容を目的のデータ圧縮プログラムEで圧縮している。

文献5）の方法は、文字コードが既知のテキストファイルが対象で、日本語も含めて使われない62のバイトコードとその連接であるバイトコードに、2回以上現れた英数字列すべてを置き換えていた。方法は単純であるが、効果に関係なく置き換えを適用するため、中には大きく膨張してしまうファイルもあった。提案手法は、時間はかかるが効果を確かめながら処理する。最悪でも高々1バイト長くなるだけである。

5　実験と結果

実験には既存のデータ圧縮プログラムgzip（http://www.gzip.org/, version 1.3.5）、bzip2（http://www. bzip.org/, version 1.0.5）及びctw（http://www.ele.tue.nl/ctw/, version 0.1）を用いた。圧縮率で優れている全く異なる方法の中から選んだ。

これらのアルゴリズムについて簡単に触れておく。gzipは、LZ77[1]とハフマン符号を用いている。bzip2は、ブロックソート法、MTF (Move-To-Front) 法及びハフマン符号を用いている[2]。文脈木重み付け法 (Context-Tree Weighting) [3] によるctwは独特の確率推定結果に算術符号を用いている。

圧縮性能は、ほとんどのファイルでgzip<bzip2<ctwであり、ctwが優る。圧縮の速さと一般的な利用の多さでは、gzip>bzip2>ctwであり、gzipが優る。

実験に用いた圧縮対象データは、Canterbury Corpusの中のテキストファイルである。ファイル名は実験結果の表中に記す。

実験結果を表1以降に示す。

表には各ファイルごとに3つのデータ圧縮プログラムについて、順に、既存プログラムのみの場合（Eのように表記。また、Eは必要なときに具体的なプログラム名に置き換えて表記。）の圧縮率Ratio（既存プログラムでの圧縮サイズをオリジナルのサイズで割って得ており、全く圧縮しない場合が100%）、提案手法（簡単にP•Eと表記）による圧縮率の改善 ΔRatio（改善したバイト数とオリジナルのサイズとの比で、縮めた場合の符号は負）、圧縮率の改善 δRatio（改善したバイト数と使用した既存プログラムでの圧縮サイズとの比で、縮めた場合の符号は負）、改善したバイト数（縮めた場合の符号は負）、ひとつのファイルの圧縮中にPがEを呼び出した回数を記している。

効果がない場合には置き換えをしないため、どのデータ圧縮プログラムと併用しても1バイトよりも長くなることはない。

$L=1$で、Kを1、2、3、10、100と変えた場合の実験結果を表1～表5に示す。表2～表5では、Kが増えたときにひとつ前の表よりも悪いデータに下線（カラー表示なら赤に見える彩色も）を施している。

Kの増加に対して改善の効果は必ずしも単調増加ではないが、Kを増やすと多くの場合に改善の度合いが高まっている。中には、δRatioで評価して、更に数%効果が出ている場合がある。

既存プログラムの使用回数はKを大きくするのに伴って増えている。

$L=1$で$K=100$とした場合とK^Lが同じ値である、$L=2$でKが10の場合の実験結果を、表6に示す。これは、2種類の置き換え毎に圧縮率を評価したものである。表6では、表5よりも悪いデータに下線（カラー表示なら赤に見える彩色も）を施している。

Lを増やした結果の優劣は一定していないが、$L=1$で$K=100$の場合と比べて、良くならない場合が多い。言い換えると、制御がより単純な$L=1$のアルゴリズムの方で、ある程度の圧縮率の改善が見込める。

6　おわりに

既存のデータ圧縮プログラムの適用前に圧縮対象に加工を加えて、全体としての符号化性能を上げられる場合のあることを示した。中には、優れた既存の圧縮手法の圧縮結果を、更に数%縮めている場合がある。

符号理論においては符号器をうまく構成して総合的に通信路をよりうまく使用することを探求してい

表 1　実験結果 ($L = 1, K = 1$)

File name	Size	gzip Ratio	P•gzip ΔRatio	P•gzip δRatio	P•gzip ΔSize	P•gzip Ex. gzip	bzip Ratio	P•bzip ΔRatio	P•bzip δRatio	P•bzip ΔSize	P•bzip Ex. bzip	ctw Ratio	P•ctw ΔRatio	P•ctw δRatio	P•ctw ΔSize	P•ctw Ex. ctw
	[Bytes]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]
alice29.txt	152089	35.79	-0.82	-2.28	-1240	16	28.41	0.00	0.00	1	2	25.94	0.00	0.00	1	2
asyoulik.txt	125179	39.10	0.00	0.00	1	2	31.61	0.00	0.00	1	2	29.03	0.00	0.00	1	2
cp.html	24603	32.52	0.00	0.01	1	2	30.99	0.00	0.01	1	2	28.84	0.00	0.01	1	2
fields.c	11150	28.19	-0.60	-2.13	-67	18	27.26	-0.15	-0.56	-17	8	24.88	-0.05	-0.22	-6	4
grammar.lsp	3721	33.40	-0.38	-1.13	-14	10	34.48	-0.11	-0.31	-4	4	29.80	-0.05	-0.18	-2	4
lcet10.txt	426754	33.95	-0.04	-0.12	-174	4	25.24	-0.03	-0.13	-138	4	22.90	-0.00	-0.01	-12	4
plrabn12.txt	481861	40.51	0.00	0.00	1	2	30.21	0.00	0.00	1	2	27.32	0.00	0.00	1	2
xargs.1	4227	41.57	0.02	0.06	1	2	41.68	0.02	0.06	1	2	37.02	0.02	0.06	1	2

“Size”:Original size,
“P”:Proposed preprocessor,
“Ratio”:(Compressed size by E)/(Original size),
“E”:Existing data compression program (gzip, bzip or ctw)
“ΔRatio”:(Compressed size by P•E)/(Original size) - (Compressed size by E)/(Original size)
=(ΔSize)/(Original size),
“δRatio”:(Compressed size by P•E)/(Compressed size by E) - (Compressed size by E)/(Compressed size by E)
=(ΔSize)/(Compressed size by E),
“ΔSize”:(Compressed size by P•E) - (Compressed size by E),
“Ex. E”:Execution count of E by P.

表 2　実験結果 ($L = 1, K = 2$)

File name	Size	gzip Ratio	P•gzip ΔRatio	P•gzip δRatio	P•gzip ΔSize	P•gzip Ex. gzip	bzip Ratio	P•bzip ΔRatio	P•bzip δRatio	P•bzip ΔSize	P•bzip Ex. bzip	ctw Ratio	P•ctw ΔRatio	P•ctw δRatio	P•ctw ΔSize	P•ctw Ex. ctw
	[Bytes]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]
alice29.txt	152089	35.79	-1.09	-3.06	-1663	24	28.41	-0.27	-0.96	-416	18	25.94	-0.04	-0.16	-65	6
asyoulik.txt	125179	39.10	-0.49	-1.26	-616	9	31.61	-0.08	-0.27	-105	6	29.03	0.00	0.00	1	3
cp.html	24603	32.52	0.00	0.01	1	3	30.99	0.00	0.01	1	3	28.84	0.00	0.01	1	3
fields.c	11150	28.19	-0.59	-2.10	-66	27	27.26	-0.15	-0.56	-17	12	24.88	-0.05	-0.22	-6	6
grammar.lsp	3721	33.40	-0.24	-0.72	-9	9	34.48	-0.11	-0.31	-4	6	29.80	-0.05	-0.18	-2	6
lcet10.txt	426754	33.95	-1.37	-4.04	-5856	45	25.24	-0.05	-0.19	-207	9	22.90	-0.00	-0.01	-12	6
plrabn12.txt	481861	40.51	-0.85	-2.10	-4098	15	30.21	-0.01	-0.03	-46	6	27.32	-0.01	-0.04	-51	6
xargs.1	4227	41.57	-0.21	-0.51	-9	12	41.68	0.02	0.06	1	3	37.02	0.02	0.06	1	3

表 3　実験結果 ($L = 1, K = 3$)

File name	Size	gzip Ratio	P•gzip ΔRatio	P•gzip δRatio	P•gzip ΔSize	P•gzip Ex. gzip	bzip Ratio	P•bzip ΔRatio	P•bzip δRatio	P•bzip ΔSize	P•bzip Ex. bzip	ctw Ratio	P•ctw ΔRatio	P•ctw δRatio	P•ctw ΔSize	P•ctw Ex. ctw
	[Bytes]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]
alice29.txt	152089	35.79	-1.10	-3.07	-1670	32	28.41	-0.25	-0.86	-373	20	25.94	-0.10	-0.40	-156	16
asyoulik.txt	125179	39.10	-0.75	-1.92	-938	28	31.61	-0.08	-0.27	-105	8	29.03	0.00	0.00	1	4
cp.html	24603	32.52	-0.17	-0.54	-43	20	30.99	0.00	0.01	1	4	28.84	0.00	0.01	1	4
fields.c	11150	28.19	-0.63	-2.23	-70	40	27.26	-0.15	-0.56	-17	16	24.88	-0.06	-0.25	-7	12
grammar.lsp	3721	33.40	-0.24	-0.72	-9	12	34.48	-0.11	-0.31	-4	8	29.80	-0.08	-0.27	-3	8
lcet10.txt	426754	33.95	-1.37	-4.04	-5853	60	25.24	-0.09	-0.37	-401	20	22.90	-0.01	-0.04	-41	12
plrabn12.txt	481861	40.51	-0.88	-2.17	-4233	24	30.21	-0.11	-0.36	-530	24	27.32	-0.02	-0.08	-99	12
xargs.1	4227	41.57	-0.21	-0.51	-9	16	41.68	0.00	0.00	0	8	37.02	0.02	0.06	1	4

表 4 実験結果 ($L = 1, K = 10$)

File name	Size	gzip Ratio	P•gzip ΔRatio	P•gzip δRatio	P•gzip ΔSize	P•gzip Ex. gzip	bzip Ratio	P•bzip ΔRatio	P•bzip δRatio	P•bzip ΔSize	P•bzip Ex. bzip	ctw Ratio	P•ctw ΔRatio	P•ctw δRatio	P•ctw ΔSize	P•ctw Ex. ctw
	[Bytes]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]
alice29.txt	152089	35.79	-1.77	-4.94	-2690	198	28.41	-0.26	-0.91	-393	55	25.94	-0.10	-0.40	-156	44
asyoulik.txt	125179	39.10	-0.93	-2.38	-1163	110	31.61	-0.08	-0.27	-105	22	29.03	-0.00	-0.00	-1	22
cp.html	24603	32.52	-1.54	-4.75	-380	616	30.99	-0.07	-0.21	-16	33	28.84	-0.10	-0.34	-24	22
fields.c	11150	28.19	-0.90	-3.18	-100	154	27.26	-0.17	-0.63	-19	66	24.88	-0.13	-0.54	-15	55
grammar.lsp	3721	33.40	-0.27	-0.80	-10	44	34.48	-0.21	-0.62	-8	33	29.80	-0.08	-0.27	-3	22
lcet10.txt	426754	33.95	-2.37	-6.97	-10103	363	25.24	-0.24	-0.94	-1009	44	22.90	-0.01	-0.04	-41	33
plrabn12.txt	481861	40.51	-2.07	-5.11	-9977	352	30.21	-0.26	-0.85	-1234	110	27.32	-0.04	-0.14	-186	66
xargs.1	4227	41.57	-0.28	-0.68	-12	44	41.68	-0.09	-0.23	-4	22	37.02	0.02	0.06	1	11

表 5 実験結果 ($L = 1, K = 100$)

File name	Size	gzip Ratio	P•gzip ΔRatio	P•gzip δRatio	P•gzip ΔSize	P•gzip Ex. gzip	bzip Ratio	P•bzip ΔRatio	P•bzip δRatio	P•bzip ΔSize	P•bzip Ex. bzip	ctw Ratio	P•ctw ΔRatio	P•ctw δRatio	P•ctw ΔSize	P•ctw Ex. ctw
	[Bytes]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]
alice29.txt	152089	35.79	-3.69	-10.31	-5610	14039	28.41	-0.20	-0.71	-308	404	25.94	-0.20	-0.77	-305	1515
asyoulik.txt	125179	39.10	-2.94	-7.52	-3681	14241	31.61	-0.22	-0.68	-270	505	29.03	-0.02	-0.06	-20	505
cp.html	24603	32.52	-1.85	-5.67	-454	8383	30.99	-0.10	-0.33	-25	404	28.84	-0.51	-1.76	-125	1616
fields.c	11150	28.19	-1.07	-3.79	-119	1919	27.26	-0.22	-0.79	-24	404	24.88	-0.39	-1.59	-44	1111
grammar.lsp	3721	33.40	-0.27	-0.80	-10	306	34.48	-0.21	-0.62	-8	157	29.80	-0.08	-0.27	-3	155
lcet10.txt	426754	33.95	-3.92	-11.55	-16734	17373	25.24	-0.33	-1.32	-1424	1414	22.90	-0.15	-0.64	-622	1212
plrabn12.txt	481861	40.51	-3.44	-8.49	-16569	13231	30.21	-0.36	-1.21	-1756	909	27.32	-0.07	-0.27	-354	1212
xargs.1	4227	41.57	-0.45	-1.08	-19	707	41.68	-0.14	-0.34	-6	202	37.02	0.02	0.06	1	101

表 6 実験結果 ($L = 2, K = 10$)

File name	Size	gzip Ratio	P•gzip ΔRatio	P•gzip δRatio	P•gzip ΔSize	P•gzip Ex. gzip	bzip Ratio	P•bzip ΔRatio	P•bzip δRatio	P•bzip ΔSize	P•bzip Ex. bzip	ctw Ratio	P•ctw ΔRatio	P•ctw δRatio	P•ctw ΔSize	P•ctw Ex. ctw
	[Bytes]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]	[%]	[%]	[%]	[Bytes]	[Times]
alice29.txt	152089	35.79	-2.10	-5.86	-3192	2886	28.41	-0.33	-1.15	-498	666	25.94	-0.10	-0.40	-156	444
asyoulik.txt	125179	39.10	-1.02	-2.60	-1272	1554	31.61	-0.16	-0.50	-196	555	29.03	-0.00	-0.00	-1	222
cp.html	24603	32.52	-1.88	-5.79	-463	7881	30.99	-0.15	-0.49	-37	555	28.84	-0.10	-0.34	-24	222
fields.c	11150	28.19	-0.99	-3.50	-110	2109	27.26	-0.17	-0.63	-19	666	24.88	-0.17	-0.68	-19	555
grammar.lsp	3721	33.40	-0.32	-0.97	-12	555	34.48	-0.21	-0.62	-8	333	29.80	-0.08	-0.27	-3	222
lcet10.txt	426754	33.95	-2.41	-7.11	-10300	4440	25.24	-0.40	-1.59	-1715	1998	22.90	-0.07	-0.32	-309	1776
plrabn12.txt	481861	40.51	-2.79	-6.90	-13461	10656	30.21	-0.27	-0.88	-1285	888	27.32	-0.05	-0.18	-238	888
xargs.1	4227	41.57	-0.31	-0.74	-13	888	41.68	-0.09	-0.23	-4	333	37.02	0.02	0.06	1	111

る。それは通信路の性質の理解や定式化が進んでいることから可能になっている。本研究では、既存のデータ圧縮プログラムが符号理論の通信路に相当し、前処理が符号器に相当する。前処理と関連付けたデータ圧縮プログラムの性質の理解や定式化が本質的な理論的課題といえる。

本報告は2つの発表6）と7）をもとにまとめたものである。

謝辞

本研究についてご議論頂いた加治佐清光先生（当時、鹿児島工業高等専門学校情報工学科）に感謝いたします。

参考文献

1) J.Ziv and A.Lempel, “A Universal Algorithm for Sequential Data Compression,” IEEE Trans. Inform. Theory, vol.IT- 23, no.3, pp.337–343, May 1977.
2) M.Burrows and D.J.Wheeler, “A Block-Sorting Lossless Data Compression Algorithm,” SRC Research Report 124, Digital Systems Research Center, Palo Alto, May 1994.
3) F.M.J.Willems, Y.M.Shtarkov and T.J.Tjalkens, “The Context Tree Weighting Method: Basic Properties,” IEEE Trans. Inform. Theory, vol.41, no.3, pp.653–664, May 1995.
4) P.Gage, “A new algorithm for data compression,” The C Users Journal, R & D Publications, Inc. Lawrence, KS, USA, vol.12, no.2, pp.23–38, Feb. 1994.
5) 中村博文, 村島定行, “英数字列をひとまとまりとして扱うデータ圧縮について,” 信学論A, Vol. J80-A, N0.9, pp.1559–1563, Sept. 1997.
6) 中村博文, 加治佐清光, 渕田孝康, “データ圧縮の効率を高める前処理の可能性について,” 情報理論とその応用シンポジウム予稿集31巻, 論文番号6.4.4, pp 698-703, Oct. 2008.
7) 中村博文, 加治佐清光, 渕田孝康, “テキストデータの圧縮効率を高める前処理の可能性について,” 電子情報通信学会情報理論研究会技術報告IT2009巻123号, pp.337–342, Mar. 2010.

付録1　置き換え内容の表現に必要な情報量の変化

一般化して記号対の置き換えの影響を確認する。

以下では、圧縮対象の1記号当たりの、ある前提とした情報源モデルにおいての、平均情報量のことを単にエントロピーと呼びHと表す。置き換えると内容が変わるが、その場合でも、置き換え前の元の圧縮対象の1記号当たりで、表現に必要な平均情報量やエントロピーと呼ぶ。

まず、全体としては可逆であるため、置き換えによって情報が失われることはない。すなわち、置き換えた内容のエントロピーが小さくなることはない。

しかし、置き換えた内容は加工されているため、置き換えた内容を表すのに必要なビット数が、情報源モデルによっては、元の圧縮対象の1記号当たりで考えてエントロピーより多い記述量が必要になるのではないかという疑問が生じる。

加工の活用の例として、分野は異なるが、誤り訂正符号の利用ではインタリーブによってバースト的な誤りへの耐性を上げられる。しかし、データ圧縮では勝手な記号のシャッフルは、通常は効果的な圧縮の邪魔になる。

シャッフルよりは簡単な加工である置き換えについて見ていく。置き換えた結果の表現に必要な平均情報量を、但し、置き換え前の元の圧縮対象の1記号当たりでの平均情報量を、Gと表す。

前提とする情報源モデルによっては$G > H$ではないかという疑問が生じる。圧縮対象のそれぞれのファイルの真の情報源モデルは分かるものではないが、本報告では、基本的な定式化として、単純マルコフ情報源として捉えた場合について付録2で検証する。

結論から述べると、$G = H$である。

付録2　単純マルコフ情報源として置き換え内容を捉える場合

単純マルコフ情報源を通常と同じく状態遷移で捉えることにする。

使用記号の集合を$\mathcal{A}$、状態の集合を$\mathcal{S}$と表す。状態$x(\in \mathcal{S})$を取る確率をP_x、状態$x(\in \mathcal{S})$から記号$i(\in \mathcal{A})$で遷移する確率をp_{xi}のように表す。状態$x(\in \mathcal{S})$から記号$i(\in \mathcal{A})$で遷移した状態を$[x, i]$のように表すことがある。そこから更に遷移した状態はコンマで区切って書き足して表す。例えば、$[x, i]$から$j(\in \mathcal{A})$で遷移した状態を$[x, i, j]$のように表す。

状態$y(\in \mathcal{S})$から遷移するときのエントロピーをHyと表す。$Hy(y \in \mathcal{S})$は、

$$H_y \quad = \quad \sum_{j \in \mathcal{A}} -p_{yj} \log_2 p_{yj} \tag{1}$$

である。$\sum$の直後の項に単項演算子がある場合に、カッコで囲むことは省略する。$-p_{yj} \log_2 p_{yj}$はエント

ロピーの構成要素である。

単純マルコフ情報源のエントロピー H は

$$H = \sum_{y\in\mathcal{S}} P_y H_y \tag{2}$$

である。ここで、

$$\begin{aligned} P_y &= \sum_{x\in\mathcal{S}} \sum_{i\in\mathcal{A},\,[x,i]=y} P_x p_{xi} \\ &= \sum_{x\in\mathcal{S}} P_x \sum_{i\in\mathcal{A},\,[x,i]=y} p_{xi} \end{aligned} \tag{3}$$

であることを用いて、

$$\begin{aligned} H &= \sum_{y\in\mathcal{S}} \sum_{x\in\mathcal{S}} P_x \sum_{i\in\mathcal{A},\,[x,i]=y} p_{xi} H_y \\ &= \sum_{x\in\mathcal{S}} P_x \sum_{y\in\mathcal{S}} \sum_{i\in\mathcal{A},\,[x,i]=y} p_{xi} H_y \\ &= \sum_{x\in\mathcal{S}} P_x \sum_{i\in\mathcal{A}} p_{xi} H_{[x,i]} \\ &= \sum_{x\in\mathcal{S}} \sum_{i\in\mathcal{A}} P_x p_{xi} H_{[x,i]} \end{aligned} \tag{4}$$

と表せる。

$P_x p_{xi} H_{[x,i]}$ は遷移全体の中での、x から i で遷移する確率と、その遷移先のエントロピーをかけた値である。$P_x p_{xi} H_{[x,i]}$ はエントロピーの構成要素である

付録 2.1　単純マルコフ情報源についての置き換え

今、x が u、i が a、y が $[u,a]$ のときと対応する

$$\begin{array}{c} \downarrow\cdots\downarrow \\ (u)-a\rightarrow([u,a]) \\ ||\cdots| \\ j(\in\mathcal{A}) \\ \downarrow\downarrow\cdots\downarrow \end{array}$$

のような連なりに、すなわち遷移に、注目する。状態はカッコの中に記している（以下同様)。ここで、u からは $[u,a]$ への遷移のみ考えているが、$[u,a]$ へは他の状態からの遷移もあり得る。

$[u,a]$ からはすべての $j(\in\mathcal{A})$ についての遷移がある。このことは、上記において、簡潔に

$$\begin{array}{c} ||\cdots| \\ j(\in\mathcal{A}) \\ \downarrow\downarrow\cdots\downarrow \end{array}$$

のように表記している。

$[u,a]$ からの b での遷移を分けると

$$\begin{array}{c} \downarrow\cdots\downarrow \\ (u)-a\rightarrow([u,a])-b\rightarrow \\ ||\cdots| \\ j(\in\mathcal{A},\neq b) \\ \downarrow\downarrow\cdots\downarrow \end{array}$$

と表せる。単純マルコフ情報源の枠組み内で、$[u,a]$ への u からの遷移の分だけを、新たな状態の表記（ここでは v）を設けて分けることができる。すなわち、

$$\begin{array}{c} (u)-a\rightarrow(\quad v\quad)-b\rightarrow \\ ||\cdots| \\ j(\in\mathcal{A},\neq b) \\ \downarrow\downarrow\cdots\downarrow \end{array}$$

のように捉える。以下では、$j(\in\mathcal{A},\neq b)$ の部分、または $j(\in\mathcal{A},\neq a)$ の部分は一層簡単に「$\downarrow\downarrow$」と表す。

u から v を選ぶことと、v についての平均情報量は

$$\begin{aligned} h_{ua} &= P_u(-p_{ua}\log_2 p_{ua}) + P_u p_{ua} H_{[u,a]} \\ &= P_u(-p_{ua}\log_2 p_{ua}) + P_u p_{ua} H_v \qquad (5) \\ &= P_u(-p_{ua}\log_2 p_{ua}) \\ &\quad + P_u p_{ua} \sum_{j\in\mathcal{A}} -p_{vj}\log_2 p_{vj} \qquad (6) \end{aligned}$$

である。

単純マルコフ情報源において、任意のバイトコードによる隣接記号対（そのひと組を ab とする）をひとつの未使用バイトコード r に置き換える場合を考える。

ab を r に置き換えても v からの遷移として考えられるすべての場合について表現に必要な平均情報量がエントロピー H と同じであることを示すには、h_{ua} だけに関連した部分について置き換えても単純マルコフ情報源として見ているときの表現に必要な平均情報量（g_{ua} と表す）が変わらないことを示せばよい。すなわち、$g_{ua}=h_{ua}$ によって $G=H$ を言う。

以下に場合分けして示す。

以下では、確率（仮に、p である場合）について $1-p$ を $\bar{p}$ のように表す。

付録 2.2　a で始まる $a\neq b$、$u\neq v$、$[v,b]\neq v$ を含む場合

まず、最も簡単な、$a\neq b$、$u\neq v$、$[v,b]\neq v$ を含む場合について考える。

u から v を通過する部分だけに注目すると

$$\begin{array}{c} (u)-a\rightarrow(v)-b\rightarrow \\ \downarrow\downarrow \end{array}$$

のように連なっている。なお、ここで、v へは u からの遷移のみ考えるが、v からはすべての $j(\in\mathcal{A})$ の遷

移を考慮する。これは、

$$\begin{array}{l}(u)-a\rightarrow(v')-b\rightarrow\\ \quad\llcorner\!\!\!-\ a\rightarrow(v'')\\ \qquad\downarrow\downarrow\end{array}$$

のように表せる。更に、ここでの上側の経路の遷移では、a の後ろでは b が確率 1 で連接するので、記号対 ab を $\mathcal{A}$ の中でそれまでに使われていなかった r と書き変えて、(但し、r を $\mathcal{A}$ に加えたものを新しく $\mathcal{A}$ と表すのは証明が一旦済んでからとする)

$$\begin{array}{l}(u)\ \longrightarrow\ r\ \longrightarrow\\ \quad\llcorner\!\!\!-\ a\rightarrow(v'')\\ \qquad\downarrow\downarrow\end{array}$$

のように書き換えられる。(v'' からの $j(\in\mathcal{A},\neq b)$ での遷移確率は $\frac{p_{vj}}{\overline{p_{vb}}}$ である。)

この部分を表現するのに必要な平均情報量を g_{ua} とする。以下、$\log_2$ の直後の積にカッコは省略する。g_{ua} は、

$$\begin{aligned}g_{ua} &= P_u p_{ua}\times\ 0\\ &\quad +P_u(-p_{ua}p_{vb}\log_2 p_{ua}p_{vb})\\ &\quad +P_u(-p_{ua}\overline{p_{vb}}\log_2 p_{ua}\overline{p_{vb}})\\ &\quad +P_u p_{ua}\overline{p_{vb}}\sum_{j\in\mathcal{A},\,j\neq b}-\frac{p_{vj}}{\overline{p_{vb}}}\log_2\frac{p_{vj}}{\overline{p_{vb}}}\\ &= P_u(-p_{ua}\log_2 p_{ua})\\ &\quad +P_u p_{ua}(-p_{vb}\log_2 p_{vb}-\overline{p_{vb}}\log_2\overline{p_{vb}})\\ &\quad +P_u p_{ua}\left(\sum_{j\in\mathcal{A},\,j\neq b}-p_{vj}\log_2 p_{vj}\right.\\ &\qquad\left.+\left(\sum_{j\in\mathcal{A},\,j\neq b}p_{vj}\right)(\log_2\overline{p_{vb}})\right)\\ &= P_u(-p_{ua}\log_2 p_{ua})\\ &\quad +P_u p_{ua}\sum_{j\in\mathcal{A}}-p_{vj}\log_2 p_{vj}\\ &= P_u(-p_{ua}\log_2 p_{ua})+P_u p_{ua}H_v \qquad (7)\end{aligned}$$

となり、h_{ua} と等しい。よって、$G=H$ である。

付録 2.3　a で始まる $a\neq b$、$u\neq v$、$[v,b]=v$ を含む場合

次に、$a\neq b$、$u\neq v$、$[v,b]=v$ を含む場合について考える。

u から v を通過する部分だけに注目すると

$$\begin{array}{l}(u)-a\rightarrow(\overset{\downarrow}{v})-b\ \overset{\downarrow}{\llcorner}\\ \qquad\downarrow\downarrow\end{array}$$

のように連なっている。なおここで、v からはすべての $j(\in\mathcal{A})$ の遷移を考慮する。v へは u からの遷移と v からの遷移がある。ここで、エントロピーの構成要素の内、状態 v に関する量は

$$P_v H_v \ = \ P_v\sum_{j\in\mathcal{A}}-p_{vj}\log_2 p_{vj}$$

である。先述の連なりは、v から自身への遷移を 1 回分だけほどいて 2 つの状態 w' と w'' を用いて表し直すと

$$\begin{array}{l}(u)-a\rightarrow(w')-b\rightarrow(\overset{\downarrow}{w''})-b-\overset{\downarrow}{\llcorner}\\ \qquad\downarrow\downarrow\qquad\quad\downarrow\downarrow\end{array}$$

と表すことができる。$H_{w''}=H_v$ である。

ここで、エントロピーの構成要素の内、状態 w' と w'' に関する量は

$$p_v\overline{p_{vb}}\sum_{j\in\mathcal{A}}-p_{vj}\log_2 p_{vj}+P_v p_{vb}H_v \ = \ P_v H_v \quad (8)$$

となり、記号対の置き換えによって変化はない。

ここでの u から w'' までの連なりは前節と同じであるので、$[v,b]=v$ の場合でも、記号対 ab を r に書き変えても $H=G$ である。

付録 2.4　a で始まる $a\neq b$、$u=v$、$[v,b]\neq v$ を含む場合

a による自身への遷移があり得る場合である。

紙面の都合で証明を略すが、$H=G$ である。

付録 2.5　a で始まる $a\neq b$、$u=v$、$[v,b]=v$ を含む場合

a と b による自身への遷移があり得る場合である。

紙面の都合で証明を略すが、$H=G$ である。

付録 2.6　a で始まる $a=b$、$u\neq v$、$[v,b]\neq v$ を含む場合

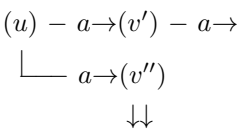
同じ記号での記号対の場合であり、更に自身への遷移がない場合である。

なお、ここでの議論には、$[v,a]=u$ の場合も含まれる。

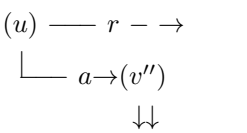
u から v を通過する部分だけに注目すると

$$\begin{array}{l}(u)-a\rightarrow(v)-a\rightarrow\\ \qquad\downarrow\downarrow\end{array}$$

のように連なっている。なおここで、v へは u からの遷移のみ考えるが、v からはすべての $j(\in\mathcal{A})$ の遷移を考慮する。これは、



のように表せる。更に、これの上側の経路の遷移では、a の後ろで a が確率 1 で連接するため、記号対 aa を $\mathcal{A}$ の中でそれまでに使われていなかった r と書き変えて、(但し、r を $\mathcal{A}$ に加えたものを新しく $\mathcal{A}$ と表すのは証明が一旦済んでからとする)



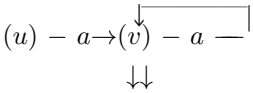
のように書き換えられる。(v'' からの $j(\in \mathcal{A}, \neq a)$ での遷移確率は $\frac{p_{vj}}{p_{va}}$ である。)

ここから先は $a \neq b$ の議論と同様にして、aa を r に置き変えても表現に必要な平均情報量が変化しないことが言える。

付録 2.7　a で始まる $a = b$、$u \neq v$、$[v, b] = v$ を含む場合

同じ記号での記号対の場合で、更に 2 番目で自身への遷移があり得る場合である。

u から v を通過する部分だけに注目すると



のように連なっている。なおここで、v からはすべての $j(\in \mathcal{A})$ の遷移を考慮する。v へは u からの遷移と v からの遷移がある。

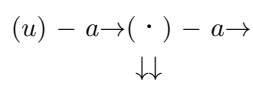
P_v について、$P_v = P_u p_{ua} + P_v p_{va}$ という関係が成り立ち、解くと

$$P_v = \frac{P_u p_{ua}}{\overline{p_{va}}} \tag{9}$$

である。

よって、エントロピーの構成要素の内、u から v を選ぶことと、v に関する量は、$P_u(-p_{ua}\log_2 p_{ua}) + P_v H_v$ である。

上記の連なりは、まず $a \neq b$ の場合に行ったように、v から v への遷移を 1 回分だけほどいて 2 つの状態を用いて表し直すと

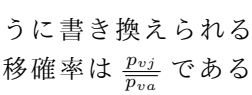


と表すことができる。名前を付けない状態については、中点を記している。ここでは、$[u, a]$ と $[u, a, a]$ がそれに当たる。$[u, a]$ を取る確率 $P_u p_{ua}$ は P_v を用いて表すと式 (9) より $P_v\overline{p_{va}}$ であり、$[u, a, a]$ を取る確率は P_v から $[u, a]$ を取る確率を引いて $P_v p_{va}$ である。また、$H_{[u,a]} = H_{[u,a,a]} = H_v$ である。

上記の連なりの、後ろの a での繰り返しの部分も考慮する必要がある。一旦、a での繰り返しの部分を、無限にほどいたと考えると、

と表すことができる。u の後ろの第 k 番目の状態が取る確率は P_v を用いて表すと、$P_v p_{va}^{k-1}\overline{p_{va}}$ である。

これの 2 番目以降の偶数番目と、3 番目以降の奇数番目とをまとめると、



と表すことができる。

これの u の後ろの状態を取る確率は $P_v\overline{p_{va}}$ である。その後ろは、先の無限の状態の 2 番目の状態を取る確率 $P_v p_{va}\overline{p_{va}}$ に、$1 + x + x^2 + \cdots = \frac{1}{1-x}$ なる関係を用いて偶数番目をすべて足し合わせて、

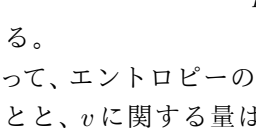
$$P_v p_{va}\overline{p_{va}}\frac{1}{1-p_{va}^2} \quad = \quad \frac{P_v p_{va}}{1+p_{va}}$$

である。更にその後ろは、先の無限の状態の 3 番目の状態を取る確率をもとにして、

$$P_v p_{va}^2\overline{p_{va}}\frac{1}{1-p_{va}^2} \quad = \quad \frac{P_v p_{va}^2}{1+p_{va}}$$

である。(ここでは確認として、u の後ろのこれら 3 つの状態のどれかを取る確率を求めると、3 者を足して P_v である。記号対の置き換えが他の状態の確率に影響を与えていないと言える。)

aa を r に書き変えると



のように書き換えられる。v'、v''、v''' はここで新たに定義した名前である。(v'、v''、v''' からの $j(\in \mathcal{A}, \neq a)$ での遷移確率は $\frac{p_{vj}}{p_{va}}$ である。)

v'、v''、v''' を取る確率はそれぞれ $P_v\overline{p_{va}}^2$、$\frac{P_v p_{va}}{1+p_{va}}$、$\frac{P_v p_{va}^2\overline{p_{va}}}{1+p_{va}}$ である。

エントロピーの構成要素の内、u から上記の後ろの部分を選ぶことと、後ろの部分の状態に関する量を簡単に h とおく。h は

$$\begin{aligned} h \quad = \quad & P_u p_{ua} \times\ 0 \\ & + P_u(-p_{ua}p_{va}\log_2 p_{ua}p_{va}) \end{aligned}$$

$$
\begin{aligned}
&+P_u(-p_{ua}\overline{p_{va}}\log_2 p_{ua}\overline{p_{va}})\\
&+P_v\overline{p_{va}}^2\sum_{j\in\mathcal{A},\,j\neq a}-\frac{p_{vj}}{\overline{p_{va}}}\log_2\frac{p_{vj}}{\overline{p_{va}}}\\
&+\frac{P_vp_{va}}{1+p_{va}}\Bigg(-p_{va}^2\log_2 p_{va}^2-p_{va}\overline{p_{va}}\log_2 p_{va}\overline{p_{va}}\\
&\qquad\qquad-\sum_{j\in\mathcal{A},\,j\neq a}-p_{vj}\log_2 p_{vj}\Bigg)\\
&+\frac{P_vp_{va}^2\overline{p_{va}}}{1+p_{va}}\sum_{j\in\mathcal{A},\,j\neq a}-\frac{p_{vj}}{\overline{p_{va}}}\log_2\frac{p_{vj}}{\overline{p_{va}}}\\
=\;& P_u(-p_{ua}p_{va}\log_2 p_{ua}-p_{ua}p_{va}\log_2 p_{va}\\
&\quad-p_{ua}\overline{p_{va}}\log_2 p_{ua}-p_{ua}\overline{p_{va}}\log_2\overline{p_{va}})\\
&+P_v\overline{p_{va}}\sum_{j\in\mathcal{A},\,j\neq a}-p_{vj}\log_2 p_{vj}\\
&-P_v\overline{p_{va}}\left(\sum_{j\in\mathcal{A},\,j\neq a}-p_{vj}\right)\log_2\overline{p_{va}}\\
&+\frac{P_vp_{va}}{1+p_{va}}\Bigg(-p_{va}^2\log_2 p_{va}-p_{va}^2\log_2 p_{va}\\
&\qquad-p_{va}\overline{p_{va}}\log_2 p_{va}-p_{va}\overline{p_{va}}\log_2\overline{p_{va}}\\
&\qquad\qquad-\sum_{j\in\mathcal{A},\,j\neq a}-p_{vj}\log_2 p_{vj}\Bigg)\\
&+\frac{P_vp_{va}^2}{1+p_{va}}\Bigg(\sum_{j\in\mathcal{A},\,j\neq a}-p_{vj}\log_2 p_{vj}\\
&\qquad\qquad-\sum_{j\in\mathcal{A},\,j\neq a}-p_{vj}\log_2\overline{p_{va}}\Bigg)\\
=\;& P_u(-p_{ua}\log_2 p_{ua})\\
&+P_v\left(\overline{p_{va}}H_v+\frac{p_{va}^2}{1+p_{va}}H_v+\frac{p_{va}}{1+p_{va}}H_v\right)\\
=\;& P_u(-p_{ua}\log_2 p_{ua})+P_vH_v \qquad (10)
\end{aligned}
$$

となり、よって、aa を r に置き換える前と比べて表現に必要な平均情報量は変わらない。

付録 2.8　a で始まる $u=v$ を含む場合

同じ記号での記号対の場合で、更に最初に自身への遷移があり得る場合である。

残っていた 2 つのケースについて、紙面の都合で証明を略すが、$[v,b]\neq v$ でも、$[v,b]=v$ でも、$H=G$ である。

付録 3　m 重マルコフ情報源として置き換え内容を捉える場合

念のために明記しておくと、$m=1$ の場合は単純マルコフ情報源でありり付録 1 から付録 2 で述べている。この節で言及するのは $m\geq 2$ の場合である。

ab を r に置き換えた内容の各場所で、過去の m 記号から次のひとつの確率を知ることができるため、証明自体は可能である。しかし、場合分けが多くなりすぎてすべてを列記できないため、証明における場合分けの方針だけ述べておく。

m重マルコフ情報源として捉える場合、先の記号について、a と r（すなわち、ab）は区別が必要である。置き換え前に

$$\underbrace{\cdots\cdots\cdots}_{m\text{ 記号}}a$$

であったものが、置き換えると

$$\underbrace{\cdots\cdots\cdots}_{m\text{ 記号}}a(\neq\underbrace{\cdots\cdots\cdots}_{m\text{ 記号}}ab)$$

及び

$$\underbrace{\cdots\cdots\cdots}_{m\text{ 記号}}r$$

になる。

これについて、r は ab と続く確率で遷移し、a は ab と続く確率を除いた確率で遷移する。

過去の m 記号に ab が含まれていた場合の置き換え後の m 記号の状態からの繊維というのは、実質的には元々の m 個より多い記号での状態から遷移することになる。また、それぞれの状態からの遷移確率は重複や欠けがなく量的に知ることができる。これは、例えば、過去の m 記号に ab が 1 対だけ含まれる場合なら、置き換え前に

$$\underbrace{\cdots a\,b\cdots}_{m\text{ 記号}}a$$

であったものが、置き換えると

$$\underbrace{\cdots r\cdots}_{m\text{ 記号}}a\;(\neq\underbrace{\cdots r\cdots}_{m\text{ 記号}}ab)$$

及び

$$\underbrace{\cdots r\cdots}_{m\text{ 記号}}r$$

になる。

過去の m 記号に含まれる ab の対（すなわち、r）の数を 2 つ、3 つ、$\cdots$ と最大 m まで増やして網羅して場合分けすればよい。